

PhD-FSTM-2020-43 The Faculty of Sciences, Technology and Medicine

DISSERTATION

Defence held on 24/09/2020 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

by

Dániel FEHÉR

Born on 5 March 1993 in Miskolc, (Hungary)

DATA ANALYTICS AND CONSENSUS MECHANISMS IN BLOCKCHAINS

Dissertation defence committee

Dr Alex Biryukov, dissertation supervisor *Professor, Université du Luxembourg*

Dr Paulo Esteves-Veríssimo, Chairman *Professor, Université du Luxembourg*

Dr Volker Müller, Vice Chairman Associate Professor, Université du Luxembourg

Dr Rainer Böhme Professor, Universität Innsbruck

Dr Ghassan Karame Head of Security Research, NEC Laboratories Europe

Abstract

Blockchains, and especially Bitcoin have soared in popularity since their inceptions. This thesis furthers our knowledge of blockchains and their uses.

First, we analyze transaction linkability in the privacy preserving cryptocurrency Zcash, based on the currency minting transactions (mining). Using predictable usage patterns and clustering heuristics on mining transactions, an attacker can link to publicly visible addresses in over 84% of the privacy preserving transactions

Then, we further analyze privacy issues for the privacy-oriented cryptocurrency Zcash. We study privacy preserving transactions and show ways to finger-print user transactions, including active attacks. We introduce two new attacks, which we call the *Danaan-gift* attack and the *Dust* attack.

Then, we investigate the generic landscape and hierarchy of miners as exemplified by Ethereum and Zcash. Both chains used application-specific integrated circuit (ASIC) resistant proofs-of-work which favor GPU mining in order to keep mining decentralized. This, however, has changed with the introduction of ASIC miners for these chains. This transition allows us to develop methods that might detect hidden ASIC mining in a chain (if it exists), and to study how the introduction of ASICs affects the decentralization of mining power. Finally, we describe how an attacker might use public blockchain information to invalidate miners' privacy, deducing the mining hardware of individual miners and their mining rewards.

Then, we analyze the behavior of cryptocurrency exchanges on the Bitcoin blockchain, and compare the results to the exchange volumes reported by the same exchanges. We show, that in multiple cases these two values are close to each other, which confirms the integrity of their reported volumes. Finally, we present a heuristic to try to classify large clusters of addresses in the blockchain, and whether these clusters are controlled by an exchange.

Finally, we describe how to couple reputation systems with distributed consensus protocols to provide a scalable permissionless consensus protocol with a low barrier of entry, while still providing strong resistance against Sybil attacks for large peer-to-peer networks of untrusted validators. We introduce the reputation module ReCon, which can be laid on top of various consensus protocols such as PBFT or HoneyBadger. The protocol takes external reputation ranking as input and then ranks nodes based on the outcomes of consensus rounds run by a small committee, and adaptively selects the committee based on the current reputation.

Acknowledgements

This dissertation would not be possible without the support and help from many people. First of all, I would like to thank my supervisor, Prof. Alex Biryukov, for leading my research and providing me with the freedom to choose many topics. I would like to thank my thesis supervision committee members, Prof. Paulo Esteves Verissimo, and Prof. Volker Müller, for following my research and giving valuable advice. I am grateful to Prof. Rainer Böhme and to Dr. Ghassan Karame for agreeing to serve on my defense as jury members.

I would like to thank the University of Luxembourg and the Interdisciplinary Centre for Security, Reliability and Trust for providing me an excellent research environment.

I am thankful to all my colleagues at CryptoLUX and at the University for the happy years and the countless conversations we had during our coffee breaks.

My greatest gratitude is to my family and friends who have always supported me in every decision I took in my life.

Contents

A	bstra	net	iii
A	ckno	wledgements	V
Li	ist of	Figures	xi
Li	ist of	Tables	xiii
Li	ist of	Abbreviations	xv
1	Inti	roduction	1
	1.1	Currencies	1
		1.1.1 Main Challenges of Digital Currencies	2
		1.1.2 Digital Currencies	4
	1.2	Bitcoin	4
		1.2.1 Transaction Structure	4
		1.2.2 Blocks	6
		1.2.3 Proof-of-Work Protocol	7
		1.2.4 Mining Hardware	9
		1.2.5 Mining Pools	10
		1.2.6 Network Communication	11
		1.2.7 Security of Bitcoin	12
		1.2.8 Advantages and Disadvantages	13
		1.2.9 Main Actors in Bitcoin	14
	1.3	Scalability of Cryptocurrencies	15
	1.4	Privacy in Cryptocurrencies	16
		1.4.1 Privacy-oriented Cryptocurrencies	17
	1.5	Contributions	19
Ι	Da	ata Analytics in Blockchains	21
2	Dea	anonymizing Miners in Zcash	23
	2.1	Zcash	23
		2.1.1 Notation	26
	2.2	Analytics tool for Zcash	27
	2.3	Related Work	28
	2.4	Deanonymizing The Miners	29
		2.4.1 Pattern T Mining Pools	29
		2.4.2 Pattern Z Mining Pools	30

		2.4.3 Results of the Heuristics
		2.4.4 Accuracy of the Heuristics
		2.4.5 Comparison of results to previous work
		2.4.6 Inflation of results
	2.5	Summary and Conclusions
3	Fur	ther Transaction Linking in Zcash 37
	3.1	Linking with Transaction Values
	0.1	3.1.1 Direct Value Linking Including Transaction Fees 38
		3.1.2 Subset sum
		3.1.3 Fingerprinted Values
		3.1.4 Further Results
	3.2	The Model for the Probability of Fingerprint Survival
		3.2.1 Experimental results
		3.2.2 Notation
		3.2.3 Sapling Transactions Dataset
	3.3	Danaan-Gift Attack (Malicious Value Fingerprinting) 48
	3.4	Dust Attack
		3.4.1 Official Linux Command-line Zcash Wallet 50
		3.4.2 GUI-based Sapling-supporting Wallets 50
		3.4.3 Combining Danaan and Dust Attacks 51
	3.5	Usage of zk-SNARKs
		3.5.1 Interaction Between Sapling and Sprout Transactions 51
	3.6	Summary and Conclusions
4	Duit	vacy of Miners in Zcash and Ethereum 53
4	4.1	Terminology
	4.1	Background and Related Work
	4.2	Mining Landscape
	4.0	4.3.1 Ethereum
		4.3.2 Zcash
		4.3.3 GPU Mining
		4.3.4 GPU vs ASIC mining
	4.4	Detecting ASIC miners
	1.1	4.4.1 Fraction of large miners in the mining power
		4.4.2 Mining Software Developer Fees 61
		4.4.3 Public Introduction of ASICs
	4.5	Mining Centralization
	4.6	Privacy of Miners
		4.6.1 Linkability of Mining rewards
		4.6.2 Countermeasures
	4.7	Summary and Conclusions
5	F.et	imating Exchange Traffic 69
J	5.1	Evaluating Identified Clusters
	5.1	Classifying Large Clusters
	5.3	Summary and Conclusions
	0.0	

II	\mathbf{C}	onsen	nsus Protocols in Blockchains	7 9
6	ReC	Con		81
		6.0.1	Related Work	83
	6.1	Existin	ng Consensus Protocols	83
		6.1.1	Proof-of-Work	83
		6.1.2	Proof-of-Stake	85
		6.1.3	Byzantine Agreement	85
		6.1.4	Hybrid Protocols	86
	6.2	Prelim	ninaries of Our Protocol	87
		6.2.1	Generic	87
		6.2.2	Assumptions	88
		6.2.3	Nodes	89
	6.3	Reput	ation module	89
		6.3.1	External Reputation	90
		6.3.2	Committee selection	91
		6.3.3	Rewards and penalties	92
		6.3.4	Probability of a forgery	93
		6.3.5	Types of Blocks	95
		6.3.6	Source of randomness	96
		6.3.7	Fairness	96
		6.3.8	Dealing with forks	97
		6.3.9	Convergence	97
		6.3.10	Pseudocode	98
	6.4	Simula	ation Results	98
		6.4.1	External reputation: discrete (no information)	99
		6.4.2	External reputation with normal distribution	100
		6.4.3	External reputation with exponential distribution	101
	6.5	Attack	ss and their mitigation	101
		6.5.1	Botnet takeover	101
		6.5.2	Sybil attack: saturation	102
		6.5.3	Sybil attack: lie and wait strategy	102
		6.5.4	Attacks on randomness	102
		6.5.5	Honest majority	103
		6.5.6	Detection based on the success rate	103
	6.6	Summ	ary and Conclusions	103
7	Sun	nmary	and Conclusions	105
	7.1	Future	e Works	107

List of Figures

1.1	Example for the transaction structure of Bitcoin	5
2.1	Zcash Sapling Turnstile	26
3.1	Average output distance until a fingerprint stays unique	42
4.1	Histogram of hash rates in Ethereum for the GTX 1050 GPU .	56
4.2	Histogram of hash rates in Ethereum for the GTX 1060 GPU .	56
4.3	Histogram of hash rates in Ethereum for the GTX 1070 GPU .	56
4.4	Histogram of hash rate in Ethereum for the RX 580 GPU	56
4.5	Histogram of hash rates in Zeash for the GTX 1050 GPU	57
4.6	Histogram of hash rates in Zeash for the GTX 1060 GPU	57
4.7	Histogram of hash rates in Zcash for the GTX 1070 GPU	57
4.8	Histogram of hash rates in Zcash for the GTX 1080 GPU	57
4.9	Overall Equihash mining power over time	59
4.10	1	60
	Profitability of mining in USD	61
	Estimated portion of large miners	62
	Lower bound of GPU mining power	63
	Change in the projected mining power	64
	Change of power distribution in mining pools	64
	Number of daily recorded miners	65
	Histogram of recorded hash rates in Zcash	66
4.18	Example of tracking a single miner	67
5.1	Poloniex's reported and chain traffic for the entire 701 day period.	71
5.2	Bitstamp's reported and chain traffic for the best matching 61 day period	75
5.3	Kraken's reported and chain traffic for the best matching 61 day	
	period	76
5.4	LocalBitcoins' reported and chain traffic for the best matching	
	61 day period	76
6.1	The reputation curve after 10,000 rounds	91
6.2	Exponential Distribution	96
6.3	Exponential Power Distribution	96

List of Tables

1.1	The structure of a Bitcoin block	6
2.1 2.2 2.3	Transaction types in Zcash	25 33 34
3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8	Number of unique fingerprints	40 42 43 47 47 48 48 51
4.1 4.2 4.3	Number of times a card has appeared in Ethereum	57 58 58
5.15.25.35.4	The four highest correlation values between the reported and observed on the chain data over 701 days	72 72 72 77
6.1 6.2 6.3	The protocol parameters	90 94 94
6.4 6.5	No external reputation, exponential selection rule: success rates	100
6.6	after 10,000 rounds	100 100
6.7 6.8	External normal distribution, selection with triangular distribution. External exponentially distributed reputation, exponential selec-	

6.9	External	ex	po	ne	$_{ m nt}$	ial	C	lis	tr	ib	ut	ic	on.	, 8	ele	ect	io	n	Wi	th	tı	ria	n	gι	ıla	ar	d:	is-		
	tribution																												10)]

List of Abbreviations

PoW Proof of Work PoS Proof of Stake

BA Byzantine AgreementBFT Byzantine Fault Tolerance

P2P Peer-to-Peer

CPU Central Processing Unit GPU Graphics Processing Unit

FPGA Field Programmable Gate Array

ASIC Application-Specific Integrated Circuit

UTXO Unspent Transaction Output

ZK Zero-Knowledge

zk-SNARK Zero-Knowledge Succinct Arguments of Knowledge

VDF Verifiable Delay Function

Chapter 1

Introduction

What is a blockchain? What is Bitcoin? Will it change how we think about money? Is it as private as cash? Such questions are asked these days about cryptocurrencies. Currently, there is no scientific consensus on most of the overarching questions, for example, will everybody use some form of blockchain in the near or far future? This doctoral thesis will try to answer a few of the questions that arise with our current cryptocurrencies and techniques.

Specifically, this thesis will explain and expand on the following subjects that are all focused on blockchains. First, we will provide an in-depth introduction to Bitcoin's workings as the original ancestor of all later blockchain technology. Then, we will shift our focus to a cryptocurrency called Zcash, which focuses on its users' privacy. We will describe how it tries to achieve its privacy guarantees. We will show novel techniques that weaken the built-in privacy of the blockchain, reducing its effectiveness to a level similar of Bitcoin. We will expand our privacy focused research with data analytical observations on Zcash and other cryptocurrencies.

Finally, we introduce consensus protocols and describe a novel approach to reach agreement among the participants of a blockchain protocol. However, in order to present this novel research, the history and functioning of Bitcoin will be elaborated.

1.1 Currencies

In the early days of civilization, humanity used barter for exchanging products between different families or tribes. As civilizations evolved, the idea of a medium of exchange has emerged for these trades. The goal of this intermediary step was to reduce the restriction of available goods, as there were cases where the two trading parties did not need any of the other's products. These later evolved into currencies, which had some essential features of their own. A currency or a coin became a store of value, and were usually based on some precious metals. Later on, with the devaluation and the weight of coins, using them became less practical. In turn, traders started to use written trade guarantees during exchanges. These guarantees were exchangeable for actual products in larger cities and banks, and were one of the ancestors of banknotes. Later, in Europe traders have started operating with private bank provided legal tenders of coin deposits, which were another originators of banknotes. Banknote based currencies have also developed in other places, most notably in China in the

11th century, on similar principles. These notes over time have evolved into the ones we use today. Until recently, in most countries the banknotes were theoretically convertible to gold or silver in an authorized bank, as they were partially backed by them.

The last step for our current monetary system came in 1971 when United States President Richard Nixon has canceled banknotes' direct conversion to gold, introducing the United States to a fiat currency model. In fiat currencies, the currency itself does not hold any value. Instead, its value is backed directly by a government, but without any tangible value. The United States was one of the last major economies to switch to the fiat model, and currently, almost all of the world's monetary systems are fiat currencies.

Parallel to these changes, we also saw the rise of electronic payments. The mainstream spread of credit cards first advanced this change. In some countries like Sweden [Rik] in 2018, only 13% used cash in their last physical purchase, and Sweden is planning to go completely cashless by 2023. Even though electronic payments provide plenty of advantages over cash, mainly in their ease of use, there are some disadvantages. The most argued problem is that all transactions go through banks or credit card companies, and payments lose their privacy compared to cash. Furthermore, the issuer of a card or electronic payment service provider can block access at any time or deny certain transactions, creating possibilities for censorship against their users. Thus the users have to blindly trust these providers, while in cash-based systems, such censorship is not possible.

The evolution of the internet, computer science and the previously mentioned issues have lead to some researchers exploring the idea of digital currencies that are independent from fiat-currencies. Here are some of the main challenges for a digital currency.

1.1.1 Main Challenges of Digital Currencies

There are multiple technical issues that digital currencies need to solve before they become a usable currency.

1.1.1.1 Consensus

Consensus is one of the leading research areas in distributed systems. It refers to the fact that all honest - well-behaved - participants of a protocol agree with the result of some operation. For example, in a monetary system, among other issues, whether all honest participants agree on the correct transactions, how much currency each user has, or whether the users can spend their currency.

The complete problem is the following. Let us assume some nodes communicate over an untrusted channel - an outside observer can listen, modify, halt or delete messages on the channel - and try to reach some form of agreement. Can the honest nodes achieve agreement if some of the nodes can be malicious? The problem has been researched actively since the early 80s when researchers introduced the so-called Byzantine Generals Problem [LSP82] and related challenges. The original Byzantine General problem is the following. Three Byzantine armies are preparing for battle against a common foe. They

1.1. Currencies 3

can only win the battle if all of them attack at the same time. On the other hand, the generals of these armies can not meet in person as they are in enemy territory, and can only communicate through messengers between the armies. How can the generals agree on the time of the attack and make sure that the other generals have confirmed the time of the attack?

Furthermore, any of the generals can act maliciously by sending confirmation to one while sending disagreement to another general, among other acts. Finally, the communication channel, i.e., the messenger can not be trusted either, as it can be captured on the way or changed to a different messenger with a different message in hand.

With these restrictions the question is what assumptions one has to make that agreement is possible and to verify that agreement, i.e., time of the attack. It is evident that if more than 50% of the generals are malicious, then the honest nodes can not achieve agreement. [LSP82] shows that if there are N nodes, then the number of malicious nodes can not exceed $\lfloor \frac{N-1}{3} \rfloor$.

There are existing solutions to this consensus problem ([CL99, AMQ13]) with different assumptions, but most of them are only practical for a handful of participants. We will refer to these protocols as classical Byzantine consensus. These protocols are usually voting-based, where every node has an equal vote on proposed operations, whether they approve those operations. However, a real digital currency would be open and allow anybody to join the protocol, which would lead to thousands and thousands of participants, something that classical Byzantine consensus protocols can not process.

1.1.1.2 Double-Spending

Another important problem in a digital currency is the issue of double-spending. Double-spending is the act of spending the same currency twice. In the real world, it could mean spending the same banknote in two different places at the same time. In a digital system a direct example would be the following. Let us assume there are ten coins in an account. A user tries to buy two separate items, each worth ten coins. If the user can get both items by starting both transactions simultaneously, while the monetary authority did not realize it is spending the same coins twice, it can double-spend its coins and get both items.

1.1.1.3 Sybil-Attack

A Sybil attack [Dou02] is when a malicious actor joins a network with a large number of nodes or users, trying to swarm and take control of a system. They are most damaging in reputation or rating based platforms, where an attacker can create thousands of fake users to boost the reputation of an entity or the rating of an item. An attacker can use Sybil-attacks in consensus protocols as well. Open public voting-based agreement protocols - like classical Byzantine consensus protocols - do not have any defense against these attacks, which is one of their most significant issues in their deployment in a broader, more open ecosystem.

1.1.2 Digital Currencies

The idea of digital currencies has been under consideration since 1983 when David Chaum introduced the term digital cash in his research paper [Cha82]. Later, in 1990 he founded his own company called DigiCash using the same underlying technology for digital payments. The company filed for bankruptcy in 1998.

In 1997 Adam Back introduced HashCash [Bac02]. It further improved on the original idea by Dwork and Naor [DN92], that required some non-precomputable challenge from a sender to combat junk mail. It was the first protocol to use hash functions as a proof-of-work protocol, which provided both the moderately hard function to compute from the prover and very fast verification from the verifier.

In 1998 Wei Dai proposed B-Money [Dai98], which was the first digital currency that used a proof-of-work protocol as a money minting procedure. It also used asymmetric cryptography, namely the public keys of users as their identifiers, and every user would keep track of the value balances of the other users.

Then in 2005, Nick Szabo proposed Bit Gold [Sza05], which was the direct predecessor to Bitcoin. It utilized the proof-of-work protocol but only as a safety mechanism. For consensus between users, it relied on voting protocols, which were susceptible to Sybil-attacks.

These developments were followed by Satoshi Nakamoto launching Bitcoin [Nak09] in 2009.

1.2 Bitcoin

Bitcoin [Nak09] was first introduced to the world in 2009, under the pseudonym Satoshi Nakamoto. To this day, there is no clear candidate for the real author behind the work. Initially, Bitcoin was deemed an interesting, novel digital currency, but an infeasible approach to solving the problem of consensus among a large number of nodes. The main issue with it was its probabilistic nature, which many considered impractical in a real life setting. However, Bitcoin has shown that such a probabilistic consensus is in fact viable and much more effective than deterministic algorithms. As such, the major innovation it provided was the Proof-of-Work (PoW) consensus protocol. In order to present the PoW protocol, we have to present how transactions function in Bitcoin.

1.2.1 Transaction Structure

Bitcoin does not use a standard account and balance based approach. Instead, it utilizes a so-called unspent transaction output (UTXO) format. The idea behind the format is the following. There are no accounts or balances, only transaction outputs. These transaction outputs are tied to a public key in an RSA type public-key infrastructure [DH76, RSA78, JMV01].

We are not going to describe public key cryptography in detail, but we will introduce the main principles. In a public-key encryption scheme, there are two

1.2. Bitcoin 5

types of keys, the public (K_{pub}) and private or secret (K_{sec}) key, respectively, and they represent a specific entity. The public key is known by everybody in the network, while this specific entity is the only one to know the private key. Then a public key encryption scheme is the following:

$$D(E(m, K_{pub}), K_{sec}) = D(E(m, K_{sec}), K_{pub}) = m$$

where E is the encryption, D is the decryption function and m is the message
$$(1.1)$$

We refer to these types of encryption protocols as asymmetric cryptographic ciphers, as they use a different key for encryption and decryption. Then, if we have such a public key signature scheme, one can introduce digital signatures as a cryptographic protocol, which ties a message m to an entity X, meaning the message was signed by somebody who has control of X's private key. The signature can be verified using the message m and the public key of X.

In Bitcoin, transaction outputs are tied to such public keys, and when we refer to an address, we refer to the public key of the output. There can be multiple outputs tied to the same public key. Every such output has a value as well, which is the coin value of it in the blockchain. If a user wants to use and spend the value in an output, it has to digitally sign the output with its private key. Then it has to create a transaction where the inputs of the transaction become these previous outputs that the user is trying to spend. Then the transaction consumes the chosen old outputs and their value, and in the outputs section of the transaction it can create new outputs tied to any known public key up to the sum of values of the inputs. The miner of the block can claim the remaining unused input as the transaction fee. Then these new outputs of the transaction become part of the UTXO set, and the input outputs of the transaction become flagged as spent (Figure 1.1).

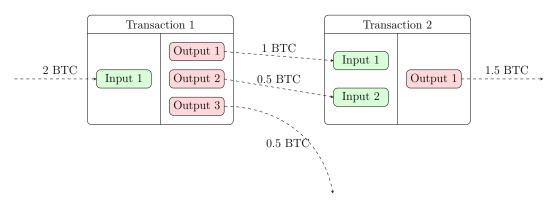


FIGURE 1.1: An example for the transaction structure of Bitcoin. See that the output of a transaction is then the input of a later transaction, chaining all transactions together.

Furthermore, when a user is creating a transaction, there can be multiple public keys in the input part as well. This also means that the creator of the transaction knows the private keys to all the public keys in the input field, or in other words, the creator has full control over all the input addresses. In Bitcoin these transactions are bundled into blocks.

1.2.2 Blocks

Let us describe in detail the attributes and features of a block in Bitcoin. Blocks are the main building elements of Bitcoin, as the confirmed transactions are spread over the network using these blocks. The Bitcoin protocol enforces a 1MB maximum block size in order to limit the network traffic between users. A block's structure can be seen in Table 1.1.

Field	Description
Magic Number	A fixed value, that is used in computer science
	to describe the type of data following this magic
	number. A Bitcoin block's magic number is
	"0xD9B4BEF9".
Blocksize	The exact size of the entire block in number of
	bytes.
Blockheader	The blockheader of a Bitcoin block consists of
	six further fields. Described in the following sec-
	tion 1.2.2.1.
Transaction Counter	The exact number of transactions in the block.
Transactions	The actual list of transactions, exactly as many
	as specified in the "Transaction Counter" field.

Table 1.1: The structure of a Bitcoin block.

1.2.2.1 Blockheader

The blockheader of Bitcoin is a crucial part of the blockheain, as the values in the blockheader are the input to the Proof-of-Work protocol. The functionality of the PoW protocol is described in Section 1.2.3. The Bitcoin blockheader consists of 6 different fields. Let us describe these in detail.

Version The version field describes the software version of Bitcoin used to create the block.

hashPrevBlock The 256 bit long hash of the previous blockheader, where the hash is an immutable short unique identifier and if anything would change in the previous block, the hash of it would change as well.

hashMerkleRoot Similar to the "hashPrevBlock" field, the "hashMerkleRoot" field is a 256 bit short unique immutable identifier of all the transactions in the current block, which means, that if anything changes in any of the transactions in the block, or new transactions are added to the block, this value would change as well.

1.2. Bitcoin 7

Time The block's timestamp of creation in number of seconds since 1970-01-01 00:00 UTC. As it is an input to the PoW protocol, it is updated every few seconds. These timestamps are more of a direction than a precise time of creation due to the feature of the PoW protocol (Section 1.2.3). There are cases in Bitcoin where the timestamp of a following block is actually sooner than the previous block.

Bits The difficulty of the PoW protocol in a compact format. It is adjusted every 2016 blocks.

Nonce A 32 bit value that is randomly chosen by the solver of the PoW protocol. The nonce is the most changed input of the PoW protocol, as it is the only value that provides complete liberty, and its purpose is to be changed a lot in order to solve the PoW protocol.

1.2.2.2 SegWit

In the early days of Bitcoin the block sizes never approached the 1MB limit, but as Bitcoin grew in popularity and usage, the blocks started to get larger and larger as well. By 2015, the majority of blocks used all the available space for transactions, which lead to developers looking for workarounds to increase the number of transactions in a block by different measures. The final solution was called SegWit [WJT20] (standing for segregated witnesses), and the idea was to decouple the digital signatures from the transactions, and storing the signatures separately, creating more space for transactions in the blocks. The protocol upgrade launched in the summer of 2017, and is still used to this day. The 1MB block size limit is still a part of the blockchain.

1.2.3 Proof-of-Work Protocol

The PoW protocol had a huge impact on consensus research, showing that practical consensus is achievable in an open setting even with thousands of participants. In a nutshell, the protocol is a cryptographic puzzle for which no algorithm better than brute force is known. Let us explain it in more detail. First, we need to learn what is a hash function.

A hash function is a function that takes as input any sized data and outputs fixed-length data. However, a hash function on its own is not enough, and we have to introduce cryptographic hash functions. Informally, a cryptographic hash function is a hash function that is deterministic, practically hard to invert, and collision-resistant. Deterministic means that the same input will always result in the same output. Hard to invert means that it is infeasible to find an input that would result in a predetermined output. Finally, collision-resistant means that it is hard to find two inputs that would produce the same output. In general, the output of a cryptographic hash function is considered a pseudorandom sequence. Then the cryptographic hash puzzle used in Bitcoin is as follows.

Definition 1. Using a cryptographic hash funtion H with integer outputs and an integer K find x such that H(x) < K.

Initially, hash puzzles were created as an anti-spam measure in e-mail services by Adam Back in HashCash [Bac02].

Bitcoin utilizes this algorithm in the following way. The puzzle's target is to find an output that starts with a minimum fixed number of zero bits, which if we translate bits into integers can directly translate to finding an output smaller than some integer K where K is a power of 2. Bitcoin uses the standardized SHA256 cryptographic hash function. In the protocol itself, whoever first finds such an input to the SHA256 function can generate the next block of transactions. It is easy to see that the optimal algorithm to solve such a puzzle is random guessing (in other words brute forcing) due to the pseudo-random nature of a cryptographic hash function. Then, the so-called mining in a cryptocurrency is the process of solving the hash puzzle, i.e., trying random inputs to the hash function.

The PoW protocol uses the hash puzzle as follows. The hash function uses the new block header as the input, where most of the values are fixed. Most importantly, all the transactions in the block are already chosen and used as inputs in the "hashMerkleRoot" value, while the previous block's hash is an input as well. In order to solve the puzzle, there is the nonce value that is chosen by the solver. This nonce is the value the solvers, or in other more well-known terminology, the miners, are manipulating to solve the puzzle. The motivation for a miner to try to solve the puzzle is a so-called block reward. The block reward is a fixed value transferred to the miner's chosen address, and it is always the first transaction in every block. This transaction is usually called the coinbase transaction. The miner can also claim transaction fees in this coinbase transaction. The transaction fee is the difference between the sum of the input and output values in a transaction (if the outputs have a larger sum of values than the inputs, the transaction is invalid). Furthermore, mining is a Poisson process with the number of proofs found in a given timeframe following the Poisson distribution and time between solutions following the exponential distribution. Thus solutions may start arriving almost immediately, but the average time is designed to be 10 minutes between blocks. To keep this average, an automatic mechanism makes the cryptographic puzzle harder or easier based on the average time difference of the most recent blocks. In Bitcoin, the recalculation of the difficulty happens after every 2016 blocks, which is roughly two weeks.

Coinbase transactions are the only way to mint new coins in Bitcoin. The base value without the transaction fees is fixed and halved roughly every four years. It originally started as 50 BTC per block while we just witnessed the most recent halving, which reduced the base reward to 6.25 BTC. With the constant increase in Bitcoin's price, more and more users have been trying to enter the mining space.

As the previous block's hash is one of the inputs to the current blocks puzzle and hash, all the blocks are chained together with these cryptographic hash functions. Due to the properties of cryptographic hash functions this results in the immutability of past blocks in the chain. This chain of blocks is the reason

1.2. Bitcoin 9

for the name blockchain. We usually refer to the current maximum length of the blockchain as the block height.

If a miner sees multiple valid blocks at the same time, it has to choose one to mine on. This choice is controlled by Bitcoin's built in longest-chain rule. This means, that the valid blockchain at any time is always the longest visible chain. If there are multiple valid blocks in the network, and they have the same block height, the miner mines on the one that it has seen the earliest. If the blocks have a different block height, it will always choose the one with the higher block height.

The situation when there are multiple valid chains parallel to each other is called a *fork*. If the longest-chain rule is invoked, that means there is a fork in the blockchain. There are different type of forks in a blockchain, let us quickly describe them all. The first is the naturally occurring one, when multiple blocks are mined around the same time, and the network has to decide which chain will it follow in the end. There can be purposeful forks as well in the history of a blockchain, of which there are two separate types. These usually coincide with an update to the structure of the chain. The first type is a soft fork, which means a change to the protocol that is backwards compatible, meaning old members who did not update their software will still recognize the new transactions and blocks as valid. The second type is a hard fork, which is not backwards compatible, every node in the network needs to update its software to keep participating in the network.

1.2.4 Mining Hardware

As the coinbase transaction is the only way to generate new coins, it quickly became a competition between the users of the currency who can have the highest mining power and generate the most coins. Mining power means the number of hash functions the user can compute in a fixed amount of time, usually 1 second. It is straightforward to see that the more hash functions one computes the likelihood of finding the correct solution increases linearly. The earned number of coins is not fixed, as mining is a probabilistic process, and the mining power only provides an expected earned number of coins over time. However, the longer the miner runs, the closer the expected earnings will get to the real ones due to the law of large numbers.

The original purpose of the PoW protocol was to have every processor in a typical desktop computer as one vote, replicating a usual voting type consensus algorithm. Then, all users would have a similar voting power, and the network would be relatively equally decentralized across all of its users. On the other hand, due to the large economic return of finding a block, miners started experimenting with different ways to increase their mining power. The first boost was to use a graphics processing unit (GPU) to solve the hash functions instead of the main CPU, as the SHA256 function needs only simple operations. The reason for this switch was that a GPU has 100s, or sometimes even 1000s of processing cores that can perform simple calculations. Due to the characteristics of the SHA256 function the mining software can be further parallelized on

a GPU, which resulted in a six time efficiency increase in mining compared to CPUs at the time.

Later, miners started using field-programmable gate arrays (FPGAs), which further increased the efficiency of the mining process, and they were roughly twice as fast as the GPUs at that time. With the constant increase in Bitcoin's monetary value, more and more development went into increasing the efficiency of mining.

Finally, application-specific integrated circuits (ASICs) were developed for mining. These are specialized pieces of hardware and software designed only for solving the SHA256 hash function. This approach lead to a further increase in the efficiency of mining. Currently, all mining in Bitcoin uses ASICs. A more in-depth description of this evolution can be found in [Tay13].

1.2.4.1 ASIC-resistant Hash Function

After some time using ASICs in Bitcoin it was argued that ASICs are unhealthy for the network and unfair to the users. They put too much power in the hardware manufacturer's hands, and miners need a substantial investment to start mining in the first place. These barriers have reduced the number of miners in the network, and blockchain designers and researchers started looking at other solutions for PoW mining.

The most popular idea is to use an ASIC-resistant hash function as the hash puzzle in the protocol. This would still allow GPU mining, but designing and manufacturing ASICs would be infeasible or unprofitable, resulting in a healthier mining ecosystem and easier access to mining hardware for average users. The difficulty is the design of such a hash function, and it is still being researched. The most notable blockchains that initially utilized ASIC resistant PoWs are Ethereum, Monero, and Zcash with different functions [eth20, vS13, BK16]. However, all of them have seen attempts from hardware designers to develop ASICs.

1.2.5 Mining Pools

Even though the idea of mining was to have one vote per user, the profitability of the mining process has resulted in new systems that created large centralized mining entities. These large entities are the so-called mining pools. The main idea behind mining pools is the following. If a miner has a fixed mining power, then the expected number of coins it will earn from mining is fixed. However, due to the randomness of the process, the variance can be quite large, especially if the miner has negligible mining power compared to the network. Thus earning those expected coins could take a long time.

On the other hand, a miner can join up with other miners and mine together with them, and if they can share the earned coins fairly compared to their provided mining power, they can reduce this variance in received coins. Meanwhile, the expected number of coins the user earns would stay the same. The amount of reduction in the variance is based on how much mining power the collection of miners has. Then the question is, how can one measure the 1.2. Bitcoin 11

amount of work produced by a user in this collection of miners, and then pay them out fairly for this work.

To solve this issue, miners introduced mining shares. In essence, a share is a solution to a relaxed version of the main cryptographic puzzle, i.e., the solution has to start with fewer zeros. If a participant finds such a share, it sends it to a central authority that monitors all the users in the collection of miners. The authority monitors the received shares looking for one that is not only a solution to the weakened puzzle but a solution to the main one as well. If there is such a hash, then the central authority creates the new block and relays it to the network. Afterward, the block reward is fairly split among the users based on the number of shares they have submitted to the authority. Such a collection of miners is called a mining pool.

In this scenario, the actual miners do not have to worry about creating the blocks or verifying the transactions. They are only performing the mining activity on the pre-filled out block headers received from the central authority, i.e., trying random nonces in the input of the hash function.

The impact of these mining pools is undeniable, and currently, in any significant PoW based cryptocurrency, the vast majority of mining is done in such mining pools (i.e., more than 90%). The problem with mining pools is that it creates a few central authorities that vote in the name of all their users, and the users have to put their trust in them. Of course, if a mining pool misbehaved, the users could change mining pool at any time. Furthermore, with mining pools, the number of entities that verify and relay transactions is massively reduced, as the miners do not have to communicate with the network.

1.2.6 Network Communication

The last remaining part of Bitcoin protocol is how the participating nodes communicate with each other. Bitcoin uses peer-to-peer (P2P) communication protocols. P2P communication protocols mean a decentralized network where the nodes directly connect without the need for a central authority. If every user had a direct connection to everybody else, it would oversaturate the network capacity. If there are N participants, there would be N^2 connections between all of them, which is not an issue when there are only a handful of nodes, but in Bitcoin's case, there are thousands of them (currently around 10,000). That would mean 50,000,000 connections that are regularly communicating with each other. To avoid these network capacity problems, Bitcoin uses a different approach.

As Bitcoin's goal is to have a decentralized currency, using central entities for control is counter-intuitive. Instead, Bitcoin uses the so-called gossip protocol [DGH⁺87]. In a gossip protocol, the number of outgoing connections from a node is limited to only a few, in Bitcoin's case, it defaults to 8, where an outgoing connection means that it sends frequent requests to these eight nodes on the latest transactions and blocks they have seen, and if there are things they have not yet seen, they request a full version of them. On the other hand, the number of nodes that can connect to a single node is not limited, only the outgoing connections are.

This type of communication between the nodes means that when a new transaction is submitted to the network or a new block is found, it is not instantly known by everybody in the network. Instead, it spreads across the users more gradually. There has been research by Decker and Wattenhofer [DW13] to estimate the propagation rate of information in the Bitcoin network and they found that the median time for a node to receive a new block is 12.6 seconds. On the other hand, even after 40 seconds, only 95% of the nodes have received the block.

1.2.7 Security of Bitcoin

The goal of Bitcoin was to achieve a decentralized, secure currency that addresses the issues presented in previous digital currencies. Let us expand on these issues, and explain how Bitcoin solves these problems. Furthermore, we will discuss a few problems that have emerged since the inception of Bitcoin.

First and foremost, Bitcoin solves the Byzantine Generals Problem with a probabilistic protocol. This solution is surprisingly robust, providing a higher security threshold compared to classical voting based protocols. This threshold is 50% of the mining power, whereas in classical Byzantine consensus protocols the threshold is 33%. Higher malicious participation than 50% would mean the attacker has a majority, and in an environment like that honest nodes could never agree, thus PoW is an ideal solution to the agreement problem. The PoW protocol was designed to solve the double-spending problem, but it has a side-effect of preventing Sybil-attacks as well. The reasoning is straightforward, as the number of new joining nodes does not influence the outcome of the PoW protocol, only the new users' mining power. Thus an attacker can not gain an unfair advantage by entering the protocol with many new nodes.

Let us look at how Bitcoin also defends against history rewriting attacks. To rewrite history, an attacker has to create a long fork, or in other words a valid chain of blocks from some point in the past to abuse the longest-chain rule to overtake the valid chain. For the attack to be successful, the attacker needs more mining power than the actual active chain, as otherwise, the new malicious chain would never catch up and get longer than the original one. If the mining power is large enough in the official chain, getting such mining power is deemed impossible. This attack can be dangerous in two cases. First, if there is new mining hardware developed that is kept secret from the public, and it is much more efficient and powerful than the current mining solutions. Such hardware can perform such attacks. Second, in case of a smaller blockchain, that uses the same PoW hash function as a major cryptocurrency, it is always vulnerable to such attacks, because the mining power is available and only has to be converted to the target blockchain. In this case, attackers may wait until it is profitable to perform such an attack on a chain.

A novel attack called selfish mining, was presented by Eyal et al. [ES18], which specifically targeted the PoW protocol. The main idea of the attack is that when a miner finds a block, it does not reveal it automatically, but starts mining the next block on it instantly. The attacker only reveals the block if other miners publish a new block. Then, using superior connectivity, the attacker

1.2. Bitcoin

can distribute its blocks faster than the organic spread of the competing new block. With such an approach, the authors showed that miners could obtain a revenue larger than their fair share based on their mining power, which proves that the bitcoin protocol is not incentive compatible. Based on some realistic assumptions of Bitcoin's network connectivity, the attacker only needs 25% of the mining power to perform the attack compared to the previous 50% bound for any attack.

Another novel attack directly targets mining pools. The attack is the block-withholding attack [Ros11], where the goal is to bankrupt an opposing mining pool. The idea is the following. A large miner joins a competitor mining pool, but it only submits shares that are not solutions to the complete cryptographic puzzle during its mining process. With this approach, the target mining pool will observe and estimate a high mining power from the attacker and pays the attacker its mined shares, but the attacker will never produce a valid solution. Thus the target pool overestimates its mining power, and over time it will not have enough funds to pay all its miners, which would lead to the miners in the target mining pool migrating to other pools.

Last but not least, the cryptographic security of the used primitives is an essential step in assessing Bitcoin's overall security. The SHA256 hash function [oST15] used in the PoW protocol is deemed secure against preimage and collision attacks by most cryptography researchers. The cryptographic security of it is 128 bits, which is well over the minimum required 80 bits. Regarding the public key cryptography used in Bitcoin, it is currently secure, but it uses elliptic curve cryptography, and is vulnerable to quantum computer attacks. If any time in the future quantum computers become a reality, every output in Bitcoin becomes vulnerable, and it would ruin current generation cryptocurrencies. However, this problem exists for any system and protocol that uses finite field-based public-key cryptography, which includes most security protocols on the internet.

1.2.8 Advantages and Disadvantages

Bitcoin users most commonly use a six-block confirmation time (1 hour) to avoid situations involving naturally occurring short-term forks in the network. The network will then have time to decide which fork they will continue mining on, thus preventing the double-spending. This one hour delay in transaction confirmation time makes Bitcoin infeasible to use on a day-to-day basis, like paying in a store. On the other hand, one hour for a transaction across continents is much faster than anything currently available commercially, and it is available to anyone in the world, as long as they have an internet connection.

The defense against the rewriting of history also means that every transaction in Bitcoin is irreversible. This can be regarded as both an advantage and disadvantage based on the situation. We would like to point out that due to the nature of the public blockchain, if a new user wants to join the network, it has to download and verify the entire chain before joining the network, which is continuously growing. The storage needed to run Bitcoin is currently 330GB, but every transaction will be stored forever and can be looked up at any time.

Finally, we would like to stress again that such an electronic currency being decentralized and not depending on any centralized authority is an extreme departure for currencies in general. Such systems were not even thought to be possible before the arrival of Bitcoin. Even then, a lot of the community was skeptical about it until it launched in the real world and showed that it is indeed a working currency.

Other serious complaints that are brought against Bitcoin is its slowness and lack of throughput. Even though, for example, VISA has 50,000 transactions per second capacity [vis15], Bitcoin's theoretical limit is only around ten transactions per second, not mentioning that it takes an hour to confirm a transaction, compared to seconds in a credit card network.

Finally we have to mention the accessibility of Bitcoin. In order to use Bitcoin, one has to use software that manages the funds and addresses. This software is usually referred to as the wallet. This wallet is usually one of two different types. A full wallet means, that the software actually downloaded all of the blockchain history and validated every transaction that happened in Bitcoin, and it stores all the corresponding information locally. Then there are so-called light wallets, which do not store a copy of the blockchain. Instead it uses a third-party server to verify and submit transactions. It only stores the corresponding private keys, and has the ability to create transactions, but for any other process it has to communicate with the third party, which can introduced new vulnerabilities.

1.2.9 Main Actors in Bitcoin

Over the last ten years of Bitcoin a clear hierarchy has evolved among the main members and actors in the community. Let us describe their roles and privileges in detail.

1.2.9.1 Miners and Mining Pools

There would be no PoW based blockchain without miners creating new blocks. As such, they are the most important and powerful entity in the community. They are the ones who mainly decide whether new updates should be launched on the network, or whether some issues should result in a hard fork of the chain. As most of mining is currently happening through mining pools, they have influence on the day-to-day activities of a blockchain as well. One might think there can be differences between what policies a miner and a mining pool supports, but pools usually do not require any commitment from the miners. This means that if a miner disagrees with its mining pool on some issue, usually nothing prevents it from switching to a mining pool that supports its agenda.

1.2.9.2 Mining Hardware Manufacturers

With the development of mining hardware from GPUs to FPGAs to ASICs, the actual manufacturer of the hardware became more and more prominent in the community. Nowadays, with ASICs as the only viable solution for Bitcoin mining the manufacturers behind them have gained a lot of power and influence

in the community. Furthermore, most of the manufacturers not only create hardware, but have their own mining farms and mining pools as well. With the steep price and inaccessability of the hardware, their role and influence is constantly increasing.

1.2.9.3 Exchanges

Exchanges are the primary platforms for users to convert coins between cryptocurrencies. Some exchanges have the legal permit to provide exchange between flat and cryptocurrencies, but there are much fewer of them. Most of the monetary exchanges go through these exchanges, which gives them influence as well. Currently a significant portion of all transactions are related to exchanges.

In some cases exchanges can be used as a wallet, as they also provide unique addresses to their users. However, in that case the user does not have any control over the private keys. If the user did receive coins to its designated address under the exchange, then more than likely those coins won't actually stay as an output in the real blockchain, instead the exchange will store the account balances in its internal database.

1.2.9.4 General Community

Finally the general community has considerable influence. Under general community we include among others the researchers who are either trying to improve, or look for problems and vulnerabilities in Bitcoin. Developers creating new services and products on top of the blockchain are a part of this community. Larger forums are also included, where many discussions on the decisions for the blockchain happen. But in these conversations the previous actors have a say as well.

1.3 Scalability of Cryptocurrencies

Many enthusiasts envision blockchains as a globally used life-changing technology. Even though Bitcoin has changed the way many people think about currencies and distributed systems, blockchains still face several issues that they have to solve before it could become a widely used approach. The first and foremost problem is the throughput of these protocols. As mentioned before, the current golden standard is VISAs electronic payment system, which can achieve up to 50 thousand transactions a second in peak conditions.

Comparatively, Bitcoin can only achieve around ten transactions a second, and other open P2P cryptocurrencies do not perform much better. This problem gets substantial attention and research, and there are constant improvements, but there are plenty of obstacles that first need to be overcome. Furthermore, not only the throughput, but the confirmation time is a crucial issue, as one can not expect to pay in a shop and wait minutes for a transaction to be confirmed.

Classic voting based byzantine agreement protocols do achieve some of the desired features, but they fall short on the openness of the network, as they can only function in closed or semi-closed networks. There has been recent work on

how to open these types of protocols up to the public. However, the resilience against Sybil attacks is a significant issue that is hard to overcome if there are no requirements for joining the network. In Bitcoin, the PoW protocol defends against it. Mainstream protocol examples for these types of blockchains are Stellar and Ripple [Maz15, CM18], which use a so-called federated byzantine agreement. There is a trust factor implemented in the network, meaning that every node chooses some other nodes that it will trust. It then needs 2/3 confirmation from these nodes only.

Some of the main ideas for PoW based protocols for scalability include the following. For both throughput and confirmation time, the idea of micro-chains has been popular. In essence, the idea entails having a single entity or a group of entities chosen by either PoW or some form of randomness. They then create multiple quick blocks of transactions until the protocol chooses a new entity or group. The most well-known such protocols are Bitcoin-NG, Byzcoin, and Algorand [EGSvR16, KJG⁺16, GHM⁺17].

Another common idea is sharding [KJG⁺18, LNZ⁺16, ZMR18]. Sharding means the division of the network into smaller parts. These smaller parts function independently, and occasionally they sync all the shards to have agreement among all of them. This allows for large scale parallel deployment of a blockchain, resulting in a linear increase in throughput with the number of shards.

Finally, there are payment channels [bit20, MMSH16]. The idea of payment channels is to have pre-set channels between two parties on a chain, where they can instantaneously (barring network speed and digital signature generation times) transact with each other, without having to send the transactions to the chain itself. Only the opening and the closing balances of the parties at the two ends of the channel are visible on the chain. This allows instant transactions between parties, solving the confirmation time problem as well. The drawback is that every channel is only between two parties, thus if there is no direct connection, the users have to use channels in between them or have the transaction on the chain itself. Furthermore, payment channels do not require any modification from the blockchain itself, as they are already available today on Bitcoin, Ethereum, and others.

1.4 Privacy in Cryptocurrencies

One of the first considerations about cryptocurrencies was that they can provide cash-like privacy for monetary transactions, as there is no direct relation between real-world entities and public keys or addresses on the blockchain. Furthermore, many argue that privacy is a human right, and no entity has the right to look into one's purchases, but that is not the case with our current electronic payment services. On the other hand, if such an electronic system existed, it would be the perfect environment for criminal usage, as the authorities could not connect people to their transactions.

As current digital transactions are always visible and trackable by banks, and with the infeasibility of cash payments in the current world, there is a definite need for a private electronic payment system. However, the question is whether cryptocurrencies are the solution.

If we take a closer look, some of the advantages of cryptocurrencies can also result in disadvantages. The first and foremost is the immutability of transactions, or in other words, transaction irreversibility. The same mechanism that defends against double-spending prevents the correction of human errors. If one sends currency to the wrong address, no higher entity or mechanism can reverse the transaction.

Furthermore, even Bitcoin is only considered pseudonymous. Every transaction has to be kept and can be verified at any point in time by any user, meaning all transactions that have happened in Bitcoin can be read and analyzed without any constraints, as it is a public ledger. If one accidentally reveals its pseudonymous address and could be connected to a real entity, an attacker can read every transaction ever done by that address, whether sending or receiving coins. These drawbacks lead to privacy concerns.

There are two main approaches for deanonymization in blockchains, namely network analysis and transaction graph analysis.

Network analysis entails observation of network traffic between nodes. For example, when and how a transaction appears in the network, or how a block propagates through the network. An attacker can use these approaches to deanonymize or identify users on the network. Earlier research [BKP14, BP15] has shown that, for example, using Bitcoin, even through Tor, a well known widely used P2P network anonymizing protocol, does not provide perfect defense against network analysis attacks.

The second approach is transaction graph analysis [RH11a,MPJ⁺13a,AKR⁺13]. In this case, the analysis takes place in a static environment, focusing on the already confirmed transactions in the chain. These methods try to connect users that otherwise do not have any direct relation, or try to cluster addresses with different methods to identify which addresses are controlled by the same entity. Furthermore, they can be used to monitor public entities' activity as well, trying to confirm or disprove their reported behavior.

There are plenty of other such observations and attacks specifically for Bitcoin, and most of them can be directly implemented in other blockchains. These problems instigated further research on how one can provide the advantages of a blockchain without the privacy disadvantages.

1.4.1 Privacy-oriented Cryptocurrencies

There are ways to have privacy-preserving services directly built into Bitcoin, and the most well known of them are the so-called mixing services. The idea behind them is that multiple otherwise unrelated users combine their transactions into a single transaction. Thus the inputs and the outputs of the transaction can not be directly linked to each other. Instead, it mixes these input-output pairs. The most well-known such service is called CoinJoin. On the other hand, Bitcoin is just not flexible enough as a protocol to support any larger scale privacy metric, and even the current techniques might be vulnerable according

to existing research [MB17]. Thus the community looked into designing new cryptocurrencies with privacy as the primary goal behind them.

There have been two main approaches to creating a privacy-oriented cryptocurrency. The first one is having the previously mentioned mixing services built into the blockchain. The first implementation of this system is in the cryptocurrency Dash [DD18], where so-called master nodes perform it. These nodes are network participants with special privileges, and they can create these mixing transactions after the users send transactions to them. To become a master node, one has to own a large sum of coins (1000 Dash). The drawback of such an approach is that the master nodes still know the exact links between inputs and outputs.

There is a second approach for transaction mixing in a blockchain where there is no need for special nodes. Instead, every transaction generated by any user is automatically a mixing transaction. In this case, the blockchain utilizes a new cryptographic protocol called ring-signatures [FS07] to achieve this effect. Ring-signatures are a new type of public-key digital signature protocol, where the main idea is the following. Let us assume there are N different public keys. Only knowing the private key pair to one of these public keys, a user can then create a ring-signature using all N public keys. With such a signature, an outside observer can only see that the ring signature was authorized by one of the N participants, but does not know which one. Then it is straightforward how to implement this in a UTXO type of blockchain by having the spender of an output during the construction of the transaction replace the digital signature with a ring signature. In this case the user would choose the other participating public keys randomly from the existing entire UTXO set. This mechanism would then replace mixing services with a built-in protocol. For further details go to [vS13, Noe15], as it is not straightforward how to correctly implement such a transaction system.

The most famous blockchain that utilizes this approach is called Monero [mon14], which is originally based on the Cryptonote [vS13] protocol. In Monero to further improve users' privacy, the protocol hides the actual values of the transactions as well using other cryptographic primitives, like commitments. It is far from obvious how to prevent double-spending in such a system, but that is outside of the scope of this thesis, and we direct the reader to the research papers on the topic for further information.

The other popular approach for privacy in a blockchain is zero-knowledge proofs [SMP87]. These are a type of mathematic theoretical proofs, where nothing is revealed apart from that it is indeed a correct spending transaction, and the sum of input and output values is the same. However, the addresses and values are completely hidden. In comparison to mixing type privacy where we still know one of the N addresses sent the transaction, with zero-knowledge proofs, the transaction could be sent and received by any of the blockchain users from the point of view of an outside observer.

These proofs are referred to as zero-knowledge succinct arguments of knowledge, or in short zk-SNARKs [BCCT12]. An argument of knowledge is different from a proof in terms of the adversary. In a proof, the adversary has unbounded

1.5. Contributions 19

computational power to try to break the proof, while in an argument of knowledge, the adversary is polynomially bounded. Succinct simply means that these arguments of knowledge are very short, in practical cases, constant sized. Finally, Zero-knowledge means that the prover proves the knowledge of a value without revealing any information about the value itself. The most well-known blockchain that utilizes zk-SNARKs is Zcash [HBHW16].

1.5 Contributions

In chapter 2, we analyze transaction linkability in Zcash based on the currency minting transactions (mining). Using predictable usage patterns and clustering heuristics on mining transactions, an attacker can link to publicly visible addresses over 84% of the volume of the transactions that use a ZK-proof. Since the majority of Zcash transactions are not yet using ZK-proofs, we show that overall 95.5% of the total number of Zcash transactions are potentially linkable to public addresses by just observing the mining activity.

In chapter 3, we further analyze privacy issues for the privacy-oriented cryptocurrency Zcash. We study shielded transactions and show ways to fingerprint user transactions, including active attacks. We introduce two new attacks, which we call *Danaan-gift* attack and *Dust* attack. Following the recent Sapling update of Zcash protocol, we study the interaction between the new and the old zk-SNARK protocols and the effects of their interaction on transaction privacy.

In chapter 4, we investigate the generic landscape and hierarchy of miners on the example of Ethereum and Zcash, two blockchains that are among the top 5 in terms of USD value of created coins. Both chains used ASIC resistant proofs-of-work which favors GPU mining in order to keep mining decentralized. This, however, has changed with the recent introduction of ASIC miners for these chains. This transition allows us to develop methods that might detect hidden ASIC mining in a chain (if it exists), and to study how the introduction of ASICs affects the decentralization of mining power. Finally, we describe how an attacker might use public blockchain information to invalidate miners' privacy, deducing the mining hardware of individual miners and their mining rewards.

In chapter 5, we analyze the behavior of cryptocurrency exchanges on the blockchain, and compare the results to the exchange volumes reported by the same exchanges. We show, that in multiple cases these two values are close to each other, which could mean that if they would not be, the exchange might fake its reported volumes. Finally, we present a heuristic to try to classify large clusters of addresses in the blockchain, and whether these clusters are controlled by an exchange.

Finally, in chapter 6, we describe how to couple reputation systems with distributed consensus protocols to provide a scalable permissionless consensus protocol with a low barrier of entry, while still providing strong resistance against Sybil attacks for large peer-to-peer networks of untrusted validators. We introduce the reputation module ReCon, which can be laid on top of various consensus protocols such as PBFT or HoneyBadger. The protocol takes

external reputation ranking as input and then ranks nodes based on the outcomes of consensus rounds run by a small committee, and adaptively selects the committee based on the current reputation.

Part I Data Analytics in Blockchains

Chapter 2

Deanonymizing Miners in Zcash

Zcash was launched at the end of October 2016, with the goal of providing a privacy preserving blockchain, where the privacy is based on mathematical proofs. However, due to technical details, it did not launch with complete protection, instead it provided the privacy techniques only as an optional, not mandatory feature. This lead to some privacy issues in the chain, that might not be obvious for the general user. Therefore multiple papers were published on the topic, and our work is part of this line of research.

This chapter focuses on the miners and mining pools of Zcash. The reason for this focus is the protocol requirement that every mined coin has to be shielded and converted to the privacy preserving part of the blockchain before it is used for general purposes. In this work we show multiple heuristics how one can link these shielding transactions to public transactions in the blockchain, negating the effect of the privacy preserving part of Zcash. Furthermore, we compare our results to earlier work on the topic by Kappos et al. [KYMM18], where they showed that the majority of shielded transactions are connected to mining pools. We show that our methods have improved their results by close to 20% (from 65.6% to 84.1%) in terms of the overall volume of revealed coins.

Before we present our finding on the privacy issues in Zcash, let us describe Zcash in detail. This chapter is based on joint work [BF19b] with Alex Biryukov.

2.1 Zcash

The first theoretic solution that used zero-knowledge proofs was the Zerocash protocol [BCG⁺14]. It was originally created as a direct extension of the Bitcoin blockchain, and it would have worked in tandem with it. It was the first protocol to utilize the novel zk-SNARK protocols.

Later the Zerocash protocol evolved into the more practical Zeash protocol, which launched as a real cryptocurrency at the end of 2016. It utilizes these zk-SNARKs for privacy, but there is some duality in the blockchain that we will explain in the following sections.

When Zcash launched, the zk-SNARK cryptography was still a novel technology, and thus the practicality of it in terms of performance was questionable. In order to create a single transaction with such a zk-SNARK one needed about 40 seconds and 3 GB of memory on an average PC at the time. This impracticality lead to a dual approach from the developers of the chain, with two

separate parts in the chain. There is a public portion, which works exactly the same way as Bitcoin does, and there is a private part which uses zk-SNARKs.

The structure of Zcash is similar to that of Bitcoin, as it is based on the Zerocash protocol. The blockchain itself is unspent transaction output (UTXO) based, using 2.5 minute block generation time and Equihash [BK16] as its proof-of-work function. The currency in the blockchain is called ZEC, while the smallest possible value is 1 Zatoshi, where 1 ZEC = 10⁸ Zatoshi. The default transaction fee is 10⁴ Zatoshi. The total supply of ZEC will be slightly less than 21 million, which is the same as in Bitcoin. The original mining reward was 12.5 ZEC per block, where 10 ZEC went to the miner who found the block and 2.5 ZEC went to the Zcash developers as the "Founder's Reward". The first halving has already occurred, so the Mining reward is now 6.25 ZEC, where again 5 ZEC goes to the miner of the block and 1.25 ZEC is used as the "Founder's Reward".

In general there are two types of transactions in Zcash. The first are transparent transactions. These transactions work the same way as Bitcoin transactions, with some previously unspent outputs as the inputs, and the new unspent outputs as the outputs of the transaction. The difference between the overall value of inputs and outputs is the transaction fee. They can only transfer coins between public or transparent addresses, which in the rest of the thesis we will refer to as t-addresses, since in the blockchain they start with a "t". Such transactions are also called t-to-t transactions and are currently the default (this may change in the future).

The second type of transactions are the ones that send or receive coins to or from a hidden address. These addresses start with a "z" and thus in the rest of the thesis we will refer to them as z-addresses. A transaction can use both t- and z-addresses, but the z-address is not revealed on the chain, only a proof that there is a valid z-address that sent or received an unknown amount of coins. In the rest of the chapter we will refer to any transaction that involves a z-address as a shielded transaction.

There can be 4 different types of shielded transactions:

- z-to-z transactions: the simplest case is where there is no public input or output, which means the transfer is only between z-addresses. The only revealed new amount is the transaction fee.
- z-to-t transactions: in these transactions there is no public input, but there is at least one public output, where the sum of the outputs has to be less than or equal to the revealed new coins, while the remainder is the transaction fee.
- t-to-z transactions: in this case, there are no public outputs in a transaction, only public inputs. The sum of the inputs has to be larger than or equal to the amount of newly hidden coins, while the remainder is the transaction fee.
- tz-to-tz transactions: the last case, where zk-SNARKs are involved, but there are public inputs and outputs as well in the transaction. In this case the transaction fee is the difference between the newly revealed coins of the zk-SNARKs and the sum of public outputs.

2.1. Zcash 25

	t-out	Yes	No	Yes
t-in	z-in/out	No	Yes	Yes
Yes	No	t-to-t	t-to-z	tz-to-tz
No	Yes	z-to-t	z-to-z	z-to-t
Yes	Yes	tz-to-tz	t-to-z	tz-to-tz

TABLE 2.1: Every type of transaction based on the type of input and output addresses, and how they are identified. Note that we can distinguish only 5 types (only 4 that are shielded) out of the 9 possible, as we do not know whether there was a z-address as input or output, when there is a t-address as input or output.

When a new coin is minted and rewarded to a Zcash miner, the miner can only claim the coin by transferring it to a shielded address first. This is an attempt from the Zcash developers to have every coin shielded at least once. Recently, Zcash has introduced [shi19] mining directly to shielded addresses. The problem with these shielded coinbase transactions is that the miner has to reveal the shielded address in the transaction, which can lead to different attacks against that address. These issues are described in more detail by Leto et al. [LD20], among other new attacks against Zcash.

The first version of Zcash was called *Sprout*. The shielded transactions in this version use a mechanism called joinsplit, which combines two previously unspent shielded outputs and creates two new shielded unspent outputs, while also being able to hide or reveal a clearly noted amount of ZEC. The limitation of this protocol apart from its efficiency is the fact that even if a user wants to send only 1 shielded output to another address, a dummy input and a dummy output still needs to be created to fill out the rest of the input and outputs of a joinsplit. Similarly, if a user wants to spend three unspent outputs, they will have to use two joinsplits in the same transaction. A joinsplit has two public parameters, the amount of previously public coins, called "vpub_old" , and similarly the amount of revealed new public coins, called "vpub_new". There is a pair of these values for every joinsplit in the transaction. We will refer to shielded addresses using this protocol as Sprout addresses. If we sum up every "vpub_old" value for every shielded transaction in a block b (let's call this sum $hidingsum_b$, and then do the same for every "vpub_new" value as well $(revealingsum_b)$, then the difference $hidingsum_b - revealingsum_b$ is exactly how many coins are in hidden Sprout addresses at the time of block b.

At block height 419,200 (29 October, 2018) the Sapling hard fork of Zcash took place, which introduced a new zk-SNARK protocol. This update is a major improvement in terms of efficiency, as proof times have been reduced from around 40 to 3 seconds, while the consumed memory is reduced from 1.5GB to around 40MB. On the other hand, the new zk-SNARK protocol is not backwards compatible, meaning that if a user wants to send coins from a Sprout to a Sapling address, it has to reveal the value in-between, which is currently only possible with the involvement of public addresses, see Figure 2.1. This means that transactions between Sprout and Sapling addresses are visible, as they have to use an in-between t-address.

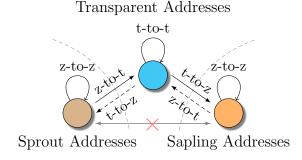


FIGURE 2.1: Sapling Turnstile

Sapling shielded transactions have also abandoned the joinsplit structure of the zk-SNARKs and they reveal exactly how many shielded inputs and outputs (these inputs and outputs are sometimes referred to as shielded notes) they have as side-channel information. The developers did consider including mandatory dummy inputs and outputs, but they decided against it in order to reduce the average transaction size. Furthermore, there is a new transaction field call "value_balance", which replaces the "vpub_old" and "vpub_new" values, and describes how many coins are being revealed or hidden in the transaction. It can be a positive or a negative value as well, depending on the type of the shielded transaction.

In the rest of the chapters, we may refer to t-to-z transaction as hiding transactions, and to z-to-t transactions as revealing transactions.

2.1.1 Notation

Let us describe in detail the notation that we will use in the rest of the chapter. The chain of blocks itself is denoted C. The n-th block of the chain is denoted C_n . The notation $C_{[n]}$ means the first n blocks of the chain, $C_{[-n]}$ means the last n blocks of a chain, while $C_{[k,n]}$ is the range of blocks from block k to block n.

The set of transactions in the block n is denoted X_n , while $X_{[n]}$, $X_{[-n]}$ and $X_{[k,n]}$ are the same as before, but in this case containing all the transactions in these blocks in their order. Shielded transactions are denoted X^{sh} . This means, that to list every shielded transaction in a range of blocks from block k to n would be denoted $X_{[k,n]}^{sh}$.

The inputs and outputs of a transaction $x \in X$ are simply denoted inputs(x) and outputs(x). To denote the input or output addresses of a transaction, we write inputs $_{adr}(x)$ and outputs $_{adr}(x)$. For the values of these addresses, we write inputs $_{val}(x)$ and outputs $_{val}(x)$. A single shielded transaction is denoted x^{sh} . The value vpub_old and vpub_new of the transaction is denoted x^{sh}_{vo} and x^{sh}_{vo} .

2.2 Analytics tool for Zcash

An important aspect of blockchains is that, in order to verify any transaction, a full node has to keep a database of every transaction that has ever happened. These data can become quite large: the Bitcoin chain is currently more than 330GB, while the Zcash chain is 27 GB. Because of the size of the database, the tools used for the analysis become an important aspect on their own, since the efficiency of the tool determines the number of different experiments we can run on the database.

We have created a tool specific for Zcash as a fork of the tool BlockSci [KML⁺20], specialized for Bitcoin [Nak09] and its hard forks. The original tool is reported to be many times faster than any previous tool. The tool is available at https://github.com/cryptolu/BlockSci/. Let us describe it in more detail.

The tool itself is written in C++. It uses an indexed file system based approach. An indexed file means that every record in a file has a unique key to it, which allows easy and fast random access to the records. In the case of BlockSci, there are multiple such indexed files for the different data formats in a blockchain (blocks, transactions, etc.). For every data format there are two files in the system. The first file is a list of pointers, where every pointer has a fixed size. Based on the index of the pointer, i.e. which place is it in the list of pointers in the index file, the index file returns a pointer that describes the exact starting point for the specific indexed record in the actual large file of records. Then, in the file of records, the record itself first has a few bytes of fixed length data, that describes exactly the length of the record, which then we can also parse. This means, that if the program knows the exact index of the record, it can retrieve that record through the index file with fast random access.

In our case, the original version of BlockSci handles Bitcoin based cryptocurrencies without any modification. As Zcash is also a Bitcoin based blockchain, the public part of the chain can be parsed with the tool without any major issues. The public addresses use the same elliptic curve based public key cryptography (ECDSA), thus these scripts did not require any changes. The only modification we had to make was to parse and save the nonces of Zcash blocks, as Zcash uses 256 bit nonces instead of 32 bit ones found in Bitcoin. The tool needed this change because block header elements has to have a fixed size in this indexed file system, and the original size for nonces was insufficient.

On the other hand, support for shielded transactions does not exist in the original version of the tool. In its original format, shielded transactions are simply recorded into the database as transactions without inputs or outputs. This behavior also affected the calculations of transaction fees. For example, in hiding transactions, the original version would return as the transaction fee the entire sum of input values, if there was no public output in a transaction, as in Bitcoin the fees are simply the difference between the sum of input and output values.

We have added full support for shielded transactions in our fork of the tool. This required changes and additions to the structure of transactions in the database. First, we have added a basic indicator value, whether a transaction

is a shielded transaction, and whether it is a Sprout or Sapling shielded transaction. Furthermore, depending on the type of shielded transaction, we have added their public information as well. In the case of Sprout transactions this means the number of joinsplits, and the exact value revealed or hidden by every joinsplit in the transaction, and we have included the sum of these "vpub_old" and "vpub_new" values as a built-in function as well. For Sapling transactions, we have added the "value_balance" field, and we also note the number of shielded inputs and outputs in a transaction. We have also modified the transaction fee calculating function to accommodate for shielded transactions.

Finally, the original tool provided a Python3 interface for the library to use. We have expanded this interface with all the additions we have made to the tool.

In our tests¹ we used the Python interface of the library, as Zcash is still fairly small in size compared to Bitcoin, and the efficiency of our functions was still manageable. As an example of the efficiency, we ran a quick test, where we cycled through every single transaction, examining whether they involve a zk-SNARK, and if they do, we keep track of how much value was hidden and revealed overall, while also keeping track of the different kind of shielded transactions (Table 2.1). This script finished in 6.5 seconds without any parallelization in the code (for 416,062 blocks and 3,993,633 transactions).

There are some drawbacks to using this tool as well. First of all, in order to compile the Python library we had to use a machine with 64GB of memory. Furthermore, the database is a static database that needs manual updates if the user wants to ensure it always sees the most recent information available. On the other hand this parsing can be relatively slow, as parsing the entire chain takes many hours. Even then, there can be issues with constantly updating the same database and it might require the full reparsing of the chain once in a while.

2.3 Related Work

There have been multiple studies focusing on deanonymizing blockchain transactions [RS13, MPJ⁺13b, BKP14], some with a special focus on the privacy preserving blockchains, mainly on Monero [MSH⁺18, KFTS17] and Dash [KML⁺20].

There has been a short paper concerning Zcash by Quesnelle [Que17] focusing on just one of the predictable usage patterns - the so-called round trip transactions. The paper describes the linkage of equal hiding and revealing values, where the time difference is short between them. Another recent article about Zcash is by Kappos et al. [KYMM18], which is in general more concerned about directly deanonymizing specific entities, while our work is purely focused on mining pools (which are the source of Zcash and produce majority of the shielded transactions). In the rest of the chapter we will show the shortcomings and inconsistencies with mining pools in that paper by reproducing their results while also providing novel clustering heuristics for mining pools that perform significantly better. Our paper also covers 11 months more data, which is

¹Computer Specs: AMD Ryzen 5 2400g, 64 GB RAM, Running Ubuntu 18.04

roughly 61% more blocks and 78% more transactions, as Zeash was still only 2 years old at the time of this study.

2.4 Deanonymizing The Miners

Until recently, Zcash block rewards were always claimed by sending the coins to a z-address first. After that, the owner of the coin can use it freely. This led us to investigate how miners and mining pools use their rewards: do they convert it back to public addresses, and if yes, could a connection be found between these two transactions? Another aspect is that currently at the time of writing there was no mining pool (that we know of) that supported payouts to a hidden address. This means that these miner payouts have to be visible somewhere on the blockchain (as they are paid to public addresses), so the mined coins are visible.

There are two general patterns for payouts. The first one is converting the mined coins back to a public address controlled by the pool, and then paying the miners in public transactions (we will call it pattern T). The second pattern is paying the miners directly from a hidden address to the public addresses (pattern Z). This means that the transaction on the blockchain appears as having no inputs (since the coins are sent from a hidden address) but having tens, hundreds, or sometimes even thousands of outputs. For both cases the single payment per address is usually in the range of 0.001-0.1 ZEC. Since this is a very specific transaction structure, it is easy to recognize.

To link transactions to mining pools, the simplest method is checking the website of the mining pool, and whether they have a top miner section with the miner's public Zcash address. If this information is available, we can scan for these addresses and their latest received transactions, mapping them to the previously identified payment structures. This way we can identify which pattern the specific mining pool is using. We will elaborate later how to identify mining pool transactions without the top miner section of a mining pool in both of these payment patterns.

The following study is performed at block height 416,062, which is equivalent to 23 October, 2018, which means it is around the two year anniversary of Zcash.

2.4.1 Pattern T Mining Pools

In the case of pattern T, after a constant public address is found, the rewards are transferred directly to a set of addresses controlled by the same entity a (signed as controlled(a)). By summing up the total amount of received coins, they become linkable to the specific hiding transactions of the mining pool and the direct connection between the hiding and revealing transactions can be made (Heuristic 1).

Even if the information on top miners is not available, the payout transactions of a mining pool can be still found. To find pools using pattern T, first scan through every shielded transaction disregarding the ones that are already identified. Calculate how much ZEC any public address received from a hidden address in the same range of blocks as before, and then compare and

Algorithm 1 Pattern T Heuristic, with a starting address a

```
procedure PatternTPool(a)

PoolAddrs \leftarrow controlled(a)

PoolTxs = \emptyset

for x \in X_{[k,n]}^{sh} do

if \exists outputs<sub>adr</sub>(x) \in PoolAddrs then

PoolTxs \leftarrow x

end if

end for

return PoolTxs

end procedure
```

correlate these values to how many blocks different mining pools mined. If one extends this approach to multiple scanned block intervals, the link becomes even stronger. We used this approach to find the corresponding addresses and will describe the results in more detail in section 2.4.3.

We have tested this approach with a varying set of block ranges and accuracy requirements. The block ranges we used were 500, 1000, 2000, 4000, 8000 and 10000. In terms of accuracy of a match we used 10%, 5% and 1%. In our tests there was no overall block range that worked for all of the pools. The reason for that is that mining pools transfer their received coins through a shielded address on a different schedule. Some pools shield their coins instantly, while others might wait to accumulate thousands of coins to hide and then later reveal them. The latter would justify using larger block ranges (e.g. more than 4000). On the other hand if a pool changes its address often, the long range might not provide enough matches, whereas a shorter range would work better.

The biggest drawback of this method is that it is difficult to link small mining pools that have only mined a handful of blocks, as in those cases there are always multiple matches for the amounts of revealed values. In statistical terms this is not a huge problem, as these are only marginal pools and they provide less than 1% of the mining power (number of blocks mined). If higher precision is necessary one could mine in these smaller pools, to identify the relevant addresses.

2.4.2 Pattern Z Mining Pools

For pattern Z, the connection between the hiding and revealing transactions is not trivial, as there is no single constant address, instead hundreds - or sometimes thousands - of addresses. However, the actual miner addresses will reappear regularly in the pool reward transactions with a frequency depending on the power of their mining hardware and on the frequency of the pool payouts. Thus we scan every shielded transaction and find the ones with the pool-payout structure (i.e. lots of outputs). Once a pattern Z transaction is found we check its outputs, and look for overlapping addresses with the already existing set of miner addresses. If the number of overlaps exceeds a certain threshold (e.g. ≥ 40), we consider that transaction to be sent by the same mining pool, and

also expand our set of miners with the new addresses. Scanning iteratively through a range of blocks until new transactions and miners can be added to the existing sets, it is possible to find most of the transactions connected to a pool (Heuristic 2).

Algorithm 2 Pattern Z Heuristic, with an sx as a starting transaction

```
procedure PatternZPool(sx, X_{[k,n]}^{sh})
    Miners \leftarrow outputs_{adr}(sx)
    PoolTxs \leftarrow sx
    OldMiners = \emptyset
    while OldMiners \neq Miners do \triangleright Execute, until the miner set cannot be
updated anymore
        OldMiners = Miners
        for x \in X^{sh}_{[k,n]} do
            if |\text{outputs}_{adr}(x) \cap \text{Miners}| \geq 40 \text{ then}
                 Miners \leftarrow outputs_{adr}(x)
                 PoolTxs \leftarrow x
            end if
        end for
    end while
    return PoolTxs
end procedure
```

The drawback of this approach is, that it is only usable for shorter periods of time, e.g. 2,000 blocks (~4 days), as we have observed that miners sometimes change their mining pools. If the range of blocks is too large, because of the migrating miners one might consider a transaction from a different pool to be the same as the currently investigated one, since the number of overlapping addresses would become too high. If that happens even for one transaction, from that point on the heuristic might identify even more transactions from the different pool, creating a very large set of transactions and miners of multiple mining pools.

The accuracy of the identified transaction can be verified by adding up the overall value of the payouts and the number of blocks the mining pool actually mined, and then comparing these two values to see if they are close to each other. The algorithm accepts if the difference is small ($\leq 5\%$).

Let us call the set of remaining not-yet-linked mining power from block k to n UnknownPower(k,n). To find a pool using pattern Z that does not have a top miners section on their website we use the following approach. First we disregard every shielded transaction that has already been identified. Then we look for transactions that have tens of outputs. For every such transaction we check with the previous method (2) for overlapping addresses in them, add up the overall received value and compare it with the number of mined blocks per mining pool (3).

Choosing the correct threshold values is a non trivial task, and because of that we have tested the heuristic with a wide selection of them. We ran the algorithm with differing block ranges (the same ones as for pattern T), while we have also used a different number of overlapping miners (10, 20, 40, 80). Our experiments show that using smaller overlap sizes (10) causes over-identification of transactions because of miners who either mine for multiple pools, or switch pools in-between, while a too large threshold value will not identify enough transactions. This problem becomes worse with each increase in the block range, as it is more likely that more miners will switch a mining pool in a larger timespan.

Algorithm 3 Heuristic for finding a pattern Z style mining pool without a base_tx

```
The set of uncovered miner transactions are used as MinerTxs  \begin{array}{l} \operatorname{UnknownTxs} = X_{[k,n]}^{sh} \setminus \operatorname{MinerTxs} \\ \operatorname{NewPools} = \emptyset \\ \text{for } x \in \operatorname{UnknownTxs} \text{ do} \\ \operatorname{NewPools} \leftarrow \operatorname{PatternZPool}(x,\operatorname{UncoveredTxs}) \\ \text{end for} \\ \text{for } p \in \operatorname{NewPools} \text{ do} \\ \text{if } \sum_{\mathbf{x} \in p} \operatorname{outputs}_{val}(x) \in \operatorname{UnknownPower}(k,n) \text{ then} \\ p \text{ is the set of payout transactions for the pool with the matching mining power} \\ \text{end if} \\ \text{end for} \\ \end{array}
```

2.4.3 Results of the Heuristics

If we consider the entire chain, then our heuristics linked 88.4% of the mining reward movement in the shielded addresses. In the following table (Table 2.2) we show the exact results per mining pool, where we also provide the amount of value mined by the pool to show that our linking was unique. Overall our heuristics linked 84% of the volume of z-to-t transactions, if we add the simple Founder Heuristic from [KYMM18]. Without those large valued transactions we cover 70.3% of the volume.

Below we show how many transactions our heuristics linked this way and how many transactions were in the different categories before and after this process.

- Remaining shielded transactions: 179,057 (originally 534,944, 66.5% have been linked)
- t-to-z: 58,557 (originally 222,306, 73.6% linked)
- z-to-t and maybe z-to-tz: 88,616 (originally 280,754, 68.4% linked)
- z-to-z: 14,431 (8% of the remaining transactions)
- tz-to-tz: 17,453 (9.7% of the remaining transactions)

²We could only verify MiningPoolHub's payouts from block 193,000 and BitClub Pool's payouts from block 120,000

Name	Pattern T	Pattern Z	Mined	Linked
Name	rattem i	rattern Z	Value	Portion
Flypool	14,435	94,277	1.79M ZEC	0.995
F2pool	1,075	0	1.35M ZEC	0.994
Nanopool	0	40,083	338K ZEC	0.981
Poolin	126	0	138K ZEC	0.996
Suprnova	12,920	0	167K ZEC	0.961
Coinmine.pl	0	7,204	78K ZEC	0.925
MiningPoolHub	$7,598^2$	0	156K ZEC	0.999
BitClub Pool	67^{2}	0	1.9K ZEC	0.969
DwarfPool	2,953	0	27K ZEC	1.0
Slushpool	3,027	0	49K ZEC	0.999
Antpool	378	0	93.8K ZEC	0.999
Zpool.Guru	88	0	824 ZEC	1.0
Nicehash	203	0	429 ZEC	0.999
Luxor	185	0	6K ZEC	1.0
Solo Miners	3,698	0	43.8K ZEC	1.0

Table 2.2: Results from our heuristic per mining pools, only for pools and miners where the linked transactions by our heuristics were verified by the overall mined values

In terms of thresholds, as we have mentioned earlier we have experimented with many of them, and for Pattern Z we achieved the best results for a range of blocks of 2,000 and an overlap of 80 for Flypool and Nanopool, while if we remove those transactions from our set, then for Coinmine.pl the best results were achieved by an overlap of 20 miners and a block range of 10,000. Choosing a too small overlap and/or too large block range, results in marking too many transactions because of miner migration and mining for multiple pools, while too large overlap and too small block range will result in marking not enough transactions.

2.4.4 Accuracy of the Heuristics

In case of pattern T payouts, the transaction linkage is sound and verifiable by comparing the number of blocks mined by an entity and the amount of ZEC the suspected address received in the same period. Indeed, these two values only differ by a small amount (5%), and there is no other entity with a similar mining power in the inspected interval (more than 10% difference).

In case of pattern Z payouts, the verification is done similarly, but in this case it can not be decided whether every single transaction was found, as a statistically negligible number of payout transactions could be missed. On the other hand, if an attacker starts with a payout transaction from a different mining pool, the resulting set of transactions will be disjoint, if the parameters are set correctly and the overall payed out value matches the number of mined blocks. Both of our heuristics are verified with the results in Table 2.2.

2.4.5 Comparison of results to previous work

Below we attempted to reproduce the results (Table 2.3) of the paper by Kappos et al. [KYMM18], where we followed the instructions step-by-step provided by their paper. We also compare our verified results to theirs on the same dataset.

Name	Kappos et. al. [KYMM18]	Reproduction	Our Work
Flypool	3	3	67,985
F2pool	720	717	717
Nanopool	4,107	3,568	19,984
Suprnova	0	0	11,185
Coinmine.pl	0	0	6,678
Waterhole	5	5	0
BitClub Pool	1,516	1,210	101
MiningPoolHub	0	0	1,335
DwarfPool	1	1	2,833
Slushpool	0	0	941
Coinotron	0	0	0
Nicehash	0	0	203
MinerGate	0	0	0
Zecmine.pro	0	0	0

Table 2.3: Comparison of results with [KYMM18] in terms of number of linked z-to-t transactions, reproduction of their and our Pattern T (1) and Pattern Z (2) heuristics on the same dataset

In their paper they claim to uncover 120,629 of the z-to-t transactions that are connected to miners, but based on the paper and the data they report one can only count their Founder transactions (1,953) and the ones in Table 2.3, which overall sums up to 8,305. On the other hand we have implemented their heuristics, and the heuristic did return 98,274 miner transactions accounting for 62.8% of the overall value (which is more than their reported 52.1%). If we remove the restriction of at least 100 outputs in the transaction, we uncover $\sim 122,000$ transactions accounting for 69.4% of the overall value. We could not find out what causes these discrepancies.

We uncovered these results using every address that ever mined a block. When we reduced the addresses to only ones controlled by a mining pool, the results reduced to 6,853 transactions and 26.9% of the revealed value. This steep reduction was caused by the following. There were 3 addresses that mined a single (or very few) block that are controlled by a large exchange. If we include all transactions where the receiver addresses are controlled by someone who mined a block, we include all the addresses that are controlled by the exchange, which results in thousands of addresses. As a lot of miners mine directly to an exchange address, most of these transaction are actually payouts but they are not found based on the logic presented in the paper. This is reinforced by the discrepancy between their reported number of uncovered transactions (120,629) and the number of z-to-t transactions linked in Table 2.3.

Our opinion is that the authors did not consider the difference between how many blocks somebody mined and how much value was paid out. With our implementation of their heuristic there are cases where a miner mined only 1 block in the history of the chain, but is somehow responsible for the payouts of thousands of transactions. Otherwise we might have misunderstood something based in the paper, but we tried to implement the heuristics literally.

2.4.6 Inflation of results

The results of [KYMM18] in percentage terms is inflated by the behavior of F2pool. F2pool is one of the largest mining pools that used a payout structure that was a mixture of pattern-T and pattern-Z payouts which we describe in detail.

First, let us refer to the address corresponding to F2Pool that received the minted coins in a coinbase transaction as T_{F2} . As we have described earlier, in order for a miner to use the newly minted coins, it has to spend it first to a shielded address. In case of F2Pool this happened on average 1.6 times a day. Then in a single revealing transaction the pool paid out all of its miners, where the value of the revealing transaction was the same as the value of the hiding transaction minus the transaction fee. Among the output addresses one address that also receives coins is T_{F2} . The next time F2Pool is hiding its mining reward coins it has the output of the previous revealing transaction among the inputs.

This constant loop of coins inflates the revealed amount of coins from a shielded address. After calculating it exactly, we have found that this single loop is responsible for 492 thousand coins of the overall 3.788 million coins (13.1%) revealed during the timespan of [KYMM18]. If we remove the loop and only consider coins that were not part of it, their original 65.6% result is reduced to 60.5%, while our approach decreases from 84.1% only to 82% (considering only mining and founder transactions in both cases).

2.5 Summary and Conclusions

In this chapter we have shown two heuristics to link mining related hiding and revealing transactions in Zcash. We have linked over 88.2% of the mined coins tracked through the shielded pool. Overall we increase the coverage based on mining pools in terms of the overall revealing transaction volume of Zcash from 65.6% in [KYMM18] to 84.1% on the same dataset.

This work shows that even if a blockchain is theoretically safe, bad use practices and an intermix of hidden and public transactions can lead to considerable information leakage defeating the very strong cryptographic privacy features of Zcash. Moreover, since hidden transactions form only 13.4% of the total number of transactions, 95.5% of all (including both public and shielded) Zcash transactions were potentially linkable, which is very close to privacy level of the original Bitcoin blockchain. This study shows that if there is no proper incentive for the mandatory usage of privacy preserving techniques, the overall effect of these techniques can be negligible. Furthermore, we can provide

a more accurate general privacy study on Zcash if we remove these mining related transactions from the shielded pool of transactions. This further study is available in Chapter 3.

Finally, we note, that if mining directly to shielded addresses by the mining pool members became more popular, it would reduce the accuracy of our heuristics, as the hiding and revealing volumes might not be equal or close to equal anymore. As of now we have not observed widespread implementation and usage for payouts directly to shielded addresses.

Chapter 3

Further Transaction Linking in Zcash

In this chapter, we continue our privacy research on Zcash, where we utilize the presented techniques in Chapter 2 to reduce the number of observed shielded transactions to a smaller set, allowing for further linkability in the blockchain. We focus on the linkability of t-to-z and z-to-t transactions with the end goal of reducing the privacy in Zcash to the equivalent of Bitcoin privacy. The overall goal of our research is to show that Zcash is not just a black-box type of solution, where if a user uses the shielded transactions, it has constant perfect privacy. Instead, we show that even the users of Zcash have to put some care into how they use the shielded transactions, and how they transfer coins between the public and private parts of the blockchain.

In this chapter first we show general transaction linkability methods based on the exact value of the hiding and revealing transactions. We also analyze the success and false positive rates of these methods. Furthermore, we show two novel active attacks - Danaan-gift attack and Dust attack - against Zcash user privacy, which shows that even Zcash users can be vulnerable to some attacks. This chapter is based on joint work [BFV19] with Alex Biryukov and Giuseppe Vitto.

The following study was completed on Zcash up to block number 472,284, which is equivalent to January 29, 2019.

3.1 Linking with Transaction Values

Utilizing the previous techniques in Chapter 2 we can eliminate a large portion of shielded transactions and only focus on ones that are probably not performed by miners. For this work we have focused on these remaining shielded transactions. This removal of transactions resulted in 92,233 hiding and 107,772 revealing transactions as a hard-core set of remaining hidden transactions. First, let us show the most trivial way of linking hiding and revealing transactions.

Heuristic 1. If a hiding and a revealing value are exactly the same, their values are unique as hiding and revealing values in the observed block range and the hiding transaction is in an earlier block than the revealing transaction, then they are considered to be linked.

Using the direct match approach (first shown in [Que17]) on these remaining transactions we find that 8,954 revealing shielded transactions out of 107,772 are uniquely matchable considering the entire Zcash blockchain.

To acquire a false positive rate for the heuristic, we deployed the following measurement. First we measure the number of unique transaction output values in the first half of the blockchain (up to block 236,142). Then we measure how many of these values stay unique in the second half of the chain. We found, that 14.7% of these values lose their uniqueness (769,071 out of 5,230,325). This means that direct value matching heuristic has false positive rate of just 14.7% over the half-blockchain duration (13 months), and less over shorter windows of time.

3.1.1 Direct Value Linking Including Transaction Fees

We have also found a handful of interesting transactions, where both the hiding and revealing value is unique, but the hiding value is larger by exactly 10,000 Zatoshi, which is the default transaction fee. This leads us to a likely explanation, that the value was also moved once as a z-to-z transaction, and then the receiver is revealing it to the public. Either a user moved the coins to himself between shielded addresses wrongly assuming that this way he gets more anonymity, or more plausibly a change of ownership happened for the coin (and also not gaining anonymity compared to a change of ownership occurring in the plain sight with the t-addresses).

We have investigated and extended the observation above to the case where instead of a direct value match there is a difference in value which is a multiple of the default transaction fee of 10,000 Zatoshi. This could correspond to a value making several hops inside the shielded pool before being revealed, each time losing 10,000 Zatoshis. From this the observer can also conjecture the number of hops that the value made.

Heuristic 2. If a hiding value is n times 10,000 Zatoshi (n < 10) larger than a revealing value, their values are unique as a hiding and revealing value in the observed block range, the hiding transaction is in an earlier block than the revealing transaction and the transactions have not been linked for any k < n, then they are considered to be linked.

If we extend the possible linkable values with this technique, the original 8,954 unique links are increased to 9,919, which is a 10.8% increase considering only 1 hop. Interestingly the number of unique links does not increase by much more by increasing the number of hops (only 400 more links with up to 9 hops¹). This seems to confirm our previous observation that many users think of the shielded pool as a perfect cryptographic anonymizer and think that a single z-z hop inside the shielded pool is sufficient. This expectation of gained privacy is not true if the value passing through the shielded pool was unique or rare in the entire chain. This also leads us to an idea of value fingerprinting described in the later section.

¹Our maximum allowed since 10 hops would be 100,000 Zatoshi, which is a common value on its own.

3.1.2 Subset sum

We have investigated the usefulness of subset sums for linking shielded transactions by connecting a single hiding transaction to multiple revealing transactions, or vice versa. The idea is to check if users hide their value in a single transaction, but reveal it over time in multiple payments, or similarly hide their coins in multiple phases, while revealing it in a single transaction.

First, we had to consider how many numbers can be summed up overall. If we consider the average number of remaining shielded transactions, we see that about 2,000 coin revealing and hiding transactions remain for every 10,000 blocks after removing the mining transactions. The smallest unit of account in Zcash is $1 \ Zatoshi = 10^{-8} \ ZEC$. Then any transaction value can be in the range of $[1, \ldots, 10^{14}]$ Zatoshis. If we consider the number of possible combinations of 2,000 inputs or outputs, then it is easy to see that even $\binom{2000}{3} = 1.3 \cdot 10^9$ which is well above the birthday bound of all possible values (which is around 10^7). This means, that some sums of just 3 values could be just coincidence. Nevertheless this could be still of interest since even in case of collisions we can reason in terms of sizes of anonymity sets (which would still be relatively small).

The technique did find a handful of interesting matches for 2 sums, especially in cases where one member of the sum is a value with multiple non-repeating non-zero digits, while the other is a round value (e.g. consider a transaction with the input value 3.54156325 ZEC and the two outputs with the values 0.40002 ZEC and 3.14154325 ZEC²). This led to a further analysis idea, which is explored in the following sub-section.

3.1.3 Fingerprinted Values

We have found another promising technique for connecting different hiding and revealing values that had no direct connections so far. We will use an approach that we call value fingerprints. In our definition, the fingerprint of a transaction value is its last 7 digits in Zatoshis. In particular the last 4 digits are especially stable as a fingerprint since this value is below the typical transaction fee of 10⁴ Zatoshis (which is currently below 1 US cent). Thus they usually have little economic meaning and represent just a remnant of previous transactions. The distinguishability of a fingerprint depends on its entropy, which in this case describes how rare the value is. Intuitively, round values are much more frequent than random values. It is worth noting that, in a regular economy, the digits below the fee threshold would typically stay zero. However this is not the case in the blockchain world, where mining pool payouts are in most cases computed with full precision thus creating random distribution in the least significant digits and which can be used for transaction fingerprinting. This is somewhat similar to the serial numbers on the paper cash banknotes, with a difference that the precision is not sufficient to keep the transactions unique.

Heuristic 3. If a hiding value matches fingerprints with a revealing value, no other hiding or revealing value matches fingerprints with either of them in the observed block range and the hiding transaction is in an earlier block

²These are fictitious values to preserve the privacy of the actual transaction.

than the revealing transaction, then they are considered to be linked with the fingerprinting technique.

We consider two fingerprints to match if either 5 of the last 7 digits are the same, or all last 4 digits are equal. Fingerprints where the last 3 digits are zeros (e.g. '0000', '1000') are disregarded.

In Section 3.1.1 the heuristics linked transactions where one of the values in the sum or the difference of the values is exactly 10,000 Zatoshi or a multiple of it. These transactions are linked with the fingerprint technique as well, since the last 4 digits remain the same.

3.1.3.1 Longevity of Fingerprint Uniqueness

One concern with this method is that the possible fingerprint set is too small, resulting in no unique links if we consider the entire chain for pairs of hiding and revealing transactions. In order to check the relative strength of the linkage, we have investigated several block ranges to see how many unique pairs are evident. Table 3.1 describes our results in terms of block ranges and the number of unique links.

The block range is a sliding window of blocks, where if the range is N, we consider hiding fingerprints in the first N/2 blocks and we check for matching revealing fingerprints in the last N/2 blocks. Once a matching pair is found, it is not considered in the subsequent checks. The valid transaction set is the same hard-core set as we used for the case of exact value linking.

Sliding Window Block Range	Unique Fingerprint Links
20	5,448
100	9,436
500	9,681
1,000	9,733
2,000	9,761
4,000	9,808
8,000	10,335
12,000	10,613
16,000	10,642
24,000	10,363
32,000	9,736
64,000	6,833

Table 3.1: Number of unique fingerprint matches through the entire chain for a given range of blocks.

We will call this feature the *longevity of fingerprint uniqueness*. As seen in Table 3.1 the largest unique match was on a sliding window of 16,000 blocks (which is approximately 4 weeks), while we received similar results between 8,000 and 24,000 blocks as well. This means that the average longevity of a fingerprint (aka "serial number") staying unique is approximately 2-6 weeks.

Note that the database is the same as for exact value matching. If we disregard the direct unique value matches (Section 3.1.1), on the range of 16,000 blocks, we receive a number of 7,228 unique matches (Section 3.1.4, Table 3.3) out of 107,772 total transactions.

Another approach for determining the longevity of fingerprint uniqueness is examining public transactions to determine how long the fingerprint of an output stays unique. We have implemented this approach by calculating the average number of transaction outputs it takes for a duplicate of a fingerprint to appear considering all (except mining) transactions in the chain. In this case we have recorded a fingerprint from an output, traced and saved the connected chain of transactions where this fingerprint still exists, and then checked when the fingerprint appears again from an unrelated transaction output.

The result of this experiment was that the average number of outputs until a fingerprint is unique was 15,979, while the median was 16,788 outputs. Then we have approximated the average number of shielded outputs in a shielded transaction from the Sapling transactions (where this information is visible) and calculated the average number of shielded outputs per block, which is 0.95. Then by dividing these two values we get 16,820 blocks for the average and 17,671 blocks for the median value, which is in line with our measurements from Table 3.1. We have also created a plot, shown in Figure 3.1, which for every last 4 digit fingerprint represents the average number of outputs generated by the blockchain during which the fingerprint stays unique (after we removed all the mining-related transactions). From this figure we see that a "good fingerprint" stays unique during generation of about 16000 new outputs by the blockchain, which is about one month. In Table 3.1 the decrease in matches after increasing the window to 24,000 blocks is in line with our measurements for the longevity of fingerprint uniqueness, as the likelihood of two random revealing values having the same fingerprint significantly increases, while the table reports only the unique matches.

3.1.4 Further Results

In Heuristic 1 we have used unique value matches. However even in the case where a matching is not unique, the proposed methods still provide a probabilistic linkability feature.

Let us call the probability of a correct value match³ of the unique input to the unique output going through the shielded pool by P. Then to estimate the probability of correct non-unique matches we can divide P by the number of possible pairings. If e.g. our baseline best case is 85.3% probability (assuming 14.7% false positive rate), with possible 3 hiding and 2 revealing values which are all identical, the probability of correct linkage is $\frac{1}{3\cdot 2} \cdot 0.853 = 0.142$. Generalizing this approach, the probability of correct linkage is $\frac{1}{|h| \cdot |r|} \cdot P$, where |h| and |r| is the number of times the exact value in question has appeared as a hiding or a revealing value respectively.

This approach can be directly translated to fingerprints as well, where a lower value of P might be applied as these links provide less accuracy in general.

³The values are called matching when they differ by single fixed z-z hop fee.

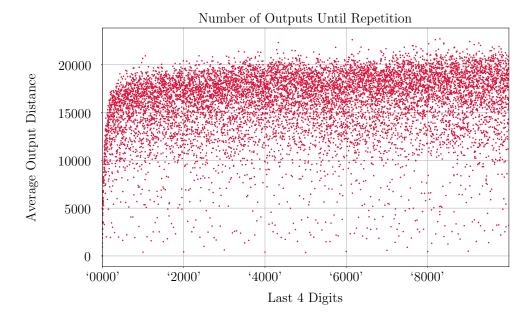


FIGURE 3.1: Average output distance until a fingerprint (last 4 digits of the transaction value) stays unique. The mining related transactions are removed from the dataset.

$ h \cdot r $	Num of links
1	9,919
2	882
3	308
4	344
5	134
6-10	721

Table 3.2: Number of equal in/out *value* pairs (modulo single z-z hop fee) entering and exiting the shielded pool over entire chain history.

Considering these metrics, Tables 3.2 and Table 3.3 summarize the number of links registered based on the possible number of pairs $(|h| \cdot |r|)$. Further exploration of probabilistic matching together with anonymity set sizes derived from the subset sums approach could be a direction for future research.

3.2 The Model for the Probability of Fingerprint Survival

We now present a statistical model which describes the probability that a fingerprint remains unmodified through several shielded transactions. More precisely, from the public and shielded transactions data, we will model the hidden walk of the fingerprinted value through different hops inside the shielded pool (i.e. shielded transactions) before being revealed in a z-to-t transaction. We'll then use this model to compute the survival probability of fingerprints, i.e. the

$ h \cdot r $	Num of links	Non-Unique Complete Matches
1	10,642	7,228
2	5,212	3,513
3	2,192	1,456
4	1,150	738
5	684	418
6-10	1,913	1,227

Table 3.3: Number of possible fingerprint pairs and how many times they happen for a sliding window of 16,000 blocks. In the last column we have removed the matches that were also tagged as unique complete matches in Table 3.2 (the first line of the table contains the unique matches). These unique matches from the previous Table 3.2 did not only remove matches from the unique matches (first line), as even though the removed pair is a unique match with direct value matching, that does not mean it is a unique fingerprint match.

probability that a value that enters the shielded pool exits with its fingerprint unmodified. As shown in Figure 3.1, not all fingerprints are equally likely and hence we cannot talk about the survival probability of any fingerprint: we will then refer only to fingerprints -the good fingerprints- that have an average output distance greater than 10,000, i.e. fingerprints above the y-axis value of 10.000 in Figure 3.1, and we will assume these to be equally likely. The following model is built accordingly.

We denote with $\mathbb{P}(FP)$ the overall survival probability of any good fingerprint going through a path inside the shielded pool and then exiting with a revealing transaction and we let \mathcal{Z} be a discrete random variable which counts the number of transaction-hops⁴ inside the shielded pool over some path. We then denote with $\mathbb{P}(\mathcal{Z}=n)$ the probability that a walk inside the shielded pool goes through exactly n hops before exiting. We further denote with $\mathbb{Z}Paths(n+1)$ the set of paths of length n+1 whose first n hops are inside the shielded pool and a last hop that is an exit z-to-t hop. We can then model the survival probability of good fingerprints as

$$\mathbb{P}(FP) = \sum_{n \ge 0} \mathbb{P}(\mathcal{Z} = n) \cdot \mathbb{P}(FP \mid \mathcal{Z} = n)$$
$$= \sum_{n \ge 0} \mathbb{P}(\mathcal{Z} = n) \cdot \left(\sum_{x \in ZPaths(n+1)} \mathbb{P}(FP \mid x) \cdot \mathbb{P}(x) \right)$$

In order to estimate $\mathbb{P}(FP)$ under this model, we computed the following values:

• Zlen: the average number of hops a path goes through inside the shielded pool before exiting. Assuming each hiding transaction is independent, we set Zlen to be equal to the ratio between the number of z-to-z transactions and the number of hiding transactions. For more details see Section 3.2.3.

⁴It is important to think of transactions as hops, since it is inside the transaction that a fingerprint may be spoiled.

- $\mathbb{P}(\mathcal{Z} = n)$: we modeled this probability using a Poisson distribution of parameter Zlen;
- $\mathbb{P}(FP \mid x)$ for any path $x \in ZPaths(n+1)$: the probability that given the path x the fingerprint survives, i.e. the product of the probabilities that the fingerprint survives in each transaction of x. The per-transaction type survival probabilities are obtained analyzing the fingerprint survival rate in transparent transactions that have the same number of inputs/outputs as the considered shielded and revealing ones;
- $\mathbb{P}(x)$ for any path $x \in ZPaths(n+1)$: the probability that x occurs, i.e. the product of probabilities to have each transaction-type occurring in x. The distribution of the different types of transactions in the three categories t-to-z, z-to-z and z-to-t is directly obtained from the blockchain.

More details on the underlying assumptions and how all these values can be estimated are reported in Section 3.2.1.

Within this model we estimated that the average number of hops a path goes through inside the shielded pool is only Zlen = 1.42 and the survival probability of good fingerprints is $\mathbb{P}(FP) \approx 16.6\%$ (Section 3.2.1).

3.2.1 Experimental results

In order to provide experimental results for the presented stochastic model in Section 3.2, we must gather the relevant statistics from the chain itself. First, we have investigated the Sapling shielded transaction set, as that provides a insight into the usual amount of shielded inputs and outputs for a shielded transaction. Secondly, we observed the public blockchain data to estimate the fingerprint survival probabilities.

3.2.2 Notation

For the ease of exposition, we introduce a notation to describe transactions based on their number of shielded and transparent inputs and outputs. We denote a Zcash transaction as

$$(s_{in} \cdot t_{in} \mid s_{out} \cdot t_{out})$$

where s_{in} and s_{out} are the number of shielded inputs and outputs, respectively, and similarly, t_{in} and t_{out} are the number of transparent inputs and outputs.

Depending on the number of shielded and transparent inputs or outputs, transactions are then divided into hiding t-to-z, fully shielded z-to-z and revealing z-to-t transactions as follows:

- t-t-t-z: $t_{in} \ge 1$ and $s_{out} \ge 1$ and negative balance value (hiding)
- z-to-z: $\mathbf{s}_{in} \geq 1$, $\mathbf{s}_{out} \geq 1$ and $\mathbf{t}_{in} = \mathbf{t}_{out} = 0$
- z-to-t: $s_{in} \ge 1$ and $t_{out} \ge 1$ and positive balance value (revealing)

Some example transactions: (0.1|1.0) is a hiding t-to-z transaction with 1 transparent input and 1 shielded output; (1.0|2.0) is a fully shielded z-to-z transaction with 1 shielded input and 2 shielded outputs; (2.0|1.1) is a z-to-t (partially) revealing transaction with 2 shielded inputs and 1 shielded and 1 transparent output.

3.2.3 Sapling Transactions Dataset

Our dataset of transactions includes all hiding, fully shielded and revealing Sapling transactions occurring from block 419,200 (mined October 29, 2018 when the official Sapling fork happened) to block 472,285 (mined January 29, 2019).

The distribution of these collected transactions is reported in Table 3.4, row *All*. Since we are mainly interested in transactions that more likely occur as coin transfers between non-miner users which are not already linkable with a direct unique value match, we removed the following transactions from our starting database:

- Mining activities: when a block is successfully mined, the miners have to transfer their rewards to a shielded address before being able to spend them, hence creating lots of hiding transactions. Many miners transfer their shielded mining rewards directly to transparent addresses, thus creating many revealing transactions. Using the heuristics from Chapter 2 we have been able to trace 6,827 hiding t-to-z transactions and 5,718 revealing back to miners' public addresses.
- Direct unique value matches: where a hiding transaction is directly followed by a revealing transaction with same unique value (Heuristic 1), i.e. the paths

$$(0.1|1.0) - (1.0|0.1)$$

We have found 37 such transactions.

• 1-Hop fingerprint matches: the paths

$$(0.1|1.0) - (1.0|1.0) - (1.0|0.1)$$

where a hiding transaction with a unique value is directly followed by a revealing transaction with a unique value, where the value difference is equal to 10,000 Zatoshis, the standard transaction fee (Heuristic 2). We have found 676 such transactions.

• Small value transactions: we noticed that while there were 2,436 t-to-z transactions with total transparent value less than 1 ZEC each (for a total hiding value of 841.74 ZEC), there were only 286 z-to-t revealing transactions with an overall value lower than 1 ZEC (for a total value of 68.76 ZEC). Checking the discrepancy between these two numbers, we have found that there were 3 transactions of the form (>10.0|0.1) and 25 transactions of the form

(>10.0|1.1), where >10 indicates that there are more than 10 shielded inputs. The total number of shielded inputs of these 28 z-to-t transactions is 1,756 for an average revealed value of 39.97 ZEC (total revealed value 1119 ZEC). Assuming these 28 transactions were independent, the ratio between the revealed value and the number of shielded inputs is 0.45 ZEC (weighted median) and 0.64 ZEC (weighted arithmetic mean). Since the Zcash official wallet, when combining spends, takes them in decreasing value order, we speculate that these spends are mainly small value spends (less than 1 ZEC) which are collected and eventually combined with few higher value ones⁵ before being transferred to a transparent address. Our speculation is enforced by the fact that in 25 transactions out of 28, the revealed values are round, e.g. 3, 5, 30, 35, 40, 50, 120 ZEC. In other words, there are many small values which are collected and then spent as soon as their total value is close to certain round amount of ZEC. Due to the unusually high number of combined shielded inputs, we suspect that these transactions are related to mining activities (e.g. transparent fractions of mining rewards that are shielded and sent to mining pool members) and can be then traced back to t-to-z transactions whose total value is less than 1 ZEC. Thus we have opted to ignore 2,436 small value t-to-z transactions⁶ and 314 z-to-t transactions.

All these dataset updates are summarized in Table 3.4 resulting after all the removals in 1,613 t-to-z hiding, 1,570 fully shielded and 1,633 revealing transactions. The distribution of different types of transaction in the final dataset is reported in Table 3.5.

Assuming that all t-to-z transactions are independent, we can then approximate the average number of transaction-hops Zlen for a generic path in the shielded pool as the ratio between the sum of all transactions types that have at least 1 shielded input and 1 shielded output and all the t-to-z transactions. We then obtain $Zlen \approx 1.42$.

A path in the shielded pool goes through transaction-hops that have at least one shielded input and one shielded output: in Table 3.6 we reported the most frequent types of these transactions ($\approx 95.8\%$) along with their distribution and the corresponding survival probability of good fingerprints observed in all public blockchain data. More precisely, given a transaction ($\mathbf{s}_{in} \cdot \mathbf{t}_{in} \mid \mathbf{s}_{out} \cdot \mathbf{t}_{out}$) happening in the shielded pool, we set the corresponding survival probability of a good fingerprint to be equal to the fingerprint survival probability observed in the transparent transactions of the same input-output degree ($0 \cdot \mathbf{s}_{in} + \mathbf{t}_{in} \mid 0 \cdot \mathbf{s}_{out} + \mathbf{t}_{out}$) multiplied by the probability that the fee ends with 4 zeroes ($\approx 96.8\%$), i.e. the fee doesn't affect the fingerprint.

Similarly, we report in Table 3.7 the relative distribution and survival fingerprint probabilities for the most frequent exit nodes ($\approx 96.9\%$), i.e. revealing z-to-t transactions.

Given a path $x = x_1 - \dots - x_n - x_{n+1} \in ZPaths(n+1)$, where x_1, \dots, x_n , are transactions inside the shielded pool while x_{n+1} is a revealing transaction,

 $^{^5}$ This could also explain why the revealed value of 1,119 ZEC is greater than 841.74 ZEC, the overall total hidden value of small value t-to-z transactions

⁶It is not statistically relevant which ones are removed, since 99.8% of t-to-z transactions are of the single input-single output form: (0.1|1.0).

	t-to-z	z-to-z	z-to-t
All	11,589	2,246	8,408
Remove Mining	4,762	2,246	2,690
Remove Direct Matches	4,725	2,246	2,653
Remove 1-hop Matches	4,049	1,570	1,977
Remove Small Values	1,613	1,570	1,663

Table 3.4: Number of transactions after each corresponding dataset update.

t-to-	${f z}$	z-to-z z-to-t		$\overline{\mathbf{t}}$	
Type	#	Type	#	Type	#
(0.1 1.0)	1,594	(1.0 1.0)	699	(1.0 0.1)	664
(0·t s·0)	15	(1.0 2.0)	491	(1.0 1.1)	504
(s·t s·0)	4	(2.0 2.0)	176	(2.0 0.1)	224
		(2.0 1.0)	78	(2.0 1.1)	129
		(3.0 1.0)	36	(3.0 0.1)	54
		(3.0 2.0)	21	(3.0 1.1)	36
		(s·0 s·0)	69	(s⋅t 0⋅t)	31
				(s·t s·t)	21
Total	1,613		1,570		1,663

TABLE 3.5: The transaction distribution of our final dataset. The s and t represents all the remaining number of input/outputs possible.

it is now straightforward to estimate both $\mathbb{P}(FP \mid x)$ and $\mathbb{P}(x)$ and, ultimately, $\mathbb{P}(FP)$.

Indeed, assuming each transaction is independent and considering only paths x made by the most frequent types of transactions, we have that $\mathbb{P}(x) = \prod_{i=1}^{n+1} \mathbb{P}(x_i) = \left(\prod_{i=1}^{n+1} p(x_i)\right) \cdot (0.958^n \cdot 0.969)$, where $p(x_i)$ denotes the probability to have a transaction type equal to x_i , i.e. the values in column "%" of Table 3.6 and 3.7. Similarly, $\mathbb{P}(FP \mid x) = \prod_{i=1}^{n+1} p_{FP}(x_i)$, where $p_{FP}(x_i)$ denotes the fingerprint survival probability for transaction x_i , which are again reported in Table 3.6 and 3.7.

Letting $ZPaths'(n+1) \subset ZPaths(n+1)$ be the set of paths consisting of the most frequent transactions types reported in Table 3.6 and 3.7 and modeling $\mathbb{P}(\mathcal{Z}=n) = \frac{Zlen^n \cdot e^{-n}}{n!}$ by a Poisson distribution with parameter Zlen, we obtain an estimation for the overall fingerprint survival probability as:

$$\mathbb{P}(FP) \approx \sum_{n=0}^{k} \frac{Zlen^{n} \cdot e^{-n}}{n!} \cdot \left(\sum_{x \in ZPaths'(n+1)} \mathbb{P}(FP \mid x) \cdot \mathbb{P}(x) \right)$$

Letting k = 5 we obtained $\mathbb{P}(FP) \approx 16.6\%$

Type	#	%	Fingerprint Survival Probability
(1.0 1.0)	699	0.318	0.968
(1.0 1.1)	504	0.230	0.475
(1.0 2.0)	491	0.224	0.475
(2.0 2.0)	176	0.080	0.155
(2.0 1.1)	129	0.059	0.155
(2.0 1.0)	78	0.036	0.204
(3.0 1.0)	36	0.016	0.086
(3.0 1.1)	36	0.016	0.066
(3.0 2.0)	21	0.010	0.066

TABLE 3.6: The relevant z-to-z and z-to-t transactions where at least some coins stay shielded. The last column is the survival probability of a fingerprint based on the public blockchain data.

Type	#	%	Fingerprint Survival Probability
(1.0 0.1)	664	0.412	0.968
(1.0 1.1)	504	0.313	0.337
(2.0 0.1)	224	0.139	0.155
(2.0 1.1)	129	0.080	0.204
(3.0 0.1)	54	0.034	0.086
(3.0 1.1)	36	0.022	0.066

Table 3.7: The relevant z-to-t transactions where some coins are revealed. The last column is the survival probability of a fingerprint based on the public blockchain data.

3.3 Danaan-Gift Attack (Malicious Value Fingerprinting)

Fingerprints (Section 3.1.3) can be used as a tool for linking the hidden and revealed values of shielded transactions. In this section we show that in some scenarios they can be exploited for transaction tagging by an active attacker.

Suppose the attacker is trying to identify the spending of a public address, which converts all its ZECs to hidden addresses regularly. The attacker can transfer a very small but carefully chosen amount of Zatoshis to this specific address, hoping that it leaves the trail of a fingerprint when they are converted from a hidden to a public address. As the attacker sees the current public value on the address, he sends a chosen value such that the resulting sum has a detectable and possibly long-living (Section 3.1.3.1) fingerprint. Of course different fingerprints could be used for different addresses. Afterwards the attacker only has to monitor the revealing transactions for his set of fingerprints. He may also enhance it using the subset sum approach and reason in terms of

anonymity sets rather than unique matches.

Such attack can be performed against entities which accept public donations (e.g. WikiLeaks) since for them receiving money from an unknown source would look less suspicious. Moreover the attacker may monitor the address and resend the fingerprint in case another donation erases his old tag.

Our statistical model can be applied to provide a chance for success against an average user. As we show in Sections 3.2, 3.2.3 the probability for a finger-print to survive if it is revealed in some way is around 16%. This means that the attack succeeds with a 16% chance if the target behaves as an average user. Compared to the low cost⁷ of the attack, we consider it as a real danger for Zcash users. As a countermeasure one should avoid de-shielding unique or rare values and should zero the digits below the transaction fee threshold. In the long run depreciation of the public t-address pool should solve the problem.

3.4 Dust Attack

Sapling shielded transactions reveal their in/out degree, i.e. the exact number of spent and output notes. The output notes would be the unspent transaction outputs (UTXOs) in Bitcoin terminology. This also means that if a user converts all of his/her shielded funds to a public address from a previous output note, it will show in the transaction that there are no new output notes, i.e. all the value from the spent notes are public. Shielded transactions that spend more than 10 outputs are very rare (35 out of 22,249 in the three months of the study period).

Using this information, we discovered an attack that would be able to track one extra hop of shielded spending. If that spending does not have public outputs, we can only verify when the object of the attack used its funds. The attack is the following.

First, the attacker learns the shielded address of the target user either by buying services from a user who has a shielded address or donating to an entity who accepts shielded Zcash. Then the attacker transfers funds in many small valued output notes. This can be done in two ways. The first way is by issuing many separate transactions to a target address either from separate addresses, or from the same address by slicing small values one after another. The second way is by transferring the funds in a single transaction with lots of spend notes. This is not supported directly by the official wallet (it returns an error as it does not allow the same address to appear more than once as an output even in a shielded transaction). However a custom wallet where this check is removed can easily do so. We have tested it on the Zcash testnet and the transaction was accepted.

Later, if the target user wants to spend its shielded coins, it has to create a shielded transaction, which can be monitored on the chain, where the number of spend notes is unusually high. At present, in the official wallets the user can not choose which outputs it wants to spend when creating a transaction. The wallet, instead, orders all available unspent shielded outputs connected to the

⁷Considering that 10,000 Zatoshis are worth 0.7 US Cents, if the exchange rate is 70 USD.

address in a decreasing manner and picks them one-by-one until it has enough value to cover the desired total output value.

This means that even if the user notices this unusual behaviour in his/her wallet, there is currently no way to avoid these dust-spending outputs. As a countermeasure, the user should always do an extra hop inside the shielded pool if his z-address received transactions from external sources. This would consolidate all the outputs into a single spend note, removing the dust-tag. Such a transaction itself would be noticed by the attacker, but no further tracking is possible if there were no public outputs in that transaction. This countermeasure does not require any customization from the wallet as it is a simple shielded transfer, but it is dependent on whether the user notices the attack. We have executed the attack on our own addresses on the Zcash testnet⁸.

Heuristic 4. An attacker tags a target address with more than n > 10 dust outputs. If later he observes that the input-degree of a shielded transaction is at least n, then he links the transaction to the target address.

Let us now investigate how the attack is presented (based on our experiment) to the target in the different Sapling-supporting wallets.

3.4.1 Official Linux Command-line Zcash Wallet

As the official Linux RPC wallet is command line based, the user will only notice this behaviour by specifically checking for it. There are separate commands for getting the total balance for all addresses or a specific one and, similarly, a command for listing the received outputs per address (which includes the already spent ones) and another one to get all currently controlled unspent outputs under all addresses. Thus, it depends on the users whether they check the correct attributes of their wallet and whether they notice the strange dust values. If a user checks for the received transactions, all outputs will show as separate transactions, despite having the same ID.

By listing the received transactions or unspent outputs, one can see the IDs of the transactions and, if the attacker sent all the outputs in the same transaction, this might alert the user.

3.4.2 GUI-based Sapling-supporting Wallets

All the currently existing GUI wallets show the recent transactions on the home page of the wallet. This means that all the dust outputs are shown as separate transactions, even when they were sent in the same shielded transaction. These programs only show the time of the transactions. The user does not see the transaction details (not even the id) in the software, instead it has to copy the transaction to receive the id or go through a link to an online explorer to discover that they are exactly the same transaction. From the time of the transactions it is visible that all of them were made at the same time, which could also be an indicator of malicious behaviour.

 $^{^8} txid:\ 48b364e082f90ae5860ad52a876eae37c84ed0cbb7cf4279dea2fd2a243bacb5$

	15/10/10	15 /01 /10
	15/10/18-	15/01/19-
	29/10/18	29/01/19
Num of Transactions	52,438	41,961
Num of Sprout Txs	7,241	3,592
Hidden Value	129K ZEC	81K ZEC
Without Mining Rewards	25K ZEC	23K ZEC
Num of Sapling Txs	0	4,748
Hidden Value	0	67K ZEC
Without Mining Rewards	0	50K ZEC

Table 3.8: Sapling zk-SNARK usage

3.4.3 Combining Danaan and Dust Attacks

An attacker might combine these two attacks, where some of the dust values also contain a fingerprint, and their sums are fingerprinted as well. This way, even if the dust attack follows only one hop of shielded values, the fingerprint later might still reveal the values when and if the coins are revealed.

3.5 Usage of zk-SNARKs

Let us investigate the usage and adoption rate of Sapling transactions compared to Sprout transactions.

From Table 3.8, we notice that even though the number of transactions decreased, the usage of shielded transactions from regular users mostly switched to Sapling transactions and the main remaining users of Sprout transactions are miners and mining pools who did not change their use practices yet.

3.5.1 Interaction Between Sapling and Sprout Transactions

Another aspect of the adoption rate for Sapling transactions is how many users have transferred their values from a Sprout shielded address to a Sapling shielded address. To investigate this, we have checked how many transaction outputs of a shielded Sprout transaction were spent directly as an input to a Sapling shielded transaction.

We have found 241 such outputs, hidden in 213 transactions overall. Although this might not seem as a huge number of transactions, in total value they cover more than 46K ZEC, which is more than 20% of all Sapling hidden value not related to mining rewards (212K ZEC) since the time of the Sapling hard fork.

Another question is whether these revealing transactions can be tracked by the linking methods presented previously in Section 3.1. The recommended method of coin transfer is described on the Zcash website⁹, where the presence of this linkability is mentioned, warning users and suggesting methods to avoid it by splitting the values into smaller round denominations and moving them with certain delays.

3.6 Summary and Conclusions

In this study we have shown some privacy issues in the Zcash cryptocurrency, mainly utilizing the hiding and revealing transactions that convert coins between the private and public part of the blockchain. We have also shown two novel active attacks - Danaan-gift attack and Dust attack - against Zcash user privacy, and we have provided a theoretical model and a statistical analysis for their success likelihood. We have reinforced the notion that Zcash is not just a blackbox solution, where if a user uses the shielded transactions, it has constant strong privacy. Instead, we show that even the users of Zcash have to put some care into how they use the shielded transactions, and how they transfer coins between the public and private parts of the blockchain.

We would like to provide some general suggestions for the users of Zcash to try to avoid linkage using any of these methods. First, if a user has to pay coins to a shielded address and all of its coins are in public addresses, the user should either shield the coins in multiple transactions, or shield more coins than it needs to transfer to avoid issues in case the receiver of the coins would reveal them in the future. Users should avoid using the default transactions fee. They should also pay attention to the number of outputs they control in shielded outputs, as spending all of them together leaves a visible trace on the blockchain. Finally, it is also safer to hide and shield coins with round values, where at least the last 4 digits of the value should be zero, especially if the economic value of those digits is negligible.

We also note that before the publication of the work we have contacted and provided the work to the Zcash developers as part of our responsible disclosure.

⁹Sapling Turnstile - https://zcash.readthedocs.io/en/latest/rtd_pages/sapling_turnstile.html

Chapter 4

Privacy of Miners in Zcash and Ethereum

In a PoW based blockchain, the most important entity is the group of miners who create the new blocks with the new transactions. Without them, the blockchain would not exist. We have mentioned mining before, and the evolution of both the mining community and mining hardware.

In Bitcoin and several other blockchains most mining uses application-specific integrated circuits (ASICs). On the other hand there are chains that aim to prevent ASICs and use ASIC resistant Proof-of-Work that favor GPU mining. One reason behind this decision is the attempt at making the chain more decentralized. This study mainly focuses on two such chains, namely Ethereum and Zcash. In these chains ASICs have been introduced at the time of this study, and while in Zcash they have become the de facto mining equipment, in Ethereum there is still visible GPU mining. We have introduced Zcash in detail before, but we have not mentioned Ethereum yet. We will not describe it in detail, as that is not needed for the scope of this research, but we will mention its main attributes and differences to Bitcoin.

Ethereum is the second largest blockchain after Bitcoin, but with a different scope and target. It does not use the UTXO transaction structure, instead it utilizes an account balance based one. Furthermore, it provides a more sophisticated scripting system on the blockchain, where even Turing complete code is executable on the chain as a contract. These contracts are called smart contracts, and a user can interact and trigger events in the contract by sending transactions to it. Usually they are used as financial contracts, where some events can trigger different events and payouts to some users, for example auctions or gambling. Smart contracts provide a larger application space for blockchain solutions, that were not possible with Bitcoin. On the other hand, the consensus protocol behind Ethereum is a PoW style protocol, with GPU mining and mining pool dominated space. Thus it lends itself to empirical studies on the mining infrastructure.

In this study we describe a general landscape of miners in these chains using both blockchain and mining pool information. This includes describing the most popular hardware per chain, what the distribution of mining power between miners is, and how GPU mined coins compare to each other in terms of profitability.

Following that, with the introduction of ASICs, questions arose whether hidden ASIC mining existed in the chain before the announcements of such hardware. We have developed methods that might be able to detect hidden ASIC mining based on mining software developer fees and the distribution of the mining power. In the observed chain (Zcash) we could give a bound on hidden ASIC mining (showing no significant hidden ASICs). The same metrics did change drastically after the public introduction of ASIC hardware to the network. We have also studied how the introduction of ASICs effects the decentralization of mining power.

Using the previously acquired knowledge on miners in these chains and the public blockchain information, we describe how an attacker might deduce the individual mining rewards and the mining hardware of a single miner, violating the privacy of miners. We also provide some countermeasures, with which miners might mitigate the leakage of privacy by the choice of a mining pool and the usage of their rewards.

We performed most of the study in May of 2018, with some follow-up measurements in the following months. This chapter is based on joint work [BF19a] with Alex Biryukov.

4.1 Terminology

There is some terminology that we have to introduce first. First, we have already described what a mining pool is. Mining power means the amount of hashes an entity can solve in a fixed amount of time. As mining is a probabilistic protocol, higher mining power means higher likelihood of solving the cryptographic puzzle first. Let us also reiterate how the pools estimate the power of their members and how they pay them. Mining pools do not require their members to solve the full cryptographic puzzle in the PoW protocol, instead the members have to solve a weaker version of the same puzzle. If a member has a solution for this weaker puzzle, it sends that solution to the pool. These weaker solutions are called shares, and they are used to estimate the mining power of the members. In principal, every such share has an intrinsic value in coins compared to the value of reward for mining a block. If one of the shares is a solution to the full cryptographic puzzle, the pool will distribute the found block in the network claiming the block reward. Usually the mining pool takes a fee for operating costs of around 1% from the payouts.

In practice, the members of the mining pool do not have the blockchain downloaded, and they are not responsible for creating the blocks. The blocks and the transactions in them are all first created by the mining pool, and they only send the block header information to the miners, reducing the communication costs. From time to time, the miner will receive a new header to mine on top of. In most mining pools a miner is gathering value in the mining pool by submitting shares, and once this value reaches a preset threshold, the mining pool pays the miner in a payout transaction. This payout transaction usually contains payouts to multiple miners at the same time, to reduce the number of transactions required from the mining pool. How the cumulative value is calculated and when the mining pool pays the miners depends on the pool's payout structure, but these structures are not important for our analysis and thus they are not part of this work.

From the miner's point of view, the miner uses mining machines that are usually referred to as mining rigs or just rigs. These rigs can be ASICs, or from single GPU to purpose built 10-12 GPU computers as well. The miner can group its mining rigs into so-called workers on the mining pool monitoring site. The idea behind this grouping is easier monitoring for the miner that all of its rigs are performing as expected. These workers are configured by the miners themselves, so they can vary in distribution. Some miners have a dedicated worker configured for every mining rig they are running for easy monitoring, but we have seen cases where a miner with huge mining power only has one worker configured on the mining pool.

4.2 Background and Related Work

Mining pools have been receiving attention in recent research, but mostly from a game-theoretic point of view. Eyal et al. [ES18] introduced the selfish miner attack, where colluding miners obtain a revenue larger than their fair share, which proves the Bitcoin protocol is not incentive compatible. For the attack only 1/4 of the mining power is required, compared to the previous 1/2 bound.

Later Eyal in [Eya15] provides a game theoretic analysis on competing mining pools attacking each other by seemingly joining the opposing pool, but not providing any actual Proof-of-Work. The paper calls the decision whether to attack the opponent pool or not the miner's dilemma. Later Tsavary and Eyal extended this work in [TE18].

We have not seen an in-depth analysis of mining pools and miners in the literature and thus we provide it in this chapter. Our analysis is mainly based on two mining pools which have pools in both Ethereum and Zcash. These are the pools run by BitFly called Ethermine for Ethereum and Flypool for Zcash, and Nanopool for both Ethereum and Zcash. We have chosen them because the average and current power of each worker per miner is accessible through their API, if an attacker learns the exact Zcash or Ethereum address of the miner. In order to obtain the list of addresses in case of Ethereum an attacker can scan the chain for recently sent transactions from the pools' main address. We have collected over 100,000 addresses. In case of Zcash, we used the methods shown in section 2 to retrieve the mining payout transactions and build up a database of miner addresses, which consists of $\sim 25,000$ addresses.

4.3 Mining Landscape

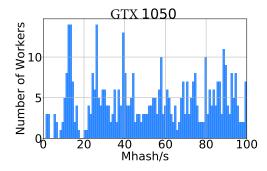
At the time of writing, Ethermine was the largest, while Nanopool was the third largest Ethereum mining pool. Flypool was dominating the Zcash mining power with over 50% hash rate until the introduction of ASICs, and is currently the number five mining pool, while Nanopool was the second largest pool, but is now ninth in rankings.

As these pools had an open API, we were specifically interested in the workers, their momentary and average power, and their names, as we have noticed that miners often name themselves after the hardware itself. Overall we have

acquired detailed worker information for 21,000 miners in Zcash and 52,000 in Ethereum. In these data sets we searched for the keywords of the card numbers or names. If we reduce the workers to ones that have a descriptive name, the dataset is reduced to roughly 10% of all the workers. Combining the average hash rates of these workers with the reported rates for different kinds of hardware, we can attach specific hardware to specific hash rates. Ethereum and Zcash use two very different ASIC resistant Proof-of-Work algorithms called Ethash and Equihash respectively. Their hash rates are measured in Mhashes per second for Ethash and solutions (Sol) per second for Equihash.

4.3.1 Ethereum

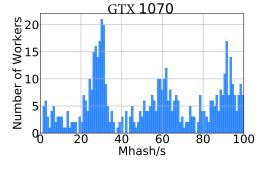
First, let us investigate Ethereum. Notice the periodic peaks in the histograms (Figure 4.1,4.2,4.3,4.4). These peaks represent the number of GPUs in the worker. For example in the case of the GTX 1050 we can distinguish 6 separate peaks at 14, 28, 42, 56, 70 and 84 Mhash/s respectively. Also notice that the peaks are getting wider. This can be partly explained by the deviation getting larger by having more GPUs. On the other hand, as the miners over-or underclock their cards by different amounts, these differences add up and create larger gaps between the configurations.



GTX 1060 you would not be seen as a seen a

FIGURE 4.1: Histogram of hash rates in Ethereum for the GTX 1050 GPU

FIGURE 4.2: Histogram of hash rates in Ethereum for the GTX 1060 GPU



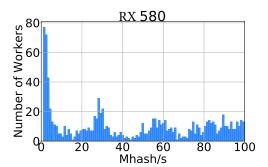


FIGURE 4.3: Histogram of hash rates in Ethereum for the GTX 1070 GPU

FIGURE 4.4: Histogram of hash rate in Ethereum for the RX 580 GPU

Overall the most popular cards are the RX 580, RX 570 and GTX 1060 with around 6,000 workers for each. The exact data and the other cards that are

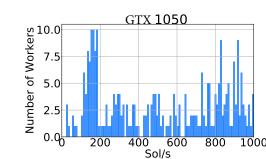
worth mentioning, are presented in Table 4.1. The dominance of AMD based cards is expected, as they are reputed to be better suited for Ethash.

Hardware	Num	Hardware	Num	Hardware	Num
GTX1050	631	GTX1050ti	856	GTX1060	5800
p106	1543	GTX1070	3327	GTX1070ti	616
GTX1080	855	GTX1080ti	1023		
RX460	656	RX470	3713	RX480	1650
RX560	1084	RX570	6068	RX580	6273

Table 4.1: Number of times a card name has been recorded in our chosen Ethereum mining pools

4.3.2 Zcash

In Zcash the most popular card is the GTX 1060 with different configurations, depending on how many of them are in a rig. The periodic peaks are observable here as well (Figure 4.5,4.6,4.7,4.8), although in this case the difference between the hash rate of cards is much larger than in Ethereum. The dominance of NVIDIA GPUs is expected, as they are reputed to be better suited for Equihash.

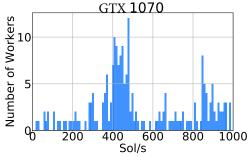


\$\frac{\fin}}}}}}{\frac{

FIGURE 4.5: Histogram of hash rates in Zcash for the GTX 1050 GPU

FIGURE 4.6: Histogram of hash rates in Zcash for the GTX 1060 GPU

GTX 1060





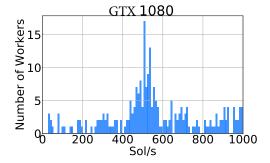


FIGURE 4.8: Histogram of hash rates in Zcash for the GTX 1080 GPU

The larger difference in hash rate between cards provides a better distinction between the different rigs. In Table 4.2 we present the most popular rig configurations and their hash rates, while in Table 4.3 we present the most common cards and the number of times they appear.

Hash rate	Suspected Hardware	Hash rate	Suspected Hardware
150 Sol/s	GTX1050	870 Sol/s	3xGTX1060
270 Sol/s	GTX1060	1100 Sol/s	2xGTX1080
270 501/5	GIAI000	1100 501/8	4xGTX1060
450 Sol/s	450 Sol/s GTX1070		2xGTX1080TI
450 501/8	GIAI070	1400 Sol/s	5xGTX1060
550 Sol/s	GTX1080	1670 Sol/s	3xGTX1080
000 001/8	2xGTX1060	1070 501/8	6xGTX1060
700 Sol/s	GTX1080TI		

Table 4.2: Hash rates and their hardware counterparts

Hardware	Num	Hardware	Num	Hardware	Num
GTX1050	327	GTX1050ti	183	GTX1060	1783
GTX1070	791	GTX1070ti	207	GTX1080	918
GTX1080ti	617	GTX970	157		

Table 4.3: Number of times a card name has been recorded in our chosen Zeash mining pools

4.3.3 GPU Mining

At the time of Zcash's launch, and in the following two years, GPU mining in general became popular. There were multiple blockchains, which used the same hash function for their PoW. One of the more popular hash functions was Equihash, the same one is used in Zcash. Equihash uses multiple parameterized setups, resulting in different difficulties for the hardware, which should also result in different difficulties developing an ASIC for a specific version of the function.

In order to build a more complete picture we investigated Equihash and GPU mining in general, to have a better understanding of the dynamics of mining. First, we built an overview of the total Equihash-based mining ecosystem by adding up hash rates of every major blockchain using the same version of Equihash as its PoW. We show the total Equihash hash rate of these chains in Figure 4.9. We have identified Zcash (ZEC), Bitcoin Gold (BTG), Zencash (ZEN) and Zclassic (ZCL) as the main chains. If a chain doesn't appear on the graph until a certain point in time, it is either because it didn't exist before (Bitcoin Gold), or its hash rate was only marginal (less then 5 MSol/s¹)

¹5 MSol/s is 5,000,000 hashes (Equihash solutions) per second.

compared to current Zcash. The huge increase in Zclassic's power in January and February 2018 is caused by a huge price increase during that time. This price increase was the result of the launch of a new blockchain - Bitcoin Private - at the time, which was forking off of the Zclassic chain. Some of this power temporarily migrated from Ethereum or other GPU-mined coins.

The most notable feature of this graph is the exponential increase in mining power from June-December 2018 which is due to the introduction of ASIC miners in Zcash. 2

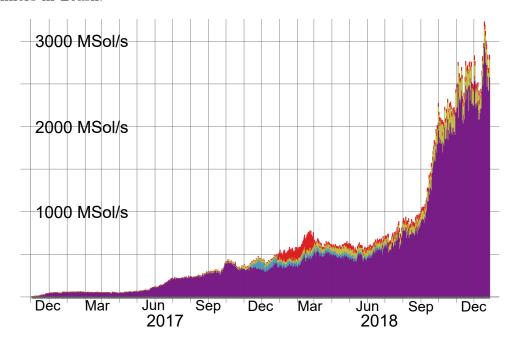


Figure 4.9: Overall Equihash mining power over time (Purple: Zcash, Blue: Bitcoin Gold, Green: Zencash, Red: Zclassic)

One more observation regarding GPU mining is comparing Equihash mining to other ASIC-resistant PoWs. We have chosen Ethereum and Monero³, as they are among the largest GPU mined coins. We compare the hash rate and the profitability of these Blockchains in Figures 4.10-4.11 by converting all rates into Sol/s. This is done by comparing the mining capabilities of the same GPUs on the different PoWs. The interesting observation in this graph is that even when there was a peak in Equihash mining (Oct 2017), compared to the sum there is no visible difference. This is caused by brief miner migration from Ethereum mining to Zcash mining for better profitability.

Figure 4.11 also shows that after the introduction of ASICs in Zcash, the profitability curve has crossed the GPU-profitability line (green line "BASE"), calculated assuming an electricity price of 0.05 USD/kWh. Following graphs confirm that there are probably no GPU miners left in Zcash.

²A small bump in the graph in Oct 2017 can not be explained directly, but is most likely a result of temporary ETH miner migration when ETH difficulty was rapidly increasing due to a difficulty-bomb. It went back to normal when the difficulty bomb was defused by an Ethereum hard-fork.

³By popular belief Monero's CryptoNight algorithm had hidden ASIC mining which was forked off. A large mining power drop after the fork is visible in the graph.

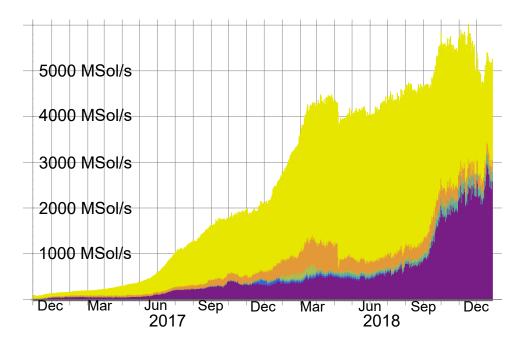


Figure 4.10: Sum of total power of (formerly) GPU-mined blockchains (Purple: Zcash, Dark Blue: ZenCash, Light Blue: Bitcoin Gold, Green: Zclassic, Orange: Monero, Yellow: Ethereum)

4.3.4 GPU vs ASIC mining

Both chains (Zcash and Ethereum) use ASIC resistant proofs-of-work which favor GPU mining in order to keep mining decentralized. This, however, has changed with the recent introduction of ASIC miners for these chains around May-June 2018. The ASIC over GPU efficiency improvement was around 2-5x for Ethereum's Ethash, and 10-30x efficiency improvement for Zcash's Equihash.

4.4 Detecting ASIC miners

On May 3 the ASIC manufacturer Bitmain announced an ASIC miner for Equihash (followed shortly after by Innosilicon). Bitmain is one of the biggest ASIC provider companies in the world, while also having large shares in mining power in Bitcoin, and other large cryptocurrencies. These announcements raise the questions of when did these companies develop an ASIC, and did they launch and test it on Equihash-based cryptocurrencies before the announcement. Were ASIC miners present in the Zcash mining ecosystem before their official shipment dates, and if yes, to what extent? The following two techniques were aimed at trying to answer these questions. First we show our techniques only until June 2018, as the ASIC hardware only started shipping then. Later we show how the metrics we presented changed with ASICs.

4.4.1 Fraction of large miners in the mining power

Using the techniques presented earlier in the thesis (Chapter 2) one can link most of the mining reward transactions. It is also relatively straightforward to

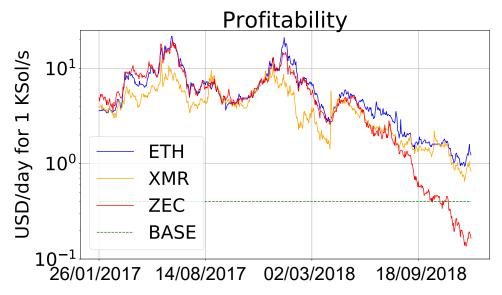


FIGURE 4.11: Profitability in USD of different chains originally using GPU mining. (Red: Zcash, Blue: Ethereum, Orange: Monero, Green: GPU profitability line)

approximate the mining power of an address based on the rewards it gets. First we take the amount of coins an address received in a payout transaction since the last time it received a payout. Then, we check how many blocks were mined between the two payout transaction, and sum up the block reward coins during this time. Finally, we can estimate the average mining power of the blockchain during these blocks based on the puzzle difficulty in every block (as in Zcash, the difficulty is updated after every block). Then, if we take the fraction of the received coins by the miner over the overall mined coins in the same time and multiply that with the average overall mining power, we get an approximate Hash rate for the specific miner.

This approximation can be used to monitor and sum up the power of larger miners in the ecosystem in Figure 4.12, where we define a larger miner as having over 8KSol/s Hash rate. We show this graph with some added information containing the exact fraction of large miners, and the daily exchange rate as well. We used 8Ksol/s as our threshold because the reported Hash rate of the developed ASICs at the time was 10KSol/s.

Analyzing this information, we can see that even though the fraction can change over time, it was quite constant in the last 5 months (Jan-May 2018). Prior to that there is another period of about 5 months (Jul-Nov 2017) when the fraction of large miners was slowly increasing and then suddenly switched from 0.6 to 0.4. The drop is probably explained by miners switching to ZClassic, as that coin had a sudden increase in price due to a fork mentioned earlier. We can not rule out ASICs either, but ASICs should have the opposite effect, as their minimum Hash rate is 10KSol/s.

4.4.2 Mining Software Developer Fees

At the launch of Zcash, there were no ASICs available as the Equihash version implemented in Zcash was considered ASIC resistant. It was designed to be

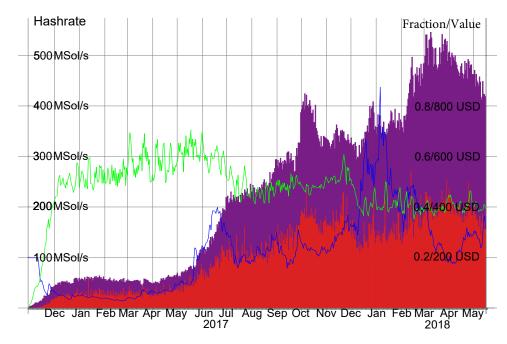


FIGURE 4.12: Estimated portion of large miners with mining power of at least 8KSol/s, with the exact fraction in green and the ZEC/USD exchange rate in blue added

mined with GPUs. As GPUs are not purpose built hardware, instead a general processing unit, the software run by the GPU became just as important as the hardware itself. This lead to a competition among mining software developers to create the most efficient program for Equihash mining. There were some projects that were open-source programs, but the most efficient ones were closed source. This of course meant that these closed source programs became the most popular. However, in contrast with open-source software, the closed source programs had a built-in developer fee, which could not be removed due to their closed nature.

This developer fee (dev-fee) became generally 2% of the mining rewards, and enforced by requiring the miners to use the developers address for mining for 72 seconds every hour (2% of an hour). Even though the developers usually tried to obfuscate this address, we could find the fee addresses of all the major GPU software miners. We found these addresses partly by searching through messaging boards of the community, and partly by reverse engineering the software looking for these specific built-in addresses. Then, we can estimate the mining power of these addresses based on the approach in the previous section, and then multiplying this power by 50 for the 2% rate⁴, which yields an estimate of the mining power for GPU Equihash miners. The interesting and important point is that there were other blockchains that used the same Equihash variant as Zcash, which meant that miners mining these chains used the exact same software as Zcash miners. However, the dev-fee mining was still performed on the Zcash addresses, meaning we have to compare the estimated GPU mining

 $^{^4\}mathrm{At}$ the beginning various miners had different fees at the start from 2-15% but it seems they converged to 2% over time.

power from dev-fees to all Equihash based currencies combined mining power.

We have produced this graph as well (Figure 4.13), where it shows about 80% rate for the first 6 months, and than reduces to about 60%, to later increase again to 80%. With this we can give an estimate that at least 80% of the Equihash mining power till May 2018 was provided by GPUs.

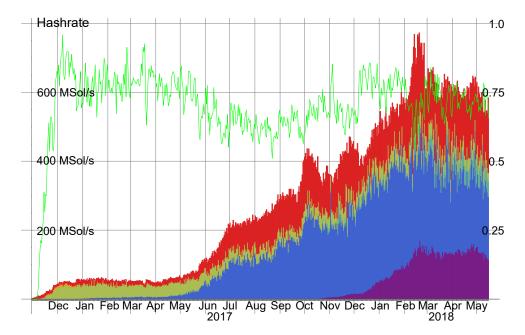


FIGURE 4.13: Lower bound of GPU mining power based on the developer fees (Green: Claymore, Blue: EWBF, Purple: dstm, Light Blue: Bminer, Orange: Optiminer, Red: Remaining Hash rate). The green line presents the covered percentage.

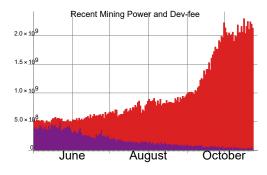
From the month April 2017 there is a big difficulty increase (from about 50 MSol/s to 200 MSol/s) but only a modest dev-fee fraction decrease from 0.8 to 0.6. ASIC or FPGA mining would have a different effect. Explanation could be that large farms learned to disable dev-fee or (more likely) a good free miner has appeared or we did not find some extra dev-fee address. The exponential difficulty increase is most likely caused by the ZEC price hike.

We also see almost linear increase from July 2017 until March 2018 and in the last 3 months a slight linear decrease in Figure 4.9. However secret ASIC or FPGA linearly growing dominance should show as a linear decrease in the dev-fee ratio in Figure 4.13, which we do not observe.

CAVEAT: Dev-fee is an interesting metric, but if it is known to the adversary, it can be cheated by sending a fraction of the ASIC mining results as fees to dev addresses. Also we hope that developers of software miners do not run mining rigs pointed to the same addresses - this is unlikely but can not be completely ruled out.

4.4.3 Public Introduction of ASICs

At the start of June 2018 the first ASIC miners were shipped by the companies Innosilicon and Bitmain, which means the previous two methods can be used to inspect the data when it is known that there are ASICs in the network. As seen in Figure 4.15 the proportion of large miners have visibly increased and below 35KSol/s miners disappeared from the network.



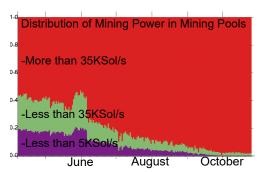


Figure 4.14: The change in projected mining power from dev-fees for the overall Equihash hash rate following Figure 4.13 in time.

FIGURE 4.15: The change in mining power distribution inside the mining pools following Figure 4.13 in time.

The projected GPU mining power from developer fees have also exponentially decreased as seen in Figure 4.14, which is a predictable effect of ASIC dominance over the mining landscape and GPUs becoming unprofitable.

4.5 Mining Centralization

The decreasing projected power from dev-fees and the increasing overall hash rate lead to questions about the decentralization of mining power for Zcash as an effect of the introduction of ASICs. To have a more detailed view on the problem, we have investigated the attributes of mining power in mining pools in more detail in the Zcash and Equihash ecosystem.⁵

If we take a look at the graph of the proportion of large miners, the increase of large miners is obvious, while the actual number of average daily recorded miners that we have recorded went down from around 60,000 in the middle of April to around 13,000 by the end of November 2018 (Figure 4.16). It is visible that miners started leaving in the beginning of June 2018, and that tendency is still observable. This also shows how the introduction of ASICs might alienate GPU miners from a chain, resulting in a more centralized mining infrastructure. The picture has to be taken with a caveat, since theoretically a GPU miner with several mining rigs could point them to different mining addresses, though this is not very likely.

⁵The appearance of a close to 40% mining power in the beginning of June 2018 resulted in questions from the community about who could have such a large mining stake as an unidentified solo miner. Two weeks after its appearance its power was distributed into 2 separate addresses (later to even more addresses, while now it is under one address again), but probably it was controlled by the same entity all the time.

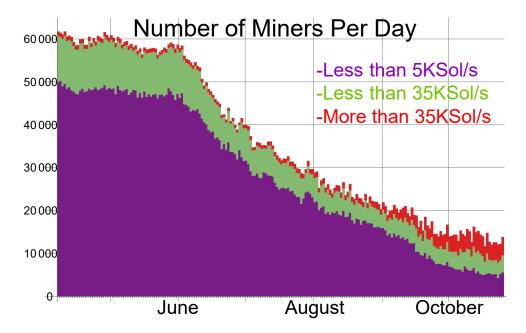


FIGURE 4.16: Number of daily recorded miners in mining pools since April 2018. The number of small miners using probably only one rig has decreased from 50,000 to 5,000, while the middle portion has decreased from 10,000 to 4,000.

4.6 Privacy of Miners

Miners might not want to reveal any information about themselves, especially if they are mining on a blockchain like Zcash, where privacy is the main feature. In that case mining on pools like Flypool or Nanopool is not advisable, as the information is publicly available for anybody who knows the miner's address. On the other hand we can still deduce most of the mining pool payouts from the public blockchain, which leads us to the question what we can learn about a miner if the only thing we see about them are their mining rewards on the blockchain?

4.6.1 Linkability of Mining rewards

Based only on blockchain information, we could record thousands of miners' approximate hash rates. As described earlier, one can estimate the mining power of a miner based on the payouts it receives. Furthermore, the more rewards the miner gets, the more precise our estimate can be. We do this analysis on GPU miners but it can be done for ASIC miners as well. First we have recorded the power of all the miners. The ones under 3.5 KSol/s are displayed in the following histogram, where the width of a column is 5 Sol/s. This graph is not the power of separate addresses, but the sum of every recorded mining power that we found. This means that if a miner had multiple payouts to the same address, then every single payout (after the first one) is an estimation of mining power, and the graph is the histogram of all of these estimates.

It is noticeable that the histogram follows a sum of independent Gaussian distributions, also called the Gaussian mixture model (mining follows a Poisson distribution, but if the distribution's λ parameter is large enough, it can be

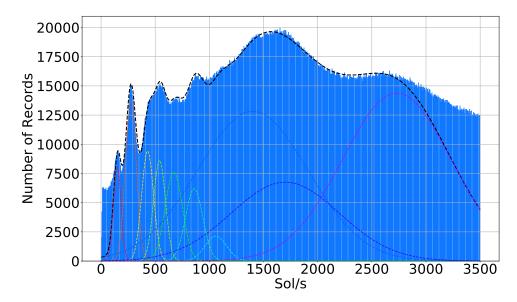


FIGURE 4.17: Histogram of recorded hash rates in Zcash. The striped line is the Gaussian fitting. The separate colours are the single Gaussians.

approximated with a normal distribution). The obvious peaks are at 150, 270 and 550 Sol/s, etc. which correspond to popular mining cards or rigs, so one could manually or algorithmically fit a sum of Gaussians with different weights onto the histogram.

Gaussian decomposition (Figure 4.17) confirms the results we observed earlier about popular cards/rigs in Table 4.2. We consider this as a validation of our metrics, and it means that an attacker might be able to extract more information from the recorded hash rates, if all those rates are connected to the same miner. Based on these values an attacker can observe how stable a miner's hash rate is. If there is a significant increase of power, that stays the same for a long time, one can suspect that new hardware has been added to the miner's farm. As we have a general idea of the possible rigs and hardware, we can reduce the list of possible new hardware (Figure 4.18).

From this information an attacker might deduce at what time a shipment of GPU or ASIC hardware was delivered to the miner. It might relate to geographical relations as well. For example if there is a GPU shortage, there might be information on when different regions receive their new batches of specialized hardware. Correlating it with increases in mining power might reveal information on the geolocation of a miner. Vice-versa, from increases in mining power one might deduce the batch schedules of the hardware manufacturer.

4.6.2 Countermeasures

As the accuracy of an attacker's measurements depends on the regularity of rewards, we suggest using a large reward payment threshold, as the fluctuations in the global hash rate combined with the irregularity of payouts could generate enough noise for the miner's power to be difficult to estimate correctly.



FIGURE 4.18: Example for the case where a miner adds new hardware to its mining rig. In this case the power increases from around 600 Sol/s to around 850-900 Sol/s, which suggests that the miner added a GTX 1060 GPU.

Next, even though instant payouts might sound user-friendly for a mining pool, it results in even more transparency. We also suggest frequent change of mining addresses, limiting the data points for the attacker.

The drawback of constantly using new addresses is that when the miner wants to spend his/her coins, it should not do it in a single transaction from all of the addresses, because if all the addresses are inputs to the same transaction, then an attacker knows that all those addresses are controlled by the same entity. Instead, in the case of Zcash our suggestion is converting the coins to hidden z-addresses. In Ethereum one might consider mining to a smart contract instead of a direct account, and then withdrawing the coins from the contract. This approach would be still visible on the chain, but it is a bit harder to follow for an attacker, as it would involve analyzing the byte-code.

4.7 Summary and Conclusions

In this chapter we have presented an overview of decentralized GPU mining in cryptocurrencies. We have studied the most popular mining hardware, while also investigating the effect of the introduction of ASICs into the mining ecosystem. We have provided methods that could be used to detect hidden ASIC farms in a network, and verified their effectiveness in practice. We have also shown how the overall effect of ASICs and reduced exchange rates damage the decentralization of mining power, leading to a disappearance of over 75% of the miners in Zcash.

Finally we have shown how using only blockchain information an attacker can learn the hash rate of a miner and might even deduce the structure of their mining rigs, reducing their privacy. This study helps to expose privacy vulnerabilities in the current mining ecosystem which is crucial for privacy-preserving currencies and privacy-conscious users.

Chapter 5

Estimating Exchange Traffic

The transaction graph of Bitcoin is constantly changing and evolving. At the start, there were only a handful of transactions in every block, and there were no official markets to exchange or buy coins. Nowadays almost every block in the chain is full with transactions from exchanges to payment channels. This large increase in transaction volume is largely related to the traffic of the exchanges in the blockchain. In order to determine how large portion of transactions are from exchanges, first we have to identify the addresses related to them. To perform this identification, first, we will use some form of grouping or clustering of the addresses. In this study we have used the BlockSci [KML⁺20] toolkit to perform our analytics.

We say an entity controls an address, if it knows the private key to the corresponding public key. Then, we call a group of addresses a cluster, if all of the addresses in the group are controlled by the same entity. A cluster can differ in size starting from a few addresses to up to millions. In fact, the largest cluster we found contains 16,492,872 addresses. Most of the huge clusters are operated by exchanges. The reason for that is the following. Most exchanges provide a unique address for each of their users, furthermore a user can request multiple addresses from the exchange. If a user receives or sends itself coins to these exchange provided addresses, the coins usually do not stay connected to them as an unspent output. Instead, exchanges store the account balances of their users internally in their database, but they move the real coins between their controlled addresses to perform their other transactions. These transactions may include coin withdrawals from users to addresses that are not controlled by the exchange. They can also be large volume trading between different exchanges, or just simply organizational transactions between the exchanges' addresses.

In general there are three main types of addresses that an exchange uses. The first type is the hot wallet, which contains the addresses that are the most active on the blockchain. These addresses perform most of the day-to-day transactions on the chain. Usually it is fairly easy to find them, as they are addresses with thousands of transactions connected to them, which is quite rare in a blockchain. The second type is the cold wallet, which are addresses that contain the exchange's reserves. Usually these addresses have a lot of coins stored on them, but they do not perform many transactions. The third type is the user addresses, where the users can send their coins, but usually exchanges move the funds from these addresses to either of the other two types.

In this research we were interested in the behaviour of the exchanges on the blockchain. To observe this behavior, first, we had to cluster their addresses. There have been multiple approaches to clustering in the literature ([RH11b, AKR+13, LF16, MPJ+13a, SMZ14, CBL17]), but we used the simplest one first proposed by Reid and Harrigan [RH11b], which we deem to be the most efficient.

In this clusterization, if there are multiple addresses as inputs in a Bitcoin transaction, the creator of the transactions has to know the private keys of those addresses (i.e., it has control over those addresses) to generate the transaction. This leads to clustering of addresses per address control, referred to as co-spend or multi-input clustering.

Once we have the co-spends from transactions, the clustering is quite simple using a basic intersecting sets algorithm. If there are multiple transactions with multiple input addresses, and some of these address sets overlap, then all the input addresses in these transactions are controlled by the same entity. Then, with the simple intersecting set algorithm that checks every transaction in Bitcoin that ever happened, one could group and cluster all addresses. The only false positives generated by this algorithm are CoinJoin transactions, which work as a mixing service, trying to hide the real inputs and outputs of a transaction. On the other hand, for our purposes (i.e., exchanges, large clusters), the effect of these transactions on clustering is very low.

We have performed this clustering and decided to investigate large clusters with at least 10,000 addresses in them, as our intuition dictates that exchanges would operate with such large clusters. First, we were interested in what portion of Bitcoin transactions are related to these clusters. Up to block 604,685 (Nov 2019) we have found 912 such clusters. A transaction is related to such a cluster if any of the input or output addresses is part of any of the 912 identified clusters. As the Bitcoin blockchain has changed a lot over time, we have only performed our measurements and studies from block 500,000, which results in an almost two year long period from December of 2017 to November of 2019. In this period, there were 192,933,996 transactions and of them 98,631,677 (51.12%) were directly related to a cluster that had more than 10,000 addresses in it.

The main questions for this study were the following. Are these clusters all exchanges, and is there a way to verify that?

5.1 Evaluating Identified Clusters

The inspected time-span for classifying clusters is from Dec 2017 to Nov 2019, a close to 2 year period (precisely 701 days) from block 500,000 to block 604,685. We have acquired the reported exchange volume per day for Bitcoin for 168 different exchanges for this time. This report, obtained from the CryptoCompare [cco19] service, sums up every Bitcoin exchange pair on these exchanges (e.g., BTC/USD, BTC/ETH).

Then, we tried to identify Bitcoin addresses associated with exchanges, using existing databases, scanning online forums for addresses, and transferring coins to the exchanges ourselves. If we found such an address, we checked whether that address is indeed in one of our large clusters. If there was such a cluster, we hypothetically connected the cluster to the specified exchange. Note, that in

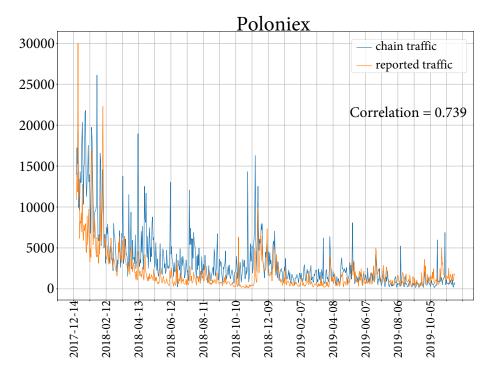


FIGURE 5.1: Poloniex's reported and chain traffic for the entire 701 day period.

some cases, we found multiple clusters for the same exchange. Overall we have found probable clusters for 32 different exchanges, which include the well-known (e.g., Bittrex, Binance) and smaller exchanges (e.g., LocalBitcoins, Mercado Bitcoin).

Using these clusters, we compared the daily exchange traffic reported by the exchange with the actual coin traffic on the chain itself. In the rest of the study we will refer to the coin traffic that happens on the chain itself as chain traffic and to the one that we acquired from CryptoCompare as reported traffic. The chain traffic is the sum of every in and outgoing coin from the entire cluster per day, i.e., if a coin is moving between addresses of the same cluster, it is not counted. The comparison between the reported and the chain traffic involved multiple approaches to find out which one is the best performing for pairing the two different traffics. Our first approach was a simple correlation statistic between the two datasets, i.e., the daily reported traffic of an exchange and the daily sum of in and outgoing bitcoins for the respective cluster (or in a few cases clusters). First, we compared the identified clusters to their respective reported traffic over the entire 701 days period. A specific example is Poloniex (Figure 5.1), which shows a correlation of 0.739 over the entire 701 day period. We found that only 4 showed (Table 5.1) any meaningful correlation between the reported traffic and the chain traffic, where we claim meaningful correlation over 0.6.

The next highest correlation value is 0.575 with the Vaultoro exchange. The meaningful correlation of these four exchanges leads us to a question: Is it possible to pair an unidentified cluster to a specific exchange by this method? To examine this question, we first performed a correlation statistic between all 32 reported traffics, to examine the uniqueness of data for these exchanges.

Exchange	Correlation
Bitstamp	0.771
Poloniex	0.739
BitTrex	0.701
Kraken	0.685

Table 5.1: The four highest correlation values between the reported and observed on the chain data over 701 days

What we have seen is that the volume of reported exchange traffic can change similarly across multiple exchanges. The trends of the market are the main driving force behind these similarities in traffic changes. The result of these trends is that plenty of datasets have a strong correlation with each other (Table 5.2).

	Poloniex	BitTrex	Binance	Huobi
Poloniex	1	0.956	0.222	-0.040
BitTrex	0.956	1	0.223	-0.024
Binance	0.222	0.223	1	0.724
Huobi	-0.040	-0.024	0.724	1

Table 5.2: The correlation values between the reported daily exchange volume between a selected set of exchanges

As one can see, there are multiple different behaviors visible. There are cases, like Poloniex and BitTrex, where their daily trading volumes have a solid correlation. On the other hand, the opposite can happen as well, for example, Poloniex and Huobi, where there is no correlation. Finally, we also have similar correlation values to what we have seen in Table 5.1, like between Huobi and Binance. We note that there is a clear distinction between western and eastern-based exchanges.

The result of these trends is that if we try to correlate the reported traffic with the observed chain traffic for every identified cluster, in some cases, the highest correlation pair is not the identified exchange-chain traffic pair. If we look at the 4 exchanges in Table 5.1, then the actual best match based on correlation might be a different exchange (Table 5.3).

Exchange	Best Match	Correlation
Bitstamp	Bitstamp	0.771
Poloniex	Korbit	0.789
BitTrex	Korbit	0.726
Kraken	Luno	0.718

Table 5.3: Exchanges from Table 5.1 and the highest correlation pair among all inspected exchanges

We have noticed, that there are high correlation matches where the actual volumes are very different, which leads us to the introduction of a second metric for comparison. We compare the value of the two datasets, as the correlation metric only measures the similarities in the change of value between two sets of data. We calculate this new metric as follows: for every daily value, we take the absolute difference of the reported and the chain traffic, divide that by the reported volume and multiply it by 100 for easier readability. When we have this new value every day, we simply take the median of them. This metric represents the median fractional difference between the reported and the chain traffic, and we will refer to this value as MFD for the rest of the study:

MFD =
$$Median\left(\frac{|R_i - C_i|}{R_i} \cdot 100, i = 1, \dots, n\right)$$

where n is the number of observed days, R_i and C_i are the reported and chain traffic on the ith day. We chose the reported traffic as the fixed divisor, because it is not dependent on the accuracy of clusterings and chain measurements, and provides a stable point of comparison.

With this new metric, we have tried multiple heuristics to see what would result in the highest number of best matchings, where the best match is the real world pair. We have divided the approaches into two separate main categories. The first one chooses the best correlation first while the second the smallest MFD. Then we use the other method to make the final selection. For example, we take the chain traffic of an exchange and compute the correlation with all other exchange traffic data. Then we choose the highest value, and take that and every other pairing that is close to it (i.e., less than 0.1 away). Afterwards, we calculate the MFD for these pairs and choose the pair with the lowest MFD as the best match.

The other way would be to choose the lowest MFD first, and if there are multiple ones around the same value, then we choose the highest correlation as our final pairing. Of course, none of these methods will be perfect, but they can sometimes provide better results than others. On the other hand, we could still only match six exchanges perfectly.

Our heuristics' last refinement was the realization that some clusters that we identified did not stay active for the entire 701 day period. This inactivity lead us to the method where instead of finding the best pairing in the entire 701 day observed time, we looked at shorter continuous periods with the previously described two methods. We have tested our datasets with 31, 61, 91, 121, 182, 241, and 361 day long intervals, and we were again interested which ones would result in correct pairings between the reported and the chain exchange traffic.

After testing the previously mentioned techniques on these time intervals, we have found that correct pairings came from the approach of first finding the highest correlation value per pairing with the chosen interval length, then keeping all pairs that are less than 0.1 away from this best correlation pair, and then choosing the one with the lowest MFD as the best matching (Heuristic 4).

Interestingly, most correct pairings resulted from the 61 day long intervals, more than when we observed the entire 701-day long interval. Our explanation of why this time-span is best, is that less then 60 days introduces too many other exchanges for close comparison. The noise and inaccuracy of the data leave multiple exchanges too similar to each other. On the other hand, on a

Algorithm 4 Heuristic for finding the best Exchange pair to a cluster C over n day long intervals

```
procedure MATCHEXCHANGE(C,n)
   CorrPerExchange = \emptyset
   for E \in Exchanges do
      CorrPerExchange \leftarrow BestCorr((C, E, n))
   end for
   BestCorr = Max(CorrPerExchange)
   Selection = \emptyset
   for E \in \text{CorrPerExchange do}
      if |\text{BestCorr} - E| < 0.1 then
          Selection \leftarrow E
      end if
   end for
   BestMatch = MinMFD(Selection)
   return BestMatch
end procedure
procedure BestCorr(C,E,n)
   MaxCorr = -1, MaxStartDay = 0
   for (i, i+n) \in C, E do
      Corr = Correlation(C_{(i,i+n)}, E_{(i,i+n)})
      if Corr > MaxCorr then
          MaxCorr = Corr, MaxStartDay = i
      end if
   end for
   return MaxCorr, MaxStartDay
end procedure
```

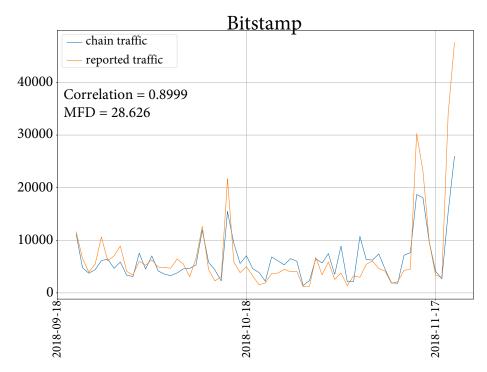


FIGURE 5.2: Bitstamp's reported and chain traffic for the best matching 61 day period

longer time-span, the likelihood of the exchange using another cluster increases, which results in this 2 month-long period.

Figures 5.2, 5.3 and 5.4 are some examples for correct matches using 61 day long intervals. Overall there were 8 perfect matches, and 2 matches where the real pair was the second or third best match (Bitstamp, Kraken, Kuna, Liqui, LocalBitcoins, Poloniex, The Rock Trading, Vaultoro and for second or third Huobi and Mercado Bitcoin).

The only similar result was when the inspected period was 241 days. In that case, there were seven perfect matches and another four top 3 matches, but out of those eleven, only five were new matches (removed were four exchanges: Huobi, Liqui, Vaultoro, TheRockTrading; the five new exchanges were: ZB, CoinBene, Luno, FCoin and Binance).

5.2 Classifying Large Clusters

As we have mentioned earlier, our definition of a large cluster is a group of addresses where the same entity controls the addresses, and the number of addresses in the cluster is at least 10,000. Utilizing the previously described heuristic (Heuristic 4), one could try to identify which of these clusters are exchange clusters and which ones are not. In the history of Bitcoin up to block 604,685, there have been 912 such clusters.

Some of these clusters became inactive even before December 2017. Of these 912 clusters, only 698 have been active during the period between block 500,000 and 604,685, but we need active clusters for at least 31, 61, or 241 days in a row. There are, in fact, 523, 462, and 292 such clusters, respectively. We chose

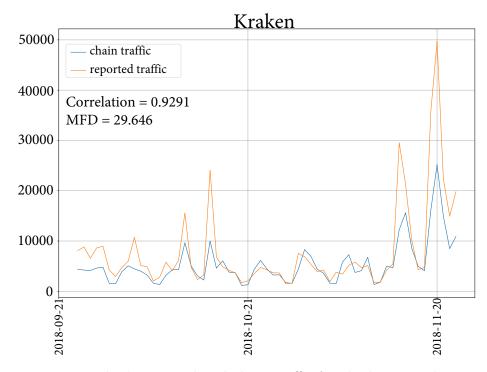


Figure 5.3: Kraken's reported and chain traffic for the best matching 61 day period

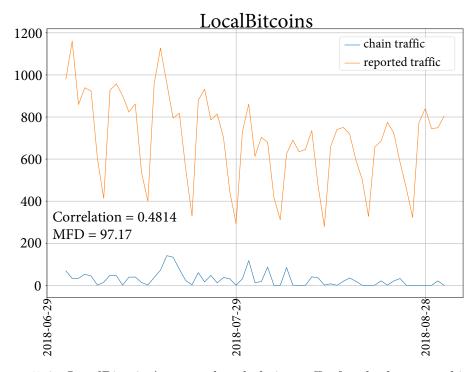


Figure 5.4: Local Bitcoins' reported and chain traffic for the best matching 61 day period

the 61 and 241 day intervals for their accuracy in correct pairings. We selected 31 days for generally having the highest correlation and lowest MFD values, as even if the interval does not provide the most correct pairings, it is a good indicator for the attributes of a cluster.

The question is, of these 523 clusters how many of them are exchanges. Our best answer is dependent on the correctness of the pairing. If the correlation is over 0.8 and the MFD is less than 40%, we consider it a confident guess that the cluster is indeed an exchange. On the opposite end as well, if no pair is over 0.3 correlation, we conclude with confidence that the cluster is not an exchange. Clusters can be a good match with over a 0.6 correlation, but the area between 0.3 and 0.6 is the trickiest. In our previous observations, we have seen cases where the exchange paired with itself as the best pairing, but the correlation was only barely over 0.5. On the other hand, no other pairing was close to it in terms of correlation.

Table 5.4 shows the distribution of these large active clusters. The first row shows the number of clusters active for at least as many days in a row as the column indicates. The second row shows the number of clusters paired by having their thresholds at 0.8 correlation and 40% MFD. The third row shows clusters that we paired with a correlation value between 0.6 and 0.8 and an MFD with less than 60%. The fourth row is correlation values between 0.4 and 0.6, while the MFD is less than 80%. The last row shows the number of clusters with a maximum MFD of 99% (to limit pairings where the size of volumes have nothing in common), and the best correlation pair is less than 0.3, or if there is no pairing where the MFD is less than 99%.

	31 days	61 days	241 days
Active Clusters	523	462	292
Over 0.8	95	34	6
0.6-0.8	172	143	50
0.4-0.6	152	152	109
Under 0.3	17	31	38

Table 5.4: The categorization of large clusters based on different time spans

The reason for the steep decrease in good matches can be explained by our observation of confirmed exchanges. There we have seen similar behavior, as the shorter observed time frames resulted in higher correlation values and matching.

5.3 Summary and Conclusions

Again, we have to mention that the goal of these matchings is not to find precisely which cluster is which exchange. We have shown that that can easily lead to mismatches. Furthermore, there are multiple cases where an exchange might use multiple clusters at the same time. Our goal is to identify which large clusters are operated by exchanges in general. In our opinion clusters in the second and third row in Table 5.4 are operated by exchanges with a high likelihood, and similarly, any cluster in the last row is likely not an exchange.

The most difficult pairings to judge are the ones in the middle territory, but with this approach saying anything more descriptive about them is difficult.

On the other hand, as we have seen with certain exchanges, this approach is another way to try to confirm the reported traffic of an exchange. Surprisingly, even if one assumes that most of the transactions in the reported traffic happen only on the exchange (as they do not need to happen on the chain, the exchange just makes the transfer on its internal database), they are comparable in terms of volume. How these volumes - both the reported and the chain traffic - change also follows similar patterns, which is the attribute we try to represent with the correlation metric. Of course, these values can be manipulated by an exchange by doing fake transactions on the chain, and it is hard to identify that this behavior is indeed happening. Nevertheless, even in cases where the volume was mostly different, and the best correlation was not convincing (LocalBitcoins, Figure 5.4), it still provided the best match with itself. We are not claiming that the reported exchange volume of e.g., LocalBitcoins is fake, as we could have missed other clusters that are controlled by them, but it is an interesting clue for further study.

Part II Consensus Protocols in Blockchains

Chapter 6

ReCon

Distributed consensus protocols, where several equal nodes establish an agreement on a sequence of operations, have been known since at least the 1980s with the appearance of the first distributed databases. Over time, protocols that tolerate faulty nodes (FT protocols, for fault tolerance [Lam98, OO14]) and later the ones that tolerate malicious nodes (BFT protocols, for Byzantine fault tolerance [LSP82]) were developed [CL99, AMQ13]. However, their application was limited as such databases have been typically constrained to a single enterprise, which can use a trusted leader to facilitate the agreement.

The Byzantine Agreement protocols tolerate up to one third of all nodes being malicious. This is satisfactory for a private system, but does not work when we design a public system with free membership. The situation changed drastically with the introduction of Bitcoin [Nak09], which revolutionized consensus protocols by using the *Proof-of-Work* concept (PoW). A Bitcoin node solves a computationally hard problem to decide which operations (transactions) to apply. The Proof-of-Work consensus tolerates malicious nodes as long as they constitute no more than 51% of the computational power or as some more conservative analysis estimates 25% of the computational power [NKMS16]. There are drawbacks to using PoW protocols as well: the two most used protocols, Bitcoin and Ethereum [Woo14], support up to 10 transactions per second at most, which is a great difference to thousands of transactions per second achieved in regular Byzantine Agreement protocols such as Tendermint [Buc16] or in private networks [vis15]; the electricity needed to perform cryptocurrency mining is also reaching new heights, as now it is comparable to the consumption of Ireland [de 18]; transaction confirmation time may take up to an hour (although it is still faster than wire transfer across continents).

Another crucial issue for open consensus protocols is their vulnerability to Sybil attacks. If there is no cost to join a network, that network will always be prone to classical Sybil attacks. There are multiple protocols that provide safety in an open network against this kind of attack. For example, in PoW this resistance is provided by the cost of mining. In Proof-of-Stake style systems the adversary needs to obtain a large enough stake in the currency to perform such attacks. In general all the different Proof-of-X style protocols require a proof in order to prevent Sybil attacks. We provide a new protocol that leverages node reputation in order to enhance Sybil-resistance of consensus protocols.

Reputation systems are abundant in our society from online auctions and marketplaces like eBay, Amazon, credit ratings like Standard & Poor's, Fitch, Moody's to social networks and even academic citations. They are forgeable

to different degrees, but in many cases it takes long time and effort to earn reputation and often there is a monetary value associated with it. In some cases monetary stakes can be used directly for ranking or reputation. One may also assume that reputation correlates with honest behavior if the setting can detect and punish malicious behavior.

We design a proof-of-concept protocol which we call ReCon (Reputation Consensus)— a protocol which can bootstrap from some hard to forge reputation source¹, suggests committee members based on their reputation, and this reputation is then updated over the rounds of the consensus protocol. The protocol is built on top of a blockchain, where multiple attributes of blockchains are utilized. In a nutshell, ReCon selects the committee in a randomized way to enable diversity and increase the cost of a Sybil attack, slightly increases the committee member's reputation if a round succeeds and significantly reduces the reputation if a round fails. Furthermore, the reward mechanism is built in such way that it stops giving rewards after a certain number of successful consensus rounds - namely if all goes well, keeping the status quo is the best. This is not fair to the newly joined honest nodes, but also keeps any Sybil from increasing the reputation of its nodes. The eventual reputation inversely correlates with maliciousness. We show that this method prevents not only a simple adversarial strategy when malicious nodes always try to disrupt the protocol, but also a smarter one, when such nodes act only if they constitute 1/3 of the committee or even only after 2/3. Since ReCon quickly penalizes the reputation rating of the validators which couldn't reach consensus, the best adversarial strategy is to wait for a supermajority of 2/3 of committee nodes. Our scheme is thus secure in the environments where these situations are ruled out, e.g., when the chance of obtaining 2/3 majority in any reputation-selected committee is very low (controlled by security parameter λ). As a module, Re-Con can be laid on top of many existing distributed consensus protocols such as PBFT [CL99], HoneyBadgerBFT [MXC+16], or Zyzzyva [KAD+09] opening them to a larger pool of permissionless validators.

Even though our protocol doesn't solve all the problems with the classical PoW chain, it does provide an alternative which drastically reduces the cost of entry to the network and reduces the energy waste, while still providing strong resistance against Sybil attacks.

We have implemented a proof-of-concept simulator of ReCon, which takes the number of nodes, the committee size, the initial reputation and the prior "maliciousness" probability as inputs and returns the reputation distribution and the posterior probability of being malicious. We also examine the case when external reputation is not available (this could be of interest if the Sybil attacker is not present at the early stages of the protocol - allowing the honest users time to acquire reputation). Experiments prove that our approach compares favourably to the other methods of detecting maliciousness by sample testing and maximum likelihood. Computational complexity of the reputation module itself is constant in each round and for each protocol node (update of reputations for a constant number of validators, typically - 100).

 $^{^{1}}$ Maintaining such source of Sybil-resistant reputation is a separate interesting research topic.

The chapter consists of the following sections. First we describe the current state of the art in consensus protocols that either use some form of reputation or use blockchains to provide an open peer-to-peer consensus protocol (Sections 6.0.1, 6.1). Afterwards we present the preliminary requirements and assumptions of our protocol (Section 6.2). We then provide the detailed description of the protocol itself (Section 6.3), followed by our proof-of-concept simulation results (Section 6.4). Finally we consider the different attacks against the protocol based on our assumptions (Section 6.5). This chapter is based on joint work [BF20] with Alex Biryukov and earlier collaboration with Dmitry Khovratovich.

6.0.1 Related Work

The literature on the reputation systems is vast and is beyond the scope of this chapter. All webpage ranking systems, for example, fall into this category with PageRank [PBMW99] being a classical example. An interested reader is referred to flow-based reputation systems adapted for P2P networks (Eigen-Trust [KSG03]), subjective logic-based schemes [LLJ+11], or privacy-preserving designs, both coin-based [ACBM08] and not [LMAG+15]. There are also works considering a rational entity in a Byzantine Agreement protocol like the BAR Primer [CLN+08], which can be represented as a reputation protocol as well. Our work has a more narrow focus as we do not consider individual ratings but work on the meta-protocol level by analyzing global events – consensus decisions only. This allows sophisticated methods to apply easily in the decentralized way.

6.1 Existing Consensus Protocols for the Blockchain Environment

There is a vast and quickly evolving body of work on consensus protocols for open peer-to-peer protocols that use the blockchain primitives to achieve their goals. Let us present a few of these protocols to show the trends in this research area.

6.1.1 Proof-of-Work

We have already presented the PoW protocol in detail (Section 1.2.3), but let us reiterate its most important features. The idea for intensive computations as a countermeasure against spam was first proposed by Dwork and Naor in [DN92] and denial of service (DoS) protection in the form of a TLS client puzzle by Dean and Stubblefield [DS01]. The amount of work is certified by a proof, thus called *Proof-of-Work*. Perhaps the simplest scheme is Hashcash [Bac02], which requires a hash function output to have certain number of leading zeros and are adapted within the Bitcoin cryptocurrency. To avoid centralization, the PoW finding process (mining) must be stochastic, where the probability of the proof generation at any given time is non-zero and independent of previous events. Therefore, mining must be close to a Poisson process with the number of proofs

found in a given timeframe following the Poisson distribution and the running time of the algorithm A_R following the exponential distribution:

$$T(\mathcal{A}_R) \sim \text{Exponential}(\lambda(R)).$$

The Poisson process is often emulated by the difficulty filter: a fixed-time algorithm \mathcal{B} , which additionally takes some nonce N as input, is concatenated with a hash function G, whose output should have a certain number of trailing zeros. In this case, the algorithm \mathcal{B} must also be amortization-free, i.e. producing q outputs for \mathcal{B} should be q times as expensive. Such PoW-based consensus protocols have a completely asynchronous structure and is widely used in cryptocurrencies, although they are relatively slow and wasteful. PoW is secure until the malicious entity has at least 51% mining power, although there are more recent attacks, that would reduce this to 25% [NKMS16]. The PoW-based protocols are also permissionless, which makes them a very good fit for the cryptocurrency environment.

6.1.1.1 Variants of Proof-of-Work

The example of Bitcoin has shown that PoWs based on standard cryptographic hash functions are prone to mining centralization due to a mining hardware arms race. One direction is to make memory-hard PoWs which would provide ASIC resistance, and thus would reduce mining centralization [PJ16]. Such PoWs are used in Ethereum [Woo14], Zcash [BK16], and several other cryptocurrencies (Litecoin and its forks use Scrypt, Monero uses CryptoNight, etc.). There are also some recent theoretical results on memory-hardness of scrypt-like constructions [ACP+17, BCS16].

6.1.1.2 From chain to DAG

There are existing PoW algorithms where instead of a chain the main structure of blocks form a DAG (directed acyclic graph), e.g. the Ghost protocol [SZ13]. The Ghost protocol is a simplified DAG, and it is implemented in Ethereum. Ghost introduces uncles apart from direct father blocks, meaning that the newest block does not have to be directly chained from the last block, but it can be chained from one of the previous relatives as well. On the other hand, the distance between a new block and its uncles is restricted, thus limiting the complexity of the DAG.

6.1.1.3 Variants of Proof Protocols

To reduce the energy waste of the PoW protocol, some works proposed alternatives to the wasteful mining, but kept the main concept of a stochastic process to determine the creator of the next block. One of these concepts is the Proof-of-Space [DFKP15] protocol, where the verifier has to prove the usage of large storage, ex. dedicated hard disk space. Other similar variants are Proof-of-Burn [KKZ20], where the verifier has to prove the burn of some amount of

currency with transferring funds to a verifiably unspendable address. Yet another idea is the Proof-of-Bandwidth protocol proposed to incentivize donating bandwidth to the Tor network.

And yet another recent concept is Intel's Proof of Elapsed Time consensus [int17], based on the tamper resistance of SGX enclaves which restores the one-CPU one-vote principle. The drawback however is that such a consensus mechanism requires trust in a single hardware manufacturer.

6.1.2 Proof-of-Stake

Another actively developed consensus protocol concept is Proof-of-Stake (PoS), where the creation of the newest block is not dependent on the hashing power, but on the amount of currency, or in other words stake possessed by an entity. The next block validator is randomly chosen, where the likelihood of being elected is equal to the ratio of the owned and overall existing currency. It is also a permissionless protocol, and removes the computational waste of PoW. On the other hand it introduces new vulnerabilities, such as the tragedy of commons or the nothing-at-stake attack, where a node can participate in every fork on the chain as there is no penalty for such behavior, compared to the computational waste in PoW. A PoS protocol is secure when the malicious entity controls less then 50% of the stake.

The best example for PoS algorithms is the Ouroboros family of provably secure PoS protocols [KRDO17, DGKR18, BGK⁺18].

6.1.3 Byzantine Agreement

The classic case of byzantine agreement (BA) was first introduced in [LSP82]. Suppose there is a decision committee, in which the members vote which transactions they accept and send their votes to everybody in the committee, while having some cryptographic proof (MACs or digital signatures) of the validity of their messages. BA protocols are permissioned, as the participants need to know the public key or the MAC key of all the members. They are generally secure if less then 1/3 of the members are malicious. Making them permissionless would leave them vulnerable to Sybil attacks.

6.1.3.1 PBFT

The first practical implementation of a BA protocol in a weakly synchronous network was PBFT [CL99]. It uses a leader, who selects a set of transactions to be accepted. If the leader acts maliciously, a leader change can be triggered with more than 2/3 of the committee members participating. The strength of this construction is its throughput, as it is capable of thousands of transactions per second, but it requires some weak or eventual synchrony, and it is efficient only with a small number of nodes (up to 100).

6.1.3.2 HoneyBadger

HoneyBadger [MXC⁺16] is a probabilistic asynchronous BA protocol, optimized for larger committees. It has to be probabilistic, as it is well known that deterministic asynchronous BA protocols are impossible. It is implemented for 104 nodes, and has better performance than PBFT.

6.1.3.3 Federated Byzantine Agreement

Federated Byzantine Agreement protocols are similar to regular BA protocols, but instead of verification from 2/3 of all the nodes, a node has a quorum slice of trusted nodes, and only accepts verification from them. The protocol requirement is that there needs to be a route of overlapping quorum slices between any two quorum slice, otherwise there could be disjunctive agreements. This structure can create larger committees than the regular BA protocols and leads to a permissionless structure. Examples of this are the Stellar [Maz15] and Ripple [CM18] consensus protocols.

6.1.4 Hybrid Protocols

Hybrid protocols are a combination of the previously mentioned type of protocols, trying to utilize the advantages of them while mitigating their drawbacks. Such protocols try to inherit from PoW and PoS the permissionlessness and asynchrony, while from BA protocols the large throughput.

6.1.4.1 Bitcoin-NG

The Bitcoin-NG protocol [EGSvR16] uses two different block types to achieve more throughput. The first type is keyblocks, which are generated by the miners. They are made once per time epoch, and the miner of the keyblock becomes the new leader in the protocol. Once a leader is elected, it starts creating microblocks without the need of a Proof-of-Work. These microblocks contain the candidate transactions the leader deems acceptable until a new key block is generated. This leads to much shorter transaction confirmation times, and increases the throughput.

6.1.4.2 ByzCoin

Byzcoin [KJG⁺16] tries to improve on the previous idea of Bitcoin-NG. Instead of choosing a single leader to create the microblocks, it uses a committee of the most recent keyblock solvers. Thus in general the miners get voting power in return for their mined blocks. The decision on which transactions to include is decided by a BA protocol, but with the technique of collective signing [STV⁺16], instead of a round-robin style of communication between every node (like in PBFT), therefore Byzcoin would not require direct communication between all the nodes in a committee.

6.1.4.3 Elastico

Elastico [LNZ⁺16] is similar to Byzcoin in terms of a PoW deciding who are the members of a BA committee, but in this case it is used for randomly distributing all the nodes in the network into smaller subcommittees (sharding). Every participating node has to solve the PoW, and the individual solutions determine which separate subcommittee a node is put into. This way the malicious participants can not force the protocol to put them in the same subcommittee. The results of the subcommittees are combined by a final committee. With Elastico the confirmation time is similar to Bitcoin, but the throughput of the protocol is much higher as the sharding property leads to a scalable algorithm, where the number of nodes is proportional to the throughput of transactions.

6.1.4.4 Proof-of-Activity

Proof-of-Activity [BLMR14] is a combination of a PoW and a PoS protocol. It uses a PoW solution of an empty block header as the seed for the next stakeholder election. Then based on this seed N stakeholders are chosen to sign the empty blockheader, and the last signer fills the block with transactions, and broadcasts it as the newest block.

6.1.4.5 Algorand

Algorand [GHM⁺17] expands on the idea of Byzcoin, where based on some function a committee is chosen that confirms transactions using a BA protocol. Instead of a PoW puzzle, it uses a PoS approach with verifiable random functions (VRFs) [MRV99] to choose the members of the next committee, where the likelihood of being elected is based on the amount of stake controlled.

6.1.4.6 RepuCoin

RepuCoin [YKDV19] is an attempt to improve upon the ByzCoin protocol, where the protocol not only utilizes the raw mining power of the miners to determine the next committee, instead it uses the reputations of the miners as well, where long-term honest miners get preferred over other powerful but much newer miners, which can result in honest operation in a network where the attacker has more than 50% of the mining power.

6.2 Preliminaries of Our Protocol

6.2.1 Generic

We work in the following model. The network is composed of N public *nodes*, which maintain a consistent state by applying transactions in the same order. Transactions are supplied to the network by clients in a pre-specified format, but we do not make any assumptions on their size or structure, nor on the number of clients and their connectivity.

Each node is an equal participant in a consensus protocol, which specifies the action sequence so that eventually the nodes agree on the transaction order (safety) and every valid transaction is accepted at some point (liveness). A protocol is called Byzantine fault tolerant (BFT) if it provides safety and liveness despite some malicious nodes violating the protocol secretly or openly. The number of malicious nodes tolerated by a BFT protocol can not exceed $\lfloor \frac{N-1}{3} \rfloor$ (one of our goals is to go beyond this bound). Protocols can involve random coin tosses or be deterministic.

Byzantine Agreement protocols typically operate in rounds. If the number of malicious nodes exceeds $\lfloor \frac{N-1}{3} \rfloor$, there may be no agreement (the round is wasted), or with equivocation the malicious nodes can create two valid blocks, which is the equivalent of a *fork* in a blockchain protocol. Dealing with these types of forks is discussed later in the chapter (Section 6.3.8). If the malicious nodes constitute more than $\lfloor \frac{2N}{3} \rfloor$, they can force an incorrect agreement – *forgery* (which usually leads to a malicious takeover).

6.2.2 Assumptions

We assume *smartly malicious* nodes, which act so that in the case of round failure an external observer can not detect who disrupted the protocol. Malicious nodes can communicate with each other to detect if they constitute the necessary $\lfloor \frac{N-1}{3} \rfloor + 1$ nodes to disrupt the round, force an equivocation or the 2/3 fraction for an incorrect agreement.

The network is considered to be asynchronous. As a non-probabilistic protocol can not provide safety and liveness at the same time in an asynchronous network, these properties of ReCon are dependent on the permissioned consensus protocol used. However the validators are selected via reputation-based rules from a much larger permissionless set of nodes.

The BFT protocol requires the nodes to sign each message in order to provide the required integrity and authentication. Even though early BFT designs used MACs, they can use fast signatures such as Ed25519 or similar. Given that the transactions are signed in batches, the performance overhead due to signatures is negligible. There has been previous work on how to create more efficient BFT consensus using digital signatures in Byzcoin [KJG⁺16] and this approach is applicable to our case as well (but without using the PoW to select the validators). As in Byzcoin, this would allow ReCon to have at least an order (and possibly two orders) of magnitude improvement in transaction throughput compared to Bitcoin.

We study both scenarios where malicious nodes are determined before the protocol run and thus no honest node can become malicious, as well as a dynamic case in which nodes can become malicious or can become honest (cleaned), new nodes entering the system at certain rate, etc.. We also study the botnet takeover scenario, in which many nodes can become malicious (at random, including some high reputation nodes), or Sybil attack scenario where many malicious nodes but with zero or low reputation are injected at a fast rate.

The motivation for these assumptions is based on real life observations in open peer-to-peer blockchain based consensus protocols, as openness allows any level of malicious behaviour, thus the assumptions have to be as hard as possible. Our synchrony assumption is the hardest and thus the protocol is secure against network-based attacks in any circumstances which also covers any observed attacks in blockchain networks as well. The same logic led to our assumption of malicious nodes, where the attacker's goal is to thwart or take over the network, but it wants to achieve that in the least detectable way. Furthermore with our assumptions, an external observer can not distinguish honest and malicious nodes in a committee after a halted consensus.

6.2.3 Nodes

Node-to-node connection is authenticated with public keys. The corresponding PKI system is maintained by the chain with transactions in the network. In order to register a new node, revoke or refresh a key the user has to send a transaction with all the necessary data included. Such blockchain based PKI systems were shown in [Bon16, ABB+15]. Nodes participating in successful rounds of the protocol are rewarded by increase in their reputation score and potentially by cryptocurrency minted. Such cryptocurrency rewards as well as reputation score (which might have value outside of the protocol) motivate the economically rational behavior. Our protocol is permissionless, apart from the initial commitment of registering the public keys by the nodes. Generic node-to-node communication of distributing the new candidate transactions and the new blocks is done via the gossip protocol.

6.3 Reputation module

In this section we describe the reputation module ReCon, which can be plugged into any Byzantine Agreement protocol with the following rules:

- The protocol consists of (arbitrarily many) rounds.
- At each round N nodes decide the fate of one or many transactions.
- At each round the nodes may reach a consensus or not, and both outcomes are visible to all nodes.
- Each round a decision is made by a public committee C of m nodes, which does not necessarily include all the nodes. The committee decision is unforgeable².
- All the committee messages are signed by the transmitting node.
- At the end of each round the results are published as the new block of the chain.

 $^{^2{\}rm The}$ implementation of the secure committee broadcast is protocol-dependent [Buc16, KJG+16].

The protocol parameters can be found in Table 6.1. The default values in the table are not fixed, their purpose is to provide a general view on the protocol. We explore the different choices for most of the values in our simulations in Section 6.4. Furthermore Figure 6.1 is an example of a single round in our protocol. An exponential committee selection function chooses the committee members based on the node ranking per reputation, and if there are more than 2/3 honest nodes (with green), then the round will succeed, and every committee member's reputation increases, otherwise if the honest nodes are less than 2/3, then the protocol halts and the members' reputation is penalized.

Parameter	Notation	Default value
Total nodes	N	5,000
Committee size	m	100
External		Discrete Uniform
reputation	F	Normal
distribution		Exponential
Ongoing reputation	\mathcal{R}	_
Default malicious rate	α_0	0.4
Minimum malicious rate	α_1	0.05
Committee	D	Exponential
selection rule	D	Triangular
Security parameter ³	λ	30

Table 6.1: The protocol parameters

6.3.1 External Reputation

ReCon instructs the protocol how to select the committee and maintains the reputation ranking $\mathcal{R}: \mathcal{N} \to [0,1]$, where \mathcal{N} is the set of nodes, so that the nodes with high reputation have low posterior probability of being malicious. The prior probability of being malicious is given to the module and is called *external reputation*. If there is no external source of reputation, or, equivalently, all nodes have equal probability α_0 to be malicious, then we set: $R(n) \equiv 0 \quad \forall n \in \mathcal{N}$.

If the probability of being malicious varies from α_0 (default) to α_1 (minimum possible), then we normalize as

$$R(n) = 1 - \frac{P(n) - \alpha_1}{\alpha_0 - \alpha_1},$$

where P(n) is the probability that node n is malicious. Equivalently,

$$P(n) = (\alpha_0 - \alpha_1)(1 - R(n)) + \alpha_1.$$

We denote the initial distribution of R() by F, and consider various distribution functions (since R() and P() are affine equivalent, their distribution

³If p is probability of forgery in one round (i.e. malicious 2/3 majority), then $\lambda = -log p$.

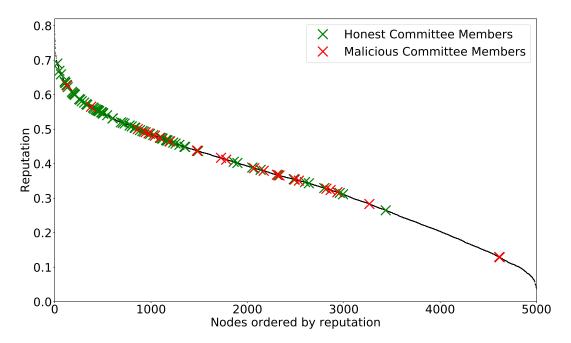


FIGURE 6.1: The reputation curve after 10,000 rounds, and the exact chosen committee members in that round with the 'X' markers (with green the honest and with red the malicious ones) using the exponential selection function. In this case 76 of the members are honest, which means the committee will reach consensus and the committee members' reputation will increase.

functions are similar). For instance, when R() follows the (0.5, 0.15)-normal distribution constrained to [0, 1], there are 23% malicious nodes in the top 10% nodes by reputation. The value Ω stands for the overall fraction of malicious nodes in \mathcal{N} . ReCon outputs a new reputation ranking \mathcal{R} , for which we experimentally estimate the posterior maliciousness probability.

6.3.2 Committee selection

The decision in a round is made by a committee, which runs a round of a BFT protocol on the current set of transactions, and decides to either apply each of them or not. If the committee comes to a consensus, the transactions are applied to the state. The committee is selected based on the current reputation \mathcal{R} the nodes inherited or earned during the previous rounds. C[r] denotes the committee of the r-th round.

The selection of the committee is based on some distribution D, where the higher reputation value R(n) would result in a higher chance of selection (e.g. exponential distribution, exponential power distribution, triangular distribution). Here P is the probability of being selected into a committee.

$$\forall n, l \in N : R(n) \ge R(l) \Rightarrow P(n|D) \ge P(l|D)$$

This selection algorithm (Figure 6.1) is implemented in the following way. First, we sort the nodes based on their reputation in a descending order. Then, based on D, m random numbers are generated in [0, N), and then taking the floor of all of them, we receive the selected nodes. In order to avoid double

selection we select the closest yet unselected node with a higher reputation. If such node does not exist, then we do the same going towards the lower reputed nodes.

We consider two different selection distributions: exponential and triangular. Exponential gives priority to the highly reputed nodes and can strongly suppress the lower ranked ones, depending on its variance. The triangular distribution is the more fair one for the new low reputation nodes – it gives priority proportionally to the reputation but at a cost of slower convergence and lower cost for a Sybil attack.

Though the actual distributions prioritize the higher reputed nodes, they will still allow low reputed nodes to be selected into the committees. In the exponential case, $\xi = -log(0.05)/N$. The distribution itself is truncated to the [0, N) interval. This ξ value means that $\int_0^N exp.dist.(\xi) = 0.95$, or in other words 95% is the chance of randomly getting an integer that is in the interval [0, N).

In a similar fashion, the triangular distribution is actually a distribution from 0 to $N + \frac{N}{5}$, truncated to the [0, N) interval, to give a chance to be included in a committee even to nodes that have a low reputation value.

6.3.3 Rewards and penalties

The reputation module observes whether the committee has reached consensus. In the "smart malicious" model we imply that these two outcomes are the result of the following configurations:

- The committee has reached consensus if there are fewer than m/3 (no influence) or more than 2m/3 (total control) malicious nodes in the committee.
- The committee has not reached consensus if the number of malicious nodes is between m/3 (non-inclusive) and 2m/3 (inclusive).

Thus in our simulation we model the protocol execution as follows:

- If C[r] has fewer than m/3 malicious nodes, then the round is declared success and every node in C[r] gets their reputation increased.
- If C[r] has m/3 or more malicious nodes, but less than 2m/3 malicious nodes, then the round is declared failure and every node in C[r] gets their reputation decreased. This event is undesirable (round time is wasted) but not catastrophic.
- If C[r] has 2m/3 or more malicious nodes, then the round is declared forgery. Since we can not detect externally if the decision is malicious or not, every node in C[r] gets their reputation increased. However in most cases this would mean a hostile takeover and such event should be avoided by the proper parameter choice in the protocol.

The exact rewards and penalties are calculated in the following way per node. In case of a reward, the reward function for node n is

$$R_r(n) = R(n) + \frac{(1-s)(1-R(n))}{d}, d \ge 1.$$
(6.1)

The penalty function is

$$R_p(n) = R(n) - \frac{s \cdot R(n)}{d}, d \ge 1,$$
 (6.2)

where s is the proportion of success rounds in the last 100 rounds (if 56 were successful, then s=0.56). The idea behind this adaptive parameter is the following. Our goal is to sort the participating nodes based on their likelihood of maliciousness. Thus we choose values, that will increase and decrease the reputation values by the same amount on average, but the nodes will be reordered based on their behaviour.

The divisor d is for optimization, as for different selection functions a different d will result in the best behaviour in our protocol. For example, in the case of exponential selection d=10, but for triangular selection d=35. These values are the results from our empirical testing of the protocol, where we simulated the behaviour of the nodes (Section 6.4). Below we explain our choice of these reward and penalty functions.

$$\frac{(1-s)(1-R(n))}{d} \lessgtr \frac{s \cdot R(n)}{d}$$
$$1-(s+R(n)) \lessgtr 0$$
$$s+R(n) > 1 \Rightarrow \text{penalty} > \text{reward}$$
$$s+R(n) < 1 \Rightarrow \text{penalty} < \text{reward}$$

In the case of s + R(n) > 1, notice that it is only true, if none of the values are 0, which means that there are definitely several consensus successes. Also notice, that if s = 0, then the value of penalty is 0, and if s = 1, then similarly the value of reward is 0.

These kind of changes in the values also provide us the feature, that if a node has a high reputation and participates in a bad round, it will be penalized more than a lower reputed node in the same failed round. It is true in the opposite direction as well, as the reward is higher for lower reputed nodes in successful rounds.

6.3.4 Probability of a forgery

We have to consider what is the probability of having a forgery $(P(k > \lfloor \frac{2m}{3} \rfloor))$, where m is the committee size and k is the number of malicious nodes in a committee). We model this as a Bernoulli trial where each member of the committee independently has probability p to be a malicious node. With this model the binomial distribution B(m,p) describes the committee selection. We also introduce a security parameter λ which describes the upper bound on the probability of forgery as $2^{-\lambda}$ (Tables 6.2). Then:

$$P\left(k > \left\lfloor \frac{2m}{3} \right\rfloor\right) < 2^{-\lambda} \iff 1 - \sum_{i=0}^{\left\lfloor \frac{2m}{3} \right\rfloor} {m \choose i} p^i (1-p)^{m-i} < 2^{-\lambda}$$

$$\frac{2^{-\lambda} 2^{-30} 2^{-60} 2^{-120}}{p 0.364 0.248 0.124}$$

Table 6.2: For m = 100, the λ security parameters and the corresponding p values, where p is the probability of selecting a bad node.

Furthermore, we can observe these p parameter values from the protocol attributes as well. In Table 6.3 we show the relative success rate (s, introduced) in Section 6.3.3) of the protocol calculated from the p parameter values found in Table 6.2. This also means that if the protocol achieves a higher success rate then 24.5% it is safe with $\lambda = 30$.

$2^{-\lambda}$	2^{-30}	2^{-60}	2^{-120}
Successful rounds (s)	24.5%	97.5%	99.99999%

Table 6.3: For m=100, the λ security parameters and the corresponding percentage of successful rounds based on the p values from Table 6.2. It shows that for $\lambda=30$ security parameter even if only 25% of the rounds reach consensus, the protocol is still safe.

If we increase the committee size m, the value of p increases as well, where $\lim_{n\to\infty} p = \frac{2}{3}$.

This only shows the values for cases of uniform choice, but in our protocol we use a ranking based on reputation and a weighted selection (Figure 6.1). In the following we show how to calculate the p parameter from an external reputation system and the selection algorithm using the law of total probability for a fixed N sized set. Let X be the event of selecting a bad node and Y_n the event of selecting the n-th node (the n-th based on the reputation ranking, see Section 6.3.2). Then:

$$p = P(X) = \sum_{n=1}^{N} P(X \cap Y_n) = \sum_{n=1}^{N} P(X|Y_n) \cdot P(Y_n) \text{ where}$$

$$P(X|Y_n) := P(Z_n) \text{ then}$$

$$P(Z_n) = P(\text{n-th node is malicious})$$

$$p = \sum_{n=1}^{N} P(Z_n) \cdot P(Y_n)$$

For the value of $P(Z_n)$ in case of an external reputation see Section 6.3.1, otherwise for an observed state:

$$P(Z_n) = \begin{cases} 1 \text{ if n-th node is malicious} \\ 0 \text{ if n-th node is honest} \end{cases}$$

6.3.5 Types of Blocks

The protocol allows 2 types of blocks to be added to the chain. The first type is the regular block which contains the transactions agreed upon by the committee. However we want our protocols to work beyond the 1/3 maliciousness limit of BFT protocols. To avoid protocol stalls over this threshold we introduce a mechanism to detect stalls but which does not introduce strict synchrony. We use a second type of block that acts as a timing epoch. To produce this block, we use Verifiable Delay Functions (VDFs, [BBBF18, Wes19, Pie19]).

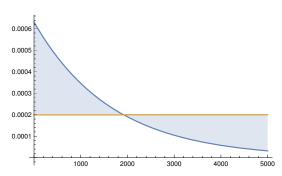
VDFs are a new type of cryptographic puzzles. Compared to PoW puzzles, which require random guessing of input values and can succeed at any try, VDFs provide a time-lock puzzle, which means that the input x of the function is fixed and the prover can not solve the puzzle in less then T steps. In essence, VDFs require an honest prover to provide a proof of sequential work of T steps on a fixed input, while an adversary prover can not parallelize the proving process. VDFs also require fast practical verification of the proofs, which is the key attribute of the protocol, as while the proof generation takes a long time, the verification of the proof is much faster. Finally, VDFs have to be unique, meaning that for all input x it is difficult to find a y, that the verification accepts x as input and y valid output, but the evaluation of x in the puzzle would not result in y.

In ReCon VDFs are used in the following way. The T value is set to be a large enough constant (e.g. a few minutes), that would give more than enough time for an honest committee to reach consensus. Then as soon as the committee is selected based on the previous block, any member⁴ of the entire network can start computing the VDF based on the last block. If a consensus is reached in time it will be distributed in the network and a new committee will be created for the next block. On the other hand, if a node creates a block with a correct VDF before it sees a new block with a correct consensus, it can gossip it in the network as the next block as a proof that the committee did not reach consensus in time.

In order to keep the protocol fair, we have to choose the time T very carefully based on comprehensive testing on what is the expected time for an honest committee to reach consensus and to avoid consequences of secret VDF computation optimization by the attacker which may allow him to penalize honest committees⁵. With this technique, there is no direct known Δ time that is given to the committee to reach consensus, but instead an unknown Δ time until the first block with a valid VDF appears, acting as a timing epoch. This way we do not require strong synchrony, as the Δ is unknown, which is in line with protocols working in asynchronous networks like PBFT.

⁴Or alternatively a smaller set of high reputation nodes, not participating in the current committee - for extra Sybil protection.

⁵Optimization free implementation of VDF is a non-trivial problem in itself. However if the protocol gains traction we expect that there will be public optimized hardware to compute VDFs.



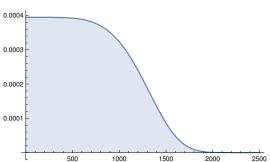


FIGURE 6.2: The filled area is the fairness of the exponential selection

FIGURE 6.3: The exponential power distribution, or sometimes also called the generalized normal distribution with the parameters (5;0;1,000)

6.3.6 Source of randomness

In our protocol we use a deterministic PRNG as randomness for the selection of the committee. The drawback of this approach is that successful committees might be able to manipulate the randomness with the list of chosen transactions (so called grinding). VDFs do provide unpredictable results, which makes them even more useful for our protocol as a re-randomization seed, but if every committee is successful there is no VDF computation. In order to still have a regular influx of hard to predict randomness we suggest that VDF computation is also performed at regular intervals (e.g. every 50 blocks).

6.3.7 Fairness

We define the fairness F of a selection function to be the L₁ distance between the uniform distribution over N nodes and the selection distribution over the same interval, namely:

$$F = \int_{0}^{N} |f(x) - \frac{1}{N}| dx \tag{6.3}$$

Where f(x) is the probability density function of the selection distribution. The idea behind the definition is to describe how close the selection distribution is to the uniform distribution, which would be considered as perfectly fair. It is the most fair, because as an observer of the protocol we do not know which nodes are malicious, thus we should give the same probability to every node to be chosen for a committee. This way our selection functions would produce the following fairness values when N = 5,000: $F_{triangular} = 0.357$; $F_{exponential} = 0.671$.

This is also the reason why both of our selection functions (triangular and exponential) are selected in such a way, that even the node with the lowest reputation will have a chance of being selected, instead of completely ignoring the last few nodes. If we were to design our selection function with the last nodes only having close to zero chance of being selected, then the fairness values would be much worse: $F_{triangular} = 0.5$; $F_{exponential} = 0.918$.

One could consider other distributions as a selection function, e.g. a selection that would act as a filter, which selects almost only from the highest reputed nodes. An example for that can be the exponential power distribution (Figure 6.3). This selection function, however, would be very unfair, as the fairness value would be $F_{exponential power} = 1.35$ and the bottom nodes have no chance to be selected for the committee.

Implicitly we consider a fairness value of above 1 as unfair based on our empirical data. However such a selection function might be useful during botnet takeover or Sybil attack events.

6.3.8 Dealing with forks

As we noted earlier, if the adversary has control over 1/3 of the committee, different scenarios can happen. Even though each BFT protocol may have its own method to resolve these situations, we list some solutions here.

The first option for the adversary is halting the protocol by not participating in the Byzantine Agreement, and thus there will be no new blocks created in that round, as $\lfloor \frac{2N}{3} \rfloor + 1$ signatures are required for a block to be accepted. In this case a valid VDF timeout block will be created. This might lead to forks as well, if a consensus succeeds, but another member of the network created a VDF block meanwhile. In this case we let the network handle the fork and simply use the longest chain rule for choosing the valid chain.

The second one is equivocation. The adversary splits the honest nodes into two subgroups, such that he has 2/3 majority with either of them combined with himself. Then he communicates different transactions to these groups, thus creating two valid blocks in the same round. Both blocks contain only valid transactions, as they need signatures from honest nodes, and honest nodes will only approve valid transactions. However, as all protocol messages are signed, an evidence of signing both blocks can be presented in the next committee rounds and the malicious nodes will have their reputation score reset to 0.

Notice the difference between the fork by VDF blocks and a fork by equivocation. In case of equivocation the committee creates two separate valid consensuses with overlapping verifiers, while with a VDF only one of the blocks is a valid consensus at the same chain height. This approach makes the protocol probabilistic, as there is a possibility for forks but keeps it asynchronous. It is well known that a protocol can not be deterministic in an asynchronous setting, either the safety or liveness would break. Considering this, forks in the blockchain temporarily break the safety of the protocol, but they keep the liveness property in all circumstances while the protocol converges to a single chain and restores the safety. This is the opposite compromise to classic BFT protocols like PBFT.

6.3.9 Convergence

We say that the BFT protocol β -converges after l rounds if the success rate (fraction of successful rounds) never goes below β after l rounds.

Concrete convergence parameters depend on the application. The values α_0 , α_1 (from Table 6.1) determine what success rate s can be guaranteed by the ReCon ranking, and the value l determines the length of the bootstrap phase needed to rank the nodes.

6.3.10 Pseudocode

The protocol description as a pseudocode (Algorithm 5), where N is the number of nodes in the network, m is the committee size and \mathcal{R} is the array of reputation values. C[r] is the selected committee in round r generated by the $gen_committee$ function. The new gossiped block $block_r$ contains the exact reward and penalty values based on the result of the consensus or VDF block. To keep the description simpler we don't specify which nodes compute the VDF, but this can be easily determined based on the previous block. An important consideration is whether we allow any node to compute the VDF, or limit it to a small set of high reputation nodes (who are not in the current committee). The latter would provide additional Sybil resistance.

6.4 Simulation Results

We ran our simulations⁶ for $10,000 \text{ rounds}^7$ with default values of total nodes⁸ $N = \{5,000,10,000,20,000,30,000\}$, committee size m = 100 and various combinations of external reputation and selection rule. Every combination is tested 100 times and the results are averaged.

We note that if there is an external reputation, we set α_1 to 0.05. We also introduce a new variable, namely Ω , which is the overall malicious rate of the nodes (so that in N nodes there are $\Omega \cdot N$ malicious ones).

We study three cases for the external reputation: (a) no external reputation, equivalent to the uniform zero reputation; (b) normally distributed reputation and (c) exponentially distributed reputation. Normal distribution of reputation is natural in scenarios where ranking or reputation is determined by many independent factors. We chose one with parameters N(0.5, 0.15) so that its restriction to the [0,1] interval covers more than 99% of events (the 3σ rule).

We take exponential distribution with $\xi = 0.3$, since according to [LLH03] the reputation distribution in an online consumer-to-consumer network as well as in most social networks is exponential.

⁶The simulator is available with a user friendly interface at https://github.com/cryptolu/ReCon.

⁷Note that if we take a conservative estimate of 60 seconds per round, 10,000 rounds would take 166 hours. Thus a bootstrap phase of a few thousand rounds is reasonable for the convergence of reputations.

⁸At the time of writing the maximum number of nodes in the Bitcoin network was around 10,000, while in Ethereum around 30,000.

```
Algorithm 5 ReCon Reputation module
  procedure Round(r, block_{r-1})
                                                    ▶ The main round function
     C[r] := gen\_committee(N, m, \mathcal{R}, block_{r-1}) \triangleright Apply reput. selection rule
     If r is divisible by l, only create a VDF block, no committee selected
     if distr\_cons(C[r], block_r) then
                                         ▶ Whether the consensus is successful
         gossip(block_r) \triangleright Contains all information from a successful consensus
         Round(r+1, block_r)
     else
         qossip(block_r)
                            ▶ Contains all information from a failed consensus
         Round(r+1, block_r)
     end if
  end procedure
  procedure distr\_cons(C[r], new\_block)
                                               ▶ Returns new block and a flag
     if myNode in C[r] then start\_consensus\_alg(C[r], myNode)
     while !new_block do wait()
                                              ▶ Wait till new block is returned
     end while
     if fork(r) then
                                   \triangleright A fork has happened in some round k < r
         Let mal_nodes be the nodes that signed both chains
         reset(mal\_nodes)
                               ▶ Set their reputation to 0, or even delete them
     end if
     if new\_block.type = CONSENSUS then
                                                      ▷ Consensus was reached
         reward(C[r])
         return TRUE
     else if new\_block.type = VDF then \triangleright VDF was faster than consensus
         penalise(C[r])
         return FALSE
     end if
  end procedure
```

6.4.1 External reputation: discrete (no information)

In this case every node has equal chance α_0 of being malicious, and the initial reputation is zero for all nodes. We consider two different selection rules and every 100 rounds we increase or decrease the variance of the selection distribution by a certain value.

First, we consider the exponential selection rule. We start with a variance of $1/N = \xi^{-2}$, and then increase it every 100 rounds by $\frac{1}{N-500i}$ starting with i=1. We do this until we reach a variance, for which $P(X < N) \ge 0.9$, where X is the random exponential variable with $\xi = \frac{1}{N-500i}$. This means that the exponential distribution is mostly restricted to the [0; N) interval. Thus at the start of the protocol every node has a similar chance to gain reputation, and later more trusted nodes have more significance.

Our results (Table 6.4) show that the protocol converges to a correct behaviour even if 45% of the nodes are malicious. However, the success rate decreases as the initial malicious rate Ω grows.

Then we consider the triangular distribution for its more fair selection process, as even the node with the lowest reputation score should have a real chance of participating in a committee (results in Table 6.5). In this case, we start with a length of 10 times N, truncate it to N, and reduce this length by N every 100 rounds. After a 1,000 rounds, we settle with the aforementioned (Section 6.3.2) length of $N + \frac{N}{5}$ truncated to N.

Ω	Success Rate					
	N = 5,000	N = 10,000	N = 20,000	N = 30,000		
0.1	100%	100%	100%	100%		
0.2	99.95%	99.8%	99.8%	99.8%		
0.25	99.7%	99.5%	99.7%	98.7%		
0.33	99.6%	99.5%	98.6%	98.2%		
0.4	98.7%	98.2%	96.5%	95.3%		
0.45	96.5%	94.2%	89.9%	76.9%		

Success Rate N = 10,000 N = 20,000 N = 30,000N = 5,000100% 100% 100% 100% 0.2 99.92% 99.9% 99.9% 99.9% 0.25 98.8% 98.1% 98% 96.7% 96.3% 95.9% 92% 87% 89.1% 85.8% 60.1% 78% 60%50.3% 23 2% 9.9%

Table 6.4: No external reputation, exponential selection rule: success rates after 10,000 rounds.

Table 6.5: No external reputation, triangular selection rule: success rates after 10,000 rounds.

The difference in success rates can be explained with our introduced F fairness value. The triangular distribution has a better fairness value, which means it will choose lower reputed nodes more often, and such it can sort them better as well. On the other hand, the same can be said about the exponential distribution, as it has a worse fairness value, and it will choose higher reputed nodes more often, but because of that it will not be able to sort out the nodes that well.

6.4.2 External reputation with normal distribution

We repeat our previous tests with external reputation distributed normally and the maliciousness probability varying from α_0 to $\alpha_1 = 0.05$. First we consider the selection rule based on exponential distribution (see Table 6.6).

	Ω	Success Rate			
α_0	3.2	N = 5,000	N = 10,000	N = 20,000	N = 30,000
0.4	0.225	99.99%	99.9%	99.9%	99.9%
0.6	0.325	99.8%	99.7%	99.6%	99.6%
0.7	0.375	99.5%	99.3%	98.8%	98.8%
0.8	0.425	98.8%	98.8%	98%	95.7%
0.9	0.475	93%	90%	84.8%	77%
1	0.525	50%	47.9%	41.4%	40.7%

Table 6.6: External normally distributed reputation, exponential selection rule: success rates after 10,000 rounds.

α_0 Ω	Success Rate				
α_0	22	N = 5,000	N = 10,000	N = 20,000	N=30,000
0.4	0.225	99.98%	99.9%	99.9%	99.9%
0.6	0.325	98.1%	97.5%	97%	94.7%
0.7	0.375	97%	96.7%	93.5%	86.7%
0.8	0.425	91%	89.8%	79%	65.2%
0.9	0.475	50%	39.7%	9.2%	9.0%
1	0.525	1%*	1%*	0.8%*	0.7%*

Table 6.7: Simulation results in the case of external normal distribution, selection with triangular distribution. The last simulation has an asterisk, as it produced a forgery in one of the runs.

The results in Table 6.6 show, that even in heavily adversarial settings of $\Omega=0.475$ the protocol 0.93-converges, while for $\Omega=0.525$ it 0.5-converges. The difference based on the selection distributions between Tables 6.6,6.7 can be explained with the same reasoning as in the previous case. We can achieve

better success rates compared to Tables 6.4,6.5 because of the pre-sorting of the nodes based on the external reputation. This is natural and demonstrates that trusted external reputation enhances the Sybil resistance of the protocol.

6.4.3 External reputation with exponential distribution

In the case of an external exponential reputation system, the number of rounds for convergence values is bigger for the same α_1 , but the overall malicious rate is much higher. For $\alpha_0 = 0.6$ we have $\Omega = 0.45$, and in the case of $\alpha_0 = 0.7$ it is 0.53. Also notice that we do not achieve our required success rate, but the 0.7 and 0.75 cases still converged to a lower value, and they never produced a forgery in our simulations. First we show the results for the selection based on exponential distribution (Table 6.8).

α_0	Ω	Success Rate			
		N = 5,000	N = 10,000	N = 20,000	N = 30,000
0.4	0.307	99.6%	99.5%	99.4%	98.9%
0.5	0.381	99.1%	99.0%	98.9%	96.6%
0.6	0.456	97%	96.8%	93.4%	87.1%
0.7	0.529	51.2%	51.1%	50.7%	47.7%
0.75	0.565	28.5%	25.7%	18.7%	8.4%
0.8	0.605	5%*	3.6%*	0.1%*	0.1%*

Table 6.8: External exponentially distributed reputation, exponential selection rule. The last simulation has an asterisk, as it produced a forgery in one of the runs

0/-	Ω	Success Rate				
α_0	α_0 12	N = 5,000	N=10,000	N = 20,000	N = 30,000	
0.4	0.307	97%	96.9%	93.6%	93.3%	
0.5	0.381	93%	92.8%	82.2%	74.6%	
0.6	0.456	67%	62.7%	31.1%	18.4%	
0.7	0.529	1%*	1%*	0.1%*	0.1%*	

TABLE 6.9: Simulation results in the case of external exponential distribution, selection with triangular distribution. The last simulation has an asterisk, as it produced a forgery in one of the runs

In triangular selection case we have similar results as with the external normal reputation system.

6.5 Attacks and their mitigation

We will now consider attacks, based on examples from real world financial blockchains, such as Bitcoin and Ethereum. We will also propose good defences.

6.5.1 Botnet takeover

Our first example is a botnet takeover, where an attacker takes over the control of a large subset of nodes, and tries to either block the protocol (Denial of Service - DoS), or even create a forgery. The success of the attack largely depends on the number of nodes taken over, but the results can be vastly different based on the reputation value of those nodes.

6.5.1.1 Mitigation

We have simulated these attacks, and in the case of a large takeover of 1,000 random nodes, where N=5,000, the success rate s of the protocol dropped heavily at first from above 95% to a minimum of 40%, but it recovered in a

few hundred rounds, and got close to its previous success rate. As discussed in Section 6.3.4, even a success rate of 25% achieves $\lambda=30$ security, as we can revert the success rate into a binomial distribution. If a large enough subset is taken over, that can cause a forgery, but that would mean the overall number of malicious nodes would be probably above 50%. Note also that botnet takeover would be noticeable by the rapid drop in the success rate of the protocol - which any node can efficiently and locally measure and which can be used to trigger a temporary switch to less fair but more robust selection rules.

6.5.2 Sybil attack: saturation

In this version of the well-known Sybil attack, a large number of new malicious nodes (more then N/5) join the protocol, and try to subvert the performance, or even create a forgery. However such nodes would have zero initial reputation.

6.5.2.1 Mitigation

The protocol may require a new node to participate only in communications without any eligibility for selection into a committee for a set amount of time (e.g. 2 weeks). Then every new node would start from reputation value 0. For an attacker to gain a large enough probability of one of its nodes being selected into a committee would require either buying and running many dedicated servers, or controlling a botnet for weeks.

It is also easy to detect if many nodes are joining the network at the same time and could be also a trigger for a switch to more conservative selection rules.

6.5.3 Sybil attack: lie and wait strategy

A more dangerous version of a Sybil attack would be if the malicious nodes only act badly, if they have 2/3 majority in a committee. At this point they just take over the network.

6.5.3.1 Mitigation

Due to random selection even nodes with high reputation might have to wait for long before getting a chance to create a forgery. Thus the adversary has to control a high number of nodes and have to keep up them active until that round. This would be costly and we choose the security parameter λ so that probability of this attack is negligible (ex. below 2^{-30} in any given round).

6.5.4 Attacks on randomness

Another attack would be simply DoS-ing the committee members, as their participation is publicly known to all the nodes in the protocol. If an attacker is a node, and learns the members of the next committee quickly enough, he can DoS a portion of them, which would stall the protocol.

6.5.4.1 Mitigation

A defense against a DoS attack could be generating multiple committees (in the limit every node being in some committee), making it harder and more expensive for the attacker to DoS more than 1/3 of the nodes in all of them. As for which committee will produce the actual block it could be decided by an external unpredictable beacon (possibly based on VDF). Note that DoS attack would be very noticeable by the sharp decrease of the success rate s of the protocol, and thus this mitigation can be switched on only when it is really needed.

6.5.5 Honest majority

Another problem could be the fact, that we require only an honest majority in the committees, and there is no rational reason for acting honestly.

6.5.5.1 Mitigation

This can be mitigated in two different ways. Firstly, there are real world examples (e.g. Bitcoin or Tor), where there is no direct reward for running a full node (or Tor relay), only the indirect reward, that the user can personally monitor the validity of transactions. Even this way there are more than 10,000 Bitcoin full nodes currently in the network (and over 7,000 Tor relays).

Secondly, we can introduce a small reward for participating in a correct committee (for example, by minting some of the cryptocurrency in the BFT process or by distributing transaction fees), which would introduce some economic rationale for acting honestly. The problem with that is, that it would decrease the cost of a Sybil lie and wait strategy (Section 6.5.3), as running nodes would not be that expensive, or would even partly pay for themselves. Because of that these rewards would have to stay either relatively small so that running even a highly reputed node would not pay for itself or the opposite, so that attacking the network would be against the economic interest of the adversary (similar to the current situation with mining in Bitcoin).

6.5.6 Detection based on the success rate

A lot of attacks are detectable by simply monitoring the success rate s (Section 6.3.3). If there is a significant drop (e.g. 10% at least) in the number of successful rounds, the protocol can automatically employ a stricter selection rule (e.g. exponential power rule), which would quickly penalize bad nodes at the cost of being temporarily unfair to some of the honest nodes. The protocol can switch back to a more democratic triangular selection rule when the success rate improves.

6.6 Summary and Conclusions

In this chapter we have described a novel approach for more scalable permissionless blockchain consensus protocols that are resilient against Sybil-attacks.

Our protocol ReCon utilizes external reputation ranking to select a small subset of validators from a large set of nodes for a fast permissioned BFT protocol. This in turn would help to improve transaction throughput by one or two orders of magnitude compared to Bitcoin's Nakamoto consensus. Our solution allows Bitcoin-style egalitarian peer-to-peer networks of thousands of validator nodes without the energy waste of a Proof-of-Work based blockchain. Our protocol also tolerates a larger threshold of malicious nodes than a BFT consensus - 1/2 instead of 1/3, although that also depends on the exact distribution of the malicious nodes.

Chapter 7

Summary and Conclusions

In this thesis we have expanded the existing academic knowledge regarding the novel blockchain research area, providing useful general analysis and attacks against user privacy in the privacy preserving cryptocurrency Zcash. These attacks can also be used as a guidance for the general users as to what kind of behaviour they should avoid. These studies also show that in general any protocol that mixes public and private services where there is direct communication between the two parts can leak privacy sensitive information and these services can not be used just as a black box to provide the desired privacy. Furthermore we have described a new blockchain based permissionless consensus protocol utilizing reputation rankings. Let us summarize our finding and user suggestions on a per chapter basis.

In Chapter 2 we have shown two heuristics to link mining related hiding and revealing transactions in Zcash. We have linked over 88.2% of the mined coins through the shielded pool of transactions.

This work shows that even if a blockchain is theoretically safe, bad use practices and an intermix of hidden and public transactions can lead to considerable information leakage defeating the very strong cryptographic privacy features of Zcash. Moreover, since hidden transactions formed only 13.4% of the total number of transactions, 95.5% of all Zcash transactions became potentially linkable, which is very close to privacy level of the original Bitcoin blockchain. This study shows that if there is no proper incentive for the mandatory usage of privacy preserving techniques, the overall effect of these techniques can be negligible.

We also note, that if mining directly to shielded addresses by the mining pool members became more popular, it would reduce the accuracy of our heuristics, as the hiding and revealing volumes might not be equal or close to equal anymore. As of now we have not observed widespread implementation and usage for payouts directly to shielded addresses.

In Chapter 3 we have shown some privacy issues in the Zcash cryptocurrency, mainly utilizing the hiding and revealing transactions that convert coins between the private and public part of the blockchain. We have also shown two novel active attacks - Danaan-gift attack and Dust attack - against Zcash user privacy, and we have provided a theoretical model and a statistical analysis for their success likelihood. We have reinforced the notion that Zcash is not just a blackbox solution, where if a user uses the shielded transactions, it has constant strong privacy. Instead, we show that even the users of Zcash have to put some care into how they use the shielded transactions, and how they transfer coins between the public and private parts of the blockchain.

We would like to provide some general suggestions for the users of Zcash to try to avoid linkage using any of these methods. First, if a user has to pay coins to a shielded address and all of its coins are in public addresses, the user should either shield the coins in multiple transactions, or shield more coins than it needs to transfer to avoid issues in case the receiver of the coins would reveal them in the future. Users should avoid using the default transactions fee. They should also pay attention to the number of outputs they control in shielded outputs, as spending all of them together leaves a visible trace on the blockchain. Finally, it is also safer to hide and shield coins with round values, where at least the last 4 digits of the value should be zero, especially if the economic value of those digits is negligible.

In Chapter 4, we have presented an overview of decentralized GPU mining in cryptocurrencies. We have studied the most popular mining hardware, while also investigating the effect of the introduction of ASICs into the mining ecosystem of Zcash. We have provided methods that could be used to detect hidden ASIC farms in a network, and verified their effectiveness in practice. We have also shown how the overall effect of ASICs and reduced exchange rates damage the decentralization of mining power, leading to a disappearance of over 75% of the miners in Zcash.

We have shown how using only blockchain information an attacker can learn the hash rate of a miner and might even deduce the structure of their mining rigs, reducing their privacy. This study helps to expose privacy vulnerabilities in the current mining ecosystem which is crucial for privacy-preserving currencies and privacy-conscious users.

In Chapter 5, we have investigated the behavior of cryptocurrency exchanges on the Bitcoin blockchain. We found, that in the case of some well-known exchanges the online reported traffic and the traffic on the blockchain match up close to each other, which can be used as a clue in verifying the correctness of the reported exchange traffics. Furthermore, we have provided heuristics that try to classify large clusters of addresses whether these clusters are controlled by exchange services. As the work is experimental in nature, we do not claim any hard truth, but the study can be used as a building block and an interesting clue for further studies.

Finally in Chapter 6, we have described a novel approach for more scalable permissionless blockchain consensus protocols that are resilient against Sybilattacks. Our protocol ReCon utilizes external reputation ranking to select a small subset of validators from a large set of nodes for a fast permissioned BFT protocol. This in turn would help to improve transaction throughput by one or two orders of magnitude compared to Bitcoin's Nakamoto consensus. Our solution allows Bitcoin-style egalitarian peer-to-peer networks of thousands of validator nodes without the energy waste of a Proof-of-Work based blockchain. Our protocol also tolerates a larger threshold of malicious nodes than a BFT consensus - 1/2 instead of 1/3, although that also depends on the exact distribution of the malicious nodes.

7.1. Future Works 107

7.1 Future Works

We would like to suggest direction for future research based on our findings. First, monitoring the evolution of the Zcash blockchain and performing studies on newer data might provide new insights. Expanding of our methodologies to study other privacy blockchains is also a natural and useful continuation of this work. Furthermore, either a more sophisticated probabilistic model or a generic simulator for blockchains is a research area that still has open problems and questions for future studies.

Regarding the mining landscape with even more data on ASICs in Zcash a further study could be performed focusing on the evolution of the new mining hardware, and how the new hardware disrupts the mining profitabilities of the older ASICs. Study of mining centralization and its effect on the blockchain ecosystem is of research interest.

Our study on exchange services is experimental in nature and the goal of it is to provide some foundation and clues for a larger general study on exchanges, focusing on their effect on Bitcoin as well as a study of interaction between exchanges and public blockchains in general.

Bibliography

- [ABB⁺15] Christopher Allen, Arthur Brock, Vitalik Buterin, Jon Callas, Duke Dorje, Christian Lundkvist, Pavel Kravchenko, Jude Nelson, Drummond Reed, Markus Sabadello, Greg Slepak, Noah Thorp, and Harlan T Wood. Decentralized public key infrastructure: whitepaper, 2015. https://danubetech.com/download/dpki.pdf.
- [ACBM08] Elli Androulaki, Seung Geol Choi, Steven M. Bellovin, and Tal Malkin. Reputation systems for anonymous networks. In Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings, volume 5134 of Lecture Notes in Computer Science, pages 202–218. Springer, 2008. https://doi.org/10.1007/978-3-540-70630-4_13.
- [ACP+17] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Advances in Cryptology EUROCRYPT 2017 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 May 4, 2017, Proceedings, Part III, volume 10212 of Lecture Notes in Computer Science, pages 33–62, 2017. https://doi.org/10.1007/978-3-319-56617-7_2.
- [AKR⁺13] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In Financial Cryptography and Data Security 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers, volume 7859 of Lecture Notes in Computer Science, pages 34–51. Springer, 2013. https://doi.org/10.1007/978-3-642-39884-1_4.
- [AMQ13] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. RBFT: redundant byzantine fault tolerance. In *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*, pages 297–306. IEEE Computer Society, 2013. https://doi.org/10.1109/ICDCS.2013.53.
- [Bac02] Adam Back. Hashcash a denial of service countermeasure. 2002. ftp://sunsite.icm.edu.pl/site/replay.old/programs/hashcash/hashcash.pdf.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I, volume 10991 of Lecture Notes in Computer Science, pages 757-788. Springer, 2018. https://doi.org/10.1007/978-3-319-96884-1_25.

- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349. ACM, 2012. https://doi.org/10.1145/2090236.2090263.
- [BCG⁺14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014, pages 459–474. IEEE Computer Society, 2014. https://doi.org/10.1109/SP.2014.36.
- [BCS16] Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In Advances in Cryptology ASI-ACRYPT 2016 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I, volume 10031 of Lecture Notes in Computer Science, pages 220–248, 2016. https://doi.org/10.1007/978-3-662-53887-6_8.
- [BF19a] Alex Biryukov and Daniel Feher. Portrait of a miner in a land-scape. In *IEEE INFOCOM 2019 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 May 2, 2019*, pages 638–643. IEEE, 2019. https://doi.org/10.1109/INFCOMW.2019.8845201.
- [BF19b] Alex Biryukov and Daniel Feher. Privacy and linkability of mining in zcash. In 7th IEEE Conference on Communications and Network Security, CNS 2019, Washington, DC, USA, June 10-12, 2019, pages 118-123. IEEE, 2019. https://doi.org/10.1109/CNS.2019.8802711.
- [BF20] Alex Biryukov and Daniel Feher. Recon: Sybil-resistant consensus from reputation. In *Pervasive and Mobile Computing*, volume 61, page 101109, 2020. https://doi.org/10.1016/j.pmcj.2019.101109.
- [BFV19] Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in zcash. In *Proceedings of the 2019 ACM*

SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019, pages 1795–1811. ACM, 2019. https://doi.org/10.1145/3319535.3345663.

- [BGK⁺18] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 913–930. ACM, 2018. https://doi.org/10.1145/3243734.3243848.
- [bit20] Bitcoin wiki payment channels, 2020. https://en.bitcoin.it/wiki/Payment_channels.
- [BK16] Alex Biryukov and Dmitry Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. In 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016.

 The Internet Society, 2016. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/equihash-asymmetric-proof-of-work-based-generalized-birthday-problem.pdf.
- [BKP14] 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, Scottsdale, AZ, USA, November 3-7, 2014, pages 15–29. ACM, 2014. https://doi.org/10.1145/2660267.2660379.
- [BLMR14] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. In SIGMETRICS Perform. Evaluation Rev., volume 42, pages 34–37, 2014. https://doi.org/10.1145/2695533.2695545.
- [Bon16] Joseph Bonneau. Ethiks: Using ethereum to audit a CONIKS key transparency log. In Financial Cryptography and Data Security FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers, volume 9604 of Lecture Notes in Computer Science, pages 95–105. Springer, 2016. https://doi.org/10.1007/978-3-662-53357-4_7.
- [BP15] Alex Biryukov and Ivan Pustogarov. Bitcoin over tor isn't a good idea. In 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015, pages 122–134. IEEE Computer Society, 2015. https://doi.org/10.1109/SP.2015.15.

[Buc16] Ethan Buchman. Tendermint: Byzantine fault tolerance in the age of blockchains. master thesis, 2016. https://atrium.lib.uoguelph.ca/xmlui/bitstream/handle/10214/9769/Buchman_Ethan_201606_MAsc.pdf?sequence=7.

- [CBL17] Rémy Cazabet, Rym Baccour, and Matthieu Latapy. Tracking bitcoin users activity using community detection on a network of weak signals. In Complex Networks & Their Applications VI Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications), COMPLEX NETWORKS 2017, Lyon, France, November 29 December 1, 2017, volume 689 of Studies in Computational Intelligence, pages 166–177. Springer, 2017. https://doi.org/10.1007/978-3-319-72150-7_14.
- [cco19] Cryptocompare, 2019. https://data.cryptocompare.com/.
- [Cha82] David Chaum. Blind signatures for untraceable payments. In Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982, pages 199-203. Plenum Press, New York, 1982. https://doi.org/10.1007/978-1-4757-0602-4_18.
- [CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, pages 173–186. USENIX Association, 1999. https://dl.acm.org/citation.cfm?id=296824.
- [CLN⁺08] Allen Clement, Harry C. Li, Jeff Napper, Jean-Philippe Martin, Lorenzo Alvisi, and Michael Dahlin. BAR primer. In *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, June 24-27, 2008, Anchorage, Alaska, USA, Proceedings*, pages 287–296. IEEE Computer Society, 2008. https://doi.org/10.1109/DSN.2008.4630097.
- [CM18] Brad Chase and Ethan MacBrough. Analysis of the XRP ledger consensus protocol. CoRR, abs/1802.07242, 2018. http://arxiv.org/abs/1802.07242.
- [Dai98] Wei Dai. B-money. 1998. http://http://www.weidai.com/bmoney.txt.
- [DD18] Evan Duffield and Daniel Diaz. Dash: A privacy-centric cryptocurrency, 2018. https://github.com/dashpay/dash/wiki/Whitepaper.
- [de 18] Alex de Vries. Bitcoin's growing energy problem. In *Joule*, volume 2, pages 801 805, 2018. http://www.sciencedirect.com/science/article/pii/S2542435118301776.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II, volume 9216 of Lecture Notes in Computer Science, pages 585–605. Springer, 2015. https://doi.org/10.1007/978-3-662-48000-7_29.

- [DGH⁺87] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 1–12. ACM, 1987. https://doi.org/10.1145/41840.41841.
- [DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Advances in Cryptology EURO-CRYPT 2018 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 May 3, 2018 Proceedings, Part II, volume 10821 of Lecture Notes in Computer Science, pages 66–98. Springer, 2018. https://doi.org/10.1007/978-3-319-78375-8_3.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. In *IEEE Trans. Inf. Theory*, volume 22, pages 644–654, 1976. https://doi.org/10.1109/TIT.1976.1055638.
- [DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Advances in Cryptology CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings, volume 740 of Lecture Notes in Computer Science, pages 139–147. Springer, 1992. https://doi.org/10.1007/3-540-48071-4_10.
- [Dou02] John R. Douceur. The sybil attack. In Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers, volume 2429 of Lecture Notes in Computer Science, pages 251–260. Springer, 2002. https://doi.org/10.1007/3-540-45748-8_24.
- [DS01] Drew Dean and Adam Stubblefield. Using client puzzles to protect TLS. In 10th USENIX Security Symposium, August 13-17, 2001, Washington, D.C., USA. USENIX, 2001. http://www.usenix.org/publications/library/proceedings/sec01/dean.html.
- [DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In 13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento,

- Italy, September 9-11, 2013, Proceedings, pages 1-10. IEEE, 2013. https://doi.org/10.1109/P2P.2013.6688704.
- [EGSvR16] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016, pages 45-59. USENIX Association, 2016. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal.
- [ES18] Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. In *Commun. ACM*, volume 61, pages 95–102, 2018. https://doi.org/10.1145/3212998.
- [eth20] Ethereum wiki ethash, 2020. https://eth.wiki/en/concepts/ethash/ethash.
- [Eya15] Ittay Eyal. The miner's dilemma. In 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015, pages 89–103. IEEE Computer Society, 2015. https://doi.org/10.1109/SP.2015.13.
- [FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Public Key Cryptography PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings, volume 4450 of Lecture Notes in Computer Science, pages 181–200. Springer, 2007. https://doi.org/10.1007/978-3-540-71677-8_13.
- [GHM⁺17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017. https://doi.org/10.1145/3132747.3132757.
- [HBHW16] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. Technical report, Tech. rep. 2016-1.10. Zerocoin Electric Coin Company, 2016. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.
- [int17] Intel: Sawtooth lake. Technical report, 2017. https://github.com/hyperledger/sawtooth-core.
- [JMV01] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). In *Int. J. Inf. Sec.*, volume 1, pages 36–63, 2001. https://doi.org/10.1007/s102070100002.

[KAD+09] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. Zyzzyva: Speculative byzantine fault tolerance. In *ACM Trans. Comput. Syst.*, volume 27, pages 7:1–7:39, 2009. https://doi.org/10.1145/1658357.1658358.

- [KFTS17] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In Computer Security ESORICS 2017 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II, volume 10493 of Lecture Notes in Computer Science, pages 153–173. Springer, 2017. https://doi.org/10.1007/978-3-319-66399-9_9.
- [KJG⁺16] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016, pages 279–296. USENIX Association, 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias.
- [KJG⁺18] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA, pages 583–598. IEEE Computer Society, 2018. https://doi.org/10.1109/SP.2018.000-5.
- [KKZ20] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn. In Financial Cryptography and Data Security 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers, volume 12059 of Lecture Notes in Computer Science, pages 523–540. Springer, 2020. https://doi.org/10.1007/978-3-030-51280-4_28.
- [KML+20] Harry A. Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. Blocksci: Design and applications of a blockchain analysis platform. In 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, pages 2721-2738. USENIX Association, 2020. https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner.
- [KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Advances in Cryptology CRYPTO 2017 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I, volume 10401

of Lecture Notes in Computer Science, pages 357–388. Springer, 2017. https://doi.org/10.1007/978-3-319-63688-7_12.

- [KSG03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, pages 640–651. ACM, 2003. https://doi.org/10.1145/775152.775242.
- [KYMM18] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018, pages 463-477. USENIX Association, 2018. https://www.usenix.org/conference/usenixsecurity18/presentation/kappos.
- [Lam98] Leslie Lamport. The part-time parliament. In *ACM Trans. Comput. Syst.*, volume 16, pages 133–169, 1998. https://doi.org/10.1145/279227.279229.
- [LD20] Duke Leto and The Hush Developers. Attacking zcash for fun and profit. Cryptology ePrint Archive, Report 2020/627, 2020. https://eprint.iacr.org/2020/627.
- [LF16] Matthias Lischke and Benjamin Fabian. Analyzing the bitcoin network: The first four years. In *Future Internet*, volume 8, page 7, 2016. https://doi.org/10.3390/fi8010007.
- [LLH03] Zhangxi Lin, Dahui Li, and Wayne Huang. Current security management & ethical issues of information technology. chapter Reputation, Reputation System and Reputation Distribution: An Exploratory Study in Online Consumer-to-consumer Auctions, pages 249–266. IGI Global, Hershey, PA, USA, 2003. http://dl.acm.org/citation.cfm?id=949953.949968.
- [LLJ⁺11] Yining Liu, Keqiu Li, Yingwei Jin, Yong Zhang, and Wenyu Qu. A novel reputation computation model based on subjective logic for mobile ad hoc networks. In *Future Gener. Comput. Syst.*, volume 27, pages 547–554, 2011. https://doi.org/10.1016/j.future.2010.03.006.
- [LMAG⁺15] Paul Lajoie-Mazenc, Emmanuelle Anceaume, Gilles Guette, Thomas Sirvent, and Valérie Viet Triem Tong. Efficient distributed privacy-preserving reputation mechanism handling non-monotonic ratings. 2015. https://hal.archives-ouvertes.fr/hal-01104837/document.
- [LNZ⁺16] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol

for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 17–30. ACM, 2016. https://doi.org/10.1145/2976749.2978389.

- [LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. In *ACM Trans. Program. Lang. Syst.*, volume 4, pages 382–401, 1982. http://doi.acm.org/10.1145/357172.357176.
- [Maz15] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. Draft, Stellar Development Foundation, 2015. https://www.stellar.org/papers/stellarconsensus-protocol.pdf.
- [MB17] Malte Möser and Rainer Böhme. The price of anonymity: empirical evidence from a market for bitcoin anonymization. In *J. Cybersecur.*, volume 3, pages 127–135, 2017. https://doi.org/10.1093/cybsec/tyx007.
- [MMSH16] Patrick McCorry, Malte Möser, Siamak Fayyaz Shahandashti, and Feng Hao. Towards bitcoin payment networks. In Information Security and Privacy 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I, volume 9722 of Lecture Notes in Computer Science, pages 57–76. Springer, 2016. https://doi.org/10.1007/978-3-319-40253-6_4.
- [mon14] Monero, 2014. https://getmonero.org.
- $[MPJ^{+}13a]$ Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Damon McCoy, Geoffrey M. ill Levchenko, Voelker, Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. In loqin Usenix Maq.volume 38. 2013. https://www.usenix.org/ publications/login/december-2013-volume-38-number-6/ fistful-bitcoins-characterizing-payments-among.
- $[MPJ^+13b]$ Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, ill Levchenko, Damon McCoy, Geoffrey M. Voelker, A fistful of bitcoins: Stefan Savage. Characterizing payamong men loqin Usenix ments with no names. In Maq.volume 38, 2013. https://www.usenix.org/ publications/login/december-2013-volume-38-number-6/ fistful-bitcoins-characterizing-payments-among.
- [MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In 40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA, pages 120–130. IEEE Computer Society, 1999. https://doi.org/10.1109/SFFCS.1999.814584.

[MSH⁺18] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. In *Proc. Priv. Enhancing Technol.*, volume 2018, pages 143–163, 2018. https://doi.org/10.1515/popets-2018-0025.

- [MXC⁺16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 31–42. ACM, 2016. https://doi.org/10.1145/2976749.2978399.
- [Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009. http://www.bitcoin.org/bitcoin.pdf.
- [NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 305–320. IEEE, 2016. https://doi.org/10.1109/EuroSP. 2016.32.
- [Noe15] Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. https://eprint.iacr.org/2015/1098.
- [OO14] Diego Ongaro and John K. Ousterhout. In search of an understandable consensus algorithm. In 2014 USENIX Annual Technical Conference, USENIX ATC '14, Philadelphia, PA, USA, June 19-20, 2014, pages 305-319. USENIX Association, 2014. https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro.
- [oST15] National Institute of Standards and Technology. Secure hash standard (shs), 2015. https://doi.org/10.6028/NIST.FIPS.180-4.
- [PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf.
- [Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In 10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA, volume 124 of LIPIcs, pages 60:1–60:15. Schloss Dagstuhl Leibniz-Zentrum für Informatik, 2019. https://doi.org/10.4230/LIPIcs.ITCS. 2019.60.

[PJ16] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. Technical report, 2016. https://doi.org/10.17487/RFC7914.

- [Que17] Jeffrey Quesnelle. On the linkability of zcash transactions. *CoRR*, abs/1712.01210, 2017. http://arxiv.org/abs/1712.01210.
- [RH11a] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011, pages 1318–1326. IEEE Computer Society, 2011. https://doi.org/10.1109/PASSAT/SocialCom.2011.79.
- [RH11b] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011, pages 1318–1326. IEEE Computer Society, 2011. https://doi.org/10.1109/PASSAT/SocialCom.2011.79.
- [Rik] Sveriges Riksbank. Payments in sweden 2019. https://www.riksbank.se/globalassets/media/rapporter/sa-betalar-svenskarna/2019/engelska/payments-in-sweden-2019.pdf.
- [Ros11] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. CoRR, abs/1112.4980, 2011. http://arxiv.org/abs/1112.4980.
- [RS13] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In Financial Cryptography and Data Security 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers, volume 7859 of Lecture Notes in Computer Science, pages 6–24. Springer, 2013. https://doi.org/10.1007/978-3-642-39884-1_2.
- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Commun. ACM*, volume 21, pages 120–126, 1978. http://doi.acm.org/10.1145/359340.359342.
- [shi19] Shielded coinbase, 2019. https://zips.z.cash/zip-0213.

[SMP87] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge proof systems. In Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, volume 293 of Lecture Notes in Computer Science, pages 52-72. Springer, 1987. https://doi.org/10.1007/3-540-48184-2_5.

- [SMZ14] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In Financial Cryptography and Data Security 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers, volume 8437 of Lecture Notes in Computer Science, pages 457–468. Springer, 2014. https://doi.org/10.1007/978-3-662-45472-5_29.
- [STV⁺16] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 526–545. IEEE Computer Society, 2016. https://doi.org/10.1109/SP.2016.38.
- [SZ13] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. https://eprint.iacr.org/2013/881.
- [Sza05] Nick Szabo. Bit gold. 2005. https://unenumerated.blogspot.com/2005/12/bit-gold.html.
- [Tay13] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In International Conference on Compilers, Architecture and Synthesis for Embedded Systems, CASES 2013, Montreal, QC, Canada, September 29 October 4, 2013, pages 16:1–16:10. IEEE, 2013. https://doi.org/10.1109/CASES.2013.6662520.
- [TE18] Itay Tsabary and Ittay Eyal. The gap game. In Proceedings of the 11th ACM International Systems and Storage Conference, SYS-TOR 2018, HAIFA, Israel, June 04-07, 2018, page 132. ACM, 2018. https://doi.org/10.1145/3211890.3211905.
- [vis15] Visa inc. at a glance, 2015. https://usa.visa.com/dam/VCOM/download/corporate/media/visa-fact-sheet-Jun2015.pdf.
- [vS13] Nicolas van Saberhagen. Cryptonote v 2.0. 2013. https://cryptonote.org/whitepaper.pdf.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III, volume 11478 of Lecture Notes in Computer Science, pages 379-407. Springer, 2019. https://doi.org/10.1007/978-3-030-17659-4_13.

- [WJT20] Pieter Wuille, Nick Jonas, and Anthony Towns. Bip-341. taproot: Segwit version 1 spending rules, 2020. https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.
- [Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 2014. http://gavwood.com/paper.pdf.
- [YKDV19] Jiangshan Yu, David Kozhaya, Jeremie Decouchant, and Paulo Jorge Esteves Veríssimo. Repucoin: Your reputation is your power. In *IEEE Trans. Computers*, volume 68, pages 1225–1237, 2019. https://doi.org/10.1109/TC.2019.2900648.
- [ZMR18] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 931–948. ACM, 2018. https://doi.org/10.1145/3243734.3243853.

List of Publications

- 1. Alex Biryukov and Daniel Feher. Portrait of a miner in a landscape. In IEEE INFOCOM 2019 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 May 2, 2019, pages 638–643. IEEE, 2019. https://doi.org/10.1109/INFCOMW. 2019.8845201
- 2. Alex Biryukov and Daniel Feher. Privacy and linkability of mining in zcash. In 7th IEEE Conference on Communications and Network Security, CNS 2019, Washington, DC, USA, June 10-12, 2019, pages 118–123. IEEE, 2019. https://doi.org/10.1109/CNS.2019.8802711
- 3. Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in zcash. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1795–1811. ACM, 2019. https://doi.org/10.1145/3319535.3345663
- 4. Alex Biryukov and Daniel Feher. Recon: Sybil-resistant consensus from reputation. In *Pervasive and Mobile Computing*, volume 61, page 101109, 2020. https://doi.org/10.1016/j.pmcj.2019.101109