UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2022-013
The Faculty of Science, Technology and Communication

DISSERTATION

Defence held on 25/02/2022 in Esch-sur-Alzette
to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

by

Tong CAO

Born on 16 July 1990 in Shaanxi, China

ANALYZING THE PRIVACY AND SECURITY OF PROOF-OF-WORK CRYPTOCURRENCIES

Dissertation defence committee

Dr. Marcus VÖLP, dissertation supervisor
Professor, University of Luxembourg

Dr. Maria POTOP-BUTUCARU
Professor, Sorbonne Université

Dr. Gilbert FRIDGEN, Chairman
Professor, University of Luxembourg

Dr. Jérémie DECOUCHANT
Assistant Professor, Technische Universiteit Delft

Dr. Peter Y. A. RYAN, Vice Chairman
Professor, University of Luxembourg

Acknowledgements

I would like to thank my advisors: Paulo Esteves-Veríssimo, Jiangshan Yu, Jérémie Decouchant, and Marcus Völp. I am fortunate to work with you. Thanks for your guidance and support. I would like to thank my wife, Ting Wang, for her love and patience during my PhD study.

Declaration

I, Tong Cao, declare that this thesis titled, *Analyzing the Security and Privacy of Proof-of-Work Cryptocurrencies* and the work presented therein are my own. I confirm that:

- this work was done wholly or mainly while in candidature for the degree ;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;
- where I have consulted the published works of others, these are clearly attributed;
- where I have quoted from the works of others, the sources are always given;
- where the work presented in this thesis is based on work done by myself jointly with others, I have clearly outlined what was done by others and what I contributed;
- with the exception of such quotations, this is entively my own work; and
- I have acknowledged all main sources of help.

Signed:

Date:

Abstract

In the past decade, we have witnessed the growth of cryptocurrencies. Nowadays, these currencies have generated significant impact in economy and society. Their main innovation relies on the fact that they combine many techniques (e.g., cryptographic primitives and fault tolerance methods) to decentralize online payment system. While cryptocurrencies have been developed since 2009, evaluating their privacy and security still remains challenging due to the complexity of the systems and the interplays between their different layers. In this thesis, we focus on proof-of-work (PoW) based cryptocurrencies (e.g., Bitcoin, Ethereum, Monero). We contribute three novel studies to improve the understanding of the privacy and security of PoW cryptocurrencies. Our first two studies focus on the impact of the network layer. First, we show that network properties are important to the system's privacy and security. We analyze PoW cryptocurrency peer to peer networks and characterize the impact of network properties on privacy and security. Precisely, we design tools to measure the Monero peer to peer network, and conduct experiments to reveal that this network was vulnerable to some network level attacks despite its strong use of cryptographic primitives in the consensus layer. Second, we define new metrics to link the network layer and consensus layer, which allow us to evaluate the system's security by considering network delays. We design a tool to measure the Bitcoin peer to peer network, and provide an empirical analysis. We conduct experiments to evaluate the impact of network delays on the consensus layer. As a consequence, we show that network delays have a significant impact on the system's security. For instance, introducing relatively small network delays already simplifies double spending and selfish mining attacks, allowing adversaries to receive significant gains. We propose an abstraction, which we call dual private chain (DPC), in our third study that further challenges Bitcoin's security and its consensus layer in the presence of Byzantine, Altruistic, and Rational nodes that are practical. We design a Markov Decision Process model for dual private chain attacks, and conduct Monte Carlo based simulations to evaluate the adversary's success rate and revenue. Our results indicate that dual private chain attacks are more threatening to Bitcoin's security compared to traditional temporary block withholding attacks. We suggest countermeasures to mitigate the effects of such attacks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the age of Internet, many attempts [1, 2, 3] have been made to decentralize the online payment system before 2008. The main objective is to avoid the single point of failures that have happened in traditional financial systems [4, 5, 6]. However, they failed because the Byzantine faults could not be tolerated in the asynchronous and permissionless network environment, which could lead to double-spending attacks [7]. In 2008, Satoshi Nakamoto proposed a peer-to-peer electronic digital cash system: Bitcoin [8] (the market capitalization of Bitcoin has grown rapidly and reached \$1.15T at the time of writing[1]), which can tolerate Byzantine faults in the asynchronous networks where everyone can join or leave under the assumption of that the majority of computing power ($>50\%$) are honest. In terms of fault tolerance, the main innovation of Bitcoin is its consensus protocol, named Nakamoto Consensus (NC) that is a combination of many rules (e.g., proof-of-work, incentive mechanism, the longest chain principle, and difficulty adjustment algorithm).

Despite Bitcoin's success, not only in its market capitalization, but also in the fact that the system has not met any collapse since 2009 (v.s., the well-established centralized system, Google was inaccessible due to some severs were collapsed[2]), many vulnerabilities have been reported, mainly referring to privacy and security. At network layer, it has been clearly shown that Bitcoin is not privacy-preserving because the transactions are linkable and traceable [9, 10]. Moreover, Bitcoin is vulnerable to Border Gateway Protocol (BGP) hijacking attacks [11], eclipse attacks [12, 13], and Man-In-The-Middle (MITM) attacks [14], which could partition the network. Previous works [15, 16] have shown that such network vulnerabilities could be used by adversaries to compromise system's security. At consensus layer, the temporary block withholding attacks [17, 15, 18, 19] have been proposed as

[1] https://coinmarketcap.com/, accessed: 2021-10-17.
[2] https://hourstv.com/google-server-crash/

the main treats to Bitcoin's security.

Although many solutions have been proposed to mitigate existing vulnerabilities, guaranteeing a high level privacy and security in Bitcoin or Bitcoin-like cryptocurrencies remains challenging. In particular, it is hard to have a perfect solution in both network and consensus layer. For instance, inspired by Bitcoin, Monero [20] was built to preserve user's on-chain privacy (i.e., the ability to prevent the transactions from being linked and traced) by leveraging the cryptographic primitives.

In [21], our results indicated that Monero had serious privacy issues despite of strong on-chain unlinkability and untraceability. Furthermore, while previous works have discussed the network delay upper bound that guarantees the consistency of Nakamoto consensus, measuring the actual network latencies and evaluating their impact on miners or pools in Bitcoin remain open questions. Additionally, it is unclear how the network delay can affect existing temporary block withholding attacks.

The goal of this thesis is to provide a theoretical and empirical analysis on PoW cryptocurrencies' privacy and security by considering both network and consensus layers. In particular, we focus on the impacts of network properties (e.g., network topology, information propagation delay, and connectivity) on the consensus layer.

1.1 Thesis Outline

This thesis is organized as follows:

- Chapter 2 presents the background knowledge and related work. We first introduce the basic data structures of Bitcoin, the infrastructures of peer-to-peer network, and main mechanisms of Nakamoto consensus. We then discuss the related literature, such as, network measurements, network level attacks, temporary block withholding attacks, and so on.

- Chapter 3 demonstrates the existing network level privacy and security issues in cryptocurrencies. We discuss the existing solutions and highlight the remaining problems. We provide the first empirical study on the first privacy-preserving cryptocurrency's (i.e., Monero) P2P network. We design the novel tools to measure Monero's P2P network, and indicate the network topology exposure issue, which can further facilitate some network level attacks, and lay down its privacy and security.

- Chapter 4 provides the first theoretical and empirical analysis about the impact of network delay on Bitcoin mining. We first define some novel metrics to quantify the impact of network delay on the mining process. We then

design a novel tool, namely Miner Entanglement (ME), to measure Bitcoin's P2P network. Our results reveal the relationship between the block reception latency, block interval, and hash power. We conduct the Monte Carlo simulation to validate our results. Lastly, we analyze the impact of network delay on existing temporary block withholding attacks using an updated Markov decision process model.

- Chapter 5 proposes a novel attack, namely dual private chain (DPC) attack, to further lay down the system's security. We transform the traditional Markov decision process model to Markov chain tree model, which allow us to use the random walk method to evaluate the adversary's revenue based on earned blocks and double spent transactions. We highlight that DPC attack is more threatening compared to traditional temporary block withholding attacks.

- Chapter 6 concludes the thesis. We present some mitigation strategies and insights as the future work.

1.2 Contributions and Publications

This thesis first looks into the performance of cryptocurrency networks in the real world, and points out that IP address leakage is an open privacy issue. The contributions are as follows:

- To further understand the cryptocurrency network, we designed a tool set to measure and analyze the Monero network. The results indicate that even though Monero is a privacy-preserving cryptocurrency, it is still possible to accurately discover nodes in the network and their interconnections.

- Our analysis provides insights about Monero's degree of centralization, and about the privacy and security issues potentially caused by a network topology exposure.

We then explores the impact of network delay on the mining process. The contributions are as follows:

- We establish several metrics to characterize the impact of heterogeneous network latencies on the mining process.

- We design ME, a tool that leverages Bitcoin mining pools' API to estimate their block reception latency. We use ME to monitor the Bitcoin network for an entire week, and quantify the impact of network latency on the revenue distribution using our metrics.

- We discuss the situation where miners would decide to deviate from the official peer-to-peer protocol and try to optimize their connections and their revenue by connecting directly to other mining pools. We show that it would lead to a Nash equilibrium among the largest miners, while other miners will have a decreased revenue.

Lastly, we analyze the impact of network delay on existing temporary block withholding attacks, and propose a novel attack. The contributions are as follows:

- The relatively low network delay can facilitate the existing block withholding attacks. The adversary's advantages would be further increased if the mining difficulty decreases.

- Dual Private Chain attacks. We present two mining strategies to enable dual private chain (DPC) attacks. DPC attacks enable an attacker to launch double spending attacks with less mining power (e.g., 17.5%) and with improved revenue. Unlike existing selfish mining style attacks, DPC attacks can be successfuly launched even when the mining difficulty remains the same.

- Modelling attacking strategies with multiple private chains. We present the Markov chain tree (MCT) model to simulate attacks with (1) multiple private chains and (2) a time parameter, that cannot be analysed with the existing models. We further analyse our proposed strategies and attacks by using random walk model to show their effectiveness.

- Impact on Bitcoin's security. Bitcoin's security is measured as a proportion of the mining power that is controlled by correct miners that are expected to altruistically follow the default protocol. However, assuming altruism is not realistic in decentralized settings. It is difficult to prevent miners' rational behaviors. Some studies [22, 23, 24] have noticed and reported the presence of petty compliant miners, also called benign dishonest miners, in Bitcoin. These miners slightly deviate from the default Bitcoin protocol, for example by doing SPV mining [25]. We show, for the first time, that these SPV miners, despite having no intentional will not only be halted by the Blockchain denial-of-service (BDOS) attacks [24], but also to harm the system or other miners (when DPC attack is taken into account), do lay down Bitcoin's security.

A list of publications and ongoing submission with co-authors and acknowledgements is presented as follows:

- Tong Cao, Jiangshan Yu, Jérémie Decouchant, Paulo Esteves-Verissimo, "Revisiting Network-Level Attacks on Blockchain Network", In the 48th An-

nual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) Workshop on Byzantine Consensus and Resilient Blockchains (BCRB), 2018

- Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, Paulo Esteves-Verissimo, "Exploring the Monero Peer-to-Peer Network", In Proceedings of the 24th International Conference on Financial Cryptography and Data Security (FC), 2020

- Tong Cao, Jérémie Decouchant, Jiangshan Yu, Paulo Esteves-Verissimo, "Characterizing the Impact of Network Delay on Bitcoin Mining", In Proceedings of the 40th International Symposium on Reliable Distributed Systems (SRDS), 2021

- Tong Cao, Jérémie Decouchant, Jiangshan Yu, "Distracting SPV Miners to Double Spend in Bitcoin", submitted to the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) under peer review, 2022

Chapter 2

Background and Related Work

This chapter first introduces the basic concepts and algorithms of PoW mining. We then illustrate the structure of cryptocurrency's P2P network. Moreover, we introduce the main mechanisms of Nakamoto consensus. We discuss the related work at the end.

2.1 PoW Mining

PoW mining is a trial and error process of solving the crypto-puzzle. Miners first select a number of pending transactions from the network, and then bind them into the block as a section of the input of the hashing function 2.1. As long as a valid hash value is discovered (i.e., a block is found), it will be disseminated in the P2P network. We introduce the PoW mining process as follows.

2.1.1 Block Structure

Blocks are chained by the hash algorithms. As shown in Fig. 2.1, each block includes a header, a block hash, and a number of transactions. Transactions are hashed and included into the root of Merkle tree as a part of the header. Miners create a Coinbase [26] transaction to receive the block reward, which is often used to identify the ownership of blocks [27, 28, 29]. The block header is the input of mining algorithms, which consists of software's version, the hash of previous block, Bits (i.e., mining difficulty), the root of Merkle tree of all included transactions, Nonce (i.e., a rolling number for trial and error), and an Unix time stamp (see Bitcoin.wiki [30] for the details about block header). The output of mining algorithms is the block hash.

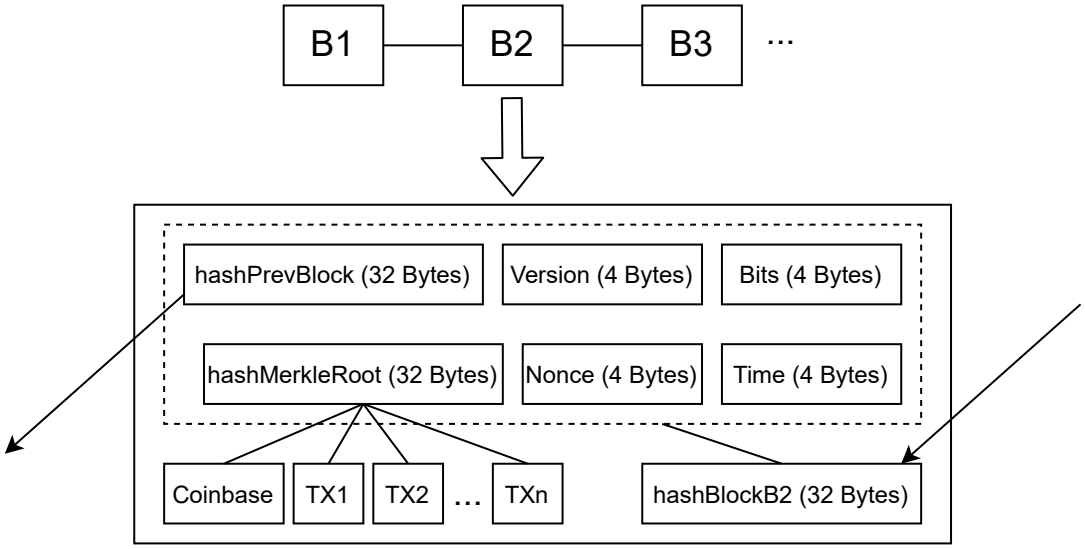


Figure 2.1: The block structure of Bitcoin.

2.1.2 Block Hashing Algorithm

Mining is a process of trial and error. Miners use their processors to execute different cryptographic hash functions to make guesses. In Bitcoin, each action of guessing is an execution of the double SHA256 algorithm 2.1, which creates a 256 bits hash value. A block is discovered when the obtained hash value is smaller than the target hash value. If not, the nonce is changed ($nonce = nonce + 1$, the range of $nonce$ is from 0 to 2^n, e.g., $n = 32$ in Bitcoin) so that the execution of the double SHA256 algorithm can make a different guess. The block hashing algorithm can be described as below:

$$\begin{aligned} &SHA256(SHA256(Version||hashPrevBlock|| \\ &hashMerkleRoot||Time||Bits||Nonce)) \leq target \end{aligned} \tag{2.1}$$

Here, we point out that, section $hashMerkleRoot$, $Time$, and $Nonce$ of the block header are changeable. When the 32 bits Nonce runs dry and the puzzle has not been solved, miners can change $Time$ or $hashMerkleRoot$ to start another 32 bits round. In this case, miners can avoid to produce the hashes that they have tried. This derives the mining to a negative hypergeometric process.

2.1.3 Mining as a Negative Hypergeometric Process

Parra-Moyano J. et al. [31] were the first to describe Bitcoin mining as a NHP (negative hypergeometric process). The NHP (N, M, r) is used to model the number of trials that are needed to have r successes when choosing items without

replacement from a population that totally has N items of which M items are the successes [32]. PoW mining process is a special case of NHP (N, M, r) when $r = 1$. For instance, in Bitcoin, the population size N is equal to 2^{256}, and the number of successes M is equal to $\frac{2^{224}}{D}$, where D denotes $Difficulty$. Therefore, the probability of solving the puzzle at the k^{th} guess P_k can be defined as:

$$P_k = \frac{k}{2^{32} \times D} \tag{2.2}$$

where, $P_k \in [\frac{1}{2^{32} \times D}, 1]$. The puzzle must be solved when the network has produced $2^{32} \times D$ hashes.

2.1.4 Mining as a Poisson Process

Most of works assume that PoW mining is a Poisson process [8, 33, 34, 35], where the success rate λ is approximately equal to 1 block per 10 minutes. By using the Poisson probability density $P(X = x) = \frac{\lambda^x \times e^{-\lambda}}{x!}$, the core properties of Bitcoin can be evaluated. For instance, Nakamoto S. used the Poisson probability density to estimate the number of confirmations needed to prevent double spending attacks in Bitcoin [8]. Decker C. and Wattenhofer R. analyzed more than 10k blocks, and revealed that the block interval distribution is approximately an exponential distribution, which reflected that the mining is a Poisson process [34]. However, strictly speaking, PoW mining is not a Poisson process because: first, the events do not occur independently; second, the probability of solving the puzzle of each hashing is different. For instance, if a block was not found in 10 minutes, it will have a higher chance to be found in the next 10 minutes. Bowden R. et al. challenged the assumption of Bitcoin mining being a Poisson process, and proved that mining is an non homogeneous Poisson process, rather than standard Poisson process [36]. In this thesis, we follow the assumption that mining is a Poisson process because the NHP can be approximated to the Binomial distribution when M and N are big enough so that the replacement can not significantly affect the probability of success in each round ($\frac{M}{N} \approx \frac{M}{N-1} \approx \frac{M}{N-2} \approx ... \approx \frac{M}{N-y}$), y is the threshold that NHP can be approximated to Binomial distribution). The Binomial distribution $X\ B(n, p)$ can be approximated to Poisson distribution with the success rate $\lambda \approx n \times p$ when n is big enough and p is closed to 0 [37]. Recall that $N = 2^{256}, M = \frac{2^{244}}{2.18 \times 10^{13}}$, which are big enough to approximate the NHP to Binomial distribution, and then to the Poisson distribution.

2.2 PoW Cryptocurrency Network

Inspired by Bitcoin [8], various proof-of-work (PoW) cryptocurrencies have been designed, such as Ethereum [38], and CryptoNote-style cryptocurrencis [39]. A

modern PoW cryptocurrency network consists of a P2P overlay and a set of mining pools. The former is designed to disseminate messages between different participants in a P2P fashion. The latter allows miners to obtain a regular revenue for their mining participation by sharing the block finding reward, and usually works as a client-server infrastructure.

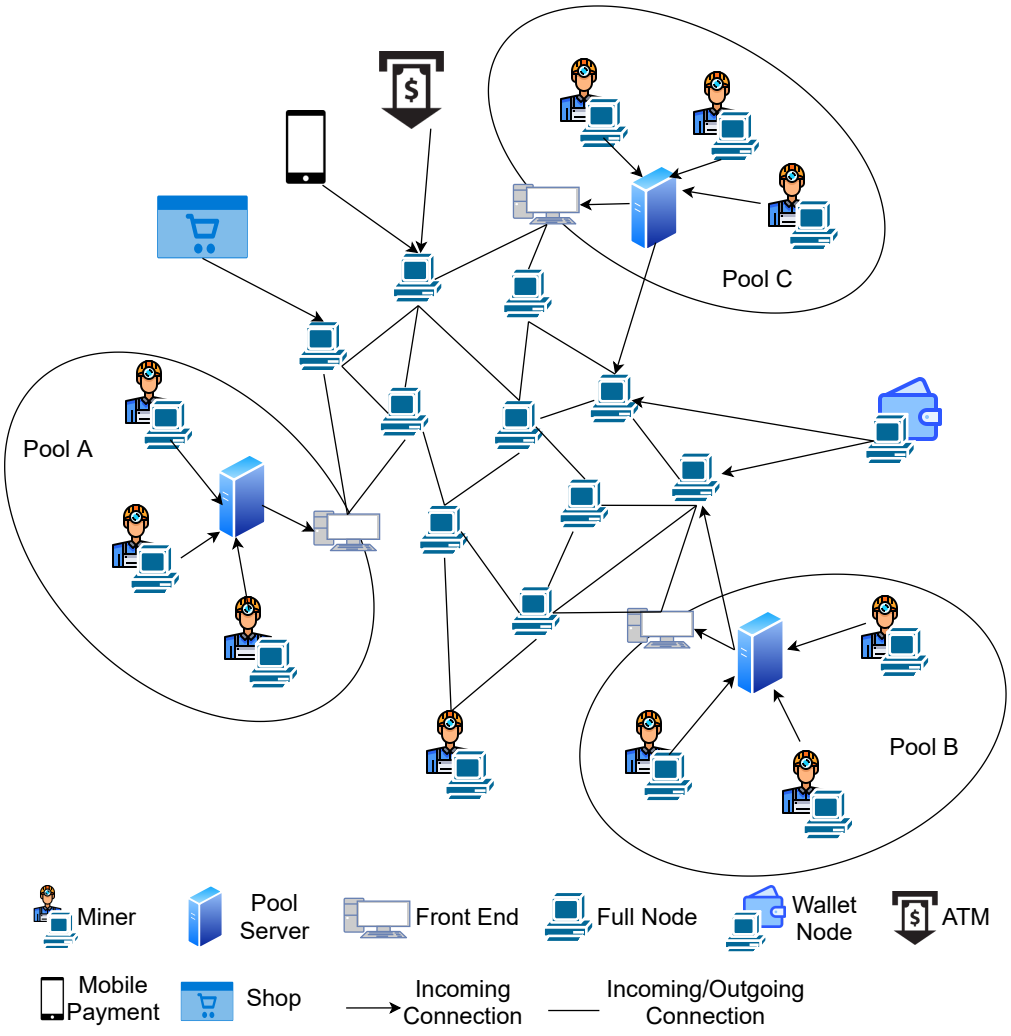


Figure 2.2: A general structure of PoW cryptocurrency network.

2.2.1 Network Overview

The network of PoW cryptocurrencies is permission less. Any computing device is able to join the network by establishing the TCP/IP connections. Initially, Nakamoto S. proposed that the network consists of nodes who play the same roles including peer discovery, transactions relay, and mining. In this case, it is very close to "one CPU, one vote", which assures a high degree of decentralization. However, the network has evolved over the past decade. The most significant change is the pooled mining, which was created to gather the miners to work on the same target, and share the revenue. By doing so, miners can get the stable revenue from the pool depending on their contributions. Nowadays, mining pools control the majority of computing power, for instance, the top 15 mining pools control more than 85% computing power in both Bitcoin and Ethereum, for the rest of computing power, they are likely the anonymous pools. The solo mining is now almost impossible. Therefore, the roles of nodes have been reassigned. As shown in Fig. 2.2. The network consists of a P2P overlay and some intra networks of pools. In the P2P overlay, full nodes are randomly distributed, which are open for being connected by any participant. The intra network of mining pools is a typical client-server network, where the centralized pool server manages the sub-miners. The front end node is deployed by mining pools to exchange messages with other participants in the P2P overlay. Wallet node is normally the user that only make transactions. Other clients nodes, such as, ATM, mobile devices, and payment system of the shops, are connected to some full nodes in the network in order to relay the transactions to miners. We conclude the roles of different nodes as follows:

- **Pools**. The mining tasks are carried by pools (and a tiny percentage of solo miners). In the intra network of mining pools, Pool Sever is used to manage the sub-miners, which builds a client-server infrastructure. In general, the URL of pools is public, miners can join the pool by establishing the TCP/IP connections with the pools. However, miners are not reachable from the outside of corresponding pools. This client-server infrastructure guarantees security for mining pools;

- **Full nodes**. The full nodes are responsible for exchanging messages, they are connected via P2P protocols (which we will introduce in Section. 2.2.2). These nodes are open to accept the incoming connections. Any participant can connect with them to receive the information of peer memberships, blocks, and transactions. The new joiners can use them to synchronize the state of the blockchain. The anarchy of P2P overlay is the core of decentralization of the system, which is mainly maintained by the full nodes. We explore the properties of P2P overlay network, and discuss the implications

and impacts in Section 3 and 3.3;

- **Wallet nodes and client nodes**. The wallet nodes and client nodes (ATM, Mobile Payments, and Shop) are mainly used to generate transactions;

- **Front end nodes**. As long as the block is found, it has to be sent to others in order to achieve consistency (we discuss this property in Section 2.3). The block propagation time is important here. The pools are willing to broadcast their blocks as fast as possible. Therefore, pools deploy a number of full nodes as their front end nodes in the P2P overly to speed up the block propagation. To differ with the Pool Server, the front end node is only used to exchange the messages between pool and P2P overlay, it does not manage the sub-miners. To prevent network attacks, such as, DDOS attacks, eclipse attacks [12], and BGP hijacking attacks [11], the IP addresses of front end nodes are normally anonymous. Moreover, pools can have a lot of front end nodes in the network.

.

2.2.2 P2P Overlay Network

A cryptocurrency network relies on the P2P scheme to have peers exchange messages without a centralized authority. Each node maintains a peer list, and periodically exchanges peer information with others to keep it up-to-date.

Peer-to-peer (P2P) networks have been designed and extensively studied to allow decentralized message exchanges. They have been getting a renewed attention since Satoshi Nakamoto described Bitcoin in 2008. Indeed, inspired by Bitcoin, thousands of cryptocurrencies serving different purposes have been created. However, no standard P2P protocol has been proposed for blockchains. Instead, different P2P protocols have been designed and adapted by different cryptocurrencies [8, 38, 39].

Here we highlight the main characteristis of the P2P protocols deployed by the most popular cryptocurrencies, namely Bitcoin, Ethereum, and Monero, to discuss their design differences. More precisely, Bitcoin's P2P protocol represents Bitcoin-like cryptocurrencies, Ethereum's P2P protocol represents Ethereum-like cryptocurrencies, and Monero's peer-to-peer protocol represents CryptoNote-based cryptocurrencies. Our analysis is summarized in Table 2.1.

Bitcoin [8] nodes can each maintain up to 8 outgoing and 117 incoming bidirectional connections with other nodes. A node can join the network by requesting a set of IP addresses of peers to a list of DNS nodes. After the initialization phase, nodes use a "getaddr" message to request "addr" answers containing up to 2,500

Table 2.1: Comparison of the Bitcoin, Ethereum and Monero P2P protocols

	Network Discovery Mechanism	**Connection Maintenance**	**Node Connectivity**
Bitcoin	Exchange up to 2,500 peer IP addresses.	Using tried table and new table to maintain peer information.	Limited to 125 connection including 8 outgoing and 117 incoming connections.
Ethereum	RLPX, Kademlia Peer Selection Protocol.	Using a peer table to maintain peer information.	Limited to 25 connections for Geth. Limited to 50 connections for Parity.
Monero	Exchange 250 peer IP addresses.	Using peer list to maintain peer information.	Changeable limit on the number of connections. By default, 1 incoming and 8 outgoing connections.

peer information from their neighbors, and then save the received peers information into their local database, which is formed by a tried table and a new table. The tried table keeps the information of all connected nodes, and the new table records the information of the unconnected nodes. If a node does not have 8 outgoing connections, it will randomly select new peers from the local database. New connections can be established if a node allows extra incoming connections. Bitcoin's peer-to-peer network is an overlay network based on the Internet, broadly speaking, it utilizes TCP port 8333 to provide the specific channels for its participants.

Ethereum [38] uses a Kademlia-style peer discovery protocol [40] to build the network. It implements another protocol named RLPx[1], which can encrypt messages, to transfer data. Each node can maintain 25/50 (25 for Geth users, 50 for Parity users) connections in total. Multiple Ethereum nodes can be implemented in one machine to share the same IP address. Ethereum's network is far more complex than Bitcoin's due to its multiple protocols. On the other hand, Ethereum's overlay network topology is not directly relevant to the network-level properties since the nodes' IDs do not respectively correspond to IP addresses. A previous work [41] provided a comprehensive study about measuring Ethereum's clients, but the network-level properties, such as connectivity, topology, and network dynamics

[1] `https://github.com/ethereum/devp2p/blob/master/rlpx.md`.

were not studied in detail.

Monero [39] uses a dedicated TCP port 18080 for its participants' communications. Each node can join the network by connecting to a list of seed nodes and DNS nodes. After the successful connection establishment, the node will receive up to 250 IP addresses from the seed/DNS node. By default, each node can maintain 1 incoming connection and 8 outgoing connections. To enrich the network connectivity, Monero allows users to modify the number of their connections. Based on the connections, each node frequently sends the top 250 IP addresses from its white list to its neighbors. The most unique design of Monero's peer-to-peer protocol is the database of recorded peer information, named peer list, which we will detail in the next section. The goal of the peer list is to help each participant to establish sufficiently enough reliable connections to prevent network level attacks. In Monero, the network dynamics are mainly due to the unlimited node connectivity and unique design of the peer list.

Because the cryptocurrency P2P network is decentralized, previous works have pointed out that the cryptocurrency P2P network is vulnerable in terms of privacy [42, 9, 43, 44, 45, 21], security [12, 11] and scalability [46, 16]. However, none has analyzed the mining fairness in the network.

2.2.3 Intra Network of Mining Pools

To adapt to their growth, PoW cryptocurrencies regularly increase the mining difficulty. This increased difficulty directly translates into a more irregular revenue for single miners. To compensate this effect, miners can join mining pools. Mining pools allow single miners to put their reward in common, which is then shared according to a miner's participation. A mining pool uses some servers to manage the miners in the pool. The miners are only connected with the servers, and receive its assigned jobs from the servers. When a miner has finished a job, it sends its result to the server. The mining pools also maintain the front-end nodes in the P2P network to exchange messages with other miners and pools. At regular intervals, the pool sends the payout to the miner depending on the jobs it has processed. The most popular protocol used by mining pools is Stratum [47]. In our real world experiment, we found that most of Bitcoin mining pools are using Stratum. The purpose of a mining pool is to attract miners to join to increase its hash power, so the mining pool has to be permission-less to the miners for being connected. The connections between mining pool and the miner can be easily established as long as the miner knows the URL of mining pool server. We summarize the URL of the measured 10 Bitcoin mining pools in Table 2.2.

Table 2.2: URL of Bitcoin mining pools. We use a machine to build connections with these pools, this allows us to deploy the sub-miners in the pools to receive the messages directly.

Kano pool	stratum+tcp://sg.kano.is:3333
f2pool	stratum+tcp://btc.f2pool.com:3333
BTC.com	stratum+tcp://cn.ss.btc.com:1800
ANTpool	stratum+tcp://ss.antpool.com:3333
Poolin	stratum+tcp://btc.ss.poolin.com:1883
Huobi pool	stratum+tcp://hk.huobipool.com:8888
Okex pool	stratum.okpool.me:3333
1thash pool	stratum.1thash.btc.top:8888
viabtc pool	btc.viabtc.com:3333
btctop pool	stratum.btc.top:8888
Novablock pool	stratum+tcp://btc.s.novablock.com:443

2.2.4 Block Propagation

In a PoW cryptocurrency, the consensus relies on the quick propagation of a block in the P2P overlay. The miner, or a pool, updates its mining process when it discovers a new block or receives it from others. Given that the expected time interval between two blocks is equal to 10 minutes in Bitcoin, the Nakamoto consensus can be achieved if more than 50% hash power can reach agreement within 10 minutes. Previous works [34, 46, 48] have proposed that the 50% block propagation time was around 8.7 seconds before 2017, which therefore secures the Bitcoin system. According to the Bitcoin network monitor [49], the 50% block propagation time has been improved to less than 2 seconds today. However, no mechanism has been implemented to make sure that all nodes receive the new block within close times, in particular because of the presence of some unreachable nodes, and network churn.

In this thesis, we focus on the block propagation rather than the transaction propagation. We measure the deviation between pools by using a novel tool ME, and report the results in Section 4.

2.3 Nakamoto Consensus

This section introduces the main mechanisms and properties of Nakamoto consensus.

2.3.1 Incentive Mechanisms

Miners are incentivized to validate transactions and find blocks in exchange for monetary rewards. The rewards include block reward and transaction fees. The block reward is made by the miner via a Coinbase transaction, which generates a number of coins for being sent to the miner. The number of coins in Coinbase transaction is controlled by a block reward scheme, which is halved every four years. At the beginning, each block generates 50 Bitcoin, at the time of writing, each block generates 6.25 Bitcoin. According to this scheme, approximately, there will be no block reward after 2140, which means, no new coin will be generated. Transaction fee is an alternative reward that miner can earn from the traders. To discourage the tiny transactions, Bitcoin allows the generator of transaction to pay a number of coins ($< 2\%$ of mining revenue) to the miner. As a consequence, the transactions with higher fees have the priority to be validated. There are a number of traders who are willing to pay higher fees to let their transactions to be validated more faster [50].

In fact, every miner creates the Coinbase transaction and selects other transactions, and then bind them into the root of Merkle tree for solving the puzzle. Only the winner can receive the rewards. Under the assumption of honest majority, only the valid blocks will be eventually appeared in the main chain, this incentivize the miners to select correct transactions. The invalid transactions or blocks will eventually rejected by the network. This is so called incentive compatibility of NC, which preserves an equilibrium in the mining game. Any miner who deviates from the default setting will suffer a loss.

However, the incentive compatibility of NC has not yet been formally proven [22]. Miner who deviates from NC could use smaller than 50% of global computing power to manipulate the main chain with a high success rate [17, 51, 15, 18, 52]. In the past few years, a number of Non-NC PoW protocols [53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68] have been proposed to improve the weaknesses that have been revealed in Bitcoin. However, no one is perfect to solve all problems that NC has. Zhang R. and Preneel B. [69] made a quantitatively analysis of these PoW protocols, and concluded that existing PoW protocols can not guarantee the perfect properties at same time, such as, attacks resistance, fairness, and scalability.

In this thesis, we focus on the network properties, which are typically in layer 0 (as shown in Fig. 2.3), and could affect the consensus protocols in layer 1. In a nutshell, we design tools to explore the network, characterize the network properties, and analyze the impact on layer 1. We report our results based on NC, which might affect other PoW protocols as well.

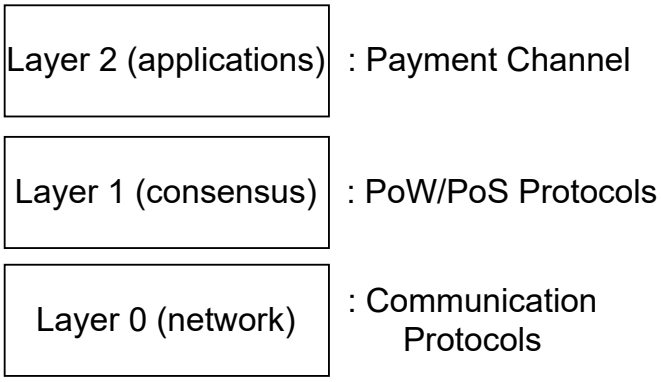


Figure 2.3: The model of blockchain system.

2.3.2 The Longest Chain Principle

In an asynchronous network, the traditional Byzantine Agreement is hard to be achieved in the permission less setting. NC solved this problem by incentivizing the miners to work on the longest chain. In this case, there is no need for the temporary agreement among all participants, which means, forks are acceptable under a threshold. Under the assumption of honest majority, all participants will eventually agree on the same chain, which is the longest. Nakamoto S. made a simple calculation in Bitcoin white paper [8], which shows that the attacker with $q \in [0, 0.5)$ of global computing power will eventually lose the race if the attacker keeps her chain private. Later on, Eyal [17] pointed out that the attacker is able to use $< 50\%$ of global computing power to manipulate the main chain by using a temporary withholding strategy. In this thesis, we point out that the threshold could be further decreased by considering the network properties.

2.3.3 Difficulty Adjustment Algorithm

To stabilize the block generation time, DAA (Difficulty Adjustment Algorithm) is used to adjust the difficulty of solving the crypto-puzzle. The adjustment occurs periodically according to the time of the period (2016 blocks in Bitcoin, 1 block in Ethereum). The maximum block generation time can be estimated as:

$$Max(d_{BI}) = \frac{D \times 2^n}{H} \tag{2.3}$$

where D notes the difficulty, H represents the total hash rate of the network, and n is a coefficient (e.g., it is equal to 32 in Bitcoin). When H is changed because of churn, DAA will adjust D to guarantee the stability in terms of block generation.

The DAA function of Bitcoin is described as follows:

$$D_{n+1} = D_n \times \max(\min(\frac{2016 \times 600}{T_{2016\times n} - T_{2016\times(n-1)}}, 4), \frac{1}{4}) \tag{2.4}$$

The DAA function of Ethereum is:

$$D_{n+1} = D_n + \frac{D_n}{2048} \times \max(1 - \frac{T_n - T_{n-1}}{10}, -99) + int(2^{\frac{n}{100000}-2}) \tag{2.5}$$

2.3.4 Revenue Fairness

Previous works [22, 46] considered the mining network to be fair when a miner having $x\%$ of the global mining power earns $x\%$ of blocks. This property is also defined as chain quality [51, 70, 69, 71]. Eyal was the first to point out that the revenue fairness is not hold if the attacker uses a temporary withholding strategy to mine [17], namely selfish mining attacks, which we discuss in Section 2.5. In this thesis, we analyze this property by considering:

- Temporary withholding strategy is not the only factor that can affect the revenue fairness. Existing mechanisms aiming to provide this property do not consider the impact of the P2P network. We characterize mining network fairness in Section 4, and prove that the deviation of block reception latency could affect revenue as well.
- The network advantage (e.g., fast send and receive the blocks) can further facilitate the selfish mining attacks.

2.3.5 Asynchrony

The network of PoW cryptocurrencies is permission less, which means, the number of participants is changeable. Thus, it is impossible to make sure that all participants can be synchronized within a bounded delay. Even though the system is asynchronous, it can achieve the eventual consistency, and tolerate up to 50% faulty computing power (33% for selfish mining attacks). This is the beauty of the PoW consensus. In this thesis, we point out that, the asynchrony of the network could lead to network unfairness, which could affect the block revenue.

2.3.6 Consistency

This property is the base of the distributed system, which is ability that the agreement can be made between all or partial participants periodically. The period here means the time interval between two events that need to be voted by all participants, which is bounded by the maximum network delay. In general, consistency can be strong or weak depending on the requirements of the system:

- Strong consistency. It requires all participants to agree on every event in a bounded time window. For instance, any message m has to be sent to all participants within t. It requests the strong fault tolerant protocols that can make the agreement despite of a number of faulty nodes. Moreover, it is sensitive to the network delay. Therefore, the network is normally a complete graph or a star graph.

- Weak consistency. It does not require all participants to always maintain the same data at the same time. Instead, different participants are allowed to have the deviations at some moments, and they will agree on the same data eventually. The strong consistency normally comes with low performance and availability. Therefore, weak consistency is commonly applied in the distributed systems.

In PoW cryptocurrencies, consistency is formally defined as the ability that can enable the participants to agree on a chain of blocks. The strong consistency is difficulty to be achieved here due to the network is being permission less. It is hard to confirm the exact number of participants in the network, and the maximum delay between all participants. Therefore, PoW cryptocurrencies provide weak forms of consistency, such as, eventual consistency across nodes. Pass R. et al. [70] proposed $T-consistency$ to describe the consistency property in Bitcoin. A system achieves $T-consistency$ when all participants can always have the common prefix (n blocks) of the chain at the $(n+1)^{th}$ round despite of folks. They were the first to use the partially ordered set to describe the chain of block. This method has been widely used to define the eventual consistency in PoW cryptocurrencies [72, 71].

2.4 Proof of Work vs Proof of Stake

PoW consensus is energy intensive, it requires miners to use computing power to solve the cryptographic puzzles, which consume electricity. For instance, Bitcoin approximately consumes 61.76 terawatt-hours (TWh) of electricty per year (0.28% worldwide electricity consumption per year), which has exceeded many countries, such as Switzerland (58.46 TWh). PoS based consensus [73] was proposed to mitigate this issue. The idea is to generate blocks by using voting power, which relies on the deposits of participants, namely stake. Here, the participants do not need to consume computing power to discover blocks, the leading block is automatically generated by a participant who is pseudonym-randomly selected depending on her stake per round. Indeed, PoS based consensus can significantly reduce energy consumption, but it comes with some issues, such as, nothing-at-stake attacks [74], temporary block withholding attacks [75, 76]. In this thesis, we focus on PoW cryptocurrencies.

2.5 Related Work

This section discusses the related researches that have been done in this field.

2.5.1 Measuring Cryptocurrency Network

Measuring cryptocurrency P2P network is an important technique to evaluate the performance of cryptocurrency in the real implementation. In general, the metrics are network size, connectivity, and node distribution, which can be inferred by collecting the peer information. In P2P level, each participant exchanges the peer information with her/his neighbors. In principle, each participant is able to find all participants sooner or later depending on the size of exchanged peer information and number of neighbors. Coinscope [43] was proposed in 2015 to measure Bitcoin network by leveraging the peer's API. The main idea is to frequently send the "get addr" messages to all known peers to find more unknown peers, and then establish the connections with new peers and repeat the former approach. As a result, the change of newly discovered peers will converged, and the network size can be inferred. Kim S.K. et al. [41] used the same method to measure the Ethereum P2P network. These works provided the crucial study and tell people how the Bitcoin and Ethereum network look like, in particular, they discussed the implications of discovered network properties and provided their insights, which inspired a lot of researches to analyze the security and privacy properties in network level.

Based on the global view of the network, more metrics can be measured, such as, transaction propagation time, and block propagation time. The idea is to observe the difference of receiving time of each transaction/block message between all participants. It is therefore that the message propagation time can be estimated. Decker C. and Roger W. [34] were the first to measure the information propagation in Bitcoin in 2013. They deployed the nodes to establish connections with all reachable nodes in Bitcoin network, and then recorded the receiving time of each messages from each peer. By removing the delay between the collectors and their peers, they can estimated the transaction and block propagation time. Today, this method is still the main technique to measure the transaction and block propagation delay in cryptocurrency. Till N. et al. [48] have monitored Bitcoin network for years based on this method.

Miller A. [43] was the first to infer bitcoin's network topology based on the updating scheme of time stamp in the "addr table" (is database to store IP addresses of peers). This method was then prevented by Bitcoin's update. Later on, Till N. [48] provide a new method to infer Bitcoin's network topology. To infer whether two reached nodes are connected in Bitcoin, Grundmann et al. [44] suggested to use double spent transactions as probing messages, and S. D. Segura et al. [45] suggested to use orphan transactions. In this thesis, we propose a novel tool set

in Monero, which can accurately infer the network topology, and implicate more security and privacy issues.

Previous works studied the network information of leading cryptocurrencies, e.g., Bitcoin and Ethereum. Decker and Wattenhofer [34, 46] measured the rate of information propagation between reached nodes in Bitcoin. Relying on the received messages from reached nodes, interconnections of reached nodes were inferred in Bitcoin [43, 48] to evaluate network properties. To infer whether two reached nodes are connected in Bitcoin, Grundmann et al. [44] suggested to use double spent transactions as probing messages, and S. D. Segura et al. [45] suggested to use orphan transactions. Kim et al. [41] deployed 30 nodes on one machine to collect network messages and measure the Ethereum network. However, node interconnections are difficult to infer in Ethereum, and unreachable nodes cannot be observed.

Previous works [43, 48] deployed machines into P2P overlay networks to collect information regarding block propagation, transaction propagation, and network coverage. The results have been adopted widely to analyze the performance of cryptocurrencies. For example, block propagation delay can be used to evaluate the transaction throughput [46]. However, there is a lack of direct information of mining pools. Such results can be improved by monitoring mining pools. By doing so, the discovery time of a block can be accurately inferred, which is more reliable than the time stamp[2] of block, which could be earlier than the discovery time. Recently, a tool is designed by a research team at MIT [77] to monitor 32 pools across 17 cryptocurrencies. However, they do not consider the network fairness. In this thesis, we design a tool to measure and analyze network fairness.

2.5.2 Network Level Attacks

Network level attacks have been studied in Bitcoin and Ethereum. Routing attacks [11, 78] are facilitated by the fact that Bitcoin's protocol makes nodes exchange messages in plain text during the peer-to-peer communications. This allows an adversary to partition the network, and delay the dissemination of messages among nodes. Eclipse attacks, in Bitcoin [12, 15] and in Ethereum [13], pointed out that unsolicited incoming connections can be leveraged by an adversary to continuously send large amounts of fake packets to a given node to fill the table of its stored IP addresses and force it to restart. These attacks demonstrated that an attacker can monopolize all connections of a targeted node with high probability. Deanonymization attacks [9, 79] have been introduced to track transactions and discover the generator's IP address. These attacks aim at linking the IP address of a node with the transactions it creates, with the intention of monitoring inter-

[2]https://en.bitcoin.it/wiki/Block_timestamp

connections. Such attacks require, or are facilitated, by an understanding of the peer-to-peer overlay and its topology.

2.5.2.1 Routing attacks

Due to the highly centralized autonomous systems in the Bitcoin network, malicious messages can be injected into one autonomous system to announce incorrect IP prefixes, which leads to the network traffics going into wrong locations. It makes the Border Gateway Protocol (BGP) hijacking attack [11] possible in the Bitcoin network, where an attacker can delay the information propagation and partition the Bitcoin network in order to waste mining power, or spend one coin more than once. The eclipse attack was described in 2015 [12], where unsolicited incoming connections are used by an attacker to send bogus information to a victim to force it to restart. After that, the attacker can monopolize all 125 connections of victim and, with a high probability, control the end host.

2.5.2.2 Deanonymisation attacks

Fanti and Viswanath [9] and Biryukov A. [79] described Deanonymisation attacks, which aim at disclosing the IP address of nodes that generate transactions, even those located behind a Network Address Translation (NAT). Since each client node has 8 entry nodes, and the generated transactions of the client are always first forwarded to its 8 entry nodes, it is possible to identify the entry nodes of a client node. In addition, some approaches rely on the Bitcoin relay pattern, which can be used to identify the IP address of a transaction creator, i.e., the payer, based on the following observations. First, a node that was the first forwarding a transaction is likely to be the payer. Second, a node is likely to be a payer if it re-transmits the transaction. Moreover, a node is likely to be a mining pool if the generated blocks from the pool were relayed frequently and firstly via that IP address during 10 days period.

2.5.3 Fast Relay Network

The purpose of the fast relay network [80] is to maintain full Bitcoin nodes to connect with miners and pools to speed up the message propagation. These nodes use unsolicited relay patterns to broadcast a new block. In this way, miners are able to speed up the blocks propagation in the network. However, this method leads to a potential centralization issue. So far, not all mining pools have adopted this technology, maybe because their access is controlled. Our work provides a motivation for mining pools to improve their connectivity, and explains the growing success of more efficient relay networks. The P2P strategy we evaluate in this thesis

(Miner Entanglement) improves a miner's network access without relying on fast relay networks using mining pools' APIs. In other words, our measurement tool does not need a trusted node in the middle between two pools.

2.5.4 Game Theoretical Analysis in Cryptocurrency

Game theoretical analysis has been used to evaluate cryptocurrency' network performance and other properties. For example, it has been shown that a Nash equilibrium can be achieved when two pools, or any number of pools use the block withholding attack [81]. In this case, pools aim to attack each other like the prisoner's dilemma, and degrade system performance. Previous works [82, 83] provide the game-theoretic models to analyze the impacts of network attacks (such as DDoS) in mining pools. Another example concerns the network creation game, which has been introduced in the lightning payment channel [84], where it was shown that the Bitcoin payment network can be more stable and efficient in a centralized network structure. In our thesis, we are interested in how nodes select their neighbors to optimize its effective hash rate, and consequently their expected mining revenue.

2.5.5 Petty-Compliant Behaviors

In decentralized system, miners are rational rather than altruistic, i.e., they will deviate from the default protocol if they can increase their benefits. In fact, it has been observed that there are many miners that do not completely follow the default protocol. These miners have been called petty-compliant [23, 24]. In this thesis, we consider two types of petty-compliant miners, which we introduce in the following.

Hash power jumping miners. These miners [85, 86] aim at optimizing their profits. In general, they are not limited to mining a single cryptocurrency, and instead maintain a computing power portfolio that spans several cryptocurrencies to against the risks. They are monitoring the price of different coins, and estimating the revenue shares that they can obtain from different chains. The hash power of a rational miner would be shifted from $\mathbb{C}_A$ to $\mathbb{C}_B$ if mining the latter is more profitable.

Simplified Payment Verification (SPV) miners. SPV miners leverage the Simplified Payment Verification (SPV) protocol, which was initially introduced to allow Bitcoin nodes to accept a block based on its header without verifying its transactions. Upon receiving a block header, SPV miners start mining the next block, a process known as SPV mining. In doing so, SPV miners assume that the transactions of a block will soon be received and that they are correct. Under normal circumstances, SPV miners can start mining on a block earlier, because

block headers are smaller than full blocks and are therefore received sooner, which helps them to increase their mining effectiveness [87]. SPV miners might deploy spy sub-miners in the mining pools to receive block headers directly from them, which is arguably faster than the block propagation in the peer-to-peer overlay network [88]. Observing blank blocks (i.e., blocks that only include one coinbase transaction) is often proposed as a way to detect SPV miners [25]. However, SPV miners could also include their own transactions into a block, which would not conflict with any previous transaction and complicate their detection. In this work we present the DPC attack that targets SPV miners to facilitate double spending.

2.5.6 Double Spending and Selfish Mining

The double spending attack on Bitcoin has been described in Nakamoto's initial whitepaper [8], and has been further analyzed since [7, 19]. An adversary with at least 51% of the global mining power is able to use a coin in a transaction that is accepted by the network and have a conflicting transaction later accepted by the network. Nowadays, $z = 6$ blocks need to be appended after a block for its transactions to be considered permanent by all Bitcoin users. Nakamoto characterized the race between the attacker and the honest miners as a random walk, and calculated the probability for an attacker to catch up with the public chain after z blocks have been appended after its initial transaction.

Selfish mining was the first mining strategy to be shown to increase the revenue of a rational miner [17]. Selfish mining was later shown to harm the mining fairness [22, 46]. It has been shown that the selfish mining strategy is not more profitable than honest mining when the mining difficulty remains constant, despite the fact that the adversary is able to increase its revenue share [89, 16]. Nayak et al. proposed plausible values for the selfish miner's propagation factor by utilizing the public overlay network data [15]. They pointed out that the attacker could optimize its revenue, and win more blocks by eclipsing the honest miners when the propagation factor increases. Gervais et al. [16] analyzed the impact of stale rate on selfish mining attack, and described the eclipse attack [12]. Negy et al. pointed out that applying the selfish mining strategy in Bitcoin is profitable after at least one difficulty adjustment period (i.e., after approximately two weeks at least) [90].

2.5.7 Combining Selfish Mining and Double Spending

Previous works [16, 91] have shown that it is feasible to combine the double spending attack with selfish mining. To do so, the adversary uses the selfish mining strategy to maintain the private chain, and generates conflicting transactions whenever the private chain is longer than the public chain. The double spending attempts

would succeed if the adversary has enough private blocks in the lead of the public chain. Since generating the conflicting blocks would not cause any additional cost for the adversary, the adversary's revenue would be increased. However, based on a single private chain, the selfish mining actions (e.g.,"adopt", "match", and "override") actually decrease the success rate of the double spending attacks. For instance, when the adversary has 2 private blocks in the lead of the public chain and the next block is generated by the honest miners, the adversary would release her private blocks, and the double spending attempt has to be terminated. Considering that 6 confirmations are required commonly, the success rate of double spending attacks on the selfish mining's private chain is actually lower than the pure double spending attacks based on a single private chain. We propose the first mining attack that simultaneously manages two private chains to launch double spending attacks.

2.5.8 Blockchain denial of service (BDOS) attacks.

The BDOS attack proposed strategies to completely/partially shut down the mining network [24]. To do so, the adversary only sends the block header to the network whenever she discovers a block that is in the lead of the public chain and there is no fork, and publishes the block body if the next block is generated by the honest miners. By doing so, the profitability and utility of the rational miners and SPV miners would be decreased, thus, they will eventually leave the mining network. However, the objective of BDOS attacks is to halt the system, which does not bring a direct benefit to the adversary. For instance, an adversary would need to spend approximately 1 million USD per day to shut down the system. Moreover, it is difficult to achieve a complete shut down if there are stubborn honest miners. In this chapter, our DPC attack frequently separates the SPV miners' hash power from the one of altruistic miners, which has some similarities with the BDOS attack's partial shut down case. However, the DPC attack is profitable for the adversary when its second private chain manages to double spend.

Chapter 3

PoW Cryptocurrency P2P Network

This chapter first provides a comprehensive study on the security and privacy of PoW cryptocurrencies. Precisely, we first introduce the basic concepts of security and privacy of layer 0 alongside with the open challenges. Secondly, we analyze the existing schemes on enhancing the security and privacy in network level, and discuss their performances (for those schemes that have not been implemented, we discuss the feasibility and give our insights). Lastly, we provide an empirical measurement study on the first privacy-preserving cryptocurrency (i.e., Monero) to reveal some network vulnerabilities that could facilitate the network level attacks.

3.1 Network Security

The network of PoW cryptocurrencies consists of P2P overlay and intra networks of mining pool, as we have introduced in Section 2.2. Both of them are based on Internet, and supported by TCP/IP. Therefore, the traditional Internet based network attacks are all feasible to the network of PoW cryptocurrencies, such as, DDOS attacks, MITM attacks, and BGP hijacking attacks. In principle, any PoW cryptocurrency's network can be attacked. For instance, the network can be partitioned if one can control a number of ASs (Autonomous Systems) [11]. The network can be dominated if one can deploy a sufficient number of nodes (Sybil nodes [92]) in the network, in this case, the adversary is able to control the speed of information propagation by dropping the messages.

The essential motivation of attackers is to gain the monetary benefits. In general, the attacks are restricted by the imbalance between the cost of attacks and the revenue of attacks. Let C_a be the cost of the attack (e.g., the resources that the adversary needs to launch the attack), R_a be the revenue of the attacker (e.g., the attacker earns a number of coins because of an attack). The frequency of that the attack happens is related to $\frac{R_a}{C_a}$. When $\frac{R_a}{C_a} > 1$, the attacker is incentivized

to launch the attack. For instance, a DDOS attack would happen in the mining pool if it satisfies $\frac{R_a}{C_a} > 1$. It does not mean that the attack can not happen when $\frac{R_a}{C_a} \leq 1$. The existing defense mechanisms of PoW cryptocurrencies make an extremely expensive cost for a successful attack, which restricts the attacks. For instance, Heilman E. et al. [12] proposed several schemes to defend eclipse attacks in Bitcoin, which request the attacker to have a vast number of active IP addresses, which is not profitable.

3.1.1 Network Attacks in P2P Overlay

Network attacks and defenses have been studied in P2P network for a long time. Compared to the client-server network, P2P network lacks censorship and governance. There are BAR nodes [93, 94] that aim at optimizing their benefits. They do not follow the default settings, and intend to change the original protocols if they can get benefits. It is hard to prevent BAR nodes in P2P network. Since the main network structure of PoW cryptocurrencies is P2P network, it is not surprising that BAR nodes are detected. For instance, the block hole nodes were detected in Bitcoin to drop the block and transaction messages [95], super nodes were detected in Bitcoin to maintain more than 1000 outgoing connections [43]. The adversary could deploy some BAR nodes in PoW cryptocurrencies to facilitate the attacks. For instance, the succcess probability of eclipse attacks increases when the attacker deploy a number of controlled nodes to continuously ping the victim node. In order to guarantee the network security, it is important to prevent BAR nodes or tolerate the faults because of BAR nodes.

One difference between the traditional P2P network and cryptocurrency P2P overlay is the weight of messages, in precise, the size of packets. The traditional P2P network is able to exchange heavy messages. The communication protocols of PoW cryptocurrencies must be lightweight in order to minimize the impact of network latency on consensus. Thus, some encryption schemes are not applied, such as, SSH [96], TLS of TCP/IP [97], and IPsec [98]. Nowadays, it takes approximately 5 seconds to propagate a 1 Mb block to 90% nodes in Bitcoin network [49], which has already affected the security and performance of Bitcoin [17, 34, 46]. Any heavy communication protocol will increase the network latency that can further decrease the performance of system, and facilitate the attacks. Thus, the messages are not encrypted in the network of PoW cryptocurrencies. The visible network information not only engages the privacy issues (which we show in Section. 3.2), but also facilitates the network attacks (attacker could gather the information to do the statistical analysis before the attack).

We have introduced the typical attacks on the P2P overlay of PoW cryptocurrencies in Section. 2.5, such as, eclipse attacks, BGP hijacking attacks, and MITM

attacks. Here, we point out that, these attacks are not completely forbid. They are still the threats. The adversary will launch these attacks as long as s/he is able to gain a positive revenue. Considering the value of crypto coins is growing rapidly, these network attacks should be further analyzed.

3.1.2 Network Security of Mining Pools

The intra network of mining pools is a typical client-server network (which we introduce in Section. 2.2), where the server node controls the sub-miners of the pool. There are less threats compared to the P2P overlay, because sub-miners could not communicate with each other. The main threats here are:

- DOS attacks on the server nodes. The IP addresses of server nodes are public, which encourage the miners to join. The adversary is able to launch DOS attacks to the server nodes of mining pools. This requires that the server nodes must have the sufficient ability to defend the DOS attacks.

- Packet injection attacks. The communications of intra network of mining pools are based on Internet. Recabarren R. and Carbunar B. [99] pointed out that the adversary is able to modify, inject, and replay the packets between different sub-miners and the server node.

3.2 Network Privacy

The network privacy here means the ability to hide the user's IP address and network topology.

3.2.1 IP Address Exposure

To avoid network partitions and bootstrap the new participants, peer discovery mechanisms are used in the network. Each node maintains a data structure (it is called IP table in Bitcoin, peer list in Monero) to store a number of IP addresses of peers. Each node periodically select a part of IP addresses (1000 in Bitcoin, 250 in Monero) from the data base, and send to its neighbors. When the node does not have enough connections, it can establish connections via the data base. The timestamp based updating scheme is used to evict the inactive IP addresses, and promote the fresh IP addresses.

The IP addresses are exposed during the whole peer discovery process. Anyone is able to deploy a node in the network to collect the IP addresses of others. In the decentralized network, nodes have to use the peer discovery protocols to find each others. The IP address exposure is hard to be prevented.

3.2.2 Network Topology Exposure

The links between nodes can be inferred by different methods. Neudecker T. [100] concluded four approaches to infer the links in Bitcoin. In Section 2.5, we have discussed these approached regarding to their dis/advantages. Here, we emphasize that attacker with sufficient resources (e.g., a large number of nodes in the network) is able to infer the network topology of PoW cryptocurrencies in the current settings. The core question is: will the attacker get enough revenue that can cover her/his cost?

3.2.3 Effects and Implications

With the knowledge of IP addresses and network topology, an attacker can potentially launch different types of attacks. For example, an attacker could launch a targeted attack by monopolizing all connections of a victim node [12], selectively partition the network [11], or even deanonymize transactions by identifying the first node relaying a transaction [79, 101].

3.3 Measuring the Monero P2P Network

As blockchains aim at implementing decentralized and trustworthy systems, they often rely on peer-to-peer (P2P) protocols for membership management and information dissemination. This makes the P2P network a critical element of blockchains, as the security of the underlying consensus protocols and the privacy of transactions are all tightly related to its implementation [12, 13, 11, 22, 79, 101, 102].

Monero is a privacy-centric cryptocurrency, and is currently ranked the first among privacy-preserving cryptocurrencies with a market capitalization of 1.248 Billion USD, and the 10th among all cryptocurrencies[1]. Much research has been done on analysing the privacy of Monero [103, 104, 105, 106, 107], with a focus on on-chain data analysis, i.e., how the mixins (a.k.a. decoy inputs) are selected in each transaction and how they provide privacy guarantees. However, little research has been done to investigate Monero's P2P network, even though network level attacks have been studied on the specific networks of Bitcoin and Ethereum [12, 13, 11, 15, 108, 109].

Analysing the resilience of a blockchain to network level attacks is challenging, as it requires a deep understanding on the underlying network. In this thesis, we present a first step of work towards analysing Monero's security and privacy against network level attacks. In particular, we provide an analysis of Monero's network protocol, and identify possible ways to infer the network topology. We

[1] https://coinmarketcap.com. Data fetched on Sept. 12, 2019.

develop a toolset to implement our findings. Our tool set includes NodeScanner and NeighborFinder. NodeScanner automatically discovers peers in the Monero network, no matter whether they are currently online or not. We classify the discovered peers in three categories, namely active and reachable nodes, active and unreachable nodes, and inactive nodes. A node is active if it is currently online, and is reachable if NodeScanner can successfully connect to it. Compared to previous works [34, 10, 48, 101], NeighborFinder is able to identify the unreachable active nodes in the network, which are the active direct neighbors of nodes that could be reached, for the network topology inference.

Our experimental results show that Monero's network is highly centralized — 0.7% of the active nodes maintain more than 250 outgoing connections, and 86.8% of the nodes do not maintain more than 8 outgoing connections. These 86.8% nodes collectively maintain only 17.14% of the overall connections in the network. Our toolset is also very effective in observing the network – after a single week of data collection, our toolset already discovered 68.7% more active peers than Monerohash [110] – a Monero mining pool that is the only known pool providing data on the Monero node distribution. In average, our toolset identified approximately 2,758 active nodes per day, while Monerohash only showed about 1,635 active nodes. Furthermore, we report our analysis of the collected data regarding an estimation of our network coverage, the network connectivity, and the node distribution in the Monero P2P network.

Disclosure. We have disclosed our research findings to the Monero team, which has been working on patching the peer-to-peer protocol, and publicly acknowledged our findings in their git commit[2].

3.3.1 Monero's P2P membership protocol

Monero relies on its peer-to-peer network to disseminate transactions and blocks. Unfortunately, a proper presentation of Monero's peer-to-peer protocol has been missing from the literature. This section describes Monero's peer-to-peer membership protocol based on its source code, which is available from Monero's official working repository[3].

3.3.1.1 Initialization

Monero hardcodes a set of hostnames, which can be translated to IP addresses through the DNS service, and IP addresses of seed nodes that new participants

[2]https://github.com/monero-project/monero/blob/960c2158010d30a375207310a36a7a942b9285d2/src/p2p/net_peerlist.h

[3]Commit hash 14a5c2068f53cfe1af3056375fed2587bc07d320, https://github.com/monero-project/monero

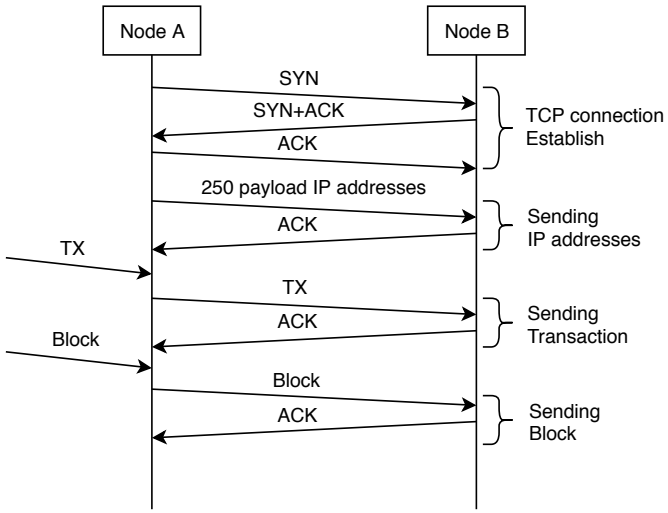


Figure 3.1: Message exchange in Monero's P2P network

can contact to be bootstrapped into the peer-to-peer network. Those seed nodes are operated by the Monero core team.

New joiners can obtain a limited number of active peers' IP addresses from the seed nodes to initialize their peer lists. They can then start initiating connections with peers, exchange membership lists and discover other peers, until they have established their desired number of connections.

3.3.1.2 Peer list

In Monero, each node maintains a peer list consisting of two parts, i.e., a *white_list*, and a *gray_list*. In the peer list of a peer A, the information of each recorded peer not only contains its identity, its IP address, and the TCP port number it uses, but also a special *last_seen* data field, which is the time at which the peer has interacted with peer A for the last time. All the peers in the lists are ordered chronologically according to their *last_seen* data, i.e., the most recently seen peers are at the top of the list.

Each time a node receives information about a set of peers, this information is inserted into its *gray_list*. Nodes update their *white_list* and *gray_list* through a mechanism called "graylist housekeeping", which periodically pings randomly selected peers in the *gray_list*. If a peer from the *gray_list* is responsive, then its information will be promoted to the *white_list* with an updated *last_seen* field, otherwise it will be removed from the *gray_list*. To handle idle connections, nodes check their connections through the IDLE_HANDSHAKE protocol, and update the *last_seen* fields if they successfully connected to the corresponding neighbors, otherwise they drop the associated connection. Nodes also periodically handshake

their current connections, and update the *last_seen* field of the associated responsive peers. If a peer does not respond to the handshake request, then the requesting node will disconnect from this neighbor, and connect to a new neighbor chosen from the *white_list*. The disconnected peers will stay in the *white_list*. The maximum sizes of the *white_list*, and of the *gray_list*, are equal to 1,000 and 5,000, respectively. If the number of peers in these lists grow over the maximum allowed size, then the peers with the oldest *last_seen* fields will be removed from the list.

Nodes broadcast messages (e.g., transactions and blocks) to their neighbors through TCP connections. Nodes choose their neighbors from the *white_list*. If not enough peers from the *white_list* are currently online, then a node will choose its neighbors from the *gray_list*. Nodes to which previous connections were established are classified as anchor nodes, and stored in the *white_list*. Monero ensures that every node is connected to at least two anchor nodes to prevent a node from being isolated by an attacker. To discover other participants, nodes exchange membership messages by sending a TCP SYN message to their neighbors. Upon receiving a SYN message, the neighbors create a message whose payload contains detailed information of its top 250 peers in the *white_list*, and send it back to the requester. The requester inserts the received peer data into its *gray_list*, and runs the graylist housekeeping protocol to update the lists. More details about the TCP connection and data transmission will be presented in Section 3.3.1.3.

3.3.1.3 Information propagation

By default, each peer maintains 8 outgoing connections and accepts 1 incoming connection. A peer residing behind a firewall or a NAT does not accept incoming connections, and only maintains 8 outgoing connections. Peers are allowed to define their maximum number of outgoing and incoming connections. Monero recommends peers to increase the number of their connections according to their capacity, for an improved network connectivity.

To enrich the network connectivity, Monero allows the node to modify the maximum number for both incoming and outgoing connection. And they encourage users to increase their connections limit in order to help to help other nodes to relay messages. For instance, some nodes that operated by mining pools are always maintaining large connections rather than normal client users. In this case, the actual maximum number of connections depends on the capability of the host. Probably, some mining pools implement a Monero node on a well equipped host to reach others nodes as much as possible, then they will gain the messages broadcast advantages to win coins. By learning the source code of Monero peer-to-peer protocol, we found that each node allows 8 outgoing connections and 1 incoming connection by default. For users those behind the firewall or NAT that do not

accept incoming connections, their just allow 8 outgoing connections. Generally, for normal client users, they just want to use the node to transfer the coins or check their balance, they are not willing to reach more other nodes because they do not concern about mining. Therefore, those client users, they are likely to use the system rather than serve it. We believe, such different user's behaviors (mining would like to serve network while client users would like to consume it) cause the inequality of connectivity of nodes in Monero peer-to-peer network, and might lead to its network vulnerability.

Three types of messages are propagated in Monero, respectively containing peers information, transactions, and blocks. As we have mentioned above, each node broadcast the top 250 IP addresses from its peer list to its neighbors, this is called IP addresses propagation. Moreover, each node plays a role to verify the incoming transactions and blocks according to its saved ledger replica, and then relay verified transactions and blocks, this is called transactions and blocks propagation. Whenever a transaction or block is generated, it is finally disseminated among all nodes if it can be verified. A node establishes connections with others through a TCP handshake (SYN-SYN-ACK) as illustrated in Figure 3.1, and can subsequently exchange peer information through the established connection. In this figure, after establishing a connection, node A sends a TCP message to node B whose payload contains the information of the top 250 peers of its *white_list*. where a node A establishes a connection with a node B. Node A initializes the communication by sending a synchronize message SYN to B, and expecting from B an acknowledge message SYN+ACK where the SYN and the ACK bits are both turned on (set to 1) in the TCP header. Upon receiving SYN+ACK, A completes the handshake by sending an acknowledge message ACK to B. SYN messages and ACK messages are indicated by the turned on SYN bit and the ACK bit inside the TCP header, respectively. Periodically, each node inserts the most recent 250 IP addresses of its *white_list* into a TCP message that it sends to its neighbors. In this way, nodes disseminate and update their peer lists. Nodes disseminate the transactions and blocks they received and verified to their neighbors through the TCP connections they established.

3.3.1.4 Monero node

i. Wallet node, it does not need to synchronize the blocks, it relies the remote node to exchange message with other nodes, and it is convenient but less secure than other nodes.

ii. Client node, it just maintains enough outgoing connections and does not accept any incoming connections, this can help the node to avoid malicious incoming connection. The client node always establish outgoing connections to open

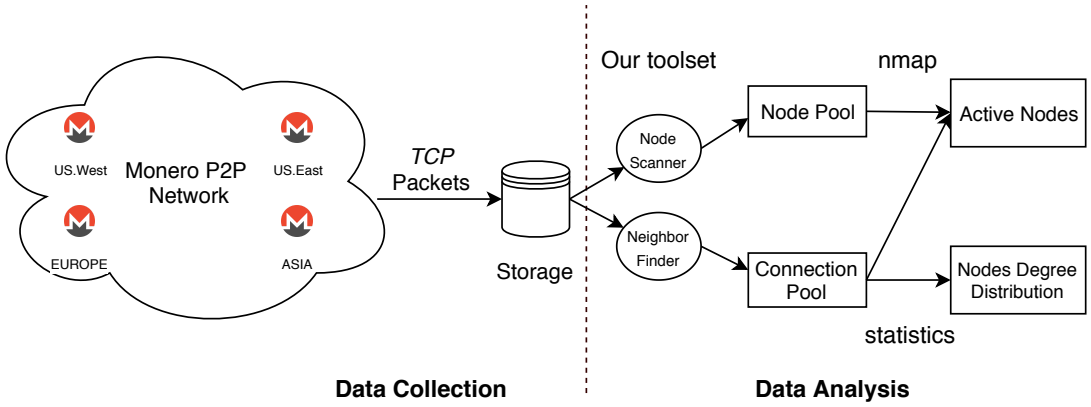


Figure 3.2: Analysis pipeline overview

node.

iii. Open node, in contrast to client node, the open node allows both incoming and outgoing connections, it aims to relay messages in order to increase the network's connectivity, it acts as bridge between client nodes. The open nodes always accepts incoming connections from client nodes and other open nodes, and establish outgoing connection to other open nodes.

iv. Remote node, it is a kind of open node that allows wallet node to use it.

v. Full node, it is actually defined as the node that supports solo-mining.

In this thesis, we classify Monero nodes into heavy node, medium node, and light node depending on the number of their neighbors. Our classifications cover all official types of Monero nodes, for instance, light nodes are likely wallet node and client node, medium and heavy nodes are likely remote nodes, open node, and full nodes. We show details in Section 3.3.3.4.

3.3.2 Analysis pipeline overview

In this section, we introduce the different data structures and the processes we implemented, along with the associated network tools they rely on. We also detail our algorithmic approaches to monitor the active Monero nodes and to infer their neighbors. The analysis pipeline is illustrated in Figure 3.2.

3.3.2.1 Construction

We deploy full Monero nodes to collect data in the Monero network. These nodes establish connections with peers in the network, and store packets into their local

storage. We adapt two network measurement tools, i.e., tcpflow[4] and nmap[5], to collect data and analyze the Monero network. As mentioned in Section 3.3.1, each received TCP packet contains the most recent 250 IP addresses of the sender's *white_list*. Thus, all received IP addresses are recent out-bound peers of the sender. We then use our first tool, NodeScanner, to collect the IP addresses of discovered Monero nodes and store them in the NodePool. We use our second tool NeighborFinder to infer the neighbors of reached nodes that sent the TCP packets to collectors, and store them in the ConnectionPool. Each connection consists of a node we reached and of its neighbor, which are both active. We introduce in greater details our developed tools in Section 3.3.2.3.

3.3.2.2 Neighbor inference based on membership messages

We used two complementary approaches to investigate how to identify a node's neighbors. First, we found that as introduced in Section 3.3.1.2, Monero clients execute a gray list housekeeping protocol and an idle connections prevention protocol to evict inactive nodes from their peer list. As a consequence, the out-bound neighbors of a node are often associated with the freshest *last_seen* in its peer list, which enables the identification of a node's neighbors from the membership messages it sends. Therefore, we conclude that the IP addresses of the out-bound neighbors of the packet sender are included in the TCP packet that are sent to its peers. The function *peerlist_manager*::*set_peer_just_seen* updates the IP addresses in the peer list by updating the *last_seen*, using the instruction $ple.last\_seen = time(NULL)$ by analyzing the Monero source code. This function is triggered when an IP address is promoted from the *gray_list* to the *white_list*, or when a node is connecting with the host.

3.3.2.3 Nodes discovery and connections inference

Our deployed nodes accept incoming connections and initiate outgoing connections to receive TCP packets from other nodes. Let $P = \{P_1, P_2, P_3, ..., P_j\}$ be the set of j TCP packets a collector receives from a reached node, such that each packet P_k ($k \in [1, j]$) contains a set $A_k = \{A_{k,1}, A_{k,2}, A_{k,3}, ..., A_{k,250}\}$ of ***IP*** addresses and a set $T_k = \{T_{k,1}, T_{k,2}, T_{k,3}, ..., T_{k,250}\}$ of *last_seen* timestamps.

NodeScanner. After having received a set P of packets from node $\mathcal{N}$, NodeScanner identifies the set $A = \{A_1, A_2, A_3, ..., A_j\}$ of included IP addresses, extracts the set $U = A_1 \cup A_2 \cup A_3 \cup ... \cup A_j$ of unique IP addresses from A, and inserts all

[4] `https://www.tecmint.com/tcpflow-analyze-debug-network-traffic-in-linux/`.
[5] `https://nmap.org/`.

unique IP addresses into the NodePool.

NeighborFinder. Our second tool aims at identifying a set N_k of neighbors from each P_k ($k \in [1, j]$). Over the various packets P_1 to P_j, it identifies the overall set of neighbors $N = N_1 \cup N_2 \cup N_3 \cup ... \cup N_j$. In the following, we first indicate our neighbors inference approach based on the time difference of the nodes' *last_seen* timestamps in a single packet, and then refine this approach by relying on several received packets.

Neighbors inference based on a single packet. For any received packet P_k from a node $\mathcal{N}$, we assume that it contains $r < 250$ neighbors. Because all neighbors of $\mathcal{N}$ are updated at the same time, the neighbors of $\mathcal{N}$ tend to be the first r adjacent *IP* addresses of A_k, and the difference between any two neighbors' timestamps tends to be small. If we assume that there is a maximum time difference μ[6] between the timestamps of any two neighbors, then we can extract a set $N'_k = \{A_{k,i}, A_{k,i+1}, A_{k,i+2}, ..., A_{k,i+r-1} |\ r \in [1, 250],\ i \in [1, 251 - r],\ \forall x \in [i, i + r - 1], T_{k,x} - T_{k,x+1} \leq \mu\}$ of neighbors from P_k as shown in Algorithm 1.

Algorithm 1: Neighbors inference based on a single received packet

Input : P_k: Packets;
μ: The maximum time difference between the *last_seen* timestamps of a node's neighbors;
A_k: the *IP* addresses of P_k;
T_k: the *last_seen* timestamps of P_k

Output: Neighbors set N'_k;

```
for (y = 1, y < 250, y++) do
    if T_{k,y} - T_{k,(y+1)} ≤ μ then
        N'_k ⟵ A_{k,y}
    end
    if T_{k,(y+1)} - T_{k,(y+2)} > μ then
        N'_k ⟵ A_{k,(y+1)}; break
    end
end
```

Each node iteratively checks its connections through the IDLE_HANDSHAKE procedure, which makes a node send SYN packets to all of its neighbors. Following this procedure, the *last_seen* timestamps of handshaked neighbors are updated with the current time if nodes can be contacted, otherwise connections are dropped. This mechanism prevents idle connections to be maintained. However, the answers

[6] We set μ to the value of the IDLE_HANDSHAKE interval, i.e., 60 seconds.

Algorithm 2: Neighbors inference based on two received packets

Input : Packets P_k and $P_{(k+1)}$;
A_k, $A_{(k+1)}$: the *IP* addresses in P_k, resp. P_{k+1};
T_k, $T_{(k+1)}$: the *last_seen* timestamps in P_k, resp. P_{k+1};
Output: Neighbors set N''_k;

```
foreach A_{k,y} = A_{(k+1),z} do
    if T_{k,y} ≠ T_{(k+1),z} then
        N''_k ⟵ A_{k,y}
    end
end
```

to the SYN packet can be received at a different time, which leads to different answer delays. It is therefore necessary to set μ to a value that is large enough to discover all neighbors, but small enough to limit false positives. This problem only exists when we rely on a single packet to infer the neighbors of a target node, and disappears when multiple packets are used.

Improved neighbors inference based on multiple packets. We have discussed how to infer a node's neighbors using a single packet. Now, we indicate how to improve it by using multiple packets. We have received a set P of packets from node $\mathcal{N}$. During a connection with a node, it frequently happens that our monitoring nodes successively receive multiple packets from a node. If an IP address appears in successive packets, and its *last_seen* has been updated, then we can conclude that the node corresponding to this IP address is a neighbor of the sender. We use the set $N'' = \{A_{k,y} \mid \exists\ y \in [1, 250], \exists\ z \in [1, 250], A_{k,y} = A_{(k+1),z}, T_{k,y} \neq T_{(k+1),z}, A_{k,y} \in P_k, T_{k,y} \in P_k, A_{(k+1),z} \in P_{(k+1)}, T_{(k+1),z} \in P_{(k+1)}\}$ to denote the IP addresses that have been updated between packets P_k and $P_{(k+1)}$. We then extract the neighbors of node $\mathcal{N}$ following Algorithm 2.

3.3.3 Experiments

This section describes our experimental settings, validation approach, data analysis methods and results. We also discuss the potential threats of a network topology exposure.

3.3.3.1 Settings

We deployed four full nodes in the Monero network: two in the U.S. (California and Virginia), one in Europe (Luxembourg), and one in Asia (Japan). Each node ran on an Ubuntu 16.04 machine with an Intel Xeon Platinum 8000 series processor.

We make use of the four nodes not only to collect data, but also to have access to a ground truth and verify our neighbor inference algorithms.

We manually modified the settings on our Monero nodes so that they could establish the largest number of connections with other nodes. First, we set the maximum number of incoming and outgoing connections to 99,999 to force our nodes to actively search for new neighbors. Second, we modified the number of opened files, socket receive buffer, and socket send buffer of used machines to the maximum number (1,048,576, 33,554,432, 33,554,432 respectively) in order to simultaneously maintain a large amount of TCP connections[7].

We collected 510 GB of raw data containing 12,563,962 peer list messages (as shown in Table 3.1). We extracted 21,678 IP addresses, which belong to 970 ASs[8]. Out of these collected IP addresses, our nodes established connections with 3,626 peers, and identified 703 peers to which no connection could not be established, but that were active and connected to reached nodes. We say that peers are active and reachable if our nodes can establish connections with them. We say peers are active but unreachable if they are connected to nodes we connected to and if a connection could not be established with them. We say a peer is inactive if it is neither connected to our nodes, nor connected to responsive peers. If our nodes were not able to connect to a peer, then it either meant that the peer was already fully connected during the data collection, or that it was offline. To reduce the number of possible false negatives, we consider that a peer is offline if the peer is not connected to our nodes or to the neighbors of our nodes, and if their *last_seen* has not been updated during the data collection process.

Table 3.1: Data collected from Tokyo (T), Luxembourg (L), California (C), and Virginia (V)

#Received Peer List Messages	**Node**		**Connection**	
T: 1,971,514; L: 2,308,968	**IP Addresses**	**ASN**	**Host Level**	**AS Level**
C: 3,892,225; V: 4,391,255	21,678	970	338,023	87,013

3.3.3.2 Validation

We used the node in Luxembourg to establish three connections with the nodes in California, Virgina, and Tokyo respectively. We compared the identities of the nodes identified by NeighborFinder as neighbors with the ground truth of our deployed nodes. Since the payload data of membership messages can contain at most 250 IP addresses, a part of a node's neighbors could not be observed in a single

[7]We use tcpflow (a linux network monitoring tool) to capture the TCP packets

[8]We use the whois (https://www.ultratools.com/tools/ipWhoisLookup) database to find the ASN for each IP address.

message when it maintained more than 250 outgoing connections. Therefore, we specifically set up a node maintaining more than 250 outgoing neighbors in Tokyo to verify our algorithms. The validation reported a precision of 100% with 97.98% recall (i.e., all inferred neighbors were real neighbors, and 2.12% of the nodes identified as Non-neighbors were false negatives) when the number of neighbors is smaller than 250, and a precision of 100% with 93.79% recall for the node in Tokyo.

3.3.3.3 Measuring the network coverage

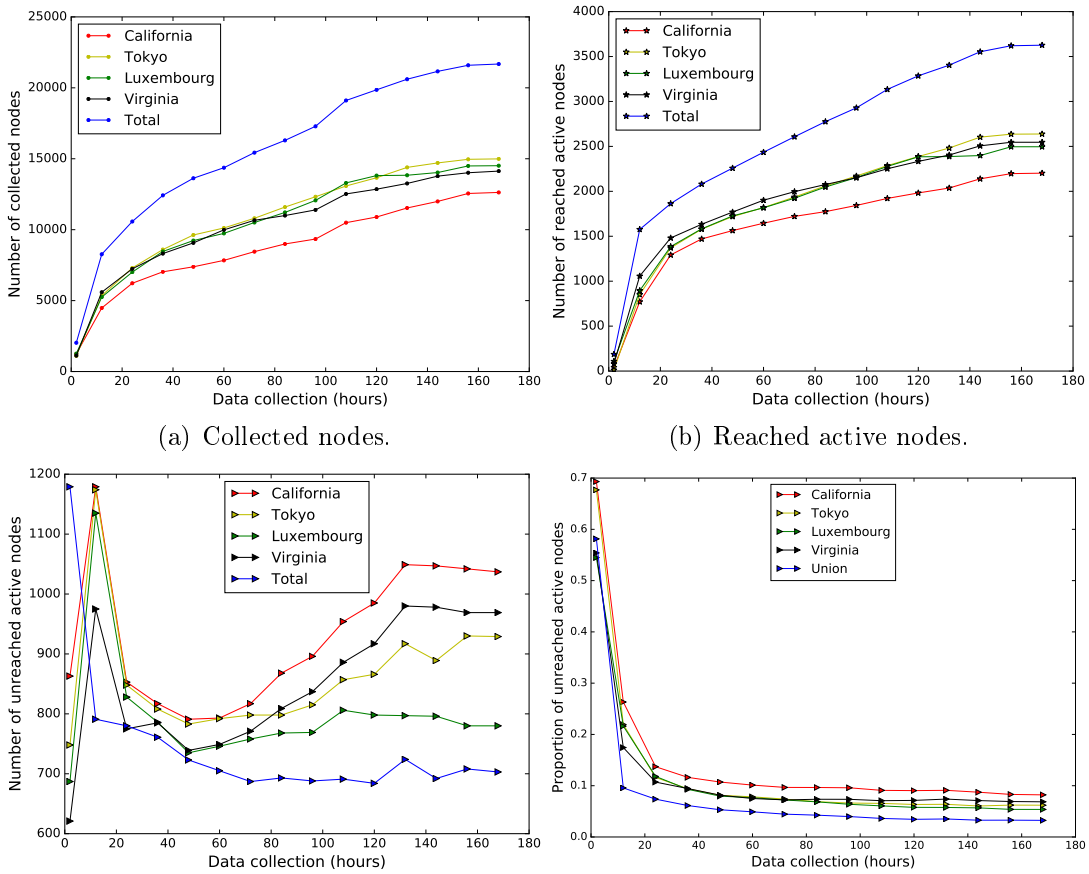
Previous tools [111, 112, 49, 41] relied on the number of reached nodes to estimate their network coverage in Bitcoin and Ethereum. However, unreachable active nodes, which are also a part of the nework, have been overlooked by these tools. In this section, we introduce our method, which takes unreachable nodes into account, to estimate the network coverage. We show the effectiveness of our tools by comparing our results with the data provided by the MoneroHash mining pool [110].

NeighborFinder determined the neighbors of reached nodes even when it was not possible to contact them. This allowed us to:

- **identify the fully connected nodes**. When a node has reached its maximum number of incoming connections, it does not accept any new inbound neighbor. In this case, previous approaches cannot identify these fully connected active nodes. However, NeighborFinder can discover them through the connections they have established with reached nodes.

- **estimate the network size by observing the proportion of unreached active nodes**. Unfortunately, there is no ground truth to validate the network size in permissionless blockchains. We use $\frac{num.\ unreached\ active\ nodes}{num.\ collected\ nodes} \in [0, 1]$ as a metric to estimate the proportion of the Monero network that has been reached. In practice, our tools have discovered almost all long-term running nodes in the network when the new reached nodes cannot present information about any new nodes. The overall proportion of unreached active nodes is illustrated in Figure 3.3(d). When the new reached nodes could not bring the unknown nodes, which means, most of long-term running nodes have been connected, the proportion of unreached active nodes $\frac{num.\ unreached\ active\ nodes}{num.\ collected\ nodes}$ stabilizes. On the other hand, new participants might affect the proportion of unreached nodes mainly due to the unmeasured network churn. For instance, the new joiners obeying the default number of connections could be connected with: our collectors or the reached nodes, which can increase the number of collected nodes or the number of unreached

nodes. However, the new joiners can only present themselves to our collectors as long as most of long-term running nodes have been connected. Thus, this can not decrease/increase the proportion of unreached active nodes too much. In this thesis, the measured network size of Monero is based on the active long-term running nodes and the observed new participants, which could be reachable or unreachable.



(a) Collected nodes.

(b) Reached active nodes.

(c) Unreached active nodes.

(d) Proportion of unreached active nodes.

Figure 3.3: Analysis of the collected IP addresses during the data collection process.

We present the data collection statistics in Figure 4.7, where we respectively show the data collected by the node in California in red, Virginia in black, Japan in yellow, and Luxembourg in green. The total number of reached nodes is represented in blue. Figure 3.3(a) shows the number of discovered peers. Figure 3.3(b) shows the number of active nodes connected to our servers. Figure 3.3(c) shows the number of active but unreachable peers. Figure 3.3(d) shows the evolution of

proportion of unreached active peers. After the first 80 hours, the proportion of unreached active nodes are stabilizing, which means that our toolset has detected almost all the long-term running active nodes. Thus, it is likely that the Monero network contains around 2,758 active nodes per day as shown in Figure 3.4. Compared with Monerohash [110], which discovered 1,635 active nodes in average per day, the number of active nodes we discovered is 68.7% higher than the number reported by the MoneroHash mining pool. To the best of our knowledge, Monerohash is the only Monero mining pool providing information related to the number of active nodes in the network. Moreover, the number of daily active nodes in Bitcoin [111] and Ethereum [112] is estimated to be close to 10,000. It is not a surprise to see that Monero has far less daily active nodes than those two more largely used cryptocurrencies.

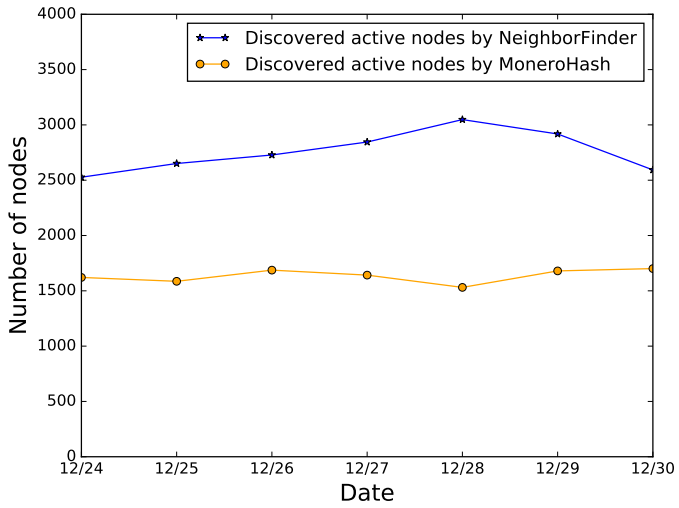


Figure 3.4: Active nodes discovered daily by *NeighborFinder* and MoneroHash.

3.3.3.4 Node distribution

In a cryptocurrency P2P overlay, different nodes play different roles and exhibit various connectivities in a real world implementation. It is essential to analyze how nodes are connected and located in the network to measure the resilience of the blockchain systems to network level attacks, which are surveyed in Section 2.2. In this section, we present the experiment results regarding to peer freshness, connectivity, and node distributions alongside with their implications.

Peer freshness. Our approach shows that only about 20% (i.e., 4,329) of the discovered nodes were active, and the remaining nodes were offline during the data collection period. This indicates that a majority of the exchanged IP addresses

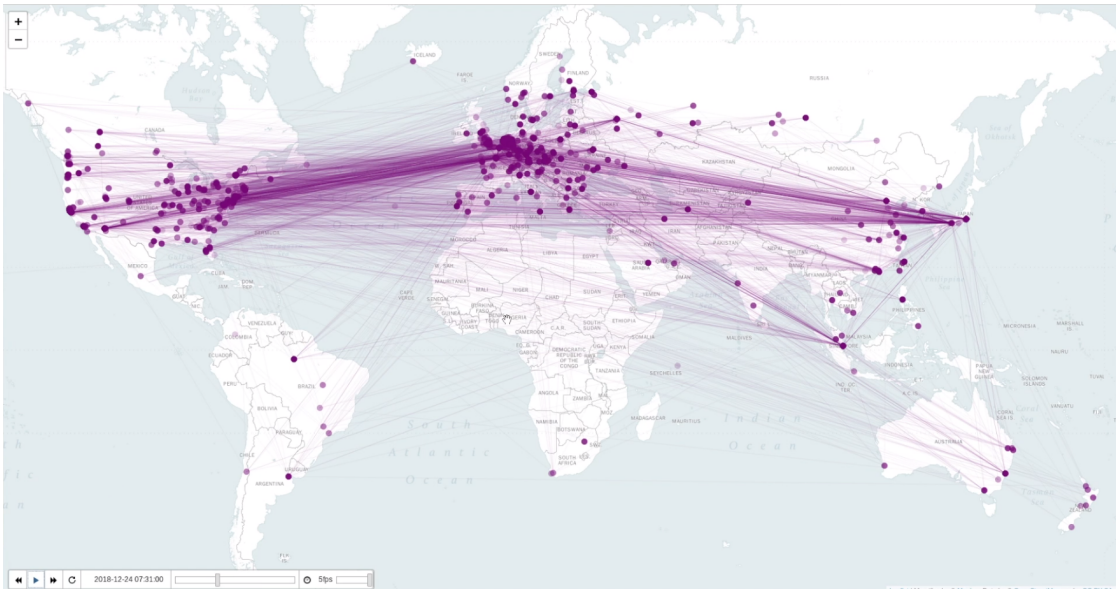


Figure 3.5: Snapshot of the Monero network obtained after one hour. Each dot represents a Monero node, whose darkness is proportional to the number of connections it maintains. The lightness of lines denotes their uptimes.

are inactive in Monero's network, and might decrease the network connectivity.

Connectivity. We say that a node is of degree N if it maintains at most N outgoing connections. We classify active nodes into three categories based on their degree: light node (degree$\leq$8), medium node (8<degree$\leq$ 250), and heavy node (degree>250). As shown in Table 3.2, most of the nodes (86.8%) collectively maintain only 17.14% of the connections, while the remaining 13.2% of the nodes maintain 82.86% of the connections. Here, we point out that a node's degree is influenced by two factors:

- Active factor. A node's degree is mostly controlled by the end user who is allowed to modify the maximum number of established connections. We found that most of the nodes kept the default 8 outgoing connections during one week. Interestingly, a number of nodes started from 8 outgoing connections, and then increased up to 250, which means that they joined the network during our data collection period, and then the users modified their maximum number of connections.

- Passive factor. The capacity of the host machine may limit the maximum concurrent connections a node can maintain. For instance, without the active limit, the front-end node of a mining pool may be able to maintain more than

1,000 concurrent connections.

On the other hand, Monero has hardcoded 8 seed nodes in the system, and we initially suspected that all of them would be heavy nodes. Our experiments showed that only 3 of the seed nodes were active, and that two of them were heavy nodes, while another one was a medium node. Later on, we contacted the Monero team for clarification, and they confirmed that 5 seed nodes were not available[9].By comparing the discovered heavy nodes with public Monero mining pools[10] and seed nodes[11], we found that 9 heavy nodes are maintained by mining pools, and that 2 heavy nodes are Monero seed nodes. Due to the lack of public information, we could not identify the other 17 heavy nodes. However, we assume that the remaining unidentified heavy nodes are likely to be the front-end nodes of private mining pools.

Table 3.2: Number of active nodes in the ConnectionPool.

	Light nodes	**Medium nodes**	**Heavy nodes**	**Total**
Reached	3146 (86.8%)	452 (12.5%)	28 (0.7%)	3626
Unreached	-	-	-	703
Total	3146	452	28	4329

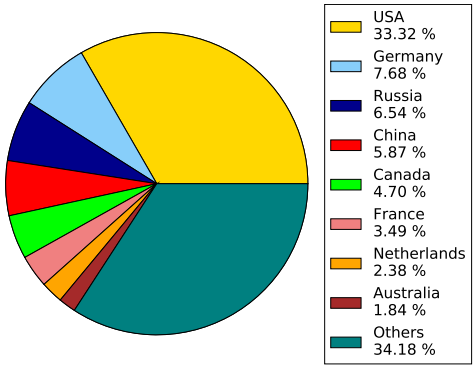
Snapshot of the Monero network topology. We collected snapshots of the network topology thanks to the *ConnectionPool*, which continuously records the connections' updates. Those snapshots provide useful information concerning the network structure. We represent a one hour snapshot of the Monero network topology observed on 12/24/2018 in Figure 3.5. It is obvious that an user's IP address is exposed along with its connections. This leaves a chance for the adversary to identify different roles (miner or client) in the network depending on their connectivity. On the other hand, we hypothesize that the vast inequality of node connectivity (In our experiment, the heaviest node could maintain more than 1000 connections, the lightest node just maintain 8 connections) might lead to network vulnerabilities [113], where the high degree node could significantly affect low degree node to select neighbors.

Geographic distribution. We present in Figure 3.6 the location of the Monero nodes depending on their classification. Approximately 50% of the heavy nodes, which are likely the mining pools, are located in the US, while the light nodes, which are likely clients, are more evenly distributed around the world.

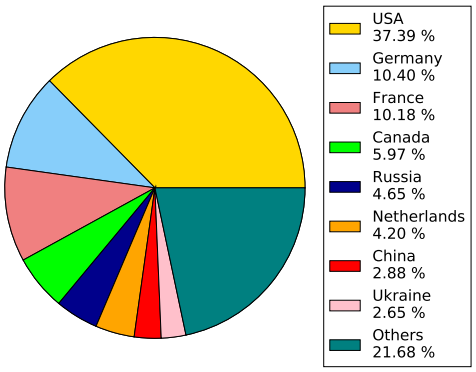
[9] `https://github.com/monero-project/monero/issues/5314`.

[10] `http://moneropools.com/`.

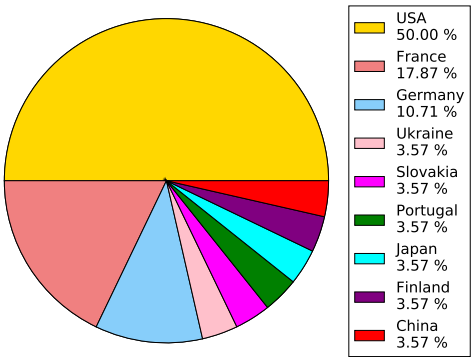
[11] `https://github.com/monero-project/monero/blob/577a8f5c8431d385bf9d11c30b5e3e8855c16cca/src/p2p/net_node.inl`.



(a) Light nodes.



(b) Medium nodes.



(c) Heavy nodes.

Figure 3.6: Nodes location distribution

Degree distribution. Monero's peer-to-peer network is unstructured, permissionless and very dynamic. In particular, a node is allowed to change its neighbors as we analyzed in Section 3.3.1. To further analyze how nodes are connected over time, we counted the numbers of neighbors of active nodes during one week, and plot their distribution in Figure 3.7. The blue dots represent the distribution of outgoing neighbors of the nodes. The results indicate that a small fraction of the nodes have more than 1000 outgoing neighbors, while a large fraction of nodes have less than 100 outgoing neighbors. The red dots represent the distributions of both incoming and outgoing neighbors. Comparing with the blue dots, one can see that the node with a large number of outgoing neighbors are likely to maintain a large number of incoming neighbors as well. More importantly, the small jumps in both blue and red dots indicate that a number of nodes have not kept the number of connections fixed by default in order to gain a better connectivity. We point out that this is an unique feature of Monero, which implies a high network dynamism.

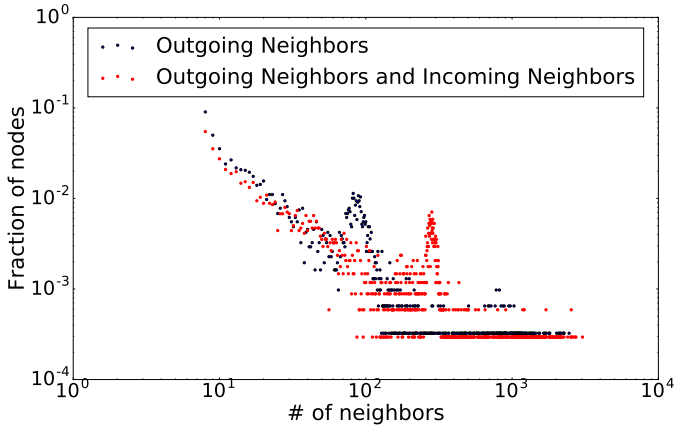


Figure 3.7: Number of outgoing neighbors of heavy, medium, and light nodes.

3.3.3.5 Potential threats

Using our tools, one can identify Monero's network topology and the connectivity of nodes. An example is shown in Figure 3.8, which illustrates the neighbors of a *light node (5.X.X.X)*[12] during the 9-hour monitoring process. Each color represents a neighbor of the node. It shows that neighbor 1-6 stayed connected with the node for the entire 9 hours, whereas the connection with neighbor 7 is dropped around the 8th hour, and a connection with neighbor 11 was established to replace neighbor 7. Similarly, a connection with neighbor 9 was established to replace neighbor 8 after 3 hours.

[12]Hidden IP address to protect the privacy of this light node.

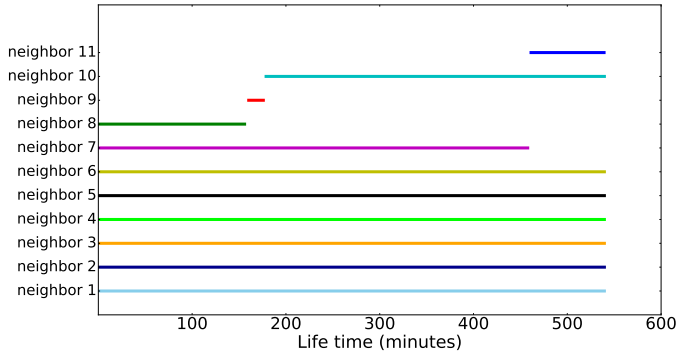


Figure 3.8: Dynamic neighbor tracking of a light node in 9 hours.

With such knowledge, an attacker can potentially launch different types of attacks. For example, an attacker could launch a targeted attack by monopolizing all connections of a victim node [12], selectively partition the network [11], or even deanonymize transactions by identifying the first node relaying a transaction [79, 101].

3.3.4 Implications and Insights

In this section, we presented methods that we developed to observe Monero's peer-to-peer network, and infer its topology. We described how one can deploy Monero nodes to discover all the nodes participating in the protocol, and their interconnections, using the *last_seen* timestamps in the peer lists that nodes exchange. For accuracy, we compared our methods' results with the ground truth of our deployed nodes. Our experiments show that even though Monero is a privacy-preserving cryptocurrency, it is still possible to accurately discover the nodes in the network and their interconnections. Our analysis provides insights about Monero's degree of centralization, and about the privacy and security issues potentially caused by a network topology exposure.

Chapter 4

Characterizing the Impact of Network Delay on Bitcoin Mining

While previous works have discussed the network delay upper bound that guarantees the consistency of Nakamoto consensus, measuring the actual network latencies and evaluating their impact on miners/pools in Bitcoin remain open questions. This chapter fills this gap by: (1) defining metrics that quantify the impact of network latency on the mining network; (2) developing a tool, named miner entanglement (ME), to experimentally evaluate these metrics with a focus on the network latency of the top mining pools; and (3) quantifying the impact of the current network delays on Bitcoin's mining network. For example, we evaluated that Poolin, a Bitcoin mining pool, was able to gain between 0.5% and 1.9% of blocks in addition (i.e., from 36.27 BTC to 137.83 BTC) per week thanks to its low network latency. Moreover, as pools are rational in Bitcoin, we model the strategy a pool would follow to improve its network latency (e.g., by leveraging our ME tool) as a two party game. We show that a Bitcoin mining pool could improve its effective hash rate by up to 4.5%. For a multi-party game, we use a state-of-the-art Bitcoin mining simulator to study the situation where all pools attempt to improve their network latency and show that the largest mining pools would improve their revenue and reach a Nash equilibrium while the smaller mining pools would suffer from a decreased access to the network, and therefore a decreased revenue. These conclusions further incentivize the centralisation of the mining network in Bitcoin, and provide an empirical explanation for the observed tendency of pools to design and rely on low latency private networks.

4.1 Bitcoin Mining Process and Network

In this section, we recall the basic concepts of Bitcoin mining, which explain why the generation times of blocks fluctuates. We then introduce the core concepts behind PoW cryptocurrency networks, which disseminate blocks to all miners with variable delays.

4.1.1 Mining Process

Mining is a trial-and-error process. From a candidate set of transactions, miners assemble a block and use their processors to identify a hash that is smaller than a target value[1]. Upon finding such a hash, a miner has successfully created a block.

Mining as a Poisson process. In practice, the discovery rate of blocks is not constant. Many works have captured this variation by modeling the PoW mining process as a Poisson process [8, 33, 34, 35], where the success rate λ corresponds to a block being generated every 10 minutes in average. The probability density function of a Poisson process is $P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$.

Block interval distribution. By modeling Bitcoin mining as a Poisson process, the block generation interval follows an exponential distribution [8, 34], whose probability density function is $G(t) = \lambda e^{-\lambda t}$, where t denotes the block interval between two adjacent blocks and λ is the average success rate. We present in Section 4.3 the empirical distribution of the Block receptions we observed during our experiments.

4.1.2 PoW Cryptocurrency Network

A PoW cryptocurrency network consists of a P2P overlay that interconnects solo miners and mining pools. This network disseminates the newly found blocks to all miners. To clarify the difference between the P2P overlay and the intra network of mining pools, we use mining pools and miners to denote the nodes of the P2P overlay, and sub-miners to denote the miners located within the mining pools.

P2P overlay network. The P2P overlay network mainly consists of full nodes[2] and client nodes. Each node maintains a peer list and periodically exchanges information with other peers to keep it up-to-date and to disseminate blocks and transactions. The solo miners could also be full nodes. The mining pools normally maintain some full nodes in the P2P overlay as the front end to exchange messages with other pools/miners. The main properties of cryptocurrency P2P networks have been widely studied [34, 43, 48, 21], such as their network size, node degree distribution, and connectivity. Existing measurements (as we have introduced in

[1] https://en.bitcoin.it/wiki/Block_hashing_algorithm
[2] https://bitnodes.io/

Chapter 2.5) rely on the P2P overlay network to evaluate the block propagation delay, which cannot reflect the delays of pools, mainly because of the unknown IP addresses of pools' front-end nodes and network churn.

Intra network of mining pools. As the mining difficulty increases, the revenue of miners that have a small hash power becomes more irregular. To compensate this effect, miners can join mining pools. The block rewards that are collectively earned by the members of a mining pool are shared among them according to their participation. The internal networks of mining pool usually work as client-server infrastructures. A mining pool uses some dedicated servers to manage its sub-miners. The sub-miners are only connected with those servers who assign them jobs and inform them of the new blocks. When a sub-miner has finished a job, it sends its result to the servers. The mining pools also maintain front-end nodes in the P2P network to exchange messages with other miners and pools. The most popular protocol used by mining pools is Stratum [47].

4.2 Quantifying the Impact of Network Delay on Mining

This section defines metrics to evaluate the impact of network latency on mining.

4.2.1 Mining and latencies

The mining process of a miner can be decomposed in three successive phases, among which the first and the last are affected by the miner's access to the network. We illustrate those phases in Fig. 4.1.

1. **New block reception.** Once a block has been discovered by a miner in the network, it is transmitted to the whole network. The delay between the discovery of a block and its reception by a miner is the block reception latency of this miner on this block. In Fig. 4.1, $d_{BR}(3,1)$ is the block reception latency of miner 3 on block 0, whose reception allows it to work on block 1. The sooner a miner receives a block the sooner it can start attempting to solve the cryptographic puzzle.

2. **Cryptographic puzzle solving.** Based on the new block, miners compute hashes to try to discover the next block, by attempting to solve the mining cryptographic puzzle. During this phase, which lasts until the next block is found in the network, miners are doing effective work. The number of hashes they can generate is the product of their computing power, expressed in hashes per second, and the time they dedicate to mining the current block.

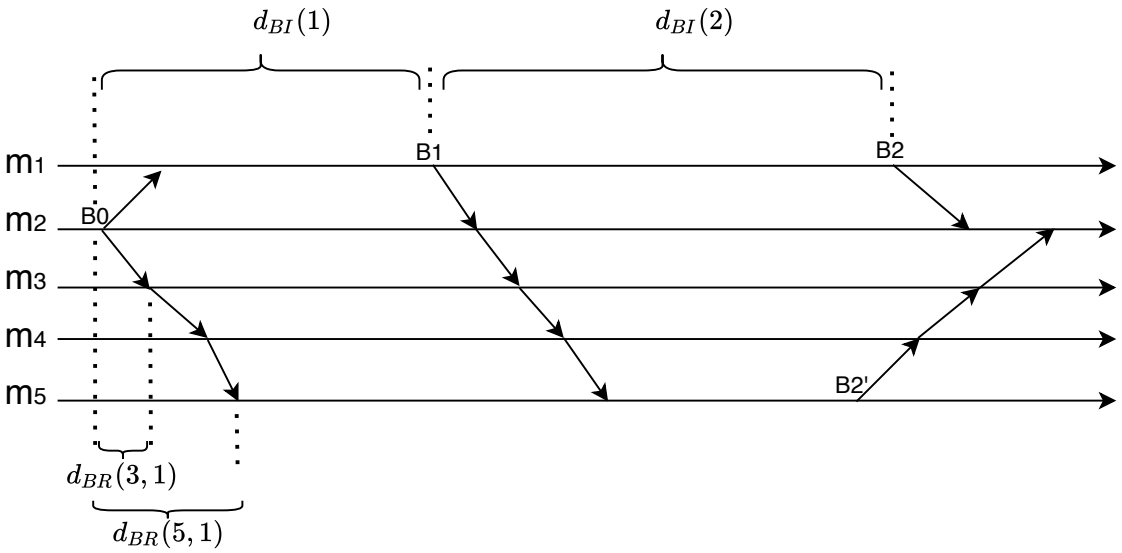


Figure 4.1: Network influence on the mining process in PoW-based cryptocurrencies. d_{BR} represents the block reception latency, and d_{BI} denotes the block interval.

In Fig. 4.1, $d_{BI}(1) - d_{BR}(3,1)$ is the time of effective work of miner 3 on block 1, and $d_{BI}(1) - d_{BR}(5,1)$ is the time of effective work of miner 5 on block 1. In this example $d_{BR}(3,1) < d_{BR}(5,1)$, which means that miner 3 has worked more effectively than miner 5 on block 1.

3. **Next block dissemination.** After a new block has been discovered, it has to be disseminated to the nodes who collectively account for more than 50% of the global hash power to be validated. The position of a node in the network influences the probability with which a new block is validated when several blocks are simultaneously discovered.

4.2.2 Block reception latency and effective hash rate

Miners might receive a newly found block at different times, which might lead those that receive a block after the others to waste hashing power on an already solved cryptographic puzzle. Because of this effect, two miners with equal hash power might consistently earn different revenue. The effect of the network latency on mining is particularly acute when a block is discovered in a short time, which happens in practice as captured by the Poisson distribution of mining process.

Obtaining the hash rate distribution in the network is difficult, as miners and pools do not reveal their real-time hash power and because the network is under constant evolution. It is therefore challenging to evaluate the impact of the mining

network on miners based on the hash rate distribution and on statistics about block discoveries. We instead define the effectiveness ratio metric to evaluate the effectiveness of an individual miner independently of the global hash rate distribution as follows.

We consider a (snapshot of a) mining network of N miners. Let H_i be the hash rate of miner $i \in [1, N]$, $d_{BI}(j)$ be the block generation interval between the $(j-1)^{th}$ block and the j^{th} block (i.e., the length of time between the generation of the $(j-1)^{th}$ and j^{th} block), and $d_{BR}(i,j)$ be the block latency of receiving the $(j-1)^{th}$ block at miner i.

Effectiveness ratio. The effectiveness ratio $f_{i,j}$ of miner i on the j^{th} block is $\left(1 - \frac{d_{BR}(i,j)}{d_{BI}(j)}\right) \in [0, 1]$.

The effectiveness ratio of a miner on a block is the proportion of time it can compute hashes during the associated block interval. An effectiveness ratio close to 1 indicates that miner i is well positioned in the network. This formula captures the fact that a mining pool that quickly receives a block and starts early to mine the next block has an advantage over the other pools, which receive the same block later. Moreover, since d_{BI} follows an exponential distribution (the probability density function $G_0(t) = \lambda \times e^{-\lambda \times t}$, where $\lambda \approx \frac{1}{600}$ in Bitcoin) [8, 34], the effectiveness ratio of a miner differs depending on the blocks. For instance, even though the expected d_{BI} in Bitcoin is 600 seconds, 1.65% ($G_0(10)$) of the blocks are such that $d_{BI} < 10$ seconds, 8.0% ($G_0(50)$) are such that $d_{BI} < 50$ seconds, and 15.3% ($G_0(100)$) are such that $d_{BI} < 100$ seconds. A few seconds delay in the reception of a block can significantly affect the effective hash rate of miners in some block rounds, and drift their revenue share.

We capture the impact of heterogeneous delays on the whole network over the j^{th} block using a 2-tuple (G_j, D_j), where $G_j \in [0, 1]$ is the Gini coefficient $gini(f_{1,j}, f_{2,j}, ..., f_{n,j})$ [114] and $D_j \in [0, 1]$ is the difference between the highest and the lowest effectiveness ratio observed among miners on block j. A small G_j indicates that the miners have in average similar effectiveness ratios over the j^{th} block, while a larger G_j shows unfairness. D indicates the amplitude of the distribution of effectiveness ratios. Note that the Gini coefficient alone describes the degree of inequality of a distribution, and that we use D to provide additional information.

We define the effective hash rate (EHR) using the effectiveness ratio, as follows.

EHR. The effective hash rate $EHR(i,j)$ of miner i on the j^{th} block is equal to $H_i \left(1 - \frac{d_{BR}(i,j)}{d_{BI}(j)}\right) \in [0, H_i]$.

The effective hash rate $EHR(i, j)$ of a miner i corresponds to the number of hashes it is able to compute after having received block $j-1$ and before block j is discovered.

4.2.3 Impact of heterogeneous network delays on revenue

The competitiveness of a miner or a pool in the mining race not only depends on its hash rate, but also on its network delay. In the following, we detail how the block revenue of a miner is influenced by the heterogeneous delays.

HR Share. The hash rate share $HR\ \ Share_i$ of miner i is $\frac{H_i}{\sum_{k=1}^{N} H_k} \in [0, 1]$.

The HR Share of miner depends on its real hash rate and on the global hash rate. Without considering the impact of network delay, the mining success rate of a miner is equal to its HR Share. However, we take the impact of network delay into account to evaluate a miner's success rate and define the EHR Share as follows.

EHR Share. The effective hash rate share $EHR\ \ Share_i$ of miner i is $\frac{H_i \times f_i}{\sum_{k=1}^{N}(H_k \times f_k)} \in [0, 1]$.

The revenue of a miner is determined by its EHR share, which in turn depends on its hash rate and on the network delays. We use this metric to further evaluate the impact of network delays on the revenue distribution.

4.3 Measurement and Evaluation

To evaluate the impact of network latency on mining, we develop a tool called Miner Entanglement (ME) to monitor the block reception latency of mining pools. ME leverages the mining pools' API to measure the time it takes for the mining pools to learn about newly discovered blocks. We deploy Bitcoin nodes running ME and quantify the impact of network delays with the metrics that we defined in Section 4.2.

4.3.1 Leveraging the API of mining pools

Miner Entanglement (ME) uses BFGminer [115] to build direct TCP connections between a local machine and mining pools. More specifically, we deploy ME in our local machine, which is registered as sub-miners in several mining pools. Since the pools recognize a ME-empowered node as a sub-miner, they directly inform

the ME-empowered node with the information about new blocks. This enables us to estimate the block reception delays of the mining pools. We connected to 10 Bitcoin mining pools, which we list in Table 2.2. Collectively, these pools own approximately 69.88% of the global computing power. We could not establish connections with the pools that own the remaining 30.12% of the global computing power because they either do not accept non-ASIC devices, or because they could not be identified.

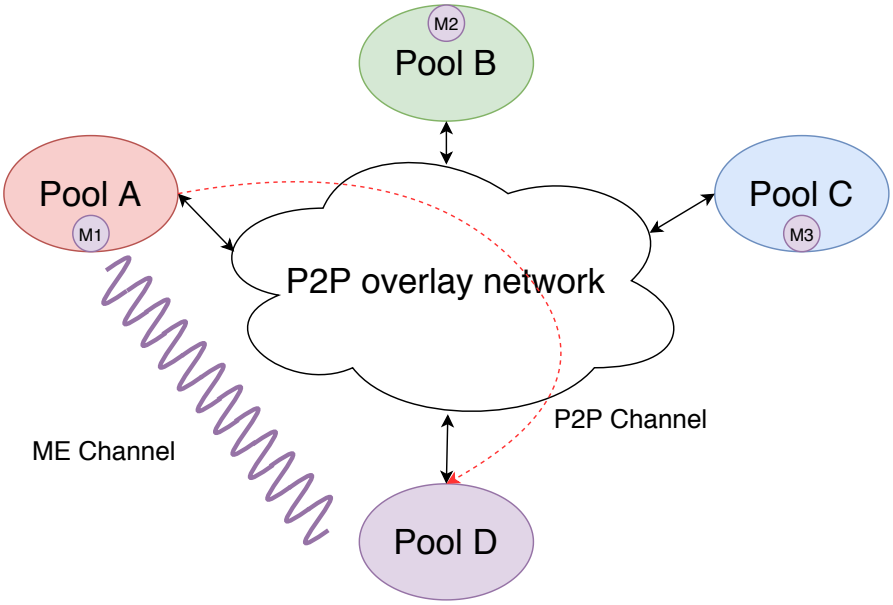


Figure 4.2: Illustration of the miner entanglement design. Pool D registers three sub-miners M_1, M_2, and M_3 in Pool A, Pool B, and Pool C respectively. This allows Pool D to receive the information from other pools directly, thus, avoiding the delay of the P2P overlay.

We illustrate the design of ME in Figure 4.2 where Pool D represents our local machine, and where we deploy sub-miners in pools A, B, and C to receive block discovery information directly via the ME channels (which are established directly between our machine and the pools). We also run a full Bitcoin node on our machine using the default P2P protocol to evaluate the latency of a normal Bitcoin full node via the P2P channel (which is randomly built between peers) as a comparison. We use an Ubuntu 16.04 desktop with an Intel Core i7-7700 processor.

We run ME for an entire week and have collected discovery notifications related to 1,116 blocks (from block 641,767 to 642,882), which represented 68.1 MB of data overall. From this dataset, we report raw data such as the block intervals and the block reception delays. We then evaluate the impact of network delays on Bitcoin

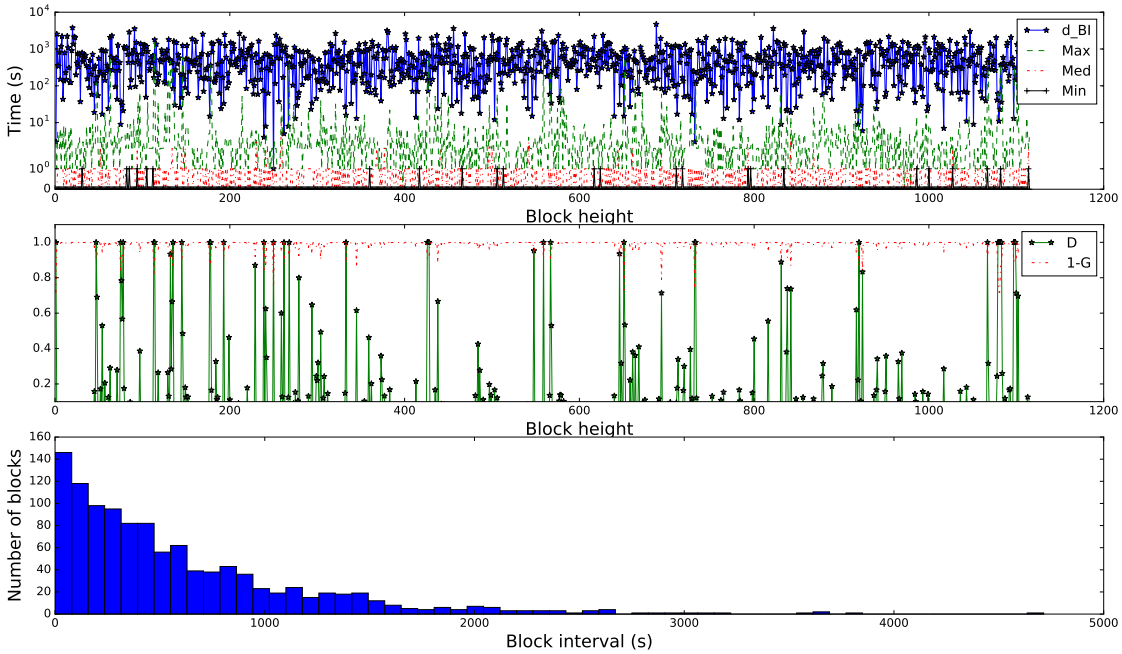


Figure 4.3: Results of measurement in Bitcoin. The top figure shows the block interval d_{BI}, and the maximum Max, median Med, and minimum Min block reception latency among 10 pools from block 641,767 to 642,882. The middle figure indicates the changes of (G, D) between 10 pools during a week. The bottom figure shows that the block interval follows the exponential distribution during a week.

mining by using our metrics.

4.3.2 Discussion

We estimate the mining power by using the number of blocks created by each miner, evaluate the effect of effectiveness ratio on mining, and infer the revenue bounds of different pools via Monte Carlo simulation.

For each pool we managed to use ME with, we use half of the round trip time (as which has been used in other works [48, 44]) between our machine and the mining pool, i.e., $\frac{RTT}{2}$, to estimate the time needed for a mining pool to receive a block and start sending it to its sub-miners. We obtain the round trip time with the ping command, which uses ICMP packets. We calculate the block sending time of the target pool using the block reception time of our sub-miner minus the block propagation delay. We therefore assume that the pools relay new blocks to their sub-miners as soon as they learn about them. This assumption is reasonable as mining pools are rational and aim to increase their revenue.

The mining pools with which we were able to establish a ME channel with collectively own approximately 69.88% of the network hash power. We evaluate the effectiveness ratio of the miners that own the remaining 30.12% considering three scenarios depending on whether: i) they use a Bitcoin full node; ii) their connectivity is close to the average connectivity of the pools we established connections with; and iii) they use ME. This allows us to present an interval of realistic values.

The effectiveness ratio of the pools would be affected when concurrent blocks are discovered simultaneously. In this case, the pools that had worked on the stale block wasted their hash power, and decreased the effectiveness ratio. The probability for this to happen in Bitcoin is low (0.41% in 2016) [16]. During our one week measurement in August 2020, we did not find any stale block. Therefore, we do not consider the impact of stale blocks in this thesis.

4.3.3 Block reception latency and block interval

At the top of Fig. 4.3, we report the minimum, median and maximum latencies that we observed for each block. We also report the block intervals (with stars in blue), which in average were equal to 600s, as expected. In a week of measurement, each block was received by half of the pools in less than 1 second (as indicated by the red line that indicates the median). However, some pools suffered more than 10 seconds of delay in some mining rounds (as the line that represents the maximum value indicates). The 1s median block reception latency that we measured through ME is similar to the block propagation delay to 50% reachable nodes in Bitcoin network that is reported by KIT [49]. However, we observed a maximum block reception latency of 13.76 seconds, which is higher than their reported block propagation

delay of 90% reachable nodes (less than 5s). This difference of block reception latency between pools leads to different effective hash rates, and its impact on the effective hash rate is amplified in the mining rounds with short block intervals. For instance, in some cases mining pools have completely missed the chance to mine a block as the block reception latency was greater than the block interval (as indicated by the crossing points between d_{BI} and Max). That is, before they have received a block, the next block had already been produced. The bottom of Fig. 4.3 reports the distribution of block intervals during one week, which seems coherent with the expected exponential distribution.

4.3.4 Effectiveness ratio

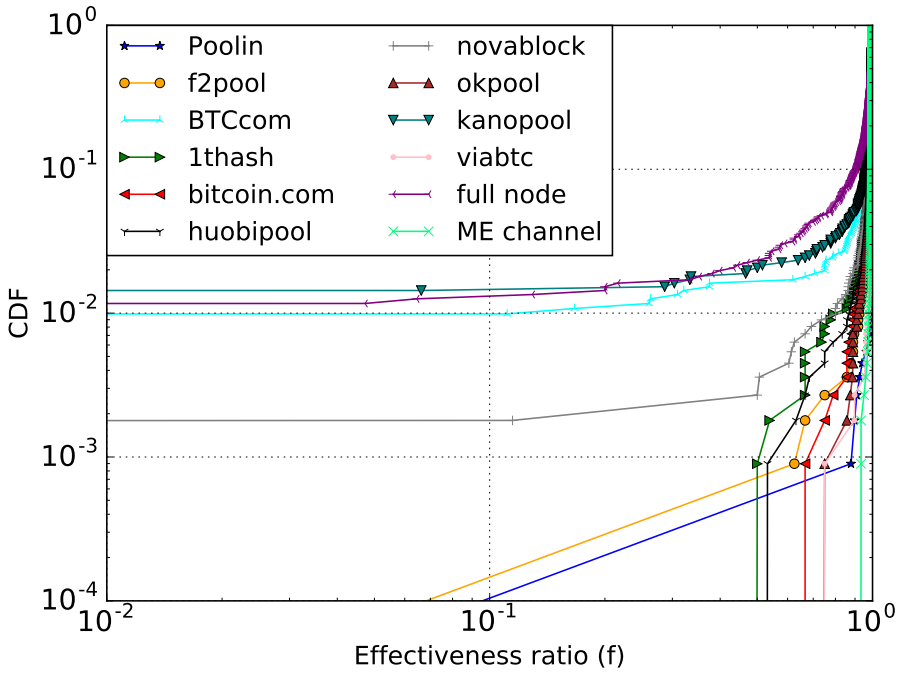


Figure 4.4: CDF of the observed effectiveness ratio of several Bitcoin mining pools during one week. The legend denotes the name of the mining pools, except for “full node", which represents the Bitcoin full node we maintained, and “ME channel" represents the results obtained with ME.

We calculate the effectiveness ratio of each pool on each of the blocks to analyze the impact of heterogeneous block reception delays in detail. First, Fig. 4.4 shows the CDF of the effectiveness ratios for each of the 10 pools. Each distribution shows how the effectiveness ratio of the pools varies during a week. For instance, the effectiveness ratio of “BTC.com” was smaller than 90% on 34 blocks during

a week, which decreased its competitiveness in the mining competition. As a comparison, the default Bitcoin full node had the worst effectiveness ratio (95.2% in average), and our node had the highest effectiveness ratio (99.7% in average) thanks to the ME channels. The average effectiveness ratio of connected pools is distributed between 96.2% to 99.2%. We evaluate the consequences on the revenue of mining pools in the next section. Second, for each block, we plot the Gini coefficient G and the maximum difference D between the pools' effectiveness ratios to illustrate the difference in the effective hash rate in the middle figure of Fig. 4.3. Notice that for better readability we plot $1-G$ instead of G. The average Gini coefficient was approximately equal to 0.011, which indicates a small average deviation between the 10 pools. However, we also observed that $D > 0.2$ for 9.3% of blocks, which means that some pools could work more effectively than others on the corresponding blocks, which in turn influenced their revenue.

4.3.5 Impact of block intervals on effectiveness ratios

As we have shown in Fig. 4.3, the block reception latency has an impact over the duration of a block interval and modifies the effectiveness ratio. Here, we study the pools' effectiveness ratio on blocks with different interval delays. We calculate the average effectiveness ratio of pools for different ranges of block interval, and illustrate the result in Fig. 4.5. The main observation one can make is that pools have smaller effectiveness ratio on the blocks with short generation times. For instance, the average effectiveness ratio of "BTC.com" is 97.39%, 96.12%, and 92.61% on the blocks with generation time ≤ 4717 seconds (the maximum block interval), < 600 seconds, and < 200 seconds respectively. We then use the actual blocks to evaluate the effect of effectiveness ratio. As shown in Tab. 4.1, we collected 1,116 blocks during one week. The revenue distribution was the following: "Poolin" earned 14.53% of blocks, "F2Pool" earned 14.26%, "BTC.com" earned 12.20%, "Huobi" earned 8.97%, "1THash" earned 7.00%, "OKpool" earned 6.64%, "ViaBTC" earned 5.47%, and "Novapool" earned 0.81%. "bitcoin.com" and "kanopool" did not get any block during one week. The pool's revenue share was affected by the block interval. When $d_{BI} < 200$, the actual revenue share of "BTC.com" was 10.8%, which represents a 13.11% decrease from 12.20%. This decrease is explained by the observations in Fig 4.4 and Fig 4.5, where "BTC.com" had the worst average effectiveness ratio during one week, in particular on the blocks with block intervals lower than 200 seconds. To evaluate the revenue losses or gains associated to the effectiveness ratio changes, we use the EHR Share metric to infer the bounds on revenue share of pools in Section 4.3.6.

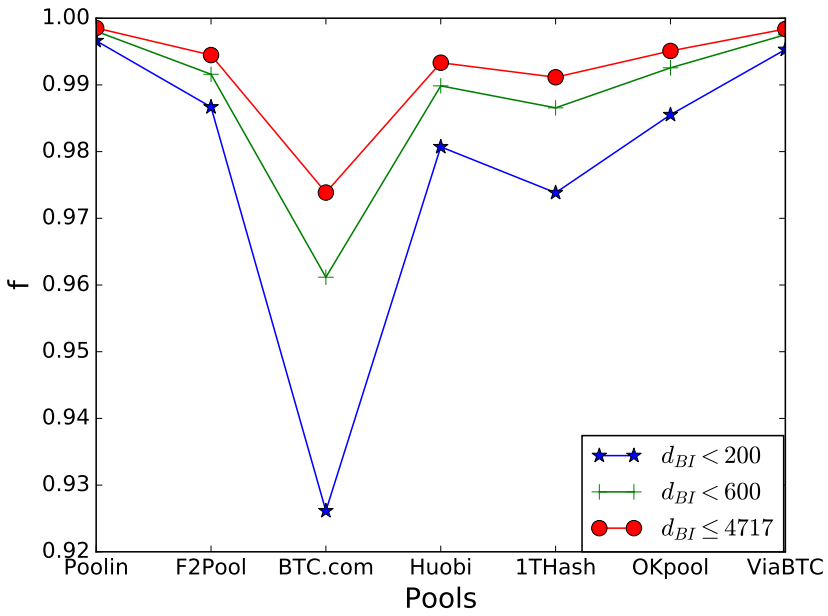


Figure 4.5: Effectiveness ratio of mining pools with three ranges of block intervals.

4.3.6 Inferring the revenue bounds

To exactly evaluate the impact of network delays on the revenue distribution, one would need to know the real hash rate distribution, and evaluate the EHR Share. It is however challenging to obtain the actual hash rate distribution. Our approach to evaluate the impact of network delays on miners revenue follows the one developed in previous works [15, 16]: (1) we calculate the hash rate distribution based on the discovered blocks; (2) we consider the impact of network delay on the mining process; (3) we use a Monte Carlo method to simulate 10K blocks to evaluate the

Table 4.1: Distribution of blocks during one week.

	Poolin	F2Pool	BTCcom	Huobi	1THash
d_{BI}<200	47	45	35	39	18
d_{BI}<600	111	107	77	63	42
In total	163	159	136	101	78
	OKpool	ViaBTC	Nova		
d_{BI}<200	20	17	1		
d_{BI}<600	43	33	8		
In total	74	61	9		

revenue distribution. We calculate the revenue drift rate as $\frac{EHR\ Share - HR\ Share}{HR\ Share}$. A positive value indicates extra earnings, while a negative value shows a loss. Figure 4.6 shows the revenue drift rate per mining pool depending on the assumptions one could make regarding the proportion of the global hash rate we could not connect establish ME channels with.

Most of the top mining pools could earn more than 1% of extra blocks if we assume that the remaining 30.12% computing power only use a Bitcoin full node ($f_{unknown} = 0.952$), except for the 3^{th} pool (BTCcom) whose revenue drift rate is smaller than 0 due to the deviation of f as we have indicated in Fig. 4.4. It is interesting to note that when the remaining 30.12% computing power have a connectivity that is equal to the mean of the pools we connected to ($f_{unknown} = 0.989$), the 3^{rd} pool finds 1.7% less blocks, and the 6^{th} pool finds 0.3% less blocks, while the other pools' drift rate is still positive (between 0 and 1%). When the unknown 30.12% computing power uses ME ($f_{unknown} = 0.997$), the revenue of the 8 connected pools decreases, as one could assume. However, even in that scenario some mining pools have an increased revenue, which confirms that some mining pools have exceptionally good network connectivity. The fact that some mining pools use different methods to optimize their connectivity [43, 116] could explain this observation.

4.4 The Impact of Large Scale Deviations

We have shown in the previous section that the miners that do not optimize their connections have a disadvantage in the mining network. In this context, a rational miner would try to improve its network access. This section discusses the situation where miners would massively decide to deviate from the official P2P protocol to try to improve their revenue. Assuming such a scenario, we first show that the top miners would reach a Nash equilibrium as they would connect to each other, while the rest of the network would not be able to do so. We conduct a simulation using 1,000 miners on a state-of-the-art mining simulator [16] to evaluate the resulting metrics.

4.4.1 Nash equilibrium among the biggest mining pools

We consider a network with N nodes $V = \{0, \cdots, N-1\}$ that includes mining nodes and non-mining nodes. If node i uses a strategy s_i to select a subset of V as its neighbors, then its strategy space, i.e., the universe of possible neighbors sets, can be denoted as $S_i = 2^{V \setminus \{i\}}$. For $i \in V$, it may use different strategies $s_0, \cdots, s_x, \cdots, s_{n-1}$ to select its neighbors. Therefore, we would obtain graph

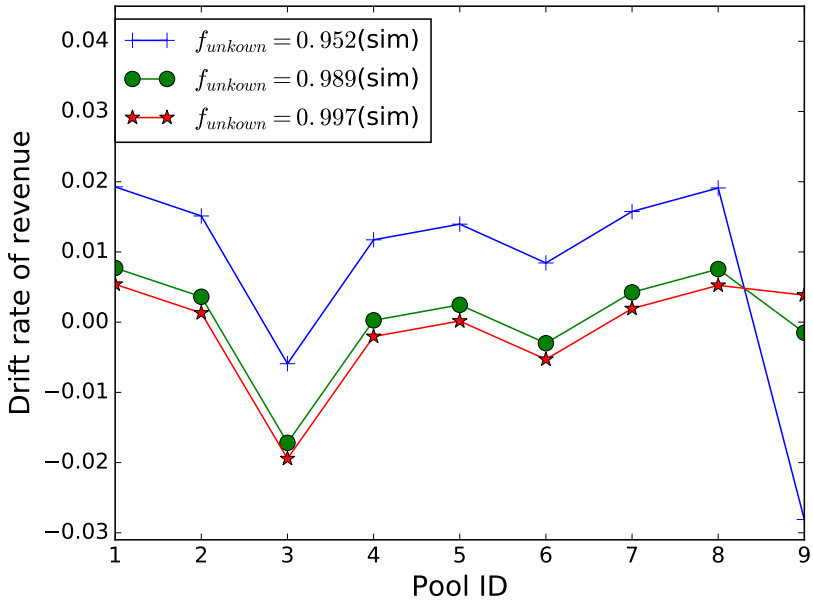


Figure 4.6: The bounds of mining fairness in Bitcoin. The drift rate of revenue is calculated by using EHR Share minus HR Share, and validated through the revenue share. The x-axis represents different mining pools. From left to right: Poolin, F2Pool, BTCcom, 1THash, Huobi, Novapool, OKpool, ViaBTC and the last one represents the remaining mining power.

$G[s] = (V, \bigcup_{i=0}^{n-1}(\{i\} \times s_i)$, where $(s_0, s_1, ..., s_x, ..., s_{n-1}) \in S_0 \times S_1 \times ... \times S_{(n-1)}$.

When node i uses strategy s_i to select p neighbors, we can use the set $N_i = \{N_{i,1}, N_{i,2}, ..., N_{i,p}\}$ to denote the set of neighbors it selects, where $N_i \in V$ and $p \leq n$. The set of edges between miner i and its neighbors can be denoted as $E_i = \{\{i, N_{i,1}\}, \{i, N_{i,2}\}, ..., \{i, N_{i,p}\}\}$.

According to the cost function that was described in [117], we have $c_i(s) = \alpha \times |E_i| + \sum_{i \neq j} stretch_{G[s]}(i, j)$, where $stretch_G(i, j)$ defines the distance cost between pool i and j, which is equal to the shortest distance between i and j using links of the graph divided by the direct distance. For instance, in a complete graph, $stretch_G(i, j) = 1$. α defines the cost of maintaining the connections, which is equal to the cost of the connections divided by the cost of *stretch*. Assuming there is a set of mining nodes $M \in V$, for any mining node i, the cost function can be expressed as $c_i(s) = \alpha \times |E_i| + \sum_{q=0}^{q=k} stretch_{G[s]}(i, j)$, *where* $i, j \in M$.

The mining node can maintain direct connections with other mining nodes, so that the minimum *stretch* has a cost equal to 1. Therefore, if the number

of mining nodes is k, the cost of mining node i that uses ME strategy s_{me} is $c_i(s_{me}) = \alpha \times |E_i| + (k-1)$. We define that a Nash equilibrium can be achieved by using ME as follows.

Lemma 1. *In a cryptocurrency network with k mining nodes and $n-k$ non-mining nodes ($k \leq n$), if each node can maintain l connections ($l > k$), then ME is a strategy that allows a Nash equilibrium to be reached.*

Proof. The ME strategy s_{me} does not increase the number of direct links for the mining node since $l > k$. The ME channel is a TCP/IP connection on the Internet, similar to standard P2P connections, so that for l connections, s_{me} and s_i have the same link cost, i.e., $\alpha \times l$. Considering that the minimum *stretch* cost can be achieved if two mining nodes are directly connected, for any two mining nodes i and j, we have $stretch_{G[s_i]}(i,j) \geq stretch_{G[s_{me}]}(i,j)$, *where* $i \neq j$, and therefore, $c_i(s_i) \geq c_i(s_{me})$. □

In practice, l is significantly greater than k. For instance, the Bitcoin network contains around 20 to 30 mining pools, and the default connectivity of a node is 125. Moreover, the connectivity of a node can easily be extended to up to 1,000.

4.4.2 Decreased revenue of the smaller mining pools

Table 4.2: Rationality against fairness.

	non-selfish (m_b)	selfish (m_b)
non-selfish (m_a)	$f_a = A$, $f_b = B$	$f_a = A$, $f_b > B$
selfish (m_a)	$f_a > A$, $f_b = B$	$f_a > A$, $f_b < B$

The PoW cryptocurrency network is permissionless and the participants are allowed to behave selfishly. Here, we point out that rationality would push the miners to select their neighbors selfishly, and thereby causes unfairness. Let us assume a network of n miners $\{m_1, \cdots, m_n\}$ with hash rates $\{H_1 > H_2 > \cdots > H_n\}$. We consider two miners m_a and m_b such that $a < b$. Each miner can maintain l connections, and $l < a - b$. We prove the game of rationality against fairness as follows, and summarize our findings in Table 4.2.

- If m_a and m_b follow the default P2P protocol, their effectiveness ratio would be respectively equal to $f_a = A$ and $f_b = B$.
- A miner can improve its effectiveness ratio if it selects its neighbors selfishly. For instance, miner m_b could establish connections with the top miners to reduce its average block reception latency d_{BR} and improve its effectiveness ratio, which would then be greater than B.

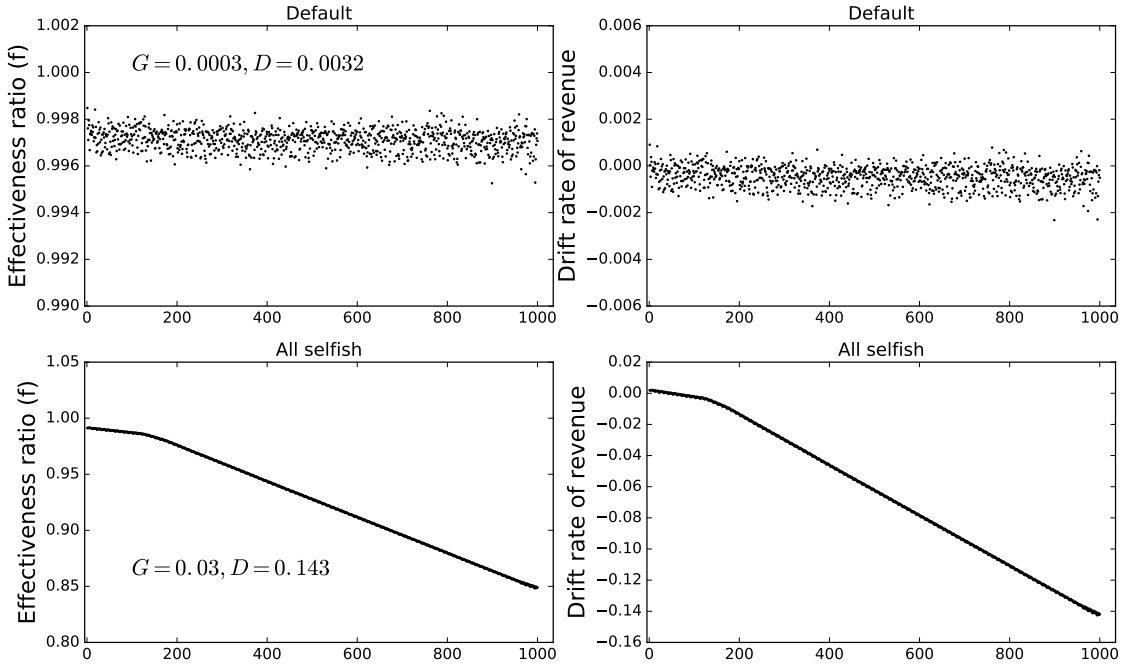


Figure 4.7: Impact of selfish behaviors on miners' effectiveness ratio and fairness. The x-axis represents miner's ID

- The effectiveness ratio of a miner is greatly influenced by its hash power when all miners are selfish. In this case, the miner with high hash power will connect with the miners whose hash power are closest to hers, eventually, leading to a chain-like network. The miners with higher hash powers then have relatively small block reception latency, thus, f_a increases, and f_b decreases. We use simulation to validate this evolution.

A miner could use the selfish neighbor selection strategy to improve its effectiveness ratio f. If all miners were to select their neighbors selfishly, a miner with a larger hash power would gain a larger advantage, and the network's fairness would decrease.

4.4.3 Simulation

We modified Gervais et al.'s mining simulator [16] and simulated a network of 1,000 miners. The hash power distribution partially follows the one of Bitcoin: we assign the hash rates of the top 15 miners based on publicly accessible data [3], and the remaining miners equally share the remaining hash rate.

[3]https://btc.com/; https://www.blockchain.com/explorer

Modifications of the simulator. We i) enable 1,000 miners instead of the original 16 miners used by default; ii) remove the accumulating delay; iii) create an interface to connect with the network topology generator. This interface allows us to examine a dynamic network and observe how the miners' network latencies would evolve depending on different strategies.

Network topology generator. Our network topology generator mimics the existing cryptocurrency P2P protocol, it includes the following main functions:

Peer list. A table to maintain a node's peers information.

Peer information exchanging. To keep the network connectivity, each node exchanges a partial information of its peer list with others.

Neighbor selection. By default, each node has to periodically select a random peer from its peer list to try to establish the new connection. However, a rational node can select its neighbors selfishly. We simulate both a random neighbor selection and a selfish neighbor selection.

Selfish behaviors. We assume that miners know each other's hash power. A selfish miner tries to connect to the peer with the largest hash power in its peer list. We compare two scenarios: *i) default:* every miner selects its neighbors randomly according to the default protocol; and *ii) all selfish:* all miners select the neighbors selfishly.

We show in Fig. 4.7 the results of these simulations, where the mining power of a miner decreases when its ID increases. Our goal is to simulate a large scale mining network.

As shown at the top of Fig. 4.7, with the default P2P protocol, all miners select their neighbors randomly, which results in a fair network. By using the metrics we defined in Section 4.2, we have $(G, D) = (0.0003, 0.0032)$. Under this network topology, the revenue rate of miners is modified from -0.2% to 0.1%, independently of the computing power. However, the default setting does not show rational miners, and in practice miners are able to select their neighbors. At the bottom of Fig. 4.7, we assume that every miner is rational, and wants to have powerful neighbors. By using our network topology generator, we show that after 1,000 rounds of selfish neighbor selections, a miner is connected with other miners with similar hash power (i.e., network assortativity [118]). The miners' effectiveness ratio is then correlated with their ID: the more powerful miners have a shorter block reception latency and a higher effectiveness ratio. Thus, the top miners increase their revenue, and the smaller miners are losing revenue. The network fairness is evaluated to (0.03, 0.143), which indicates that the revenue distribution is unfair.

4.5 The Impact of Network Delay on The Temporary Block Withholding Attacks

Temporary block withholding (TBW) strategies (i.e., double spending attack and selfish mining attack) have been proposed as the security threats to Bitcoin. In this chapter, we first analyze the success rate and profitability of the existing TBW strategies theoretically, in particular, we consider the race between attacker and honest miners as a random walk game. We then propose a novel TBW strategy based on two independent private chain: double private chain strategy (DPC), which improves the issues of existing TBW strategies, and optimizes the profits of the adversary. We conduct the Monte Carlo simulation to validate our theoretical analysis, and show that DPC is optimal, and more practical compared to the existing TBW strategies.

In the default setting, the miner sends/adopts the blocks immediately when it finds/receives them. However, this setting could be easily undermined by the adversary to withhold some blocks temporarily in order to increase her profit. There are mainly two types of TBW, DS and SM, the former aims at break the integrity of exchange between BTC and other goods, and the latter targets the extra blocks except for the expected revenue share. We introduce DS starting from Bitcoin's white paper, and then we discuss the success rate and profitability of DS. After that, we present the background knowledge of SM.

Nakamoto's calculation. In Bitcoin, the input and output of different transactions are linked through the digital signature, which ensures that each coin can be verified. Transactions are then validated and recorded into the blocks by miners. Despite that the correctness of on-chain transactions is guaranteed by the honest majority, the adversary with smaller than half of the global computing power is able to get her coins back after she spent the coins meanwhile the goods were dispatched. The adversary first generates transaction TX_1 to use some coins buy the some goods from the receiver, and then generates the transaction TX_2 to send the same coins back to her another account. In order to use TX_2 to replace TX_1, the attacker uses her computing power to maintain a private chain. She can succeed when her private chain is longer than the public chain because of the longest chain principle. Nakamoto S. characterized the race between the attacker and the honest miners as a random walk, and calculated the probability of the attacker catching up with z confirmations. Depending on different possible hash power of the adversary, Nakamoto S. suggested different number of confirmations to limit the success rate of the attacker below 0.1%.

Selfish Mining Attacks. Eyal and Gün Sirer presented the selfish mining strategy, and discussed how the selfish miner's propagation factor, i.e., the proportion of honest miners it can convince to adopt and relay its block, determines

its revenue [17]. They pointed out that NC is vulnerable to selfish mining attacks, which could lead to an unfair revenue (revenue share > hash rate share). In the general case, attacker with > 25% of the global hash power is able to succeed. Previous works [15, 16] have described network-level attacks that are more practical than corrupting a majority of the network, which could optimize selfish mining attacks. Nayak et al. proposed plausible values for the selfish miner's propagation factor by utilizing the public overlay network data [15]. They pointed out that the attacker could optimize her revenue, and win more blocks by eclipsing the honest miners when the propagation factor increases. Gervais et al. [16] analyzed the impact of stale rate on selfish mining attack, and described eclipse attack [12].

4.5.1 Nakamoto's Evaluation

Nakamoto characterized the race between the double spending attacker and the honest miners as a random walk, the probability of the DS attacker catching up with z confirmations was estimated in the Gambler's Ruin model. The attacker with α of the global computing power would catch up with z confirmations with the probability q_z:

$$q_z = \begin{cases} 1 & \text{if } \alpha \geq 0.5 \\ (\frac{\alpha}{1-\alpha})^z & \text{if } \alpha < 0.5 \end{cases} \tag{4.1}$$

In reality, $z = 0$ when the adversary initializes the DS attack, and then it is going to be a *different heights mining race*. When z ($z > 0$) confirmations are attached in the public chain, the adversary's progress can be various. For instance, the adversary could be down by 1 block, or 2 blocks, ..., or z blocks. It is necessary to evaluate the adversary's progress in order to estimate the success probability of the adversary. Nakamoto considered the adversary's progress as a Poisson process, and calculated the probability of different states of the adversary by using the Poisson density function. Let k be the number of blocks that are found by the adversary when z confirmations are attached in the public chain, λ_1 be the Poisson success rate of the public chain, λ_2 be the Poisson success rate of the private chain, assuming that $\lambda_1 = 1$ during z confirmations, then $\lambda_2 = z\frac{\alpha}{1-\alpha}$, therefore, the success probability of the adversary P can be calculated as follows:

$$P = 1 - \sum_{k=0}^{z} (Pr(X = k) * (1 - (\frac{\alpha}{1-\alpha})^{z-k})) \tag{4.2}$$

where, $Pr(X = k)$ is the Poisson density function, and α denotes that the attacker owns α of the global computing power.

Nakamoto's approach mainly focuses on estimating how many confirmations a transaction needs to prevent the DS attacker with the probability that is greater

than 99.9%. The adversary here is deemed to be stubborn, which means, she persists in catching up with the public chain no matter how many blocks she is down by until she succeeds. In reality, the rational adversary can restart the attack when z confirmations are attached in the public chain and her success probability is lower than the expected value, since rebooting a double spending attack would not cause an extra loss to the adversary (the adversary would receive the goods from the merchant even though the double spending attack fails). The crux here is how to evaluate the adversary's success rate at different time. Assuming that a DS attack starts at T_0, what would the adversary's success rate be at T_n (T_n means the time when n blocks have been found since T_0)?

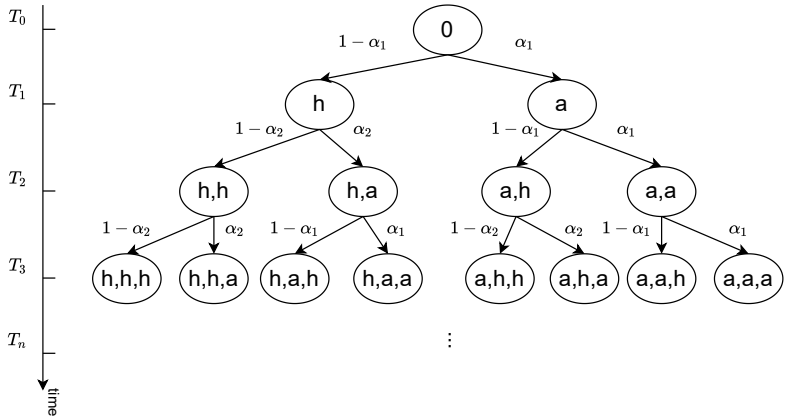


Figure 4.8: Illustration of all possible random walk outcomes of double spending attacker. The attacker maintains a private chain, and competes with the public chain. All possible random walk outcomes are illustrated by a binary tree, where the left child node denotes that a block is found by the honest miners, and the right child node denotes that a block is found by the attacker.

4.5.2 Reconstructing the Double Spending Model

We reconstruct Nakamoto's random walk model in the time domain, which allows us not only to evaluate the double spending attacker's success rate at different time, but also to take the impact of network delay into account. As shown in Fig. 4.8, we use a binary tree to illustrate all possible random walk outcomes of the adversary, and the model is described as follows:

States of the random walk. Let n be the number of blocks that have been found in the whole network since T_0, and $T_1, T_2, ..., T_n$ be the time when the $1^{th}, 2^{th}, ..., n^{th}$ block is found respectively, the number of all possible random walk states at T_n is 2^n, all possible random walk states at T_n can be denoted by set $S(T_n) : \{s(T_n, 1), s(T_n, 2), ..., s(T_n, 2^n)\}$. Each random walk state s is represented

by the corresponding path (ω_1,ω_2,...,ω_n), where $\omega_1, \omega_2, .., \omega_n \in \{h, a\}$, and n represents the length of the path at T_n. h denotes that the block is found by the honest miners, and a denotes that the block is found by the adversary. For instance, if the first block on a path is discovered by the honest miners, we have $\omega_1 = h$.

State transitions. Each state s has two possible transitions during the random walk process, one is triggered when a block is found by the honest miners (we use s' to denote the state of the outcome), and another one is triggered when a block is found by the adversary (we use s'' to denote the state of the outcome). Let α be the hash power of the adversary, without considering the network delay, we have $P(s, s') = 1 - \alpha$, and $P(s, s'') = \alpha$, where $P(s, s')$ and $P(s, s'')$ represent the probability of transition from s to s' and s to s'' respectively. In reality, the transition probability can be affected by the network delay. Let d be the average block reception latency of the honest miners, when an honest miner finds a block, other honest miners have to wait d seconds to receive the block, but the adversary does not suffer the delay. In this case, we have $P(s, s') = \frac{(1-\alpha)*\mu}{\alpha+(1-\alpha)*\mu}$, and $P(s, s'') = \frac{\alpha}{\alpha+(1-\alpha)*\mu}$, where $\mu = 1 - \frac{d}{\Delta T}$, ΔT is the block's generation time between s and s' or s and s''. Let d' be the average block reception latency of the adversary, when the attack starts, the adversary takes d' seconds to receive the block if the last block is discovered by the honest miners, in this case, we have $\mu' = 1 - \frac{d'}{\Delta T}$, otherwise, $d' = 0$ and $\mu' = 1$. Therefore, by considering the network delay, the transition probability is calculated as follows:

$$P(s, s') = \begin{cases} 1 - \alpha & \text{if the last block at } s \text{ was found by the adversary} \\ \frac{(1-\alpha)*\mu}{\alpha*\mu'+(1-\alpha)*\mu} & \text{if the last block at } s \text{ was found by the honest miners and } s = s(T_0, 1) \\ \frac{(1-\alpha)*\mu}{\alpha+(1-\alpha)*\mu} & \text{if the last block at } s \text{ was found by the honest miners} \end{cases} \tag{4.3}$$

$$P(s, s'') = \begin{cases} \alpha & \text{if the last block at } s \text{ was found by the adversary} \\ \frac{\alpha*\mu'}{\alpha*\mu'+(1-\alpha)*\mu} & \text{if the last block at } s \text{ was found by the honest miners and } s = s(T_0, 1) \\ \frac{\alpha}{\alpha+(1-\alpha)*\mu} & \text{if the last block at } s \text{ was found by the honest miners} \end{cases} \tag{4.4}$$

Here, we assume that the adversary has a strong connectivity, which leads to

$\mu' \approx 1$. Let $\alpha_1 = \alpha$ and $\alpha_2 = \frac{\alpha}{\alpha+(1-\alpha)*\mu}$, we show the transition probability between different states in Fig. 4.8.

State probability. Let $pr(T_n, x)$ be the state probability of $s(T_n, x)$ ($x \in [1, 2^n]$), and $p_{\omega_1}, p_{\omega_2}, ..., p_{\omega_n}$ be the probability of transitions that happened on the path of $s(T_n, x)$, we have:

$$pr(T_n, x) = \prod_{i=1}^{i=n} p_{\omega_i} \tag{4.5}$$

where, $p_{\omega_i} \in \{\alpha_1, \alpha_2\}$. Thus, for all possible states at T_n, we have the space of state probability $PR(T_n) : \{pr(T_n, 1), pr(T_n, 2), ..., pr(T_n, 2^n)\}$.

The adversary's success rate. The goal of the adversary here is to undermine the transaction with z confirmations. Let H, A be the number of blocks that were discovered by the honest miners and the adversary respectively on the path of state s. The DS attack is successful when: (1) the adversary finds z blocks before the honest miners ($A \geq z$, and $H < z$); or (2) the adversary withholds at least z blocks when the honest miners find z blocks ($A \geq z$, and $H = z$); or (3) the adversary withholds at least z+j blocks when the honest miners find z+j blocks ($A \geq H > z$). The precondition of the success of the adversary is that at least z blocks are found in the whole network, which means, the minimum time that the adversary needs is $T_z - T_0$. Let $n \geq z$, recall that all possible states at T_n are $s(T_n, 1), s(T_n, 2), ..., s(T_n, 2^n)$ with the state probability $pr(T_n, 1), pr(T_n, 2), , ..., pr(T_n, 2^n)$ respectively, we use set $S'(T_n) : \{s(T_n, x_1), s(T_n, x_2), ..., s(T_n, x_n)\}$ to denote the set of the states that the adversary wins, where $S'(T_n) \in S(T_n)$, and $\forall s \in S'(T_n) : (A \geq z \wedge H < z) \vee (A \geq z \wedge H = z) \vee (A \wedge H > z)$. The success rate of the adversary at T_n can be calculated as follows:

$$P_{T_n} = \sum_{i=1}^{i=x_n} pr(T_n, x_i) \tag{4.6}$$

Evaluating the success rate of DS. We use the Monte Carlo method to simulate the random walk of DS attacker following the time. As shown in Fig. 4.9, we show the success rate of DS attack for the transactions that request 6 confirmation at different time, the attack is feasible only after 12 time steps (i.e., 12 blocks have been created in the network), after that, the success rate of DS increases as time goes by. It reaches the upper bound after 100 time steps. To validate the correctness of our model, we compare the upper bound of our evaluation with S. Nakamoto and R. Mani's evaluations, as a result, the upper bound of our evaluation matches R. Mani's, and higher than S. Nakamoto's. This is because that S. Nakamoto assumes that the arrive rate of honest miners' blocks is equal to 1, which is not realistic[refs].

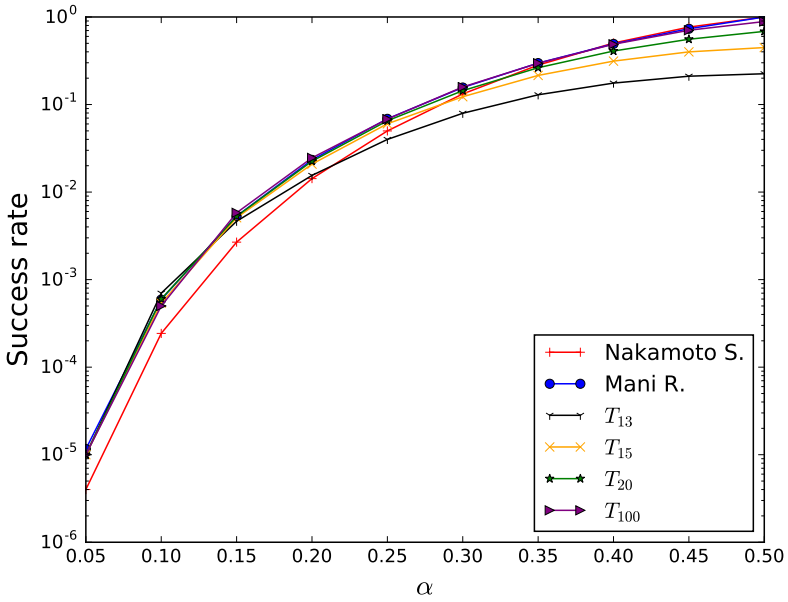


Figure 4.9: The success rate of DS for the transactions with 6 confirmations.

4.5.3 Rebuilding the Selfish Mining Model

Eyal et al. analyzed the selfish mining strategy by using a Markov model [17], which considered 4 types of transition probability during the whole process: α, $1-\alpha$, $(1-\gamma)(1-\alpha)$, and $\gamma(1-\alpha)$, where α represents the adversary's hash rate share, and γ represents the percentage of the honest miners would work on the adversary's block in case of conflicts. The impact of network delay in their model is represented by γ, which mainly focuses on the ability of block propagation under the scenarios when there are conflicting blocks. However, the block reception latencies of miners that could affect the transition probability was not considered. We build the random walk model for the selfish mining attacker, which allows us to take the block reception latencies of miners into account and generalizes the analysis of selfish mining strategy.

States of the random walk. As shown in Figure 4.10, each state is denoted by a 3-tuple (BL, LD, R). BL represents the number of block lead of the attacker, so that $BL \in \{0, 0', 1, 2, ...\}$ (0' represents the conflicting case [17]). LD is the miner that discovered the last block, and $LD \in \{h, a\}$, where h denotes the set of honest miners and a denotes the attacker. R denotes the accumulated rewards (in number of blocks) of the honest miners and the adversary respectively at the state since T_0. For instance, state $s(2, a, (2, 0))$ is the state where the attacker has a lead of 2 blocks, the last block was discovered by attacker. If the adversary terminates the selfish mining attack at this state, she wins 2 blocks, and the honest miners do not win any.

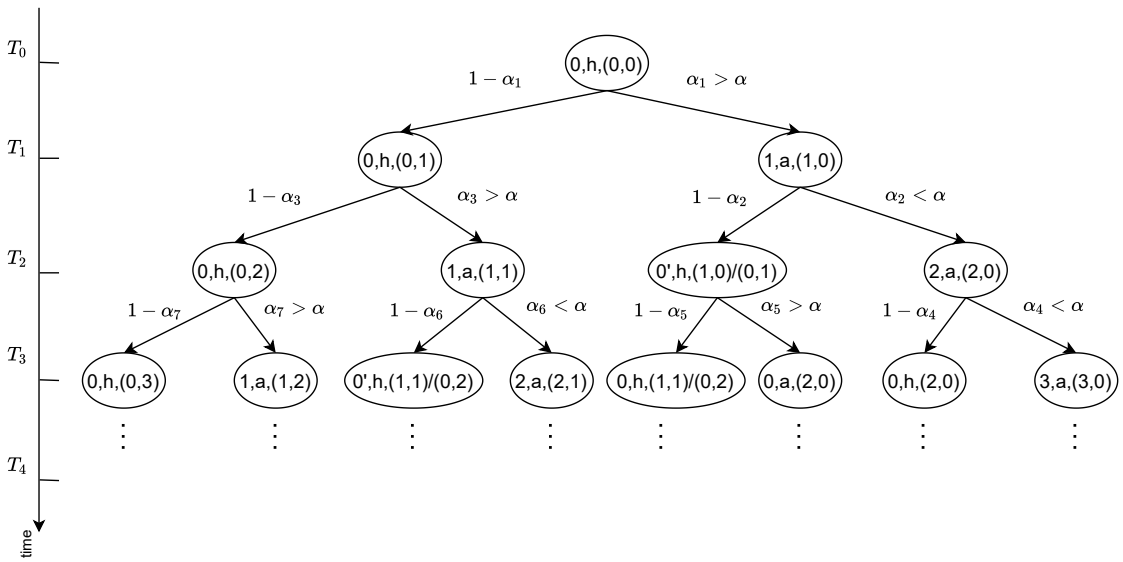


Figure 4.10: The random walk of selfish mining attacker.

Table 4.3: The actions between different states. W, R, and A represent the "Withhold", "Release", and "Adopt" actions respectively.

	BL=0	BL=0'	BL=1	BL=2	BL=3	BL>3
LD=a	W	R	W	W	W	W
LD=h	A	A	R	R	W	W

Selfish mining actions. A selfish mining attacker can execute the following actions to try to optimize its revenue.

- **Withhold.** The attacker keeps on working on her private chain.
- **Release.** The attacker publishes her private chain.
- **Adopt.** The attacker stops working on her private chain, and accepts the block that is discovered by the honest miners.

We summarize attacker's actions between different states in Table 4.3. From state s to s', the decision of action is determined by BL of s and LD of s'. For instance, $BL = 0, LD = a$ means that the attacker will withhold a block when she discovers one and has no block lead. $BL = 2, LD = h$ means that the attacker will publish her private chain when the honest miners discover a block, in this case, attacker can override the main chain, that causes an unfair revenue (2,0). Apparently, when $BL \geq 2$, the honest miners have no chance to earn blocks.

State transitions. A transition between states is triggered by a block discovery in the network. Following each transition, the attacker can take the decision to

withhold, release, or adopt a block. Thus, the probability of transition is actually the probability of a block to be discovered by the attacker or by honest miners. Let α be the hash rate share of the adversary. In [17, 15, 16], the transition probability between any two states s and s' is calculated in 4 cases: (1) it is equal to α when a block is discovered by the adversary; (2) it is equal to $1-\alpha$ when a block is discovered by the honest miners and the $BL \neq 0'$ at state s; (3) it is equal to $(1-\gamma)(1-\alpha)$ when $BL = 0'$ and a block is discovered by the honest miners that work on the adversary's branch; and (4) it is equal to $\gamma(1-\alpha)$ when $BL = 0'$ and a block is discovered by the honest miners that work on the honest miners' branch. As we have analyzed in the previous section, the transition probability can be affected by the block reception latencies of miners during the random walk process. Therefore, we have two additional cases to calculate the transition probability: (5) it is equal to $\frac{\alpha}{\alpha+(1-\alpha)*\mu}$ when $LD = h$ and a block is discovered by the adversary; and (6) it is equal to $\frac{(1-\alpha)*\mu}{\alpha+(1-\alpha)*\mu}$ when $LD = h$ and a block is discovered by the honest miners. The transition probability between any two states during the random walk process can be calculated in these 6 cases.

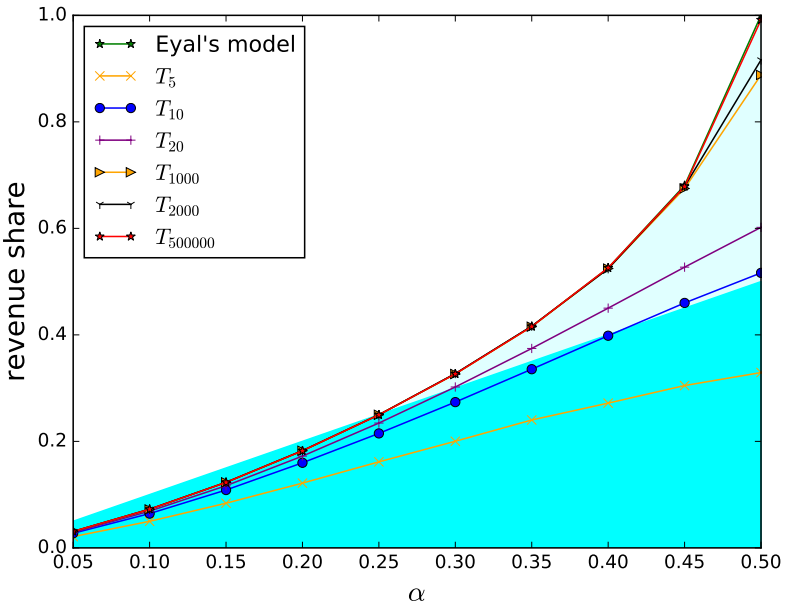


Figure 4.11: Revenue of selfish mining attacker in the general case ($\gamma = 0.5$) when $f_h = 1$. The x axis represents the hash power of attacker. The y axis represents the corresponding revenue share.

State probability. Let $pr(T_n, x)$ be the state probability of $s(T_n, x)$ ($x \in [1, 2^n]$), and $p_{\omega_1}, p_{\omega_2}, ..., p_{\omega_n}$ be the probability of transitions that happened on the

path of $s(T_n, x)$ (as we have defined in the previous section), we have:

$$pr(T_n, x) = \prod_{i=1}^{i=n} p_{\omega_i} \tag{4.7}$$

where, $p_{\omega_i} \in \{\alpha_1, \alpha_2\}$. Thus, for all possible states at T_n, we have the space of state probability $PR(T_n) : \{pr(T_n, 1), pr(T_n, 2), ..., pr(T_n, 2^n)\}$.

The expected revenue share of the selfish mining attack. Let R_a and R_h be the rewards (in number of blocks) of the adversary and the honest miners at state s respectively, the expected revenue share of the selfish mining attack $E(R)$ at T_n can be calculated as follows:

$$E(R)_{T_n} = \frac{\sum_{i=1}^{i=n}(pr(T_n, i) * R_a(T_n, i))}{\sum_{i=1}^{i=n}(pr(T_n, i) * R_a(T_n, i)) + \sum_{i=1}^{i=n}(pr(T_n, i) * R_h(T_n, i))} \tag{4.8}$$

Security bounds at different time. The selfish mining attack succeeds at T_n when $E(R)_{T_n} > \alpha$ [17]. The security bound (i.e., the minimum hash rate share that the adversary needs to succeed) can be inferred by solving this inequality. We can calculate $E(R)_{T_n}$ at different time, and show the result in Fig. 4.11, where the cyan area denotes the adversary's failure ($E(R) < \alpha$), and the light cyan area represents DS's success ($E(R) > \alpha$). The revenue share of DS attacker increases as the time steps increases, and reaches the upper bound as same as Eyal's evaluation. Remarkably, the adversary with different hash power needs different time steps to succeed, for instance, with 30% of the global hash power, the adversary would need to wait for 20 blocks in order to gain more than 30% of revenue, which means, s/he must suffer a loss of revenue share within the time period of 20 blocks.

4.5.4 The Accumulated Advantages for the Adversary

When the global computing power is separated into two parts that maintain two independent chains, the competition between the adversary and the honest miners would be the *different heights mining race* as we have introduced in Section 4.1. In this case, whenever a honest miner/pool finds a block, other honest miners would need some time to be synchronized. In general, honest miners/pools are randomly distributed in the network, the block reception latencies between them can be affected by the network properties, such as, network connectivity, network size. The maximum block reception latency between top mining pools of Bitcoin was measured as 10 seconds in 2020. In contrary to the honest miners/pools, the adversary is barely affected by the block reception latency. When $BL \geq 1$, the adversary would not need to receive the block that is discovered by the honest miners, thus, no block reception latency for the adversary. When $BL < 1$ and a

block is discovered by the honest miners, the adversary would take an relatively smaller time (i.e., the single-hop delay of Internet) to receive the block from the honest miners, compared to a few seconds delay of the honest miners, the adversary has an advantage. This results in that the probability of that the adversary finds the blocks increases during the random walk process, which gives the advantages to the adversary. These advantages can be added up to make more benefits for the adversary.

4.6 Discussion and Future Work

Cryptcurrencies rely on a loosely connected and asynchronous P2P network to exchange messages. In this thesis, we explored the effect of block reception delays on the mining process of PoW cryptocurrencies. We proposed several metrics to quantify the network fairness, and evaluated its effect on the majority and selfish mining attacks. We showed that the security bounds can be decreased because of network latencies in both attacks. By leveraging the API of mining pools, we measured that Bitcoin's default settings do not preserve network fairness, which could facilitate attacks. For instance, we evaluated that an attacker with 48.72% of the global hash power can launch a majority attack, respectively 24.02% for the selfish mining attack. If large scale deviations were to occur, we proved that a Nash equilibrium would be achieved among the larger miners (i.e., mining pools). However, this would also lead to an increased unfairness of the network, and in particular for smaller miners. It is an open problem to design fair and efficient P2P networks for PoW Blockchains, which we plan to tackle in future work.

Chapter 5

Dual Private Chain Attacks

The Bitcoin consensus algorithm relies on miners that repetitively attempt to solve cryptopuzzles, and upon success create a new block using pending transactions and append it to the public chain [8]. Bitcoin miners use a double SHA256 function with different inputs to mine [119]. A block is mined when a hash output lower than a given threshold is produced. Miners gain coins and transaction fees from the blocks they mine as a compensation for their contributed computing power when the blocks they mined have been accepted by the majority of the global hash power. In case of visible forks, each correct participant eventually accepts the longest chain of blocks.

Bitcoin's security level is traditionally measured as the proportion of the mining power that an adversary must control to successfully attack it. S. Nakamoto assumed that an adversary would not control the majority of the mining power [8]. If this assumption does not hold, an attacker is able to spend a coin twice, which is known as a double spending attack or 51% attack. The soundness of the honest majority assumption has been discussed in the literature and mechanisms have been proposed to harden the mining process against the 51% attack without managing to completely eliminate it [120, 68, 121, 122]. Some of these works also argued that it is not realistic to assume that, apart from the adversary, all miners are altruistic.

First, despite offering rewards such as newly minted coins and transaction fees to miners, the Bitcoin mining process has been shown to be vulnerable to selfish behaviors. Using selfish mining, a miner withholds mined blocks and releases them only after the honest miners have wasted computing resources mining alternative blocks. Selfish mining increases a miner's revenue beyond the fair share it would obtain by following the default Bitcoin mining protocol [17]. In practice, selfish mining has been shown to be profitable after a difficulty adjustment period in Bitcoin for any miner with more than 33% of the global hash power [16, 90], and variants of selfish mining further optimize a miner's expected revenue [18].

Second, the presence of petty compliant miners, which are also called benign dishonest miners, have been noticed and reported in Bitcoin [22, 23, 24]. These miners slightly deviate from the default Bitcoin protocol, for example by doing SPV mining [25]. In particular, Mirkin et al. showed that SPV miners, despite having no intentional will to harm the system, can lead the blockchain to an halt following a Blockchain denial-of-service attack [24]. It was estimated that more than 50% of Bitcoin's global hash power were following SPV mining in 2015 [123]. Nowadays, some important mining pools, which collectively account for 40% of the global hash power, are suspected to continue SPV mining [29, 124, 28].

In this chapter, we present the Dual Private Chain (DPC) attack on Bitcoin. This attack is, to the best of our knowledge, the first attack where an adversary temporarily sacrifices part of its hash power to favor its double spending attack. Incidentally, the DPC attack is also, to the best of our knowledge, the first attack where an adversary simultaneously manages two private chains. The first chain inhibits the public chain's growth, so that the second chain has more favorable conditions for a double spending attack. The DPC attack manages these two chains in a non trivial manner to dedicate a larger hash power to the double spending chain whenever possible.

The DPC attack lures SPV miners away from extending the public chain by releasing the block headers of created blocks that the adversary finds and keeping their transactions private, therefore creating branches on which SPV miners mine that are never accepted into the public chain. An important building block of the DPC attack is therefore a novel mining strategy that distracts SPV miners from extending the public chain. We call this strategy perishing mining since the adversary does not perceive mining revenue when it applies it to maintain its first private chain. However, the adversary perceives a net benefit when its double spending attempts succeed. Perishing mining leverages the selfishness of SPV miners to waste their hash power.

To evaluate the impact of the distraction chain on the public chain we first establish the Markov decision process (MDP) of perishing mining. From this MDP, we obtain the probability for the system to be in each state, and quantify the impact of perishing mining on the public chain, i.e., its growth rate decrease. We further describe the DPC attack and its associated MDP, and evaluate its expected success rate and minimum revenue for the attack to be the most profitable strategy based on Monte Carlo simulations. Counterintuitively, our results show that the adversary increases its double spending success rate by dedicating a fraction of its hash power to slow the public chain down, instead of attacking it frontally with all its hash power.

Overall, this work makes the following contributions.

- We present perishing mining, a mining strategy that is tailored to slow down

the progress of the public chain in presence of SPV miners. Using perishing mining an adversary releases the headers of blocks that extend the public chain so that SPV miners mine on them while honest non-SPV miners keep mining on the public chain, which effectively divides the honest miners hash power. The adversary also wastes the SPV miners' hash power whenever they mine on one of its blocks' header since it never releases the full block. We present the pseudocode of the perishing mining strategy, establish its Markov chain model and quantify its impact on the public chain growth.

• Building on perishing mining, we describe the DPC attack that an adversary can employ to double spend by maintaining up to two private chains. The first chain leverages the perishing mining strategy to slow down the public chain's growth and ease the task of the second chain, which aims at double spending. We provide the pseudocode of the attack, and characterize the states and transitions of its Markov chain model.

• We evaluate the perishing mining strategy and the DPC attack based on extensive Monte Carlo simulations. Our results indicate that perishing mining slows down the public chain progress down by 69% when the adversary owns 20% of the global power. In comparison, selfish mining, which aims at optimizing a miner's revenue share, would only decrease it down by 15%.

In presence of SPV miners, the DPC attack enables an attacker with a minority of the hash power to double spend. Our evaluation actually shows that owning 30% of the global hash power is sufficient to double spend with 100% success rate when 50% of the hash power belongs to SPV miners. With 20% of the global mining power, an attacker is able to improve its double spending success rate from 2.3% to 39% when 50% of the hash power belongs to SPV miners. In addition, the DPC attack allows an adversary to increase its revenue past the revenue it would obtain by mining honestly or running the classical double spending attack if the transaction value that it attempts to double spend is high enough.

5.1 System Model

This section introduces the categories of miners we consider, and the adversary that launches a DPC attack. Table 5.1 summarizes the notations we use throughout this chapter.

5.1.1 Bitcoin mining

Bitcoin mining is a trial-and-error process[1]. From a candidate set of transactions, miners assemble a block and use their processors to identify a hash that is smaller

[1] https://en.bitcoin.it/wiki/Block_hashing_algorithm

Table 5.1: Notations.

Symbol	Interpretation
$\alpha \in [0, 0.5]$	Mining power of the adversary.
$\beta \in [0, 1]$	Fraction of its mining power that the adversary dedicates to its first private chain.
$\mu \in [0, 0.5]$	Mining power of SPV miners.
v_t	Value of the transaction the adversary inserts in a block when starting the DPC attack and attempts to double spend.
v_b	Mining reward per block.

than a target value. Upon finding such a hash, a miner has successfully created a block and gains newly minted coins and transaction fees as a reward for its contributed computing power, after the mined block has been accepted by the majority of hash power. The public blockchain (or chain) is visible to all participants, and is maintained in normal conditions by honest miners. To distinct the public blockchain branch maintained by honest miners from the adversary maintained private blockchain branches, we use "public chain" to refer to the public branch and "private chain" to refer to a private branch.

To achieve consistency, participants accept the longest chain in case of visible forks [70, 72, 125]. However, temporary block withholding attacks have been shown to threaten Bitcoin's security [7, 19, 17, 18]. Full Bitcoin nodes monitor the network to verify block headers and verify transactions. Miners compete to identify block headers using proof-of-work.

We assume that miners follow the traditional block exchange pattern [34, 24] using the overlay network. Block dissemination over the overlay network takes seconds, whereas the average mining interval is 10 minutes. We therefore neglect accidental forks among honest miners, which occur in average about every 60 blocks [34], and evaluate mining and double spending strategies using event-based simulations where an event is the discovery of a block by a category of miner. Upon receiving a block, all miners except SPV miners verify its transactions, and then immediately relay it if they are all valid. We note v_b the mining reward that miners obtain whenever a block they have discovered is permanently included in the blockchain.

5.1.2 Miner Categories

We consider two types of honest Bitcoin miners: altruistic miners and SPV miners.

Altruistic miners follow the default mining protocol and never leave the mining network. In particular, altruistic miners are mining on the longest chain of full blocks and do not apply SPV mining. A block header that extends a concurrent chain is not sufficient to make these miners mine on it.

SPV miners [87, 123, 29, 124] are profit-driven miners. As Bitcoin leaves open the possibility for miners to accept and generate new blocks without verifying their transactions, SPV miners start mining on a new block that would belong on the longest chain as soon as they can without verifying the transactions they contain. As they do not (require to) know the transactions contained in the received blocks, they mine blocks containing no transactions (except for the coinbase transaction[2]), to avoid including a conflicting transaction in the blocks they create. We use μ to denote the share of the public mining power controlled by SPV miners. In 2015, it was estimated that more than half of computing power were doing SPV mining [123] in Bitcoin. Nowadays, some large mining pools that collectively account for more than 17% of the global hash power continue to do SPV mining on Bitcoin [29, 124] and Ethereum [25]. We analysed the Bitcoin blockchain and found that Antpool, Binance, F2pool, Huobi, Poolin, ViaBTC published empty blocks from January 2021 to February 2022 and collectively represent more than 60% of the global power. We obtained 70% with a similar analysis on Ethereum's blockchain.

The adversary owns a fraction α of the global hash power and its aim is to double spend. We consider $\alpha \in [0, 0.5]$ since with $\alpha > 0.5$ the classical double spending attack always eventually succeed. In particular, obtaining an unfair share of the mining revenue is not the adversary's main goal. When launching its attack the adversary introduces a transaction of value v_t in a block that is included in the public chain and that it attempts to double spend. We also assume that the adversary cannot break cryptographic primitives. Contrary to the selfish mining's adversary model, our model does not assume that the adversary has a privileged network access, which is required in selfish mining when the adversary releases a conflicting block it had premined in reaction to the extension of the public chain by an honest miner.

5.2 Attack Overview

This section provides a high-level description of Dual Private Chain (DPC) attacks, where an adversary maintains two private chains. It then summarizes the

[2]https://en.bitcoin.it/wiki/Coinbase

respective roles of these two private chains and their interactions.

5.2.1 Intuition

In a DPC attack, the adversary maintains two private chains from which it might release block headers or full blocks with the ultimate goal of double spending. During the attack, both of the adversary's private chains compete with the public chain and may diverge from it starting from different blocks. At a given point in time, the adversary might dedicate its full hash power to focus on one of its private chains, or divide its hash power to simultaneously extend its two private chains.

The DPC attack starts when the adversary creates a transaction of value v_t that is the basis for its double spending attempt. Once the adversary receives the block that contains its transaction it initializes both its private chains with it and start mining on it. Initially the two chains are therefore equal, but they might diverge or converge again later on depending on the created blocks. The double spending attack succeeds if the double spending chain becomes longer than the public chain and if more than $z = 6$ blocks have been included in the public chain since the block that contains the initial transaction.

Role of the Distraction Chain. The first private chain that the adversary maintains is called the ***distraction chain***. We present a strategy, namely perishing mining, that the adversary mainly employs to maintain its first private chain to waste the hash power of SPV miners and slow down the progress of the public chain. Whenever the adversary divides its hash power to simultaneously mine on its two private chains it dedicates $\alpha\beta$ of its hash power to mine on its first private chain. This chain is private in the sense that the adversary never releases the full blocks, but only the corresponding block headers. The strategy that the adversary applies on its distraction chain divides the honest miners so that they mine on different blocks, and wastes the hash power of SPV miners, which collectively account for hash power μ. The adversary leverages a BDOS-like attack to only share the header of blocks it discovers on the distraction chain (see §5.3). As the body of those blocks contain adversary-created transactions that are never publicly released, only SPV miners mine on them. In this way, the adversary distracts the SPV miners from mining on the public chain.

Role of the Double Spending Chain. The adversary maintains a second private chain to attempt to double spend, and we therefore call this chain the ***double spending chain***. Whenever the adversary is simultaneously mining on its two private chains it dedicates $\alpha(1-\beta)$ of the global hash power to its second private chain. This chain is private in the sense that, even though block headers might be released, the actual blocks it contains are only published if the double spending attack is successful. Following previous analyses [8, 19], we consider that a double spending attempt is successful when: (i) the double spending chain's length is

larger than or equal to the public chain's length; and (ii) z-1 blocks have been appended after the block that contains the adversary's initial transaction ($z = 6$ in Bitcoin).

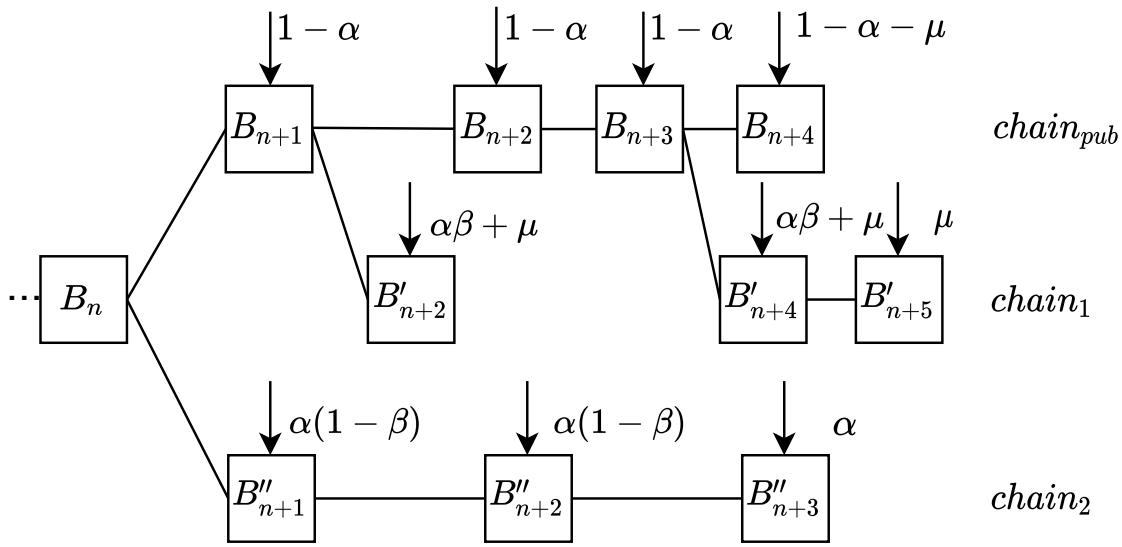


Figure 5.1: Illustration of the DPC attack on SPV miners. Whenever the distraction chain discovers a block before the public chain, the adversary releases the block header so that SPV miners mine on a different block than altruistic miners.

5.2.2 Interplay Between the Two Chains

Whenever the two categories of honest miners, i.e., altruistic and SPV miners, are mining on the same block the adversary divides its hash power to concurrently mine with hash power $\alpha\beta$ on the last block of its distraction chain, which is then equal to the public chain, and mine with mining power $\alpha(1-\beta)$ on its double spending chain. The adversary's goal is then to create a fork so that altruistic and SPV miners mine on different blocks. Whenever the honest miners mine on different blocks, the adversary focuses its whole hash power α on its double spending chain. The two chain abstractions never disappears since the two chains become different again if the public chain becomes longer than the double spending chain.

We use Fig. 5.1 to illustrate a state of a DPC attack. In this figure, time flows from left to right. This Figure shows multiple blocks that were found by miners in the network, the heads of the adversary's private chains, and indicates the overall mining power that was or is dedicated on a block whenever the block was part of the longest chain for a category of miners (using vertical arrows).

The attack is initialized based on block B_n. A pair of conflicting transactions that are then generated by the adversary, one is for paying to the merchant that

is public and included into B_{n+1}, another one is to transfer the same coins to the adversary that is hidden until the double spending succeeds. The adversary's final double spending chain is made of blocks B_n, $B"_{n+1}$, $B"_{n+2}$ and $B"_{n+3}$. At some point in time the adversary's distraction chain was equal to B_n, B_{n+1} and B'_{n+2}, but it is equal to B_n, B_{n+1}, B_{n+2}, B_{n+3}, B'_{n+4}, B'_{n+5} in this state. The non-SPV miners are mining on block B_{n+4} with power $1-\alpha-\mu$, while SPV miners are mining on block B'_{n+5} with power μ and the adversary is mining on block $B"_{n+3}$ with its whole hash power α (because the altruistic and the SPV miners are mining on different blocks).

Whenever the adversary discovered a block on its private chain, it released the header of the block so that SPV miners (of power μ) mined on it, which happened with block B'_{n+2}, while altruistic miners (of power $1-\alpha-\mu$) kept mining on the latest full block of the public chain. However, the adversary did not release the corresponding block body, which led all blocks of the distraction chain to eventually become invalid.

5.3 The Dual Private Chains Attack

This section presents the details of the DPC attack, which attempts to lure SPV miners away from extending the public chain and facilitate a double spending attack. We first describe perishing mining, a strategy that a miner can use to slow down the progress of the public chain by making the altruistic and SPV miners mine on different blocks. We then describe the full DPC attack that builds on perishing mining to maintain the adversary's first private chain.

5.3.1 Perishing mining

We call *perishing mining* the strategy that the adversary uses on the distraction chain (whenever it is mining on it). With perishing mining, the adversary uses $\alpha\beta$ of the global hash power to inhibit the public chain's growth. To do so, upon discovering a block that extends the public chain, the adversary only relays the block header and never releases the block body. Upon reception of this block header, SPV miners accept it and mine on it. This reduces the speed at which the public chain is extended, because SPV miners work on a chain that cannot be validated, and facilitates the double spending attack the adversary is working on with computing power $\alpha(1-\beta)$.

After the initialization of the perishing mining strategy, the distraction chain and the public chain mine on the same block. The adversary's action then depends on whether the next block is discovered by the public miners or by itself, as shown in Algorithm 3. First, when the adversary discovers a block B_{n+1} that makes

Algorithm 3: *The perishing mining strategy.*

```
1:  function Init()
2:      chain_pub ← publicly known blocks
3:      chain_1 ← publicly known blocks
4:      l_pub = l_1 = 0
5:      mine on chain_1's head
6:
7:  upon event Adv. finds block B on chain_1 do
8:      Adv. appends B to chain_1
9:      l_1++
10:     release Header(B)
11:     mine on B
12:
13: upon event Non-SPV miners find block B on chain_pub do
14:     Non-SPV miners append B to chain_pub
15:     l_pub++
16:     if l_pub > l_1 then
17:         Init()
18:
19: upon event SPV miners find block B do
20:     if l_1==0 then
21:         Init()
22:     else
23:         SPV miners append B to chain_1
24:         l_1++
25:         mine on B
```

its distraction chain longer than the public chain, it releases the corresponding block header to the network (Alg. 3, line 10). Upon receiving this header, the SPV miners start mining based on it, while the honest non-SPV miners continue working on block B_n. Second, when the altruistic miners discover a block, the public chain is extended (Alg. 3, line 15). Third, when the SPV miners find a block, the public chain is extended when the public chain is equal to the private chain (Alg. 3, line 21), and the private chain is extended when the public chain is not equal to the private chain (Alg. 3, line 24).

Fig. 5.2 illustrates the MDP model of the perishing mining strategy. In this Markov chain, α, μ and $1-\alpha-\mu$, are respectively the probabilities for the adversary, the SPV miners and non-SPV miners to discover a block. In the context of the DPC attack, the adversary does not always follow the perishing mining strategy, and when it does only dedicates a part of its hash power to it. In this Figure, we assume that the adversary is dedicating a power α to perishing mining.

State 0 represents the situation where the public chain and the distraction chain are synchronized. State $0'$ represents the situation where the public chain and the distraction chain have the same length but are not synchronized. The

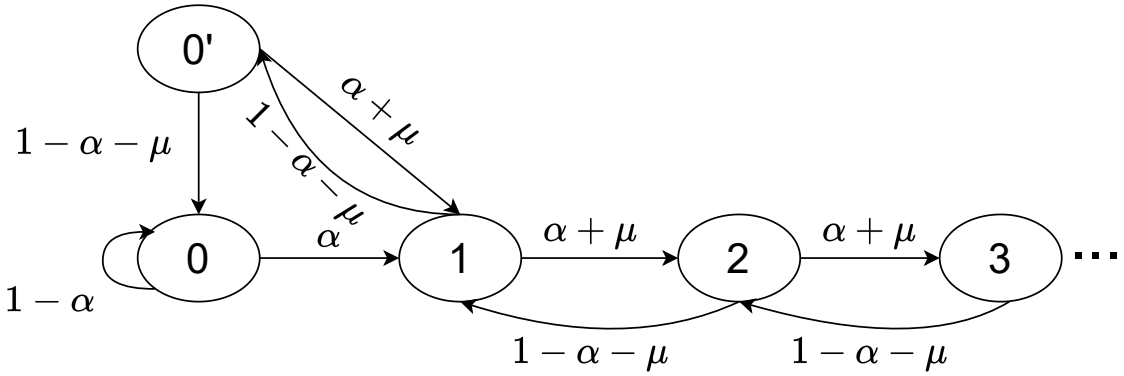


Figure 5.2: The Markov chain model of perishing mining.

other states are denoted by the length difference between the distraction chain and the public chain, i.e., $l_1 - l_{pub}$. Note that this difference is always positive since the distraction chain is reinitialized when the public chain becomes longer than the distraction chain.

We note p_i the probability for the system to be in state i. From the MDP model, one can obtain the following equations.

$$\begin{cases} \alpha p_0 = (1-\alpha-\mu)p_{0'} \\ (\alpha+\mu)p_{0'} = (1-\alpha-\mu)p_1 \\ (\alpha+\mu)p_1 = (1-\alpha-\mu)p_2 \\ \forall k \geq 2 : (\alpha+\mu)p_k = (1-\alpha-\mu)p_{k+1} \\ \sum_{k=0}^{\infty} p_k + p_{0'} = 1 \end{cases} \tag{5.1}$$

These equations can be manipulated to obtain the probability for the system to be in each state. Excluding the possibility for $\alpha+\mu$ to be equal to 1, which would mean that there are no altruistic miners in the system, we have to distinguish two cases, depending on whether $\frac{\alpha+\mu}{1-\alpha-\mu}$ is smaller or larger than 1, because p_k depends on the sum of a geometric series where the k-th value is equal to $\left(\frac{\alpha+\mu}{1-\alpha-\mu}\right)^k$. When $\frac{\alpha+\mu}{1-\alpha-\mu} < 1$, the system contains a majority of altruistic miners, and the state

probabilities are:

$$\begin{cases} p_{0'} = \dfrac{\alpha - 2\alpha^2 - 2\mu\alpha}{\alpha^2 + 3\mu\alpha - 2\alpha - 3\mu + 2\mu^2 + 1} \\ p_0 = \dfrac{1-\alpha-\mu}{\alpha} p_{0'} \\ p_1 = \dfrac{\alpha+\mu}{1-\alpha-\mu} p_{0'} \\ \forall k \geq 2 : p_k = \dfrac{\alpha+\mu}{1-\alpha-\mu} p_{k-1} \end{cases} \tag{5.2}$$

When $\frac{\alpha+\mu}{1-\alpha-\mu} > 1$, the system contains a minority of altruistic miners and the geometric series does not converge, but we express the probability for a system to be in a given state after T transitions. We can express the state probabilities depending on $\mathsf{GS}(T) = \frac{1-(\frac{\alpha+\mu}{1-\alpha-\mu})^T}{1-\frac{\alpha+\mu}{1-\alpha-\mu}}$, which is the sum of the T first values of the geometric series, as follows:

$$\begin{cases} p_{0'} = \dfrac{1}{\frac{1-\alpha-\mu}{\alpha} + \mathsf{GS}(T)} \\ p_0 = \dfrac{1-\alpha-\mu}{\alpha} p_{0'} \\ p_1 = \dfrac{\alpha+\mu}{1-\alpha-\mu} p_{0'} \\ \forall k \geq 2 : p_k = \dfrac{\alpha+\mu}{1-\alpha-\mu} p_{k-1} \end{cases} \tag{5.3}$$

One can notice that with perishing mining, the public chain is only extended when: (i) the honest miners or the SPV miners discover a block in state 0; or (ii) the honest miners discover a block in one of the remaining states. Thus, the chain growth rate, which is defined as the probability for the public chain to be extended with a full block when a block is discovered, is equal to $p_0(1-\alpha) + \sum_{k=1}^{\infty}(p_k(1-\alpha-\mu)) + p_{0'}(1-\alpha-\mu)$. We discuss in Section 5.5 how an adversary can leverage this formula to evaluate the proportion μ of SPV miners in a real system.

5.3.2 Combining Perishing Mining and Double Spending

The DPC attack leverages the perishing mining strategy to distract SPV miners and facilitate double spending.

Algorithm 4 details the attack's pseudocode, where l_1, l_2, and l_{pub} represent the length of the first private chain $chain_1$, the second private chain $chain_2$, and the public chain $chain_{pub}$ respectively. During the DPC attack, we always have the two following invariants: $l_2 \leq l_1$ and $l_{pub} \leq l_1$. The distraction chain is therefore always

the longest chain among the three chains, and can adopt the public chain and the double spending chain when it is not the longest chain. For example, if it happens that the double spending becomes the longest chain then the distraction chain is set to be equal to the double spending chain. As a consequence, the SPV miners would mine on the headers of the double spending chain, which would facilitate the double spending attack.

When the DPC attack starts, all three chains are equal and all miners mine on the same block (Alg. 4, line 1). The adversary's actions are defined in reaction to block discoveries.

When the adversary finds a block on the distraction chain (Alg. 4, line 9), it releases the corresponding block header so that SPV miners mine on it, because the distraction chain is the longest chain. If the two private chains are equal the newly found block also extends the double spending chain. As a consequence of the adversary extending the distraction chain the honest miners are mining on different blocks, the altruistic miners mine on the last full block of the public chain while the SPV miners mine on the last block header of the distraction chain. The adversary then allocates all its hash power (α) to mining on the double spending chain.

When the adversary finds a block on its double spending chain (Alg. 4, line 22), it releases the block header if the second private chain becomes the longest chain. In this case, the SPV miners then mine on the double spending chain. The first private chain also adopts the second private chain so that the total hash power on extending double spending chain is $\alpha + \mu$. When the second private chain is shorter than the public chain, the adversary keeps mining on it with $1 - \beta$ of its hash power. As soon as the double spending chain becomes longer than the public chain and that at least 6 blocks have been appended to the public chain since the beginning of the attack, the adversary uses the double spending chain to override the public chain, and the DPC attack succeeds.

When the altruistic miners find a block (Alg. 4, line 34), they extend the public chain. If the public chain becomes the longest chain, then all honest miners will mine on the public chain and the adversary modifies its first private chain so that it adopts the public chain. The adversary then allocates $\alpha\beta$ of hash power to its distraction chain so that it tries to generate a block that will divide again the honest miners.

When the SPV miners find a block (Alg. 4, line 45), three cases are possible. First, the double spending chain and the distraction chain are extended if the two private chains were equal. Second, the public chain and the first private chain are extended if the public chain and the first private chain were equal. Finally, in the other cases, the first private chain is extended and the SPV miners have wasted their hash power because this private chain never become accepted by the

non-SPV miners, since the adversary does not release the block bodies.

5.3.3 Markov Decision Process of the DPC Attack

We establish the Markov decision process (MDP) of the DPC attack by simultaneously considering the two private chains and observing that each state is a 5-tuple $(l_{pub}, l_1, l_2, s_{(pub,1)}, s_{(1,2)})$. l_{pub}, l_1, and l_2 are respectively the lengths of the public chain $chain_{pub}$, the first private chain $chain_1$, and the second private chain $chain_2$. $s_{(pub,1)}, s_{(1,2)} \in \{\texttt{t(rue)}, \texttt{f(alse)}\}$ respectively indicate whether $chain_{pub}$ is equal to $chain_1$, and whether $chain_1$ is equal to $chain_2$.

We identified 10 types of states, and we determined for each state its possible transitions, the transition probabilities, and the distribution of mining rewards. Because the number of states of this MDP is infinite we summarize this information in Table 5.2. In particular, case 0 is the initial state of the attack. Case 4 captures the success of the DPC attack. Cases 1.x, 2.x, 3.x are all possible intermediary states and consider scenarios that differ based on the lengths of the chains, and whether or not they are equal. Our Monte Carlo simulations are based on this MDP.

We emphasize that an adversary that executes the DPC attack earns a mining reward only when the double spending chain succeeds. In this case, the adversary earns the block mining reward that corresponds to the private blocks it mined that end up in the public chain and the value of the transaction it managed to double spend. We note value of blocks v_b, but also the value of double spent transactions v_t. We evaluate the DPC attack's success rate and revenue in Section 5.4.

5.4 Analysis using Monte Carlo Simulations

This section evaluates the perishing mining strategy and the DPC attack using Monte Carlo simulations that react based on the even of block discovery.

5.4.1 Methodology and Settings

We evaluate perishing mining and the DPC attack using random walks in their respective Markov decision processes. Our evaluations are based on Python scripts. In each scenario we consider, we simulate the creation of 2,016 blocks, repeat each configuration 10,000 times and report the average of the metrics of interest. Simulating the creation of 2,016 blocks maintains the mining difficulty constant during the experiment since Bitcoin's mining difficulty is adjusted every 2,016 blocks. We quantify the impact of perishing mining on the public chain's growth rate, evaluate the double spending success rate of the DPC attack, and analyze

the adversary's expected revenue. We compare the success rate of the DPC attack to the success rate of the classical double spending attack using the success rate formulas that were obtained by Nakamoto [8] and Rosenfeld [33]. We study the various strategies with $\alpha, \mu \in [0, 0.1, 0.2, 0.3, 0.4, 0.5]$ and $\beta \in [0, 1]$ (by 0.01 steps).

To evaluate the adversary's revenue when it executes the DPC attack we consider the mining reward v_b per block to be equal to 6.25 BTC (which is the case since May 2021). We do not consider the transaction fees as their impact is negligible on the adversary's revenue. We evaluate the adversary's revenue depending on the value v_t of the transaction it attempts to double spend compared to the mining reward per block, since it relates to the revenue it would obtain by mining honestly. The adversary might distribute the value of v_t over the transactions of the attack initialization block since each Bitcoin block (1MB) normally includes from 1,500 to 2,500 transactions.

5.4.2 Impact of Perishing Mining on Chain Growth

In a DPC attack, the adversary dedicates at most a fraction β of its hash power α at a given time to follow the perishing mining strategy. Perishing mining facilitates the double spending attack the adversary works on with its remaining hash power. For this experiment, we consider a scenario where the adversary would dedicate a fraction of its full hash power to follow the perishing mining strategy to quantify its effect on the growth rate of the public chain.

Fig. 5.3 represents the relative public chain growth rate of a system under attack, which is expressed as a fraction (in %) of the public chain growth rate in the attack-free case. We compare perishing mining to selfish mining and vary the global hash power μ of SPV miners 0 to 0.5 (i.e., ranging from 0% to 50% of the global hash power). One can see that the public chain is extended at a lower rate when the adversary's power increases and when the global power of SPV miners increases. Compared to selfish mining, which was designed for another purpose namely to increase a miner's revenue, perishing mining has a greater effect on the public chain growth rate, i.e., even when $\mu = 0$. When $\mu = 0.5$ and $\alpha = 0.5$ the public chain growth is almost equal to 0 because there are no honest non-SPV miners in the system and because as soon as the adversary finds a block and releases its header SPV miners would mine on it and therefore never create a chain of full blocks.

Note that an adversary that follows the perishing mining strategy does not perceive any mining revenue because it never releases the transactions of the blocks it discovers. Indeed, the distraction chain, the private chain that is maintained using the perishing mining strategy, never leads the honest non-SPV miners to mine on it because the full blocks are never released, which prevents the adversary from obtaining a mining revenue.

5.4.3 Double Spending Success Rate

Fig. 5.4 illustrates the success rates of the DPC attack for different μ and with the best β value that we obtained experimentally, and for $\beta = 0$. It is interesting to observe the differences between the lines corresponding to a given μ with the best β value and $\beta = 0$ to see that maintaining two chains makes a real difference. An adversary would be able to determine the best β after estimating μ, as we discuss in Section 5.5.

In the presence of SPV mining (i.e., when $\mu > 0$), the DPC attack's success rate is higher than the one of the traditional double spending attack. For example, if $\mu = 0.1$ and $\alpha = 0.3$, the DPC attack's success rate is equal to 31.0% while it is equal to 16% for the classical double spending attack.

The DPC attack lowers Bitcoin's safety bound (i.e., the minimum hash power that the adversary needs to double spend). For instance, when $\mu = 0.4$, a DPC adversary with 35% of the global hash power could completely manipulate the blockchain, which is more threatening than the existing block withholding attacks. Assuming $\mu = 0.5$ is realistic since in 2015 more than 50% of Bitcoin's hash power were doing SPV mining [123]. The success rate of the double spending attack (with 6 confirmations) with $\alpha = 0.2$ (the power of the biggest mining pool) also increases from 2.3% to 39% depending on μ by replacing the classical double spending attack by the DPC attack.

5.4.4 The Effect of The DPC Attack on Bitcoin's Sustainability

Fig. 5.5 plots the minimum value for $\frac{v_t}{v_b}$ that the adversary would need to launch a profitable DPC attack (i.e., the adversary would earn more than performing mining honestly). When $\mu = 0.4$ and $\alpha = 0.2$, if the value of the transactions embedded in the initial block is larger than 10 times the block mining reward then the DPC attack becomes profitable. Currently, Bitcoin's block reward is 6.25BTC, and each block can contain approximately 2500 transactions. In this case, the suggested average safe transaction value to prevent DPC attack is 0.025BTC. In the future, v_b would be decreased due to Bitcoin's block reward halving countdown, which would decrease both the threshold to launch profitable DPC attacks and the safe transaction value. We are therefore reaching the same conclusion that Carlsten et al.'s work [23], we observe that the DPC attack has a negative impact on Bitcoin's sustainability.

5.4.5 Adversary's Revenue

Upon successfully executing a double spending attack, the on-chain revenue of the adversary consists of the value of the double spent transaction added to the mining rewards associated to the blocks that it mined and that are part of the longest chain. We calculate the expected revenue V_{dpc} of the DPC attacker as follows:

$$V_{dpc} = v_t k + \sum_{i=1}^{k} v_b(i) \tag{5.4}$$

where k represents the number of successful double spending attacks during the simulation, v_t notes the value of the double spent transactions, and the set $\{v_b(1), v_b(2), ..., v_b(k)\}$ denotes the adversary's block revenue for each double spending.

We now plot in more details the adversary's revenue when $vt = 10v_b$ to compare the different strategies the adversary might follow, and to provide additional information that the logarithmic scale of Fig. 5.5 is not showing. Larger v_t values become more realistic as time passes since the block mining reward is regularly halved. Fig. 5.6 compares the expected adversary's revenue for the DPC attack with the revenue it would perceive using honest mining (Honest), selfish mining (SM) [17], the double spending attack (DS) [19], and the combination of double spending attack and selfish mining attack based on a single private chain (SM+DS) [16]. In this experiment, the adversary generates a block with transactions that it attempts to double spend and continues in its attack until it succeeds or until the end of the simulation. Each time its attack succeeds, the adversary double spends and immediately generates another block of transactions that it attempts to double spend. In particular, the adversary keeps extending its double spending chain if it is longer than 6 blocks and longer than the public chain until it can double spend, i.e., until the public chain contains at least 6 blocks.

Selfish mining is less profitable than honest mining, which is expected because the mining difficulty is constant in our experiments [17, 16]. When $v_t = 10v_b$ and $\alpha \in [0, 0.5]$, the DPC attack is the only strategy for which the adversary's expected revenue can be higher than with honest mining if $\mu > 0$ and if α is large enough. For example, when $\mu > 0.4$, an adversary with at least 20% of the global hash power (i.e., $\alpha = 0.2$) benefits more from the DPC attack than from honest mining. Using larger v_t values would also make the DPC attack more profitable than other strategies for given μ and α values.

When $v_t = 10 * v_b$ and $\mu = 0.2$, an adversary with 34% of the global hash power can earn more with the DPC attack than by performing honest mining. Such bound also decreases when v_t increases (e.g., the minimum α would be 27% when $v_t = 100 * v_b$ and $\mu = 0.2$).

One can notice the inflexion points of the DPC attack lines. For example, the DPC attack's revenue when $\mu = 0.5$ and $\alpha = 0.4$ is lower than with $\mu = 0.4$

and $\alpha = 0.4$. In those cases, the adversary's second private chain is much longer than the public chain and the adversary has to wait to for the public chain to be long enough since a double spending can happen only when the public chain contains at least 6 blocks. For example, the adversary would have to wait if its private chain contains 20 blocks while the public chain only has 3 blocks. The adversary could update its strategy to further increase its revenue by detecting that its double spending attack will eventually succeeed and immediately starting the next attack.

5.5 Discussion

5.5.1 Attack Variants

We have presented the DPC attack we found to be the most effective under the constraint that the adversary might split its hash power in two constant parts $\alpha\beta$ and $\alpha(1-\beta)$. Similarly to what happened with selfish mining and its variants [7, 19, 17, 18], we foresee that one could devise variants of the DPC attack we present. In these variants the adversary would mine on different blocks depending on the system's state, or dedicate a different fraction of its hash power to extend each of its two private chains. We leave the study of these variants to future work.

5.5.2 Estimating μ and Selecting β

It is sufficient for the adversary to know an approximate value of parameter μ, which is the proportion of the global hash power that belongs to SPV miners, for a DPC attack to be successful, as our experimental results demonstrate. However, in practice, an adversary would be able to optimize its DPC attack by precisely determining μ. The adversary can estimate μ based on some public websites [29], or establish direct connections with the public mining pools to perform a statistical analysis. Moreover, the perishing mining strategy that we present in this chapter can be used as a probing technique to measure μ, since the adversary can directly monitor its impact on the public chain growth and compute μ based on it. Once the exact value of μ is known, an adversary can find the best β for the DPC attack by replicating our experiments.

5.5.3 Reinitializing the Double Spending Chain

Given a time period, a clever adversary would restart the double spending attack whenever its estimated success rate would become lower than the initial success rate of the attack. For example, the adversary should restart the double spending

attack whenever it does not discover any block and 6 blocks have been appended to the public chain. Such a reinitialization strategy would further optimize the adversary's revenue but would not change the attack's success rate. Establishing exact reinitialization conditions for the DPC attack is more complicated than with the classical double spending attack because the adversary does not dedicate a constant hash power to its double spending chain and is therefore future work. However, approximate conditions could be obtained through simulations.

5.5.4 Attack Detection and Prevention

The DPC attack leverages the fact that SPV miners accept block headers without waiting for and verifying the corresponding block bodies (i.e., the transactions). By doing so, they can start working on the next block earlier than the altruistic miners. Therefore, it is not safe to assume that all miners would deliberately choose to stop SPV mining. To prevent the DPC attack, we suggest that the SPV miners setup a bound between the block header arrival time and the block body arrival time. The SPV miners would stop mining if the time bound is reached. This would mitigate the impact of the DPC attack. However, the adversary could also update its strategy to regularly send the unmatched block bodies so that SPV miners do not stop mining on its blocks. In this case, the SPV miners would not help to extend the double spending chain, but the DPC attack would still remain more harmful than the classical double spending attack. For instance, based on our experiments, with $\alpha = 0.2$ and $\mu = 0.5$ the DPC attack's success rate would still be equal to 28% instead of 2.3% with the classical double spending attack and 39% with the DPC attack presented in this chapter.

5.6 Implications and Insights

In this chapter, we proposed perishing mining, a novel mining strategy that divides honest miners by leading SPV miners to waste their hash power on a block header generated by the adversary while non-SPV miners keep mining on the latest full block. Building on perishing mining, we described the dual private chain (DPC) attack where an adversary sacrifices a part of its hash power to run the perishing mining strategy and launch a double spending attack, which benefits from the fact that honest miners are divided. We established the Markov decision process (MDPs) of perishing mining and of the DPC attack, and used Monte Carlo simulations to quantify the impact of perishing mining on the public chain growth, evaluate the success rate and the expected revenue of the DPC attack. Our performance evaluation showed that the DPC attack is more powerful than the classical double spending attack as soon as a fraction of the miners are SPV miners. For

instance, an adversary with 30% of the global hash power can always double spend if 50% of the network is SPV mining. For an adversary with sufficient funds or with sufficient hash power, the DPC attack is more profitable than all currently known mining strategies, such as honest mining, selfish mining, and the classical double spending attack.

Algorithm 4: *The DPC attack.*

```
 1: function Init do
 2:     chain_pub ← publicly known blocks
 3:     chain_1 ← publicly known blocks
 4:     chain_2 ← publicly known blocks
 5:     mine on chain_1's head with power αβ
 6:     mine on chain_2's head with power α(1 − β)
 7:     l_pub = l_1 = l_2 = 0
 8:
 9: upon event Adv. finds a block B on chain_1 do
10:     if chain_1≠chain_2 then
11:         Adv. appends B to chain_1
12:         l_1++
13:         release Header(B)
14:         mine on chain_2's head with power α
15:     else if chain_1==chain_2 then
16:         Adv. appends B to chain_1 (and chain_2)
17:         l_1++
18:         l_2++
19:         release Header(B)
20:         mine on B with power α
21:
22: upon event Adv. finds a block B on chain_2 do
23:     if chain_1≠chain_2 then
24:         Adv. appends B to chain_2
25:         l_2++
26:         release Header(B)
27:         mine on B with α(1 − β)
28:     else if chain_1==chain_2 then
29:         Adv. appends B to chain_1 (and chain_2)
30:         Execute lines 17 to 20
31:     if l_2 ≥ l_pub ≥ 6 then
32:         override chain_pub with chain_2
33:
34: upon event Non-SPV miners find block B on chain_pub do
35:     if l_1==l_pub then
36:         Non-SPV miners append B to chain_pub
37:         chain_1 = chain_pub
38:         l_pub++
39:         l_1++
40:         mine on B with power αβ
41:     else if l_1>l_pub then
42:         Non-SPV miners append B to chain_pub
43:         l_pub++
44:
45: upon event SPV miners find block B do
46:     if chain_1==chain_2≠chain_pub then
47:         SPV miners append B to chain_2 (and chain_1)
48:         Execute lines 17 to 20
49:     else if chain_1==chain_pub then
50:         SPV miners append B to chain_1 (and chain_pub)
51:         Execute lines 37 to 40
52:     else
53:         SPV miners append B to chain_1
54:         l_1++
55:         mine on chain_2 with power α
```

Table 5.2: State transition of DPC attack that targets SPV miners. We use (r_a, r_h) to denote the revenue of the adversary and other miners, v_b to denote the value of a single block, and v_t to denote the value of the double-spent transactions. Because the SPV miners could help to extend the double spending chain, we use n_{spv} to denote the number of blocks that were discovered by the SPV miners on the 2nd private chain.

State S $(l_{pub}, l_1, l_2, s_{(pub,1)}, s_{(1,2)})$	**Event**	**Prob.**	**Destination state**	**Revenue** (r_a, r_h)
Case 0 (Contains Init state) $(l_{pub}, l_1, l_2, \texttt{t}, \texttt{t})$	Adv. extends $chain_1$=$chain_2$	α	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	SPV or non-SPV extend $chain_{pub}$	$1-\alpha$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f}))$	(r_a, r_h+v_b)
Case 1.1 $(l_{pub}, l_1 > l_{pub}, l_2, \texttt{f}, \texttt{t})$	Adv. or SPV extend $chain_1$=$chain_2$	$\alpha+\mu$	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h+v_b)
Case 1.2 $(l_{pub}, l_1 = l_{pub}, l_2, \texttt{f}, \texttt{t})$	Adv. or SPV extend $chain_1$	$\alpha+\mu$	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f})$	(r_a, r_h+v_b)
Case 2.1 $(l_{pub}, l_1 = l_{pub}, l_2 < l_{pub}, \texttt{t}, \texttt{f})$	Adv. extends $chain_1$	$\alpha\beta$	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	Adv. extends $chain_2$	$\alpha(1-\beta)$	$(l_{pub}, l_1, l_2+1, \texttt{t}, \texttt{f})$	(r_a, r_h)
	SPV or non-SPV extend $chain_{pub}$	$1-\alpha$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f})$	(r_a, r_h+v_b)
Case 2.2 $(l_{pub}, l_1 = l_{pub}, l_2 = l_{pub}, \texttt{t}, \texttt{f})$	Adv. extends $chain_1$	$\alpha\beta$	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	Adv. extends $chain_2$	$\alpha(1-\beta)$	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	SPV or non-SPV extend $chain_{pub}$	$1-\alpha$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f})$	(r_a, r_h+v_b)
Case 3.1 $(l_{pub}, l_1 > l_{pub}, l_2 < l_1, \texttt{f}, \texttt{f})$	Adv. extends $chain_2$	α	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	SPV extend $chain_1$	μ	$(l_{pub}, l_1, l_2+1, \texttt{f}, \texttt{f})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h+v_b)
Case 3.2 $(l_{pub}, l_1 > l_{pub}, l_2 = l_1, \texttt{f}, \texttt{f})$	Adv. extends $chain_2$	α	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	SPV extend $chain_1$	μ	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h+v_b)
Case 3.3 $(l_{pub}, l_1 = l_{pub}, l_2 < l_1, \texttt{f}, \texttt{f})$	Adv. extends $chain_2$	α	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	SPV extend $chain_1$	μ	$(l_{pub}, l_1, l_2+1, \texttt{f}, \texttt{f})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f})$	(r_a, r_h+v_b)
Case 3.4 $(l_{pub}, l_1 = l_{pub}, l_2 = l_1, \texttt{f}, \texttt{f})$	Adv. extends $chain_2$	α	$(l_{pub}, l_1+1, l_2, \texttt{f}, \texttt{f})$	(r_a, r_h)
	SPV extend $chain_1$	μ	$(l_{pub}, l_1+1, l_2+1, \texttt{f}, \texttt{t})$	(r_a, r_h)
	Non-SPV extends $chain_{pub}$	$1-\alpha-\mu$	$(l_{pub}+1, l_1+1, l_2, \texttt{t}, \texttt{f})$	(r_a, r_h+v_b)
Case 4 (Double spending) S s.t. $l_2 \geq l_{pub} \geq 6$	(instantaneous transition) $chain_{pub} = chain_2$	1	$(l_{pub}=0, l_1=l_1-l_{pub},$ $l_2=0, s_{(pub,1)}, \texttt{t})$	$(r_a+(l_2-n_{spv})v_b+v_t,$ $r_h-(l_{pub}-n_{spv})v_b)$

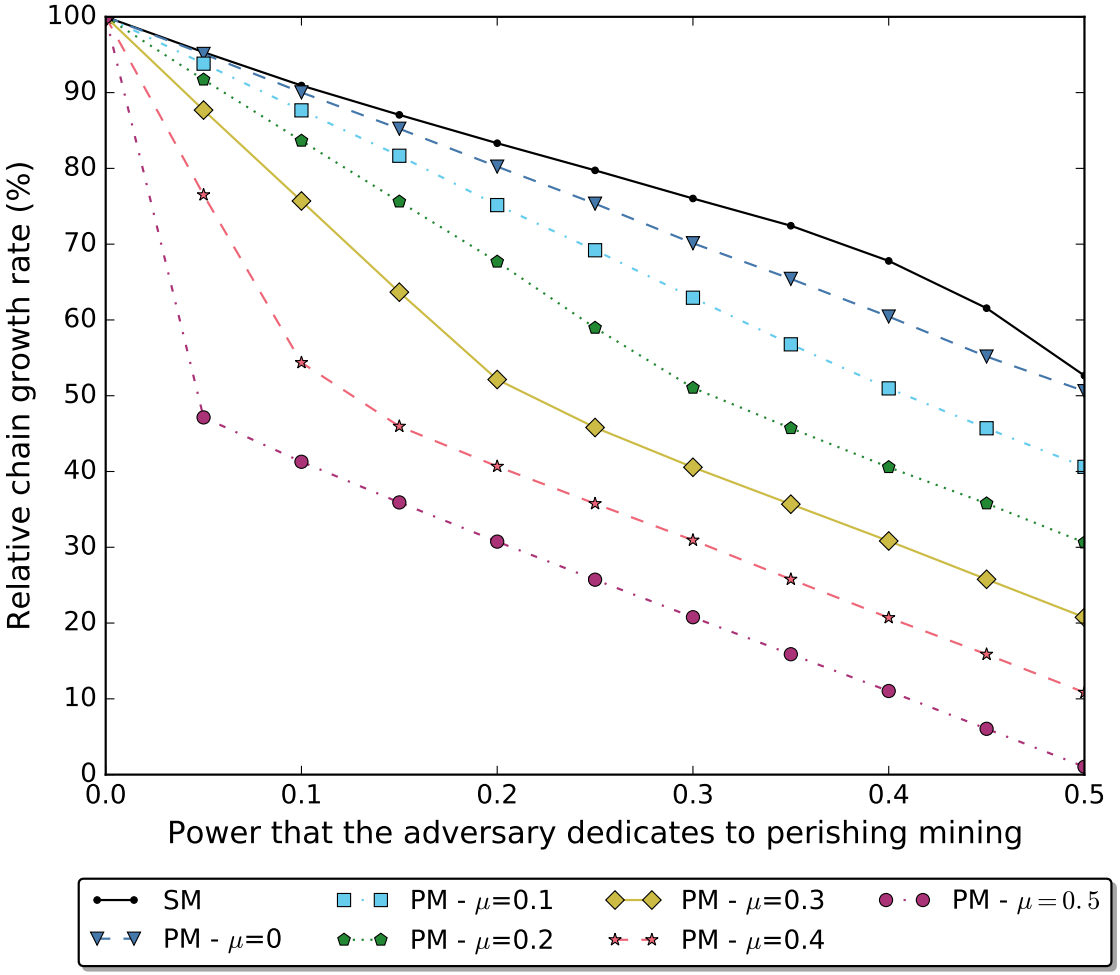


Figure 5.3: Relative growth rate of the public chain (compared to the attack-free case) when the adversary uses selfish mining (SM) or perishing mining (PM) and when SPV miners own a fraction μ of the global power.

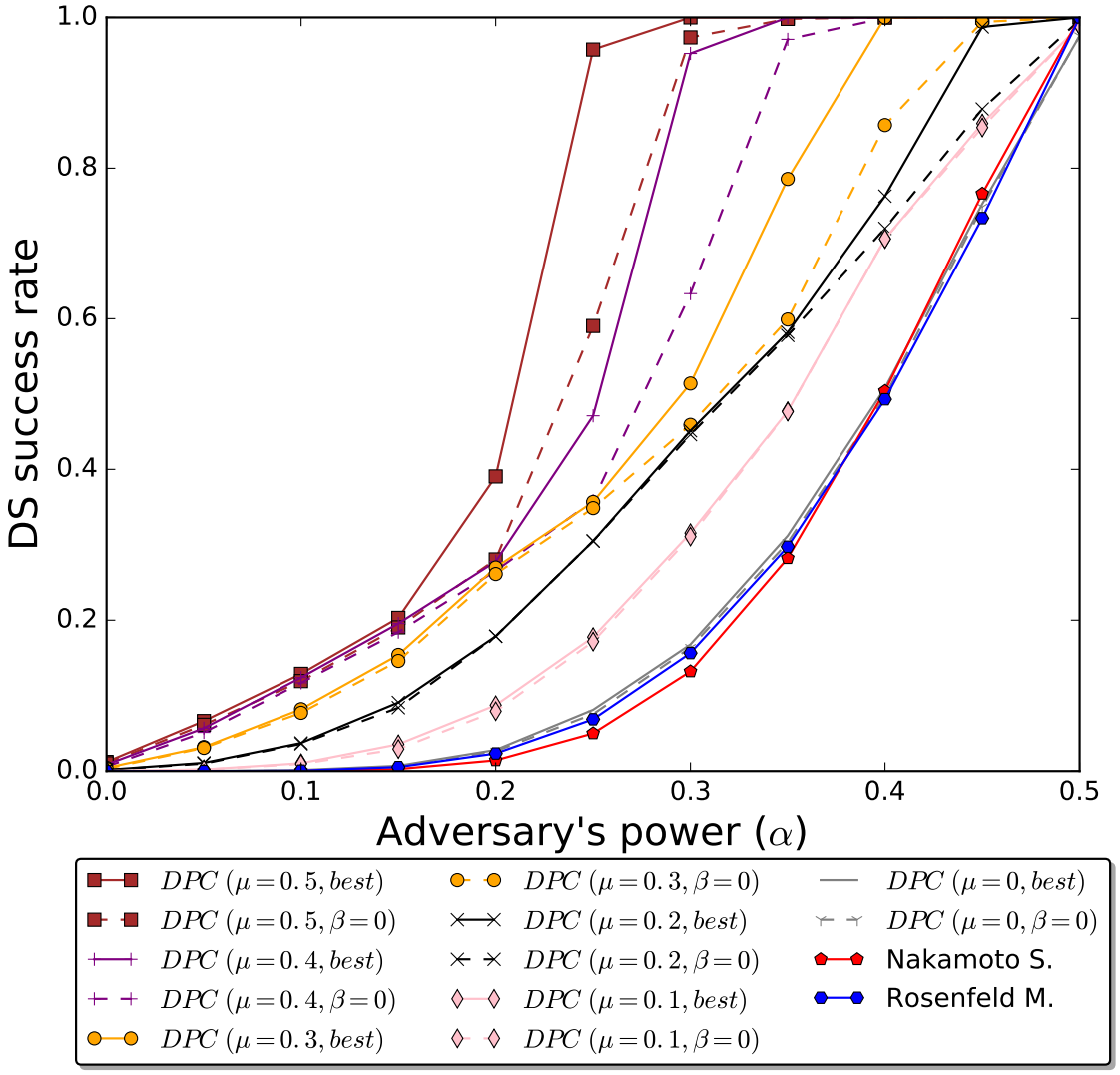


Figure 5.4: DPC attack's success rate for several fractions μ of SPV miners with the best choice of parameter β and for $\beta = 0$ (i.e., a single chain variant of the DPC attack).

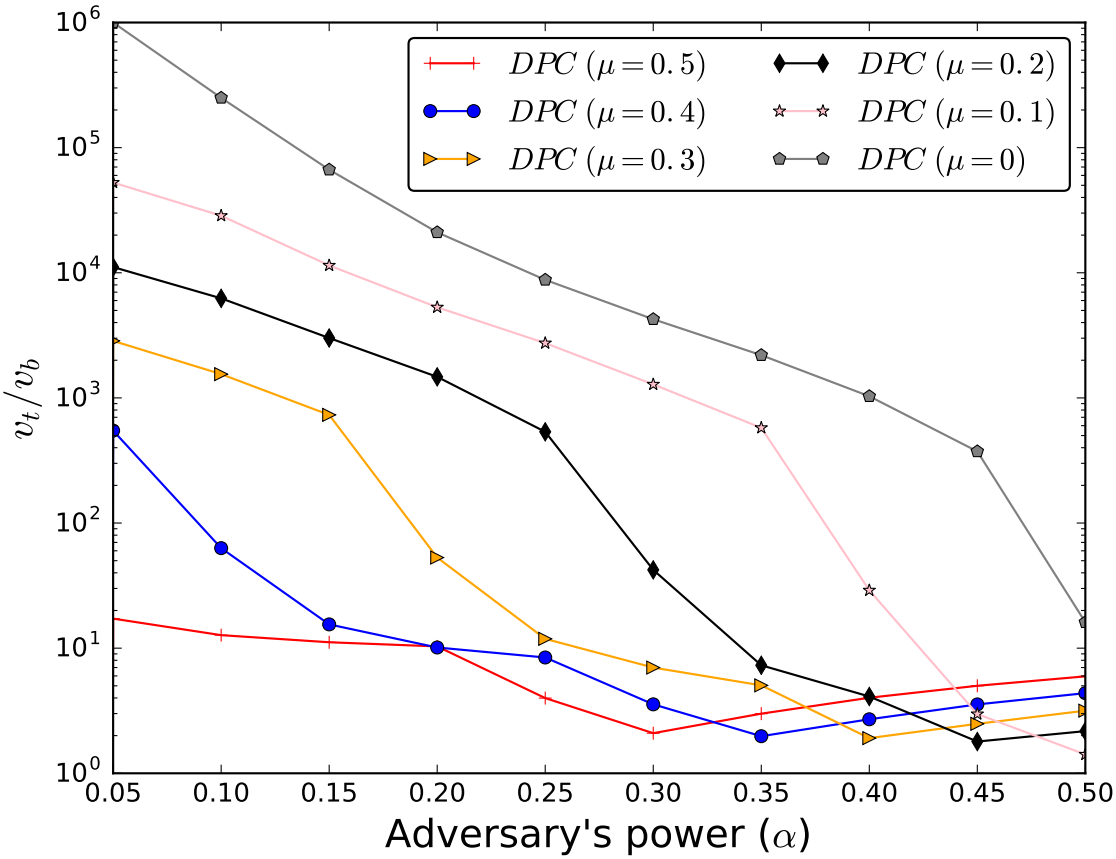


Figure 5.5: Minimum value for $\frac{v_t}{v_b}$ to make the DPC attacks more profitable than honest mining depending on the global power μ of SPV miners.

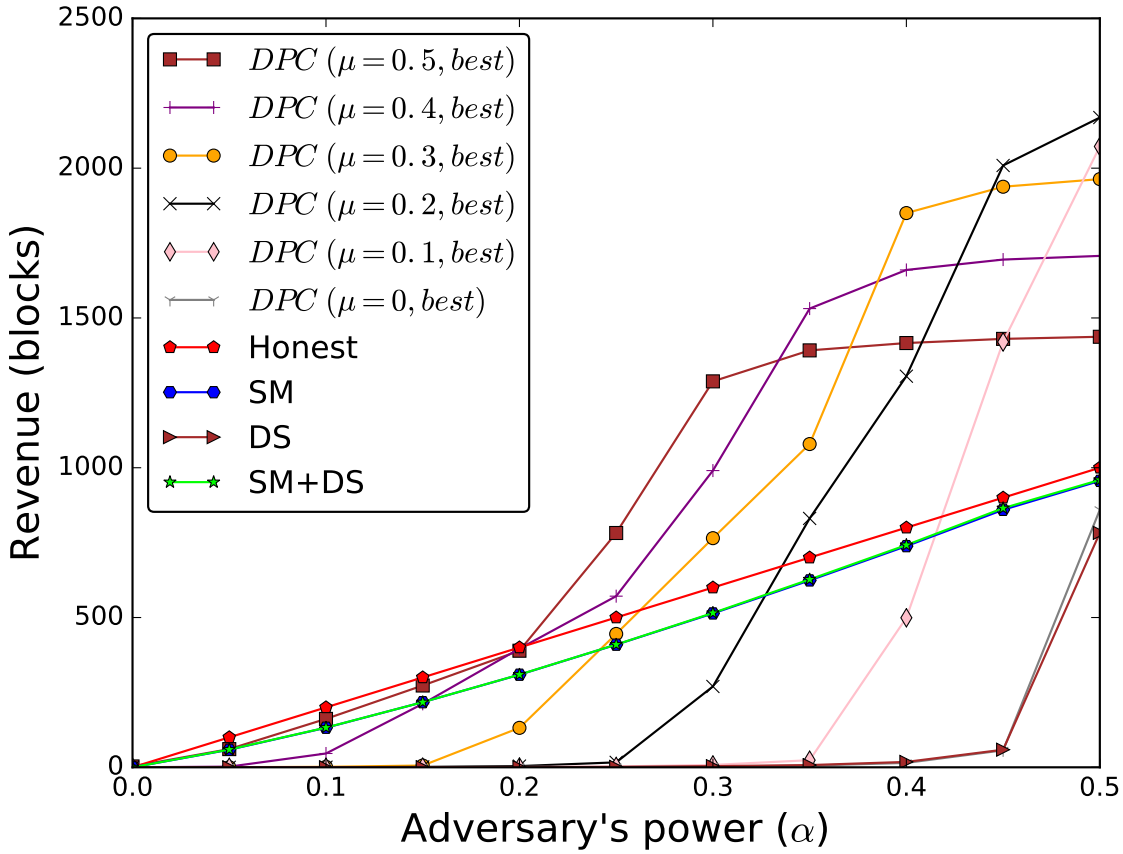


Figure 5.6: Revenue of the adversary when it repetitively attempts to double spend a block with transactions of value $v_t = 10v_b$ with the DPC attack, or with a previous mining strategy or attack, over a period of $2,016$ discovered blocks.

Chapter 6

Conclusion

In this thesis, we provided the empirical and theoretical studies to improve the understanding of PoW cryptocurrencies' privacy and security. In Chapter 3, we analyzed PoW cryptocurrency's network properties, and discussed their impacts on user's privacy and system's security. We designed tools and conducted experiments to examine the Monero's peer to peer network. Our results indicated that even though Monero is a privacy-preserving cryptocurrency, it is still possible to accurately discover the nodes in the network and their interconnections. Our analysis provided insights about Monero's degree of centralization, and about the privacy and security issues potentially caused by a network topology exposure.

In Chapter 4, we defined some novel metrics to link the network layer and consensus layer, which allowed us to evaluate the impact of network delay on the miners' revenue. We showed that the security bounds can be decreased because of network latencies in both attacks. By leveraging the API of mining pools, we measured that Bitcoin's default settings do not preserve network fairness, which could facilitate attacks. For instance, we evaluated that an attacker with 48.72% of the global hash power can launch a majority attack, respectively 24.02% for the selfish mining attack. If large scale deviations were to occur, we proved that a Nash equilibrium would be achieved among the larger miners (i.e., mining pools). However, this would also lead to an increased unfairness of the network, and in particular for smaller miners.

In Chapter 5, we proposed an abstraction "dual private chain" to upgrade the temporary block withholding attacks and examine Bitcoin's security. We built the Markov Decision Process model to analyze the success rate and revenue of DPC attacks targeting SPV miners. Our results indicated that the DPC attacks were more profitable than the honest mining, selfish mining, and double spending based on a single private chain during a short time period by considering the value of double spent transactions. We reported that the threshold to launch the DPC attacks is from 16% to 17.5% depending on the SPV miners' hash power. We

pointed that this threshold is feasible to be reached in the existing mining network of Bitcoin. Future work will evaluate whether the DPC attack is also applicable to other blockchains, such as Ethereum and its proof-of-work chain [126, 29], or could be extended to target other categories of rational miners, such as hash power jumping miners [127, 128].

Bibliography

[1] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds., 1983.

[2] W. Dai, "B-money-an anonymous, distributed electronic cash system," 1998.

[3] N. Szabo, "Bit gold," *Website/Blog*, 2008.

[4] G. Morgenson, "Secrets of the bailout," http://www.nytimes.com/2011/12/04/business/secrets-of-the-bailout-now-revealed.html.

[5] H. Stewart, "Eurozone bailouts: which countries remain?" https://www.theguardian.com/business/2013/dec/13/eurozone-bailouts-greece-portugal-cyprus-spain.

[6] J. Treanor, "Rbs sale: Fred goodwin, the £45bn bailout and years of losses," https://www.theguardian.com/business/2015/aug/03/rbs-sale-fred-goodwin-bailout-years-of-losses.

[7] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12.

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[9] G. Fanti and P. Viswanath, "Deanonymization in the bitcoin p2p network," in *NIPS*, 2017.

[10] D. Koshy, *An Analysis of Anonymity in Bitcoin Using P2P Network Traffic*. Pennsylvania State University, 2013.

[11] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *SP*, 2017.

[12] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network." in *USENIX Security*, 2015.

[13] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network," *IACR Cryptology ePrint Archive*, vol. 2018, p. 236, 2018.

[14] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen, "Partitioning attacks on bitcoin: Colliding space, time, and logic," in *ICDCS 2019*.

[15] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *EuroS&P 2016*.

[16] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *CCS*, 2016.

[17] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *FC*, 2014.

[18] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 515–532.

[19] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.

[20] N. van Saberhagen, "Cryptonote v 2.0," 2013. [Online]. Available: https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf

[21] T. Cao, J. Yu, J. Decouchant, X. Luo, and P. Veríssimo, "Exploring the monero peer-to-peer network," in *FC 2020*.

[22] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *SP*, 2015.

[23] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[24] M. Mirkin, Y. Ji, J. Pang, A. Klages-Mundt, I. Eyal, and A. Juels, "Bdos: Blockchain denial-of-service," in *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, 2020, pp. 601–619.

[25] "Empty blocks," https://medium.com/@ASvanevik/why-all-these-empty-ethereum-blocks-666acbbf002.

[26] "Coinbase transaction," https://en.bitcoin.it/wiki/Coinbase, accessed: 2021-03-31.

[27] "Blockchain.com," https://blockchain.com/, accessed: 2020-09-24.

[28] "Btc.com," https://btc.com/, accessed: 2020-09-24.

[29] "f2pool is doing spv mining," https://bitcointalk.org/index.php?topic=700411.msg11790734#msg11790734.

[30] "Block header," https://en.bitcoin.it/wiki/Block_hashing_algorithm, accessed: 2021-03-31.

[31] J. Parra-Moyano, G. Reich, and K. Schmedders, "A note on the non-proportionality of winning probabilities in bitcoin," *SSRN*, 2020.

[32] E. F. S. . W. R. Sype, "On the negative hypergeometric distribution," *International Journal of Mathematical Education in Science and Technology*, 1987.

[33] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *CoRR*, 2011.

[34] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P*, 2013.

[35] A. Miller and J. J. LaViola Jr, "Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin," *University of Central Florida Tech. Report*, 2014.

[36] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Block arrivals in the bitcoin blockchain," *CoRR*, 2018.

[37] J. L. Hodges and L. Le Cam, "The poisson approximation to the poisson binomial distribution," *The Annals of Mathematical Statistics*, 1960.

[38] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[39] N. Van Saberhagen, "Cryptonote v 2.0," 2013.

[40] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS*, ser. Lecture Notes in Computer Science, 2002.

[41] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring ethereum network peers," in *ACM IMC*, 2018.

[42] G. C. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath, "Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees," *POMACS*, 2018.

[43] A. Miller, J. Litton, A. Pachulski, N. S. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," in *eprint*, 2015.

[44] M. Grundmann, T. Neudecker, and H. Hartenstein, "Exploiting transaction accumulation and double spends for topology inference in bitcoin," in *FC 2018 International Workshops*.

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

[46] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *FC 2016*.

[47] "Stratum mining protocol," en.bitcoinwiki.org/wiki/Stratum_mining_protocol, accessed: 2020-09-24.

[48] T. Neudecker, P. Andelfinger, and H. Hartenstein, "Timing analysis for inferring the topology of the bitcoin peer-to-peer network," in *IEEE UIC*, 2016.

[49] "Bitcoin monitor," https://dsn.tm.kit.edu/bitcoin/index.html, accessed: 2020-09-24.

[50] M. Möser and R. Böhme, "Trends, tips, tolls: A longitudinal study of bitcoin transaction fees," in *International Conference on Financial Cryptography and Data Security*, 2015.

[51] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015*.

[52] L. Bahack, "Theoretical bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013.

[53] E. Group, "Ethereum white paper: Modified ghost implementation," *Ethereum wiki*.

[54] E. Heilman, "One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner," in *International Conference on Financial Cryptography and Data Security*, 2014.

[55] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*, 2015.

[56] P. R. Rizun, "Subchains: A technique to scale bitcoin and improve the user experience," *Ledger*, 2016.

[57] S. D. Lerner, "Decor+ hop: A scalable blockchain protocol," *Semantic Scholar*, 2015.

[58] R. Zhang and B. Preneel, "Publish or perish: A backward-compatible defense against selfish mining in bitcoin," in *Cryptographers' Track at the RSA Conference*. Springer, 2017, pp. 277–292.

[59] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: Serialization of proof-of-work events: confirming transactions via recursive elections," *Cryptology ePrint Archive, IACR*, no. 1159, 2016.

[60] I. Bentov, P. Hubácek, T. Moran, and A. Nadler, "Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 300, 2017.

[61] Y. Sompolinsky and A. Zohar, "Phantom," *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.

[62] S. D. Lerner, "Rsk," 2015.

[63] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th {usenix} security symposium ({usenix} security 16)*, 2016, pp. 279–296.

[64] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.

[65] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2017, pp. 315–324.

[66] G. Bissias and B. N. Levine, "Bobtail: A proof-of-work target that minimizes blockchain mining variance (draft)," *arXiv preprint arXiv:1709.08750*, 2017.

[67] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb: A proof-of-work parallel-chain architecture for massive throughput," *Chainweb Whitepaper*, vol. 19, 2018.

[68] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.

[69] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 175–192.

[70] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 643–673.

[71] E. Shi, "Foundations of distributed consensus and blockchains," *Book manuscript*, 2020.

[72] P. Gaži, A. Kiayias, and A. Russell, "Tight consistency bounds for bitcoin," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 819–838.

[73] J. Siim, "Proof-of-stake," in *Research Seminar in Cryptography*, 2017.

[74] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 297–315.

[75] M. Neuder, D. J. Moroz, R. Rao, and D. C. Parkes, "Low-cost attacks on ethereum 2.0 by sub-1/3 stakeholders," *arXiv preprint arXiv:2102.02247*, 2021.

[76] C. Schwarz-Schilling, J. Neu, B. Monnot, A. Asgaonkar, E. N. Tas, and D. Tse, "Three attacks on proof-of-stake ethereum," *arXiv preprint arXiv:2110.10086*, 2021.

[77] "Pool detective," https://dci.mit.edu/research/tag/pool+detective, accessed: 2020-09-24.

[78] P. Ekparinya, V. Gramoli, and G. Jourjon, "Impact of man-in-the-middle attacks on ethereum," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2018, pp. 11–20.

[79] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin p2p network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 15–29.

[80] M. Corallo, "Bip 152: Compact block relay," https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki, 2016.

[81] I. Eyal, "The miner's dilemma," in *SP*, 2015.

[82] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-theoretic analysis of ddos attacks against bitcoin mining pools," in *FC 2014 Workshops*.

[83] A. Laszka, B. Johnson, and J. Grossklags, "When bitcoin mining pools run dry - a game-theoretic analysis of the long-term impact of attacks between mining pools," in *FC 2015 International Workshops*.

[84] Z. Avarikioti, L. Heimbach, Y. Wang, and R. Wattenhofer, "Ride the lightning: The game theory of payment channels," in *FC 2020*.

[85] "ahashpool," https://www.ahashpool.com/.

[86] "coindance," https://cash.coin.dance/blocks/profitability.

[87] "Spvmining," https://bitcoin.stackexchange.com/questions/38437.

[88] T. Cao, J. Decouchant, J. Yu, and P. Esteves-Veríssimo, "Characterizing the impact of network delay on bitcoin mining," in *International Symposium on Reliable Distributed Systems*, 2021.

[89] J. Göbel, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, 2016.

[90] K. A. Negy, P. R. Rizun, and E. G. Sirer, "Selfish mining re-examined," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 61–78.

[91] Y. Sompolinsky and A. Zohar, "Bitcoin's security model revisited," *arXiv preprint arXiv:1605.09193*, 2016.

[92] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[93] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services," in *Proceedings of the twentieth ACM symposium on Operating systems principles*, 2005, pp. 45–58.

[94] J. Decouchant, "Collusions and privacy in rational-resilient gossip," 2015.

[95] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh, "Erlay: Efficient transaction relay for bitcoin," in *CCS 2019*, 2019, pp. 817–831.

[96] T. Ylonen, C. Lonvick *et al.*, "The secure shell (ssh) protocol architecture," 2006.

[97] E. Rescorla, N. Modadugu *et al.*, "Datagram transport layer security," 2006.

[98] N. Ferguson and B. Schneier, "A cryptographic evaluation of ipsec," 1999.

[99] R. Recabarren and B. Carbunar, "Hardening stratum, the bitcoin pool mining protocol," *arXiv preprint arXiv:1703.06545*, 2017.

[100] T. Neudecker, "Security and anonymity aspects of the network layer of permissionless blockchains," Ph.D. dissertation, 2019.

[101] A. Biryukov and S. Tikhomirov, "Deanonymization and linkability of cryptocurrency transactions based on network analysis," in *EuroS&P*, 2019.

[102] C. Natoli, J. Yu, V. Gramoli, and P. J. E. Veríssimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure," *CoRR*, 2019.

[103] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, and N. Christin, "An empirical analysis of traceability in the monero blockchain," *PoPETs*, vol. 2018, no. 3, pp. 143–163, 2018.

[104] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of monero's blockchain," in *ESORICS*, 2017, pp. 153–173.

[105] Z. Yu, M. H. Au, J. Yu, R. Yang, Q. Xu, , and W. F. Lau, "New empirical traceability analysis of cryptonote-style blockchains," in *FC*, 2019.

[106] J. Yu, M. H. A. Au, and P. Veríssimo, "Re-thinking untraceability in the cryptonote-style blockchain," in *IEEE Computer Security Foundations Symposium (CSF)*, 2019.

[107] D. A. Wijaya, J. Liu, R. Steinfeld, D. Liu, and J. Yu, "On the unforkability of monero," in *ACM Asia Conference on Information, Computer and Communications Security (ASIACCS)*, 2019.

[108] C. Natoli and V. Gramoli, "The balance attack or why forkable blockchains are ill-suited for consortium," in *DSN*, 2017.

[109] P. Ekparinya, V. Gramoli, and G. Jourjon, "Impact of man-in-the-middle attacks on ethereum," in *SRDS*, 2018.

[110] "Monerohash - monero mining pool," https://monerohash.com/nodes-distribution.html, accessed: 2019-01-12.

[111] "Bitnodes," https://bitnodes.earn.com/nodes/, accessed: 2019-01-12.

[112] "Ethernodes," https://www.ethernodes.org/network/1, accessed: 2019-01-12.

[113] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against eclipse attacks on overlay networks," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. EW 11. ACM, 2004.

[114] R. Dorfman, "A formula for the gini coefficient," *The review of economics and statistics*, 1979.

[115] "Bfgminer," http://bfgminer.org/, accessed: 2020-09-24.

[116] "Mining cartel attack," https://bitcointalk.org/index.php?topic=2227, 2010.

[117] T. Moscibroda, S. Schmid, and R. Wattenhofer, "Topological implications of selfish neighbor selection in unstructured peer-to-peer networks," *Algorithmica*, 2011.

[118] M. E. Newman, "Assortative mixing in networks," *Physical review letters*, 2002.

[119] "Bitcoin-wiki," https://en.bitcoin.it/wiki/Block_hashing_algorithm.

[120] J. Bonneau, "Why buy when you can rent? - bribery attacks on bitcoin-style consensus," in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC*, 2016, pp. 19–26.

[121] R. Han, Z. Sui, J. Yu, J. K. Liu, and S. Chen, "Fact and fiction: Challenging the honest majority assumption of permissionless blockchains," in *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*, J. Cao, M. H. Au, Z. Lin, and M. Yung, Eds. ACM, 2021, pp. 817–831.

[122] C. Badertscher, Y. Lu, and V. Zikas, "A rational protocol treatment of 51% attacks," in *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, ser. Lecture Notes in Computer Science, T. Malkin and C. Peikert, Eds., vol. 12827. Springer, 2021, pp. 3–32.

[123] "Half mining power were doing spv mining," https://bitcoin.org/en/alert/2015-07-04-spv-mining#cause.

[124] "spv mining pools," https://en.bitcoin.it/wiki/Comparison_of_mining_pools.

[125] A. Dembo, S. Kannan, E. N. Tas, D. Tse, P. Viswanath, X. Wang, and O. Zeitouni, "Everything is a race and nakamoto always wins," in *CCS*, 2020.

[126] M. Dotan, Y.-A. Pignolet, S. Schmid, S. Tochner, and A. Zohar, "Survey on blockchain networking: Context, state-of-the-art, challenges," *ACM Computing Surveys (CSUR)*, 2021.

[127] Y. Kwon, H. Kim, J. Shin, and Y. Kim, "Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash?" in *IEEE (SP)*, 2019.

[128] A. Spiegelman, I. Keidar, and M. Tennenholtz, "Game of coins," *arXiv preprint arXiv:1805.08979*, 2018.