An Empirical Analysis of Chain Reorganizations and Double-Spend Attacks on Proof-of-Work Cryptocurrencies

by

James Peter Thomas Lovejoy

S.B. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2019

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2020

Signature of Author: _____
Department of Electrical Engineering and Computer Science
May 19, 2020

Certified by: _____
Neha Narula
Director, Digital Currency Initiative
Thesis Supervisor

Accepted by: _____
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

An Empirical Analysis of Chain Reorganizations and Double-Spend Attacks on Proof-of-Work Cryptocurrencies

by

James Peter Thomas Lovejoy

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2020 in Partial Fulfillment of the
Requirement for the Degree of Master of Engineering in
Electrical Engineering and Computer Science

ABSTRACT

Nakamoto consensus has powered Bitcoin and the cryptocurrency industry over the past 10 years, but its security properties when an adversary's economic incentives are taken into account remain poorly understood. Recently, reports of successful real-world attacks against some coins have served as a wake-up call for the industry to review each coins' consensus risk. This research contributes a new system for detecting transaction reordering events against live cryptocurrencies. We deployed the system on a spectrum of different cryptocurrencies and combined our results with historical market data to analyze how the properties of each coin affect its consensus risk and evaluate the effectiveness of existing theoretical models for quantifying the cost of attack. We also describe some of the significant attacks we detected, providing empirical evidence that launching an attack can be practical, and that counterattacking may be a viable strategy for victims to defend themselves from an economically rational adversary.

Thesis Supervisor: Neha Narula
Title: Director, Digital Currency Initiative

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Nakamoto consensus algorithm, Bitcoin's primary contribution to the field of computer science, has remained the backbone of permissionless cryptocurrencies for over 10 years. It is designed to solve the "double-spend" problem, where an adversary spends the same coins more than once, reversing payments that the recipient already considered settled and irreversible. In Nakamoto consensus (often called "proof-of-work"), agents known as "miners" continuously perform a computationally expensive lottery in which they compete to append new transactions to a shared ledger in return for payment in the form of newly minted coins and transaction fees. A miner's probability of winning the lottery first is equal to its proportion of the total computational power being used by all miners. Thus, security from double-spending relies on a key assumption that adversaries are unable to control greater than half of the total computational power. If this assumption does not hold, an adversary could launch a "51% attack", in which they use their majority to rewrite settled transaction history or censor transactions from users or other miners.

For many years this assumption persisted unchallenged until a large number of alternative cryptocurrencies were launched, each being mined with much less computational power than Bitcoin. Combined with the introduction of mining rental markets where miners can outsource their computation to the highest bidder, for many less prominent coins the

assumption that a 51% attack is realistically impossible no longer holds. Challenging this assumption, theorists have since designed models to explain the incentives against double-spend attacks in a world where computational power for mining is available in a liquid market. These models suggest that double-spend attacks should in fact have negligible cost unless additional factors such as post-attack price depreciation are considered.

Since 2018, multiple alternative cryptocurrencies have been successfully 51% attacked in practice, with millions of dollars having been disclosed as double-spent in media reports. Despite the central importance of Nakamoto consensus to the security claims of a majority of the cryptocurrency market, scarce empirical research has been performed to understand the nature of 51% and double-spend attacks in practice. Reorganizations or "reorgs", events in which one suffix of transactions is replaced by another in the transaction history, are transient such that one must observe them in the moment. This is because the ledger only commits to a single, linear list of transactions without branches and discards any alternative transaction histories that may have been proposed in the past. It is thus not possible to determine whether a reorg occurred by retroactively analyzing the transaction history. Therefore, until this research, the field's understanding of 51% attacks has been limited to media reports, exchange disclosures and theoretical models.

To solve this problem, we designed a new system called the "Reorg Tracker" that monitors cryptocurrency networks to detect, save and analyze reorg events in real-time. We used the system to monitor 23 different coins over periods of up to 10 months. We analyzed the dataset in aggregate form to present empirical data on the frequency and severity of reorgs between coins. We combined the reorg data with historical price and mining rental market data for each coin to estimate the costs of reorgs. We use the information to provide empirical evidence for the accuracy of the theoretical models of attack cost. We find that the value of mining rewards is a good approximation of the real-world marginal-cost investment required to append to the ledger for many coins. By extension, we find that the 51% attacks we detected were likely break-even or profitable even without considering the double-spent

transactions they included, validating the behavior predicted in theory.

Furthermore, we consider how a coin's market conditions and parameters affect its reorg frequency, and quantitatively how much financial incentive it provides against attack. We find that average reorg frequency for a coin is inversely correlated with both the compensation it issues to miners per discrete update appending to the ledger and the average interval between updates, but that average reorg severity is uncorrelated with these two parameters. We also show that for every coin we studied, the compensation offered to miners per hour is significantly lower than the transaction volume in the same period. For coins vulnerable to attack using rental markets, this indicates that users in aggregate are currently willing to transact far more value than the initial investment required for an adversary to perform a double-spend. Based on the ratio between miner compensation and transaction volume, we present a metric describing the relative efficiency that coins achieve for their security budget which varies drastically between different cryptocurrencies. We identify that just under half of the cryptocurrencies we studied have an abundance of computational power available on the open market to rent for mining, confirming the theory that mining rental markets are a viable method for launching attacks. Rental markets make it possible for an adversary to only have to pay the marginal cost of computational power for the duration of attack preparation, avoiding the need for them to make any investment in physical mining hardware.

We implemented a module that analyzes reorgs for double-spent transactions and our monitoring detected likely double-spend attacks on at least 3 different cryptocurrencies: Bitcoin Gold, Vertcoin and Litecoin Cash, where we were the first to publicly disclose the attacks. One of the events allowed us to confirm for the first time the existence of counterattacks as a viable strategy for defending against a profit-driven adversary that had previously been presented in theory [12]. Another event provides strong evidence that mining rental markets are used in practice to carry out 51% attacks, and the viability of active pool monitoring as an early-warning system for impending attacks. The third event al-

lowed us to identify deficiencies in a variation on Nakamoto consensus that was originally intended to prevent attacks but failed to do so in practice. Finally, we present evidence for miner-executed double-spend and front-running attacks against Ethereum smart contracts.

Although the reorg tracker is primarily used in this paper to learn retrospective lessons about the state of security among Nakamoto consensus cryptocurrencies, it is also useful as a tool for exchanges, merchants, traders and users to help manage their consensus risk. We implemented a set of "reorg policies" that could be automatically executed by the reorg tracker in the event of a severe reorg to protect a victim from further damage, or potentially use their economic status and attempt to influence incumbent miners to defend them. Any security policy typically has three components: prevention, detection and recovery. Our system implements attack detection and presents some strategies for recovery of varying complexity, while empirically evaluating Nakamoto consensus' attack prevention capacity across different coins.

The remainder of this paper is divided into the following chapters: chapter 2 covers the background information required to understand Nakamoto consensus in the context of double-spend attacks, its security argument as described in the original Bitcoin white-paper, the updated models for security based on economic incentives and the mechanics of mining pools and rental markets. Chapter 3 discusses the related work to this paper including the theoretical models of double-spend attacks, techniques that have been presented for preventing or deterring attacks and prior efforts at attack detection on cryptocurrencies. Chapter 4 describes the design of the reorg tracker, the method of coin selection, server configuration, how mining rental market and price data was gathered and how it was used to calculate event variables. Chapter 5 presents the results of our data collection, evaluates theoretical attack models, explains observed reorg variables in terms of coin parameters and quantifies the cost and feasibility of attack between each coin. It also summarizes the double-spend attacks we observed, providing case studies into the events that provide evidence for rental market involvement and counterattacking, and discusses the double-spends we

observed on Ethereum. Chapter 6 explores some possible mitigations that potential victims could implement in order to limit damage or launch a defense in the event of an attack. Finally, chapter 7 examines avenues for future study based on this research and chapter 8 concludes.

Chapter 2

Background

This chapter covers the background knowledge required to understand the work presented in this research. It first explains the mechanism of Nakamoto consensus and the structure of transactions within different designs of cryptocurrency systems. It also explains the original security argument for Nakamoto consensus in the absence of economic incentives and the attacks that could be possible. The second section presents the more recently developed theoretical model for security when considering economic incentives and its implications for safe transaction values. The final section describes how mining pools and hashrate rental markets work and how they can be exploited by an adversary to launch a practical attack.

2.1 Nakamoto Consensus

2.1.1 Cryptographic Hash Functions

Cryptographic hash functions are a key component of Nakamoto consensus, used both for committing to the order of transactions contained in the ledger and as the function for generating and validating entries to the mining lottery. A cryptographic hash function H takes an arbitrary length input x to produce a fixed-length output y where the output length d is specified by the function's implementation: $H(x) = y,\ x \in \{0,1\}^n,\ y \in \{0,1\}^d$. The

hash function's output should be unpredictable such that the set of outputs over the input space is uniformly distributed over the output space: $p(y[i] = 1 \mid x) = 0.5 \ \forall i \in [0, d)$. It should be impossible for a computationally bounded attacker to find an inverse function H^{-1} that can find a valid input for a given output under H: $\nexists\ H^{-1}(y) = x$. Furthermore, it should be computationally infeasible to find two distinct inputs x and x' such that their outputs under H are equal: $H(x) = H(x') \implies x = x'$.

2.1.2 Transactions and Blocks

In the abstract, a cryptocurrency maintains a shared ledger between networked nodes mapping quantities of unspent coins to the credentials required to spend them. There are two dominant models for representing this information: account-based and output-based. Account-based cryptocurrencies represent the current state of the system as a list of accounts (identified by the credentials required to spend the coins, usually a public key or custom contract) mapped to the account's current balance. Output-based systems maintain a list of unspent "outputs" containing the number of coins an output is worth and the credentials required to spend it. Ethereum-derived cryptocurrencies use the account-based model and Bitcoin-derived systems are output-based.

In both models, transactions are used to change the state of the ledger and move funds from one account to another, or spend a set of outputs to create a new set. In an account-based system, a transaction includes the origin account, destination account, the number of coins to be transferred, a sequence number and the credentials required by the origin account to spend from it. The sequence number is required to protect against transaction replay attacks where an attacker re-uses credentials from a previous transaction to issue a new transaction without authorization. An account-based system will check that each transaction referencing a particular origin account has a monotonically increasing sequence number over the previous transaction for that account. Since any credentials provided for a transaction (usually a signature) are predicated on a specific sequence number and account

pair, increasing the sequence number would invalidate the transaction, and transactions with a previously used sequence number are considered invalid.

In an output-based system, a transaction contains a list of currently unspent outputs to be spent (called "inputs"), a list of new outputs to be created and the credentials required for spending the unspent outputs referenced in the inputs. Since outputs are uniquely identified and removed from the system state when they are spent, additional protection from replay attacks is not required. Attempting to re-spend an output using the credentials from the transaction in which it was spent originally, simply results in the output not being found in the current ledger state, making the transaction invalid.

Both mechanisms make it impossible for an adversary to spend the same coins more than once, assuming the ledger state remains consistent after the transaction has been considered settled by the recipient. In a double-spend attack, the adversary changes the ledger state after a transaction has been settled to reinstate the output or sequence number they already spent and issue a new transaction spending it to a different recipient or back to themselves, invalidating the original transaction. Two transactions are mutually exclusive if they spend the same output in the output-based model or spend from an account using the same sequence number in the account-based model.

Unprocessed transactions are grouped into "blocks" in order to update the state of the ledger. A block represents an atomic unit of operations on the ledger such that if a block is valid, the effect of every transaction contained within it will be applied to the ledger. An invalid block (containing one or more invalid transactions) will result in none of the transactions in the block being applied to the ledger until they are included in a different, valid block. Furthermore, nodes that receive invalid blocks or transactions will not forward them onto other participants of the network.

Blocks are hashed with a cryptographic hash function to generate a unique block ID. Blocks refer to their parent block via the parent's ID, such that the block hash of the parent is included in the pre-image of the child block's hash. This serves as a cryptographic

commitment to the sequence of blocks that have led to the most recent block. Due to the properties of a cryptographic hash function, it is not possible to change the contents of a block without also changing its hash. Since the parent block hash is included in a child block, changing any prior block in the sequence changes the hash of subsequent blocks [1]. Therefore, attempts to tamper with the established sequence of blocks can be easily detected by ensuring every block from the tail to head of the sequence refers to the correct parent block hash.

2.1.3 Mining

Generating many distinct hashed chains of blocks containing valid transactions is not computationally difficult. Proof-of-Work consensus thus uses a process called "mining" to guard against Sybil attacks and allow network participants to decide which chain represents the state of the ledger that most other network participants will use with high probability. Mining exploits the property of cryptographic hash functions (and thus block hashes) that the output of the function for a given pre-image is unpredictable and uniformly distributed.

As a result, the probability of the hash of a block having a specific value y is $Pr(H(block) = y) = \frac{1}{2^d}$, where d is the fixed length of the hash function's output in bits. By extension, the probability that the hash of a block is below a certain value is $Pr(H(block) < y) = \frac{y}{2^d}$. Therefore, the expected number of distinct blocks that would need to be hashed to find a block with a hash below y is $\frac{2^d}{y}$. By tuning d and y, the expected number of hash attempts to find a block hash below y can be made large such that it *is* computationally difficult to generate multiple distinct chains of blocks. All that is required is to add the constraint that every block's hash must be below some network-agreed parameter y, referred to as the "block target".

Presenting a block with $H(block) < y$ to the network serves as a "proof-of-work" that with high probability the entity that generated the block performed the expected number of hash attempts. Blocks contain a numerical field called the "nonce" that miners iterate

over to generate different values for $H(block)$ without otherwise having to modify the block's contents. As a reward for conducting the necessary work to generate a block with a valid nonce, a successful miner is compensated with newly created coins and any transaction fees paid by the transactions included in the block.

Network participants still need a way to select between multiple candidate sequences of blocks, all of which could contain valid transactions and proof-of-work. The tie-breaking is done by comparing the "total work" of the competing chains and selecting the chain with the most total work. Intuitively, total work can be thought of as the expected number of hash attempts required to generate a sequence of blocks. Total work can therefore be calculated using the summation $\sum_{n=1}^{N} \frac{2^d}{y_n}$, where N is the number of blocks in the chain and y_n is the target for a given block n.

It is possible that multiple miners will find blocks, perhaps containing different sets of transactions, whose hashes satisfy the block target at similar times, resulting in multiple candidate chains with the same total work but different ledger states. Due to network propagation delay, nodes will receive each of these blocks in different orders. In this case, nodes simply select whichever valid block they received first. Once a subsequent block is mined creating a chain with more total work, all nodes will switch to that chain. For the nodes that received a block first that is now not part of the greater total work chain, this triggers a chain "reorganization" or "reorg". During this process, the block the node had previously selected is discarded and becomes an "orphan" and is replaced by the blocks in the greater total work chain. Multiple chains with the same total work can persist to generate forks that are more than one block deep if miners repeatedly find competing blocks around the same time. However, over time it is expected the network will converge to the same sequence of blocks if miners follow the protocol and build upon the most work chain, due to the relatively low probability of miners repeatedly finding blocks at similar times.

Originally, Bitcoin was mined using consumer-grade processors meaning that anyone with a computer could meaningfully participate in the mining process, contribute blocks

to the chain and receive mining rewards. As miners attempted to create a competitive advantage for themselves, they began to use more specialized Graphic Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs) to perform hashes faster and with less energy consumption per hash, making CPUs uncompetitive. Today, Bitcoin mining is a professionalized industry where miners use specially designed hardware that can only mine SHA256 (Bitcoin's hash function) called Application Specific Integrated Circuits (ASICs), operated in large data-centers located close to sources of cheap wholesale electricity. ASIC performance is continuously improving and Bitcoin ASICs are not commodity hardware making it difficult for consumers to obtain competitive devices before they are made obsolete by a new generation of device.

A number of alternative cryptocurrencies have chosen a hash function for mining that aims to be "ASIC-resistant", or implement governance policies that regularly change the coin's mining hash function to thwart ASIC development. These coins can still be mined profitably using commodity hardware and residential electricity prices. Miners using general purpose hardware can easily switch between different hash functions, allowing them to optimize which coin they are mining at any given time to maximize their total profit. ASIC miners can also switch between different coins, but only those that share the hash function the ASIC was designed for, limiting the scope for optimization. Miner profit optimization causes a large disparity in the volume of mining resources being deployed on different coins. This means that in many cases, the total mining resources being deployed across all coins in a given mining class (GPU/FPGA-mined, ASIC-mined) are much greater than the resources deployed on a particular coin.

2.1.4 Security Argument

The classical security argument for Proof-of-Work consensus is that the majority of the computational power dedicated to mining will choose to behave honestly and follow the protocol. This means that honest miners always mine referencing the greatest total work valid block

sequence they have seen and broadcast any new blocks they mine to the network without delay. Consider a minority attacker mining in secret attempting to generate a competing fork and cause a chain reorganization, or "reorg", where one suffix of blocks is replaced by a different suffix containing more total work, potentially changing the set of transactions and their ordering in the chain. As long as the majority of mining power behaves as the protocol prescribes, the security argument says that the adversary will have exponentially diminishing probability of success the more blocks the honest majority generates. Assuming miners are rate-limited in terms of the number of hash attempts they can perform per unit time, the expected time for a miner to find a block can be modeled as a Poisson distribution. In that setting, Nakamoto provides the following equation in the Bitcoin paper to describe the probability that a minority attacker will succeed in causing a reorg.

$$P_z = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} (1 - (\frac{1-p}{p})^{z-k}) \tag{2.1}$$

If there are a hashes per unit time being performed by the honest majority of miners and b across the whole network, then the honest miners have a probability $p = \frac{a}{b}$ of being the first to find a block with a hash below the target. Therefore $\lambda = z\frac{1-p}{p}$ is the expected number of times the attacking miners find a block first out of z blocks generated by the honest miners. If $p > 0.5$, P_z tends to zero as z (the number of blocks mined by the honest miners), and p increases. This shows that if there is an honest majority of miners, the probability that an attacker will successfully cause a reorg drastically diminishes as the chain progresses, meaning that Proof-of-Work consensus provides eventual consistency. Since different nodes can thus have inconsistent views of the ledger state even without an attack taking place, recipients must wait for transactions that credit them to be z blocks deep in the chain, chosen so that P_z is sufficiently small to satisfy the recipient that the transaction will not be reverted due to a reorg. This waiting time is often referred to as the "confirmation time" or "escrow period" and is subjective, allowing recipients to choose different values for z depending on their risk profile and belief about the true value of p, trading off with the

amount of time both transacting parties have to wait for settlement.

2.1.5 Mining Attacks

If $p < 0.5$ in equation 2.1 then P_z tends to one as z increases, meaning that the attacker will eventually always succeed in causing a reorg. This leads to a class of mining attacks known colloquially as "51% attacks". With greater than 50% of the network hashrate, an attacker miner is able to generate alternative chains in secret and later release them to the network after nodes have already considered a sequence of blocks to be final with high probability. The attacker can then perform a "double-spend" attack by including a conflicting transaction in their secret fork, that would invalidate a transaction in the existing honest chain after a reorg [9].

More concretely, the attacker sends coins to their counterparty via a transaction included in the honest chain. Once the transaction is included in an honest block, the attacker begins building an alternative chain in secret, referencing a block before the one in which the transaction was included. The attacker omits the original transaction and includes a mutually exclusive transaction in their secret chain sending the coins intended for the counterparty elsewhere. Once the attacker's chain has more total work than the honest chain, and enough blocks have elapsed on the honest chain since the transaction was originally included for the counterparty to accept the transaction as final, the attacker reveals their fork to the network and causes a reorg.

Due to the reorg, the transaction sending coins from the attacker to the counterparty is reversed and invalidated by the mutually exclusive transaction sending the same coins back to the attacker. Since the counterparty believed the original transaction was final with high probability having been sufficiently deep in the honest chain, they would have credited the attacker's transaction and provided whichever good or service the funds were in exchange for. Now that the original transaction has been reversed and invalidated, unless the attacker sends an additional payment, the attacker keeps both their payment and the

goods or services rendered.

Other attacks are possible if a miner controls greater than 50% of the network hashrate besides double-spending. An adversary could perform a denial of service attack where they refuse to include transactions from other users and not reference blocks generated by other miners, preventing honest miners from updating the ledger. This would generate continual reorgs for nodes that have a lower network latency to the honest miners than the adversary, as the honest miners' blocks are always subsequently replaced by attacker blocks. A majority miner could also be more subjective in their denial of service, choosing to replace blocks from other miners that contain high transaction fees to include the transactions in their own blocks and claim the fees. Similarly, an adversary could enforce a particular transaction fee above which they are willing to include transactions in their blocks, holding users to ransom. Although nodes will typically not accept blocks that contain timestamps referencing a time in the future, a miner with majority hashrate could manipulate block timestamps to alter the difficulty target to their benefit, allowing them to produce more blocks with less work, increasing their mining profitability.

2.2 Economic Security of Proof-of-Work

This section explains the current theory in the literature describing how an economically rational miner should behave in a situation where the mining market for a coin is at equilibrium. It relaxes the assumption that a majority of the network's computational power is unobtainable and instead considers an attacker who is able to utilize enough mining resources to launch a successful attack at marginal cost per hash. The model's output is the safe threshold for the payoff value from a successful attack, above which it would be more rational for an adversary to attack than to mine honestly.

2.2.1 Mining at Market Equilibrium

At equilibrium the marginal revenue will equal the marginal cost of mining a block ($MC = MR$) [2]. This leads to the following equation:

$$Nc = V_{block} \tag{2.2}$$

Where V_{block} is the reward a miner receives for successfully mining a block which for Bitcoin is the block subsidy (6.25 BTC) plus any additional transaction fees. N is the expected number of hashes required to be performed in order to find a block hash that satisfies the network block target and c is the marginal cost of performing a single hash. Plugging real numbers from Bitcoin into equation 2.2, we can estimate c. As of writing the average reward per block is $Nc = V_{block} = 6.25 + 1.37 = 7.62$ BTC and $N = 6.92 * 10^{22}$ [1]. Therefore $c = \frac{V_{block}}{N} = \frac{7.62}{6.92*10^{22}} = 1.10 * 10^{-22}$ BTC or $1.06 * 10^{-18}$ USD per hash.

At equilibrium due to equation 2.2:

$$E(honest\ mining) = V_{block} - Nc = 0 \tag{2.3}$$

An attacker who performs a double spend has the following expected payoff [3]:

$$E(attack\ mining) = V_{attack} + E(honest\ mining) \tag{2.4}$$

Where V_{attack} is the payoff an attacker miner receives for successfully performing their attack. For example, this value could be the cost of goods or services that have been delivered, the earnings from a short position taken out against the asset's price (betting on a negative market reaction to the attack) or the profit from reordering transactions to front-run a smart contract. For a Proof-of-Work system to be secure against a rational attacker who aims to make a profit the following inequality is required to hold:

[1] Data retrieved 05/18/20 from `https://bitinfocharts.com/bitcoin/`

$$E(attack\ mining) < E(honest\ mining) \tag{2.5}$$

Which leads to the following impossibility result for Proof-of-Work security in a hashrate market at equilibrium:

$$V_{attack} < 0 \tag{2.6}$$

Equation 2.6 states that at market equilibrium there is no transaction value that is safe from double-spending by an economically rational adversary who can obtain more than half of the network hashrate. This is because the expected reward from honest mining should equal that of attacking without requiring any double-spends to be included. The result means that the mining market for a coin must not be in equilibrium for transactions to be secure from double-spending, or that other factors must reduce the value of attack mining relative to honest mining [3].

2.2.2 Post-Attack Price Changes

The model can be expanded to consider the asset's price changing as a result of an attack. A decrease in price could be expected if the attack was successful and negatively affected the market's faith in Nakamoto consensus, or if an important economic actor was the target, leading to uncertainly regarding their solvency. By including a change in price post-attack, the attacker's expected payoff can be redefined as the following:

$$E(attack\ mining) = P_{attack}(V_{attack} + V_{block}) - Nc \tag{2.7}$$

Where P_{attack} is the relative purchasing power of an asset after a successful attack. In the case of no change in price $P_{attack} = 1$, if the price drops by 50% $P_{attack} = 0.5$. It is important that the price falls due to the attack rather than coincidentally for a different reason. This is because the adversary must expect that the price will decrease as a direct

result of their choice to engage in an attack rather than honest mining for them to be deterred from attacking in the first place. This leads to a new security condition:

$$V_{attack} < \frac{V_{block}(1 - P_{attack})}{P_{attack}} \tag{2.8}$$

Table 2.1 shows the maximum safe transaction value from a rational attacker as a factor of V_{block} for different values of P_{attack}. Evidently, a large price decrease is required to occur post-attack for high-value transactions to be secure, with a 50% price drop needed just for V_{attack} to equal V_{block}. If the price increases post-attack, no transaction value is secure because attacking would be more profitable than honest mining even without an additional payoff from V_{attack}. The transaction volume on Bitcoin is far greater than the maximum safe V_{attack} for $P_{attack} = 0.01$, suggesting that the risk of a price decrease cannot be the only additional factor providing security from attack.

Table 2.1: Maximum safe BTC transaction sizes for different values of P_{attack} under equation 2.8

P_{attack}	Maximum safe V_{attack}	BTC value
1.0	0	0
0.99	$0.01V_{block}$	0.13
0.9	$0.11V_{block}$	1.39
0.5	V_{block}	12.68
0.1	$9V_{block}$	114.12
0.01	$99V_{block}$	1255.32
0.0	∞	∞

2.2.3 Potential for Attack Failure

In the event of an attack, users could collaborate and choose not to switch from the existing honest chain to the attacker's new chain, rejecting the fork even though it has more total work, breaking the rules of Nakamoto consensus. This could be achieved by distributing modified node software or by users making remote-procedure calls to their node that manually invalidate the adversary's blocks. We can express the probability that users accept the

attacker's fork and the attack succeeds as p_{reorg} and adjust the expected payoff from attack mining as follows:

$$E(attack\ mining) = p_{reorg}P_{attack}(V_{attack} + V_{block}) - Nc \tag{2.9}$$

Leading to the following security condition:

$$V_{attack} < \frac{V_{block}(1 - p_{reorg}P_{attack})}{p_{reorg}P_{attack}} \tag{2.10}$$

2.2.4 Non-Repurposable Hardware

As previously mentioned in subsection 2.1.3, some cryptocurrency networks (notably Bitcoin and Litecoin) use hash functions for proof-of-work (double-SHA256 and Scrypt, respectively) for which ASICs dominates the total network hashrate. In this case, miners have a large fixed cost V_{fixed} associated with acquiring their mining hardware that must be remunerated over the course of the hardware's lifetime for the miner to receive a return of investment. Since the hardware is not re-purposable, if the prices of assets it mines depreciate, the lifetime value of the hardware will also depreciate proportionally to the drop in asset prices. Miners using general purpose hardware have a much lower or even negligible V_{fixed} as the hardware can be trivially sold or re-purposed, meaning that its value is not significantly influenced by the price of a certain cryptocurrency.

A further parameter $\alpha \in (\,0.5, 1]$ representing the attacker's share of the total mining network can be introduced to account for the case where the attacker miner has less than 100% of the total mining power and thus has a proportionally lower fixed cost.

This modifies the expected attack mining payoff as follows:

$$E(attack\ mining) = p_{reorg}P_{attack}(V_{attack} + V_{block}) - Nc - (1 - P_{attack})\alpha V_{fixed} \tag{2.11}$$

Leading to the security condition:

$$V_{attack} < \frac{V_{block}(1 - p_{reorg}P_{attack})}{p_{reorg}P_{attack}} + \frac{(1 - P_{attack})\alpha V_{fixed}}{p_{reorg}P_{attack}} \tag{2.12}$$

2.3 Mining Pools

2.3.1 Centralized Pools

Due to the large network hashrate of a cryptocurrency network relative to the hashrate of an individual miner, miners typically mine in collectives known as "pools" in order to reduce the time variance between receiving block rewards. The function of a pool is to aggregate the hashrate of constituent miners such that they all mine together as if they were a single entity. The pool divides up the search space for a block with a hash below the block target between its miners and distributes the block rewards once a block is found based on the miners' proportion of work contributed. Figure 2.1 shows a block diagram detailing the internal structure of a centralized mining pool and how it interacts with client miners.

Communication between the pool and its miners is conducted via a specialized protocol known as Stratum [4]. The pool runs a node for the cryptocurrency being mined and generates block templates containing transactions of the pool's choosing. The pool sends a summary of the template called a "block header" to its miners containing the previous block hash, timestamp, block version number, difficulty target, a commitment to the set of transactions in the block and a nonce. The miners iterate over values of the nonce and calculate the hash of the resulting block header to find a block whose hash is below the target. If such a nonce is found for the provided block template, the miner sends the nonce back to the pool which then broadcasts the newly found block to the network.

The pool maintains a database of the proportion of work performed by miners within the pool and uses that data to divide the block rewards it receives between the miners and send the coins to them via a normal transaction. Since it could take years for a small hashrate

miner to find a block, pools require regular heartbeats from miners to prove they are working on the block template the pool sent them and to keep track of relative work between miners without miners having to regularly find blocks. This is achieved by setting the block target value required by the pool much higher than the actual network target such that blocks are easier to find.

Eventually a miner will submit a nonce that generates a block hash that satisfies both the pool target and the network target, allowing the block to be broadcast to the network. Nonces that only satisfy the pool target (known as "shares") can then be used as a Proof-of-Work that miners are attempting to find a nonce that satisfies the network target, and use the frequency of share submission as a measure of the relative hashrate contribution of each miner.

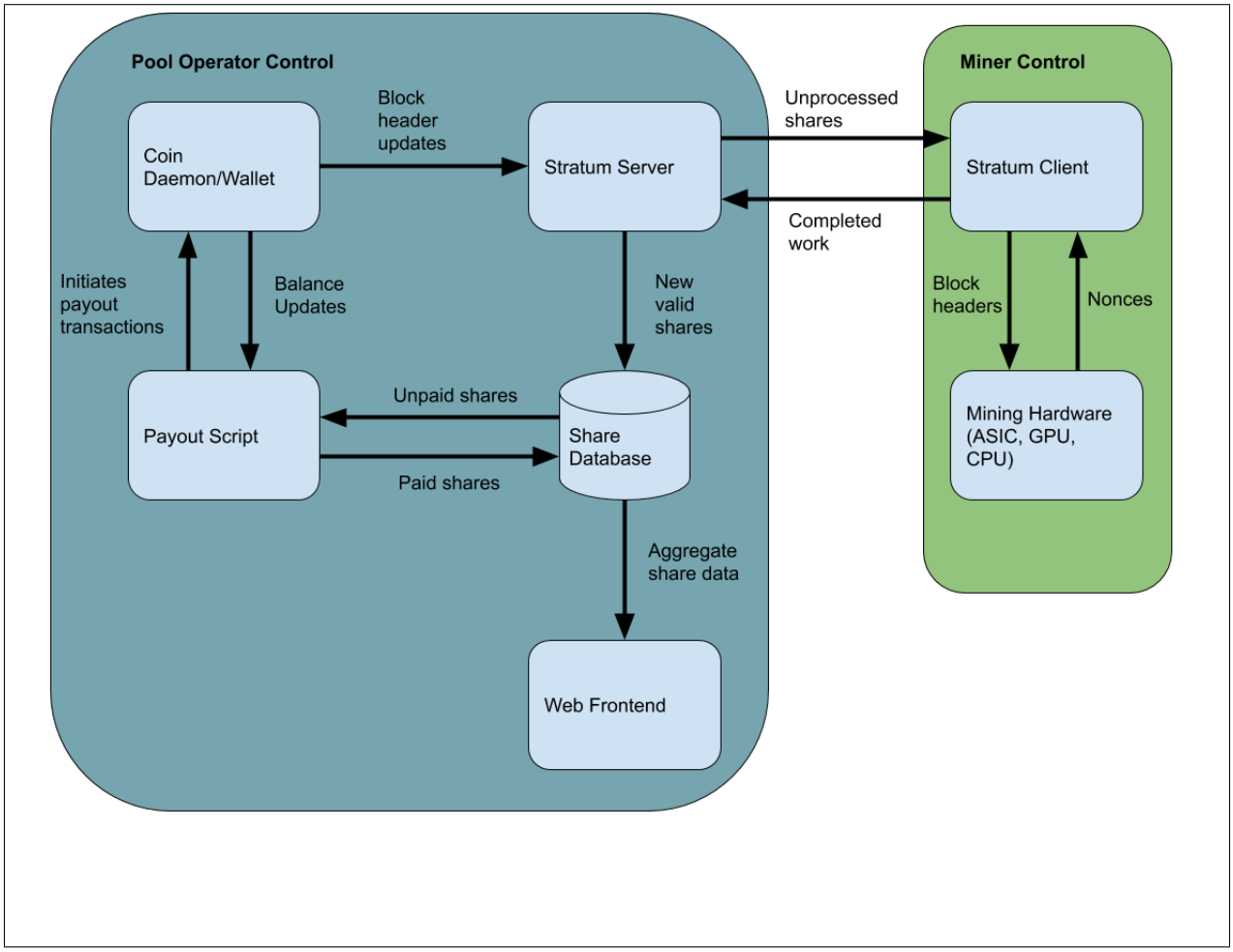

Figure 2.1: Block diagram describing the typical structure of a centralized mining pool

2.3.2 Mining Rental Markets

The large selection of coins available for miners to mine coupled with the strong desire to keep earned income in a more liquid asset (specifically Bitcoin) has led to the creation of rental markets for hashrate. The most popular such market, known as Nicehash [2], allows owners of mining hardware to connect to their platform as if it were a regular mining pool and provides a marketplace for renters to bid and purchase allocations of lessors hashrate. Renters direct purchased hashrate to a Stratum pool of their choosing and pay in Bitcoin, receiving any proceeds of mining directly from the pool the renter selects. In that sense Nicehash operates as a stratum-to-stratum proxy, indirectly connecting physical miners to different pools based on orders from renters.

Miners are paid by Nicehash directly in Bitcoin per valid share submitted. This makes it easier for miners to optimize their profitability by paying them up-front in Bitcoin, negating the need for miners to sell their coins in the market and reason about potential asset price changes before the coins can be sold. Furthermore, miners do not have to decide which coin to mine and instead simply have to pick which hash function is most profitable on Nicehash for their hardware.

Renters bid for hashrate via a live auction in which orders are filled via a greedy algorithm from the most expensive order to the least. Bids are denominated in hashes per second per day and can optionally specify a limit to the hashrate that Nicehash will dedicate to the order. Nicehash markets for each hash function are split between different server regions based on geographical location allowing renters to minimize network latency between the miners and the renter's desired pool. Miners on Nicehash are price-takers and play no role in deciding how much they are paid or to which orders their hashrate is allocated. Additionally, Nicehash and its constituent miners are unaware of which coin (or fork) they are mining without analyzing the work issued by the upstream mining pool or Nicehash's Stratum server respectively. The only data Nicehash collects from renters is the identity and

[2] `https://www.nicehash.com/`

log-in credentials of the upstream pool server an order is to mine on and this information is not relayed to downstream lessors. Figure 2.2 shows a block diagram describing the internal structure of Nicehash, and how it interacts with mining pools, renters and miners. Dashed lines represent the movement of cryptocurrency funds and solid lines show the transport of data.

Nicehash has vastly increased the liquidity of hashrate in the market due to its popularity and ease-of-use for both miners and renters. Many Proof-of-Work coins use the same hash function of a coin with a much larger network hashrate. This has led to market conditions where there is excess hashrate available to rent and perform a 51% attack on coins whose mining algorithm is in the same mining class (generic hardware or algorithm with deployed ASICs) as another coin that is dominant in its class.[3] Since the amount of hashrate required to launch an attack for some coins is small compared to the available supply, the price of hashrate is elastic enough that renting a multiple of a small coin's hashrate can be trivial.

[3] `https://www.crypto51.app/` maintains a list of coins mapped to the estimated cost to perform a 51% attack using Nicehash and the available supply of hashrate relative to the coin's network hashrate.

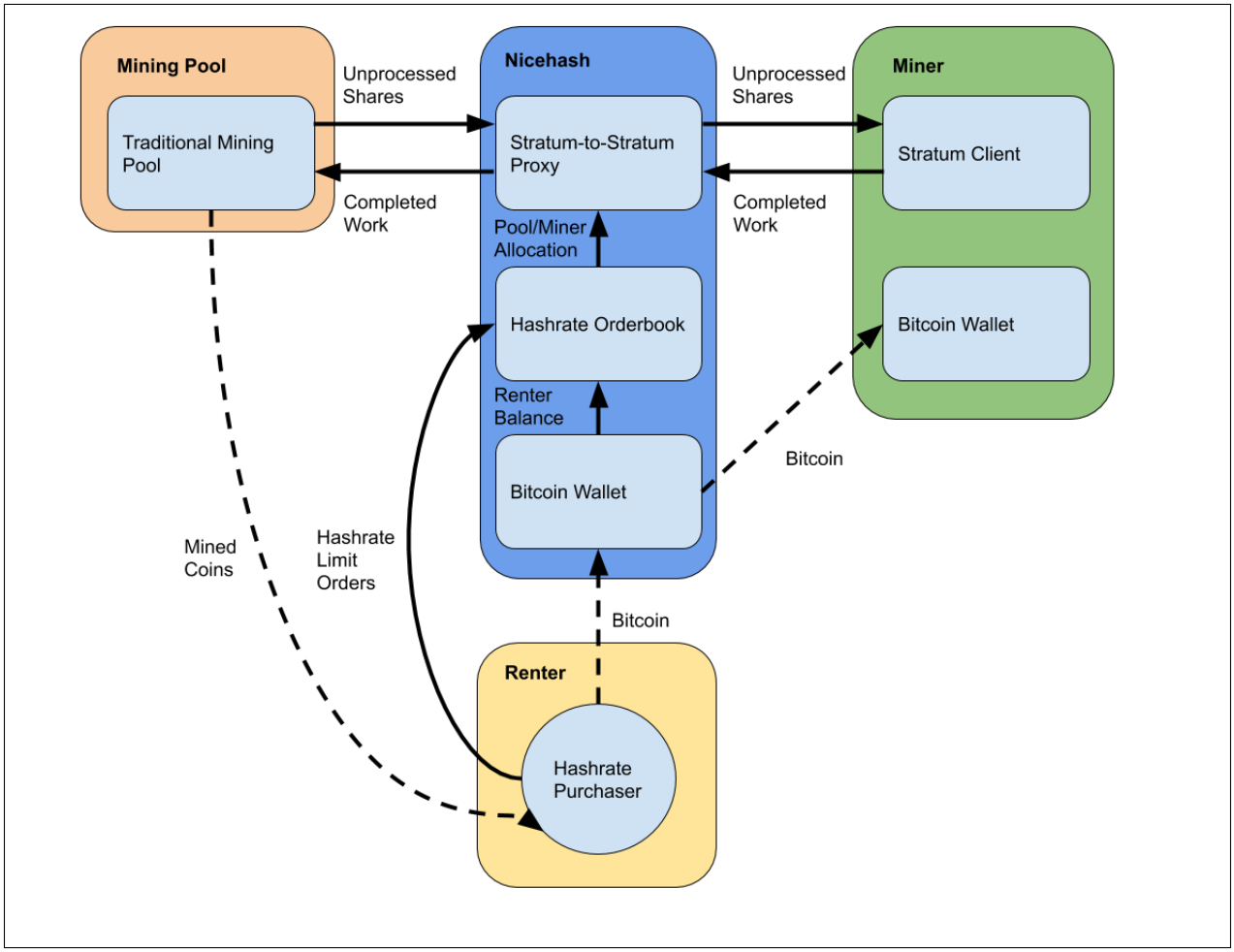

Figure 2.2: Block diagram describing the structure of the Nicehash rental market and how it interacts with renters and lessors

Chapter 3

Related Work

This chapter briefly enumerates the prior literature related to this research, split into four categories. The first section covers works that aim to provide theoretical models for the profitability of performing a double-spend attack in a range of scenarios. The second section cites some of the more interesting proposals (and deployed implementations) from academics and coin developers intended to reduce the profitability or feasibility of attacking. The third section describes efforts by the cryptocurrency industry thus far to detect and study attacks against coins in the wild and to quantify their consensus risk. The final section explains other notable transaction reordering attacks on Nakamoto consensus that do not necessarily involve double-spent transactions.

3.1 Theoretical Models of Double-Spend Attacks

Meni Rosenfeld's paper analyses the economic security of Proof-of-Work in the context of Nakamoto's original security argument that an attacker will be unable to acquire a majority of Bitcoin's hashrate [9]. They provide a model to describe the transaction value and escrow period thresholds above which a miner should be incentivized to attempt a double-spend attack even with less than 50% of the network hashrate at their disposal. Due to the probabilistic nature of block production, a malicious actor with minority hashrate has a small

but non-zero probability of outpacing the rest of the miners. Thus for certain transaction values and escrow periods the expected reward for success outweighs the expected loss from failure.

Eric Budish (and separately Raphael Auer [37]) presents in his paper a model for the economics behind launching a double-spend attack when the mining rental market for a coin is in equilibrium and supply is not limited [2]. In this model, Budish considers an attacker who is able to rent arbitrary amounts of hashrate at the market price to launch a 51% attack, and determines the transaction value threshold above which miners are incentivized to attack. Unfortunately Budish makes a subtle error in his model where he states that mining with more hashrate causes an increase in attack cost. This is not the case in reality as renting more hashrate allows an attacker to reduce the duration for which the hashrate is rented, still producing the same amount of total work.

Hasu et al refine Budish's model, introducing new parameters to model non-repurposable hardware, coin price changes post-attack and the potential for users to manually reject a malicious reorg [3]. They conclude that 51% attacks should be break-even at market equilibrium if miners have no fixed costs associated with their hardware that they need to recoup. For Proof-of-Work consensus to be incentive compatible against double-spends, they argue the asset must depreciate after an attack and miners must be committed to mining that specific asset, meaning depreciation reduces their potential to recoup fixed costs for hardware.

Dan Moroz et al extend Budish's model and present a game theoretic model for a victim of a double-spend attack to counter-attack, leading to a war of attrition in which the equilibrium of the game is that the attacker will not attempt an attack [12]. They suggest a credible counter-attack threat by a victim could be a realistic deterrent to double-spend attacks even when miners have no fixed-costs for hardware or commitment to mining a specific asset.

3.2 Techniques for Preventing Attacks

In a blog post, Vitalik Buterin introduces a concept he calls "weak subjectivity" where nodes weigh blocks that were seen earlier higher than those seen later when selecting the most work chain in the case of a potential reorg [6]. The intention is to punish miners who delay submitting their blocks to the network by requiring them to produce a much higher total work chain than under traditional Nakamoto consensus before nodes will reorg.

The Horizen developers have deployed their own flavor of weak subjectivity on their cryptocurrency Zencash (which has previously been hit with a double-spend attack) as a protective measure against future attacks [7]. The protocol suffers from a potential attack where an attacker generates two hidden forks and reveals them to two distinct subsets of nodes. In this case, nodes are unlikely to converge to a canonical chain over time as convergence would require one set of nodes to perform a deep reorg, and they disagree on which fork was observed first [38]. Furthermore, new nodes that are performing initial block download will be unable to decide which fork was revealed first, and thus can only follow the most total work chain, as in traditional Proof-of-Work. This means that manual developer intervention would likely be required to resolve the chain split that arises in a timely manner.

Litecoin Cash has deployed a hybrid Proof-of-Work/Proof-of-Stake protocol their developers call "Hive Mining" where Proof-of-Stake blocks are interlaced with Proof-of-Work blocks. The intention is that an attacker would need both majority mining power and significant stake of the coin to launch an attack [8]. The assumption is that it would be costly to acquire a significant proportion of the coins and the network hashrate. Furthermore, attacking the system would cause a devaluation of the attacker's stake, disincentivizing an attack. The continued role of Proof-of-Work in the system prevents the need for a coordinator to guard against attacks on Proof-of-Stake such as zero-cost rollbacks if a single staker acquires more than 51% of the stake or stake grinding.

Ittay Eyal and Emin Gün Sirer proposed Two-Phase Proof-of-Work in response to a Bitcoin mining pool accruing greater than 51% of the network hashrate in 2014 [10].

The algorithm modifies Proof-of-Work to require a miner to sign each block they mine using the private key associated with the block rewards payout address. This makes pool mining impractical as the pool would have to share its private key for block rewards with its constituents in order for them to join the pool. This algorithm would effectively enforce solo-mining, and restrict pooled mining to only be between trusted entities. The argument is that 2P-PoW would make it much more difficult to acquire 51% of the network hashrate as un-trusted parties would no longer be able to pool hashrate and behave as a single entity. 2P-PoW also makes hashrate non-outsourceable, meaning it would be impossible to create a proxy pool like Nicehash where hashrate can be rented.

Kevin Liao and Jonathan Katz introduce "Whale Transactions", transactions with large fees for miners that are only valid on one side of a fork. They can be used to incentivize miners to extend the fork that the issuer of the whale transaction desires, abandoning the classical rule of Nakamoto consensus that miners always extend the most-work chain [5]. In the context of double-spend attack defense, whale transactions have a problem. While the victim can issue a high-fee transaction spending the coins that were double-spent, incentivizing the miners to counter-reorg the attacker's fork, the attacker can do the same thing. The attacker can broadcast their own whale transaction spending the proceeds of their double-spend, incentivizing miners to continue mining the malicious fork. This leads to a war of attrition in which the victim and attacker bid for the miners' hashrate until the entire value of the double-spent coins is offered as miner fees and miners can choose which fork they prefer for equal reward.

In an alternative take on incentivizing miners to abandon the most-work chain rule, Aljosha Judmayer et al describe a smart contract that will compensate miners if they can prove they are mining on the fork that the contract specifies, including or excluding specific transactions in blocks or ordering transactions in a certain way [11]. This method improves upon whale transactions by generalizing the set of miner behaviors that can be incentivized to anything that is verifiable by a smart contract, and by compensating participating miners

even if the attack fails. Whale transactions require the attack to succeed for the participating miner to receive any guaranteed compensation leading to risk-averse miners not participating, reducing the pool of bribe-able hashrate.

"An Axiomatic Approach to Block Rewards" [35] considers the properties that a block reward allocation rule should satisfy in the context of miners competing to generate a single block. They show that Bitcoin's proportional rule is optimal and unique for the ideal properties they describe, if miners are risk-neutral. If miners are risk-averse they provide an impossibility result saying that there is no non-zero allocation rule satisfying their ideal properties. By relaxing the set of desirable properties they discuss other possible allocation rules and which subset of the properties they satisfy.

3.3 Efforts at Attack Detection

Two websites initially drew attention to the low cost of a 51% attack using Nicehash on some cryptocurrencies: "Crypto51" [13] and "How Many Confs?" [14]. *Crypto51* calculates the US Dollar price to rent 51% of a cryptocurrency network's current hashrate for an hour based on the rental price on Nicehash for the hash function used by the coin for mining. Similarly, *How Many Confs?* uses the same data to calculate the number of confirmations you would need to wait for a transaction on a given cryptocurrency for the cost to be the same as performing a 6-block deep reorg on Bitcoin. Neither of these websites take into account the depth of the Nicehash market and how the price might change if an attacker were to attempt to rent such a large amount of hashrate. *Crypto51* indicates whether there is currently enough hashrate allocated to orders in the Nicehash market for a hash function to exceed 51% of the network hashrate. These two websites showed that the theoretical cost of attack for many small cryptocurrencies was much lower than expected, sometimes lower than a few hundred dollars.

fork.lol [15] and a website by Pieter Wuille [16] display plots of the number of days it

would take rewrite the Bitcoin blockchain with the current network hashrate. Since mining hardware tends to improve in performance over time, the time to generate the total chain work required to overtake the existing chain (causing a reorg back to the genesis block) does not always increase. In fact, there have been periods of time when the network hashrate could have regenerated the entire chain work in under fifty days. *fork.lol* expands this analysis to include the Bitcoin fork Bitcoin Cash. It provides plots of how long it would take for Bitcoin's network hashrate to regenerate the total chain work of Bitcoin Cash which stands at twenty-five days at the time of writing. The plots indicate the relative cost of attacking each coin over time and could help to suggest if an attack is possible due to changing market conditions and hardware performance.

As the primary targets of double-spend attacks, exchanges are motivated to detect attacks quickly so that deposits and withdrawals using the coin can be disabled. At Coinbase, Mark Nesbitt detected and reported double-spend attacks on Ethereum Classic and Vertcoin using their proprietary reorg detection system [17][18]. Similarly, BitMEX operates a public fork detector currently monitoring Bitcoin and Bitcoin Cash [19]. It has detected a number of non-malicious orphan blocks on Bitcoin, as well as reorgs on Bitcoin SV that were caused by long block propagation delays due to abnormally large blocks [20].

3.4 Other Reordering Attacks

Selfish mining is a technique first described in a paper by Ittay Eyal and Emin Gün Sirer in which a miner delays broadcasting the blocks they find to other nodes in order to mine a larger proportion of the blocks in expectation over time [33]. When an adversary controls more than a third of the network hashrate they can keep the block they find secret in the hope that they mine a subsequent block that extends their own chain. Other miners will have no knowledge of the secret chain and will continue to mine blocks referencing the previous most-work block. Thus, if the adversary is successful in mining two blocks while the rest

of the network miners one, the adversary can broadcast their secret chain and replace the block from the other miners. The adversary waits a timeout before broadcasting their block if they are not successful in out-pacing the public chain, creating a risk that their block will be orphaned. However, if the selfish miner's hashrate is greater than a third, the expected reward from delaying block broadcasting outweighs the risk.

Flash Boys 2.0 by Philip Daian et al. studies the transactions issued by arbitrage bots to decentralized exchange smart contracts on Ethereum and discovers that they engage in front-running [34]. The bots exploit the fee market for unconfirmed transactions to observe the pending orders of other traders and outbid the fees of their transactions so that the bot's orders receive sequencing priority in blocks. This allows adversaries to make more profitable trades than ordinary users by anticipating the other traders' orders ahead of execution and changing their strategy in response. They also introduced a term called "miner extractable value" that describes the profits miners can gain from smart contracts, such as increased overall fees from competition due to front-running.

Miles Carlsten et al. analyze how rational miners would behave when Bitcoin (and other deflationary Nakamoto consensus cryptocurrencies) no longer have a block subsidy and compensate miners solely with transaction fees [36]. They show that miners will choose to cause reorgs and fork existing blocks from the chain in order to include the transactions they contain in their own blocks, meaning they receive the transaction fees. In circumstances where there is a transaction with an especially fee relative to expected rewards for some time in the future, miners will repeatedly fight over the transaction, causing the chain not to make progress. The research brings into question the long-term feasibility of deflationary cryptocurrencies that use Nakamoto consensus unless the transaction fee market is sufficiently liquid to reduce miner reward variance.

Chapter 4

Methodology

The methodology chapter describes the design of our system for detecting reordering events on cryptocurrencies and how it detects any associated double-spends. Additionally, it discusses potential strategies for handling divergent coin implementations and how the strategy we selected affects the space of detectable events. We also describe the configuration of our system deployment for data gathering and enumerate the list of cryptocurrencies that were monitored as part of our study. The second section explains how we gathered historical price and hashrate rental market data, and related it to event variables for estimating quantities such as reorg cost, attack profitability and feasibility. Finally, the third section presents a series of automated policies that we implemented, which could be deployed by potential victims to prevent further damage due to a successful attack, or help victims coordinate with others to launch an active defense.

4.1 Reorg Tracker

4.1.1 System Design

We built a new system to detect reorgs across multiple cryptocurrency networks in order to gather empirical data on the frequency of reorgs and analyze the blocks involved to suggest

how much money could have been stolen. It was necessary to design and deploy a new system because of how little data was previously available that reliably stored reorg blocks for later analysis. Although cryptoID[1] provides aggregate data on the frequency of orphaned blocks for the coins it supports, and Etherscan[2] saves Ethereum blocks that were previously in the primary chain, their data is insufficient for in-depth analysis and covers too few cryptocurrencies. We used the system to monitor twenty-three different cryptocurrencies at various starting dates with the earliest observations beginning in June 2019. We ran full nodes using the respective coin developers' software for each of the Bitcoin-like cryptocurrencies we monitored. We also ran full nodes for some Ethereum-like cryptocurrencies except for Ethereum and Ethereum Classic[3] for which we used public APIs to access blockchain data.

We wrote a monitoring process responsible for detecting, analyzing and saving reorg events. Each coin had an individual process that connected to either the coin daemon or a remote block explorer API and repeatedly requested the current head block. If the head block changed, the monitor used a series of *getblock* RPCs to retrieve any unseen blocks and construct its own block graph in memory. *getblock* is a simple RPC supported by every coin we monitored that simply returns the raw block in JSON format for a given block hash or height in the chain. If a new head block caused the block data source to switch to a new fork in the block graph, a reorg had been detected, and the blocks involved were saved to the database.

Figure 4.1 illustrates the state of the reorg tracker's block graph and which blocks are saved to the database when a new head block is contiguous with the existing head block, and when a new head block causes a reorg by referring to a block deeper in the chain. Figure 4.2 is a block diagram describing the components of the reorg tracker, how they interact with external data sources and the flow of data between components.

[1] `https://chainz.cryptoid.info/`
[2] `https://etherscan.io/`
[3] We used Infura to monitor Ethereum and ethereumclassic.network to monitor Ethereum Classic..

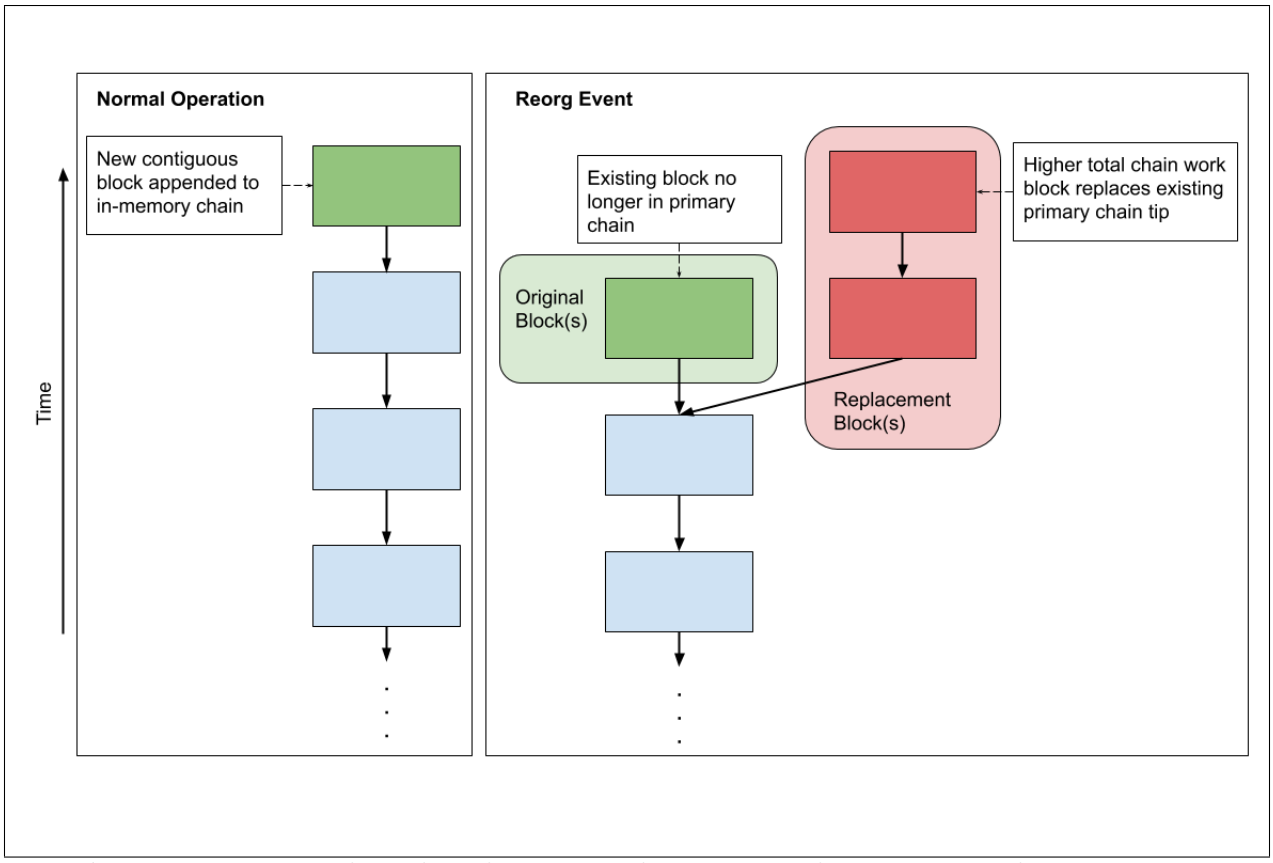

Figure 4.1: Internal tracker chain state during normal operation and reorg events.

Supporting Multiple Protocols

The system was designed to be generic across different cryptocurrency implementations because there are hundreds of active proof-of-work cryptocurrencies, each of which use different codebases that have many variations in terms of available RPCs, block and transaction formats. While many cryptocurrencies inherit the majority of their code from a common parent (usually Bitcoin or Ethereum), the code is forked form varying versions of their upstream, resulting in subtle divergence between implementations. Since we wanted to monitor a broad spectrum of the cryptocurrency market, a generic design was required to avoid extensive re-implementation when adding new coins to the tracker. As a result, the reorg tracker requires two functions to be implemented for each cryptocurrency. The first is $poll(tip\_id)$ which returns either the block following the block with the given tip_id, $null$ if there is no newer block or the block following the fork block if the tip_id is no longer in the primary

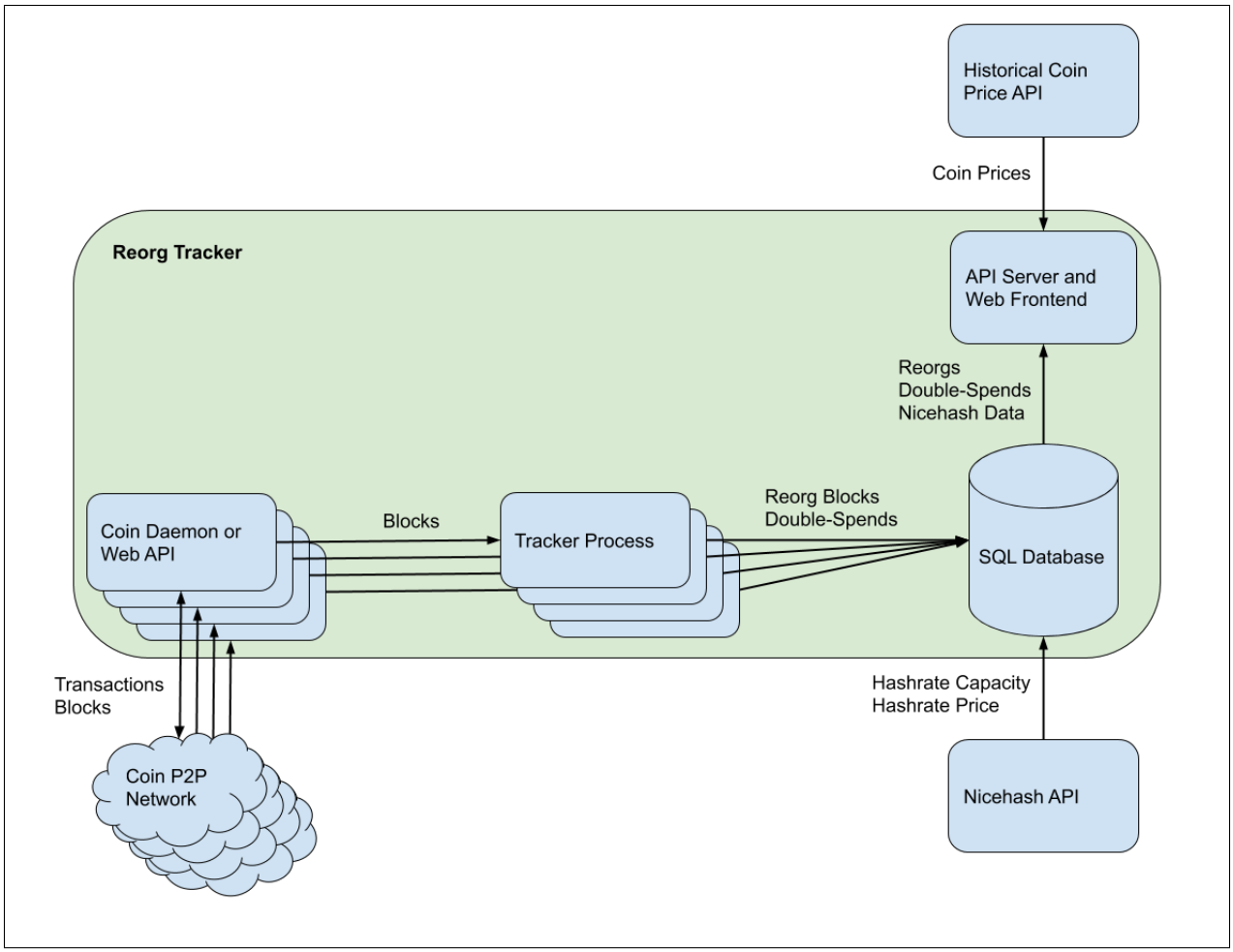

Figure 4.2: Block diagram describing the components of the Reorg Tracker. Components inside the box labeled "Reorg Tracker" were internal to the system and operated on our own server. Blocks outside the box are external and third party data sources.

chain (i.e. a reorg has occurred). The second is *genesis_id*() which returns the block ID of the genesis block.

These two functions can usually be implemented using only *getblock* RPCs to cryptocurrency node daemons that return JSON serialized block data given a block height in the primary chain or block ID. A module to support a new coin thus only has to implement two functions that translate JSON-RPCs from the coin daemon to a set of classes in software that represent blocks and transactions in a generic format. The blocks and transactions for diverse systems can then be easily saved in a structured database and analyzed once reorg events are detected.

Saving blocks directly to the tracker's database removes the reliance upon assumptions about the node software's behavior for storing block data. Node software could be unreliable and crash during a deep reorg or while processing a malformed block. This could lead to data corruption and the inability to retrieve blocks involved in reorg events for future analysis. Node software does not store orphaned blocks indefinitely and access to blocks via *getblock* RPCs could be lost if software is updated or a bug necessitates a chain resynchronization. Furthermore, running nodes in "pruned mode" where old blocks are discarded once they have been processed removes access to historical blocks that remain in the primary chain. When using remote APIs to access block data the reported state of the chain can even be inconsistent between *getblock* calls, where an API will successfully return a block for one call, but complain the same block cannot be found for subsequent calls. This seems to be caused by high-traffic APIs multiplexing requests between multiple back-end coin daemons, each of which have slightly different views of the network due to varying delays and peer connections. Thus in order to support a diverse range of coins and not require a software review for each new implementation it was safest to store relevant block data ourselves.

While using only *getblock* RPCs leads to a more universally usable system across different protocols, it trades off some amount of the potential granularity in recording changes to the state of a given chain. Our system is only able to observe blocks that were included in the primary chain at some point in time while the tracker was actively monitoring the network. This means that the tracker is unable to detect blocks that were orphaned from our nodes' perspective and never became part of the primary chain. Due to network delays in receiving a block, another instance of the reorg tracker hosted in a different location might have received the orphaned block first and thus recorded a reorg event when the block was superseded and became an orphan. Similarly, the reorg tracker cannot see minority chain forks even if they are being publicly generated. While a minority fork could be indicative of an in-progress attack, unless the fork becomes part of the primary chain it is invisible to the reorg tracker.

The tracker records the time when it sees a block (the first time the block is returned by *poll*). Although blocks contain their own timestamp for which each protocol has rules about the range of allowed values, these are updated infrequently by miner software, leading to minutes of drift between node receipt times and block timestamps. When a reorg occurs, any blocks that were previously in the minority chain are given the receipt time of the block that caused the reorg. This is because the tracker had not seen the blocks until they became part of the primary chain and thus does not know if they were received by the node at an earlier time. This makes it difficulty to know via data reported by the tracker whether an alternative chain was generated in secret and broadcast to the network in full once it overtook the existing primary chain, or was broadcast incrementally as the chain was being mined.

Alternative approaches to *getblock* RPCs would have been to use *getchaintips* RPCs, monitor peer-to-peer network traffic directly or modify node software to report the desired information about reorgs and orphans. *getchaintips* returns the set of distinct tip blocks for each chain fork the node has seen. While *getchaintips* can return information about blocks that have never been part of the primary chain, it is only implemented by node software that is based upon relatively recent versions of Bitcoin Core. Recording peer-to-peer traffic would provide the most detailed view of the chain state but would require reimplementing the protocol rules for each coin. The network messages would have to be processed in order to determine which blocks are valid and at which point in time (if any) a block was part of the primary chain. Modifying node software to return detailed chain information for many coins is cumbersome due to the large variations between codebases.

Double-spend Detection

Once a reorg has been detected and saved, the system also analyzes both sides of the fork for double-spent transactions. In a more general sense, double-spends can be interpreted as pairs of transactions that are mutually exclusive (both cannot exist in the same chain)

and are each included on one side of a fork. For Bitcoin-like cryptocurrencies that use unspent transaction outputs (UTXOs), mutually exclusive transactions have to spend the same transaction output. This can lead to a complex double-spend situation where the relationship between double-spent outputs in a reorg is not bijective. Account-based systems are easier to analyze for double-spends as transactions have one input and output. The input to an account-based transaction is a pair of an account ID and a nonce, thus a mutually exclusive transaction simply has to use the same account-nonce pair as its input, leading to a bijective relationship between the original and replacement transactions. Figure 4.3 illustrates the difference in possible relationships between double-spent transactions in both types of systems.

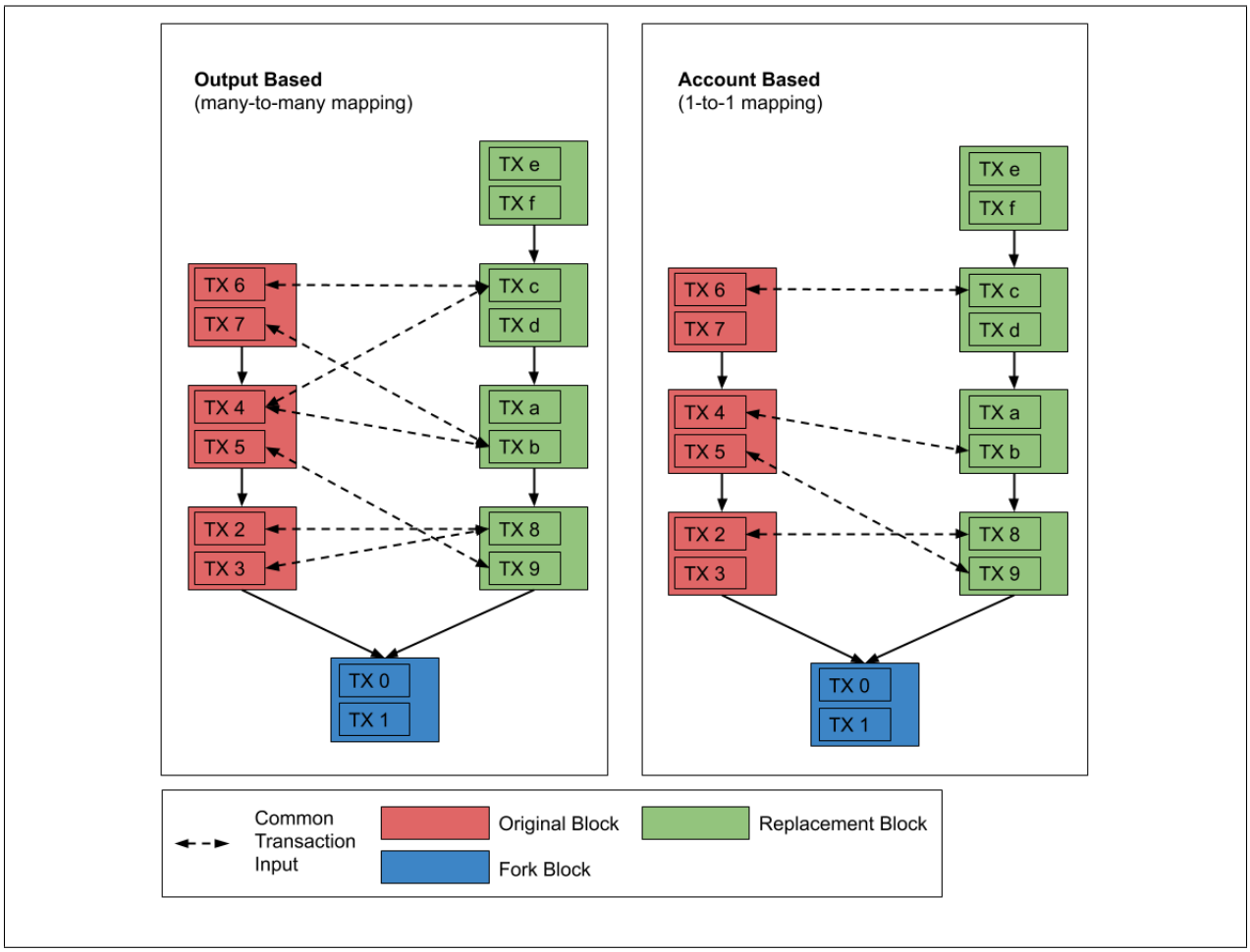

Figure 4.3: Possible mappings between inputs to double-spent transactions in output-based versus account-based systems.

Each of the coins we monitored provide at least a pseudonymous level of transaction privacy. This means that we do not know who the addresses belong to for transactions that are involved in reorg events. Although we can identify the addresses themselves and the amount of coins moved between each address, we cannot know the identity of the victims or perpetrators of a double-spend attack. This makes determining the cost of an attack and the amount of coins stolen an estimation based on assumptions about the typical format of transactions and the motivation of the attacker. In all cases an attack that could be considered successful could equally be an entity attacking themselves, where every address involved in the event is owned by the same people. Additionally, wallet software can cause an apparent double-spend to occur with no malicious intent if it allows the user to increase the fee offered for an existing transaction. The software will generate a new transaction re-spending the output or account nonce with the same recipient and amount but a greater fee and broadcast it to the network. If the two transactions then get included on opposing sides of a reorg, the event appears to the reorg tracker as a double-spend, even though the recipient receives the same amount of coins.

4.1.2 System Deployment

Tracked Coins

We selected coins to add to the reorg tracker largely based on the order of their market capitalization and whether they were known to have previously suffered from deep reorgs. Each of the coins were based either on Bitcoin Core or Geth for which we had implemented modules for the reorg tracker, with specific caveats for the differences between each coin's protocol.

Table 4.1: List of coins tracked by the Reorg Tracker and the date of first reorganization observation.

Ticker	Name	1st Event	Mkt Cap [4]	Algo	Base	Block Time (s)	Notes
BCH	Bitcoin Cash	19-08-21	$4.4bil	SHA256d	Core	600	
BSV	Bitcoin SV	19-07-05	$3.7bil	SHA256d	Core	600	
BTC	Bitcoin	19-10-16[5]	$130bil	SHA256d	Core	600	
BTG	Bitcoin Gold	20-01-23	$180mil	ZHash	Core	600	
CLO	Callisto	19-07-03	$1.5mil	Ethash	Geth	13	
DBIX	Dubaicoin	19-08-13	$240k	Ethash	Geth	89	
DGB	Digibyte	19-12-10	$100mil	Multiple	Core	15	[6]
DOGE	Dogecoin	19-09-14	$250mil	Scrypt	Core	60	[7]
ETC	Ethereum Classic	19-06-22	$650mil	Ethash	Geth	13	
ETH	Ethereum	19-06-21	$20bil	Ethash	Geth	13	
EXP	Expanse	19-07-02	$470k	Ethash	Geth	21	
HANA	Hanacoin	19-12-13	- [8]	Lyra2REv3	Core	90	
IMG	Imagecoin	19-08-21	$450k	X11	Core	60	
LCC	Litecoin Cash	19-07-04	$2.7mil	SHA256d	Core	150	[9]
LTC	Litecoin	19-09-06	$2.9bil	Scrypt	Core	150	
MONA	Monacoin	19-06-26	$79mil	Lyra2REv2	Core	90	
PAC	PAC Global	19-08-13	$1.0mil	X11	Core	187	[10]
PIRL	Pirl	19-08-06	$230k	Ethash	Geth	17	
RVN	Ravencoin	19-12-14	$100mil	X16Rv2	Core	60	
VTC	Vertcoin	19-06-27	$14mil	Lyra2REv3	Core	150	
XVG	Verge	19-12-09	$45mil	Multiple	Core	30	[11]
ZCL	ZClassic	19-07-01	$1.3mil	Equihash	Core	75	[12]
ZEC	ZCash	19-06-29	$380mil	ZHash	Core	74	[13]

[4]Market capitalizations are in USD to two significant figures as reported by `https://coinmarketcap.com/` on 2020-04-18.

[5]Coins such as Bitcoin that have very infrequent reorgs were being tracked earlier than their first observation date suggests.

[6]Digibyte blocks can use one of five mining algorithms: Scrypt, SHA256d, Skein, Qubit and odocrypt. There is an independent difficulty target for each algorithm. The difficulty adjustment algorithm aims to maintain an equal proportion of blocks using each mining algorithm over time.

[7]Dogecoin is often merge-mined with Litecoin and other Scrypt proof-of-work coins.

[8]Historical price and market capitalization data for Hanacoin is unavailable.

[9]Litecoin Cash blocks can use either a proof-of-work using SHA256d or a valid proof-of-stake lottery ticket to be valid. Proof-of-stake ticket difficulty adjusts so that tickets mature at a fixed rate per n blocks. Tickets cost a fixed amount of LCC regardless of the volume of tickets being purchased. Initially LCC required any proof-of-stake block to follow a proof-of-work block, but this was changed after an attack was detected to require strict alternation between proof-of-work and proof-of-stake blocks.

[10]PAC switched from X11 proof-of-work to a masternode-based proof-of-stake system on 2019-07-29.

Server Configuration

We operated the reorg tracker for just over nine months on a single server located in Cambridge, MA. The server ran coin daemons for each coin (except for ETH and ETC) and their associated tracking processes, a database server and an API server for accessing data. Each coin daemon or tracking process ran in its own docker container to provide isolation between their system environments. The coin daemons were not permitted to listen for incoming connections from their peer-to-peer networks and used the software's default peer management algorithm to make outgoing connections to other nodes. If permitted by the software, the coin daemons were run in pruned mode with up to two gigabytes of archival block storage.

Note that our data is from the perspective of one node (or third party in the case of Ethereum and Ethereum Classic) so nodes located in different networks or geographical locations could have observed reorg events at different times or not at all. Similarly, there may be events that other nodes observed that our node did not. Furthermore, due to the reorg tracker's design, blocks that were orphaned and never appeared in our node's primary chain are not detected as reorgs. Another node could have received the orphaned block first and thus included it in its primary chain meaning that it would have observed the orphaning event as a reorg.

However, we do not think our single server approach negatively affects the lessons that can be learned from the data. We believe it is realistic that a user would only run a single node per coin and it is unlikely our nodes were singled-out and eclipsed since our system was not a public target for attack. Therefore, aggregate data regarding shallow, randomly occurring reorgs is likely to be representative of the situation for the overall network. Deep reorg data is unlikely to differ significantly based on a nodes' vantage point because the

[11] Verge blocks can use one of five mining algorithms: Scrypt, X17, Lyra2REv2, myr-groestl and blake2s. There is a independent difficulty target difficulty for each algorithm. The difficulty adjustment algorithm aims to maintain an equal proportion of blocks using each mining algorithm over time.

[12] ZClassic uses zero-knowledge shielded transactions making it impossible to fully determine transaction volume.

[13] ZCash uses zero-knowledge shielded transactions making it impossible to fully determine transaction volume.

length of time over which deep reorgs occur is much longer than the network propagation delay or block interval.

4.2 Event Analysis

4.2.1 Market Data

In order to estimate the cost of mining the blocks for each reorg, the value of any coins that were double-spent and the possibility of rental markets being used to launch attacks, we collected price data for each coin and hashrate rental prices for each mining algorithm. We gathered hashrate price and available quantity data as provided by Nicehash's public API.[14] This data is quantized into market snapshots every fifteen minutes from which we used the closest data point to our desired timestamp when querying the dataset. A Nicehash order contains the order ID, the mining algorithm the order is for, the bid price, how much hashrate is currently allocated to the order, the maximum hashrate the order will accept and the server region the order is for. Nicehash does not reveal their matching algorithm for allocating general-purpose computing power to the market for each mining algorithm.

Nicehash only provides the instantaneous amount of hashrate currently allocated to the market for an algorithm. Therefore, it is not possible to determine the true market supply of hashrate on Nicehash for a given algorithm as we do not know how hashrate would be reallocated from other hash functions, or how many additional miners would enter the market if higher price bids were placed. Instead, we assume Nicehash operates as an efficient market and take the instantaneous hashrate availability as an estimate of the market supply for the rental price at a specific point in time. Nicehash hashrate is priced in Bitcoin so we used the market price data for BTC to convert hashrate prices to US Dollars.

We gathered historical market price data in US Dollars for each coin using a public

[14] API documentation: `https://docs.nicehash.com/`

API from CryptoCompare.[15] This data is available as market snapshots every five minutes providing a high, low, close and open price. We used the closest data point to our desired timestamp when querying the dataset and took the close price as the spot price in our calculations. Since many of the cryptocurrencies we monitored do not have direct markets with US Dollars, CryptoCompare converts price information from available trading pairs (usually with Bitcoin) using the historical price of the base currency in US Dollars.

4.2.2 Estimating Event Variables

Reorg Duration

We considered the start of a reorg to be the time at which the reorg tracker first saw the "fork block" (its "receipt time") which we receive before we know that a reorg will take place. Similarly, we used the receipt time of the block that caused the reorg as the event end time. The block that causes the reorg event is the first block of the alternative chain that has a greater total chain work than the tip block of the existing primary chain. The difference between these two timestamps we use as an estimate of the "reorg duration", which is the time taken for the alternative chain to be generated.

We measure the "depth" of a reorg as the number of blocks removed from the existing primary chain. This is the difference between the height of the fork block and the height of the tip block of the existing primary chain. Similarly, the "length" of a reorg is the number of new blocks added to the primary chain, or the difference between the fork block height and the height of the block that caused the reorg.

The "chain work delta" is the difference between the total work of the fork block and the block that caused the reorg. This value represents the expected number of hashes that would need to be performed by a miner in order to generate the given set of blocks. This is equivalent to the sum of the difficulties for each of the blocks in the sequence.

[15] API documentation: `https://min-api.cryptocompare.com/documentation`

Budish Reorg Cost

The Budish reorg cost uses the efficient market hypothesis to argue that the cost of generating a set of blocks should be equal to the expected reward for the miner. We estimate this quantity for each reorg by summing the miner reward for each block in the alternative chain that caused the reorg event and converting the total to US Dollars. This is trivial to calculate for output-based cryptocurrencies as the newly generated coins and transaction fees awarded to miners are enumerated in a single transaction within a block (the "coinbase"). Note that this is even the case for cryptocurrencies with shielded transactions, as the transaction fees are always unshielded and still included in the coinbase transaction.

For account-based cryptocurrencies, one must determine the amount of "gas" used by each transaction (as reported by the transaction receipt) and multiply by the transaction's gas price to determine the fee denominated in the base currency. These values can then be summed for each transaction in a block, and added to the amount of newly-generated coins per block to calculate the total miner reward. We do not perform a detailed analysis of how "uncle" payments (which compensate miners of orphan blocks in cryptocurrencies based on Ethereum) increase the overall miner reward, though we do provide an upper bound on their potential value.

Nicehash Reorg Cost

The Nicehash reorg cost is an estimation of the cost to generate the alternative chain of a reorg using the hashrate spot price on Nicehash at the start time of the reorg. Hashrate is priced in Bitcoins per hash per second per day. To convert this into a cost to generate the reorg, we multiply by the chain work delta to provide a rate of expenditure in Bitcoins per second per day. By dividing by the number of seconds in a day, one acquires the reorg cost in Bitcoin, which is finally converted to US Dollars.

This quantity is difficult to calculate for multi-algorithm cryptocurrencies as there are multiple distinct hashrate markets involved and the relationship between the chain work

delta and the amount of hashrate contributed by each mining algorithm is less clear. We therefore omitted calculating the Nicehash reorg cost for multi-algorithm coins. Furthermore, the true Nicehash cost may be much higher than the estimate in the case where the speed of available hashrate in the market is less than or close to the speed required to generate an alternative chain with enough total chain work. This is because the attacker would have to outbid existing market orders to encourage additional hashrate to sell into the market, or to reallocate hashrate from existing orders.

Chapter 5

Reorg Analysis

Since there are two primary categories of reorg events requiring different analyses, we split the results chapter into two main sections. The first section focuses on analyzing shallow reorgs that probably occur due to the random nature of mining, are are not due to an active attack. We evaluate the effectiveness of the Nicehash rental price and block reward as methods for estimating reorg costs, the relative reorg characteristics between coins based on coin parameters and whether the cost of mining safeguards the coin from double-spend attacks. The second section explains the deep reorg attacks and double-spends that we detected. We present case studies into a subset of the more interesting attacks involving counterattacks, rental market usage and exploiting reorg defences. The final section discusses evidence for attacks on smart contracts through transaction reordering in shallow reorgs.

5.1 Dataset Summary

Table 5.1 shows a summary of the 10-month dataset. There is large diversity in the characteristics of reorgs between each coin. Some coins experience reorgs very infrequently with Bitcoin and Bitcoin Cash registering less than five events each over the course of the dataset. By contrast, coins such as Ethereum experienced nearly forty-thousand reorg events in the same period. For most coins the vast majority of reorgs are shallow being only one block

deep, with the remaining reorg depths following an exponential distribution. The relationship between reorg depth and the proportion of reorgs observed is shown in Figure 5.1.

Some coins do not follow an exponential decay trend for reorg depth. Litecoin Cash, Bitcoin Gold and Verge are most prominent from Figure 5.1. These three coins along with Expanse, Vertcoin and Hanacoin all experienced reorgs deeper than six blocks. In total we detected fifty such events. The six block confirmation time mentioned in Nakamoto's whitepaper was based on Bitcoin's ten-minute block interval, and thus it does not capture the different block intervals for each coin. Instead, it is more useful to normalize for each coin's block interval to determine how many "block-hours" have elapsed: $t = \frac{nb}{3600}$, where n is the number of blocks and b is the coin's block interval in seconds, such that one block-hour is six blocks on Bitcoin and twenty-four on Litecoin. In total we detected twenty-two reorg events one block-hour or deeper on Litecoin Cash, Bitcoin Gold and Vertcoin. The details of these events are described in more detail in Section 5.2.

Block-hours is a more useful quantity because it allows one to compare the actual amount of time a user would be expected to wait for a given amount of miner expenditure to be accumulated. Although block interval is not a factor in Nakamoto's security argument (equation 2.1), in the liquid hashrate model the security argument no longer applies as it relies on a majority of the hashrate being honest. Instead, the security argument in equation 2.6 suggests that no transaction value is safe unless the coin's price drops post-attack, there is a probability of attack failure due to manual intervention by users, or miners have large fixed costs which they would be unable to recover if the coin depreciates in value. In this case, the only security provided by Nakamoto consensus is measured by the up-front cost of capital for the adversary to attack their victim (which they recoup when the reorg takes place).

Coins with a large frequency of shallow reorgs have a corresponding large number of "total time reversed". Total time reversed represents the cumulative block-hours of orphaned blocks due to detected reorg events. This can usually be interpreted as wasted work, as miners

will have consumed energy to generate the blocks but do not receive any reward. This is not always the case for coins based on Ethereum for which "uncles" (orphaned blocks) can be cited in subsequent blocks giving their miners a smaller reward and without including any transactions or fees from the uncle [21]. An in-depth analysis of uncles and how they affect mining cost is not considered in this paper, though the following paragraph estimates an upper-bound.

Since the Constantinople Ethereum hard-fork, the uncle reward is ≈ 1.63 ETH per block which is $\approx 86\%$ of the primary block reward [22]. Thus uncles are certainly a non-negligible factor and may recoup a large portion of the estimated lost mining expenses. Although a block can only reference a maximum of two uncle blocks, in the time we detected 38970 reorgs on Ethereum, over 2 million blocks elapsed on the most work chain. This means there is ample capacity for every orphaned block we detected to be referenced and compensated as an uncle. Therefore, in the best possible case where the miner of every orphaned block is compensated, they only fail to earn 14% of the primary block reward and the transaction fees per orphan.

By multiplying the block-hours reversed by the Nicehash spot price for mining an hour's worth of blocks at the time of each reorg we acquire an estimate for the "lost mining expenses" incurred by miners of orphaned blocks. Total lost mining expenses is largely affected by the cost of mining blocks for a given coin, with Expanse having under half as much total block-time reversed compared to Ethereum, but over 3500 times fewer lost mining expenses. Our data shows that even considering uncle payments in Ethereum-like coins, miners across the industry cumulatively lose tens of thousands of dollars of mining revenue every month due to orphaned blocks in reorgs.

Assuming a liquid hashrate market, the "average reorg cost" for a coin is the mean block reward per reorg received by miners of the blocks that get added to the primary chain. The definition does not hold for all coins and is discussed in more detail in Section 5.1.3. That being said, average reorg cost gives a sense of the reorg generation cost that is typical

for a coin, with values ranging from negligible for Imagecoin to over $\$150,000$ for Bitcoin per reorg.

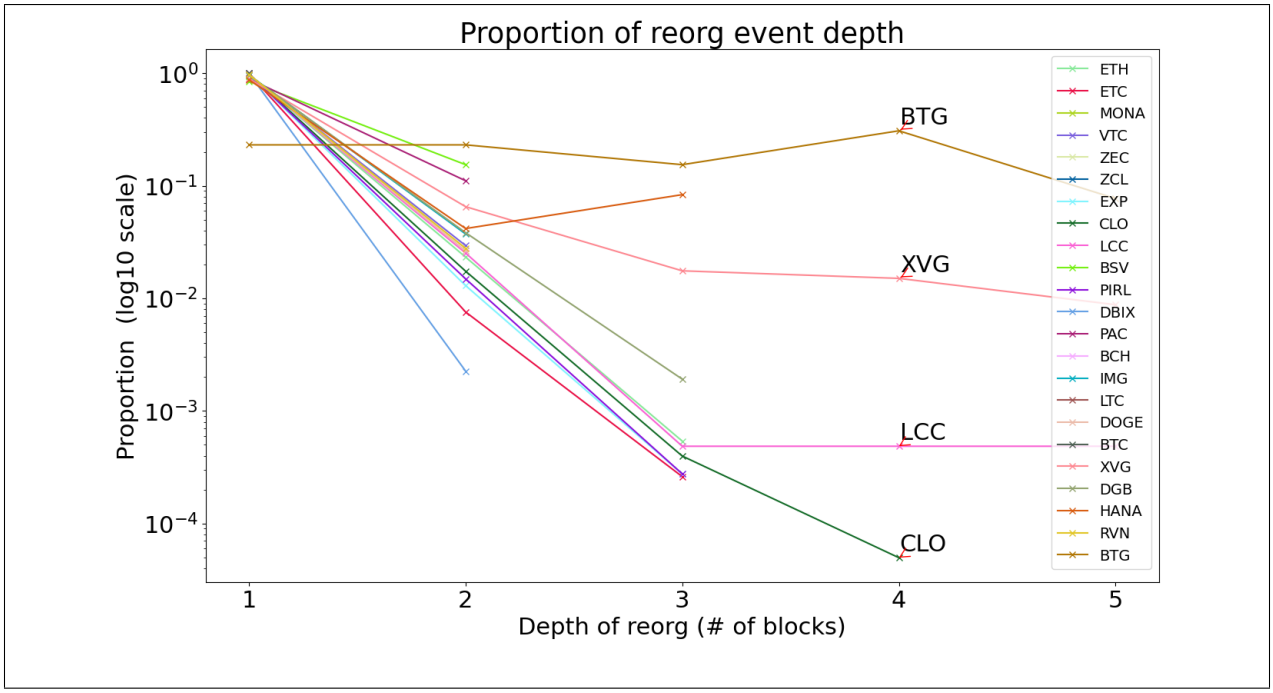

Figure 5.1: Plot of the proportion of the depths reorg events that were detected for each coin.

Table 5.1: Summary of all observed reorgs, their depths, costs and estimated losses to non-attacking miners. Total Time Reversed is the cumulative block time removed from the chain. This is the sum of all reorg depths in number of blocks multiplied by the coin's block interval in hours. Lost Mining Expenses is the cumulative estimated cost of mining if using Nicehash that would be wasted due to orphaned blocks in reorgs. Cells containing "-" represent missing data due to lack of historical price availability or methodology for estimating rental mining costs for multi-algorithm coins.

Ticker	Event Count	Total Time Reversed (hrs)	Lost Mining Expenses (USD)	Depth Max	Depth Mean	Depth StdDev	Average Reorg Cost (USD)
BCH	2	0.33	6991.87	1	1.0	0.0	7114.02
BSV	18	3.33	35909.67	2	1.1	0.3	5208.26
BTC	4	0.67	324075.25	1	1.0	0.0	166084.86
BTG	28	40.33	21305.5	23	8.6	6.4	925.47
CLO	20149	74.10	7789.27	4	1.0	0.1	0.01
DBIX	503	12.46	365.11	2	1.0	0.0	0.83
DGB	1048	4.55	-	3	1.0	0.2	6.56
DOGE	600	10.27	154274.37	2	1.0	0.2	47.15
ETC	3835	13.96	55315.49	3	1.0	0.1	22.56
ETH	38970	144.14	12825310.52	3	1.0	0.2	674.34
EXP	10876	64.76	3499.98	63	1.0	0.6	4.38
HANA	28	0.95	2.69	6	1.4	1.0	-
IMG	300	5.18	104.61	2	1.0	0.2	0.0
LCC	2512	124.96	15129.81	102	1.2	3.5	1.7
LTC	24	1.00	15353.47	1	1.0	0.0	1253.85
MONA	58	1.45	1371.17	1	1.0	0.0	59.47
PAC	271	15.64	33.64	2	1.1	0.3	0.83
PIRL	22188	106.36	1555.37	3	1.0	0.1	0.05
RVN	36	0.62	5218.69	2	1.0	0.2	273.72
VTC	36	26.62	3209.24	603	17.8	98.9	72.74
XVG	867	9.59	-	37	1.3	1.7	4.58
ZCL	96	2.00	132.43	1	1.0	0.0	8.24
ZEC	180	3.70	63336.02	1	1.0	0.0	932.22

5.1.1 Budish vs Nicehash Cost

One of the goals of this research was to evaluate the two primary methods of estimating the cost of mining in the context of reorgs to determine which coins operate in the liquid hashrate model and how closely measured block rewards equal estimated mining costs. To that end, in this subsection we compare using the block rewards as a proxy for mining cost (the "Budish Cost") to estimating the cost of mining using Nicehash (the "Nicehash Cost"). Figure 5.2 shows the average hourly cost of mining per block-hour on Nicehash versus the average block reward in the same time period for each coin over all reorgs. Due to general equilibrium theory, in a completely ideal liquid hashrate market one would expect these two quantities to be equal as miner revenue would equal expenditure to eliminate profits.

Figure 5.2 shows that the theoretical trend between block rewards and market-rate mining costs holds in a broad sense, though there are significant outliers. It is also useful for showing the exponential distribution of mining costs across coins. Bearing in mind that the figure is log-scale on both axes, coins are distributed linearly along the diagonal, indicating that mining expenditure in the market is clustering towards the most popular coins. Figure 5.3 instead shows the ratio between reward and cost in order to normalize for the relative cost of mining and value of rewards on each coin. This plot better demonstrates the difference between Nicehash market conditions relative to each coin.

One must consider that many of the coins share mining algorithms, and that the hashrate market will reach an equilibrium based on whichever coin in dominant for its algorithm. This explains why it is much more expensive to mine coins such as Callisto, Pirl, Dubaicoin, Expanse and Ethereum Classic on Nicehash than a miner would expect to receive in rewards. These coins all use "Ethash" as their mining algorithm which they share with Ethereum whose network hashrate is an order of magnitude larger than Ethereum Classic and four orders of magnitude larger than Expanse. The Nicehash market for Ethash has thus reached equilibrium based primarily on Ethereum's expected reward per hash making it inefficient to mine coins with a much smaller reward. Similarly, Dogecoin uses the "Scrypt"

mining algorithm which it shares with Litecoin, and Litecoin Cash uses "SHA256", used primarily by Bitcoin. It is unlikely that the hashrate market for an algorithm would reach equilibrium for a coin whose hourly miner compensation is significantly smaller than that of a larger coin. The trading volume in the hashrate market owing to the smaller coin would also be lower than the larger coin simply because renters will not spend more than they expect to receive in rewards.

The data provides evidence that the theoretical model stating that mining should be break-even in equilibrium is correct in practice. This suggests that if there is sufficient hashrate available to rent in the market, attacks involving reorgs should be at least break-even if the reorg succeeds. Security from reorgs in Nakamoto consensus must therefore come from other factors as a larger cost of mining only increases the initial investment required to initiate an attack. For coins whose hashrate rental price for its algorithm is much higher than its block reward, the attack would no longer be break-even using Nicehash. However, this does not make an attack impossible, rather an attacker would have to steal a larger amount to recoup the differential between reward and cost. For example in the case of Callisto, a cost-reward ratio of greater than seventy may not be prohibitive to attacks given its average reorg cost is only $0.01.

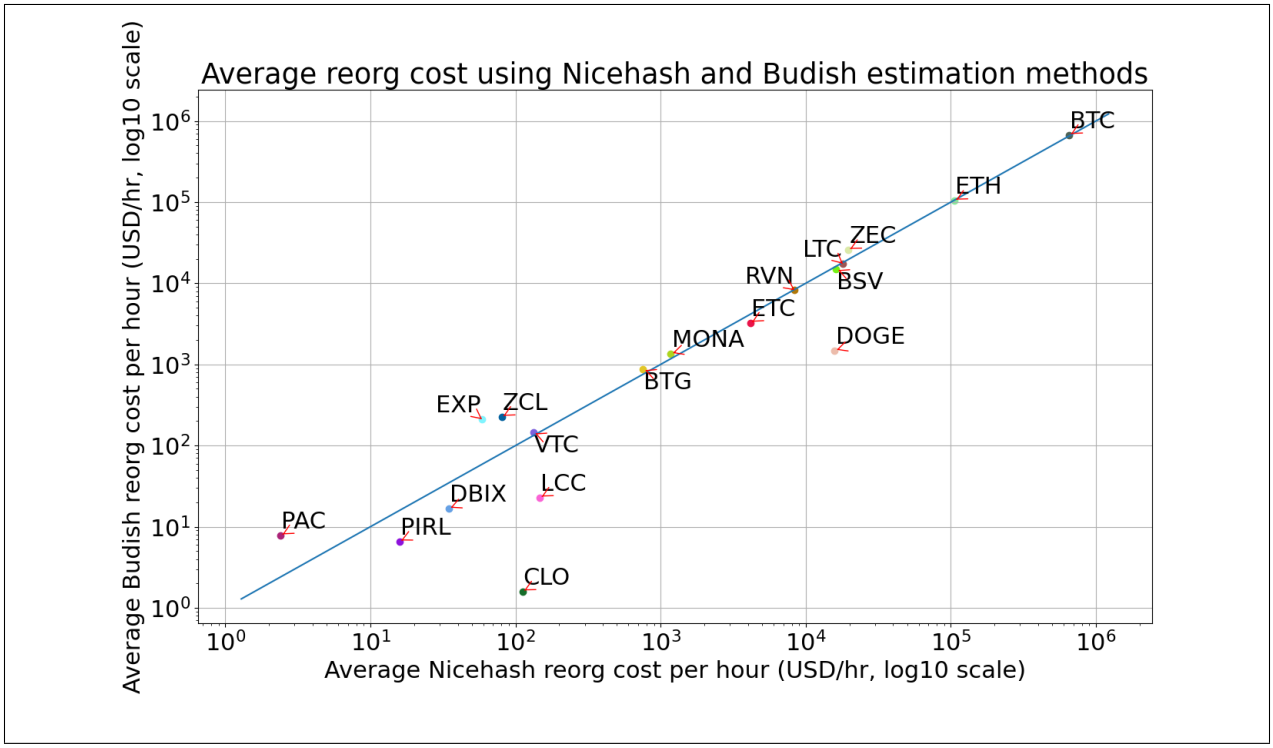

Figure 5.2: Plot of the average reorg cost using Budish and Nicehash estimation methods across all observed reorgs.

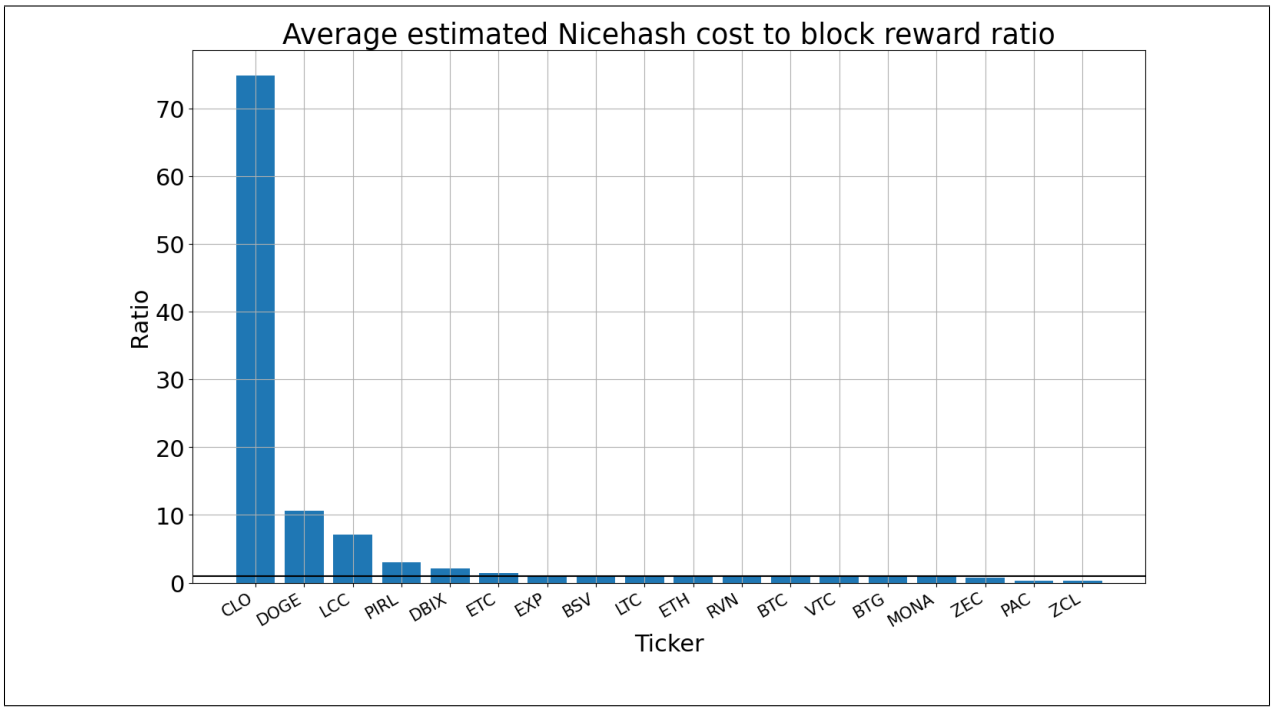

Figure 5.3: Plot of the average ratio between the Nicehash reorg cost estimation and the reorg block rewards. The line at a ratio equal to one indicates when the Nicehash market for a coin's algorithm is in equilibrium for that coin, meaning the estimated cost of mining equals the expected reward. Coins with very high ratio pay less in miner rewards per hash than the cost of rental per hash, usually meaning there is another coin with the same hash function that pays miners significantly more per hash.

5.1.2 Observed Reorg Rate

In this subsection we consider the characteristics of each coin and how they relate to its observed reorg rate. Figure 5.4 shows the average reorg reward per block plotted against the average daily reorg count. When interpreted in the liquid hashrate market model, it also shows the expected reorg cost assuming it is approximately equal to the block rewards. There is evidently an inverse relationship between reorg cost per block and the number of reorg events that occur per day.

The block interval of a coin also appears to be an important factor in determining the frequency of reorg events. Figure 5.5 shows that a coin's block interval is inversely related to its average daily reorg count. This is expected as blocks are found by miners at a higher rate in coins with shorter block intervals, leading to more opportunities per day for multiple miners to find a block within the propagation delay of the network. However, there is a wide variance in reorg frequency for coins with the same block interval suggesting that the reorg cost per block is an important complimentary metric.

The clear outlier in the dataset is Ethereum. Ethereum's disproportionately high reorg frequency compared to its block interval and reorg cost per block might be explained by the method we used to observe the network. As mentioned in Section 4.1.1 we used Infura, a remote API to access Ethereum chain data. This API has multiple distinct backend nodes that serve block data, each of which has a slightly different view of the network. Coupled with Ethereum's short block interval, successive API calls result in being served by a different backend node that returns a contradictory block tip at a higher rate than if querying a single local node. However, we are not certain this behavior is the sole cause of Ethereum's high observed reorg rate and there could be other factors involved such as the high value of uncle rewards or efforts to front-run smart contracts.

By contrast, Figures 5.6 and 5.7 show that average reorg depth is uncorrelated with the average reorg cost per block or block interval respectively. We have removed the coins that experienced deep reorgs from the plots in order to not skew the axes. This makes it

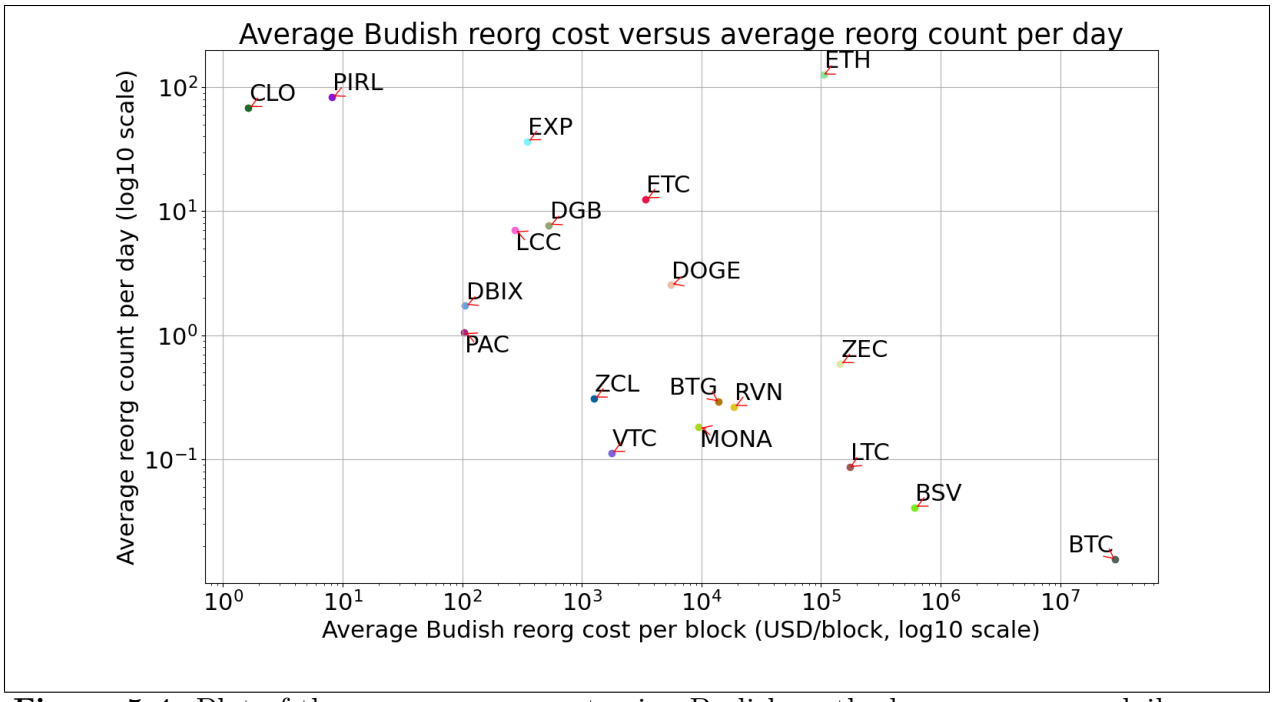

Figure 5.4: Plot of the average reorg cost using Budish method versus average daily reorg count.

difficult to predict the distribution of reorg depths for a coin without empirical measurement. Instead, other factors such as network connectivity, block size and miner behavior must be determining the distribution. It is important to know the expected rate and depth of reorgs for a coin so that users can set sane confirmation requirements for the normal case in which there is not an attack. Further research into the determinants of the distribution of random reorg depths between coin is left to future work.

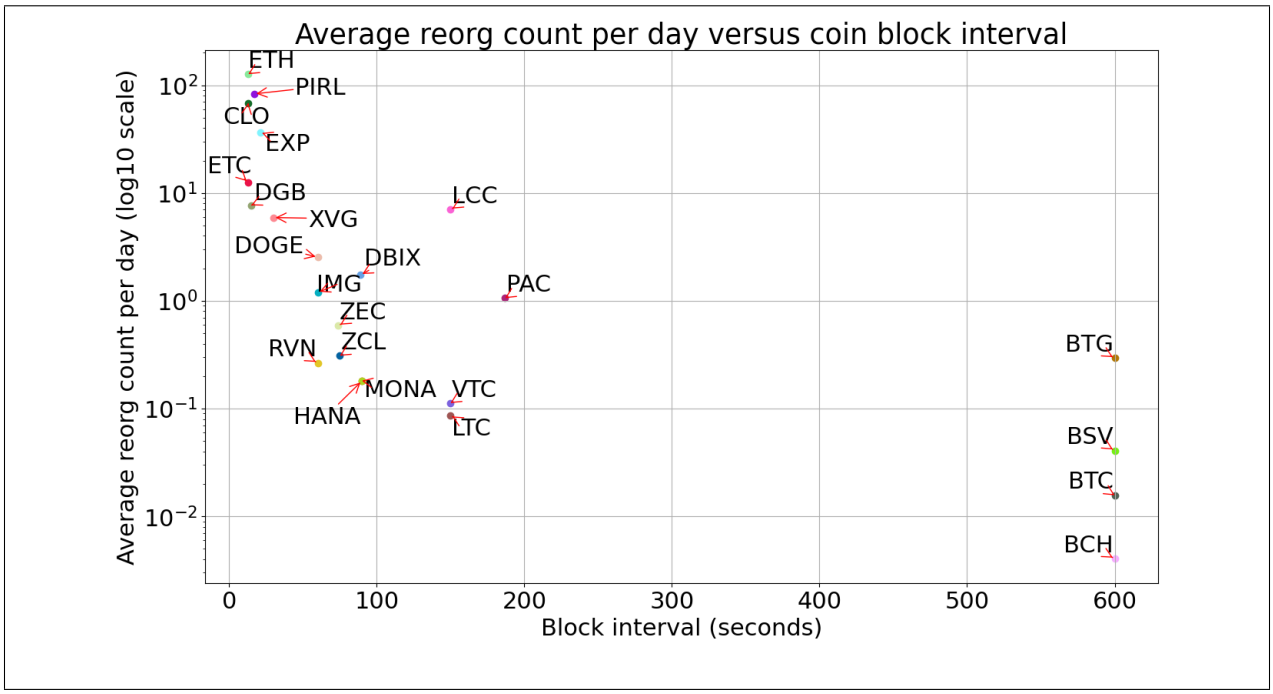

Figure 5.5: Plot of the average number of reorgs per day compared to the block interval for each coin.

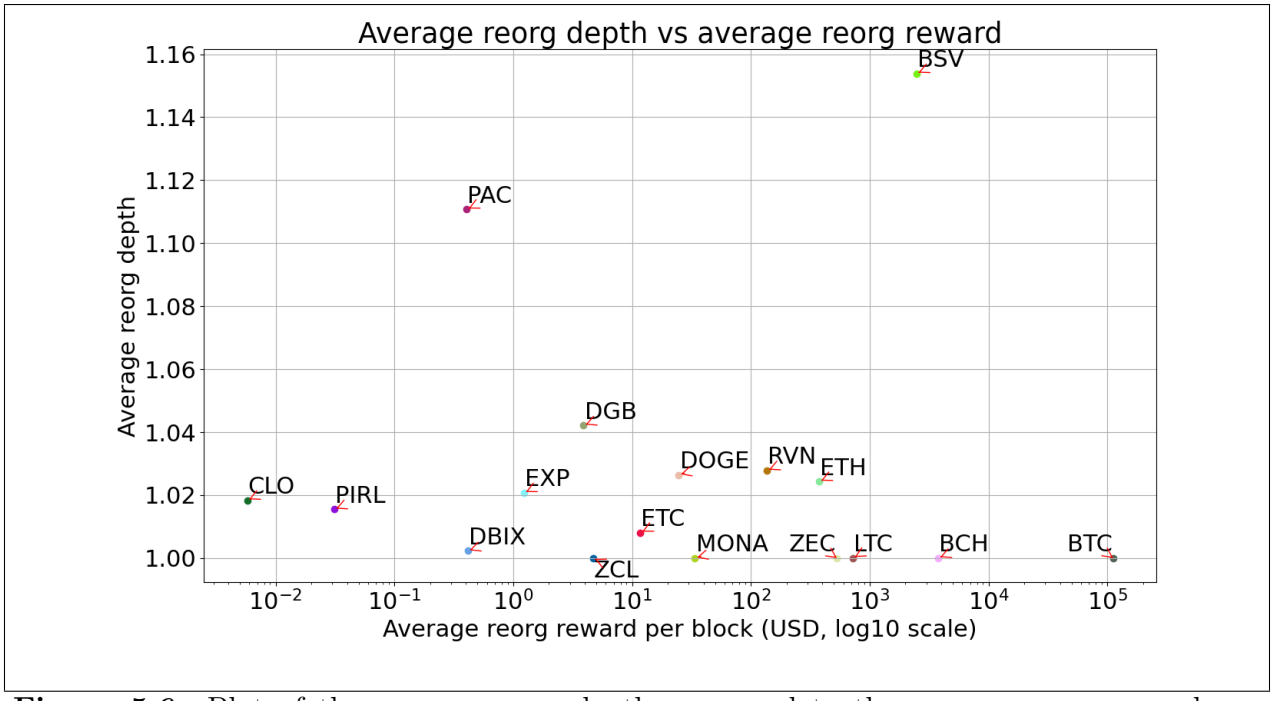

Figure 5.6: Plot of the average reorg depth compared to the average reorg reward per block.

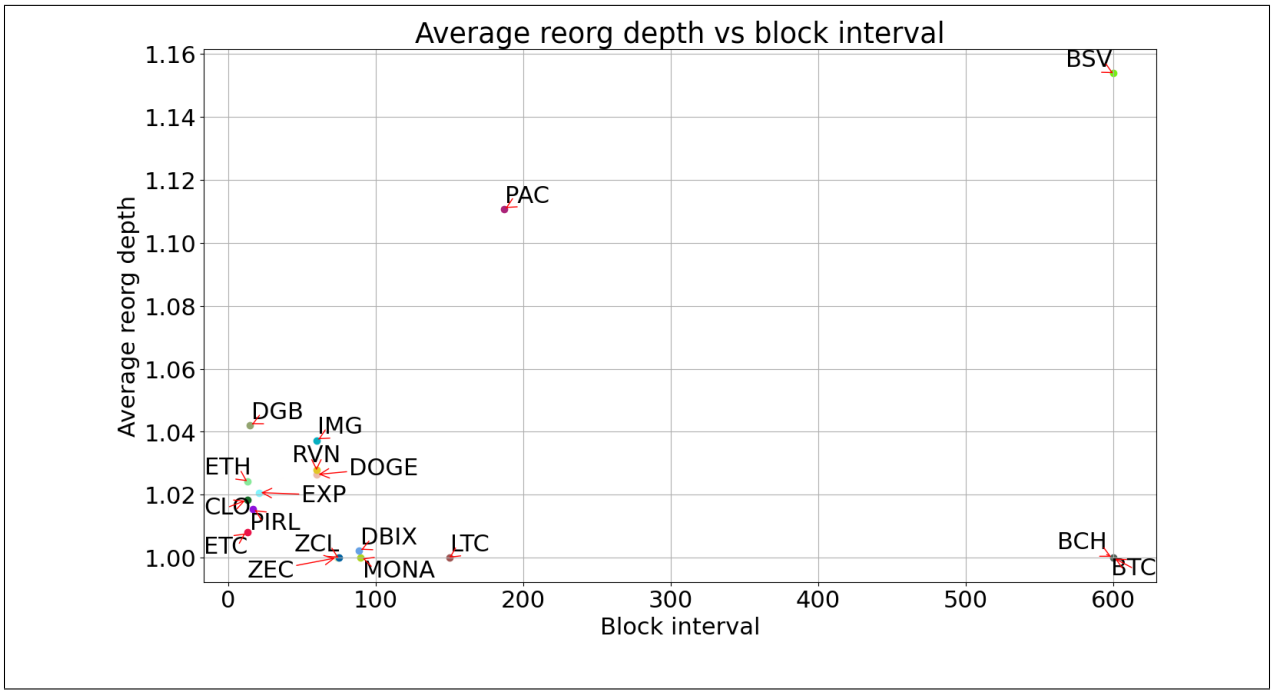

Figure 5.7: Plot of the average reorg depth compared to block intervals.

5.1.3 Transaction Safety

This subsection analyzes the security coins provide against transactions being reverted or reordered. We also consider what potential there is for coins to be attacked using Nicehash. Users often set arbitrary numbers of elapsed blocks to wait before considering a transaction confirmed, a practice we aim to dispel by providing empirical data on the consensus risk of different coins. So that we can compare the relative security of coins with different block intervals, it is useful to normalize numbers of blocks to block-hours. This allows us to compare the amount of time a user would have to wait for a transaction to be secure from a certain initial attack investment threshold. Furthermore, since an attacker could be double-spending every unconfirmed transaction in the worst case, we have to consider the overall transaction volume as the maximum possible attack reward. Thus, we compare the ratio of transaction volume to miner rewards for each coin.

Figure 5.8 plots the maximum observed reorg depth for each coin in block-hours. At initial glance the figure may be surprising because it shows Ethereum with a lower maximum depth than Bitcoin. However, this is because the maximum observed reorg depth on Ethereum was three blocks, or thirty-nine seconds of block-time, compared to one block on Bitcoin, equating to ten minutes. Coins that experienced deep reorg attacks are on the left of the plot showing multiple hours of block-time being reverted in their deepest attacks. Less block-time being reversed in the worst case does not necessarily imply that one coin provides more security than another in a shorter amount of time as the maximum observed depth says nothing about the cost of future attacks. This figure does however highlight the importance of considered block interval when comparing the impact of reorgs of varying depths between coins.

To determine whether a coin is attackable in practice by utilizing rented hashrate we compared the expected number of hashes required to produce each reorg with the available hashrate on Nicehash at the time. Figure 5.9 plots the average proportion of the required hashrate to produce each reorg that was available on Nicehash with a reference line included

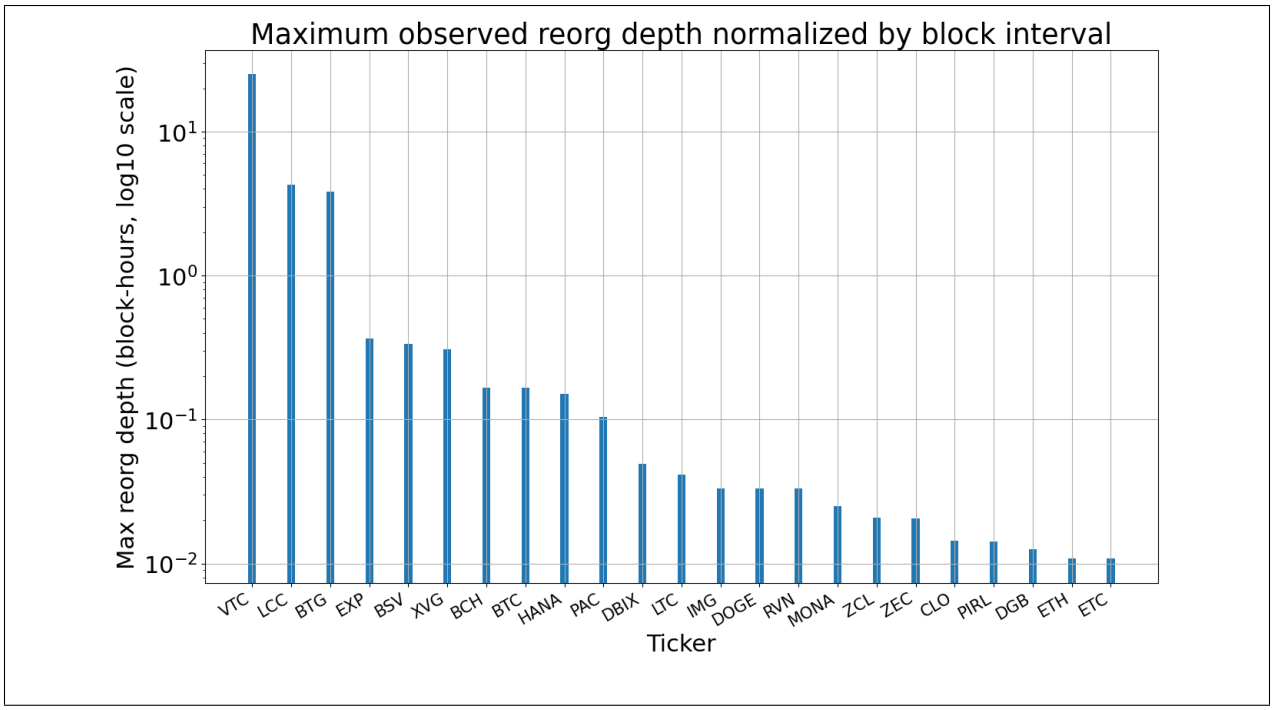

Figure 5.8: Plot of the maximum observed reorg depth for each coin normalized by block interval to hours of blocks.

at $y = 1$ to indicate 100% was available. The data shows that just under half of the coins we observed had at least enough hashrate available on Nicehash to generate the reorg, and over a quarter had at least ten times as much available. This suggests that some of the coins in our study are highly susceptible to future attacks using Nicehash even though they have not been attacked while under our observation.

If the liquid hashrate market model holds in reality, the only barrier to attack disregarding external factors is the initial cost of investment, which is recouped in full if the reorg is successful. Thus, Figure 5.10 plots the average reorg block reward per block-hour against the average value transacted in orphaned blocks per block-hour. Note that every data point is below the $y = x$ reference line, meaning that every coin we studied has a theoretical initial reorg investment cost less than its average transaction volume reverted in reorgs. This is sensible given that it would be very expensive to require every transaction to provide an equal value in fees for security, but raises the question of how Nakamoto consensus can withstand attacks in the liquid hashrate model without an additional mechanism. Nonetheless

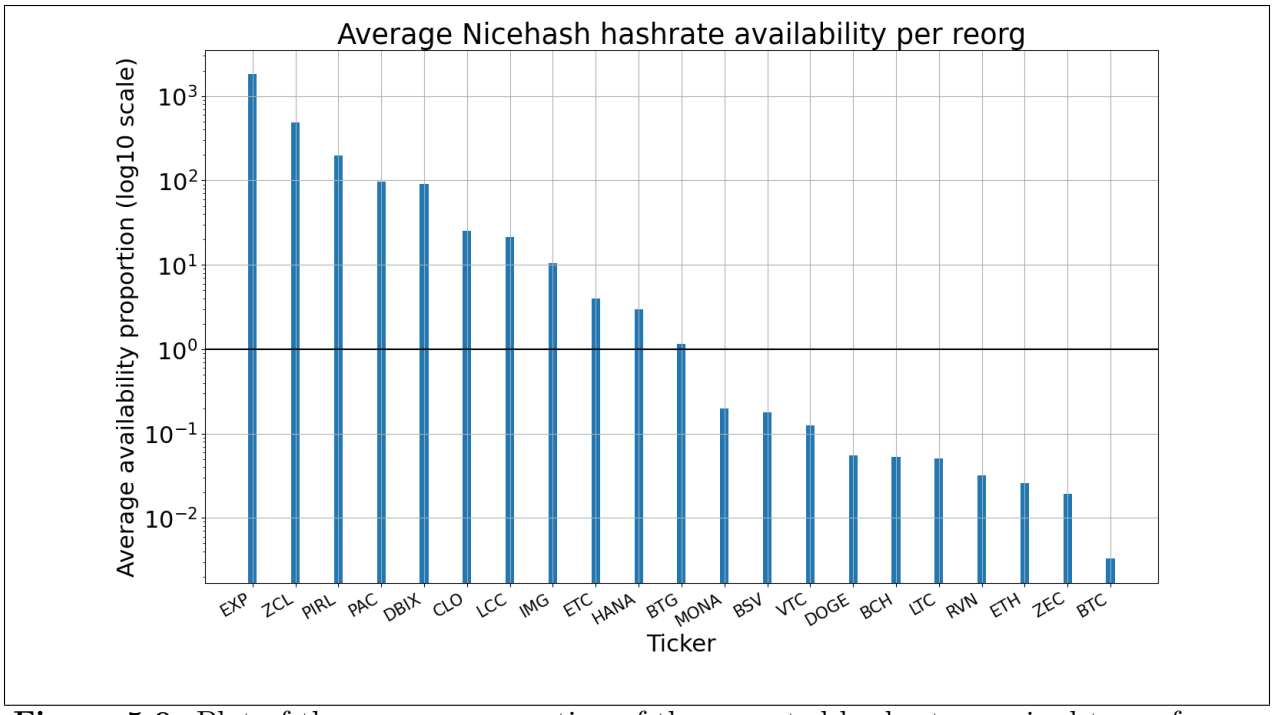

Figure 5.9: Plot of the average proportion of the expected hashrate required to perform a reorg that was available on Nicehash.

it is concerning that Bitcoin only compensates miners $\approx$ \$10 million to mine honestly rather than double-spend transactions in a system with $\approx$ \$1 billion in transaction volume in the same time period.

To overcome the exponential progression of block rewards and transaction volumes between coins, Figure 5.11 presents the information in Figure 5.10 as a ratio of transaction volume to block reward. In the liquid hashrate model this plot represents the "value for money" coins are getting for their security payments in terms of transaction fees and supply inflation. A subset of these coins experienced actual double-spend attacks, so it is interesting that there appears to be no correlation between coins that were attacked in practice and the efficiency of the system's security payments. Given that optimizing for a high transaction volume to block reward ratio seems like an obvious long-term goal for any scalable cryptocurrency, further investigation into how coins such as Dogecoin and Litecoin can achieve such high ratios without experiencing attacks would be useful to the industry.

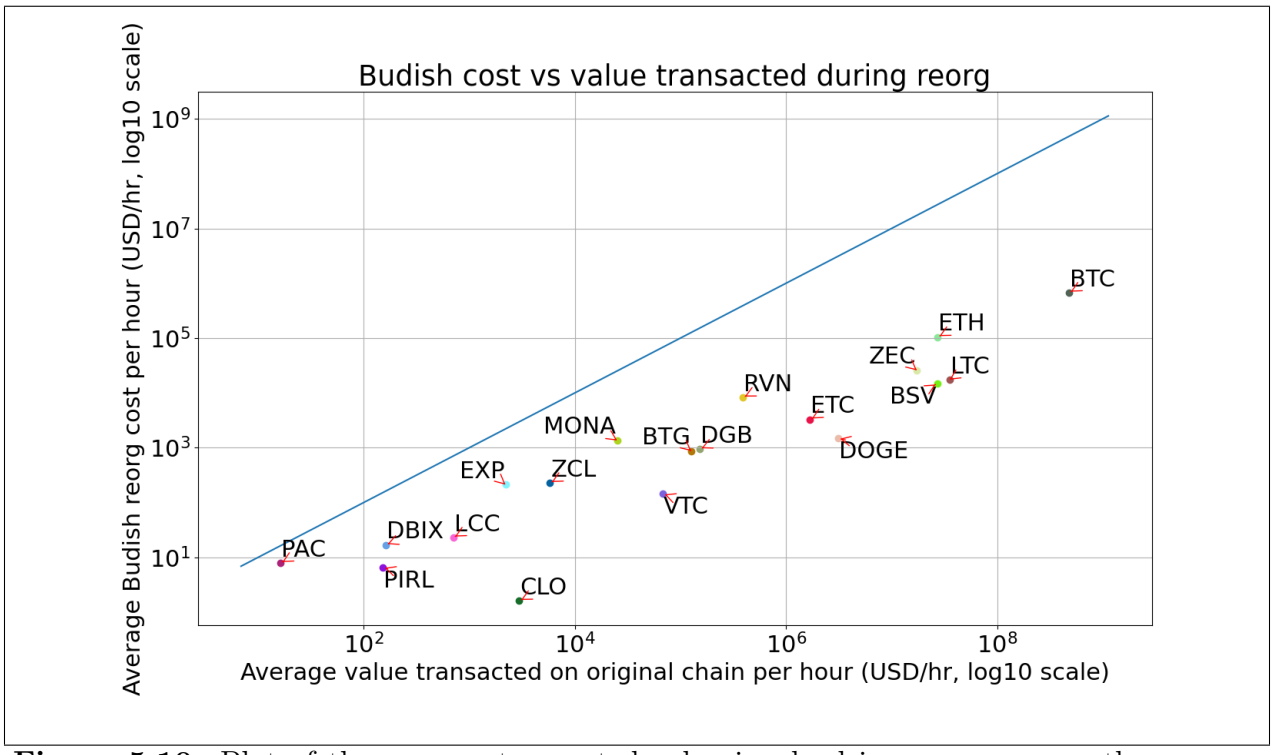

Figure 5.10: Plot of the average transacted value involved in a reorg versus the average Budish reorg cost.

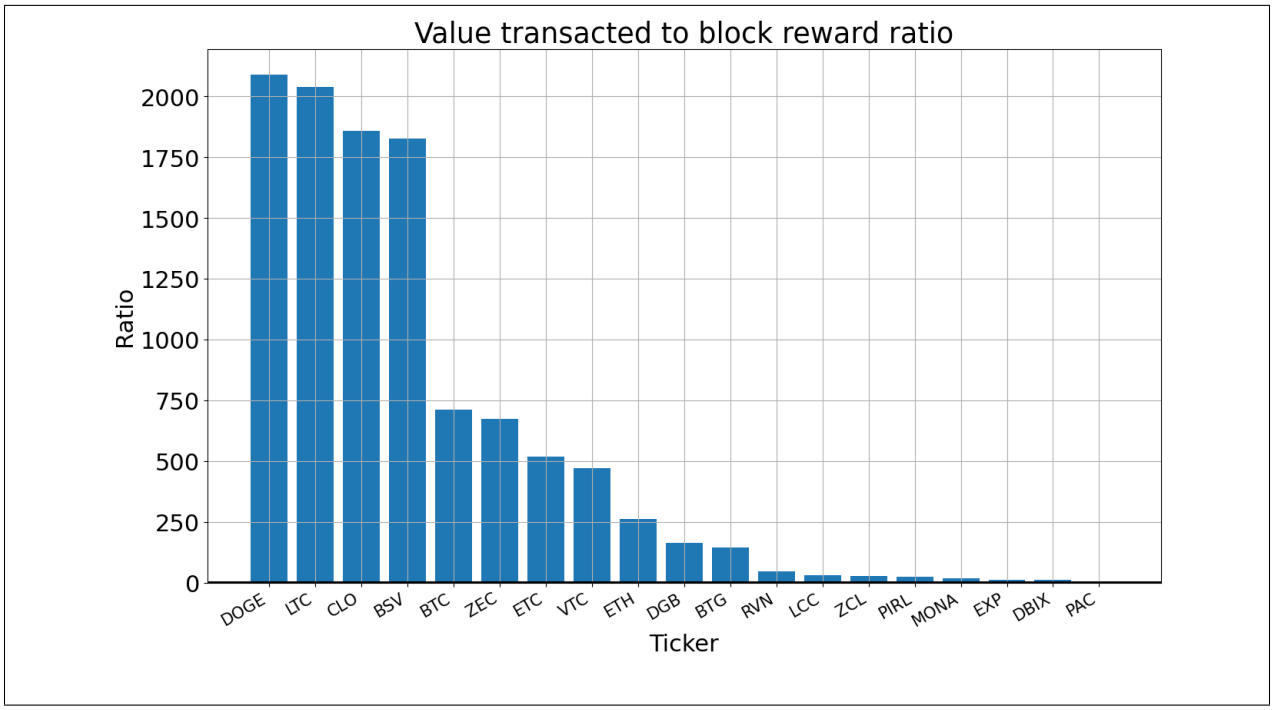

Figure 5.11: Plot of the average hourly reorg value transacted to block reward ratio.

5.2 Deep Reorg Attacks

The primary focus for this research was to detect and study deep reorg attacks in practice. We wanted to determine whether attacks happen in practice, the costs associated with attacking, the origin of attacks, how much value is stolen and how modifications to Nakamoto consensus affect attacks. Furthermore, we wanted to observe miner and user behavior in the face of attacks as well as how a coin's market price is affected. These two metrics are important to estimate the real-world values of P_{attack} and p_{reorg} for each coin which are the parameters used to calculate the security condition in equation 2.10. By studying the qualities of real-world attacks we investigated the accuracy of the theoretical attack models described in Section 2.2. We also perform a manual analysis of the double-spent transactions in each case and correlate the attacks with Nicehash market and coin price data.

5.2.1 Summary of Attacks

Our study detected 40 distinct deep reorg events of varying severity across 6 coins, 18 of which involved double-spent transactions. Table 5.2 summarizes the set of attack events that were detected. There is no strict threshold for labeling a reorg of a certain depth to be "deep", and we do not know the identities of the victims or how many blocks they waited before considering a transaction finalized. Therefore, we split deep reorg events into two categories, those that were at least 6 blocks deep and those at least 1 block-hour deep. Litecoin Cash, Vertcoin and Bitcoin Gold experienced reorgs at least 1 block-hour deep containing double-spent transactions. The details of these attacks are described in Subsection 5.2.3. We omit a detailed analysis of the attacks on Verge, Expanse and Hanacoin in this paper because their depth was borderline for the events to be significant considering their short block intervals. Additionally, Verge's deep reorg events contained no double-spent transactions. A description of Expanse's attack event is available in an external document [23].

Each of the coins that were attacked have low market capitalizations and correspond-

ing small block rewards, contributing to their low cost of attack. Bitcoin Gold, the coin with the highest market capitalization in Table 5.2 of under $200 million at the time of writing,[1] had an average deep reorg cost of under $1800. Clearly, relative to Bitcoin Gold's market capitalization and thus potential maximum transaction size, 1 block-hour of mining payments is dramatically insufficient insurance against a double-spend attack. The deepest attack was on Vertcoin, it was over 25 block-hours deep and cost only $\approx$ $3000. In each case we were the first source to publicly disclose the attacks and no victims or perpetrators stepped forward to claim association with the events.

Our data suggests that the attacks were likely to have been profitable or recovered the attacker's costs, even without the double-spends succeeding. Table 5.2 shows the average reorg return, calculated as the mean ratio between the block rewards and the estimated rental cost using Nicehash. It also shows the average availability of hashrate in the market as a proportion of the estimated hashrate required to generate the reorg. Hashrate availability was close to or exceeding 100% for each coin, suggesting that the coins were operating close to the liquid hashrate market model. Therefore, the theory that deep reorg attacks in a liquid hashrate market should be close to break-even appears to hold in practice.

The attack events either occurred in clusters with several distinct deep reorgs over a period of hours, or as an isolated event. Figure 5.12 plots the depth of each attack event in block-hours as well as the cumulative value of double-spent transactions against time. By the conclusion of the study, over $500,000 worth of coins had been involved in double-spent transactions. The largest contribution is from the attacks on Bitcoin Gold where multiple successive reorgs battled over the same group of conflicting transactions in a repeated counterattack game, described in Subsection 5.2.3.

The cumulative value of double-spent transaction outputs does not necessarily equal

[1]Source: `https://coinmarketcap.com/currencies/bitcoin-gold/`, retrieved 05/03/2020.

[2]Historical price data for Hanacoin was not available so reorg return analysis is omitted.

[3]Nicehash data for LCC's algorithm, SHA256, begins on 07/12/19, a week after the attack cluster. The values in the table use the closest available sample.

[4]Verge is a multi-algorithm coin for which we are uncertain how to accurately estimate rental mining cost, thus Nicehash analysis for Verge is ommited.

Table 5.2: Summary of reorg events where at least six blocks were reversed. "Double-spent Transaction Input Count" is the total number of UTXOs or account-nonce pairs that appeared in double-spent transactions. "Mean Deep Reorg Cost" is the average estimated reorg cost using the Nicehash hashrate rental price. "Mean Deep Reorg Return" is the ratio of the average reorg cost estimated using the Nicehash rental price to the average miner rewards. It represents the average estimated deep reorg profitability ratio with 1.0 being break-even. "Mean Nicehash Availability" is the average proportion of the coins' network hashrate that was available on Nicehash at the start of the reorg event.

Ticker	Deep Reorg Count ($\geq$ 6 blocks)	Deep Reorg Count ($\geq$ 1 hr)	Double-spent Transaction Input Count ($\geq$ 6 blocks)	Total Double-spent (USD)	Mean Deep Reorg Cost (USD)	Mean Deep Reorg Return	Mean Nicehash Availability
BTG	15	15	63	685613.36	1704.18	1.27	1.20
EXP	3	0	1	12.09	7.50	131.57	18305.91
HANA	1	0	28	151.86	0.83	- [2]	0.93
LCC	6	6	287	8868.50	74.79 [3]	0.77	2.67
VTC	1	1	5	29.50	3134.58	1.04	0.47
XVG	14	0	0	24547.17	- [4]	-	-

the total value that was actually stolen, as the same coins can be fought over multiple times. Furthermore, most transactions include change which is transaction volume that was never in possession of the victim but is nonetheless redirected as part of the double-spend. This is why manual analysis of the double-spent transactions is required to estimate the likely stolen value based on heuristics about typical transaction formats and address groupings.

5.2.2 Post-Attack Price Change

To determine real-world values of P_{attack} and to test the theory that a coin's market price should decrease significantly after a successful attack, we compared the price changes for each coin post-attack. Figure 5.13 shows the average proportional change in the coins' market prices at the start of the reorg compared to the time that the reorg occurred and an hour, day, week and month later. The plot illustrates that the assumption double-spend attacks

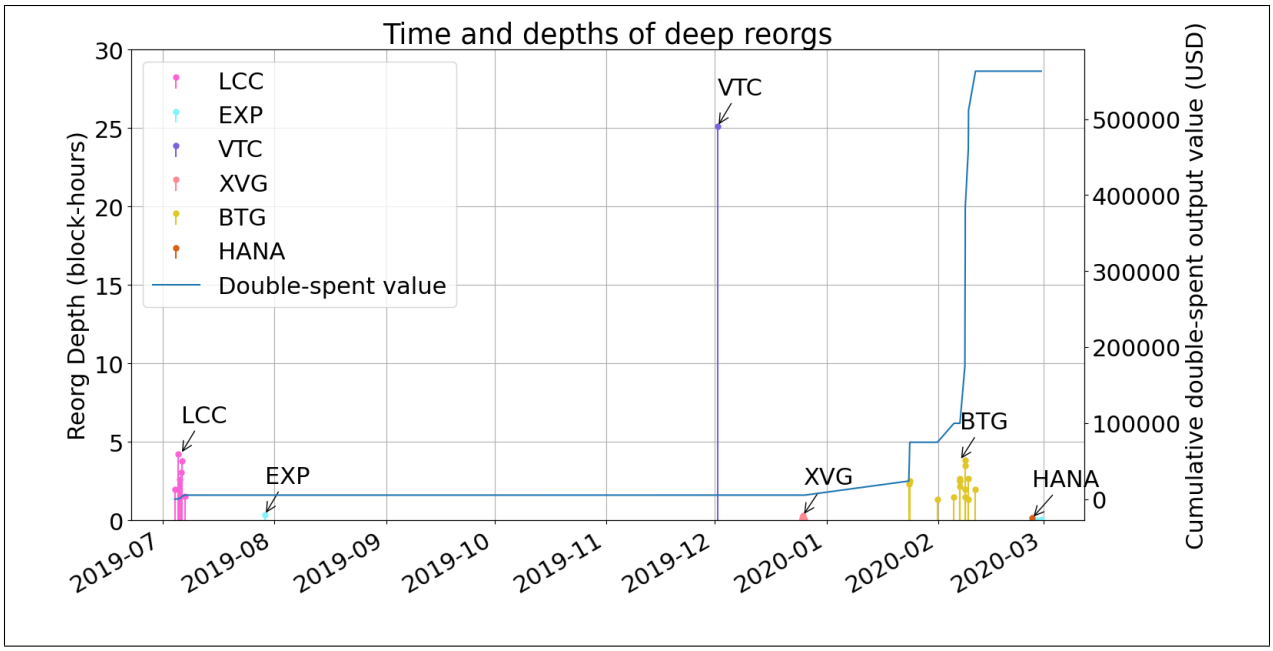

Figure 5.12: Plot of the depth and time of detected deep reorgs and the cumulative value of double-spent outputs.

will cause a significant decrease in a coin's market price is largely false.

Only Vertcoin's price decreased significantly after the attack in a way that could have affected the attacker's ability to sell their mined and double-spent coins at a profit. Even then, the price remained stable for at least an hour after the attack, meaning it is not unfeasible the attacker could find a buyer for their coins. Bitcoin Gold's price also decreased after a month, but increased for at least a week after the attack allowing the attacker to potentially make a profit from the block reward appreciation alone. The obvious out-lier is Expanse, which experienced a very large market movement at the time of the attack during which the price increased over seven times within a few hours.

From the attacks we detected, it is clear that P_{attack} does not have to be less than or equal to 1 as industry folklore would suggest. This means that according to equation 2.8, there is no transaction value that is safe from an incentive-compatible double-spend attack. Of course the expanded equation 2.10 contains the factor p_{reorg}, which would still allow for non-zero incentive-compatible transactions to be sent if the probability of the network accepting the reorg was sufficiently low. In practice, p_{reorg} was 1 for every attacked coin except

Bitcoin Gold in which there were counterattacks explained later in Subsection 5.2.3. V_{fixed} from equation 2.12 was zero for every attacked coin because each coin's mining algorithm was either abundantly available on Nicehash or mined using general purpose hardware.

We cannot reasonably predict how the values of P_{attack} and p_{reorg} would behave differently for more prominent coins such as Bitcoin or Ethereum. One could speculate that an attack on a more prominent coin would be more likely to lead to a loss in confidence, assuring $P_{attack} < 1$. Our attack disclosures on Bitcoin Gold and Vertcoin were covered by several media outlets [24][25], creating the possibility that the increased exposure from news about under-reported cryptocurrencies outweighed any loss of confidence due to the attacks themselves. Arguably this would have the opposite effect for more prominent coins if mainstream media reported on the attack as that would signal a general loss of confidence in Nakamoto consensus as a technology.

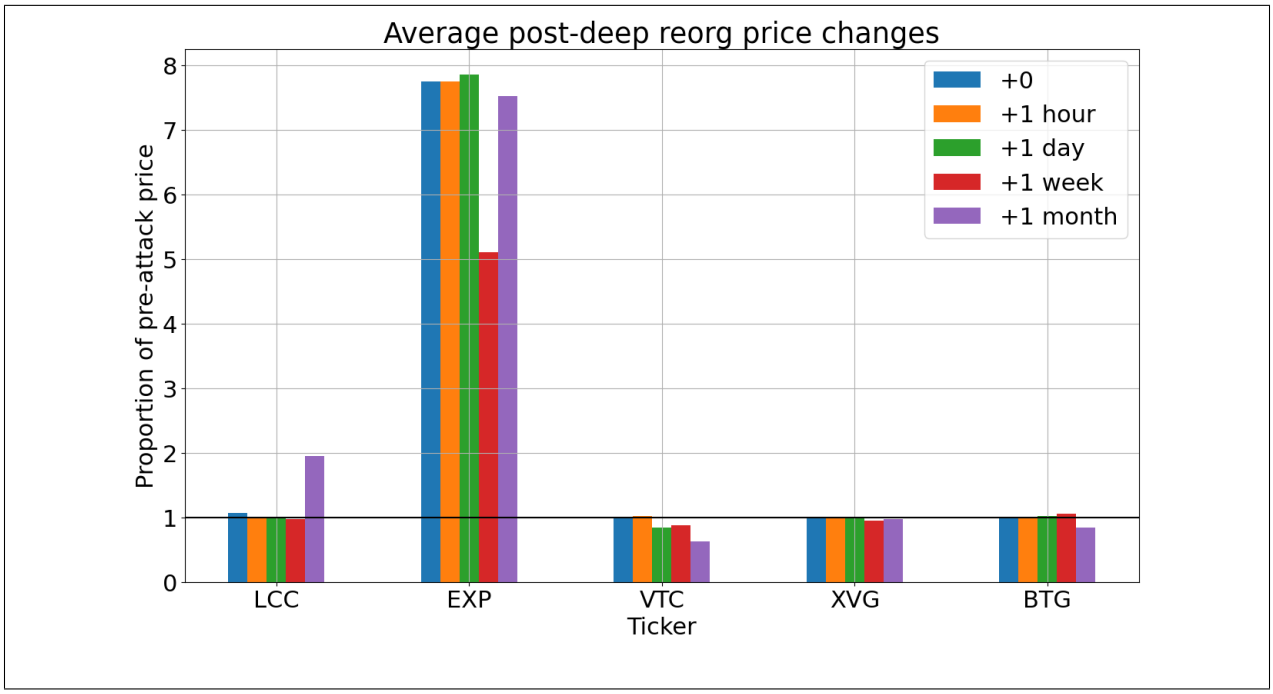

Figure 5.13: Plot of the average change in coin market price post-attack up to one month later.

5.2.3 Case Studies

Bitcoin Gold

Bitcoin Gold is a Bitcoin hard-fork that aims to be GPU-mineable by using an algorithm known as "Zhash." The Bitcoin Gold website claims Zhash "uses more memory than an ASIC can muster, but runs fine on many graphics cards".[5] Bitcoin Gold was previously 51% attacked in May 2018 when it was estimated that up to $18 million worth of BTG was double-spent [26]. In late January through early February 2020 we detected a series of deep reorgs in which an attacker and defender counterattacked each other to make their desired fork the primary chain. We disclosed the attacks in greater detail in a blog post that we released in March [27]. The discovery provides the first evidence that counterattacking may be a viable strategy for a victim to defend themselves from a rational adversary, as theorized by D. Moroz et al [12].

As described in the theory, we observed a repeated retaliation game in which an attacker initially caused a reorg that double-spent a transaction, but a new miner entered the system and extended the original fork, restoring it as the primary chain. Subsequently, the attacker continued extending their malicious fork causing a third reorg, restoring the double-spend. Finally, the defending miner further extended the original fork causing a fourth reorg, at which point the attacker capitulated and allowed the original fork to prevail. Days later we observed two additional counter-attack games in which the defender only had to counterattack once before the attacker gave up. In the repeated counterattack sequence, the parties were fighting over an estimated value of 4,490 BTG, or $\approx \$44,000$ at the time.

In each counterattack game, the defender eventually succeeded in deterring the attacker and restoring the original pre-attack fork. Both the attacker and defender mined their blocks using addresses that had not been previously used on the network. It is thus not clear who the intended target was, or if the attack was merely simulated where the attacker and defender were in-fact the same entity. It is unlikely that the phenomena oc-

[5] Bitcoin Gold website: https://bitcoingold.org/for-miners/. Quote retrieved 05/04/20.

curred due to a bug, network partition or random chance due to the deliberate construction of the double-spent transactions and the use of ephemeral miner addresses. Furthermore, the incumbent miners of the network can be seen in the reorg event trace blindly generating blocks referencing whichever fork had the most total chain work at the time.

Figure 5.14 plots the hashrate price and capacity for the ZHash Nicehash market during the deep reorgs on Bitcoin Gold. Both metrics are incredibly volatile at baseline, with the available capacity sometimes changing nearly 500% in a single day. This makes it difficult to attribute the hashrate used to generate either the adversary's or defender's chains as having originated from Nicehash. While the plot shows spikes in capacity occurring close to deep-reorg events, there are similarly large spikes that are unassociated with an event.

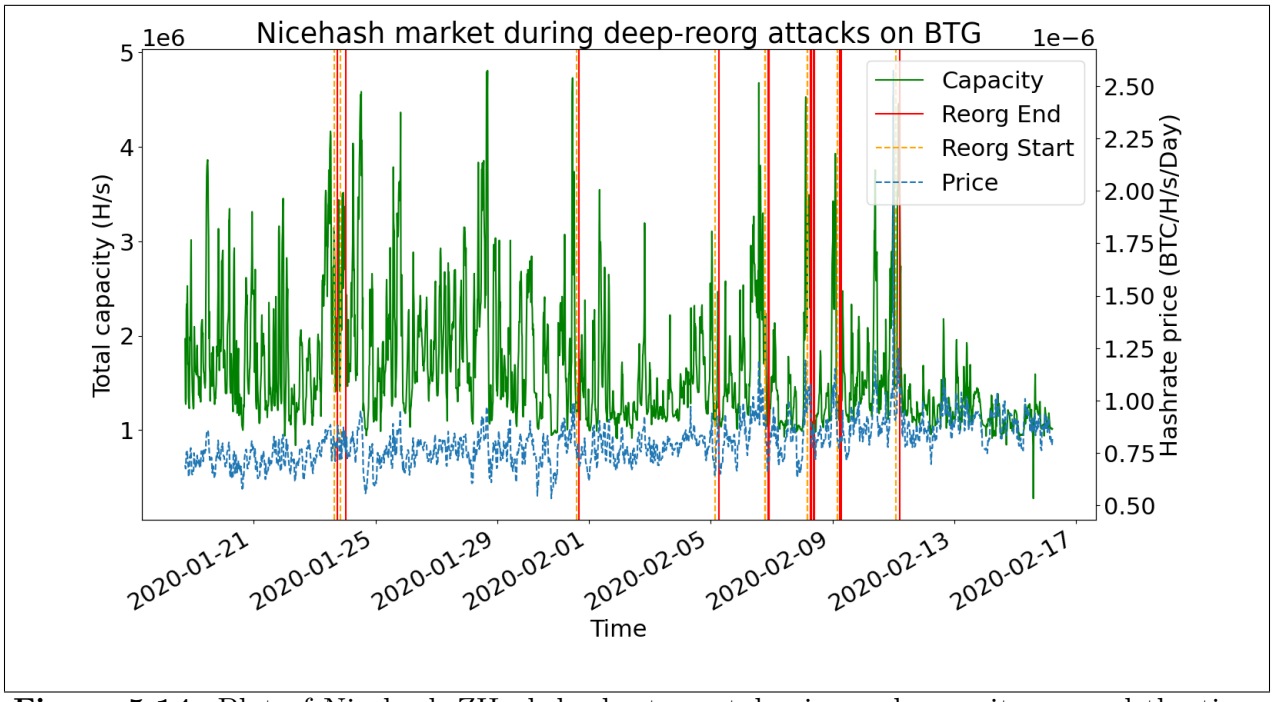

Figure 5.14: Plot of Nicehash ZHash hashrate rental price and capacity around the time of the deep-reorg attacks on Bitcoin Gold.

Vertcoin

Vertcoin is a Bitcoin Core clone that aims to be ASIC-resistant by proposing hard forks to new mining algorithms whenever ASICs are deployed on the network. Vertcoin was

previously 51% attacked in December 2018 [17] and has since changed its proof-of-work algorithm from Lyra2REv2 to Lyra2REv3 in February 2019 [28]. In December 2019 we detected a single 603 block deep reorg on Vertcoin, equating to over 25 hours of block-time reversed [29]. Only $\approx 125\ VTC$ or $\approx$ \$29 at the time was double-spent in the attack, though 13825 VTC or $\approx$ \$3600 in block rewards were redirected from the incumbent miners to the attacker. We estimated that the investment required to perpetrate the attack using Nicehash would have only been approximately \$3100 meaning that the attacker may have been able to make a profit based on diverted block rewards alone. We received forewarning that this attack would occur via a user who reported that their miner, connected to the Nicehash marketplace for Lyra2REv3, was receiving work for non-public Vertcoin blocks.

The user was receiving work from Nicehash to mine Lyra2REv3 blocks of a height greater than the height of the Vertcoin blockchain. This indicates that there was a non-public fork being generated using hashrate rented from Nicehash. At the time, Vertcoin was the only significant Lyra2REv3 coin, the other being Hanacoin with network hashrate that is two orders of magnitude smaller. Figure 5.15 shows the market conditions on Nicehash for the Lyra2REv3 algorithm at the time of the attack. There is a clear spike in hashrate price and available capacity coinciding with the start of the attack. Subsequently, both quantities return to baseline levels once the attack is over and the adversary's fork becomes the most work chain. Combined with the jobs for non-public blocks that Nicehash was sending to miners, the market activity provides strong evidence that Nicehash was used at least in part to carry out the attack.

Bittrex, the largest volume exchange for Vertcoin was requiring 600 confirmations, or 1.04 days of block-hours to credit deposits prior to the attack. Exercising maximal caution, we contacted Bittrex and recommended that they temporarily disable deposits in case there was an attack in progress. As it turned out, the deep reorg that occurred several hours later was over 600 blocks deep and thus could have defeated Bittrex's confirmation requirement. Therefore, a possible explanation for the lack of a significant double-spend is that the most

likely target prevented the adversary from trading or withdrawing the coins they deposited before the 600 block limit has elapsed.

An alternative explanation of course is that there was no victim, and the attack was only intended to steal from incumbent miners and cause disruption, or as a proof of concept. Bittrex did not indicate that any of the addresses involved in the attack belonged to them. The attacker had mined blocks previously to the reorg fork block and the inputs they used for the double-spent deposit transactions were from those prior coinbases. However, we are fairly certain that this was a deliberate attack and not simply a miner misconfiguration.

The construction of the original and replacement mutually exclusive transactions on both sides of the fork appear deliberate. The original deposit transactions were included 120 and 121 blocks after the fork block on the original chain. These were subsequently invalidated by a single transaction that swept all of the block rewards on the attacker's fork, ignoring the coinbase maturity delay, as well as the inputs to the deposit transactions. Since the deposit transactions were not deep enough in the original chain to be credited by Bittrex at the time of the reorg under normal circumstances, it is possible that the attacker originally intended the reorg to be deeper but stopped early once they knew the double-spend would not succeed.

Litecoin Cash

Litecoin Cash is a Bitcoin Core clone which uses a hybrid proof-of-work/proof-of-stake consensus algorithm in an attempt to alleviate 51% attacks on its network. Litecoin Cash's proof-of-work hash function is SHA256 but its network hashrate is many orders of magnitude smaller than Bitcoin's, making it highly vulnerable to 51% attacks. This was demonstrated in May 2018 when the exchange YoBit reported that Litecoin Cash had been 51% attacked [31]. The LCC whitepaper describes a system they call "Hive Mining", which is effectively a proof-of-stake lottery in which users can purchase "bees" (lottery tickets) that have the potential to be eligible to propose a proof-of-stake block for each new proof-of-work block

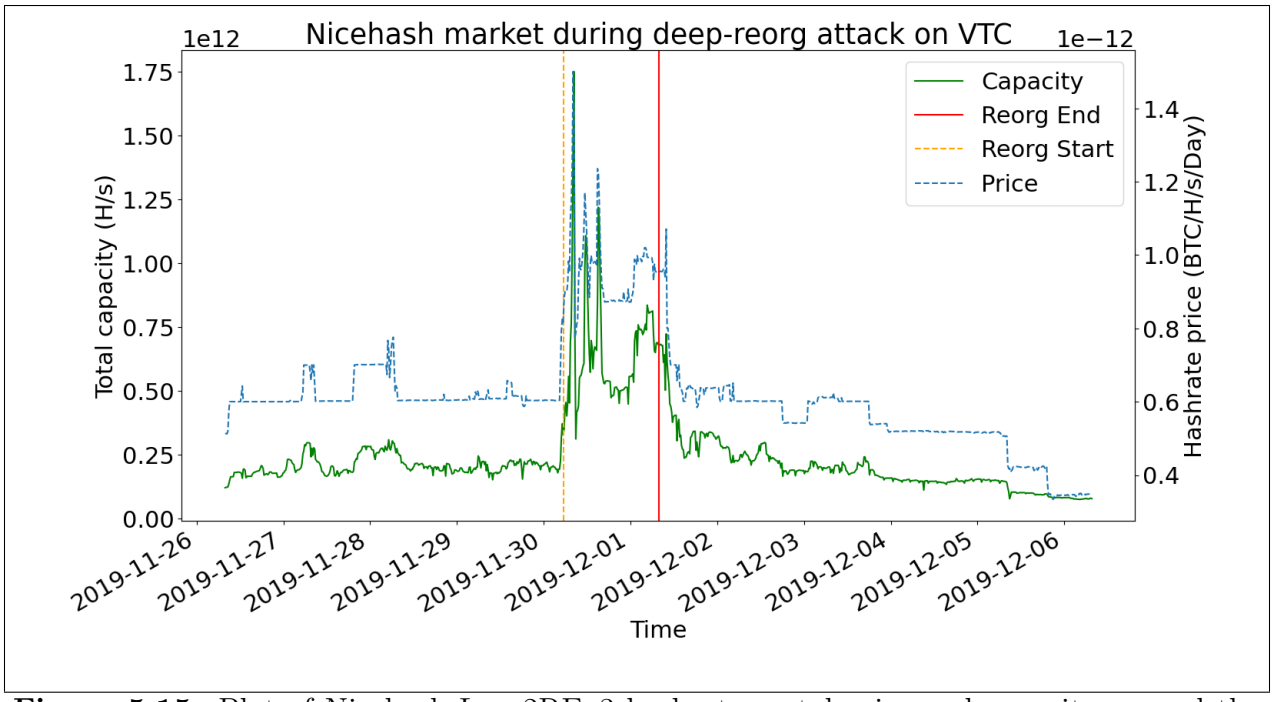

Figure 5.15: Plot of Nicehash Lyra2REv3 hashrate rental price and capacity around the time of the deep-reorg attack on Vertcoin.

[8]. The authors claim this scheme provides protection from 51% attacks by interlacing both types of blocks, and giving proof-of-stake blocks more relative weight than proof-of-work when determining the fork with most total chain work.

Between July 4th and 7th 2019 we detected a series of 6 distinct reorg events on Litecoin Cash during each of which over 1 block-hour was reversed [30]. The deepest reorg of 102 blocks reverted over 4 hours of block-time. In total we estimated that over $\$5,500$ could have been stolen due to double-spent transactions, in addition to the block rewards from incumbent miners, totaling approximately $350. As with the other attacks we have discussed, we do not know the identity of either the attacker or victim.

The series of events is particularly interesting because of the mechanism of attack. To defeat the Hive mining algorithm, the attacker simply neglected to mine any proof-of-stake blocks, making up the difference in total chain work by mining higher difficulty proof-of-work blocks. This resulted in reorgs with an original fork that contained proof-of-work and proof-of-stake blocks interlaced with each other, and a replacement fork containing only proof-of-

work blocks. This was likely possible because the original Hive mining implementation only included a modest increased weight for proof-of-stake blocks and imposed no restriction on the number of sequential proof-of-work blocks there could be.[6] Therefore, it was trivial for the attacker to overcompensate with hashrate to overcome the deficit in total chain work and cause a reorg by simply mining with a greater speed than the incumbent miners plus equivalent hashrate contributed by proof-of-stake blocks. Since the attack, the Litecoin Cash developers have deployed Hive 1.1 which introduces a requirement that all proof-of-work blocks must follow a proof-of-stake block, disabling this specific attack mechanism [32].

The sequence of double-spent transactions across successive reorg events appeared intended to drain an exchange's hot wallet by repeatedly depositing coins before double-spending them. The attacker included the deposit transaction for the double-spend due to the subsequent reorg in their replacement blocks for the previous reorg, requiring the attacker to replace a subset of their own blocks in sequential reorgs. For each double-spend, the original recipient and the address the deposited coins were swept to in the replacement transactions were the same, indicating the same attacker and victim for each event.

5.3 Shallow Double-Spends

Included in the thousands of shallow reorgs we detected on Ethereum, we also observed a significant number of double-spent transactions. Of course, none of the reorgs we observed on Ethereum were greater than 3 blocks deep, or just 39 seconds of block-time. This means that these transactions should not be interpreted in the same way as double-spends that were part of a deep reorg because an exchange, merchant or user should not reasonably consider the transaction to be finalized. The vast majority of the double-spent transactions were zero-value in terms of the base currency (ETH) and instead called functions on smart contracts.

[6] Litecoin Cash block work function at the time of the attacks: `https://github.com/litecoincash-project/litecoincash/blob/adcfb6f9437a89f212fa9341d453ae1d4809ace7/src/chain.cpp#L123`.

Reorgs contained mutually exclusive transactions that spent the same account-nonce pair and often used the same destination contract address in the replacement transaction but with different parameters. Therefore, we believe that these mutually exclusive transactions in low-depth reorgs could be evidence for front-running attacks against smart contracts in practice.

Over the 38970 reorg events that we detected on Ethereum, there were 11190 double-spent account-nonce pairs, leading to an average of 0.29 conflicting transactions per reorg. This phenomena is unique to Ethereum, with the other Ethereum-like coins we monitored reporting negligible or zero shallow double-spend events. The mutually exclusive transactions called functions on a common set of contract addresses, the 20 most frequent of which are listed in Table 5.3 along with the contract name if known.[7] A number of tokens are included in the list such as Tether, Tellor Tributes and CpcToken, games such as MyCryptoHeroes, and decentralized exchanges such as 0x and dYdX.

Pairs of mutually exclusive transactions can be sorted into one of four broad categories: spending to different accounts, spending to the same account with a different value, changing only the input data and changing only the gas price. The percentages of overall conflicting transactions accounting for each category are 23%, 3%, 22% and 51% respectively. While half of the events can simply be ascribed to fee bumping which is a normal wallet behavior, the remainder are potentially more concerning. Changing the account the original transaction sends to would allow an adversary to cancel the call to the contract, reversing its effects on Ethereum's state. Similarly, changing the value or input to the contract call would allow the adversary to change order parameters issued to decentralized exchanges or the destination and value of a transfer within a token contract.

Since an analysis of smart contract execution was out of scope for this research, we do not know specifically how each double-spend would have affected Ethereum's internal state, account balances or trading outcomes. That being said, it is certain that a different state

[7] Contract names retrieved 05/05/20 from `https://etherscan.io/`

Table 5.3: List of the twenty most common Ethereum smart contract addresses experiencing double-spends in shallow reorgs.

Contract Address	Number of events	Contract Name
0xa57bd00134b2850b2a1c55860c9e9ea100fdd6cf	603	Unknown Contract
0xdac17f958d2ee523a2206206994597c13d831ec7	534	Tether USD
0x8018280076d7fa2caa1147e441352e8a89e1ddbe	423	Unknown Contract
0x39755357759ce0d7f32dc8dc45414cca409ae24e	396	MatchingMarket
0x1e0447b19bb6ecfdae1e4ae1694b0c3659614e4e	288	SoloMargin
0x9ba6c9d09db964771cb4d526188f6af81e9bcf9e	264	Unknown Contract
0xfb80bfa19cae9e00f28b0f7e1023109deeb10483	233	ConversionRates
0x860bd2dba9cd475a61e6d1b45e16c365f6d78f66	182	Unknown Contract
0x798abda6cc246d0edba912092a2a3dbd3d11191b	145	ConversionRates
0x85c5c26dc2af5546341fc1988b9d178148b4838b	136	Unknown Contract
0x0ba45a8b5d5575935b8158a88c631e9f9c95a2e5	133	Tellor Tributes
0x2a842bc64343fad4ec4a8424ba7ff3c0a70b6e55	126	SignedOperationProxy
0xce5702e94ad38c061e44a4a049cc29261f992846	123	Unknown Contract
0x8fdcc30eda7e94f1c12ce0280df6cd531e8365c5	116	CpcToken
0x5e07b6f1b98a11f7e04e7ffa8707b63f1c177753	114	MCHDailyActionV3
0x8a91c9a16cd62693649d80afa85a09dbbdcb8508	106	Unknown Contract
0x080bf510fcbf18b91105470639e9561022937712	104	0x Exchange v2.1
0xb958a8f59ac6145851729f73c7a6968311d8b633	81	Unknown Contract
0xeb91a27e79bddbac9569d512f15064c8466883da	78	Unknown Contract
0x0559324025d6ef4a715c2e22562b90210fcfe25a	78	ConversionRates

would have resulted depending on which side of the reorg fork ultimately became part of the primary chain. It is unclear whether the reorgs and mutually exclusive transactions are directly linked, with some of the reorgs being perpetrated by an attacker in order to reorder or change contract function calls.

An alternative explanation is that client software associated with the smart contracts does not use strictly incrementing account nonces for successive transactions, instead querying the Ethereum state for the most recently used account nonce. In the case of a reorg, client software may believe a transaction has been unsuccessful, and decide to replace it with an updated transaction with different parameters. Alternatively, the reorgs and conflicting transactions could be unrelated, with an adversary hoping to replace existing function calls

using fees to incentivize miners to include an alternative transaction should there be a randomly occurring reorg, with some probability of success. Given the remaining uncertainty about the nature of this phenomenon, we believe it warrants further study in the future.

Chapter 6

Double-Spend Attack Mitigations

Here we discuss some of the potential strategies that victims of double-spend attacks could implement. These range from simple local actions such as halting further transactions to using economic clout to influence which chain users select or bribing miners to reverse the effects of reorg. We implemented these policies as modules for the reorg tracker that can automatically take actions in the event of a malicious reorg.

6.1 Halt Operations

Given real-time detection of reorg events, we implemented several automated strategies that could help mitigate further damage to an exchange or merchant. An attack could occur at any time and failure to react immediately could result in a now-insolvent exchange continuing to credit deposits and issue withdrawals for some time. At a minimum an exchange should cease operations with the attacked cryptocurrency until they are certain any damage can be mitigated. Therefore, one of our automated policies is to immediately panic and halt the coin daemon in the case of a reorg deeper than a user-configured safety threshold. This would give a system administrator time to become aware of the attack and analyze their system for damage while remaining protected from further undefined behavior due to the consistency of the underlying cryptocurrency being broken.

Halting the coin daemon in the events of a deep reorg could be dangerous for the coin's liveness. If a large number of nodes implemented this policy, an attacker could cause nodes to shut down by launching a deep enough reorg, preventing nodes from communicating with each other to relay new blocks. This could potentially disconnect parts of the network, and nodes that did not shut down would continue operating, perhaps following a chain that will later be rejected after manual intervention by users implementing the policy. During this time, new nodes may struggle to find online peers from which they can download the chain, leaving them more susceptible to an eclipse attack where an adversary sends the new nodes an alternative chain with less total work than the primary chain.

6.2 Economic Confirmations

As we have demonstrated earlier in this paper, for many coins it is unsafe to assume that greater than half of the coin's network hashrate is unobtainable for an adversary. In these cases, potential victims should set their confirmation time for a coin to be based on the expected investment an adversary would have spend up-front in order to carry out the attack. By selecting a security parameter n representing the minimum investment in dollars for an adversary to launch an attack, we propose the following equation to set the confirmation time z in number of blocks:

$$z \geq \frac{3600 * n}{min(C_{budish}, C_{nicehash}) * C_{capital} * t_{interval}} \tag{6.1}$$

C_{budish} is the coin's average Budish cost in dollars per block-hour, and $C_{nicehash}$ is the estimated mining cost on Nicehash in dollars per block-hour. $C_{capital}$ represents the estimated cost of borrowing for the adversary such that a 3% interest rate would equate to $C_{capital} = 1.03$. $t_{interval}$ is the coin's average block interval in seconds. By selecting the minimum estimated reorg cost between Nicehash and the Budish method we opt for the worst-case scenario where the adversary is able to acquire hashrate at the cheaper of both

prices.

Plugging in the experimental numbers we calculated for Vertcoin assuming a security parameter of 1 million USD invested and a borrowing premium of 3%, we acquire $z \geq \frac{3600*1000000}{132.39*1.03*150} \approx 176000$. Clearly, this is an incredibly large and impractical delay equating to over 300 days of block time, but it highlights how little security can actually be provided by coins that are unable to heavily compensate miners. Note that the security parameter in this equation is only measuring initial investment cost for the adversary. If the attack succeeds and the adversary's fork remains in the most work chain, the adversary still recoups their investment in full. Using this equation would thus be ineffective against an economically irrational attacker, or an attacker without budget constraints.

6.3 Fixed Reorg Depth Limit

When dealing with less prominent coins, exchanges have greater scope for mitigation strategies. For many coins a single exchange is the dominant economic actor within the system because overall trading volume for the coin is low and traders gravitate towards the exchange with the most liquidity in order to get the best prices. This makes the exchange far more powerful in influencing the state of the system than if they had multiple competitors. Another of our policies implements a fixed reorg depth limit. This strategy would prevent a node from switching to an alternative chain that would cause a reorg deeper than a user-provided limit. This effectively finalizes blocks that are deeper in the chain than the configured limit. An exchange would then be able to continue using its original primary chain locally even if the rest of the network has switched to an alternative in which the exchange has been double-spent. Assuming the exchange has sufficient influence over the coin's users due to its economic importance, users could eventually be encouraged to switch and use the exchange's preferred chain in order to continue using the exchange's services. Furthermore, miners will eventually need to sell their coins in order to pay for real-world electricity costs or rental

payments in Bitcoin, limiting their scope to hold-out against the exchange's wishes unless there is a competing exchange available. This policy has already been deployed by some of the coins included in this research such as Ravencoin[1] and Bitcoin Cash.[2]

This policy departs from Nakamoto consensus rules as the order in which blocks are received by a node becomes a determining factor in selecting the primary chain. Selecting the primary chain in Nakamoto consensus is independent of block receipt order and only depends on which valid tip block has the most total work. This means that nodes observing the same blocks in a different order may select different chain tips, giving scope for an adversary to cause a chain split that is unresolvable without manual user intervention. If an adversary can accrue enough hashrate to generate two secret chains in parallel, they could wait until both chains are longer than the reorg depth limit and broadcast each chain to two different nodes. Neither node will ever reorg to the other node's fork as doing so would violate the fixed reorg depth limit. Therefore, the nodes will never converge to a unified chain no matter how much additional work is added to either fork.

6.4 Precious Addresses

Nakamoto consensus assumes nodes will always follow the most total work chain they are aware of. Instead, we implemented a policy that selects the current chain fork based on which is most valuable to the user. By providing a list of "precious addresses", the policy analyzes reorg events determine which chain results in the most coins belonging to the precious addresses. In the event of a reorg deeper than a given threshold, the policy analyzes both sides of a fork and selects the most valuable side. Exchanges and merchants could set the addresses to be for deposits, and miners could set them to be their payout addresses for receiving block rewards. This strategy would allow incumbent miners not to immediately switch to an alternative chain fork where they lose all their block rewards and end up helping

[1] `https://github.com/RavenProject/Ravencoin/issues/619`
[2] `https://blog.bitmex.com/bitcoin-cash-abcs-rolling-10-block-checkpoints/`

the attacker to solidify their alternative chain by adding more work to it.

6.5 Whale Transactions

As an extension to this policy, we implemented a variant of whale transactions [5]. When the precious addresses policy is activated due to a double-spend event, the policy generates a list of bribery transactions that spend from the double-spent transactions. The bribery transactions are successively time-locked in a chain and make use of transaction fees to compensate miners. There are two parameters, the proportion of the double-spend value to be offered as a bribe p, and the number of blocks to spread the bribe over N.

The first transaction t_1 spends the outputs from the double-spent transactions that were initially destined for the precious addresses with cumulative value v. Every transaction $t_i \forall i \in \{1, N-1\}$ has one output time-locked for 1 block with value $v(1 - \frac{p(N-i)}{N})$ spendable by the policy user. t_i spends from t_{i-1}'s time-locked output for $i > 1$. Since there is a difference of $\frac{vp}{N}$ between the input and output values in the bribery transactions, the miners can include these as transaction fees in their coinbase outputs.

The miners can retrieve the bribery transactions by subscribing to an API endpoint and include them in their blocks to claim the over-sized transaction fees. The bribery transactions increase the expected value of mining on top of the original chain containing the double-spent transactions. If the double-spend value is large enough that the bribe can exceed the expected value of mining on top of the attacker's fork, it would be rational for incumbent miners to switch back to the original fork. If the attacker is budget limited and cannot maintain a majority of the hashrate forever, this policy can help miners and double-spend victims to coordinate and drain the attacker's budget.

The consecutive time-locks are required to prevent miners competing over a large transaction fees that greatly outweigh the expected value of mining future blocks. This would prevent the original fork from progressing and increasing in total chain work, making

it unlikely to overtake the attacker's fork. By spreading out the reward over multiple blocks, the miners have many opportunities to receive a payout from the bribe, discouraging them from fighting over a single bribery transaction. The time-lock prevents the miners from including more than one bribery transaction in a single block, forcing them to mine multiple blocks.

If the additional subsidy is large enough, miners will be incentivized to work together and grow the chain that will eventually result in a larger profit for them. If the attacker chooses to switch to the original chain and claim the bribe or another entity rents hashrate to mine in secret and claim the entire bribe, this could still be beneficial to the policy user. In this case they still get some proportion of their double-spent transaction back, even if the incumbent miners lose out on their block rewards.

6.6 Implementation

Both the fixed reorg depth limit and precious addresses policies risk causing a chain split if the user is not influential enough in the network to ensure their chosen strategy persists and is subsequently replicated by other actors. They should therefore only be used with extreme caution and are more intended to initiate discussion on alternatives to the default most-work chain rule. Both policies can be implemented today by issuing a series of *invalidateblock* RPCs to nodes based on sufficiently recent Bitcoin Core versions. They do not currently work on Geth-based cryptocurrencies (which do not support *invalidateblock*) and coins based on older Bitcoin Core versions. Retrieving block data from third party APIs also prohibits more complex reorg policies as one has no control over which set of blocks are considered the primary chain. Note that none of the policies described above require any kind of hard or soft consensus fork and could all be implemented locally by individual users.

Chapter 7

Future Work

Although the reorg tracker design presented in this paper was able to detect a large number of reorg events, there are situations in which it would not detect a reorg, and it is unable to detect failed or in progress reorgs. Therefore, an important item of future work is to re-implement the reorg tracker to use cryptocurrency network messages to gather timing data and store all blocks relayed to the observation node. This would enable more fine-grained analysis of the overall state of the chain at any given point in time rather than only recording changes when reorg events are detected. A further design improvement would be to distribute multiple observation nodes widely throughout the Internet. This way, relative block arrival times between observation points could be compared to determine how a node's view of the chain tree changes based on time, geographical location and set of peers. With access to samples from more observation points, it would be possible to analyze what proportion of the network was affected by a particular reorg and determine whether reorgs were caused by network partitions or delays.

The double-spend analysis module of our system was unable to evaluate the result of transactions that had side-effects, making it impossible to fully understand the implications of double-spends on Ethereum-like coins. While it is trivial to detect two transactions that are mutually exclusive, determining how account balances or contract states would

differ depending on which transaction is included in the primary chain is only possible by executing them and comparing the resulting states. Given Ethereum transactions can have side-effects, the result of a transaction can depend on the other transactions and their order in a sequence of blocks. Therefore, mutually exclusive transactions cannot be executed in isolation to understand their effect in a reorg. Instead, both sequences of original and replacement blocks must be replayed based on the state immediately after the common fork block, after which the two resulting states can be compared for differences. An updated double-spend analysis module would thus have to include a method to evaluate Ethereum blocks and generate a resulting state, perhaps by directly integrating the Ethereum Virtual Machine (EVM) or by modifying the go-ethereum software to allow for manual chain tip selection via RPC.

Prior to the attack we detected on Vertcoin, we received forewarning from a Nicehash miner that an attack might be in progress. It would therefore be useful to continuously sample the work Nicehash distributes to its constituent miners in order to detect potential attacks in advance. This would require mining on Nicehash, maintaining hardware for each algorithm to be monitored and recording the stratum jobs sent by the server. By comparing the block header hashes being issued by Nicehash as the previous block hash (that miners reference in newly generated blocks) with the current primary chain tip hash for each coin, one could detect non-public blocks being mined. Furthermore, it would be possible to estimate the current depth and length of the alternative chain while it is being built, conclusively attribute attacks to rented hashrate and provide an automated early warning system for users.

Given this research demonstrated that counterattacking may be a viable real-world deterrent against 51% attacks for some coins, our work on reorg policies could be expanded to design a system for counterattack coordination between potential victims and miners. Although our code allows a single victim to issue bribery transactions if one of their addresses suffers a double-spend attack, there is currently no widespread monitoring for these

transactions by miners. Miners are also not yet equipped to automatically include the bribes and switch back to extending the original fork in the event of a reorg. While victims await the strategy of miners to grow in sophistication, victims could exploit the hashrate liquidity that Nicehash provides, playing the attackers at their own game. If potential victims were able to collectively pool funds as insurance against attacks, one could design a system that automatically rents hashrate from Nicehash to perform counterattacks as a response to reorgs containing double-spends.

In this research, we omitted Nicehash reorg cost calculations for multi-algorithm coins, Verge and Digibyte, because it is unclear how their difficulty readjustment algorithms and total chain work calculations behave under different hashrate distributions over the mining algorithms. For a single algorithm coin the calculation for expected number of hashes required to generate a chain of blocks simply uses the difference in chain work. For multi-algorithm coins, each algorithm has a varying contribution to the total chain work, such that an algorithm with half the hashrate of another still has equal weight, breaking the one-to-one relationship between total chain work and expected number of hashes. Understanding how to estimate the possible distributions of hashrate for each algorithm required to produce a given amount of total chain work would allow a realistic estimated Nicehash reorg cost to be calculated.

Our calculation of estimated Nicehash reorg cost was based upon the hashrate spot price for the coin's algorithm at the time the fork block was generated. Given the hashrate market is not infinitely elastic, it would not be possible to rent an arbitrary speed at the spot price as was assumed by our analysis. In order to calculate a more accurate reorg cost, one would need to understand bidder behavior in the Nicehash marketplace to determine the optimal order placement strategy and how much hashrate it can deliver at a given price per hash. Furthermore, given that some of the algorithms are mined using general purpose hardware, miners can reallocate themselves between algorithms in order to maximize profits. This means that if a renter places a bid with a high enough price, they are able to

increase the available hashrate in the market for a particular algorithm, while potentially decreasing supply for another, less profitable market. Research to quantify the market depth for algorithms on Nicehash and the influence renters can have on hashrate supply would enable more accurate attack feasibility and pricing.

It would be interesting to combine reorg data with address ownership information to determine the identities of victims and adversaries. We were unable to attribute any of the addresses involved in the attacks we detected to specific entities, meaning that do not know if exchanges have lost customer funds due to the attacks. Therefore, despite being able to detect deep reorgs in real-time, we are still reliant on exchange honesty to learn how many coins have actually been stolen in reality. Address identities would also reveal the chain of custody for any stolen funds or mining rewards post-attack and potentially the source of funding for the initial deposit transactions that are double-spent.

Chapter 8

Conclusion

Once considered unrealistic, we have shown that 51% attacks are now an unavoidable reality for cryptocurrency stakeholders. High-value targets like exchanges must now seriously consider the consensus risk associated with coins they list to protect themselves from double-spend attacks that appear to be profitable in a large number of cases. It is no longer sufficient for exchanges to set arbitrary settlement delays for deposits on each coin and must instead take a data-driven approach that appropriately manages the risk for their expected deposit volume. In many cases this will require exchanges to make difficult decisions for some coins about whether to inconvenience customers with extremely long waiting times, severely limit their daily deposit limit or de-list the coin entirely. Given the evidence from smaller coins that double-spend security is weak in a liquid hashrate market, stakeholders in larger coins should prepare for a potential future where rental mining is a realistic attack vector, or transaction fees fail to compensate for declining mining subsidies.

Honest miners and potential victims must become more advanced in their strategies if they continue operating with vulnerable cryptocurrencies. We demonstrated that counterattacking, previously a theoretical concept, is a tool that incumbent miners and double-spend victims should use to defend themselves from attacks and deter future adversaries. More generally, the industry should build upon our reorg tracker design and invest in reli-

able monitoring infrastructure to rationalize the non-negligible consensus risk that exists for cryptocurrencies using variations of Nakamoto consensus. Without reliable, real-time data on the investment cost of attack for a cryptocurrency, claims made about security guarantees are hearsay when economic incentives are considered and do not scale to the transaction values coins can support, currently bound only by a coin's supply.

If cryptocurrency is truly to become a reliable and widespread asset class, the industry must admit that its understanding of economic security against double-spending is lacking and make efforts to build trust that widely-held assumptions actually hold in practice. This paper challenges the idea that the price of a coin should drop significantly as a result of a successful 51% attack, a key requirement of the current theoretical model for Nakamoto consensus' economic security. Lack of reliable reporting about attacks prevents markets and stakeholders from reacting to events in a rational way, making effective detection and disclosure essential future components of the cryptocurrency industry. Armed with this information, users will be able to design effective attack mitigation strategies while converging towards using coins with stronger security.

Bibliography

[1] S. Nakamoto.
Bitcoin: A Peer-to-Peer Electronic Cash System.
https://bitcoin.org/bitcoin.pdf, 2008.
Retrieved 01/21/20.

[2] E. Budish.
The Economic Limits of Bitcoin and the Blockchain.
https://faculty.chicagobooth.edu/eric.budish/research/Economic-Limits-Bitcoin-Blockchain.pdf, 2019.
Retrieved 01/21/20.

[3] Hasu, J. Prestwich, B. Curtis.
A model for Bitcoin's security and the declining block subsidy.
https://uncommoncore.co/wp-content/uploads/2019/10/A-model-for-Bitcoins-security-and-the-declining-block-subsidy-v1.01.pdf, 2019.
Retrieved 01/21/20.

[4] M. Palatinus.
Stratum Mining Protocol.
https://slushpool.com/help/topic/stratum-protocol/, 2012.
Retrieved 01/21/20.

[5] K. Liao, J. Katz.
Incentivizing Blockchain Forks via Whale Transactions.
https://www.cs.umd.edu/~jkatz/papers/whale-txs.pdf, 2017.
Retrieved 01/29/20.

[6] V. Buterin.
Proof of Stake: How I Learned to Love Weak Subjectivity.
https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-

subjectivity/, 2014.
Retrieved 01/29/20.

[7] A. Garaoffolo, P. Stabilini, R. Viglione, U. Stav.
Proposal To Modify Satoshi Consensus To Enhance Protection Against 51% Attacks.
https://www.horizen.global/assets/files/A-Penalty-System-for-Delayed-Block-Submission-by-Horizen.pdf, 2018.
Retrieved 01/29/20.

[8] I. Craig et al.
The Hive: Agent-Based Mining in Litecoin Cash.
https://hive.litecoinca.sh/whitepaper.pdf, 2018.
Retrieved 01/29/20.

[9] M. Rosenfeld.
Analysis of hashrate-based double-spending.
https://bitcoil.co.il/Doublespend.pdf, 2012.
Retrieved 01/29/20.

[10] I. Eyal, E. Sirer.
How to Disincentivize Large Bitcoin Mining Pools.
http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/, 2014.
Retrieved 01/29/20.

[11] A. Judmayer et al.
Pay-To-Win: Incentive Attacks on Proof-of-Work Cryptocurrencies.
https://eprint.iacr.org/2019/775.pdf, 2019.
Retrieved 01/29/20.

[12] D. Moroz, N. Narula, D. Parkes.
Double-Spend Counter-Attacks: Threat of Retaliation in Proof-of-Work Systems.
Cryptoeconomic Systems Journal, 2020.

[13] T. Dickman.
Crypto51.
https://crypto51.app.
Retrieved 04/22/20.

[14] L. Childs.
How Many Confs?.
https://howmanyconfs.com.
Retrieved 04/22/20.

[15] T. Salmon.
fork.lol.
https://fork.lol/security/fork.
Retrieved 04/22/20.

[16] P. Wuille.
Bitcoin network graphs.
http://bitcoin.sipa.be/index.html.
Retrieved 04/22/20.

[17] M. Nesbitt.
Vertcoin (VTC) was successfully 51% attacked.
https://medium.com/coinmonks/vertcoin-vtc-is-currently-being-51-attacked-53ab633c08a4, 2018.
Retrieved 04/22/20.

[18] M. Nesbitt.
Deep Chain Reorganization Detected on Ethereum Classic (ETC).
https://blog.coinbase.com/ethereum-classic-etc-is-currently-being-51-attacked-33be13ce32de, 2019.
Retrieved 04/22/20.

[19] BitMEX Research.
Fork Monitor.
https://forkmonitor.info/.
Retrieved 04/22/20.

[20] BitMEX Research.
Bitcoin Cash SV – 6 block chainsplit.
https://blog.bitmex.com/bitcoin-cash-sv-6-block-re-organisation/, 2019.
Retrieved 04/22/20.

[21] S. Lerner.
Uncle mining, an ethereum consensus protocol flaw.

https://bitslog.com/2016/04/28/uncle-mining-an-ethereum-consensus-protocol-flaw/, 2016.
Retrieved 05/01/20.

[22] Consensys.
The Thirdening: What You Need To Know.
https://media.consensys.net/the-thirdening-what-you-need-to-know-df96599ad857, 2019.
Retrieved 05/02/20.

[23] J. Lovejoy.
Expanse (EXP) was 51% attacked.
https://gist.github.com/metalicjames/01222049f95f85df8c0eb253de54848b, 2019.
Retrieved 05/03/20.

[24] D. Palmer.
Bad Actors Rent Hashing Power to Hit Bitcoin Gold With New 51% Attacks.
https://www.coindesk.com/attackers-rent-hashing-power-to-hit-bitcoin-gold-with-new-51-attacks, 2020.
Retrieved 05/04/20.

[25] D. Palmer.
The Vertcoin Cryptocurrency Just Got 51% Attacked – Again.
https://www.coindesk.com/the-vertcoin-cryptocurrency-just-got-51-attacked-again, 2019.
Retrieved 05/04/20.

[26] R. Sharma.
Bitcoin Gold Hack Shows 51% Attack Is Real.
https://www.investopedia.com/news/bitcoin-gold-hack-shows-51-attack-real/, 2018.
Retrieved 05/04/20.

[27] J. Lovejoy.
Reorgs on Bitcoin Gold: Counterattacks in the wild.
https://medium.com/mit-media-lab-digital-currency-initiative/reorgs-on-bitcoin-gold-counterattacks-in-the-wild-da7e2b797c21, 2020.
Retrieved 05/04/20.

[28] Vertcoin Developers.
Vertcoin Development Update — January 2019.
https://medium.com/vertcoin-blog/vertcoin-development-update-january-2019-8dc39f6df210, 2019.
Retrieved 05/04/20.

[29] J. Lovejoy.
Vertcoin (VTC) was 51% attacked.
https://gist.github.com/metalicjames/f2acdb9ef448ec5298173b36c7c54133, 2019.
Retrieved 05/04/20.

[30] J. Lovejoy.
Litecoin Cash (LCC) was 51% attacked.
https://gist.github.com/metalicjames/82a49f8afa87334f929881e55ad4ffd7, 2019.
Retrieved 05/04/20.

[31] J. Wilmoth.
Litecoin Cash Allegedly the Latest Small-Cap Altcoin to Suffer 51 Percent Attack.
https://www.ccn.com/litecoin-cash-latest-small-cap-altcoin-to-suffer-51-percent-attack/, 2018.
Retrieved 05/04/20.

[32] F. Newbloke.
Hive 1.1 Litecoin Cash network security upgrade.
https://medium.com/litecoincash/hive-1-1-litecoin-cash-network-security-upgrade-66ba60d59523, 2019.
Retrieved 05/04/20.

[33] I. Eyal, E. G. Sirer.
Majority is not Enough: Bitcoin Mining is Vulnerable.
https://arxiv.org/pdf/1311.0243.pdf, 2013.
Retrieved 05/13/20.

[34] P. Daian et al.
Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges.

https://arxiv.org/pdf/1904.05234.pdf, 2019.
Retrieved 05/13/20.

[35] X. Chen, C. Papadimitriou, T. Roughgarden.
An Axiomatic Approach to Block Rewards.
https://arxiv.org/pdf/1909.10645.pdf, 2019.
Retrieved 05/13/20.

[36] M. Carlsten et al.
On the Instability of Bitcoin Without the Block Reward.
https://www.cs.princeton.edu/~arvindn/publications/mining_CCS.pdf, 2016.
Retrieved 05/13/20.

[37] R. Auer.
Beyond the Doomsday Economics of "Proof-of-Work" in Cryptocurrencies.
Globalization and Monetary Policy Institute Working Paper No. 355, 2019.
Available at SSRN: https://ssrn.com/abstract=3375168.

[38] A. Ouyang.
Simulating Horizen's Penalty System for Delayed Block Submission.
https://github.com/anneouyang/horizen/blob/master/README.md, 2020.
Retrieved 05/18/20.