IMPLEMENTATION AND ASSESSMENT OF THE REPUTATION-BASED MINING PARADIGM BY A COMPREHENSIVE SIMULATION

by

Pouya Pourtahmasbi

A Thesis Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, FL

May 2021

IMPLEMENTATION AND ASSESSMENT OF THE REPUTATION-BASED MINING PARADIGM BY A COMPREHENSIVE SIMULATION

by

Pouya Pourtahmasbi

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Mehrdad Nojoumian, Department of Computer and Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

[illegible]

Mehrdad Nojoumian, Ph.D.
Thesis Advisor

[illegible]
Imadeldin Mahgoub (Apr 19, 2021 13:01 EDT)

Imadeldin Mahgoub, Ph.D.

Taghi Khoshgoftaar
Taghi Khoshgoftaar (Apr 19, 2021 15:07 EDT)

Taghi Khoshgoftaar, Ph.D.

[illegible]
Hanqi Zhuang (Apr 19, 2021 15:08 EDT)

Hanqi Zhuang, Ph.D.
Chair, Department of Computer and Electrical Engineering and Computer Science

[illegible]

Stella N. Batalama, Ph.D.
Dean, The College of Engineering and Computer Science

Robert W. Stackman Jr.
Robert W. Stackman Jr. (Apr 19, 2021 21:35 EDT)

Robert W. Stackman Jr., Ph.D.
Dean, Graduate College

April 20, 2021

Date

ACKNOWLEDGEMENTS

I would like thank my graduate advisor, Dr. Nojoumian for giving me the opportunity of working on this research project at Florida Atlantic University. Through this research project, I was able to put my programming and problem solving knowledge into practice. I learned a lot by being involved in this research project.

I would like to thank Dr. Imadeldin Mahgoub and Dr. Taghi Khoshgoftaar, who agreed to be my thesis committee members.

I would like to thank all my professors who helped me to understand the fundamental concepts in mathematics and computer science.

This thesis was sponsored by the ARO and was accomplished under Grant Number W911NF-18-1-0483.

ABSTRACT

Author:	Pouya Pourtahmasbi
Title:	Implementation and Assessment of the Reputation-Based Mining Paradigm by a Comprehensive Simulation
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Mehrdad Nojoumian
Degree:	Master of Science
Year:	2021

Since the introduction of Bitcoin, numerous studies on Bitcoin mining attacks have been conducted, and as a result, many countermeasures to these attacks have been proposed. The reputation-based mining paradigm is a comprehensive countermeasure solution to this problem with the goal of regulating the mining process and preventing mining attacks. This is accomplished by incentivizing miners to avoid dishonest mining strategies using reward and punishment mechanisms. This model was validated solely based on game theoretical analyses and the real-world implications of this model are not known due to the lack of empirical data. To shed light on this issue, we designed a simulated mining platform to examine the effectiveness of the reputation-based mining paradigm through data analysis. We implemented block withholding attacks in our simulation and ran the following three scenarios: Reputation mode, non-reputation mode, and no attack mode. By comparing the results from these three scenarios, interestingly we found that the reputation-based mining paradigm decreases the number of block withholding attacks, and as a result, the actual revenue of individual miners becomes closer to their theoretical expected revenue. In addition, we observed that the confidence interval test can effectively detect block withholding attacks however, the test also results in a small number of false positive cases. Since the effectiveness of the reputation-based model relies on attack detection, further research is needed to investigate the effect of this model on other dishonest mining strategies.

IMPLEMENTATION AND ASSESSMENT OF THE REPUTATION-BASED MINING PARADIGM BY A COMPREHENSIVE SIMULATION

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Since the introduction of Bitcoin [2] in 2009, the popularity of cryptocurrencies has risen. It has reached the point that in the current markets, Bitcoin and other similar cryptocurrencies stand side by side with the world's most dominant national currencies as well as the traditional precious metals such as gold and silver. Today the total value of Bitcoin is over one trillion dollars [3] and by far it is the most valuable cryptocurrency. Despite Bitcoin's fast growth in value and popularity, the exact place of Bitcoin in the future's trade and reserve is not clear at this point. However, it is likely that the impact of the cryptocurrency on business and trade will only increase in the future. Furthermore, the backbone technology of Bitcoin, namely blockchain, has found its way into other major industries and it is predicted that the blockchain platforms and systems will remain important and may eventually dominate in future industries. [4]

Unlike the traditional currencies that are controlled and regulated by the central banks, Bitcoin along with the other cryptocurrencies are neither regulated nor controlled by any intermediary entity. The original idea behind Bitcoin is to make an autonomous and self-regulating cash systems that is open to the public. To accomplish this goal, Bitcoin takes advantage of peer-to-peer network technology [5] and state-of-art cryptographic protocols including public key, digital signature and cryptographic hash functions. The basic assumption is, as long as no single entity controls 50% or more of the total computational power of the network, the system is able to remain functional and consistent. While this fundamental assumption is backed by mathematical proofs, it is also widely accepted that a complete immunity to all attacks is not guaranteed in any system. In the case of decentralized cryptocurrencies, a number of vulnerabilities and attack scenarios have been studied over the past several years and are pervasive in the cryptocurrency literature. These vulnerabilities can be categorized into four different groups as follow:

1. Network attacks such as distributed denial of service [6] and sybil attacks. [7]
2. Transactional attacks such as double spending. [8]
3. Client-side attacks such as wallet theft. [9]
4. Mining attacks such as block withholding [10] [11] and selfish mining. [12] [13]

In this thesis, we exclusively focus on mining attacks. After reviewing many existing mining attack countermeasures from the literature, we focus on a reputation-based model that is proposed by Nojoumian et al. [1] This solution is aimed to combat mining attacks in cryptocurrency platforms. After reviewing the architecture of the reputation-based model, we present our computer program that simulates the cryptocurrency mining environment. We specifically design this simulation for the purpose of evaluating the effectiveness of the reputation-based model.

The rest of this thesis is organized as follows: In chapter 2, we review the existing mining attack scenarios along with the countermeasures from the literature. In chapter 3, we introduce our computer program and its main components. In chapter 4, we present the results of our simulation followed by our analysis. Finally in chapter 5, we conclude this thesis with our final remarks and the future direction of this research.

1.1 OUR MOTIVATION

The reputation-based paradigm is designed to effectively reduce the number of mining attacks by introducing a reward-punishment based mechanism. The proposed solution relies on game theoretical concepts and it is aimed to incentivize the miners to avoid malicious activities and commit to the honest mining strategies.

The proposed reputation-based model as well as vast majority of the other proposed countermeasures, are based on mathematical analysis and game theoretical concepts. Even though the theoretical arguments are still scientific and valid, due to the lack of sufficient empirical data, the real world implications of these theoretical solutions are not known. When real-world experimentation with a theory-driven scenario is not achievable, a computer simulation can bring new insights into both the problem and the hypothetical solution. Therefore, to evaluate the effectiveness of the reputation-based paradigm on mining attacks, we designed and implemented a Bitcoin mining simulation environment with the goal of deriving empirical data. From this empirical data, we then perform data analysis on the results to deduce whether the proposed solutions are significant and confirm the real world benefits of the reputation-based paradigm solution.

1.2 OUR APPROACH

For our simulation experiment, we examine the effect of the reputation-based model on block withholding attacks. As with any punishment-reward scheme, the key element that predominantly contributes to the effectiveness of the scheme, is the attack detection success rate. For various mining attacks, different detection solutions are available in the literature. Since our goal for this simulation is not to examine any particular attack detection solution, we choose to implement the block withholding attack scenario due to the relative simplicity of its detection method. Other mining attacks such as selfish mining and eclipsing are more sophisticated thus their detection methods are more complex. The complex nature of these attacks and their detection methods requires a more sophisticated simulation environment. This should be the subject of future research projects.

1.3 THE FUNDAMENTAL CRYPTOCURRENCY CONCEPTS

1.3.1 Blockchain

Blockchain is a linked sequence of ledgers known as blocks. Each block contains many transaction records that are secured using a cryptographic hash function known as SHA-256. Each record contains a set of transactional data including the amount, payer's ID, payee's ID and transaction's timestamp. Each block also contains its own unique hash value as well as the the preceding block's hash value. The first block in the blockchain is known as the genesis block. Since each block's hash is related to the hash from the previous block through SHA-256 algorithm, any modification to one block's content will make the hash of the subsequent block invalid and this continues all the way to the last block. The cryptographic relationship between the content of each block and its hash value, makes it very difficult for a malicious entity to temper with the content.[14] This feature brings immutability to blockchain and since each block is connected to its previous block in a chronological order, the information stored on the blockchain is verifiable and traceable by any entity who has access to the blockchain network. The combination of immutability and verifiability makes blockchain an appropriate technology to record the history of transactions. Therefore, blockchain can provide transparent and reliable information for its clients.[15] Blockchain is stored and managed through a peer-to-peer (P2P) network system. This means that all participants of the blockchain network have a local copy of the whole blockchain. The blockchain transactions are validated and stored by all network nodes and any new block that is added by a node will be distributed to all other nodes in the network. This feature makes blockchain decentralized and as a result, there is no need to trust any authority as the platform is self-regulated through a distributed consensus mechanism.[16] In other words, blockchain brings trust for the entire system without the need for a trustworthy authority. This also applies even if the blockchain clients don't trust each other.

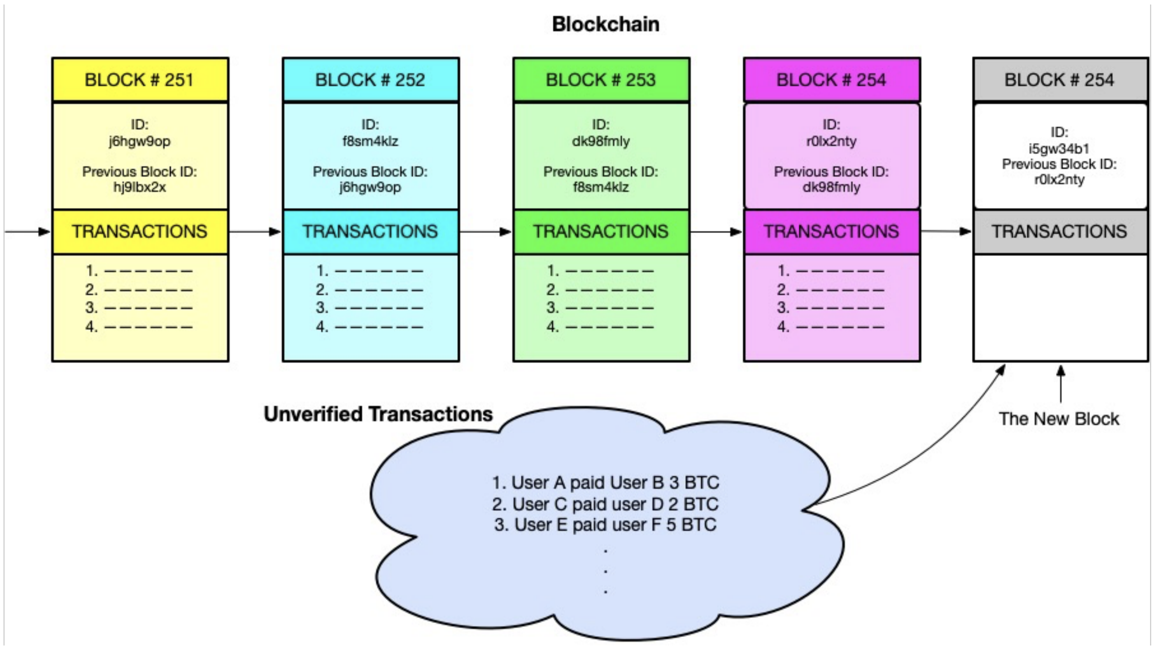


Figure 1.1: The Structure of Blockchain

1.3.2 Miners and the Proof-of-Work Mechanism

A new block can be added to the last block blockchain in a process known as mining. The network nodes who participate in mining are referred as miners. Mining is the process of finding a 64 digit hex hash value that must be smaller than a target value. This value cannot be calculated or estimated directly. The only way of finding a valid value is the process of trial and error. A miners simply tries different random hash values until a valid hash value is found. Since the sample space for a hash value is enormously large, finding a valid hash value, requires an extremely intensive computational work.[17] However, once the correct value is found by a node, it can be easily and quickly verified by other nodes in the network. This intensive computational process is referred to as proof-of-work (POW). A miner who successfully provides a POW, generates a new block in the blockchain. Then, the new and unverified transactions are recorded in this new block. The miner who provide POW is also rewarded with an amount of freshly mined cryptocurrency.

1.3.3 Mining Pools

Since the mining process is extremely competitive, the expected time intervals between each POW are very long. This issue becomes more severe as the ratio of the miner's hash power to the whole network hash power becomes smaller. Therefore the miners may need to wait a long period of time until they receive their mining rewards. Since miners must pay for their power consumption, the accumulation of the power costs for a long period of time may become too large to be affordable. To overcome this problem, a group of miners often form a coalition known as a mining pool. Once a miner from the coalition finds a POW, he will then send the solution to the pool manager. The pool manager publishes the new block on behalf of the pool and subtracts the pool's fee from the reward. The rest of the reward is distributed among all member miners by the pool manager. Each miner receives a portion of the reward equal to his hash power ratio to the entire pool's hash power. The expected revenue from pool mining is slightly smaller than the revenue from solo mining due to the pool fee. But mining in pools will reduce the expected time interval between each reward and as a result, the income from pools is more consistent and predictable than the revenue from solo mining. For miners who hold a very small portion of the hash power, it is not economically justifiable to mine solo as the expected time duration for a reward can be even longer than the miner's lifetime.

CHAPTER 2
A LITERATURE REVIEW OF MINING ATTACKS AND THE COUNTERMEASURES

2.1 MINING ATTACKS

As cryptocurrencies gained more popularity, Mining became more profitable and as a result, more miners would join the network to participate in the mining competition. This situation also makes the cryptocurrency platforms an attack target. Mining attacks are referred to the dishonest and malicious strategies that an individual miner or a coalition of miners can conduct for the purpose of gaining more revenue at the expense of honest miners. Game theoretic analyses [18] [19] [20] [6], demonstrate that there is an incentive for malicious miners to attack the cryptocurrency platform in many different ways. In section 1, we overview the various types of mining attacks. In section 2, we review the proposed countermeasures for individual attacks in the literature. In section 3, we review the reputation-based countermeasures that are aimed to incentivize the miners to be reputable.

2.1.1 Block Withholding

A miner who is the member of pool A, can practice a block withholding attack [11] against pool A. An attacker who supposedly is honest and mines for pool A, does not submit the full POW once he finds it and instead, he withholds the newly generated block. As a result, the honest members of the pool earn less than what they could have earned theoretically. By performing a block withholding attack, the attacker does not contribute in generating revenue for the pool, while he shares the reward with the pool if an honest miner submits a POW. The attacker will not be able to obtain anything directly by withholding a block since he cannot submit the POW independent from the the mining pool he is registered with. This is due to a protocol for mining pools that requires only pool administrators to submit a newly generated block. [21] Therefore, this kind of attack hurts both the attacker and all honest miners from Pool A. Block withholding attacks can be performed by the malicious miners for the purpose of sabotage. [11] This is usually originated from a suspect pool. Eyal [20] used game theoretical analysis to show that “no attack” is not a Nash equilibrium. This means that two mining pools attack one another for the purpose of increasing their revenue. Ironically, the consequence of this strategy is the opposite, both pools gain less if they attack each other. The author in [11] however, presents a more sophisticated scenario when miner m withholds blocks in favor of another pool, namely pool B. In this case, there is a secret alliance between miner m and pool B where, miner m withholds the blocks from pool A and he instead provides the solution for pool B. In return, miner m receives a percentage of the reward as a bribe from pool B. To be economically justifiable, this bribe must be greater than the percentage of reward that miner m receives from pool A. Block withholding attack can be practiced by one miner repeatedly. It can also be practiced by more than one miner from a pool. In fact, pool B may have a secret alliance with multiple miners from pool A. This can result in even more revenue loss for pool A and on the opposite side, pool B will gain significantly more revenue than its theoretical expected revenue. Bag et al. [10] shows that there is an incentive for a malicious miner to attack larger pools while receiving bribes from smaller pools. Block withholding can also be a prelude to selfish mining attacks.

2.1.2 Selfish Mining

In the honest mining strategy, if the miner finds a POW, he publishes the newly generated block publicly. Consequently, all other miners will stop solving the already solved hash puzzle and they will move on to the next block. Selfish mining is a mining strategy in which, the selfish miner finds a POW but withholds it from the network so the newly generated block remains hidden from the rest of miners. Then the attacker tries to find the next POW when the rest of miners waste their mining power on the previous block that is already generated. Because there is a potential to increase the attacker's profit, there exist an incentive for selfish mining strategy. [12][13]

In the original bitcoin paper, Satoshi [2] stated that the Bitcoin network is secure as long as the computational power of the honest miners remains the majority. However according to the more recent studies, the selfish miner can still undermine the security and the integrity of the cryptocurrency platform even if they don't control the majority of the computational power. In particular, Eyal and Sirer [13] show that in order for a pool to conduct a profitable selfish mining attack, it only needs more than a quarter of the total computational power. In a different paper, Sapirshtein et al. [22] analyzed the minimum resource required to conduct a selfish mining attack namely 'optimal selfish mining strategy'. This malicious mining strategy will remain at least as profitable as honest mining strategy therefore, there is an incentive for the miner to practice the selfish mining strategy.

2.1.3 Stubborn Mining

In general, the selfish mining strategy is used by the attacker to acquire short term rewards. From the attacker's point of view, if his private chain is longer than the public's chain, he will continue mining on his private chain for the hope of generating even more blocks. But if the attacker's private fork, falls behind the public chain, he disregards his private fork and reverts back to the public chain. Stubborn mining, first introduced by Nayak et al. [23], is similar to selfish mining strategy but the attacker persists on selfish mining even if his private chain falls behind the public's chain. So in general a stubborn miner will not give up on selfish mining easily. The authors argue that the selfish mining strategy is not optimal. It means that by following the stubborn strategies, the attacker will be able to gain even more rewards than if he only employs selfish mining strategy. Then the authors introduce three different stubborn strategies namely, Lead stubborn, Equal fork stubborn and Rail stubborn.

In these strategies, the attacker strategically reveals his hidden individual blocks based on the current state of the blockchain rather than just revealing his entire fork at once. This allows the attacker to increase his revenue by up to 25%. The authors also studied a model in which, stubborn mining is combined with an eclipse attack, hence the attacker can gain up to 30% more rewards in comparison with the less advance use of eclipse attack. The profitability of stubborn mining strategies depends on the duration of the attack and the analysis of the expected revenue. [24]

2.1.4 Eclipsing

Bitcoin's is a decentralized system that is built upon peer-to-peer network infrastructure. This network is designed to be open to the public. Unlike the server-client based network communications, peer-to-peer communication is free from cryptographic authentications. Each peer is identified directly by its IP address and the communication with other peers is established by using a randomized algorithm. This algorithm finds 8 random peers from the network and forms long lived outgoing connections. For the purpose of saving and broadcasting the addresses of other peers, the maximum number of 117 unsolicited incoming connections are also allowed. Through this communication, peers can broadcast and receive the latest state of the blockchain in the network. This network communication protocol can successfully provide an open and decentralized network environment. But there is a crucial security trade-off. The malicious peers can also join the peer-to-peer network with the intention of abusing honest peers or undermining the security of the network. An attacker can target a victim node from the network, isolating its connections to only the IP addresses that are controlled by the attacker. Then the attacker can manipulate what the victim can see in the network.

Eclipse attack often conducted for different purposes including Delivery-tampering attacks [25] engineering block races, 0- and N-confirmation double spending, generating a hidden fork on the network, making the victims to waste their computational power on old or obsolete blocks [26] and finally, hijacking the victim's computational power. The minimum required resources to conduct an eclipse attack has been the subject of many studies. Marcus et al. [27] presented an eclipse attack model on Ethereum network that can be imposed by an adversary who controls only two IP addresses on two computers. Heilman et al. [26] use a mathematical model to demonstrate an eclipse attack, then to confirm the practicality of this model, the authors perform it on their own Bitcoin. They conclude that, an attacker with only 32 distinct IP addresses or a 4600-node botnet is able to conduct an eclipse attack against a victim with at least 85% chance of success.

2.1.5 Routing Attack

This attack was first introduced by Apostolaki et al. [28] Routing attack can be conducted in small-scale targeting individual peers or large-scale targeting the whole network. The authors describe two kinds of routing attacks. The first kind is called "Partitioning Attack". By conducting this attack, the attacker is aimed to disconnect a set of nodes from the network completely. To do so, the attacker needs to disconnect all the connections between the victim and the rest of the network. Due to the complicated structure of Bitcoin network, the initial isolation may not be complete and the isolated node may still be able to hold some communications with the rest of the network. The attacker is able to detect and eliminate these connections until the victim nodes are completely isolated and the network is partitioned.

The second kind is called "Delay Attack". The purpose of this attack is to slow down the propagation of block sending from or received by the victim nodes. In contrast to partition attacks that requires a perfect isolation, a delay attack can be conducted by just intercepting a subset of connections between the victim nodes and the rest of the network. The detection of a delay attacks requires more challenges than a partition attack.

Routing attack can cause a significant loss to mining pools and individual miners. By partitioning the network or interrupting the propagation of blocks, the attacker can force the victim miners to waste their mining power on the blocks that are eventually discarded. Saad et al. [29] further studies various partitioning attacks on Bitcoin. The authors show that the Bitcoin network is getting more and more centralized at the AS-level. Through data collection and analysis, the authors show that the consensus among Bitcoin peers is non-uniform. As a result, the network is becoming more vulnerable to partitioning attack. The authors then demonstrate four variations of partitioning attack: spatial, temporal, spatio-temporal, and logical.

2.1.6 Pool Hopping

The availability of numerous mining pools, allows the miners the opportunity to switch between pools if they realize they can increase their profit by doing so. Lewenberg et al. [18] studied the reward sharing mechanism in mining pools by developing game theoretic models. They concluded that when the rate of transactions are high, it can be very difficult or impossible to keep the distribution of the accumulated revenue stable. Therefore, there is always an incentive for some miners to switch between pools. This pool switching can potentially sabotage the victim mining pools if it is practiced extensively. [11] A malicious miner constantly leaves and rejoins a mining pool based on the expected financial rewards the pool offers. When the mining pool offers low rewards, the attacker leaves and when the rewards are high, the attacker rejoins. This leaving and rejoining the mining pool, enables the attacker to receive more rewards than his expected hash power. On the other side, the honest miners who mine for the same mining pool consistently, receive less rewards than their expected hash power.

This situation prevents the victim mining pool to operate effectively and to mine blocks successfully before their competitors. [30] As a result, the honest miners will lose money by staying in the victim mining pool and eventually, they have to leave the pool. Rosenfeld demonstrates in [11] that the current reward system practiced by mining pools, encourages pool hopping attack in the sense that pool hopping is more profitable than mining continuously for a pool.

2.2 INDIVIDUAL ATTACK COUNTERMEASURES

In this section, we discuss mining attack detection solutions and countermeasures that have been studied since the emergence of Bitcoin in 2009. The majority of these countermeasures and detection methods are aimed to address one of the attack types that we explained in the previous section.

2.2.1 Countermeasures for Block Withholding

Block Withholding within a mining pool can be detected by relying on the theory of probability. For every pool, the ratio between the computational power of the mining pool and the whole network's computational power represents the probability of finding a POW by that mining pool. If the actual number of POW is significantly bellow the theoretical number of POW, there is a high possibility that the pool is under block withholding attacks. Even though it is not difficult to detect such an attack, finding the source of the attack could be challenging. [31] Rosenfeld [11] suggests a trickery that can be employed by the pool manager to dupe a dishonest miner into withholding a supposedly POW. Pools can catch dishonest miners by incorporating this method at the expense of dedicating a portion of their computational power on a task that does not produce any revenue.

As discussed earlier, withholding a POW alone does not benefit the perpetrator. Often this kind of attack is just the tip of the iceberg. Block withholding might be the sign of a deeper and more complicated adversary activity. Since the block withholding attack can be conducted for different purposes, various countermeasures have been suggested based on both the incentive of the attacker and the complexity of the operation.

Bag and Sakurai [32] consider a model in which, a miner from a mining pool lunches a block withholding attack on a target mining pool. Then the authors investigate the parameters that could increase the profit of the attacker and finally they propose a rewarding scheme called "special reward". The goal of this new scheme is to discourage the attackers by rewarding them additionally if they find a new block. In addition an attacker who never submits a POW will be denied from this reward and as a result, the attacker losses revenue.

Lee and Kim [33] present a method that stops block withholding attacks. In this method, rather than the attack itself, the detection of the infiltration is the focus. The authors argue that, a block withholding attack most likely is the subsequent of an infiltration. Therefore, the attack will be prevented if the infiltration is detected. A pool that is suspicious of being attacked by another pool, can infiltrate into the attacker pool to investigate the attack.

Solat and Potop-Butucaru [34] proposed a new algorithm to prevent intentional block withholding attack. This new algorithm namely, ZeroBlock, incorporates the expected time measurement to validate newly generated blocks. For every transaction, an expected time is considered and calculated locally by the nodes. The expected time depends on the size of the network as well as the difficulty level for solving the POW puzzle therefore, is predictable. If the adversarial node does not submit the new block before the calculated expected time, the block will become invalid and subsequently disregarded by honest nodes. The authors state that, their solution will prevent an intentional fork creation on blockchain.

2.2.2 Countermeasures for Selfish Mining

When selfish mining first introduced by Eyal and Sirer [13], the authors suggest an adjustment to Bitcoin consensus that will block selfish mining strategies. The new adjustment includes an automated mechanism that prevents mining pools from conducting a profitable selfish mining strategy. This mechanism would work effectively as long as the maximum mining power of the pools is smaller than 25% of the total mining power. In addition, the prevention of the selfish mining strategy requires at least two third of the miners to be honest.

As an extension to the countermeasure presented in [13], Heilman [35] presents a new concept called "Freshness Preferred (FP)". The new defense increases the minimum share of mining power from 25% to 32%. The new solution incorporates the use of an unforgettable timestamp. The author also shows that, it will be difficult for a selfish miner group to work against the proposed defense mechanism. This is due to the situation that a member of the selfish mining group is able to anonymously reveals the fact that the group have compromised the system. This ability gives the members an incentive to blackmail other dishonest miners. Therefore, it will be difficult and risky for selfish miners to form a group.

Zhang and Preneel [36] state that the solutions suggested by [13] and [35] are only effective if the selfish chain is shorter than the public chain and as a result, these defenses are incapable of combating a resourceful selfish miner. Then, the author proposes another solution that is effective even if the selfish miner's chain is longer than the public's chain. This solution is based on revising the fork-resolving policy that dismisses the blocks that are not submitted on time. Instead, this policy promotes the blocks that are linked to the competing blocks of their predecessors. According to the authors, while this solution outperforms the previous solutions suggested by [13] and [35], it is also backward compatible.

The solution introduced in [34] as a defense against block withholding attack, is also suggested by the same authors in a different paper [37] as an effective and practical solution to address selfish mining attacks.

Kokoris-Kogias et al. [38] introduces a new algorithm called "ByzCoin", which is claimed to be Byzantine fault tolerant [39]. This algorithm is designed to increase the security and consistency of the blockchain based systems such as Bitcoin. The authors suggest that, by equipping Bitcoin with the ByzCoin algorithm, the selfish mining strategy becomes ineffective. The algorithm resolves forks instantly, making a private fork a waste of time and resource.

2.2.3 Countermeasures for Stubborn Mining

Nayak et al. [23] state that eclipse attacks and stubborn mining can be detected by monitoring the stale block rate. Stale blocks are the blocks that contain a valid POW and transaction data but they are not included in the main chain. [40] The appearance of stale blocks is probable on a random basis even in the absence of malicious miners. The difference is, if all miners are honest, the relative rate of stale block appearance is equal for all miners. If there is a stubborn miner, the rate of stale blocks would vary depending on the attacker's strategy and the network parameters. An attack can be detected by counting the number of stale blocks and comparing it with the baseline.

2.2.4 Countermeasures for Eclipsing

Overlay networks in general can be vulnerable to eclipse attacks. There have been number of studies on eclipse attacks. [41][42][43][44] In these studies, more constrains and structural changes are suggested to effectively prevent eclipse attacks. Bitcoin on the other hand, is an unstructured network, which means that each node is connected to a an arbitrary subset of nodes determined by a randomized algorithm. Many other studies [45][46][47] are focused on designing a new unstructured network scheme that is tolerant of Byzantine attacks.

Jesi et al. [48] proposed a new detection mechanism that can identify and blacklist the adversary nodes. The authors state that their method is effective when large number of malicious nodes are present in the network. This defense mechanism however does not fully preserve the open and decentralized structure of the Bitcoin network.

Heilman et al. [26] study eclipse attacks on Bitcoin network exclusively and present countermeasures. These countermeasures are inspired by botnet architectures and are designed to preserve the Bitcoin's network architecture. The authors suggest few techniques that can be used to save the IP addresses of the trusted nodes. If the user is connected to unknown peers, these unknown peers' IP addresses are saved in a different variable namely "tried". The communication between the user and other peers in the network depends on the trust level that can be changed from time to time. The authors also mention that the attack has some specific features that makes it identifiable such as a sudden TCP connections from variety of IP addresses that send ADDR messages containing "trash" addresses. Once detected, these malicious addresses will be blacklisted from the network.

2.2.5 Countermeasures for Routing Attacks

Apostolaki et al. [28] present both short-term and long-term countermeasures against routing attacks. The short-term measures are compatible with the current protocols so the early adopters can immediately take the advantage of increased protection. These countermeasures include: increasing the diversity of node connections, selecting peers while taking routing into account, monitoring round-trip time, monitoring additional statistics, embracing churn, using gateways in different ASes and preferring peers hosted in the same AS and in 24 prefixes.

The long-term measures on the other hand, require some adjustments to the Bitcoin protocol including: encrypting Bitcoin communication and/or adopting MAC, using distinct control and data channels, using UDP heartbeats and finally requesting a block on multiple connections.

2.2.6 Countermeasures for Pool Hopping Attacks

Belotti et al. [49] propose a solution to detect pool hopping based on the order of rewarding transactions. To determine whether a miner is practicing a hopping strategy, it is crucial to focus on his rewarding transactions. This detection strategy is based on time epoch which refers to a time window that the miner has received rewards from different mining pools. Time epochs can be determined by analyzing the Bitcoin transactions in the miner's wallet. The authors shows that pool hoppers' rewards are significantly more than the static miners. Also the behavior of pool hoppers does not necessarily correlates with the value of the cryptocurrency.

Slush pool is the first mining pool that has implemented an optimized rewarding system for the purpose of pool hopping prevention. In this mechanism, rewards are given to the miners in proportion to the score they have received in each round. The scoring algorithm dynamically computes a score for each share based on its submission time. Rosenfeld [11] discusses a scoring mechanism that is used to compute rewards for the members of a mining pool. The proposed algorithm is based on the scoring mechanism similar to Slush's implemented method but different in a way that the scores for each share remains the same.

Salimitari et al. [50] propose a prospect theory which can help a new miner to find the most profitable pool. The authors state that the best pool for a specific miner is not necessarily the best pool for all miners. The suggested utility function can be used to calculate the value function that determines the risk and loss for each miner according to their hash power and their electricity costs. The main priority to this method is loss avoiding rather than profit making. The authors evaluate the accuracy of their theoretical methodology by joining five different pools and mining for 40 days, then comparing the actual results with the predictions that are provided by the utility function. The results however show that the predictions are not as accurate as expected.

Luu et al. [51] propose a new protocol for an efficient decentralized mining pool called "SmartPool". This new protocol can be implemented in the existing cryptocurrency platforms. It is aimed to resolve the problem with centralized mining pools in Bitcoin and Ethereum by structuring a platform where mining is completely decentralized. The authors conduct an experiment in Ethereum testnet and conclude that the protocol is efficient in practice and therefore, ready to be deployed in the real blockchain networks.

Singh et al. [30] introduce a new prevention scheme namely, smart contract-based pool hopping attack prevention. The main purpose of this scheme is to secure the fair relationship between the miners by requiring them to share their computational power faithfully. This model provides number of benefits including: (1) The pool manager is able to monitor the action of each miner before they can join the mining pool. (2) It requires the miners to submit coins as escrow. If a miner try to abandon the mining pool, his escrow coins are seized as punishment. (3) To facilitate the calculation of the exact amount of escrow, a detailed numerical model is provided.

To examine the practicality of the proposed model, the authors implemented a case study for an IoT smart home based Ethereum blockchain network. The authors state that there are only a few limitations in this model that need to be addressed in future research.

2.3 REPUTATION-BASED COUNTERMEASURES

The deployment of a reputation-based paradigm is known to be a robust approach in controlling malicious activities on various platforms and environments. Particularly in decentralized platforms, a trust measurement scheme can compensate for the lack of the moderator entity. Unlike the mentioned mining attack countermeasures, the reputation-based countermeasures are not aimed to target a particular attack type. Instead, they are designed to effectively reduce all kinds of mining attacks by incentivizing the miners to be committed to honest mining strategies. In this section we review reputation-based solutions that are designed to be implemented on top of the blockchain-based cryptocurrency platforms.

Nojoumian et al. [1] propose a new reputation-based scheme for the POW computation. The authors rely on game theoretical analyses to demonstrate the effectiveness of their proposed model. By relaying on a system of reputation, the solution is meant to encourage the miners to stay away from dishonest mining strategies. Each miner is assigned a public reputation value and this value reflects how trustworthy the miner has been so far. The authors demonstrate that, by using this solution, honest mining becomes Nash Equilibrium. It means that, the miners who are more reputable, have a higher chance of receiving mining invitations from the pool managers than the ones who are not trustworthy. As a result, the miners are incentivized to maintain their reputation continuously. Even if miners can increase their short term revenue by mining dishonestly, in a long term, it will be in their best interest of miners to remain reputable.

Freeman et al [52] proposed a scoring mechanism that relies on a machine learning algorithm. This mechanism can help users to identify the risky and potentially fraudulent transactions. The user can interpret the reputability of other users in the network thus, deciding which users are trustworthy to do trades with. The scoring system is implemented in three steps. First, the users who have a history of theft and other malicious activities are blacklisted. In the second step, the honest users are differentiated from the adversary users. In the third step, based on the classification, a risk score is calculated. This score would represent the user's trust factor based on the involvement in fraudulent transactions.

Carboni [53] argued that a distributed and decentralized feedback-based reputation system can be implemented on top of the cryptocurrency blockchain. Then the author proposed a feedback mechanism in which, a cryptographic link is established between the translations and the services. This link then is recorded in the blockchain. The author acknowledges that the proposed model is not formally proven to be resistant to various attacks but it is still robust and reasonable when it is compared against the current feedback paradigm that exists on many popular online platforms such as eBay.

Zhuang et al [54] proposed a reputation-based consensus mechanism namely, proof-of-reputation (PoR). In this framework, all nodes have a reputation. This reputation is developed based on the node's transaction activities, assets and participation. In PoR, the consensus is processed in a round that includes the selection of the leader node who has the highest reputation. Then the leader node generates and publishes the block. The verification process is done by other higher reputation nodes through voting for the block. The top twenty precent of the nodes in the sorted reputation list are considered high reputation nodes. If the sum of the reputation value of the nodes that voted consent is greater than the sum of the reputation values of the two thirds of the highest reputation nodes then, the block is verified and will be added to the blockchain. Otherwise, the block is invalid and will be disregarded. When a new block is added to the blockchain, the reputation value of the nodes are updated.

Yu et al [55] proposed a reputation scheme namely RepuCoin. The authors claim that this proposed mechanism can withstand an attack even if the attacker has temporarily obtained more than 50% of the network's hash power. Also, the proposed framework is claimed to have throughput of 10000 transactions per second. This is done through the proof-of-reputation process, but the rate of voting power growth of the whole system is limited. In this framework rather than hash power, a miner's power is their reputation, which is the work the miner has done over the entire life of the blockchain.

Sun et al [56] proposed a reputation paradigm scheme for e-commerce blockchain called RTChain. In this system, the nodes' transactions and consensus activities affect its reputation value. RTChain is expected to provide the highest throughput need of e-commerce. In addition, the authors claim that the proposed framework is resistant against the majority of known attacks such as selfish mining, double spending, and eclipse attacks. The authors provide their evaluation for their prototype and and demonstrate that the proposed framework meets the requirements of e-commerce and is deployable.

CHAPTER 3
THE IMPLEMENTATION OF THE REPUTATION-BASED MINING PARADIGM

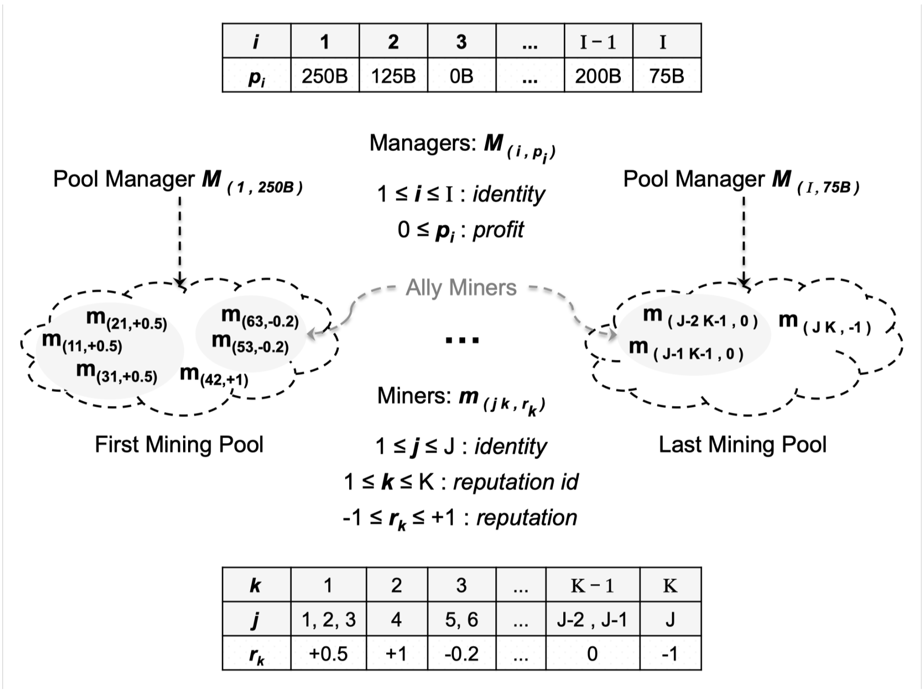
In chapter 2, we discussed various kinds of mining attacks as well as many proposed countermeasures from the literature. In this chapter, we introduce our mining simulation program. First, we demonstrate the architecture of our simulation followed by an in-depth explanation of the parameters, the algorithms, and the settings we incorporated in our simulation. Then in chapter 4, we provide the different scenarios in which, we run our simulation program along with the results for each scenario.

3.1 AN OVERVIEW OF THE REPUTATION-BASED MODEL

The reputation-based model includes a set of pool managers $M_{(i,p_i)}$ who form mining coalitions for $1 \leq i \leq I$, where $0 \leq p_i$ shows the profit that pool manager has accumulated so far; a set of miners $m_{(jk,r_k)}$ who perform mining, for $1 \leq j \leq J$ and $1 \leq k \leq K$ where $-1 \leq r_k \leq 1$ represents the reputation value of a miner.
In the current Bitcoin framework, each miner is given a unique identity i In the reputation-based model, along with i, each miner is also given a public reputation value. The value of r_k shows how reputable the miner has been so far and r_k is updated in specific time intervals based on the miner's commitment to honest mining within that time period. Honest mining denoted by $\mathcal{H}$ will result in an increase in and likewise Dishonest mining denoted by $\mathcal{D}$ will result in a decrease in r_k.
In the reputation-based model, the pool managers evaluate their pool members after a certain period of time. They send invitations to miners based on their reputation

value. The reputable miners are more likely to receive invitations compared to the miners who are not trustworthy. The miners who have received multiple invitations will have the option to join the pool they prefer whereas, the miners who have not receive any invitations cannot participate in pool mining.

The reputation-based model relies on the detection success rate. This means, the goal of the reputation-based model is only accomplished if effective attack detection solutions are incorporated into the mining scheme. Also, the reputation-based model must be immune to re-entry attacks. This means that the dishonest miner cannot exit out of the system and come back with a new reputation value. To accomplish this goal, the proposed model utilizes the approach of rational trust modeling [57]. In this model a permanent reputation parameter is linked to the identity of the miner and it will be preserved over time. This ensures the system is immune to re-entry attack.



i	1	2	3	...	I − 1	I
p_i	250B	125B	0B	...	200B	75B

k	1	2	3	...	K − 1	K
j	1, 2, 3	4	5, 6	...	J-2 , J-1	J
r_k	+0.5	+1	-0.2	...	0	-1

Figure 3.1: The Architecture of the Reputation-Based Setting [1]

3.2 THE ARCHITECTURE OF THE SIMULATION

The core of our simulation program is consisted of a set of miner entities, a set of pool manager entities, a set of interaction procedures, a random number generator engine and a result database creator.

Each entity has its own unique set of parameters which, we will explain in the next section. The values for entity parameters are generated by the random number generator engine. The probability distribution specification for each parameter is defined inside a separate module namely, the setting module.

The interaction procedures are divided into three modules: the pool joining module, the game module, and the attack module. These modules also rely on a number of parameters. The value for each parameter is either generated randomly by the random number generator engine, or defined as a constant inside the setting module.

The miner entities are contained inside a list data structure as a singleton object. The same is true for pool manager entities. The number of miner entities are dynamic throughout the game and it is effected by various conditions and procedures which we will explain in the section 4. The number of pool managers however, is constant and predefined inside the setting module. The architect of our simulation program is shown in Figure 3.2.

As shown in Figure 3.2, the simulation environment events and entity settings are derived from both pre-defined values and randomly generated values. Also the probability distribution property for random variables are derived from the pre-defined parameters. The result database includes the following simulation data from the simulation:

1. The complete set of all simulation entities including miners and pool managers with all their data.

2. The snapshot of each mining round that includes the simulation's statistical

data up to that round.

3. The list of all pool with their statistical data

4. The complete list of all block withholding events with the involved entities.

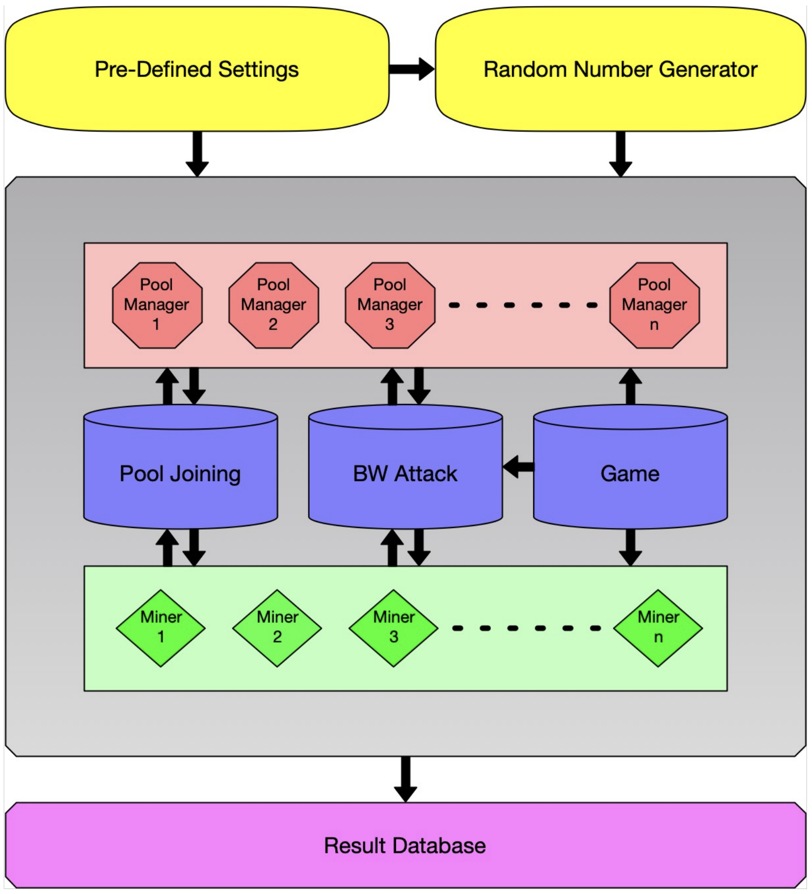


Figure 3.2: The Architecture of the Simulation Program

3.3 THE SIMULATION PARAMETERS AND ALGORITHMS

We design our mining simulation to be as realistic as possible. We consider many parameters, conditions and random events. There are several global variable parameters for our simulated mining game including: the total hash power of the system, the price of the cryptocurrency, mining profitability, and time intervals between each POW.

Also the entities, including the miners and the pools, are programmed in a way that each entity has its own unique set of parameters and characteristics. This approach mimics the real environment where a number of various parameters would affect the outcomes for both the entire system and the individual entities. In this section, we define the fundamental parameters and algorithms for our simulation program.

3.3.1 The Miner Parameters

The miners in our simulation are a set of entities who participate in the mining game to gain profit. Each miner is given a number of constant parameters as follow:

1. Hash Power: The hash power for the miners are generated randomly and the values are distributed normally with predefined mean, standard deviation, minimum and maximum.

2. Dishonesty Factor: It is a value between 0 and 1 that defines the probability that the miner commits to a dishonest behavior given that there is an opportunity. The larger this value, the more probable that the miner commits to a dishonest strategy. This value is normally distributed it is defined by constant mean, standard deviation, minimum and maximum.

3. Power Cost Rate: Each miner is given a power cost rate that is normally distributed with fixed mean and standard deviation. Investment: The value is derived from the value of hash power. It represents the amount of investment that the miner has dedicated to mining. In the real world, this can cover the costs for mining computers and other necessary equipments and facilities that are essential to mining.

4. Loss Tolerance Threshold (LTT): It is a negative profit threshold. If the miners profit becomes less than or equal to LLT, the miner stops mining and will no longer stays in the system. This value derived from value of investment: $LTT =$

$-p \times I$ where, p is a normally distributed random variable with predefined mean and standard deviation.

5. Target Profit Threshold (TPT): Is a positive profit threshold and the opposite of LLT. If this value is reached, the miner will stop mining and will no longer stays in the system. This value is also directly related to the value of the investment: $TPT = q \times I$ where q is a normally distributed random variable with predefined mean and standard deviation.

6. No target profit: This is a boolean value and if it is true, the miner ignores Target Profit parameters and continues mining even if Target Profit threshold is reached. This value is generated randomly with a constant probability.

7. Pool Fee Sensitivity: A Boolean parameter that defines whether the miner chooses a pool that charges the lowest fee or not. If this value is 'false', the miner joins a random pool and discards the fee differences between the available pools. The probability for this value is constant.

Since the mining game is essentially an economic activity, each miner is also given a set of variables that are updated by the interaction procedures throughout the simulation lifetime. These parameters are includes:

1. Actual number of provided POW: The total number of POW that the miner has provided throughout the game.

2. The Expected Number of POW: The number of POW that the miner is expected to provide based on the theoretical probability.

3. Revenue from Solo Mining: When a miner participate in solo mining, the sum of his revenue earned from solo mining, is saved inside this variable.

4. Revenue from Pool Mining: When a miner mines for a pool, the sum of his revenue earned from pool mining, is saved inside this variable.

5. Dishonest Revenue: The dishonest revenue is the sum of the amount of bribe that the miner has received from the suspect pools for committing block withholding attack against his own pool.

6. Power Costs: Mining requires power costs and the miner pays for the power cost as he mines. This cost is directly related to the amount of hash power the miner holds. This variable saves the total power costs that the miner has paid. This cost is calculated for each round based on the duration of the round. Unlike the revenue, the costs are calculated based on the power cost rate in dollar units.

7. Profit: This variable is the sum of all earned revenue minus the total costs. Since the revenue is calculated in the cryptocurrency unit, the amount of the profit depends on the current price of the cryptocurrency and it changes whenever the price of the cryptocurrency changes.

3.3.2 Pool Joining Utility Function

Miners can mine solo or they can join a pool and and share the rewards. Theoretically, the amount of the revenue that miner m earns remains the same whether miner m mines solo or mines for a pool. The only difference is the average time duration between each reward. If miner m mines solo, he will receive the full amount of reward once he provide a POW. If miner m mines for a pool p, miner m will receive a percentage of the reward every time miner m or another miner from p provides POW. The amount of this reward is significantly less than the amount of reward earned from solo mining however, the frequency of earning this reward is equally higher than the reward from solo mining. Therefore, within a sufficient long period of time, the expected revenue for miner m remains the same regardless of solo mining of mining for a pool. The miner m can determine the expected number of rounds it would take to provide a POW. Let i be the current round and $i + k$ is the round that the miner m is expected to provide a POW. The expected profit at round $i + k$ is calculated as follow:

$$E\left[P_{i+k}\right] = P_i - \frac{\Delta C}{p(x)} \tag{3.1}$$

In the above equation, P_i denotes the profit until mining round i , ΔC is the average power cost per round and $p(x)$ is the miner's hash power ratio to the whole network's hash power. If $E[P_{i+k}] \leq LTT$, then there is a high probability that the miner's profit will fall bellow LTT before miner m can earn any reward. Therefore, the miner will join a pool to reduce the expected number of mining rounds for each reward thus reducing the risk of falling bellow LTT.

3.3.3 The Pool Manager Parameters and Settings

In our simulation, a pool manager is an entities that contains a subset of miners. Once a member miner provides POW, the pool manager takes the pool's share and then distributes the rest of the reward among the member miners based on their hash power ratio to the whole pool's hash power. Similar to a miner entity, a pool manager entity is given a set of constant parameters as follow:

1. Pool Fee: A normally distributed random variable that defines the percentage of the reward that the pool takes before it distributes the reward among the member miners.

2. POW Reward Percentage: This is the extra reward for the miner who provide full POW for the pool. The value is normally distributed with fixed mean and standard deviation.

3. Honesty: A boolean value that defines whether the pool commits to a dishonest strategy or not. If this value is 'false' the pool never attempts to attack other pools. This value is generated randomly with a constant probability.

Also, a pool manager entity is given a set of variables that holds the result of the mining outcome for the entity as following:

1. Actual number of POW: The total number of POW that that all the miners from the pool have provided throughout the game.

2. Expected number of POW: The theoretical number of POW that is based on the probability that is defined by pool's hash power and the total network hash power ratio.

3. Honest Revenue: The total amount of the revenue the pool earned from the regular mining activity of the member miners.

4. Dishonest Revenue: The total amount revenue that the pool earns from the POW that the non-member miners from other pools provide.

5. Bribe Costs: The total amount of the bribe the pool has paid to non-member miners for block withholding attack.

Unlike miners, the number of pool managers in our simulation is constant and it is set to Eight. Therefore, each miner has the option to mine solo or join any of these eight pools.

As we explained earlier, each pool manager is given a constant fee rate. Pool managers subtract their fee from the reward before distributing it among miners. In our simulation, the miners select pools based on a non-uniform probability distribution that is defined by the pools' fee rate. This means that the pools with a lower fee rate are more likely to be joined by the miners than the pools with a higher fee rate. The pool managers also update the reputation value for all member miners periodically. In the following sections we explain the reputation calculation fundamentals and procedure.

3.3.4 Miner Population Functions and Parameters

The miner's population is a variable throughout the life of the simulation. The simulation starts with a fixed population and population growth is determined by a set of sigmoid functions. Note that the mentioned functions only contribute to the growth of the population. The decline in the population is the result of miners falling bellow their LTT thus leaving the system. The growth rate is controlled in two phases.

1. Phase 1: For $0 < t < \lambda$ the population is defined by a sigmoid function as follow:

   $$P(t) = \frac{P_M}{1 + e^{-\alpha(t-\theta_m)}}$$

   In the above equation P_M is the maximum population, θ_m is the time that the population reaches exactly $P_M/2$ and α is the steepness in growth. P_M, θ_m, α and λ are all predefined parameters.

   Since the population in our simulation increases over time, the program must determine the population growth at the end of each mining round. This ensures that the appropriate number of miners are added to the current population as the simulation progresses. To determine the growth rate for the population at time t, we can use the derivative function assuming dt is a very small time value:

   $$dx = P\prime(t)\, dt$$

   Now let assume the population is $P(t_1)$ at time t_0 and we need to determine how many new miners must be added to the population at time t_1. Since $t_1 > t_0$ and the population function is continuous, we can use the following procedure to calculate the population change at time t_1

   $$\Delta P = \sum_{i=0}^{n} P\prime(t_0 + i\, dt)\, dt$$

   where

   $$n = \frac{t_1 - t_0}{dt}$$

and

$$P(t_2) = P(t_1) + \Delta P$$

2. Phase 2: $t \geq \lambda$ For the population growth is calculated using the following function:

   $$\Delta P = \frac{P_M}{1 + e^{-1\beta(P(t) - P_M \theta_r)}}$$

   The above formula is related to the price function which will be explained in the next paragraph. This function controls the population of miners by using parameters from the revenue function. Therefore, when the ratio between the rewards and the costs are high, the mining is profitable and more miners will join the system so the population will increase. As this ratio decreases, the growth becomes smaller and once it passes zero and become negative no new miner will join the system. If miners keep leaving the system, the population will decrease.

3.3.5 Mining Game

The outcome of the mining game in our simulation is determined by a probabilistic algorithm. The probability distribution of the random variable that is generated by this algorithm, is similar to the probability distribution of a real mining environment. Let $M = \{m_1, m_2, m_3, ..., m_n\}$ be the set of all miners and $H = \{h_1, h_2, h_3, ..., h_n\}$ be the set of the corresponding hash powers. The hash power for the total network is:

$$T_H = \sum_{j=1}^{n} h_j$$

where $n = |M|$

The value of h_j/T_H defines the probability that miner m_j provides a POW at round i. To simulate the mining game, we generate a random number that is uniformly distributed in the range $[1, T_H]$ and by the pseudocode in Algorithm 1, a random miner can be selected. Note that since the value of T_H may change for each round, the probability of providing a POW for each miner also changes.

Algorithm 1 The Mining Game Procedure

```
1: procedure MiningGame($M, H$)
2:     let $l$ be a random number between 1 and $T_H$
3:     Let $M\prime$ be the list containing a random permutation of all elements in $M$
4:     let $H\prime$ be the list containing all elements in $H$ corresponding to $M\prime$
5:     $sum = 0$
6:     for $j = 1$ to $n$ do
7:         $sum = sum + H'_j$
8:         if $sum \geq l$ then return $M'_j$
```

After each round of the game, the mining power costs are updated for all miners. The time between each mining round is calculated randomly using a pre-defined normal distribution property. Next, the costs for round i is calculated and added for all miners. This cost value is calculated based on each miner's power cost rate and the time duration of round i.

In our simulation, the number of cryptocurrency rewards per Blocks is defined as constant and the price of cryptocurrency is a variable that is calculated immediately after the end of each round using the following formula:

$$p = \frac{\bar{C}(r+1)}{\mu}$$

Where, μ is the number of cryptocurrency reward per new block and $\bar{C}$ is the sum of the costs for all miner per second for the last round:

$$\bar{C} = \frac{1}{T}\sum_{i=1}^{n} C_{m_i}$$

r is the current cost-reward ratio that is calculated using the following sigmoid function:

$$r = \frac{\gamma}{1 + e^{-1\beta(P(t) - P_M\theta_r)}} - \frac{\gamma}{2}$$

In the above function the value of r is always bounded between $-\gamma/2$ and $\gamma/2$. θ_r is the time that makes $r = 0$ and it is predefined. The values for γ, β and P_M are predefined as well.

3.3.6 Continuous Random Number Generator

A typical random number generator can generate random numbers within a defined range. However, a sequential set of numbers generated by a typical random number generator are expected to be discrete. This means that, the numbers are expected to fluctuate randomly and the rate of change between any two sequential values is indeterminate. This type of random number generator is not suitable in many cases. For example, the price of a currency always fluctuates, but the fluctuation is not completely random. When it is shown as a graph, it often resembles a smooth and continuous line. To simulate such a property, a continuous random number generator is required.

For this simulation, the continuous random number generator is based on the trigonometric function as follow:

$$f(t) = r \sin\big(\frac{\theta\pi}{2}(t - a)\big) - b$$

In this function, r is the range for change (amplitude), θ is the time duration until the function reaches the next extrema (a quarter of phase) a and b are offsets. The range for the value of r must be defined so the output of the function always stays within this defined range. Likewise, to avoid extreme low or high frequencies (the rate of change), the range for θ must to be defined as well.

This function can generate value for any given t (time in second). In order to generate smooth and random movements, the value of r and θ are generated using an ordinary random number generator and the values for a and b are initially set to zero (For the first function $f_1(t)$, $r = r_1$, $\theta = \theta_1$, $a = 0$, $b = 0$). This ensures that the function will return 0 for $t = 0$. Then, using the derivative function $f\prime(t)$, the point for function's next extrema is calculated. Let $E_1 = \big(t_1, f(t_1)\big)$ be such a point. For all $t \leq t_1$ the algorithm will generate values using function $f_1(t)$. Once $t > t_1$, the algorithm generates a new function $f_2(t)$ with new random values for r and θ. This time, the values for a and b are calculated in a way that the extrema point of the previous function $f_1(t)$ is also the point that makes the derivative of the new function $f_2(t)$ equal to zero. This ensures that $f_2(t)$ is connected to $f_1(t)$ at the junction point E_1 so the continuity of random values is preserved.

Next, the extrema for $f_2(t)$ is calculated and the algorithm will continue generating numbers using $f_2(t)$ as long as $t \leq t_2$. Once $t > t_2$ The mentioned process of generating a new function repeats. The continuous random number generator is used to modulate certain parameters and variables of our simulation. For each desired parameter or variable, a unique instance of the continuous random number generator object is created. Once the value of the parameter or variable is calculated, hen it is modulated by the designated continuous random number generator object. This scheme results in creation of numbers that are only partially random. The level of randomness can be controlled by increasing or decreasing the range of modulation:

$$m(y) = y + y.f(t)$$

The modulated parameters and variables in our simulation are the following:

1. Cryptocurrency Price: Once the new price is calculated, it is also modulated using the modulator algorithm. This will make the price changes slightly random.

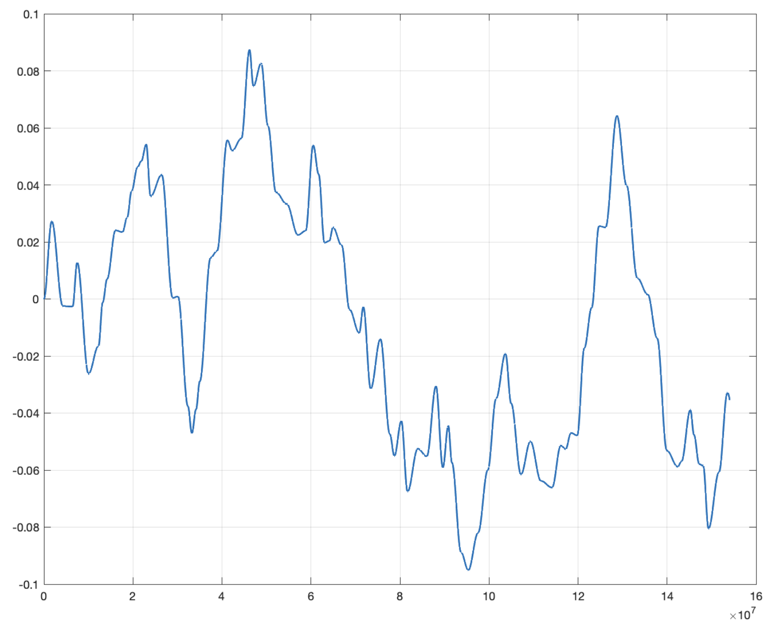


Figure 3.3: An output example of the Continuous Random Number Generator when $f(t)$ is bounded between -0.1 and 0.1

2. Maximum Population: This parameter is predefined for the simulation however, it slightly change by the modulator every time a new calculation takes place.

3. Zero Point for the Reward-Cost Ratio: Similar to the Maximum Population, this modulation adds a slight randomness to the Reward-Cost ratio parameter.

3.3.7 Attack Setup

In our simulation, the block withholding attack always originates from a dishonest pool. As mentioned earlier, there are 8 pools in our system; 3 pools are dishonest and 5 pools are honest. A dishonest pool only attacks a honest pool. This setting will result in a more appropriate data set for the purpose of highlighting the effects of a block withholding attack on the revenue of honest and dishonest entities separately. The block withholding attack is conducted in two phases as follow:

1. Initialization: Let D be the set of dishonest (suspect) pools and H be the set

of honest (victim) pools. After each round, a random dishonest pool d from D and a random honest pool h from H is selected. Then pool d will try to find a malicious miner from pool h. d searches for a malicious miner who also has a high hash power. If such a malicious miner is found, then the entities are set and the initialization is successful. Otherwise the initialization will terminate.

2. Process: Let m be the malicious miner from pool h who is committed to withholding one or more POW from pool h and instead, provides POWs for pool d for k times. Once miner m finds a POW, it is delivered to pool d and in return, pool d will pay miner m a percentage of the POW reward as bribe. This activity is repeated for k times under the condition that m finds a total k POW.

In this model, pool d would distribute the reward among its member miners the same way that it would have distributed it if the reward was earned honestly. The values for k and the bribe percentage are random based on a set of defined normal distribution properties.

3.3.8 Attack Detection

The attack detection method for a block withholding attack is relatively simple and relies on basic statistical analysis. As mentioned in part 4.2, the probability that miner m provides a POW in round i is simply h_m/T_{H_i}, where T_{H_i} is the total hash power at round i. Consequently, the expected number of POWs after playing n number of rounds can be calculated as follow:

$$E[x] = \sum_{i=1}^{n} \frac{h_m}{T_{H_i}}$$

The difference between $E[x]$ x and may be significant for small values of n, but for a sufficiently large n, it is expected that $E[x] \approx x$. Therefore, after a sufficient number of rounds, the pool manager can perform a statistical test on the individual miners to determine whether the difference between their actual and expected POW is significant or not. If the difference is indeed significant for one or more miners, particularly for miners with higher hash power, the pool manager can conclude that the pool is under a block withholding attack. To examine whether the difference between their actual and expected POW is statistically significant, we use the confidence interval (CI) test. First we calculate the ratio between $E[x]$ and x as follow:

$$\hat{x} = \frac{E[x]}{x}$$

Then the confidence interval is calculated as follow:

$$CI = \left(\hat{x} - z_c\sqrt{\frac{\hat{x}(1-\hat{x})}{x}} \ , \ \hat{x} + z_c\sqrt{\frac{\hat{x}(1-\hat{x})}{x}} \right)$$

For this experiment, different confidence levels such as 0.95 , 0.98 or 0.99 can be used. A lower confidence interval will result in a higher attack detection rate, but the drawback is that the number of false positive cases are expected to increase. Therefore, for this simulation we select 0.98 confidence level which provides the optimal results.

3.3.9 The Trust Function

The original idea for our trust function was first proposed in [58][59]. Our trust function is used to calculate the level of reputability for the player p. The reputation value is denoted as r and it is calculated for the player p after each round of game. For player p initially, $r_0 = 0$. This means that in the beginning, the reputability of player p is not known since p has not played yet. At each round i, the player p can either cooperate or defect. The cooperation will be rewarded by an increase in the reputation and the defections will be punished by a decrease in the reputation. The range for the reputation value is $[-1, 1]$ where, -1 is the lowest and 1 is the highest possible reputation values.

Our trust function is designed to maintain the reputation history of player p by updating and saving only few parameters. In our method, a defection not only causes a decrease in the reputation value, but it also causes a decrease in the growth rate of the reputation when player p chooses to cooperate in the future. In other words, when player p increases the number of times he has defected, he will have to spend exponentially more time cooperating in order to compensate for the reputation loss. After few defections, as player p cooperates repeatedly and consecutively, the growth rate of his reputation will increase until it is restored to the original value. Even when the growth rate is restored to its original value, if player p defects again, the reputation value as well as the growth rate will drop dramatically and further defections will exponentially cause more negative impact. In conclusion, our reputation calculation procedure is defection sensitive and it is aimed to incentivize the players to avoid defections if they are willing to stay in the system for a long period of time. To maintain the reputation for player p, our reputation function requires to update and maintain the following parameters:

1. The Total Number of Defections: Denoted as α and it is initially 0. It increments every time player p defects.

2. The Trust Deficit Value: Denoted as λ_i where i represents the round and $\lambda_i \geq 0$. Initially $\lambda_0 = 0$ then the value of λ_i increases every time the player p defects. Unlike α, the value of λ will decrease as the player cooperates but the decrease is at lower rate than the increase. The rate of decrease has an inverse relationship with α.

3. The Trust Variable: Denoted as x and x_i is the value of the trust at the end of round i. Initially $x_o = 0$ then, the value of x_i is calculated at the end of each round.

The complete procedure for calculating the reputation for player p at the end of round

i is given. Note that the first step of the calculation is to determine whether player p has cooperated or defected at round i. Then, the calculation must follow the order given bellow.

1)

$$\alpha_i = \begin{cases} \alpha_{i-1} + 1 & , \ \mathcal{L}_i = 0 \\ \alpha_{i-1} & , \ \mathcal{L}_i = 1 \end{cases}$$

2)

$$\lambda_i = \begin{cases} \lambda_{i-1} + ln(e + \lambda_{i-1}) & , \ \mathcal{L}_i = 0 \\ \lambda_{i-1} - log_{e^{\alpha_i}+1}(1 + \lambda_{i-1}) & , \ \mathcal{L}_i = 1 \end{cases}$$

3)

$$x_i = \begin{cases} x_{i-1} - \big(ln(e + \lambda_i)\big)^e & , \ \mathcal{L}_i = 0 \\ x_{i-1} + (e-1)^{-\lambda_i} & , \ \mathcal{L}_i = 1 \end{cases}$$

4)

$$r_i = \begin{cases} -1 & , \ r_{i-1} \leq \epsilon - 1 \ \ and \ \mathcal{L}_i = 0 \\ 1 & , \ r_{i-1} \geq 1 - \epsilon \ \ and \ \mathcal{L}_i = 1 \\ \frac{2}{1+e^{-\gamma x_i}} - 1 & , \ otherwise \end{cases}$$

In the above procedure, $\mathcal{L}_i = 1$ denotes that the miner has cooperated in round i and $\mathcal{L}_i = 0$ denotes that the miner has defected in round i. γ is a constant and positive steepness parameter. Larger values for γ results in a higher steepness for the function. ϵ is a small positive and constant parameter. $1 - \epsilon$ and $\epsilon - 1$ set an upper and a lower bound for the calculation of the reputation. Therefore, when $1 - \epsilon \leq r_{i-1} < 1$ and the player p has cooperated in round i, the function returns 1 and When $-1 < r_{i-1} \leq \epsilon - 1$ and the player p has defected in round i, the function returns -1. This procedure prevents the divergence of x and consequently, x will always remain within the proximity of the usable range of the Sigmoid function. In our trust calculation procedure, we used e as a constant as well as the base for the logarithm function for the calculation of λ and x. However, for further adjustments and tweaks, this constant parameter may be changed.

3.3.10 Using the Trust Function to Calculate the Reputation of Miners

The proposed trust function is used to calculate the reputation value for miners. There are many different ways to implement the trust calculation procedure inside the cryptocurrency framework. In our simulation, the trust function is managed by the pool managers. This means that, the trust function only affects the miners who are the member of a pool. Therefore, the reputation of solo miners remains zero as long as they don't join a pool. In our simulation, the pool managers perform a block withholding detection investigation after a number of mining rounds. This number is a random variable that is uniformly distributed within a defined range. Since the block withholding detection method relies on the confidence interval test, the quantity of samples is the key to accuracy. larger number of samples will result in a more reliable test. Therefore, we select a fairly large number as the low bound for the probability distribution range.

In the previous section we defined i as the number of round for simplicity. In our implementation however, the value of i represents the detection cycle that can contain any number of rounds within the probability distribution range. Therefore the reputation value for all miners from pool p will be updated by the pool manager once a new detection cycle takes place. The pool manager simply compares the number of actual POW against the number of expected POW for all member miners. Next, the reputation value for each miner is updated accordingly. The reputation updating procedure is given in Algorithm 2.

In the pseudocode shown in Algorithm 2, x_j denotes the actual POW and $E[x_j]$ denotes the expected POW since the last detection cycle for miner M_j. CI is the con-

Algorithm 2 Updating Miners' Reputation

```
1: procedure UPDATEREPUTATION($M, i$)
2:     for $j = 1$ to $n$ do
3:         if $x_j < E[x_j]$ and $x_i \notin CI$ then
4:             set $\mathcal{L}_i = 0$ for $M_j$
5:         else
6:             set $\mathcal{L}_i = 1$ for $M_j$
```

fidence interval that we introduced in section 3.4.8 and denotes the current detection cycle. In other words, the pool manager compares the actual POW and the expected POW for all member miners for the period between the detection cycle $i - 1$ and i. If the condition given in Algorithm 2 is true for miner M_j then, the pool manager identifies miner M_j as attacker.

3.3.11 Attack Utility Algorithm

Nojoumian et al [1] showed that when an attack detection mechanism as well as an appropriate punishment measurements exist in a system, attacking is no longer Nash equilibrium. Therefore it is expected that the players act rationally and they avoid dishonest behaviors. When miner m is given an attack opportunity, he can determine whether it is in his best interest to commit to the attack or not. The key to this determination is the rate of attack detection. Miners are incentivized to be honest as more dishonest miners in the system are detected and punished. In this case the punishment is a lower level of reputation and possible chance of losing mining in mining pools. The miner's long term utility function that is used in our simulation is based on this concept. Therefore, the rate of detection defines the extend that the miners act honestly. The pseudocode for attack utility is given in Algorithm 3.

In the pseudocode given in Algorithm 3, r is the attack detection ratio where, $0 \leq r \leq 1$ and d is a dishonest parameter that is unique to the miner where, $0 < d < 1$

. The higher the value of d, the more probable that the miner will accept the attack offer.

Algorithm 3 Attack Utility

```
1: procedure AcceptDishonestRevenue($dishonestReward, r$)
2:     if $dishonestReward + profit \geq targetProfit$ then return TRUE
3:     $\alpha =$ a random number between 0 and 1
4:     $t = (1 - r)^{\frac{1}{d}}$
5:     if $\alpha < t$ then return TRUE
6:     return FALSE
```

3.4 SUMMARY

In this chapter we introduced our simulation program. We demonstrated the architecture of our design and we explained the main parameters, algorithms and protocols we incorporated into our simulation. This simulation program is designed to examine the effectiveness of the proposed reputation-based model. To do so, we simulated a block withholding attack scenario in which, a suspect pool hires miners from a victim pool to conduct block withholding attack against the victim pool. This will cause an increase of revenue for the suspect pool and a decrease in revenue of the victim pool. We explained the detection method for block withholding attacks and we introduced our trust function that the pool managers use to update the reputation value for miners. In the next chapter we demonstrate the results of our simulation.

CHAPTER 4
THE ASSESSMENT OF THE REPUTATION-BASED MINING PARADIGM

In this chapter first, we explain the modes in which, we perform our simulation program then, we demonstrate the results for each scenario using graphs and tables. Finally we compare all results and conduct a statistical analysis to measure the differences between each simulation mode. This will enable us to determine the impact of the reputation-based model on block withholding attacks.

4.1 THE SIMULATION MODES

In order to evaluate the effectiveness of the reputation-based paradigm, we need to make a comparison between the reputation-based and the non-reputation-based mining environments. We also require a third scenario where no attack takes place. This simulation mode resembles the ideal environment where no dishonest activity takes place. The results from this mode will provide a reference point for the evaluation of reputation-based model. The data similarity between Reputation mode and No Attack mode, demonstrates a high level of success and effectiveness for the reputation-based paradigm.

We run the simulation program for 250,000 rounds of mining in each mode and the data for each mode is recorded individually. The probability distribution attributes for all random parameters are the same for all three modes. For example, the value for the hash power is generated randomly for all miners. These random hash power values are distributed normally and the defined mean and the standard deviation parameters are equivalent in our three modes. This will ensure that the distribution of hash power in all three modes are statistically similar. This setting is true for all other random parameters in our simulation. Therefore the differences between the results in each mode is guaranteed to be influenced by the reputation-based paradigm in the system and not the consequence of randomness. In addition, the settings for each pool including the pool's fee rate and the state of honesty or dishonesty are equivalent in all three modes.

1. **Non-Reputation Mode:** This mode resembles the current setting of the Bitcoin network where, there is no mining attack detection scheme and reputation-based paradigm are in place. In this setting, attacking is Nash Equilibrium since there is no potential consequence for block withholding attack. Miners are always engaged in attack if the opportunity is available. In this mode, the miners can join pools freely and the pool managers always allow any miner in the network to join their pool. The actual distribution of rewards among miners is expected to be significantly different from the theoretical probability distribution because the dishonest miners gain more rewards while the honest miners loose.

2. **Reputation Mode:** In this mode, all miners are given a reputation value that is initially zero and is periodically updated by their pool managers based on their level of their commitment to honest mining. In Reputation Mode, attacking is not Nash Equilibrium. If pool manager M detects an attack conducted by miner m, pool manager M will apply a defect function on the reputation value of miner m and subsequently expels miner m from the pool. The short-term utility function is the same as it is in the Non-Reputation mode. The long-term utility function. however, considers the long-term consequences of committing an attack and is used whenever an opportunity for an attack exists. The miner will determine whether the attack is profitable in the long term. In this mode,

Table 4.1: The Differences Between the Simulation Modes

	Non-Reputation	**Reputation**	**No Attack**
Pool Joining Mechanism	Free	With Invitation Only	Free
Implemented Attack	-	Block Withholding	-
Attack consequence	No Consequence	- The reputation value will be effected negatively - The attacker miner will be expelled from the pool	-
Utility Functions	Pool Joining	Pool Joining, Attack	Pool Joining

the miners can only join a pool if they received an invitation from that pool. The actual distribution of rewards among miners is expected to be different from the theoretical probability distribution, but not as drastically different as in the Non-Reputation mode.

3. **No Attack Mode:** This mode represents the ideal situation where all miners are committed to honest mining at all times. This settings in mode are similar to the other two modes with the exception that block withholding opportunities are not available to the miners thus no attacks will take place. The pool joining mechanism in this mode is identical to Non-Reputation mode where miners join pools based on a probability distribution that is defined by the pools' fee percentage. The actual distribution of rewards among miners is expected to be the closest to the theoretical probability distribution.

4.2 THE RESULTS

In this section we provide the results of our simulation program in all three modes explained in in the previous section. First we provide a summary of the results in tables and figures and then we present the statistical analysis for the results.

4.2.1 The Summary of the Results

The summary for 250,000 rounds of mining in each scenario is shown in Table 4.2. Note that the values for distributed rewards and costs are shown in dollar. The reward values are calculated based on the price of cryptocurrency at the end of round 250,000 and the changes in price over time for all 3 scenarios are shown in Figure 4.2 The value for miners' costs is the accumulation of all miners' costs until the last mining round.

The active miners are the miners whose profit value is still above LTT and the inactive miners are the miners whose profit value fell bellow LTT at some point during the game and as a result they have left the system. All the statistical results that are shown in tables and graphs, include the data for both active and inactive miner groups. The number of block withholding cases indicate the number of times an individual attack case is processed. For each attack case, the miner who attacks may withhold blocks more than once. The number of block withholding instances for each attack is set when the attack is initialized.

The results for each pool in shown in Table 4.3 to 4.5. Note that the percentage of hash share for each pool shown in Table 4.3 to 4.5 and Figure 4.3, varies throughout the life of the simulation. The given results represent the distribution at the end of round 250,000. The hash power distribution in Non-Reputation and No Attack modes are very close, but it is slightly different in Reputation mode. This is due to the fact that the pool joining mechanism in Reputation mode is invitation based and explains the slightly higher percentage of solo miners in the Reputation mode. The pool joining mechanism in No Attack and Non-Reputation modes are the same therefore, the insignificant differences between the two modes must be the result of randomness in the simulation.

The differences between the actual number of POWs and the expected number of POWs for each pool shown in Figure 4.4, indicates a significant difference between

Table 4.2: The Summary of Simulation in Each Mode after round 250,000

Description	**Non-Reputation**	**Reputation**	**No Attack**
Time in Seconds	154,092,032	154,090,378	154,332,223
Total Hash Power	61,270,349	67,394,318	67,822,864
Number of Active Miners	20,744	22,801	22,938
Number of Inactive Miners	9,298	7,257	8,042
Percentage of Pool Miners	93.7%	93.1%	94.4%
Price of the Cryptocurrency	$1,168.28	$1,375.37	$1,237.23
Total Distributed Rewards among All Miners	$5,840,080,000	$6,870,660,000	$6,186,100,000
Total Power Costs for All Miners	$6,312,700,000	$6,290,500,000	$6,374,900,000
Percentage of Miners with Positive Profit	31.3%	49.9%	36.4%
Total Block Withholding Cases	1,128	557	0
Detected Block Withholding Cases	0	382	0
False Detected Block Withholding Cases	0	23	0

Table 4.3: The Summary of Pools after Round 250,000 in Non-Reputation Mode

Pool	BYT-728	KRM-664	MME-935	RLC-061	SJN-888	UFR-774	VPK-703	WSQ-559
Hash %	19.7%	4%	3.56%	4.74%	5.45%	16.6%	17.3%	17.1%
E-POW	41,509	10,153	8,712	8,729	10,608	37,499	39,670	38,802
A-POW	45,838	8,275	7,003	12,877	14,805	34,044	36,749	35,845

Table 4.4: The Summary of Pools after Round 250,000 in Reputation Mode

Pool	BYT-728	KRM-664	MME-935	RLC-061	SJN-888	UFR-774	VPK-703	WSQ-559
Hash %	21.98%	2.76%	2.8%	3.3%	3.99%	17.3%	17.1%	16.97%
E-POW	47,251	7,063	7,027	7,298	8,680	40,070	37,988	38,635
A-POW	48,336	6,750	6,718	8,672	10,044	38,755	37,124	37,749

Table 4.5: The Summary of Pools after Round 250,000 in No Attack Mode

Pool	BYT-728	KRM-664	MME-935	RLC-061	SJN-888	UFR-774	VPK-703	WSQ-559
Hash %	17.99%	4.26%	4.8%	3.22%	4.84%	18.12%	18.38%	18.28%
E-POW	39,232	9,064	9,553	7,340	10,682	39,318	40,173	39,909
A-POW	39,266	9,004	9,545	7,330	10,756	39,257	40,019	39,860

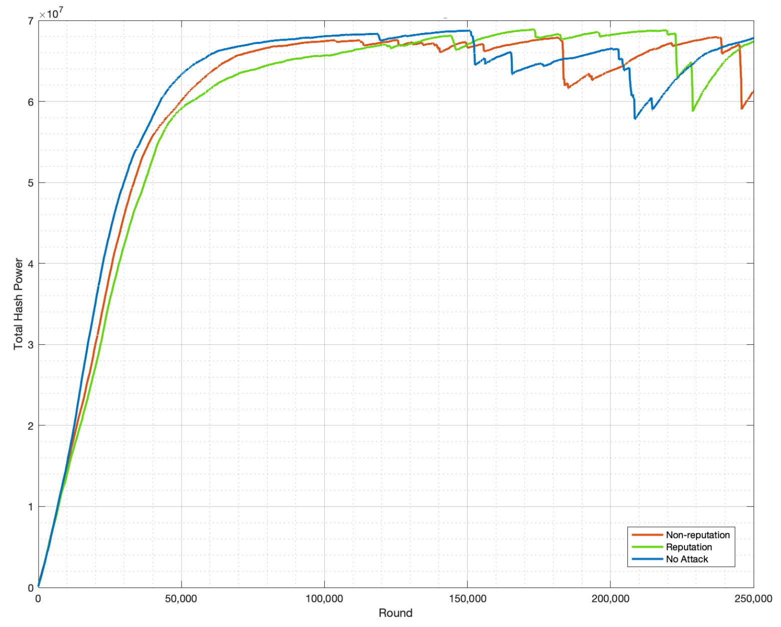


Figure 4.1: The Change of Total Hash Power

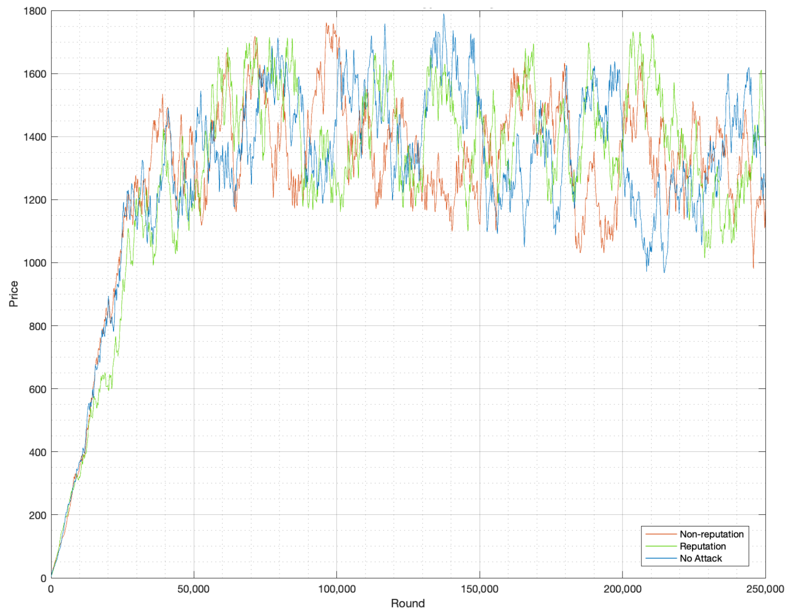


Figure 4.2: The Change of Cryptocurrency Price

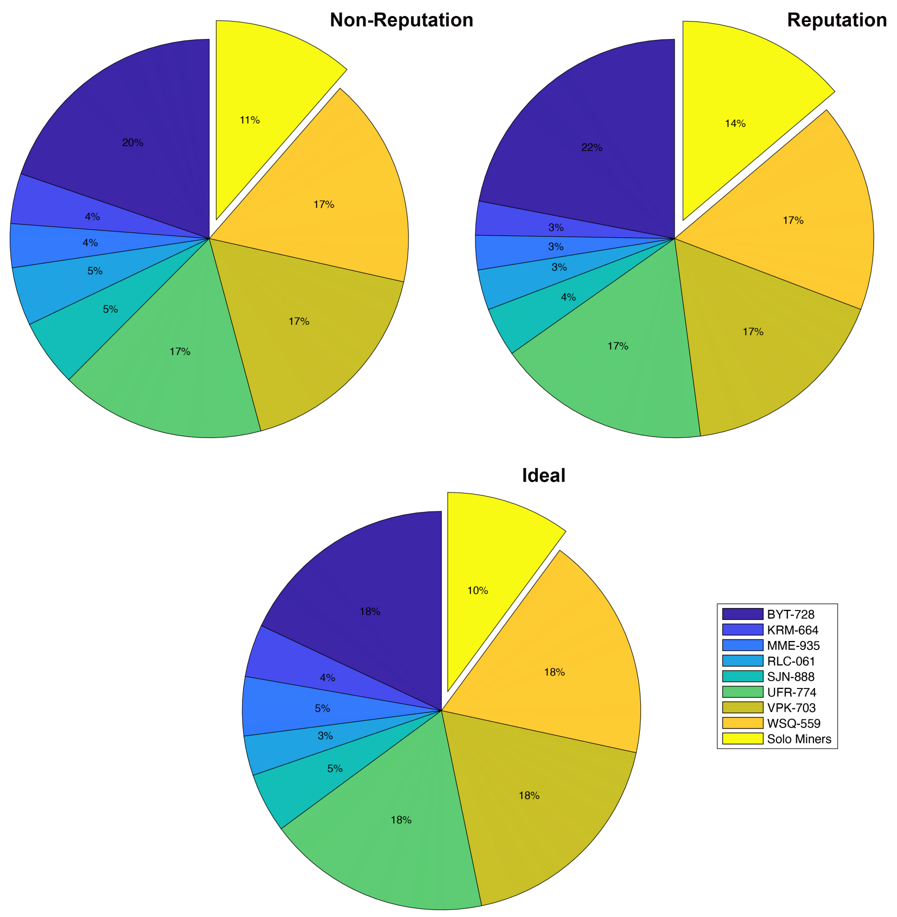


Figure 4.3: The Distribution of Hash Power Among Pools in Each Simulation Mode

each mode. The high values for the Non-Reputation mode are the consequences of block withholding attacks where the actual number of POWs are much higher than the expected number of POWs for all suspect pools. The opposite is true for all victim pools. We observe that in the Reputation mode, this similar pattern still exists but at a much lower magnitude. In contrast, the small differences between the actual POW and the expected POW in the No Attack mode, is the natural result of the probability outcome and it does not follow any pattern. Figure 4.5 demonstrates the number of block withholding attacks in groups of 50,000 rounds. In both modes the number of attacks are in decline for the first 100,000 rounds. At first glance this pattern may seem unusual, but a deeper analysis demonstrates that as the number of miners

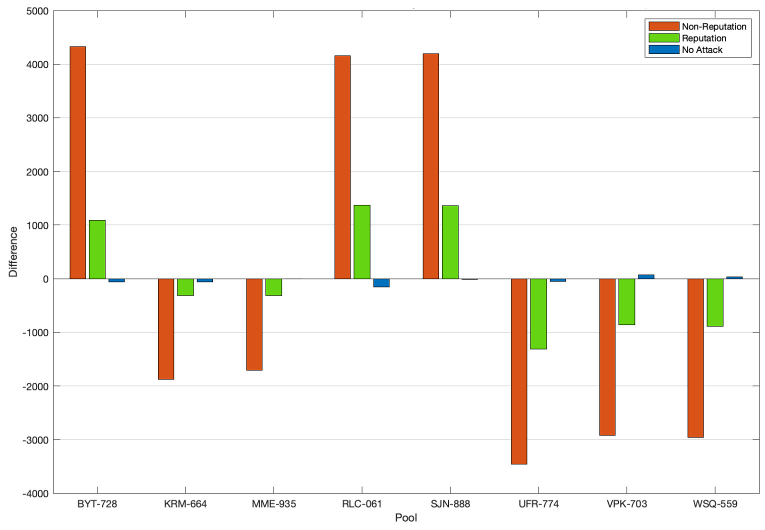


Figure 4.4: The Differences Between Actual and Expected Number of POW for all 8 Pools After 250,000 Rounds of Mining

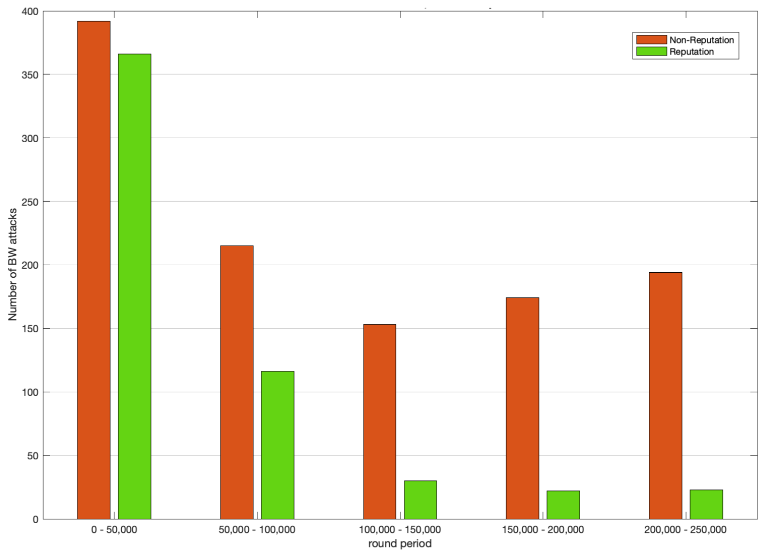


Figure 4.5: The Number of BW Attacks in periods of 50,000 rounds

increase in the system, the probability that a single block withholder miner provides a POW decreases. This pattern is followed by a decrease in the success rate of block

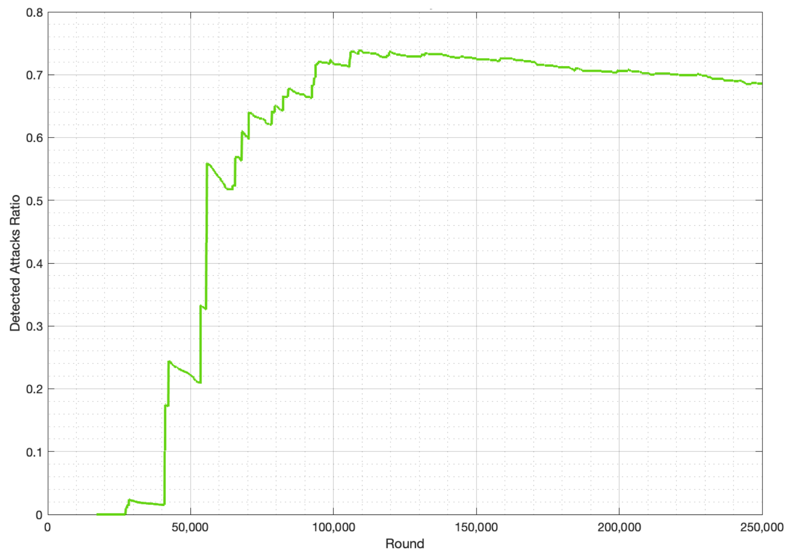


Figure 4.6: The Attack Detection Ratio in the Reputation Mode

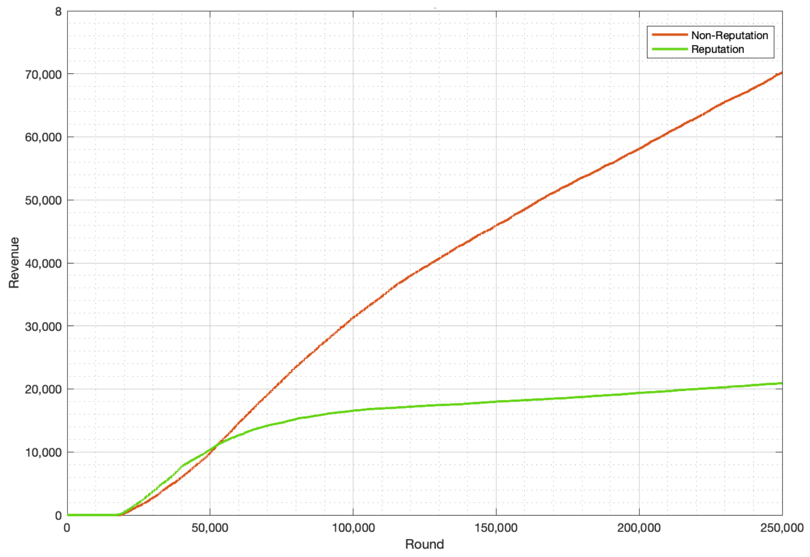


Figure 4.7: The Accumulation of Bribe Paid to Miners by the Suspect Pools

withholding attacks. Figure 4.1 shows that the total hash power of the system reaches its peak at approximately round 100,000. While the attack initialization rate remains relatively constant, this pattern perfectly correlates with the decline in the number of attacks until round 100,000.

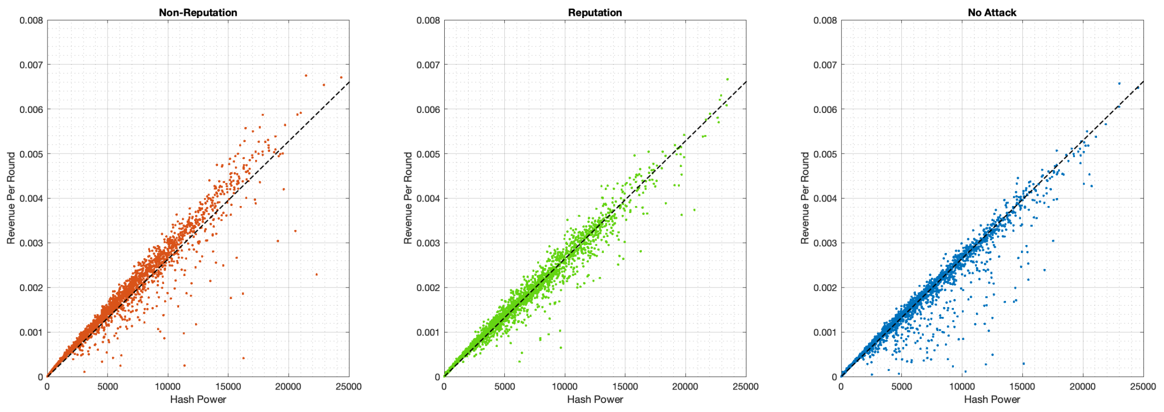


Figure 4.8: Average Revenue Per Round for All Miners from Suspect Pool BYT-728

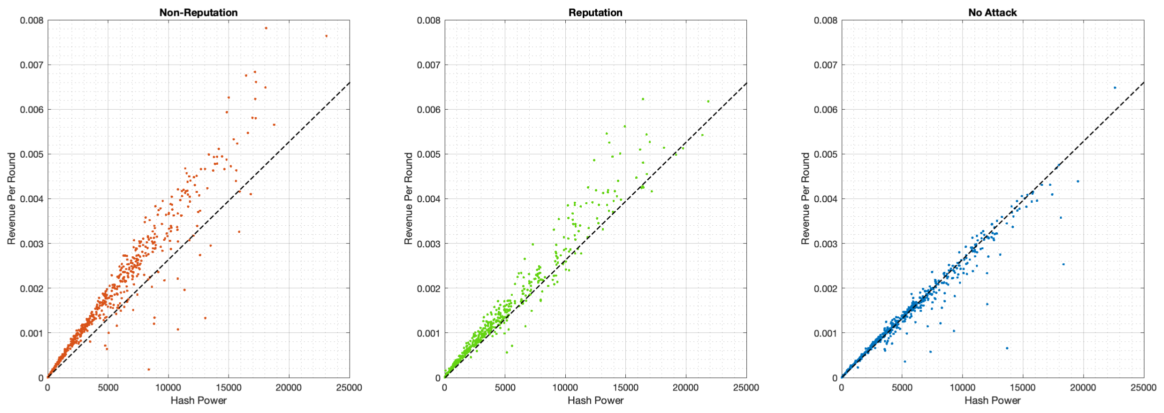


Figure 4.9: Average Revenue Per Round for All Miners from Suspect Pool BYT-728

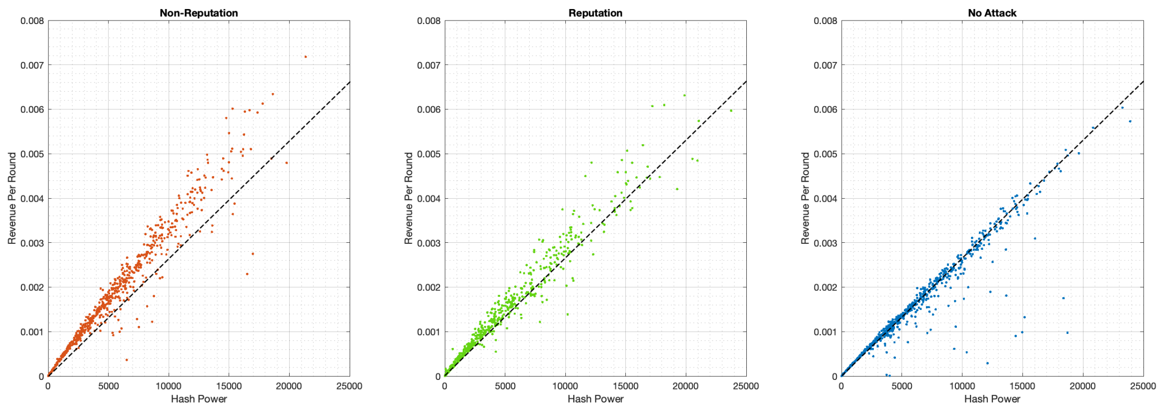


Figure 4.10: Average Revenue Per Round for All Miners from Suspect Pool BYT-728

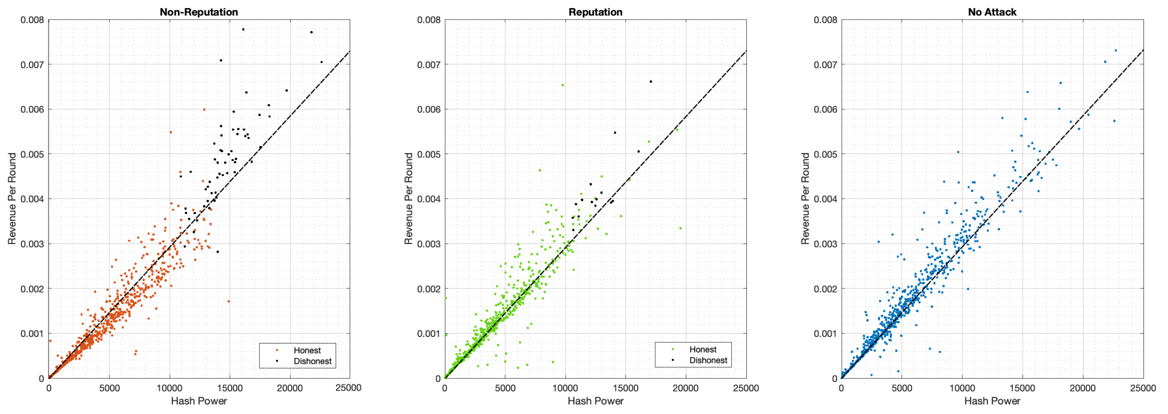


Figure 4.11: Average Revenue Per Round for All Miners from Victim Pool KRM-664

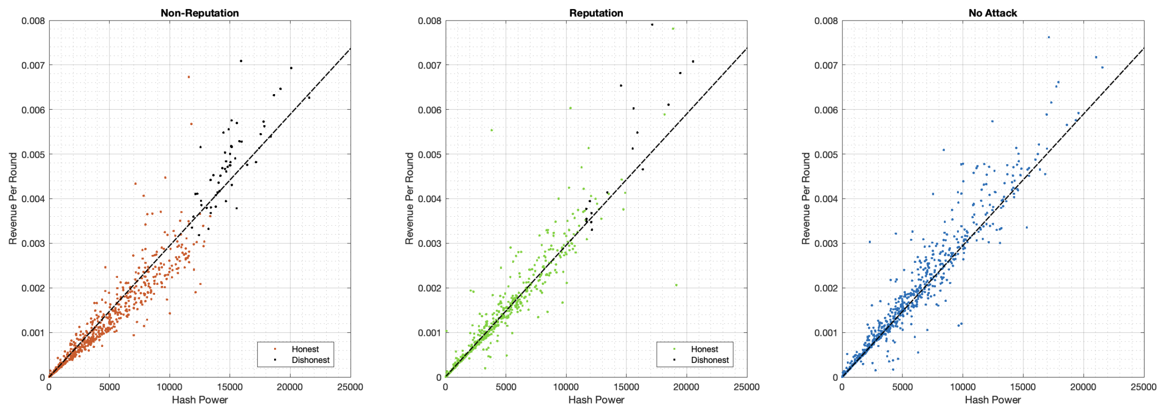


Figure 4.12: Average Revenue Per Round for All Miners from Victim Pool MME-935

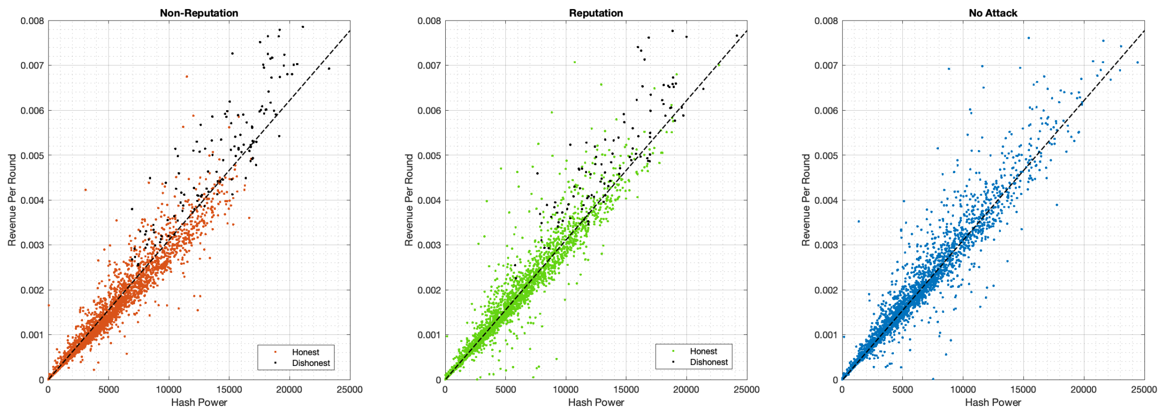


Figure 4.13: Average Revenue Per Round for All Miners from Victim Pool UFR-774

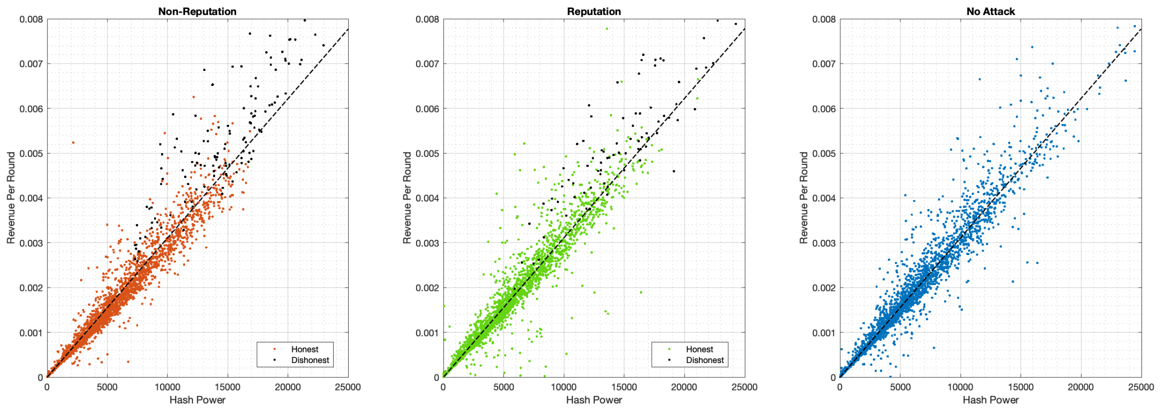


Figure 4.14: Average Revenue Per Round for All Miners from Victim Pool VPK-703

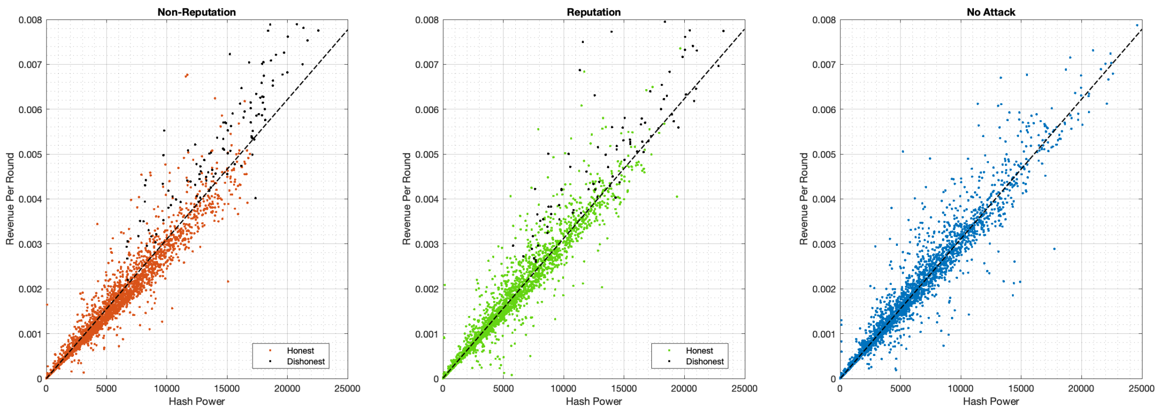


Figure 4.15: Average Revenue Per Round for All Miners from Victim pool WSQ-559

Figure 4.6 demonstrates the ratio between the detected attacks and all attacks in the Reputation mode. The bar heights in Figure 4.5 correlate with the attack detection success ratio shown in Figure 4.6. It is evident that as the detection ratio increases, the number of attacks also decrease. From a different perspective, Figure 4.7 demonstrates that as the number of attacks decrease, the growth rate of the total amount of bribes received by the block withholding miners drops.

The basic statistical analysis for the revenue of all miners are presented in Table 4.6. The first row of the table demonstrates the square error between the actual POW and the expected POW for all miners in each mode. The rest of table shows

Table 4.6: The summary of the Calculated Statistics for Miners

Description		Non-Reputation	Reputation	No Attack
Square Error for Actual POW and expected POW for All Miners		26.3	10.1	8.1
Revenue for All Miners from all Pools (Including Dishonest Revenue)	Mean	139.2	143.8	132.1
	Median	11.7	11.1	10.7
	Standard Deviation	222.9	222.8	209.7
Revenue for All Honest Miners from the Victim Pools	Mean	107.5	123	131.8
	Median	9.2	9.7	10.5
	Standard Deviation	162.2	184.9	209.2
Dishonest Revenue for Block Withholder Miners from the Victim Pools	Mean	141.8	46.4	-
	Median	105.6	39.8	-
	Standard Deviation	109	35.7	-
Revenue for All Miners from the Suspect Pools	Mean	168.4	149.7	132.8
	Median	13.7	12.2	10.8
	Standard Deviation	256.7	232.7	10.8

the values for mean, median and standard deviation for all miners, honest miners from the victim pools, dishonest miners from the victim pools and the miners from the suspect pools respectively.

The scatter plots presented in Figure 4.8 through 4.15, show the distribution of the revenue for individual miners throughout the life of the simulation. The scatter plots are presented for miners from each pool individually. These plots highlight the changes in revenue for the whole group of miners in all 3 modes. The straight line in these plots represents the the linear trend in No Attack mode. The same line on the other two modes highlights the revenue shift resulting from block withholding attacks conducted by dishonest miners.

4.2.2 The Analysis of the Results

1. The effect of the Reputation-based model on the distribution of POW: As demonstrated in Figure 4.5 to 4.7, the number of block withholding attacks

are reduced as the detection ratio increases and consequently, the dishonest revenue for the entire network reduces. In addition, Figure 4.4 showcases the difference between the actual and the expected number of POWs for all pools and how the revenue for all pools becomes more stable and predictable. As the attack detection success ratio improves, the miners are more incentivized to commit to honest mining strategies. As it is shown in Figure 4.6, the attack detection success ratio is stabilized around round 100,000. This is due to the fact that as the number of mining rounds increases, the predictability of potential attacks improves. This stability also has a relationship with the selected confidence intervals. A large number of trials are required for an effective and precise attack detection.

2. The effect of the Reputation-based model on the distribution of Revenue: The statistical measurements presented in Table 4.6, show that the reputation-based model significantly improved the revenue of honest miners significantly. It is also evident that the dishonest revenue is significantly decreased by the reputation-based model. However, when we observe the measurements for all miners, it shows slight fluctuations in the median and mean values and no significant change in the standard deviation. This indicates that the reputation-based model does not significantly change the statistical attributes for the revenue of the whole system. When we compare the No Attack mode with the other two modes, it is evident that the distribution of revenue is slightly more balanced in the No Attack mode as its standard deviation value is smaller. Note that the large standard deviation values for the revenue are predominantly the result of an unbalanced distribution of hash power among the miners. In our simulation, a small percentage of miners own a relatively large percentage of the total hash power. This will directly cause an unbalanced distribution of revenue among miners regardless of block withholding attacks

In figure 4.8 to 4.15 it is observable that the reputation-based model significantly improves the predictability and reliability of the revenue distribution for both suspect and victim pools as the actual distribution of revenue among miners resembles a similarity between Reputation and No-Attack modes. On the other hand, the revenue distribution in the Non-Reputation mode is predominantly above the straight line for the suspect pool and below the straight line for the victim pools.

3. The False Attack Detection Observation: As it is shown in Table 4.2, there are 23 false positive attack detection cases. In these cases, the miners are falsely detected for block withholding attacks where they had committed no attack. The difference between their expected and actual POW was only a result of chance. We can reduce this effect by increasing the test confidence parameter and by increasing the number of trials per interval. However, this will negatively effect the detection success rate and the number of undetected attacks will increase. Practically, it is not feasible to rely on this statistic-based test with 100% certainty of not having false positives. If such a goal is desired, other attack detection methods are needed in conjunction with the probability-based methods.

4.3 SUMMARY

We performed our simulation program in 3 different modes namely, Non-Reputation, Reputation and No-Attack mode, and each mode consisted of 250,000 rounds of mining. The results show that the Reputation-based model can significantly reduce the number of block withholding attacks and consequently, the distribution of revenue among miners becomes more predictable and reliable. We showed that the statistical method used for the detection of a block withholding attack can result in a small percentage of false positive cases and other detection methods need to be incorporated in order to resolve this issue. Further research is needed in order to examine the effectiveness of the reputation-based model on other mining attacks such as selfish mining and eclipsing.

CHAPTER 5
CONCLUSION AND THE FUTURE DIRECTION

In this thesis we discussed the mining attacks their available countermeasures. We reviewed the proposed reputation-based model that is designed to incentivize the miner to commit to honest strategies. Then, we presented our mining simulation in order to evaluate the effectiveness of the proposed reputation-based model. We designed our simulation in a way that resembles the real mining environment. We did this by considering various variable parameters such as total hash power and price of the cryptocurrency and we programmed miners and pool managers in a way that each individual had a unique set of characteristics. We implemented block withholding attacks in our simulation to determine if the reputation-based model could reduce the number of block withholding attacks. The reputation-based model is designed to effect all kinds of mining attacks, however, for our experiment we implemented block withholding attacks due to their simplicity relative to other attacks. We performed our simulation program in 3 different modes namely, Non-Reputation, Reputation, and No-Attack mode and each mode consisted of 250,000 rounds of mining. The results show that the Reputation-based model can significantly reduce the number of block withholding attacks and consequently, the distribution of revenue among miners becomes more predictable and reliable. We showed that the statistical method used for the detection of a block withholding attack can result in a small percentage of false positive cases. In order to increase the reliability and performance of attack detection, other detection methods need to be incorporated. Even though our experiment demonstrate that the reputation-based model can effectively decrease the number of block withholding attacks in mining pools, it is not known how it will perform against other mining attacks. Since many mining attacks such as selfish mining, routing attacks, and eclipsing are more complex in nature, a more sophisticated mining simulation is needed to conduct a similar experiment to the one we performed in this research. In addition, the reputation-based model relies heavily on the effectiveness of the attack detection methods. In order to achieve a comprehensive reputation-based solution, reliable and effective attack detection solutions for all kinds of mining attacks are required. The future research projects should focus on these developments and improvements.

BIBLIOGRAPHY

[1] M. Nojoumian, A. Golchubian, L. Njilla, K. Kwiat, and C. Kamhoua, "Incentivizing blockchain miners to avoid dishonest mining strategies by a reputation-based paradigm," in *Science and Information Conference*, pp. 1118–1134, Springer, 2018.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," tech. rep., Manubot, 2019.

[3] "Bitcoin market capitalization." `https://www.blockchain.com/charts/market-cap`, 2021. Accessed: February, 23 2021.

[4] J. Al-Jaroodi and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.

[5] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proceedings First International Conference on Peer-to-Peer Computing*, pp. 101–102, IEEE, 2001.

[6] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-theoretic analysis of ddos attacks against bitcoin mining pools," in *International Conference on Financial Cryptography and Data Security*, pp. 72–86, Springer, 2014.

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

[8] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 906–917, 2012.

[9] "Securing your wallet." `https://bitcoin.org/en/secure-your-wallet#online`, 2018. Accessed: December, 6 2020.

[10] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack: Analysis and mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1967–1978, 2016.

[11] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *arXiv preprint arXiv:1112.4980*, 2011.

[12] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, "A deep dive into blockchain selfish mining," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[13] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” in *International conference on financial cryptography and data security*, pp. 436–454, Springer, 2014.

[14] S. Seebacher and R. Schüritz, “Blockchain technology as an enabler of service systems: A structured literature review,” in *International Conference on Exploring Services Science*, pp. 12–23, Springer, 2017.

[15] M. Pryanikov and A. Chugunov, “Blockchain as the communication basis for the digital economy development: advantages and problems,” *International journal of open information technologies*, vol. 5, no. 6, pp. 49–55, 2017.

[16] S. Raval, *Decentralized applications: harnessing Bitcoin’s blockchain technology.* ” O’Reilly Media, Inc.”, 2016.

[17] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction.* Princeton University Press, 2016.

[18] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, “Bitcoin mining pools: A cooperative game theoretic analysis,” in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 919–927, 2015.

[19] A. Laszka, B. Johnson, and J. Grossklags, “When bitcoin mining pools run dry,” in *International Conference on Financial Cryptography and Data Security*, pp. 63–77, Springer, 2015.

[20] I. Eyal, “The miner’s dilemma,” in *2015 IEEE Symposium on Security and Privacy*, pp. 89–103, IEEE, 2015.

[21] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, “On power splitting games in distributed computation: The case of bitcoin pooled mining,” in *2015 IEEE 28th Computer Security Foundations Symposium*, pp. 397–411, IEEE, 2015.

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

[23] K. Nayak, S. Kumar, A. Miller, and E. Shi, “Stubborn mining: Generalizing selfish mining and combining with an eclipse attack,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 305–320, IEEE, 2016.

[24] C. Grunspan and R. Pérez-Marco, “On profitability of stubborn mining,” *arXiv preprint arXiv:1808.01041*, 2018.

[25] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 692–705, 2015.

[26] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 129–144, 2015.

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

[28] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 375–392, IEEE, 2017.

[29] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen, "Partitioning attacks on bitcoin: Colliding space, time, and logic," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1175–1187, IEEE, 2019.

[30] S. K. Singh, M. M. Salim, M. Cho, J. Cha, Y. Pan, and J. H. Park, "Smart contract-based pool hopping attack prevention for blockchain networks," *Symmetry*, vol. 11, no. 7, p. 941, 2019.

[31] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 195–209, 2017.

[32] S. Bag and K. Sakurai, "Yet another note on block withholding attack on bitcoin mining pools," in *International Conference on Information Security*, pp. 167–180, Springer, 2016.

[33] S. Lee and S. Kim, "Countering block withholding attack efficiently," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 330–335, IEEE, 2019.

[34] S. Solat and M. Potop-Butucaru, "Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin," *arXiv preprint arXiv:1605.02435*, 2016.

[35] E. Heilman, "One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner," in *International Conference on Financial Cryptography and Data Security*, pp. 161–162, Springer, 2014.

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

[37] S. Solat and M. Potop-Butucaru, "Zeroblock: Preventing selfish mining in bitcoin. corr, abs/1605.02435," 2016.

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

[39] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, pp. 203–226, 2019.

[40] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE symposium on security and privacy*, pp. 104–121, IEEE, 2015.

[41] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 299–314, 2002.

[42] A. Singh *et al.*, "Eclipse attacks on overlay networks: Threats and defenses," in *In IEEE INFOCOM*, Citeseer, 2006.

[43] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against eclipse attacks on overlay networks," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, pp. 21–es, 2004.

[44] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *International Workshop on Peer-to-Peer Systems*, pp. 261–269, Springer, 2002.

[45] E. Anceaume, Y. Busnel, and S. Gambs, "On the power of the adversary to solve the node sampling problem," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XI*, pp. 102–126, Springer, 2013.

[46] A. Bakker and M. Van Steen, "Puppetcast: A secure peer sampling protocol," in *2008 European Conference on Computer Network Defense*, pp. 3–10, IEEE, 2008.

[47] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

[48] G. P. Jesi, A. Montresor, and M. van Steen, "Secure peer sampling," *Computer Networks*, vol. 54, no. 12, pp. 2086–2098, 2010.

[49] M. Belotti, S. Kirati, and S. Secci, "Bitcoin pool-hopping detection," in *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, pp. 1–6, IEEE, 2018.

[50] M. Salimitari, M. Chatterjee, M. Yuksel, and E. Pasiliao, "Profit maximization for bitcoin pool mining: A prospect theoretic approach," in *2017 IEEE 3rd international conference on collaboration and internet computing (CIC)*, pp. 267–274, IEEE, 2017.

[51] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "Smartpool: Practical decentralized pooled mining," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 1409–1426, 2017.

[52] D. Freeman, T. McWilliams, S. Bhattacharyya, C. Hall, and P. Peillard, "Enhancing trust in the cryptocurrency marketplace: A reputation scoring approach," *SMU Data Science Review*, vol. 1, no. 3, p. 5, 2018.

[53] D. Carboni, "Feedback based reputation on top of the bitcoin blockchain," *arXiv preprint arXiv:1502.01504*, 2015.

[54] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of reputation: a reputation-based consensus protocol for blockchain based systems," in *Proceedings of the 2019 International Electronics Communication Conference*, pp. 131–138, 2019.

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

[56] Y. Sun, R. Xue, R. Zhang, Q. Su, and S. Gao, "Rtchain: A reputation system with transaction and consensus incentives for e-commerce blockchain," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–24, 2020.

[57] M. Nojoumian, "Rational trust modeling," in *International Conference on Decision and Game Theory for Security*, pp. 418–431, Springer, 2018.

[58] M. Nojoumian and T. C. Lethbridge, "A new approach for the trust calculation in social networks," in *E-business and Telecommunication Networks: 3rd International Conference on E-Business, Best Papers*, vol. 9 of *CCIS*, pp. 64–77, Springer, 2008.

[59] M. Nojoumian, *Novel Secret Sharing and Commitment Schemes for Cryptographic Applications.* PhD thesis, Department of Computer Science, University of Waterloo, Canada, 2012.