# On Power Splitting Games in Distributed Computation: The Case of Bitcoin Pooled Mining

Loi Luu*, Ratul Saha*, Inian Parameshwaran*, Prateek Saxena*, Aquinas Hobor*†
*School of Computing, †Yale-NUS College, National University of Singapore
{loiluu, ratul, inian, prateeks, hobor}@comp.nus.edu.sg

February 24, 2015

***Abstract*—Several new services incentivize clients to compete in solving large computation tasks in exchange for financial rewards. This model of competitive distributed computation enables every user connected to the Internet to participate in a game in which he splits his computation power among a set of competing pools — the game is called a computation power splitting game. We formally model this game and show its utility in analyzing the security of pool protocols that dictate how financial rewards are shared among the members of a pool.**

**As a case study, we analyze the Bitcoin cryptocurrency which attracts computing power roughly equivalent to billions of desktop machines, over $70\%$ of which is organized into public pools. We show that existing pool reward sharing protocols are insecure in our game-theoretic analysis, when considering a single attack strategy called the "block withholding attack". This attack is a topic of debate, initially thought to be ill-incentivized in today's pool protocols: *i.e.*, causing a net loss to the attacker, and later argued to be always profitable. Our analysis shows that the attack is always well-incentivized in the long-run, but may not be so for a short duration. This implies that existing pool protocols are insecure, and if the attack is conducted systematically, Bitcoin pools could lose millions of dollars worth in months. The equilibrium state is a mixed strategy—that is—in equilibrium all clients are incentivized to probabilistically attack to maximize their payoffs rather than participate honestly. As a result, the overall capacity of the Bitcoin network is under-utilized.**

## I. Introduction

Distributed computation enables solving large computational problems by harnessing the availability of machines connected to the Internet. A new paradigm of distributed computation is emerging wherein participants contribute their computing resources in exchange for direct financial gain or monetary compensation. In this paradigm, all participants *compete* in performing computation tasks to obtain rewards. We call such computation *competitive distributed computation*. There are many examples of such competitive distributed computation today. Public challenges for testing the strength of cryptographic constructions (e.g., the RSA Secret-Key challenge [1]) invites participants to find and exploit weaknesses using huge computational resources in exchange for monetary prizes. Crowd-sourced security testing of applications is an emerging commercial industry (c.f. BugCrowd [2], CrowdCurity [3], the HeartBleed challenge [4]), wherein computation is dedicated to penetration testing tasks in software. Here, bug bounties are offered to the first participant that finds exploitable bugs. Perhaps one of the most direct examples of competitive distributed computation are cryptocurrencies, such as Bitcoin, which attract computation power equivalent to nearly a billion desktop computers. In cryptocurrencies, participants—often called *miners*—solve cryptographic puzzles as "proof-of-work" [5] in exchange for obtaining rewards in cryptocurrency coins.

Distributed computation scales by incentivizing large number of participants to contribute their computation power. When the computation problems demand high resources, participants resort to *pooling* their resources together in the competition. This is both natural and useful as it reduces the uncertainity or variance in obtaining rewards for the pool participants. Typically, such computation pools have a designated *supervisor* who is responsible for distributing computation sub-tasks to users and distributing the reward obtained from winning the competition. When such delegation of computation tasks is in place, the question of designing fair *pool protocols*—which ensure that each participant get paid for the computation they perform and *only* for the computation they perform—become important.

Problems with designing secure or fair pool protocols are relatively less explored, especially in the setting of competitive distributed computation. Previous work have investigated solutions to prevent participants from specific forms of cheating, often considering a single supervisor system [6]. Indeed, for example in Bitcoin pool protocols, techniques for preventing misbehaving clients in a single pool are known and have been implemented. For instance, solutions preventing unfair supervisors, lazy clients who claim more than what they have done, and hoarders that keep the results secret to gain extra reward from it (e.g., by gaining lead time for another competition) are known [6, 7, 8, 9]. Many other systems such as the SETI@home project use redundancy to check for mismatch in replicated computation tasks [10]. However, a generic model for security analysis of pool protocols when there are multiple competing supervisors is a subject of open investigation.

There are some unique characteristics of competitive distributed computation that makes designing secure pool protocols difficult. First, solving the computation task is competitive. The first supervisor publishing the valuable results gets the reward, and others get nothing. Here, the competition game is zero-sum and timing is critically sensitive. For example, in the RSA Secret-Key Challenge [1], a client once finds

a possible plaintext should submit to the group supervisor immediately to claim for the reward, otherwise other groups may find it and make the result obsolete. Second, the computation tasks can be delegated to anonymous participants—in fact, the primary function of pool operators is to securely delegate tasks. This opens up analysis of the incentives of the participant which can decide to split its computation across multiple supervisors. Protocols that may be secure in a single supervisor setting [6] (e.g. with no delegation) are often used in practice, but can turn out to be insecure in multiple-supervisor setting. Detecting if and how a participant splits its power is difficult since participants are anonymous or can form a large sybil sub-network.

***The Computing Power Splitting Game.*** In this paper, we introduce a new distributed computation model which includes *multiple* supervisors competing with each other to solve computationally large problems. Participants with computation power play a game of solving computation problems by acting as a supervisor or joining other pools, which we call as the *Computing Power Splitting* game or the CPS game. Participants have the choice to contribute their power to one supervisor's pool or anonymously spread it across many pools. Each participant can choose to either follow the pool protocol honestly or deviate from it arbitrarily. The goal of each participant is to maximize its expected profits. A pool protocol is secure with respect to the CPS game if following the protocol maximizes each participant's profit. We show an example analysis of the CPS game in the this paper, to illustrate how it acts as a powerful tool in analyzing protocols in competitive distributed computation scenarios.

***The Case Study on Bitcoin Network.*** Bitcoin [5] is the largest cryptocurrency reaching a market capitalization of over $5.5$ billion US dollars in $2014$ [11]. Bitcoin is representative of over $50$ new cryptocurrencies or alt-coins which have a similar structure. In Bitcoin, each participating client (or miner) contributes computation power to solve cryptographic puzzles in a process called *block mining*, which acts as the basis for minting coins (Bitcoins). The computational resources required for Bitcoin mining increases over time and is already significant: finding a block in late 2014 requires computing about $2^{70} \approx 10^{21}$ SHA-256 hashes; the Bitcoin network as a whole finds a block approximately once every 10 minutes. Since the computational difficulty is high, most users join *mining pools*, where they aggregate their computing resources into a pool and share the reward. Pooled mining constitutes $72\%$ of the Bitcoin computation network today.

Bitcoin pools are a direct example of competitive distributed computation. In each round of mining (which roughly takes 10 minutes), pools compete to solve the puzzle and the first one to solve claims a set of newly minted Bitcoins. This can be viewed as the CPS game. Each pool has a designated pool supervisor or operator, who then distributes the earned rewards among pool members using a pool protocol. Existing pool protocols are designed carefully to block several attacks from its anonymous miners [12]. For instance, the pool protocol ensures that all blocks found by miners can only be reported via the pool operator, thereby ensuring that a lucky miner cannot claim the rewards directly from the network. However, the answer to the question—*does a miner maximize its profit by following the pool protocol honestly?*—is not yet known.

***Findings.*** In our case study, we investigate the utility of one cheating technique called *block withholding* using the CPS game formulation of Bitcoin. In a *Block Withholding (or BWH)* attack, when a miner finds a winning solution, he does not submit it to the pool, nor can he directly submit it to the Bitcoin network. Instead, he simply withholds the finding, thereby undermining the overall earnings of all miners in the victim pool, including himself. Existing pool protocols are secure against this attack when one considers a single pool in the system. However, when we carefully analyze the existing popular pool protocols using our CPS-game formulation, we find that it is insecure. Specifically, we establish that rational miners are well-incentivized to withhold blocks and earn higher profits by being dishonest. Further, a sybil network of dishonest miners can cost pool operators large fraction of their earnings (often millions of US Dollars per month). Further, this finding implies that big pools can dominate the Bitcoin network by carrying out the BWH attack on new or smaller pools, yet earning more reward than mining honestly by themselves. We study the damage a set of miners (say of one pool) can cause to another pool, and the conditions under which such behavior is well-incentivized. This makes BWH a real threat to the viability of pooled mining with existing protocols in cryptocurrencies. We further show that this game has no Nash equillibrium with pure strategies; in fact, this implies that the pure strategy of all players being honest is not a Nash equillibrium. As a result, in the equillibrium state all miners are devoting some fraction of computation for witholding rather than mining honestly, and therefore the network as a whole is under-utilized.

We point out that witholding attacks are well-known, but their efficacy is a topic of hot debate on Bitcoin forums [13, 14, 15] and recent papers [16, 17]. Intuition and popular belief suggests that these attacks are ill-incentivized [13, 14, 15] because in a single pool game, the attacker strictly loses parts of their profits by witholding. However, we study the incentives with respect to the general CPS game, in which we show existing pool protocols to be insecure. The profitability explains why one such real attack conducted on a Bitcoin pool in April $2014$ could indeed be well-incentivized, though pool operators claimed that such attacks have no incentives for attackers [18]. The attack caused nearly $200{,}000$ USD in damage to the victim pool. We further study whether the attacks are profitable over a short period of time or over a long period of time, and under what conditions.

Finally, we initiate a study on effective strategies to achieve secure protocols in CPS games, specifically in the context of Bitcoin. We discuss several public proposals to mitigate these attacks in § VI. We recognize that for a defense to become immediately practical on the existing Bitcoin network, it should be non-intrusive, i.e., should require no incompatibility with the existing Bitcoin protocol—however, our conclusion in achieving a secure solution is still an open problem worthy of future work. Finally, we hope that our work provides a building block for designing cryptocurrencies which support

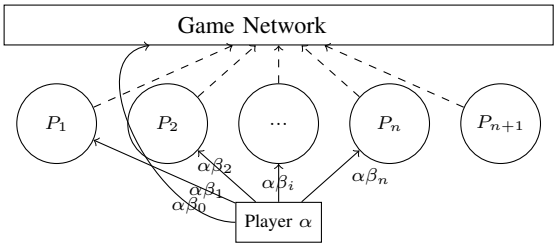


Fig. 1: The CPS game setup of $n+1$ pools with respect to a player with $\alpha$ fraction of total computing power in the game.

pooled mining natively in their core protocol, unlike Bitcoin.

***Contributions.*** To summarize, this work makes the following main contributions:

- *Computing Power Splitting Game*. We give a novel formulation as a CPS game for a new model of distributed computation, in which multiple supervisors compete with each other. The CPS game precisely captures a number of real world applications. We expect that our model will enable more research questions regarding designing a secure protocol.
- *Analysis of BWH in Bitcoin pooled mining*. Applying our novel CPS model allows us to systematically study the security of pooled mining protocols in Bitcoin. We explain why block withholding is well-incentivized for rational miners, providing an algorithmic strategy to gain higher rewards than honest mining. We confirm our findings by experiments running real Bitcoin miners and pools on Amazon EC2.

## II. The Computing Power Splitting Game

A player with non-zero computing power naturally has the ability to choose among different ways of multihoming among pools accessible to him. We intend to analyze strategies for such a player to distribute his power into different pools such that his net reward is maximum. We formulate this problem as a multi-player game where each player independently and anonymously participates in.

A pool is *accessible* (*inaccessible*) to the player if he can (cannot) anonymously join the pool. For simplicity, from now on we consider all inaccessible pools to be grouped into a single inaccessible pool.

The *Computing Power Splitting (CPS)* game consists of:

- **Computationally Difficult Problem $T$:** A problem that requires a large amount of computation to solve.
- **Partition function $\phi(T) \rightarrow \{T_1, T_2, .., T_n\}$:** The function $\phi(T)$ splits $T$ into many smaller tasks $T_i$, such that the difficulty of solving $T$ is equivalent to the total difficulty of solving all $T_i$. For example, in the RSA Secret-Key challenge, the key value space $X$ is split into various $X_i$, each of which covers a specific range of the key space and will be delegated to some particular client to perform the search. Similarly, in crowd-source scanner, each client will scan some particular set of program paths in a program with an exponential number of paths, to check if one is exploitable.
- **Players:** A client with positive computing power is a player in this game who has a fraction of the total network computing power. In this game, we study the behavior of a miner who has a specific (say) $\alpha$ fraction of the total computing power.
- **Information available to all players:** There is a finite set of $n+1$ pools $\mathcal{P} = \{P_1, P_2, \ldots, P_n\} \cup \{P_{n+1}\}$ where $P_1, P_2, \ldots, P_n$ are accessible pools and $P_{n+1}$ is the inaccessible one to the player. The computing power function $cp : \mathcal{P} \rightarrow (0, 1]$ describes the power of each pool as a fraction of the total computing power in the game. Thus, the total computing power in the game is $\alpha + \sum_{i=1}^{n+1} cp(P_i) = 1$.
- **Actions:** For any particular player, a *Strategy Distribution Vector (SDV)* $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$ is defined such that (i) the player plays privately with computing power $\alpha\beta_0$, (ii) for each $i \in \{1, \ldots, n\}$, the player mines in pool $P_i$ with contribution $\alpha\beta_i$ power of the whole network, (iii) $0 \leq \beta_i \leq 1$ for each $i \in \{0, 1, \ldots, n\}$ and $\sum_{i=0}^{n} \beta_i = 1$. The player moves by choosing an SDV $\vec{\beta}$ for one game. For simplicity, we assume that $\vec{\beta}$ remains constant for all players in one game.
- **Payoff Scheme:** There is a payoff distribution scheme applied in a pool where, irrespective of the internal implementation, an individual player's payoff is proportional to the number of smaller tasks $T_i$ that he has solved.
- **Utility:** Let $U_i$ denote the random variable describing the reward the player receives from pool $P_i$ by playing for one game with $\beta_i$ fraction of his power $\alpha$. Let $R = \sum_{i=1}^{n} U_i$ denote the random variable representing total reward for one game for the player. The player's goal is to maximize $\mathbb{E}(R)$, the expected reward.

The CPS game formulation enables us to study a variety of attack strategies that a player can carry out to maximize his profit. Specifically, as a case study in this work, we present a new strategy to utilize the *block withholding attack* in Bitcoin pooled mining, that is always more rewarding than mining honestly. Under this attack, a number of pools suffer financial losses whereas the attacker gains a better reward than from the honest strategy.

***Assumptions for the CPS game.*** For simplicity, we make the following realistic assumptions for the CPS game:

- *A1. Other Players are Honest*. The attacker is the only rational player and hence is carrying out some attack. The rest of the miners will pick the honest strategy, i.e, following the protocol. We also discuss effects of relaxing this assumption. Nonetheless, we assume all other players are honest unless explicitly specified.
- *A2. Known Power Distribution*. The computing power distribution of the game—including the accessibility and payoff mechanism of all the pools—is correctly estimated by the attacker at the start of the game. This is a fairly practical assumption. For example, in Bitcoin, most of the information is publicly available [19] and also easy to be estimated by listening on the Bitcoin network for a small period of time.
- *A3. Constant power distribution throughout the game.* We

assume that the network state stays constant throughout the game. In practice, if there is significant variation in one game, it can be analyzed as multiple smaller games.
- *A4. Independence of games*. We assume that the reward and the winner in one game have no effect on that in the other. More specifically, finding a solution in a game does not yield any advantage in wining any subsequent ones.

We do not make any assumption about other properties of the game state, e.g., the total mining power or if the problem difficulty remains constant across games. In fact, in Section V, in the case study of Bitcoin pooled mining, we show that these factors do not affect our analysis results significantly. We also demonstrate that our analysis results for Bitcoin pooled mining still hold without the assumptions *A2, A3*.

## III. A CASE STUDY OF BITCOIN POOLED MINING

### A. Background

***Mining Bitcoins.*** Unlike traditional monetary systems, Bitcoin is a decentralized crytocurrency with no central authority to issue fiat currency [20]. In Bitcoin, the history of transactions between users is stored in a global data structure called the *blockchain*, which acts as a public ledger of who owns what. Users perform two key functions: (i) verifying newly spent transactions and (ii) creating a new block (or proof-of-work) to include these transactions. In the Bitcoin protocol, both these functions are achieved via an operation called *mining*, in which a *miner* validates the new transactions broadcasted from other users and in addition solves a computational puzzle to demonstrate a proof-of-work [21], which is then verified by a majority consensus protocol [5]. The first miner to demonstrate a valid proof-of-work is said to have "found a block" and is rewarded a new set of minted coins, which works as an incentive to continue mining for blocks.

In terms of a CPS game, the computationally large problem $T$ in the Bitcoin protocol is based on the pre-image resistance of a cryptographic hash function SHA-256 [22]. Specifically, the puzzle involves finding a value whose hash begins with some zero bits derived from a variable $D$, which represents the global network difficulty. For each block, this value includes the already computed hash for the previous block, information of some transactions and a nonce — the miner's goal is to find a suitable nonce such that the hash of the corresponding block has at least $f(D)$ leading zeros [23]. The network self-adjusts $D$ after every $2016$ blocks found, such that the time to find a valid block is roughly $10$ minutes. Relating to the CPS game model, the search space $X$ of $T$ has the size $|X| = 2^{|f(D)|}$. At present, $f(D)$ is roughly $70$, thus the average *hashrate* (H/s—the number of SHA-256 hash computations per second) required to find a block in $10$ minutes is around $1.96 \times 10^{18}$ H/s. One can verify that with a standard computer having a hashrate of $1$ million H/s, a miner has to mine for on an average of $62{,}000$ years to find a block.

***Pooled mining.*** The probability of an individual miner to find a new block every $10$ minutes is excruciatingly small, which led miners to combine their computing power into a group or *pool*. If anyone in the pool finds a block, the block reward is split among members according to their contributed processing power. This shared mining approach is called *pooled mining*, which effectively reduces the uncertainty or "variance" in the reward for individual miners [12]. Typically, the pool operates by asking its miners to solve easier problems $T_i$ with a smaller difficulty $d$ ($d < D$) whose solution, called *shares*, has probability $\frac{d}{D}$ to be the solution for the new block. Shares do not have any real value other than acting as the main reference when distributing the reward. For example, instead of searching in a space of size $|X| = 2^{70}$, the pooled miners only need to search in a smaller space of size $|X_i| = 2^{40}$, i.e., finding hashes with 40 leading zero bits. Every block is trivially a valid share, because a hash value with $70$ leading zeros also has $40$ leading zeros—however, the probability of a share being a block is $1/2^{30}$.

When a member in a pool finds a share that is also a valid block, the pool operator submits it to the Bitcoin blockchain and distributes the claimed reward to all miners in the pool. The pool protocol ensures that the work is distributed in a way which prevents miners from directly claiming rewards for found blocks, thereby forcing all rewards to be funneled through the pool operator.

***Payoff schemes in pooled mining.*** There are multiple approaches to design a fair reward distribution system in pooled mining [12]. Some of the popular schemes include (i) Pay-per-share (PPS)—where the expected reward per share is paid, (ii) Pay-per-last-N-shares (PPLNS)—the last $N$ submitted shares are considered for payment when a block is found. While there are differences among these schemes (and their variations), all of them aim to distribute the reward such that the payoff of an individual miner is proportional to the number of shares he submitted, which in turn is proportional to his individual computing power contributed to the pool.

The main question we study is, given a payoff scheme, does the miner have incentive to follow the *honest mining* strategy—i.e., to honestly contribute all of his available power to pools to maximize his profit? Intuitively, if all pools employ fair protocols and if the miner contributes his complete power to one or more of them, he should receive rewards proportional to his true computing power. We study the correctness of this intuition and whether the attacker can systematically exploit mining pools to extract higher profits.

### B. Block Withholding (BWH) attack

Our focus in this paper is on studying the efficacy of one attack strategy called block withholding to gain more reward. When a pool is under BWH attack, the attacker submits all shares he computes to the pool except shares which are also valid blocks. Since these withheld blocks would have directly translated into rewards for the pool, such an attack decreases the overall profit of the pool, thereby decreasing the reward for all individual miners in the pool including the attacker. For example, later in § IV-E, we analytically and experimentally show that miners in a pool with $25\%$ of the total computing power in the Bitcoin network will lose $10.31\%$ of their reward, if $20\%$ of the pool carries out the BWH attack. Therefore, a naive intuition may suggest that miners do not have any

incentive to conduct such an attack. We claim that this intuition is against the rational choice for any miner.

To see if BWH attack is well-incentivized, we consider the two extreme options for a miner—(i) to withhold all blocks or (ii) to submit all found blocks honestly on a pool. In practice, the attacker may withhold some of the blocks he finds and our analysis can be easily extended to model this degree of withholding behavior. With BWH attackers present, the overall efficiency of a pool is no longer proportional to its miners' actual total computing power—i.e. the overall reward generated by the pool is proportional only to the computing power contributed by honest miners. Nonetheless, the reward earned is shared equitably with all miners, proportional to their submitted shares. This imbalance allows a miner to collect (reduced) reward even from pools in which he withholds.

***The Block Withholding Attack in the CPS game.*** To systematically study the attacker's advantage, we define a version of the CPS game called the CPS-BWH game. Specifically, the following extension is made to the generic CPS game:

- When the attacker makes a move, in addition to choosing the distribution $\vec{\beta}$ of his own computing power, he also decides which pools to withhold in—denoted by the *attack vector* $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$. The attack vector is chosen such that $\gamma_i = 1$ if the attacker withholds *all* blocks he finds in pool $P_i$ and $\gamma_i = 0$ otherwise.
- All the assumptions stated in *A1 — A4* are valid.

The goal is to find the optimal SDV (say) $\vec{\beta_a}$ and the attack vector (say) $\vec{\gamma_a}$, such that the expected gain over honest mining is maximum. Let $R_h$ denotes the expected reward with the honest mining strategy, i.e., attack vector $\vec{0}$. Similarly, let $R$ denotes the expected reward with the attack vector $\vec{\gamma_a}$. Our goal is to maximize the expected gain, defined as

$$\Delta_R = \frac{R - R_h}{R_h}.$$

***Incentive for BWH attack.*** The main insight which incentivizes the BWH attack is that Bitcoin mining is a *zero-sum* game, i.e. to find a block all pools compete and exactly one pool wins by consensus, all others do not get any reward. In this game, although the attacker's reward drops in the pool being victimized, this loss could be compensated from the reward gained from other pools in which the attacker mines honestly. The victim pool's loss due to withholding translates into better rewards for other pools competing in the game, since pools with no withholding miners have a competitive advantage of being rewarded a block. Thus, if the attacker mines only on the victim pool, he will definitely share the loss with the pool and earn less reward as in the aforementioned intuition, i.e., the reward protocol is fair and secure in a single supervisor system. However, when he also mines strategically on another pool at the same time, his reward gain from that pool may outweigh his loss on the victim pool and make his overall extra reward positive.

To illustrate this, we show a concrete example (shown in Figure 2) of two pools constituting the Bitcoin network. An attacker with $25\%$ computing power can split his power — $5\%$ to conduct a BWH attack on the first pool, while mining honestly on the second pool with $20\%$. It is clear that the attacker's expected reward from the victim pool falls to $4.67\%$ as intuition suggests. However, the total reward earned by other pool increases, as the first pool's loss shifts to the second pool's gain, and thus the attacker overall makes more reward.

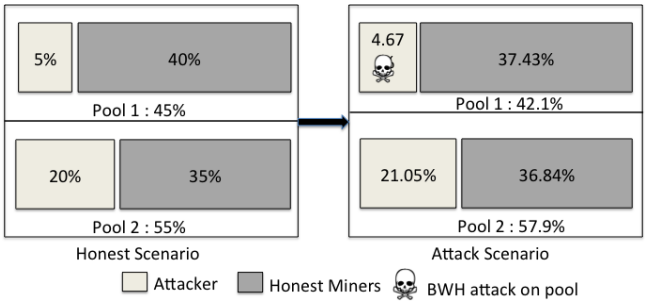


Fig. 2: Simple illustration of the BWH attack: if an attacker with 25% of the mining power of the network attacks Pool 1 with 5% of the network mining power, he gains 25.72% of the reward instead of the expected 25%.

### C. Approach overview

The example above shows the feasibility of a BWH attack. To analyze the scenarios and the extent of damage to pools and honest miners, we study the following research questions in this work.

- *Q1.* We ask whether the attack is well-incentivized regardless of the number of pools, their respective computing power and the attacker's computing power, i.e., whether the attacker always has a strategy to gain more reward than by honest mining.
- *Q2.* Given that the BWH attack is well-incentivized, rational miners will tend to dishonestly attack the pools to gain extra reward. This raises a question whether the attack is still profitable when the pool is "contaminated" by, say, a factor $c$, i.e., the BWH miners account for $c$ fraction of the mining power in the pool.
- *Q3.* We study the best strategy that maximizes the attacker's expected reward when the attacker attacks one or multiple pools.
- *Q4.* We seek the stable equilibrium in Bitcoin when block withholding miners are participating in pooled mining.

To answer these questions, we leverage our CPS-BWH game to study the behavior of miners. In this game, a miner is considered to be a player, and he makes a move by distributing his power to pools in order to maximize his reward. Our theoretical analysis uses the CPS game formulation to address questions *Q1, Q2 and Q3* in several attack scenarios in § IV. To empirically verify our analysis findings, we run experiments in a custom Testnet Bitcoin network on Amazon EC2 using roughly $70,000$ CPU-core-hours for several months with a popular Bitcoin client [24], mining software [25] and pool server software [26]. We answer question *Q4* in § IV-F.

## IV. Block withholding attack Analysis

***Analysis overview.*** We discuss several Block withholding attack scenarios in this section. In what follows, we treat $\mathbb{E}(R)$ as $R$. Our goal is to find an optimal strategy for the attacker

such that his gain in expected reward is maximum. Table I gives an illustrative overview of the attacker gain, given a network distribution before the attack happens (as in Figure 3), in several attack scenarios.

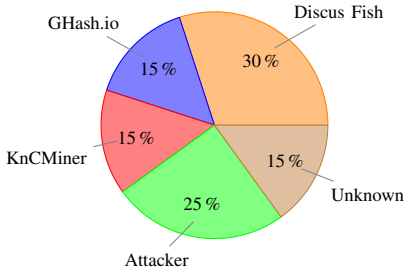


Fig. 3: Mining power distribution before the attack happens. This constructive example is similar to the Bitcoin network state in November 2014 [27].

| § | Scenario | Victim(s) | $R_h$ | $R$ | $\Delta_R$ |
|---|---|---|---|---|---|
| **IV-B** | One pool | Discus Fish | 25.00% | 25.56% | 2.26% |
| **IV-C** | Multiple pools | All, except Unknown | 25.00% | 26.19% | 4.76% |
| **IV-D** | One "contaminated" pool | Discus Fish, 2.5% is contaminated | 25.64% | 25.86% | 0.89% |

TABLE I: Example results of several attack scenarios that we study in this paper given the pool distribution as in Figure 3.

In this section and the rest of the paper, for simplicity, we use the term "private mining" to represent the honest mining part of the attacker, which can be from solo mining or joining a public pool. A careful reader may be concerned about the high variance in solo mining if the attacker's mining power is not large enough. However, it is easy to see that he can avoid it by honestly mining on one pool and carrying out the attack on the target pools to achieve similar expected reward with low variance.

***Experiment setup and goals.*** To support our analysis, we run several experiments in our customized Testnet Bitcoin network using computation resources from Amazon EC2. Each experiment simulates the actual mining in the real Bitcoin network for 2 to 3 months. The detailed report of our experiment setup is described in Appendix B.

We argue that the empirical validation of Bitcoin attacks is essential to check the correctness of our analysis and to show that our CPS game is faithful to the actual Bitcoin mining software implementation. Meanwhile, an algebraic and probabilistic model of computation used in previous work [28] does not capture all the network factors (e.g., geographic placement, latency) and Bitcoin network properties which may considerably affect the validity of the numerical analysis. Thus, to our best knowledge, this work is the first attempt to simulate the exact mining behavior that models the underlying implementation. Moreover, our experiments are motivated by discussions with real Bitcoin pool operators, who suspected that variations in difficulty, distribution of pool power, hashrates, etc., would play a role in the total payoff of the attacker. Our experiments in § V confirm that these intuitive misgivings do not affect results significantly. Thus, our game-theoretic approach is valid to deduce practical results.

### A. Intuition: Bitcoin network as one accessible pool

To demonstrate the intuition behind the BWH attack, we start with a toy attack scenario where the whole Bitcoin network is one large pool accessible to the attacker. We assume that the pool $P_1$ has computing power $cp(P_1) = 1 - \alpha$ and naturally $\gamma_a = (1)$. The attacker is the only rational player in this game, i.e., aware of the BWH strategy and the rest of the players are mining with honest mining strategy. We assume that the attacker attacks with SDV $(1 - \beta, \beta)$, i.e., he mines privately with $\alpha(1-\beta)$ and attacks pool $P_1$ with $\alpha\beta$ fraction of the network computing power. However, if the attacker were mining honestly, the expected reward would have been directly proportional to his computing power, i.e. $R_h = \alpha$, no matter how he chooses the SDV.

In the attack, the fraction of computing power of $P_1$ remains at $(1 - \alpha)$, while the reward generated has to be split with $1 - \alpha(1 - \beta)$ fraction of the network. Since only $(1 - \alpha\beta)$ of the network is actually mining blocks now, the expected reward for the attacker from private mining is

$$R_0 = \frac{\alpha(1-\beta)}{1-\alpha\beta}.$$

For $P_1$, the pool is rewarded $\frac{1-\alpha}{1-\alpha\beta}$ and, on average, the expected reward for the attacker from the pool is

$$R_1 = \frac{1-\alpha}{1-\alpha\beta} \times \frac{\alpha\beta}{1-\alpha(1-\beta)}.$$

Hence the total reward for the attacker is

$$R = R_0 + R_1 = 1 - \frac{(1-\alpha)^2}{(1-\alpha\beta)(1+\alpha\beta-\alpha)}.$$

Comparing the reward after attacking with that of the honest mining, we get

$$\frac{R}{R_h} = \frac{\alpha\beta - \alpha\beta^2 + 1 - \alpha}{(1-\alpha\beta)(1+\alpha\beta-\alpha)},$$

and we prove in Appendix C that $\forall \alpha, \beta \in (0,1)$, $\frac{R}{R_h} > 1$. This shows that regardless of his mining power and the strategy vector, *the attacker always has an incentive* to carry a BWH attack in this particular scenario.

We also prove that for $\beta = 0.5$, the attacker gains maximum relative reward for any $\alpha$ in Appendix C. Thus, by performing the attack, the attacker of power $\alpha$ gains maximum $\Delta_R = R/R_h - 1 = \frac{\alpha-\alpha^2}{(2-\alpha)^2}$ more than the original mining [1]. More specifically, for $\alpha = 0.2$, we have $\Delta_R = 0.05$, i.e. the attacker obtains 5% more than mining with honest strategy. We illustrate the percentage of extra reward that the attacker gains corresponding to his power proportion in Figure 4.

***Experimental evaluation.*** We evaluate our results when $\beta = 0.5$ for $\alpha = 0.2$ and $\alpha = 0.4$. As reported in Section A of Table V, when $\alpha = 0.2$, the attacker receives 20.78% of the network reward, which is close to the 20.98% given by our analysis. Similarly, he receives 43.29% reward, which is 8.2% relatively more than his honest reward, while having only 40% mining power of the network.

[1] Our result differs here from the previous paper [16], because their analysis overestimates $R_1$, thus giving imprecise result

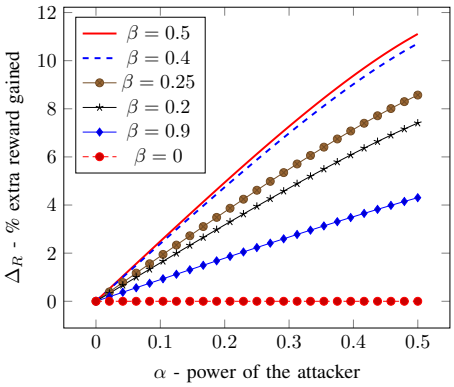


Fig. 4: The attacker's extra reward ($\Delta_R$) in the scenario where the whole network is considered as one public pool. We plot reward gain for several $\beta$ to show that the attacker gains maximum reward when $\beta = 0.5$.

### B. Multiple pools: attack only one victim pool

We have established that if the attacker can access the whole network, then by spending a fraction of his power for withholding, he can gain extra reward. The intuition behind the result is that the loss of the victim pool, which everyone joins, will go to the private mining part of the attacker. However, in a different attack scenario where part of the Bitcoin network is not accessible to the attacker, or the attacker only wants to attack a specific pool, the loss from the victim pool also pays for the gain of other miners outside the victim pool. Thus, the above result may or may not hold if the attacker spends too much power on attacking the victim pool so that the gain from the private part is not sufficient to compensate for his loss in the victim pool. We study this scenario next.

To study the attack in this particular scenario, we assume that there are two pools—one target pool $P_1$ and one inaccessible pool [2] $P_2$. Let the computing power of $P_1$ and $P_2$ be $p'$ and $(1-p'-\alpha)$ respectively. The SDV is $(1-\beta, \beta)$, i.e. the attacker mines privately with $\alpha(1-\beta)$ and attacks pool $P_1$ with $\beta\alpha$ fraction of the whole Bitcoin network. Thus, the computing power of $P_1$ when the attack happens is $p = p' + \alpha\beta$.

| Pool | Pool 1 | Pool 2 | Solo | Total |
|---|---|---|---|---|
| Attacker | $\alpha\beta$ | 0 | $\alpha(1-\beta)$ | $\alpha$ |
| Other miners | $p'$ | $1-p'-\alpha$ | 0 | $1-\alpha$ |
| Pool(s) total | $p = p' + \alpha\beta$ | $1-p'-\alpha$ | $\alpha(1-\beta)$ | 1 |

TABLE II: Mining power distribution when part of the network is inaccessible to the attacker. Note that $0 < \alpha, \beta, p < 1$.

We now compute the expected reward for the attacker similar to the previous analysis. The reward from honest private mining is:

$$R_0 = \frac{\alpha(1-\beta)}{1-\alpha\beta}.$$

However, $P_1$ has to split the reward to $p$ fraction of the network even though only $p' = p - \alpha\beta$ fraction is spent on actual mining. The reward that the attacker gets from pool $P_1$ is

$$R_1 = \frac{p-\alpha\beta}{1-\alpha\beta} \times \frac{\alpha\beta}{p}.$$

[2]Either it is inaccessible or the attacker chooses not to attack.

The reward $R$ and the relative gain $\Delta_R$ for the attacker are

$$R = R_0 + R_1 = \frac{-\alpha^2\beta^2 + \alpha p}{p(1-\alpha\beta)},$$
$$\Delta_R = \frac{R}{R_h} - 1 = \frac{\alpha\beta(p-\beta)}{p(1-\alpha\beta)}. \quad (1)$$

From Equation (1), we imply the following results.

**Theorem IV-B.1** (Always withhold rule)**.** *The attacker always gains more reward by mining dishonestly.*

*Proof.* The attack gains extra reward when $\Delta_R > 0$, or from (1) we have $p > \beta$, or $\beta < \frac{p'}{1-\alpha}$. Since $p' > 0$ & $\alpha < 1$, there always exists $\beta$ that helps the attacker to gain more payoff regardless of $\alpha$ and $p'$. □

An immediate consequence of Theorem IV-B.1 is that the network state when all players are honest is not a Nash equilibrium. That is, in the Nash equilibrium state, at least some of the miners are withholding, thereby wasting computing resource for competitive gains.

**Theorem IV-B.2** (Stay Low rule)**.** *The attacker of power $\alpha$ gains more reward only when the power he spends on BWH attacking a pool less than a specific threshold $\alpha\beta_t$.*

*Proof.* Equation (1) also shows that the attacker will gain "negative" extra reward, i.e., start losing, if $\beta > \frac{p'}{1-\alpha}$. The threshold value $\beta_t$ in Theorem IV-B.2 is $\frac{p'}{1-\alpha}$. □

**Theorem IV-B.3** (Target Big rule)**.** *The attacker has more incentive to target big pools than small ones.*

*Proof.* Equation (1) can be rewritten as

$$\Delta_R = \frac{\alpha\beta}{(1-\alpha\beta)} - \frac{\alpha\beta^2}{(p'+\alpha\beta)(1-\alpha\beta)}.$$

With a given $\alpha, \beta$, it clearly shows that $\Delta_R$ is larger if $p'$ is large (since $p' > 0$), or the pool is big. □

**Theorem IV-B.4** (Best strategy)**.** *There exists a $\beta$ that maximizes the attacker reward.*

*Proof.* We prove that given the attacker power $\alpha$, the target pool power $p'$, the attacker gets maximum payoff when

$$\beta = \beta_{max} = \frac{-\sqrt{-p'^2(\alpha p' + \alpha - 1))} - \alpha p' + p'}{\alpha(\alpha + p' - 1)},$$

if $a + p' < 1$, otherwise $\beta = 1/2$. □

***Experimental evaluation.*** We have run several experiments to simulate different CPD-BWH game settings in which the value $\beta$ equal to, less than and greater than the threshold value $\frac{p'}{1-\alpha}$. We illustrate our experiment results in Figure 5.

We discuss each of our theoretical results based on our experimental results as follows.

- *Stay low* rule (Theorem IV-B.2). Our experiments show that, when $\beta$ exceeds the threshold value $\frac{p'}{1-\alpha}$, the attacker will get less payoff than from mining honestly. For example, when $p' = 0.1, \beta = 0.4, \alpha = 0.25$, the attacker receives only 21.82% reward, thus making

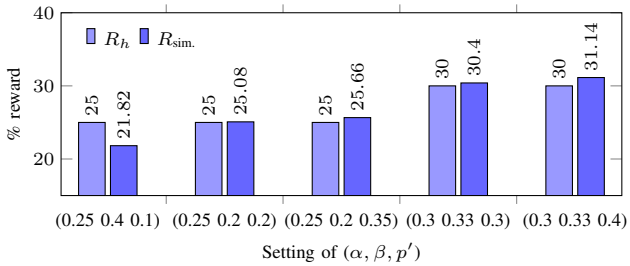


Fig. 5: Simulated reward $R_{sim}$ and honest reward $R_h$ of the attacker $\alpha$ in different CPD-BWH game settings when he spends $\alpha\beta$ power to attack only one pool $p'$.

a relative loss of 12.72%. On the other hand, when $p' = 0.35, \alpha = 0.25, \beta = 0.2 < \frac{0.35}{1-0.25} = 0.47$, he earns 25.65% reward, which is 2.64% relatively more.

- *Target big* rule (Theorem IV-B.3). This rule easily applies to our experiment results. Specifically, given a specific $\alpha = 0.25, \beta = 0.2$, reward that the attacker earns from carrying out BWH attack is more when target the pool of $p' = 0.35$ ($R = 25.66\%$) than to the pool of $p' = 0.2$ ($R = 25.08\%$). We experience the same results for the setting of $\alpha = 0.3, \beta = 0.33$ and two targeted pools of size 0.3 and 0.4. Thus, our experiments support our Theorem IV-B.3.
- *Always withhold* rule (Theorem IV-B.1) and *Best strategy* rule (Theorem IV-B.4). Our existing experimental results for $\alpha = 0.25$ and $\alpha = 0.3$ show that the attacker always has incentive to cheat, i.e., BWH attack, the pool if he keeps his $\beta$ smaller than the threshold. The *Always withhold* rule holds in our experiments although we were not able to split our resource to even finer grained settings, say $\alpha = 1\%$, to intensively verify them.

### C. Mutilple pools: Attack as many as possible

We now consider a general strategy to attack a set of pools such that the SDV is $(\beta_0, \beta_1, \ldots, \beta_n)$ and the attack vector $(\gamma_1, \gamma_2, \ldots, \gamma_n)$. From § IV-B and § IV-A, one clear intuition is to attack every pool that the attacker can access. In this section, we formally study the intuition and find the best strategy for the attacker to gain maximum profit.

The expected reward for attacker from pool $P_i$ is

$$\frac{cp(P_i)}{1-\alpha\sum_{i=1}^{n}\beta_i\gamma_i} \times \frac{\alpha\beta_i}{cp(P_i)+\alpha\beta_i}, \text{ if } P_i \text{ is attacked,}$$

$$\frac{\alpha\beta_i}{1-\alpha\sum_{i=1}^{n}\beta_i\gamma_i}, \text{ if } P_i \text{ is not attacked.}$$

Thus the total reward for the attacker is

$$R = \sum_{i=1}^{n}\left[\frac{cp(P_i)}{1-\alpha\sum_{i=1}^{n}\beta_i\gamma_i} \times \frac{\alpha\beta_i}{cp(P_i)+\alpha\beta_i} \times \gamma_i + \frac{\alpha\beta_i}{1-\alpha\sum_{i=1}^{n}\beta_i\gamma_i} \times (1-\gamma_i)\right]. \quad (2)$$

Finally, the extra reward that the attacker receives is

$$\begin{aligned}\Delta_R &= R/R_h - 1 \\ &= \sum_{1\le i,\gamma_i=1} \frac{1-\alpha\beta_i}{1-\alpha\sum_{\gamma_i=1}\beta_i} \times \frac{\alpha\beta_i(cp(P_i)+\alpha\beta_i-\beta_i)}{(1-\alpha\beta_i)(cp(P_i)+\alpha\beta_i)} \\ &= \sum_{1\le i,\gamma_i=1} \frac{1-\alpha\beta_i}{1-\alpha\sum_{\gamma_i=1}\beta_i} \times \Delta_i \quad (3)\end{aligned}$$

Note that the term $\Delta_i$ is the reward gain ($\Delta_{Ri}$) that the attacker gets when he *only* attacks pool $P_i$ as shown in Equation (1). Since $\forall\ \beta_i \in [0,1], \frac{1-\alpha\beta_i}{1-\alpha\sum_{\gamma_i=1}\beta_i} \geq 1$, the attacker always gains more reward if he follows the *Stay low* rule in each pool. From (3), it is clear that attacking one pool, say $P_2$ ($\beta_2 > 0$), will make the extra reward in another pool, say $P_1$, bigger and vice versa. Hence, as proved in Appendix C.4, $\gamma_i = 1\ \forall i$ will give the attacker the maximum reward, i.e., he is well-incentivized to attack all the pools he can access and privately mine with the rest of his power. However, as explained earlier in this section, if the variance in private mining is a concern, the attacker can honestly mine in one pool and attack the rest.

Thus, the *Best strategy* problem is now converted to finding the optimal SDV $(\beta_0, \beta_1, \ldots, \beta_n)$ such that $R$ is maximum given $\vec{\gamma} = (1, 1, \ldots, 1)$. One can use a variety of optimization techniques to find the optimal value. As an example, we have performed Sequential Least Squares Programming technique [29] with this strategy on the scenario illustrated in Figure 3. We have found that the optimal SDV is

$$(0.60644771, 0.19677677, 0.09838776, 0.09838776)$$

i.e., to mine privately with 0.60644771 fraction of the attacker's power, attack Discus Fish with 0.19677677, Ghash.io with 0.09838776 and KnCMiner with 0.09838776. The corresponding reward that the attacker receives is 26.19%, which is 4.76% relatively more than his honest reward.

***Experimental evaluation.*** We run an experiment with the above optimal SDV setting and the reward that the attacker receives accounts for 26.23% of the network reward, which is close to our analytical result within an experimental error of 0.15% (see Section B, Table V).

### D. BWH when dishonest miners dominate Bitcoin

In the analysis in § IV-B, we assume that all miners except the attacker are honest. We now consider the case of more than one player being rational and incentivized to carry out the BWH attack. Hence, our attack scenario is quite similar to that in § IV-B with two pools $P_1, P_2$ of mining power $p'$ and $1-\alpha-p'$ respectively, except that $P_1$ includes the "contaminated" power fraction $c$ ($0 < c < p'$). For simplicity, we also assume that miners in Pool $P_2$ are all honest. Intuitively, the reward even when the attacker only mines privately honestly is $R_h = \frac{\alpha}{1-c} > \alpha$, since he can pick the loss from the contaminated pool. In this section, we ask whether the attacker still gains more reward than $\frac{\alpha}{1-c}$ by attacking $P_1$. If so, we further study the validity of Theorems IV-B.1, IV-B.2, IV-B.3, and IV-B.4 in this new scenario.

Our CPS game now has an SDV $\vec{\beta_a} = (1-\beta, \beta)$ and an

attack vector $\vec{\gamma_a} = (1)$. Thus the power distribution will be as in Table III.

| Pool | | $P_1$ | $P_2$ | Solo | Total |
|---|---|---|---|---|---|
| Attacker | | $\alpha\beta$ | 0 | $\alpha(1-\beta)$ | $\alpha$ |
| Other miners | *Dishonest* | $c$ | 0 | 0 | $c$ |
| | *Honest* | $p' - c$ | $1 - p' - \alpha$ | 0 | $1 - \alpha - c$ |
| Pool(s) total | | $p = p' + \alpha\beta$ | $1 - p' - \alpha$ | $\alpha(1-\beta)$ | 1 |

TABLE III: Mining power distribution while there is other dishonest miners, in $P_1$. When $c = 0$, we have the distribution in § IV-B.

The analysis is analogous to the previous ones, but with only $(1 - \alpha\beta - c)$ power of the network is mining. Thus, the reward that the attacker will get is as follows:

$$R_0 = \frac{(1-\beta)\alpha}{1-c-\alpha\beta} \text{ (private mining)},$$
$$R_1 = \frac{\alpha\beta}{p} \times \frac{p-\alpha\beta-c}{1-c-\alpha\beta} \text{ (from pool } P_1),$$
$$R = R_0 + R_1 = \frac{p\alpha - \alpha^2\beta^2 - \alpha\beta c}{p(1-\alpha\beta-c)}. \quad (4)$$

Thus, the attacker gets extra reward by conducting a BWH attack when:

$$R > R_h \Leftrightarrow \frac{p\alpha - \alpha^2\beta^2 - \alpha\beta c}{p(1-\alpha\beta-c)} \geq \frac{\alpha}{1-c}$$
$$\Leftrightarrow \beta \leq \frac{p'\alpha - c(1-c)}{\alpha(1-\alpha-c)}. \quad (5)$$

Equation (5) shows that, if $p'\alpha - c(1-c) \leq 0$, or $\alpha < \frac{c(1-c)}{p'}$, the attacker should not attack Pool 1 because regardless of the strategy he uses, he will lose out. Thus it also shows that, the *Always withhold* rule in the previous analysis does not hold if $\alpha < \frac{c(1-c)}{p'}$. However, the following rules still apply, and are verified with the experimental results reported in Table IV.

- *Stay Low Rule*. If $\beta > \frac{p'\alpha - c(1-c)}{\alpha(1-\alpha-c)}$, the attacker will get less payoff than from mining honestly. For example, if $\alpha = 0.20, c = 0.05, p' = 0.35$, the attacker loses his reward ($R = 20.77\% < R_h = 22.22\%$) if he uses $\beta = 0.25$ to attack. On the other hand, in the first and second experiment, he still earns more if he attacks with $\beta = 0.125$ which is smaller than the threshold.
- *Target Big Rule*. With a given $\alpha, \beta, c$, Equation (4) shows that $R$ is large if $p'$ is large, i.e., the pool is big. Thus, the rule still holds. For example, with the same setting of $(\alpha = 0.4, c = 0.025, \beta = 0.125)$, the attacker gets more reward when targeting a pool with $p' = 0.375$ ($R = 42.30\%$) than another one with $p' = 0.325$ ($R = 41.74\%$).
- *Best strategy Rule*. When $\alpha < \frac{c(1-c)}{p'}$, there exists a strategy other from honest mining for the attacker to maximize his reward. The value $\beta$ for that strategy is the value that maximizes the Equation (4).

### E. Quantifying loss for honest miners

In this section, we discuss the loss of the pool and other honest miners in the pool when the BWH attack happens. Intuitively, the pool of size $p'$, when attacked by a power of $c$ fraction, will receive only $\frac{p'}{1-c} \times \frac{p'}{p'+c} \leq p'$ reward. We take the scenario when $(\alpha, \beta, p') = (0.2, 0.25, 0.2)$ as an example.

| $\alpha$ | $\beta$ | $c$ | $p'$ | $R$ | | $R_h$ | $\Delta_R$ |
|---|---|---|---|---|---|---|---|
| | | | | **Theory** | **Sim.** | | **Sim.** |
| 0.2 | 0.125 | 0.05 | 0.35 | 21.32% | 20.98% | 21.05% | -0.33% |
| | 0.125 | 0.05 | 0.4 | 21.14% | 21.27% | 21.05% | 1.00% |
| | 0.25 | 0.1 | 0.35 | 21.08% | 20.77% | 22.22% | -6.43% |
| 0.4 | 0.125 | 0.025 | 0.375 | 42.29% | 42.30% | 41.02% | 3.33% |
| | 0.125 | 0.025 | 0.325 | 42.16% | 41.74% | 41.02% | 1.76% |
| | 0.25 | 0.05 | 0.3 | 42.64% | 41.87% | 42.11% | -0.57% |

TABLE IV: The reward $R$ and relative reward $\Delta_R$ gained by attacker when there is already a "contamination" factor of $c$ in the pool. We report the expected (theoretical) results (Theory column) as well as our simulation results (Sim. column) of $R$ and $\Delta_R$ in each game.

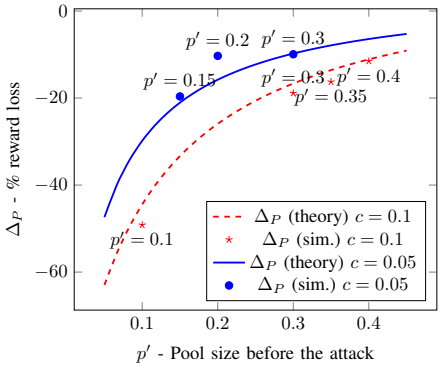


Fig. 6: The pool's loss in experiments and in theory for different pool size ($p'$) and contaminated factor ($c$).

Although the attacker does not gain or lose any reward, the honest miners in the pool lose $\Delta_P = 10.31\%$ of their reward. Tha is because other miners outside the target pool also enjoy the gain, even though they do not attack the pool. We plot the theoretical and experimental loss of the pool in Figure 6.

Although the big victim pools bring more reward to the attacker than the small ones, the pool of smaller size will have to bear much more damage than the bigger one. For example, a contaminated factor of $c = 0.05$ causes a 15%-pool around 20% loss in reward, almost twice as much as the 10.31% loss on a 20% pool.

***Relating to current Bitcoin network.*** Our experiments show that, the pool of size 30% ($p' = 0.3$)—size of Discus Fish, the real biggest pool as of November 2014—will lose 9.94% of its reward if attacked with a contamination power of $c = 0.05$. Given the price of a ₿ is 350 US Dollars and the attack happens for one month, it may cost Discus Fish miners around *1 million USD* in that month.

### F. The Nash Equilibrium

Since we have shown that the BWH attack is profitable and causes a serious loss to a honest miner, the implication is that rational miners will be incentivized to form a private group and carry out the attack widely. More interestingly, since from the network point of view, a pool is similar to one "big" miner, thus the pool is also able to attack others. We study whether there exists a Nash equilibrium with a pure strategy in this game. Specifically, does there exist a deterministic attack strategy for each miner? For sake of simplicity, we assume that the Bitcoin network comprises of only two accessible pools $P_1$ and $P_2$, each has only one miner with computing power $\alpha_1$, $\alpha_2$ respectively. We also assume that $P_1$ and $P_2$ are both

| No. | Settings | | | | $R$ | | EErr | # of blocks |
|---|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ | $p'$ | c | Sim. | Theory | | |
| **A. Bitcoin as one pool** | | | | | | | | |
| 1 | 0.2 | 0.5 | 0.8 | 0 | 20.78% | 20.98% | 0.95% | 10929 |
| 2 | 0.4 | 0.5 | 0.6 | | 43.29% | 43.75% | 1.05% | 6507 |
| **B. Attack multiple pools** | | | | | | | | |
| 1 | 0.25 | Strategy in § IV-C | | | 26.23 | 26.19 | 0.15% | 2905 |
| 1 | 0.25 | Strategy in § V | | | 26.49% | N/A | N/A | 10934 |

TABLE V: The theoretical and experimental rewards for several experiment settings. The parameters are $\alpha$: attacker's power, $\beta$: amount of power that attacker uses for BWH attack, $p'$: the pool power before the attack and $c$: the fraction of BWH attacker already in the pool.

rational and motivated to perform the BWH attack on the other pool with $c_1$ ($c_1 < \alpha_1$) and $c_2$ ($c_2 < \alpha_2$) power. Before each miner makes a move, the network state is known to everyone. The goal of each of them is to adjust his attacking power $c_i$ properly to achieve the higher reward.

We show that there exists no pure strategy for the miner in this two-pool setting. Thus, this game has *only a mixed strategy* in its equilibrium. Therefore, we can conclude that the game does not have any pure strategy, otherwise the other miner always finds a better strategy in his move to win back the game. To arrive at this result, we prove the following Theorem IV-F.1.

**Theorem IV-F.1.** *In the two-pool game, given any network state, if the player $i \in (1, 2)$ has picked a strategy with $c_i$ fraction to attack, then the opponent has a strategy to gain more reward in the game.*

*Proof.* In the two-pool game, given $(\alpha_1, \alpha_2, c_2)$, $P_1$ wants to determine $c_1$ that optimizes his payout $R_1$, which is computed in the same fashion as in previous sections:

$$R_1 = \frac{1}{1 - c_1 - c_2}\left(\frac{(\alpha_1 - c_1)^2}{\alpha_1 - c_1 + c_2} + \frac{c_1(\alpha_2 - c_2)}{\alpha_2 - c_2 + c_1}\right).$$

Similarly, $P_2$ wants to maximize $R_2$ given $\alpha_1$, $\alpha_2$, and $c_1$,

$$R_2 = \frac{1}{1 - c_1 - c_2}\left(\frac{(\alpha_2 - c_2)^2}{\alpha_2 - c_2 + c_1} + \frac{c_2(\alpha_1 - c_1)}{\alpha_1 - c_1 + c_2}\right).$$

As we prove in Appendix C, for any given network state, there exists a $c_i$ value for the miner $P_i$ to increase his reward $R_i$ and cause the other pool a loss. □

Theorem IV-F.1 implies that being honest is not the best strategy in Bitcoin pooled mining. Since there exists a mixed strategy, a fraction of the network is always dishonest and the overall network resource is under-utilized.

## V. Do network state and game duration matter?

This section is partially motivated from our discussion with Bitcoin pool operators to address their concern that variations in difficulty, distribution of pool power, hashrates, etc. would affect our findings.

### A. Is it necessary to have a constant network state?

Our aim in this experiment is to show that the attacker still gains extra reward even when both the total mining power, thus the network difficulty, and the power distribution are not stable through out the simulation. We also show that the attacker only

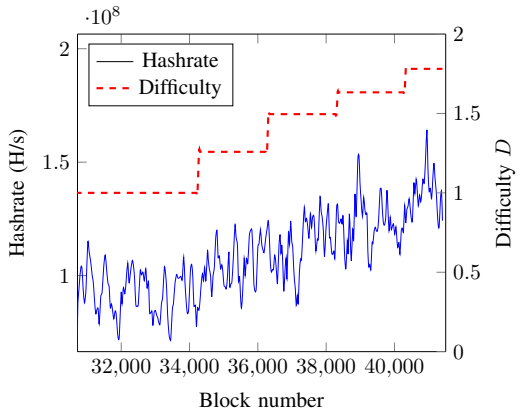


Fig. 7: Hashrate & difficulty of the network in our experiment in § V.

adjusts his power distribution after the change in the power distribution happens. We only keep the attacker power $\alpha$ as constant as a fraction of the entire network power, but the difficulty $D$, the total power, the pool power distribution $cp$ and the vector $\vec{\beta}$ will change frequently throughout the experiment. We only use the best strategy vector $\vec{\beta}$ for the attacker initially. When the network and its power distribution changes, it takes some time for the attacker to adjust his $\vec{\beta}$. Thus after the first change, the exact power distribution of the current network is no longer available to the attacker.

Typically, we start with the same setup as in § IV-C where the attacker attacks multiple pools and add more mining power to the network for five times. We allocate the additional power to the pools but still keep $\alpha = 0.25$. We only adjust the $\vec{\beta}$ of the attacker strategy corresponding to the distribution after the $i$-th change when the next ($i + 1$-th) change happens.

The power and difficulty changes in our experiments are illustrated in Figure 7. The attacker's power distribution, $\vec{\beta}$ value, and attacker reward after each change are illustrated in Figure 8. The attacker always receives more than 25% of reward and make a final gain of 5.96%. This makes our assumption about the constant distribution power fair and practical. Our experimental results also imply two additional important points. First, network and difficulty fluctuation do not have any significant impact on the attacker gain, as we expected. Second, since the power distribution change may require the attacker some time to recognize, one "safe strategy" is to set all $\beta_i$ lower than the value in best strategy and adjust them later when the attacker is aware of the change. This "safe strategy" will secure the positive gain for the attacker even when he is not able to immediately recognize the power distribution change.

### B. Does duration of BWH attack matter?

In our analysis, we have have ignored the duration variation of each game and only take into account the game reward to consider the profit of the attacker. However, a common factor used in practice is the reward rate, say per day. Thus, although we have showed that the BWH attack yields a net profit in a game for the attacker, it is not always the case that his reward rate gets increased. In fact, we show that the attacker most likely gains profit by carrying out the attack in a long period of time, but that may not hold in the short term one.

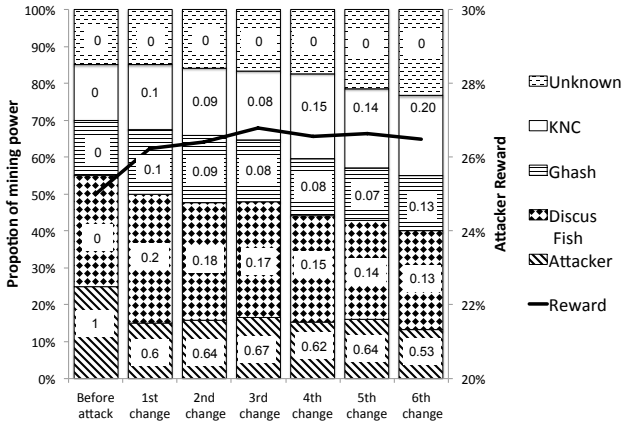


Fig. 8: Power distribution, attacker's strategy and reward vary several times in our experiment. The number in the white box represents the $\beta$ of the attacker for that pool. The attacker reward, however, is always greater than the reward he receives when mining honestly.

***Short-term profit.*** When BWH attack happens, a fraction of the network is wasted performing the attack, thus taking the network longer to find a block, i.e., finish the game. In fact, the attacker gets better reward rate in a game only if the condition in Theorem V-B.1 holds.

**Theorem V-B.1.** *A miner with computing power $\alpha$ using $c$ as the contamination factor to attack will only gain higher reward rate if the following inequation is satisfied: $\frac{\Delta_R}{\alpha} > \frac{c}{1-c}$*

Here, $\Delta_R$ is the extra reward computed in our various analysis scenarios in § IV. We prove Theorem V-B.1 in Appendix C.

Theorem V-B.1 implies that in a short period of time, whether the attacker's reward rate increases depends on various factors, e.g., $c, \alpha$, although his reward per game gets increased.

***Long-term profit.*** We show that, in a longer duration, the BWH attack allows rational miners to gain higher reward rate. The following theorems establishes the claim.

**Theorem V-B.2.** *Over any fixed number of games, a rational miner always gains more absolute reward by withholding.*

*Proof.* We have established that there exists a mixed-strategy for the rational miner to maximize his absolute reward in a game, albeit at a different reward rate than honest mining. Thus, when the attacker plays his mixed-strategy in every game, his total reward in any number of games will be strictly more than that by honest mining. □

Note that in a long period of time, the number of blocks mined (or games played) remains constant. This is because Bitcoin is a self- adjusting network, i.e., the difficulty $D$ adjusts after every $2,016$ blocks (~2 weeks) to make the average time to find a block is $10$ minutes. This accounts for any power lost due to withholding. Therefore, the number of blocks solved in a sufficiently long duration stay constant, which is consistent with the empirical observation [30]. Since the number of games over a given time period (say 3 months) stays constant, Theorem V-B.2 implies that the BWH attack is profitable.

## VI. Discussion on Defenses

Since the BWH attack yields net profit for the attacker, as the attack becomes better understood it may become popular unless countermeasures are developed. Rational miners will face a troubling choice: mine honestly solo or in private pools with those they trust, or—if dishonest—attack any accessible pool in which honest miners are operating. In this section we first discuss about how to detect a possible BWH attacker in a pool. We then describe several fixes and discuss their drawbacks.

### A. Desired properties

In order to determine whether a fix is adequate and practical, we propose a set of desired properties for a pool and a fix. A pool is considered ideal if

- **P1.** It does not favor either big or small player, and should treat them equally as long as they are honest.
- **P2.** It disincentivizes both pool operator and player to drop valid blocks.

The current pooled mining protocol does not satisfy **P2**, thus making the fix necessary. We also define several practical properties required in a desirable fix—these are specifically to Bitcoin and may or may not apply in other CPS applications.

- **P3.** It preserves the existing Bitcoin blockchain.
- **P4.** It is compatible with existing mining hardware.
- **P5.** It does not affect miners who are not in the pool.
- **P6.** It requires a minor Bitcoin protocol's change.
- **P7.** It does not make a pool non-ideal.

One possible approach to eliminate the attack and satisfy all properties is to early detect the attacker. We introduce two detection tests using statistics and cross checking technique, then explain why these tests are not practical in Appendix A.

### B. Change to payoff scheme

One of the main reasons that make the BWH attack profitable is that every share has the same value from the miner's perspective. Thus, we propose that some shares which are also valid blocks should be considered to be more valuable than others. While this intuition is solid, the key question is how much more reward is necessary for these shares.

Call $x$ the fraction of the current block reward that will be necessary to ensure that attackers do not drop blocks. We first prove that this reward scheme is still fair in Appendix C. Since the reward for carrying out the BWH attack depends on the amount of computational power $\alpha$ controlled by the attacker, it makes sense that $x$ depends on $\alpha$. In fact, in Appendix C, we prove that $x = \alpha$ is the smallest fraction necessary to dissuade an attacker completely. In other words, to prevent an attacker with $\alpha = 0.25$ from withholding blocks, we need to make the valid-block share worth $25\%$ of the block reward. Since when $\alpha > 0.5$ the attack is meaningless anyway (the attacker could simply take control of the entire blockchain), setting $x = 0.5$ is a very conservative upper bound.

***Drawbacks.*** Although this technique satisfies the above properties **P3** to **P6**, it suffers from several drawbacks:

- **P1** breaks down: normal shares are worth significantly lesser. Thus, compared to present experience in pools,

volatility increases for all pool participants and especially for smaller ones.
- Fundamentally, the technique does not prevent the attack, but merely disincentivizes it. Attackers may have other reasons to attack (such as a desire to discredit the Bitcoin ecosystem by a disapproving state-sponsored actor).

### C. Bitcoin protocol with native support for pooled mining

As argued above, changing payoff schemes does not prevent the BWH attack completely. Thus, we seek for a change in the Bitcoin protocol to prevent the threat. Despite the fact that the BWH attack has been a controversial topic, some developers and researchers still proposed the fix to mitigate the attack. Till date we know of two proposed solutions, which require changes in the proof-of-work (PoW) algorithm [31, 12]. The general idea of these approaches is to not allow miners to recognize which share is a valid block, thus not allowing dishonest miners to drop the block on his will.

Luke Dashjr proposed to include the next block candidate hash in the PoW of the current block [31]. Thus, the miners never know which share is a valid block until the subsequent block is also found. We find that his solution can actually defeat the attack but is not necessarily complicated and changes the Bitcoin protocol significantly — violating property **P4, P6**, which the Bitcoin community is highly reluctant to do. Further more, it changes the blockchain structure and affects solo miners largely, for example it takes longer time to validate a transaction now, thus not satisfying properties **P5**.

In [12], Rosenfeld proposes a solution which requires less modification than Luke's by introducing the oblivious share concept which is to ensured that miner is unable to determine if a share is a valid block. More specifically, he suggests a two-part PoW with 3 additional fields to each block, namely *SecretSeed*, *ExtraHash* and *SecretHash*, in which *ExtraHash*= `SHA256`*(SecretSeed)*. The two-part PoW works as following.

- *The hard (public) part.* The *ExtraHash* is included in the block header which is given to the miner to try all the *Nonce*. A hash is a valid share iff it satisfies difficulty $d_1$.
- *The easy (secret) part.* The pool operator will compute $\mathit{SecretHash} = \texttt{SHA256}(\mathit{SecretSeed} \mid\mid \mathit{Share})$ and check if it satisfies a difficulty $d_2$. If so, the *SecretHash* is also a valid block and the operator will broadcast it to the network. Since only the pool operator obtains the *SecretSeed*, the miners do not know which share is corresponding to a valid block.

In the above PoW, the total difficulty of both parts $d_1+d_2$ is greater or equal to the network difficulty $D$. Thus, solo miners may not need to split the mining into two parts but only set $d_2 = 0$ & $d_1 = D$ to mine as currently.

***Drawbacks.*** We find that this proposal is quite simple and easier to implement. It satisfies all properties mentioned above except the **P2**, i.e., it is not compatible to the current ASIC (application-specific integrated circuit) miners [32, 33], which is a substantial mining force in Bitcoin currently.

## VII. Related work

***Detection Cheating in Distributed computation.*** Numerous previous works have considered distributed computation tasks which are not competitive or time-sensitive, and often consider a single supervisor system rather than one with many supervisors outsourcing tasks [6]. One practical line of work which focuses more on detecting cheating clients in distributed computation [6, 8, 7]. A complimentary line of work studies the problem of verifiable computing, which enables checking if an arbitrary program has computed correctly from designated inputs using cryptographic constructions or using trusted hardware [34, 35, 36, 37, 38, 39, 40]. These techniques can help in ensuring that the pool protocol is strictly followed, disallowing players from diveating from prescribed behavior. In contrast, our work studies the question of eliminating the incentives for cheating by using secure payoff schemes.

***Block withholding attacks.*** BWH attacks have been a subject of a few recent papers. In [12], Rosenfeld *et al*. discusses BWH and considers it as a non-incentivized sabotaging attack, simply to sabotage the pool profits. Recently, an initial work by Nicolas *et al*. showed that the BWH is possibly profitable and well-incentivized [16]. However, the analysis in [16] is inordinately abstract and an overestimation leads to imprecise results, as we explain in the footnote of § IV-A. Further, their work only analyzes a simplified case where the whole network is one large pool (a special case of our analysis in § IV-A).

We are aware of a recent paper, concurrent and independent to our work, that discusses how pools can use BWH attacks to infiltrate each other [17]. We have privately communicated with the author of [17] in November 2014. The two works are similar—we approach the problem of understanding the incentive structure for miners in an arbitrary CPS game, where the concurrent work aims to calculate infiltration rates of pools at war. Both [17] and our work arrive at some consistent findings, for example, that the honest mining is not the stable equilibrium (§ IV-F) and the amount of loss to pools(§ IV-E). However, our work studies the problem from a different perspective and considers several other scenarios, for example, the general case of attacking multiple pools and when there are multiple dishonest miners in the victim pool. Our work further explains the temporal conditions under which the attacks are possibly profitable, conduct experimental tests and discuss potential defenses. The work in [17] additionally explains the Nash equilibrium for two pools and multiple pools of symmetric power, which are interesting special cases of the general game.

***Other Bitcoin attacks.*** A number of previous works study non-withholding attacks. Our CPS game model generalizes previous studies and can be useful to systematize the study of these attacks in the future. In particular, in [12] Rosenfeld *et al*. discusses (i) "pool hopping" in which miners hop across different pools utilizing a weakness of an old payoff scheme, and (ii) "Lie in wait" attacks where the miner strategically calculates the time to submit the found block. Another line of work studies attacks that subvert the basic guarantees of the Bitcoin consensus protocol, such as preventing double-spending. Eyal *et al*. introduced "Selfish mining", where a pool with more than $1/4$th of the total computing power can subvert the Bitcoin consensus protocol [28], improving over the well-understood $51\%$-attack [41, 42]. Johnson *et al*.

distributed denial-of-service (DDoS) attacks between pools, taking a take a game-theoretic approach to understand the economics of DDoS attack [43]. This game-theoretic model is different from our CPS game, since ours is appropriate for studying the incentive structure for individual miners.

## VIII. Conclusion

In this paper, we introduce a new game model called computing power splitting game, which is useful for studying the security of payoff schemes in competitive distributed computation tasks. As a case study, we analyze the susceptibility of existing Bitcoin mining pool protocols. We find that these protocols are insecure against block withholding. Our CPS game model generalizes such reasoning in many other cryptocurrency attacks and is a step towards systematizing the study of such attacks.

## IX. Acknowledgements

We thank Jason Teutsch, Meni Rosenfeld, Luke Dashjr, Andrew Miller, Jason Hughes, Gregory Maxwell, Ittay Eyal, Alex Cook, and the anonymous reviewers of an earlier draft of this paper for their helpful feedback. This work is supported by the Ministry of Education, Singapore under Grant No. R-252-000-560-112. All opinions expressed in this work are solely those of the authors.

## References

[1] RSA Secret-Key Challenge. http://en.wikipedia.org/wiki/RSA_Secret-Key_Challenge, February 2015.

[2] Bugcrowd - Your Elastic Security Team. https://bugcrowd.com/, February 2015.

[3] Crowdcurity. https://www.crowdcurity.com, February 2015.

[4] The Heartbleed challenge. https://blog.cloudflare.com/the-results-of-the-cloudflare-challenge/, February 2015.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org*, 2009.

[6] Philippe Golle and Ilya Mironov. Uncheatable distributed computations. In *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, CT-RSA 2001, pages 425–440, London, UK, UK, 2001. Springer-Verlag.

[7] Michael T. Goodrich. Pipelined algorithms to detect cheating in long-term grid computations. *Theor. Comput. Sci.*, 408(2-3):199–207, November 2008.

[8] Wenliang Du and Michael T. Goodrich. Searching for high-value rare events with uncheatable grid computing. In *Proceedings of the Third International Conference on Applied Cryptography and Network Security*, ACNS'05, pages 122–137, Berlin, Heidelberg, 2005. Springer-Verlag.

[9] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: Practical accountability for distributed systems. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 175–188, New York, NY, USA, 2007. ACM.

[10] SETI@home project. http://setiathome.ssl.berkeley.edu/.

[11] Crypto-Currency Market Capitalizations. http://coinmarketcap.com/.

[12] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *CoRR*, abs/1112.4980, 2011.

[13] ckolivas. A block-withholding miner. https://bitcointalk.org/index.php?topic=267181.msg2860365#msg2860365.

[14] Bitsaurus. Withholding attacks - analysis of 200 tera-hash withholding attack. https://bitcointalk.org/index.php?topic=731663.msg8270133#msg8270133.

[15] How is block-solution-withholding a threat to mining pools. http://bitcoin.stackexchange.com/questions/1338/how-is-block-solution-withholding-a-threat-to-mining-pools.

[16] Nicolas T. Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *CoRR*, abs/1402.1718, 2014.

[17] Ittay Eyal. The miner's dilemma. In *To appear at Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '15. IEEE Computer Society, 2015.

[18] Block withholding attack on Eligius mining pool. https://bitcointalk.org/?topic=441465.msg7282674.

[19] Bitcoin statistics. https://blockchain.info/stats, October 2014.

[20] Montgomery Rollins. *Money and Investments 1928*. Kessinger Publishing, 2003.

[21] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology — CRYPTO' 92*, number 740 in Lecture Notes in Computer Science, pages 139–147. Springer Berlin Heidelberg, January 1993.

[22] Adam Back. Hashcash - a denial of service counter-measure. Technical report, 2002.

[23] Bitcoin Foundation. Bitcoin difficulty. https://en.bitcoin.it/wiki/Difficulty.

[24] Bitcoind client. https://github.com/bitcoin/bitcoin.

[25] `cpuminer` mining software. https://github.com/pooler/cpuminer, October 2014.

[26] Stratum poolserver in node.js. https://www.npmjs.org/package/stratum-pool, October 2014.

[27] Bitcoin hashrate distribution. https://blockchain.info/pools, Jan 2015.

[28] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *arXiv preprint arXiv:1311.0243*, 2013.

[29] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[30] Controlled supply in Bitcoin. https://en.bitcoin.it/wiki/Controlled_supply.

[31] Luke-Jr. Defeating the block withholding attack. http://sourceforge.net/p/bitcoin/mailman/message/29361475/, 2012.

[32] Cryddit. Redesign of Bitcoin block header. https://bitcointalk.org/index.php?topic=626377.msg6975690#msg6975690.

[33] Canaan-Creative. Miner manager. https://github.com/Canaan-Creative/MM.

[34] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*,

45(1):70–122, January 1998.
[35] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. ACM.
[36] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, New York, NY, USA, 1988. ACM.
[37] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, New York, NY, USA, 2012. ACM.
[38] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology–CRYPTO 2013*, pages 71–89. Springer Berlin Heidelberg, 2013.
[39] Elaine Shi, Adrian Perrig, and Leendert Van Doorn. Bind: A fine-grained attestation service for secure distributed systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 154–168, Washington, DC, USA, 2005. IEEE Computer Society.
[40] Jonathan M. McCune, Bryan J. Parno, Adrian Perrig, Michael K. Reiter, and Hiroshi Isozaki. Flicker: An execution infrastructure for tcb minimization. In *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, Eurosys '08, pages 315–328, New York, NY, USA, 2008. ACM.
[41] Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Workshop on the Economics of Information Security*, June 2013.
[42] RHorning. Mining cartel attack. https://bitcointalk.org/index.php?topic=2227.0.
[43] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 72–86. Springer Berlin Heidelberg, 2014.
[44] Goodman SN. Toward evidence-based medical statistics 1: The P value fallacy. *Annals of Internal Medicine*, 130, 1999.
[45] Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.
[46] Stats in Eligius pool. http://eligius.st/~gateway/stats.
[47] `cpuminer` block withholding version. https://github.com/annonymized-for-submission/cpuminer, October 2014.
[48] Avalon ASIC miner. https://ehash.com/product/avalon3-1-2t/.

## APPENDIX

### A. Detecting BWH attack in a pool

Bitcoin is an anonymous currency, and is proliferating with computation-on-rent services — this makes it hard to conclusively detect if a user is withholding blocks and mining honestly elsewhere. The problem of detecting Block Withholding miner in a pool is widely considered hard, and a correct solution is still an open question. In this section, we describe potential approaches to address the problem.

*1) Statistical detection test:* We explain how statistical methods can help determine malicious activity in pools. It is important to note that statistical methods can only infer whether a user's pool activity is close to its theoretically estimate or is "anomalous"; being anomalous does not imply with certainty that users are mining dishonestly. However, they can be helpful in determining anomalous activity by a user or a group of users. For the purpose of this measurement, we use publicly available data about the submitted shares by users on a real-world pool named Eligius. Eligius is presently the only pool that makes such data public.

***Choice of Statistical Inference Strategy.***

In our work, we use statistical significance testing. Intuitively, in this approach, one adapts the standard proof-by-contradiction idea to probabilistic domain, as follows. Suppose the user mines honestly—let us call this the null hypothesis. Compute the probability (p-value) that the user would behave as observed in our dataset, assuming the null hypothesis is true. If the probability is very low (say below a threshold of confidence), then the null hypothesis can be confidently rejected. In other words, given the data we have, we can say that there is statistically significant evidence against the null hypothesis implying that the user mined honestly. We use a standard [44] threshold of $0.05$ as a confidence level. We then perform the Holm-Bonferroni method [45] to perform a simultaneous inference and report the users with suspicious behavior.

***Calculating P-value.*** Let $P$ be a pool we are interested in, with a set of miners $U$. Let $d_I$ and $D_I$ be the pool and network difficulty respectively over time interval $I$. Since $d_I$ and $D_I$ changes over a typically long time period $I$, we split $I$ in ordered and disjoint time intervals $I_1, I_2, \ldots, I_t$ such that $d_{I_i}$ and $D_{I_i}$ are fixed for all $i \in \{1, \ldots, t\}$. Naturally, the new time intervals are maximal in the sense that either $d_{I_i} \neq d_{I_{i+1}}$ or $D_{I_i} \neq D_{I_{i+1}}$ for all $i \in \{1, \ldots, t-1\}$. We often simplify $I_i$ indexed variables with $i$ when the context is clear.

We are interested to analyze if over a given time interval $I$, any user $u \in U$ is withholding blocks in pool $P$, thus performing a block withholding attack. Let $N_I^u$ be the number of shares submitted by user $u$ over the time period $I$. Similarly, let $B_I^u$ denotes the number of blocks submitted by user $u$ over $I$. Since the user $u$ can potentially withhold blocks, the number of blocks he submits can only be less than or equal the number of blocks he actually mines. Let $X_I^u$ denote the random variable representing the number of blocks *mined* by $u$ over time period $I$. Again, we omit mentioning $u$ when the context is clear.

It is easy to see that in the time period $I_i$, the probability of a valid share being a block is $d_i/D_i$ in pool $P$. Thus $X_i$ follows a Poisson distribution with mean $\lambda_i = N_i \times d_i/D_i$ for $i \in \{1, \ldots, t\}$ and $X_I = \sum_{i=1}^{t} X_i$. Since the number of blocks mined over two disjoint time periods are independent, $\{X_i\}_{i\in\{1,\ldots,t\}}$ are mutually independent. Consequently, $X_I$ follows Poisson distribution with mean $\lambda_I = \sum_{i=1}^{t} \lambda_i$.

Let us hypothesize that user $u$ is honest in pool $P$. We now calculate $Pr(X_I^u \leq B_I^u \mid u \textit{ is honest})$, the probability that the number of blocks he actually mined is less than or equal to $B_I^u$, with the assumption that $u$ is honest. If this probability value (p-value) falls below a certain predetermined threshold (say) $s$, we reject the hypothesis. Since $X_I^u$ follows Poisson distribution with mean $\lambda_I$, we can readily calculate $Pr(X_I^u \leq B_I^u)$ using the cumulative density function for Poisson distribution.

The numbers of shares and blocks submitted by 58 users in Eligius are illustated in Table VI.

***Results.*** We analyze the public data of top $58$ users who have submitted the most shares on the Eligius pool [46] from $1^{st}$ Jan 2014 to $1^{st}$ Oct 2014. Presently, Eligius is the the fourth largest Bitcoin mining pool. Since the number of shares submitted is sufficiently large for major users, for example the minimum and maximum values of that are $1,321,008,768$ and $2,020,713,113,960$ respectively, there are enough samples to perform statistical inference test. After performing Holm-Bonferroni method with significance level $0.05$, we reject null hypothesis for 1 user, while the remaining users are well above the threshold. We find that the user reported in our test was actually banned by the pool operator in April $2014$ for block withholding [18]. Thus, this method detects the banned user reliably.

***Drawbacks.*** In order to run the statistical inference test, we have to collect enough samples. Thus, the attacker can simply bypass the test by splitting his mining force into multiple smaller sub-miners, making the number of samples of each individual sub-miner insufficient for the test. However, it is open to see how one can group multiple users into one and apply similar statistical method to detect group of users with anomalous activity.

*2) Cross-checking:* Another possible technique to detect if there is block withholding miner in a pool is to send a block template which allows miners to instantly return a valid block. One way to do that is to extract the block template from a newly submitted block, and send it randomly to, say, $5\%$ of the miners in the pool. Since the work required to find a valid block from that template is negligible, the pool operator should expect those miners to return the work within some seconds [3]. Thus, after several blocks, the pool operator will be able to detect with high probability if someone is dropping the valid block.

***Drawbacks.*** However, cross-checking has several drawbacks. First, the attacker can divide his power to work as many smaller sub-miners, and check if two sub-miners have the same work template. Second, the technique will cost the pool operator more as he has to pay for the redundant shares and blocks, also reduce the overall efficiency of the pool.

[3]With the current design of the block, it requires only $2^{31}$ SHA-256 hash computation on average to find a valid block from such a block template.

### *B. Experiment setup*

We evaluate our theoretical analysis on our customized Testnet Bitcoin network. Specifically, we incorporate all important components of the current network into our simulation.

- ***Testnet Bitcoin.*** We create our own Testnet Bitcoin network in our simulation by modifying the official Bitcoin `Bitcoind` client [24] to adjust the expected time to generate a block to 1 minute instead of 10 minutes as in the real network to increase the rate at which blocks are generated. We start our experiment with the network difficulty $D$ of 1, thus a valid block hash needs to have at least 32 leading zeros. During the experiments, $D$ varies within the range of $(1, 2)$. Other properties of the network stay unchanged, for example the network adjusts its difficulty after every $2016$ blocks.
- ***Mining pools.*** We setup our pools as stratum pools using a popular Node.js module [26]. Stratum protocol is used in almost all mining pools currently since it significantly reduces the network overhead in pooled mining. The share difficulty in every pool is fixed at $d = 0.001$. We run our pools in one Amazon EC2 instance and in our local server, which connect to Testnet Bitcoin node there. We compute the reward by using the Proportion scheme, i.e., distributing the reward proportionally to the number of shares submitted.
- ***Mining software.*** Our miners run within Amazon instances from different regions and use `cpuminer` software [25] to mine on our stratum pools. In each experiment, we use 20 to 60 CPU cores from $5$ different AWS regions. We also modify `cpuminer` to perform the BWH attack. Our BWH version of `cpuminer` is available at our public repository [47].

To our best knowledge, this work is the first attempt to simulate the exact mining behavior by mining in a modified Testnet Bitcoin network. Previous work only uses probabilistic programing models to back up their analysis and do not closely model the real Bitcoin network as we do [28]. The reason is Bitcoin mining is an extremely resource-extensive activity and may require a long period of time to see the result, especially when one wants to conduct the experiment in the real network. For example, the cost to have $10\%$ of the network mining power currently is roughly 26 million US dollars by buying $23,0000$ latest Avalon AISC miner [48]. Further, we decided against carrying out the BWH in the real network for ethical reasons. Thus, the best we can do to empirically verify our analysis results is to mine on our custom Bitcoin network, with our CPU miners from Amazon EC2.

### *C. Proof of Analysis*

**Lemma C.1.** $\Delta_R > 0$ *in Section IV-A.*

*Proof.* We prove that $\forall \alpha, \beta$ that $\alpha, \beta \in (0, 1)$, we have:

$$\frac{R}{R'} = \frac{\alpha\beta - \alpha\beta^2 + 1 - \alpha}{(1 - \alpha\beta)(1 + \alpha\beta - \alpha)} > 1 \tag{6}$$

| address | shares | blocks | address | shares | blocks | address | shares | blocks |
|---|---|---|---|---|---|---|---|---|
| 1Fxky... | 413,904,209,640 | 40 | 1A7tU... | 185,009,798,672 | 10 | 13RGB... | 20,216,769,408 | 0 |
| 19ChC... | 210,887,124,992 | 8 | 1LLb3... | 86,104,016,000 | 4 | 1ACZk... | 18,702,475,776 | 0 |
| 145Xk... | 25,593,869,184 | 0 | 1LaiU... | 247,113,783,248 | 26 | 17m3V... | 31,124,492,864 | 3 |
| 1Nbq2... | 2,020,713,113,960 | 389 | 17ZT9... | 7,452,058,240 | 0 | 1EHwN... | 34,294,658,816 | 0 |
| 1swrt... | 1,573,798,539,880 | 160 | 181pf... | 2,494,064,324 | 1 | 159Fe... | 192,214,203,316 | 28 |
| 1Fewy... | 470,566,555,648 | 19 | 1JR61... | 45,317,980,416 | 5 | 1Jf5j... | 35,966,974,784 | 2 |
| 1Hzo3... | 619,240,529,040 | 72 | 1JvyT... | 20,709,143,168 | 2 | 1Q3Ra... | 17,466,170,576 | 2 |
| 12igy... | 520,417,170,656 | 27 | 1LV1u... | 109,057,555,744 | 14 | 1ARpm... | 39,879,195,180 | 2 |
| 1CChw... | 2,147,350,272 | 0 | 18p8H... | 102,476,205,104 | 7 | 19WS1... | 35,148,570,624 | 0 |
| 1FLaP... | 19,624,097,408 | 0 | 13fV6... | 11,316,227,712 | 1 | 12uqb... | 16,732,044,544 | 0 |
| 15WoP... | 182,515,162,484 | 6 | 17pXe... | 52,436,585,344 | 1 | 16UCd... | 26,741,572,480 | 1 |
| 1Dxgm... | 4,062,457,344 | 0 | 1J77q... | 87,764,939,376 | 3 | 1L2Ji... | 15,847,872,256 | 2 |
| 124dy... | 176,867,929,600 | 6 | 1B5Ye... | 97,768,452,736 | 1 | 1LTZY... | 2,580,223,544 | 0 |
| 1G9Bp... | 119,790,292,272 | 8 | 1Axys... | 2,703,766,736 | 0 | 1Bitm... | 60,076,837,712 | 7 |
| 16aCW... | 1,321,008,768 | 1 | 1DMoe... | 55,222,216,704 | 0 | 159QB... | 30,577,090,380 | 1 |
| 1DBgt... | 49,867,838,256 | 1 | 14gdn... | 38,655,262,720 | 0 | 13d1h... | 64,919,323,472 | 7 |
| 191R2... | 161,416,275,072 | 5 | 1LTLE... | 6,062,968,704 | 0 | 1HGYn... | 13,609,541,760 | 0 |
| 1BgcK... | 98,735,700,624 | 4 | 1g7tQ... | 37,565,265,376 | 0 | 17JkL... | 136,884,607,664 | 1 |
| 1Bso6... | 39,354,938,368 | 1 | 1MHbC... | 8,287,148,160 | 0 | 1Gu8z... | 34,191,432,784 | 6 |
| | | | 1QFHe... | 9,007,225,472 | 0 | | | |

TABLE VI: Numbers of shares and blocks submitted by 58 major users in Eligius pool from Jan-2014 to Nov-2014.

The proof is simple. since both $\alpha$ and $\beta$ are in the range $(0, 1)$, the denominator and numerator of (6) are both positive. Thus,

$$\begin{aligned}(6) &\Leftrightarrow \alpha\beta - \alpha\beta^2 + 1 - \alpha > (1-\alpha\beta)(1+\alpha\beta-\alpha)\\ &\Leftrightarrow \alpha\beta - \alpha\beta^2 + 1 - \alpha - 1 + \alpha^2\beta^2 + \alpha - \alpha^2\beta > 0\\ &\Leftrightarrow \alpha\beta - \alpha\beta^2 + \alpha^2\beta^2 - \alpha^2\beta > 0\\ &\Leftrightarrow \alpha\beta(1-\alpha)(1-\beta) > 0\end{aligned}$$

It always holds since $0 < \alpha, \beta < 1$. □

**Lemma C.2.** *The attacker gains maximum reward when* $\beta = 0.5$ *in Section IV-A.*

*Proof.* We define

$$\mathtt{F}(\alpha,\beta) = \Delta_R = \frac{R}{R'} - 1 = \frac{\alpha\beta - \alpha\beta^2 + 1 - \alpha}{(1-\alpha\beta)(1+\alpha\beta-\alpha)} - 1 = \frac{\alpha\beta(1-\alpha)(1-\beta)}{(1-\alpha\beta)(1+\alpha\beta-\alpha)}$$

as a function represents the fraction of reward that the attacker gains by performing the BWH attack. We will show that

$$\forall 0 < \beta, \alpha < 1, \mathtt{F}(\alpha,\beta) \le \mathtt{F}(\alpha, 0.5) \tag{7}$$

Our proof is as follows:

$$\begin{aligned}(7) &\Leftrightarrow \mathtt{F}(\alpha,\beta) - \mathtt{F}(\alpha,0.5) \le 0\\ &\Leftrightarrow \frac{\alpha\beta(1-\alpha)(1-\beta)}{(1-\alpha\beta)(1+\alpha\beta-\alpha)} \le \frac{\alpha(1-\alpha)}{(2-\alpha)^2}\\ &\Leftrightarrow \frac{\beta(1-\beta)}{1+\alpha\beta-\alpha} \le \frac{1}{(2-\alpha)^2} \text{ (since } 0<\alpha,\beta<1)\\ &\Leftrightarrow (4-4\alpha+\alpha^2)(\beta-\beta^2) \le (1-\alpha\beta)(1+\alpha\beta-\alpha)\\ &\Leftrightarrow 4\beta(1-\beta)(1-\alpha) \le 1-\alpha\\ &\Leftrightarrow 4\beta(1-\beta) \le 1\\ &\Leftrightarrow (2\beta-1)^2 \ge 0\end{aligned}$$

□

**Lemma C.3.** *The reward scheme in Section VI-B is still a FRS.*

*Proof.* We have the expected reward for a miner with mining power $\alpha$ is $\alpha$ when joining a pool $p$ with normal payoff schemes. Now consider the scheme in which the pool pays $x (0 \le x \le 1)$ portion of the block value to the founder. If we excluded the extra reward by submitting the block out of the total reward, the miner would get

$$\alpha(1-x) \tag{8}$$

reward on average. However, the miner is expected to contribute $\alpha$ portion of all the blocks that the pool found. Thus on average, he will get $\alpha$ of the extra reward given to the founder of the blocks. In other words, he expects $\alpha x$ reward more for submitting valid blocks to the operator. Combining with (8), we have the average reward that the miner gets is $\alpha$ — same as of normal fair payoff schemes. □

**Lemma C.4.** *Given $\vec{\beta} = (\beta_0, \beta_1, ...\beta_n)$, a pool $P_k$, where the attacker spends $\alpha\beta_k > 0$ power to mine on pool $P_k$, the extra reward $\Delta_{R1}$ when the attacker attacks the pool $P_k$ (or $\gamma_k = 1$) is always greater than $\Delta_{R0}$ when he honestly mines on $P_k$ (or $\gamma_k = 0$) (Section IV-C).*

*Proof.* From (3), we have

$$\Delta_{R0} = \sum_{1 \le i, \gamma_i = 1} \frac{1 - \alpha\beta_i}{1 - \alpha \sum_{\gamma_i = 1} \beta_i} \times \Delta_i$$

$$\Delta_{R1} = \Delta_{R0} \times \frac{1 - \alpha \sum_{\gamma_i = 1, i \neq k} \beta_i}{1 - \alpha \sum_{\gamma_i = 1, i \neq k} \beta_i - \alpha\beta_k} + \frac{1 - \alpha\beta_k}{1 - \alpha \sum_{\gamma_i = 1} \beta_i} \times \Delta_k$$

It is easy to see that $\Delta_{R0} < \Delta_{R1}$. □

**Lemma C.5.** *For non-technical solution in Section VI-B, $x = \alpha$ is the smallest fraction necessary to dissuade an attacker completely.*

*Proof.* Following the analogous analysis in Section IV-B, the attacker reward includes two parts: the reward from the pool

$$R_1 = \frac{p'}{1 - \alpha\beta} \times \frac{\alpha\beta}{p' + \alpha\beta} \times (1 - x)$$

and the reward from honest mining with the rest of his power

$$R_0 = \frac{\alpha(1 - \beta)}{1 - \alpha\beta}$$

Thus his total reward is

$$R = \frac{-\alpha^2\beta^2 + \alpha(p' + \alpha\beta)}{(p' + \alpha\beta)(1 - \alpha\beta)} - \frac{\alpha\beta p'}{1 - \alpha\beta} x$$

and the relative extra reward that he gets is

$$\Delta_R = \frac{R}{R_h} - 1 = \frac{\alpha\beta(p' + \alpha\beta - \beta)}{(p' + \alpha\beta)(1 - \alpha\beta)} - \frac{\beta p'}{(p' + \alpha\beta)(1 - \alpha\beta)} x.$$

To disincentivize the attack, we must choose $x$ such that $\Delta_R < 0$, or

$$x > \alpha + \frac{\alpha(\alpha\beta - \beta)}{p'} \tag{9}$$

From Section IV-B, we have $0 \le \beta \le \frac{p'}{1-\alpha}$, which makes $0 \le \alpha + \frac{\alpha(\alpha\beta - \beta)}{p'} \le \alpha$. Thus, $x = \alpha$ will ensure that the attacker of mining power up to $\alpha$ will not be incentived to perform the attack. □

**Lemma C.6.** *For any given network state, there exists a strategy for the pool to make his attack profitable (Section IV-F).*

*Proof.* Without loss of generality, we show that, given a fixed network state $(\alpha_1, \alpha_2, c_2)$, there exists $c_1$ that maximizes $R_1$ and makes a loss on $R_2$. We have the formulas for $R_1, R_2$ as below

$$R_1 = \frac{1}{1 - c_1 - c_2} \left( \frac{(\alpha_1 - c_1)^2}{\alpha_1 - c_1 + c_2} + \frac{c_1(\alpha_2 - c_2)}{\alpha_2 - c_2 + c_1} \right)$$

$$R_2 = \frac{1}{1 - c_1 - c_2} \left( \frac{(\alpha_2 - c_2)^2}{\alpha_2 - c_2 + c_1} + \frac{c_2(\alpha_1 - c_1)}{\alpha_1 - c_1 + c_2} \right)$$

If both the miners are honest, i.e., $c_1 = c_2 = 0$, we have $R_1 = \alpha_1$ and $R_2 = \alpha_2$. Thus, if any of the miners carry out the BWH attack by selecting his best value $c_i$ on the other pool, his reward would increase while the other's will decrease. For example, if $P_2$ properly attacks $P_1$, we will have $R_2 > \alpha_2$ and $R_1 < \alpha_1$, thus $\frac{R_1}{R_2} < \frac{\alpha_1}{\alpha_2}$.

We show that, given any fixed value of $(\alpha_1, \alpha_2, c_2)$ that $P_2$ optimally picks, there exits $c_1$ that makes $\frac{R_1}{R_2} > \frac{\alpha_1}{\alpha_2}$. We have

$$\begin{aligned}
\frac{R_1}{R_2} &= \frac{\frac{(\alpha_1 - c_1)^2}{\alpha_1 - c_1 + c_2} + \frac{c_1(\alpha_2 - c_2)}{\alpha_2 - c_2 + c_1}}{\frac{(\alpha_2 - c_2)^2}{\alpha_2 - c_2 + c_1} + \frac{c_2(\alpha_1 - c_1)}{\alpha_1 - c_1 + c_2}} \\
&= \frac{\frac{(\alpha_1 - c_1)(\alpha_2 - c_2 + c_1)}{\alpha_2 - c_2} + \frac{c_1(\alpha_1 - c_1 + c_2)}{\alpha_1 - c_1}}{\frac{(\alpha_2 - c_2)(\alpha_1 - c_1 + c_2)}{\alpha_1 - c_1} + \frac{c_2(\alpha_2 - c_2 + c_1)}{\alpha_2 - c_2}} \\
&= \frac{(\alpha_1 - c_1) + \frac{c_1}{\alpha_2 - c_2} + c_1 + \frac{c_2}{\alpha_1 - c_1}}{(\alpha_2 - c_2) + \frac{c_2}{\alpha_1 - c_1} + c_2 + \frac{c_1}{\alpha_2 - c_2}} \\
&= \frac{\alpha_1 + \frac{c_1(\alpha_1 - c_1)}{\alpha_2 - c_2} + \frac{c_2 c_1}{\alpha_1 - c_1}}{\alpha_2 + \frac{c_2(\alpha_2 - c_2)}{\alpha_1 - c_1} + \frac{c_1 c_2}{\alpha_2 - c_2}}
\end{aligned}$$

Thus

$$\begin{aligned}
&\frac{R_1}{R_2} > \frac{\alpha_1}{\alpha_2} \\
\Leftrightarrow &\left( \frac{c_1(\alpha_1 - c_1)}{\alpha_2 - c_2} + \frac{c_1 c_2}{\alpha_1 - c_1} \right) \alpha_2 > \\
&\left( \frac{c_2(\alpha_2 - c_2)}{\alpha_1 - c_1} + \frac{c_1 c_2}{\alpha_2 - c_2} \right) \alpha_1 \\
\Leftrightarrow &(\alpha_1\alpha_2 - c_1\alpha_2 - \alpha_1 c_2)(c_1^2 - c_1\alpha_1 + c_2^2 - c_2\alpha_2) < 0
\end{aligned} \tag{10}$$

It is trivial to see that there exists $c_1 < \alpha_1$ to make Inequation 10 valid. □

**Lemma C.7.** *A player of computing power $\alpha$ using $c$ as the contamination factor to attack will only gain higher reward rate if the inequation in Theorem V-B.1 is satisfied (Theorem V-B.1).*

*Proof.* Denote $T_h$ and $T$ are the original time to find a block, the time when the attack happens respectively. We have $\Delta_T = T - T_h$. When A miner of computing power $\alpha$ using $c$ as the contamination factor to attack, only $1 - c$ fraction of the network power contributing to find the block. Thus, the time to find a block increased as $T = T_h/(1 - c)$, giving us $\Delta_T = T_h \frac{c}{1-c}$.

The attacker gains better reward rate only if:

$$\begin{aligned} \frac{R}{T} &> \frac{R_h}{T_h} \\ \Leftrightarrow (\alpha + \Delta_R) T_h &> \alpha T_h \frac{c}{1-c} \\ \Leftrightarrow \Delta_R &> \alpha \frac{c}{1-c} \\ \Leftrightarrow \frac{\Delta_R}{\alpha} &> \frac{c}{1-c} \end{aligned}$$

□