The Miner's Dilemma

Ittay Eyal
Cornell University

***Abstract*—An open distributed system can be secured by requiring participants to present proof of work and rewarding them for participation. The Bitcoin digital currency introduced this mechanism, which is adopted by almost all contemporary digital currencies and related services.**

A natural process leads participants of such systems to form pools, where members aggregate their power and share the rewards. Experience with Bitcoin shows that the largest pools are often open, allowing anyone to join. It has long been known that a member can sabotage an open pool by seemingly joining it but never sharing its proofs of work. The pool shares its revenue with the attacker, and so each of its participants earns less.

We define and analyze a game where pools use some of their participants to infiltrate other pools and perform such an attack. With any number of pools, no-pool-attacks is not a Nash equilibrium. We study the special cases where either two pools or any number of identical pools play the game and the rest of the participants are uninvolved. In both of these cases there exists an equilibrium that constitutes a tragedy of the commons where the participating pools attack one another and earn less than they would have if none had attacked.

For two pools, the decision whether or not to attack is *the miner's dilemma*, an instance of the iterative prisoner's dilemma. The game is played daily by the active Bitcoin pools, which apparently choose not to attack. If this balance breaks, the revenue of open pools might diminish, making them unattractive to participants.

I. Introduction

Bitcoin [1] is a digital currency that is gaining acceptance [2] and recognition [3], with an estimated market capitalization of over 4.5 billion US dollars, as of November 2014 [4]. Bitcoin's security stems from a robust incentive system. Participants are required to provide expensive proofs of work, and they are rewarded according to their efforts. This architecture has proved both stable and scalable, and it is used by most contemporary digital currencies and related services, e.g. [5], [6], [7], [8], [9]. Our results apply to all such incentive systems, but we use Bitcoin terminology and examples since it serves as an active and archetypal example.

Bitcoin implements its incentive systems with a data structure called the *blockchain*. The blockchain is a serialization of all Bitcoin transactions. It is a single global ledger maintained by an open distributed system. Since anyone can join the open system and participate in maintaining the blockchain, Bitcoin uses a *proof of work* mechanism to deter attacks: participation requires exerting significant compute resources. A participant that proves she has exerted enough resources with a proof of work is allowed to take a step in the protocol by generating a block. Participants are compensated for their efforts with newly minted Bitcoins. The process of creating a block is called *mining*, and the participants — *miners*.

In order to win the reward, many miners try to generate blocks. The system automatically adjusts the *difficulty* of block generation, such that one block is added every 10 minutes to the blockchain. This means that each miner seldom generates a block. Although its revenue may be positive in expectation, a miner may have to wait for an extended period to create a block and earn the actual Bitcoins. Therefore, miners form *mining pools*, where all members mine concurrently and they share their revenue whenever one of them creates a block.

Pools are typically implemented as a *pool manager* and a cohort of miners. The pool manager joins the Bitcoin system as a single miner. Instead of generating proof of work, it outsources the work to the miners. In order to evaluate the miners' efforts, the pool manager accepts partial proof of work and estimates each miner's *power* according to the rate with which it submits such partial proof of work. When a miner generates a full proof of work, it sends it to the pool manager which publishes this proof of work to the Bitcoin system. The pool manager thus receives the full revenue of the block and distributes it fairly according to its members power. Many of the pools are open — they allow any miner to join them using a public Internet interface.

Such open pools are susceptible to the classical *block withholding attack* [10], where a miner sends only partial proof of work to the pool manager and discards full proof of work. Due to the partial proof of work it sends to the pool, the miner is considered a regular pool member and the pool can estimate its power. Therefore, the attacker shares the revenue obtained by the other pool members, but does not contribute. It reduces the revenue of the other members, but also its own. We provide necessary background on the Bitcoin protocol, pools and the classical block withholding attack in Section II, and specify our model in Section III. For a broader view of the protocol and ecosystem the reader may refer to the survey by Bonneau et al. [11].

In this work we analyze block withholding attacks among pools. A pool that employs the *pool block withholding attack* registers with the victim pool as a regular miner. It receives tasks from the victim pool and transfers them to some of its own miners. We call these *infiltrating* miners, and the mining power spent by a pool the *infiltration rate*. When the attacking pool's infiltrating miners deliver partial proofs of work, the attacker transfers them to the victim pool, letting the attacked pool estimate their power. When the infiltrating miners deliver a full proof of work, the attacking pool discards it.

This attack affects the revenues of the pools in several ways. The victim pool's effective mining rate is unchanged, but its total revenue is divided among more miners. The attacker's mining power is reduced, since some of its miners are used for block withholding, but it earns additional revenue through its infiltration of the other pool. And finally, the total effective mining power in the system is reduced, causing the Bitcoin protocol to reduce the difficulty.

Taking all these factors into account, we observe that a pool might be able to increase its revenue by attacking other pools. Each pool therefore makes a choice of whether to attack each of the other pools in the system, and with what infiltration rate. This gives rise to the *pool game*. We specify this game and provide initial analysis in Section IV.

In Section V we analyze the scenario where exactly two of the pools take part in the game and only one can attack the other. Here, the attacker can always increase its revenue by attacking. We conclude that in the general case, with any number of pools, no-pool-attacks is not a Nash equilibrium.

Next, Section VI deals with the case of two pools, where each can attack the other. Here, analysis becomes more complicated in two ways. First, the revenue of each pool affects the revenue of the other through the infiltrating miners. We prove that for a static choice of infiltration rates the pool revenues converge. Second, once one pool changes its infiltration rate of the other, the latter may prefer to change its infiltration rate of the former. Therefore the game itself takes multiple rounds to converge. We show analytically that the game has a single Nash Equilibrium and numerically study the equilibrium points for different pool sizes. For pools smaller than 50%, at the equilibrium point both pools earn less than they would have in the non-equilibrium no-one-attacks strategy.

Since pools can decide to start or stop attacking at any point, this can be modeled as the *miner's dilemma* — an instance of the iterative prisoner's dilemma. Attacking is the dominant strategy in each iteration, but if the pools can agree not to attack, both benefit in the long run.

Finally, we address in Section VII the case where the participants are an arbitrary number of identical pools. There exists a symmetric equilibrium in which each participating pool attacks each of the other participating pools. As in the minority two-pools scenario, here too at equilibrium all pools earn less than with the no-pool-attacks strategy.

Our results imply that block withholding by pools leads to an unfavorable equilibrium. Nevertheless, due to the anonymity of miners, a single pool might be tempted to attack, leading the other pools to attack as well. The implications might be devastating for open pools: If their revenues are reduced, miners will prefer to form closed pools that cannot be attacked in this manner. Though this may be conceived as bad news for public mining pools, on the whole it may be good news to the Bitcoin system, which prefers small pools. We examine the practicality of the attack in Section VIII and discuss implications and model extensions in Section IX.

In summary, our contributions are the following:

1) Definition of the pool game where pools in a proof-of-work secured system attack one another with a pool block withholding attack.
2) In the general case, no-pool-attacks is not an equilibrium.
3) With two minority pools participating, the only Nash Equilibrium is when the pools attack one another, and both earn less than if none had attacked.
   Miners therefore face the miner's dilemma, an instance of the iterative prisoner's dilemma, repeatedly choosing between attack and no-attack.
4) With multiple pools of equal size there is a symmetric Nash equilibrium, where all pools earn less than if none had attacked.
5) For Bitcoin, inefficient equilibria for open pools may serve the system by reducing their attraction and pushing miners towards smaller closed pools.

The classical block withholding attack is old as pools themselves, but its use by pools has not been suggested until recently. We overview related attacks and prior work in Section X, and conclude with final remarks in Section XI.

II. Preliminaries — Bitcoin and Pooled Mining

Bitcoin is a distributed, decentralized digital currency [12], [13], [1], [14]. Clients use the system by issuing transactions, and the system's only task is to serialize transactions in a single ledger and reject transactions that cannot be serialized due to conflicts with previous transactions. Bitcoin transactions are protected with cryptographic techniques that ensure that only the rightful owner of a Bitcoin can transfer it. The transaction ledger is stored by a network of *miners* in a data structure caller the *blockchain*.

A. Revenue for Proof Of Work

The blockchain records the transactions in units of blocks. The first block, dubbed *the genesis block*, is defined as part of the protocol. A valid block contains the hash of the previous block, the hash of the transactions in the current block, and a Bitcoin address which is to be credited with a reward for generating the block.

Any miner may add a valid block to the chain by (probabilistically) proving that it has spent a certain amount of work and publishing the block with the proof over an overlay network to all other miners. When a miner creates a block, it is compensated for its efforts with Bitcoins. This compensation includes a per-transaction fee paid by the users whose transactions are included, and an amount of minted Bitcoins that are thus introduced into the system.

The work which a miner is required to do is to repeatedly calculate a a hash function — specifically the SHA-256 of the SHA-256 of a block header. To indicate that he has performed this work, the miner provides a probabilistic proof as follows. The generated block has a nonce field, which can contain any value. The miner places different values in this field and calculates the hash for each value. If the result of the hash is smaller than a target value, the nonce is considered a solution, and the block is valid.

The number of attempts to find a single hash is therefore random with a geometric distribution, as each attempt is a Bernoulli trial with a success probability determined by the target value. At the existing huge hashing rates and small target values, the time to find a single hash can be approximated by an exponential distribution. The average time for a miner to find a solution is therefore proportional to its hashing rate or *mining power*.

To maintain a constant rate of Bitcoin generation, and as part of its defense against denial of service and other attacks, the system normalizes the rate of block generation. To achieve this, the protocol deterministically defines the target value for each block according to the time required to generate recent blocks. The target, or *difficulty*, is updated once every 2016 blocks such that the average time for each block to be found is 10 minutes.

Note that the exponential distribution is memoryless. If all miners mine for block number b, once the block is found at time t, all miners switch to mine for the subsequent block $b+1$ at t without changing their probability distribution of finding a block after t. Therefore, the probability that a miner i with mining power m_i finds the next block is its ratio out of the *total mining power* m in the system.

Forks

Block propagation in the overlay network takes seconds, therefore it is possible for two distant miners to generate competing blocks, both of which name the same block as their predecessor. Such bifurcations, or forks, are rare since the average mining interval is 10 minutes, and they occur on average once every 60 blocks [15]. The system has a mechanism to solve forks when they do occur, causing one of the blocks to be discarded.

We ignore bifurcations for the sake of simplicity. Since the choice of the discarded block on bifurcation is random, one may incorporate this event into the probability of finding a block, and consider instead the probability of finding a block that is not discarded.

B. Pools

As the value of Bitcoin rose, Bitcoin mining has become a rapidly advancing industry. Technological advancements lead to ever more efficient hashing ASICs [16], and mining

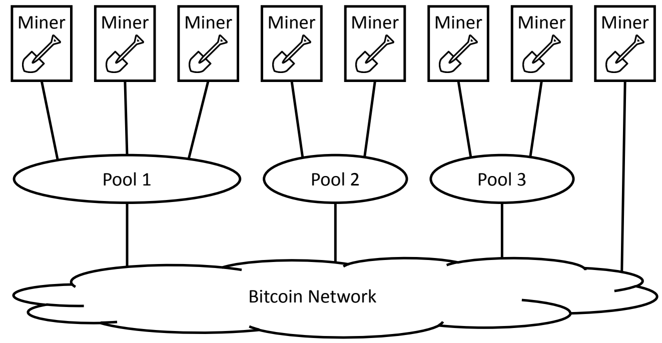


Figure 1. A system with 8 miners and 3 honest pools. Pool 1 has 3 registered miners, pools 2 and 3 have 2 registered miners each, and one miner mines solo.

datacenters are built around the world [17]. Mining is only profitable using dedicated hardware in cutting edge mining rigs, otherwise the energy costs exceed the expected revenue.

Although expected revenue from mining is proportional to the power of the mining rigs used, a single home miner using a small rig is unlikely to mine a block for years [18]. Consequently, miners often organize themselves into mining *pools*. Logically, a pool is a group of miners that share their revenues when one of them successfully mines a block. For each block found, the revenue is distributed among the pool members in proportion to their mining power[1]. The expected revenue of a pool member is therefore the same as its revenue had it mined *solo*. However, due to the large power of the pool, it finds blocks at a much higher rate, and so the frequency of revenue collection is higher, allowing for a stable daily or weekly income.

In practice, most pools are controlled by a centralized pool manager.[2] Miners register with the pool manager and mine on its behalf: The pool manager generates tasks and the miners search for solutions based on these tasks that can serve as proof of work. Once they find a solution, they send it to the pool manager. The pool manager behaves as a single miner in the Bitcoin system. Once it obtains a legitimate block from one of its miners, it publishes it. The block transfers the revenue to the control of the pool manager. The pool manager then distributes the revenue among the miners according to their mining power. The architecture is illustrated in Figure 1

In order to estimate the mining power of a miner, the pool manager sets a partial target for each member, much larger (i.e., easier) than the target of the Bitcoin system. Each miner is required to send the pool manager blocks that are correct according to the partial target. The partial target is chosen to be large, such that partial solutions arrive frequently enough for the manager to accurately estimate the power of the miner, but small (hard) to reduce management overhead.

[1]This is a simplification that is sufficient for our analysis. The intricacies of reward systems are explained in [10].

[2]A notable exception is P2Pool [19], which we discuss in Section IX.

Pools often charge a small percentage of the revenue as fee. We discuss in Section IX the implications of such fees to our analysis.

Many pools are open and accept any interested miner. A pool interface is typically comprised of a web interface for registration and a miner interface for the mining software. In order to mine for a pool, a miner registers with the web interface, supplies a Bitcoin address to receive its future shares of the revenue, and receives from the pool credentials for mining. Then he feeds his credentials and the pool's address to its mining rig, which starts mining. The mining rig obtains its tasks from the pool and sends partial and full proof of work, typically with the STRATUM protocol [20]. As it finds blocks, the pool manager credits the miner's account according to its share of the work, and transfers these funds either on request or automatically to the aforementioned Bitcoin address.

Too Big Pools

Despite their important role of enabling small-scale mining, pools can constitute a threat to the Bitcoin system if their size is too large. If one pool controls the majority of mining power, the system becomes unstable [21], [22] (and [23] warns that the system is unstable with even smaller pools).

Arguably, in realistic scenarios of the Bitcoin system no pool controls a majority of the mining power. As an example, for one day in June 2014 a single pool called GHash.IO produced over 50% of the blocks in the Bitcoin main chain. The Bitcoin community backlashed at the pool (which has done nothing worse than being extremely successful). GHash.IO reduced its relative mining power and publicly committed to stay away from the 50% limit.

C. Block Withholding and its Detection

Classical Block Withholding [10] is an attack performed by a pool member against the other pool members. The attacking miner registers with the pool and apparently starts mining honestly — it regularly sends the pool partial proof of work. However, the attacking miner sends only partial proof of work. If it finds a full solution that constitutes a full proof of work it discards the solution, reducing the pool's total revenue.[3] This attack is illustrated in Figure 2.

The attacker does not change the pool's effective mining power, and does not affect directly the revenue of other pools. However, the attacked pool shares its revenue with the attacker. Therefore each miner earns less, as the same revenue is distributed among more miners.

Recall that the proof of work is only valid for a specific block, as it is the nonce with which the block's hash is smaller than its target. The attacking miner cannot use it.

[3]Although the term block withholding has become canonical, note that the block is discarded and never introduced into the system as the name block withholding implies.

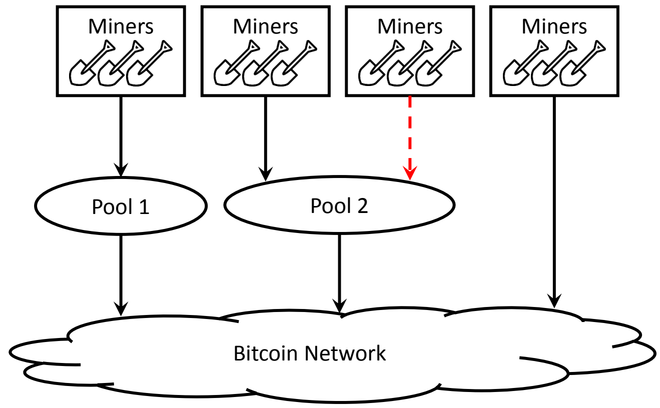


Figure 2. Classical Block Withholding attack. A group of miners attack Pool 2 with a block withholding attack, denoted by a dashed red arrow.

Moreover, this attack reduces the attacker's revenue compared to solo mining or honest pool participation: It suffers from the reduced revenue like the other pool participants, and its revenue is less than its share of the total mining power in the system. This attack can therefore only be used for sabotage, at a cost to the attacker.

Detection: Even if a pool detects that it is under a block withholding attack, it might not be able to detect which of its registered miners are the perpetrators. A pool can estimate its expected mining power and its actual mining power by the rates of partial proofs of work and full proofs of work, respectively, supplied by its miners. A difference above a set confidence interval indicates an attack. To detect whether a single miner is attacking it, the pool must use a similar technique, comparing the estimated mining power of the attacker based on its partial proof of work with the fact it never supplies a full proof of work. If the attacker has a small mining power, it will send frequent partial proofs of work, but the pool will only expect to see a full proof of work at very low frequency. Therefore, it cannot obtain statistically significant results that would indicate an attack.

An attacker can use multiple small block withholding miners and replace them frequently. A small miner is, for example, a miners whose expected full proof of work frequency is yearly. Such a miner will see a non-negligible average daily revenue (฿$25/365 \approx$ ฿0.07). If the attacker replaces such a small miner every month, he will collect about ฿2 at the end of each month. The pool must decide within this month whether the miner is an attacker (and revoke its earnings), or just an unlucky honest miner. Since an honest miner of this power is unlikely to find a full proof of work within a month (probability of about 8% according to the exponential distribution) a pool that rejects miners based on this criterion would reject the majority of its honest miners. The alternative of rejecting small miners in general or distributing revenue on a yearly basis contradicts the goal of pooled mining.

III. Model and Standard Operation

We specify the basic model in which participants operate in Section III-A, proceed to describe how honest miners operate in this environment in Sections III-B and III-C, and how the classical block withholding attack is implemented with our model in Section III-D.

A. Model

The system is comprised of the Bitcoin network and nodes with unique IDs, and progresses in steps. A node i generates *tasks* which are associated with its ID i.

A node can work on a task for the duration of a step. The result of this work is a set of partial proofs of work and a set of full proofs of work. The number of proofs in each set has a Poisson distribution, partial proofs with a large mean and full proofs with a small mean. Nodes that work on tasks are called a miners, miners have identical power, and hence identical probabilities to generate proofs of work.

The Bitcoin network pays for full proofs of work. To acquire this payoff an entity publishes a task *task* and its corresponding proof of work to the network. The payoff goes to the ID associated with *task*. The Bitcoin protocol normalizes revenue such that the average total revenue distributed in each step is a constant throughout the execution of the system. Any node can transact Bitcoins to another node by issuing a Bitcoin transaction.

Nodes that generate tasks but outsource the work are called pools. Pools send tasks to miners over the network, the miners receive the tasks, perform the work, and send the partial and full proofs of work to the pool.

Apart from working on tasks, all local operations, payments, message sending, propagation, and receipt are instantaneous.

We assume that the number of miners is large enough such that mining power can be split arbitrarily without resolution constraints.

Denote the number of pools with p, the total number of mining power in the system with m and the miners participating in pool i ($1 \leq i \leq p$) with m_i. We use a quasi-static analysis where miner participation in a pool does not change over time.

B. Solo Mining

A solo miner is a node that generates its own tasks. In every step it generates a task, works on it for the duration of the step and if it finds a full proof of work, it publishes this proof of work to earn the payoff.

C. Pools

A pool is a node that serves as a coordinator and multiple miners can register to a pool and work for it. In every step it generates a task for each registered miner and sends it over the network. Each miner receives its task and works on it for the duration of the step. At the end of the step, the miner sends the pool the full and the partial proofs of work it has found. The pool receives the proofs of work of all its miners, registers the partial proofs of work and publishes the full proofs. It calculates its overall revenue, and proceeds to distribute it among its miners. Each miner receives revenue proportional to its success in the current step, namely the ratio of its partial proofs of work out of all partial proofs of work the pool received. We assume that pools do not collect fees of the revenue. Pool fees and their implications on our analysis are discussed in Section IX.

D. Block Withholding Miner

A miner registered at a pool can perform the classical block withholding attack. An attacker miner operates as if it worked for the pool. It receives its tasks and works on them, only at the end of each round it sends only its partial proofs of work, and omits full proofs of work if it had found any. The pool registers the miner's partial proofs, but cannot distinguish between miners running honestly and block withholding miners.

The implications are that a miner that engages in block withholding does not contribute to the pool's overall mining power, but still shares the pool's revenue according to its sent partial proofs of work.

To reason about a pool's efficiency we define its per-miner revenue as follows.

Definition 1 (Revenue density)**.** *The* revenue density *of a pool is the ratio between the average revenue a pool member earns and the average revenue it would have earned as a solo miner.*

The revenue density of a solo miner, and that of a miner working with an unattacked pool are one. If a pool is attacked with block withholding, its revenue density decreases.

E. Continuous Analysis

Because our analysis will be of the average revenue, we will consider proofs of work, both full and partial, as continuous deterministic sizes, according to their probability. Work on a task therefore results in a deterministic fraction of proof of work.

IV. The Pool Game

A. The Pool Block Withholding Attack

Just as a miner can perform block withholding on a pool j, a pool i can use some of its mining power to infiltrate a pool j and perform a block withholding attack on j. Denote the amount of such infiltrating mining power at step t by $x_{i,j}(t)$. Miners working for pool i, either mining honestly or used for infiltrating pool j, are *loyal* to pool i. At the end of a round, pool i aggregates its revenue from mining in the current round and from its infiltration in the previous round. It distributes the revenue evenly among all its loyal miners according to their partial proofs of work. The pool's miners are oblivious to their role and they operate as regular honest miners, working on tasks.

B. Revenue Convergence

Note that pool j sends its revenue to infiltrators from pool i at the end of the step, and this revenue is calculated in pool i at the beginning of the subsequent step. If there is a chain of pools of length ℓ where each pool infiltrates the next, the pool revenue will not be static, since the revenue from infiltration takes one step to take each hop. If $\ell_{\max}$ is the longest chain in the system, the revenue stabilizes after $\ell_{\max}$ steps. If there are loops in the infiltration graph, the system will converge to a certain revenue, as stated in the following lemma.

Lemma 1 (Revenue convergence)**.** *If infiltration rates are constant, the pool revenues converge.*

Proof: Denote the revenue density of pool i at the end of step t by $r_i(t)$, and define the revenue density vector

$$\mathbf{r(t)} \triangleq (r_1(t), \ldots, r_p(t))^T \ .$$

In every round, pool i uses its mining power of $m_1 - \sum_j x_{1,j}$ used for direct mining (and not attacking), and shares it among its $m_1 + \sum_j x_{j,1}$ members, including malicious infiltrators (all sums are over the range $1, \ldots, p$). Denote the direct mining revenue density of each pool (ignoring normalization, which is a constant factor) with the vector

$$\mathbf{m} \triangleq \left(\frac{m_1 - \sum_j x_{1,j}}{m_1 + \sum_j x_{j,1}}, \ldots, \frac{m_p - x_{p,j}}{m_p + \sum_j x_{j,p}} \right)^T \ .$$

The revenue of Pool i in step t taken through infiltration from pool j's revenue in step $t-1$ is $x_{i,j} r_j(t-1)$. Pool i distributes this revenue among its $m_i + \sum_k x_{k,i}$ members — loyal and infiltrators. Define the $p \times p$ *infiltration matrix* by its i, j element

$$\mathbf{G} \triangleq \left[\frac{x_{i,j}}{m_i + \sum_k x_{k,i}} \right]_{ij} \ .$$

And the revenue vector at step t is

$$\mathbf{r}(t) = \mathbf{m} + \mathbf{G}\mathbf{r}(t-1) \ . \quad (1)$$

Since the row sums of the infiltration matrix are smaller than one, its largest eigenvalue is smaller than 1 according to the Perron-Frobenius theorem. Therefore, the revenues at all pools converge as follows:

$$\mathbf{r}(t) = \left(\sum_{t'=0}^{t-1} G^{t'} \right) \mathbf{m} + G^t \mathbf{r}(0) \xrightarrow{t \to \infty} (1 - \mathbf{G})^{-1} \mathbf{m} \ . \quad (2)$$

■

C. The Pool Game

In the pool game pools try to optimize their infiltration rates of other pools to maximize their revenue. The overall number of miners and the number of miners loyal to each pool remain constant throughout the game.

Time progresses in rounds. Let s be a constant integer large enough that revenue can be approximated as its convergence limit. In each round the system takes s steps and then a single pool, picked with a round-robin policy, may change its infiltration rates of all other pools. The total revenue of each step is normalized to $1/s$, so the revenue per round is one.

The pool taking a step knows the rate of infiltrators attacking it (though not their identity) and the revenue rates of each of the other pools. This knowledge is required to optimize a pool's revenue, as we see next. We explain in Section VIII how a pool can technically obtain this knowledge.

D. General Analysis

Recall that m_i is the number of miners loyal to pool i. and $x_{i,j}(t)$ is the number of miners used by pool i to infiltrate pool j at step t.

The mining rate of pool i is therefore the number of its loyal miners minus the miners it uses for infiltration. This effective mining rate is divided by the total mining rate in the system, namely the number of all miners that do not engage in block withholding[4]. Denote the direct mining rate of pool i at step t by

$$R_i \triangleq \frac{m_i - \sum_{j=1}^p x_{i,j}}{m - \sum_{j=1}^p \sum_{k=1}^p x_{j,k}} \quad (3)$$

The revenue density of pool i at the end of step t is its revenue from direct mining together with its revenue from infiltrated pools, divided by the number of its loyal miners together with block-withholding infiltrators that attack it:

$$r_i(t) = \frac{R_i(t) + \sum_{j=1}^p x_{i,j}(t) r_j(t)}{m_i + \sum_{j=1}^p x_{j,i}(t)} \ . \quad (4)$$

Hereinafter we move to a static state analysis and omit the t argument in the expressions.

No attack

If no pool engages in block withholding,

$$\forall i, j : x_{i,j} = 0 \ ,$$

and we have

$$\forall i : r_i = 1/m \ ,$$

that is, each miner's revenue is proportional to its power, be it in a pool or working solo.

[4]Recall that difficulty is only adjusted periodically, and there are transient effects that are not covered by this stable-state analysis. We discuss this in Section VIII.

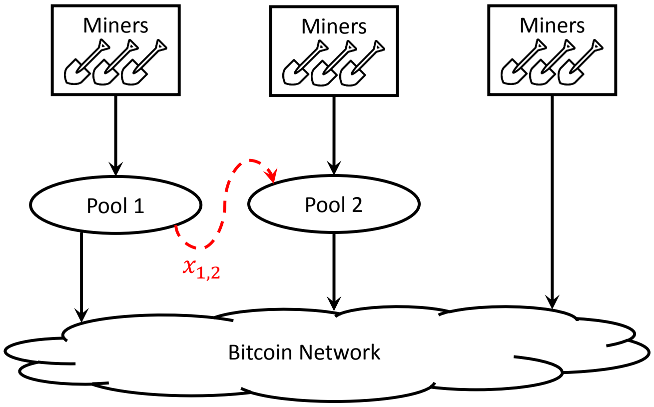


Figure 3. The one-attacker scenario. Pool 1 infiltrates pool 2.

V. One Attacker

We begin our analysis with a simplified game of two pools, 1 and 2, where pool 1 can infiltrate pool 2, but pool 2 cannot infiltrates pool 1. The $m - m_1 - m_2$ miners outside both pools mine solo (or with closed pools that do not attack and cannot be attacked). This scenario is illustrated in Figure 3. The dashed red arrow indicates that $x_{1,2}$ of pool 1's mining power infiltrates pool 2 with a block withholding attack.

Since Pool 2 does not engage in block withholding, all of its m_2 loyal miners work on its behalf. Pool 1, on the other hand does not employ $x_{1,2}$ of its loyal miners, and its direct mining power is only $m_1 - x_{1,2}$. The Bitcoin system normalizes these rates by the total number of miners that publish full proofs, namely all miners but $x_{1,2}$. The pools' direct revenues are therefore

$$\begin{aligned} R_1 &= \frac{m_1 - x_{1,2}}{m - x_{1,2}} \\ R_2 &= \frac{m_2}{m - x_{1,2}} \quad . \end{aligned} \tag{5}$$

Pool 2 divides its revenue among its loyal miners and the miners that infiltrated it. Its revenue density is therefore

$$r_2 = \frac{R_2}{m_2 + x_{1,2}} \quad . \tag{6}$$

Pool 1 divides its revenue among its registered miners. The revenue includes both its direct mining revenue and the revenue its infiltrators obtained from pool 2, which is $r_2 \cdot x_{1,2}$. The revenue per loyal Pool 1 miner is therefore

$$r_1 = \frac{R_1 + x_{1,2} \cdot r_2}{m_1} \quad . \tag{7}$$

We obtain the expression for r_1 in Equation 7 by substituting r_2 from Equation 6 and R_1 and R_2 from equation 5:

$$r_1 = \frac{m_1(m_2 + x_{1,2}) - x_{1,2}^2}{m_1(m - x_{1,2})(m_2 + x_{1,2})}$$

A. Game Progress

Pool 1 controls its infiltration rate of pool 2, namely $x_{1,2}$, and will choose the value that maximizes the *revenue density* (per-miner revenue) r_1 on the first round of the pool game.

The value of r_1 is maximized at a single point in the feasible range $0 \le x_{1,2} \le m_1$. Since pool 2 cannot not react to pool 1's attack, this point is the stable state of the system, and we denote the value of $x_{1,2}$ there by $\bar{x}_{1,2} \triangleq \arg\max_{x_{1,2}} r_1$, and the values of the corresponding revenues of the pools with $\bar{r}_1$ and $\bar{r}_2$.

Substituting the stable value $x_{1,2}$ we obtain the revenues of the two pools; all are given in Figure 4, normalizing $m = 1$ to simplify the expressions.

B. Numerical Analysis

We analyze this game numerically by finding the $x_{1,2}$ that maximizes r_1 and substituting this value for r_1 and r_2. We vary the sizes of the pools through the entire feasible range and depict the optimal $x_{1,2}$ and the corresponding revenues in Figure 5. Each point in each graph represents the equilibrium point of a game with the corresponding m_1 and m_2 sizes, where we normalize $m = 1$. The top right half of the range in all graphs is not feasible, as the sum of m_1 and m_2 is larger than 1. We use this range as a reference color, and we use a dashed line to show the bound between this value within the feasible range.

Figure 5a shows the optimal infiltration rate. In the entire feasible range we see that pool 1 chooses a strictly positive value for $x_{1,2}$. Indeed, the revenue of pool 1 is depicted in Figure 5b and in the entire feasible region it is strictly larger than 1, which the pool would have gotten without attacking ($x_{1,2} = 0$). Figure 5c depicts the revenue of Pool 2, which is strictly smaller than 1 in the entire range.

Third parties: Note that the total system mining power is reduced when pool 1 chooses to infiltrate pool 2. Therefore, the revenue of third parties, miners not in either pool, increases from $1/m$ to $1/(m - x_{1,2})$. Pool 2 therefore pays for the increased revenue of its attacker and everyone else in the system.

C. Implications to the general case

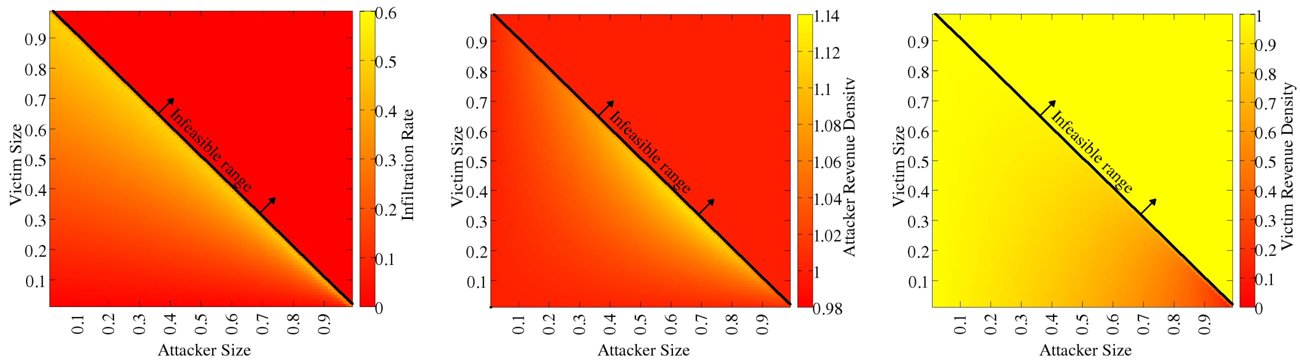
Consider the case of p pools. For any choice of the pools sizes $m_1, \ldots, m_p$, at least one pool will choose to perform block withholding:

Lemma 2. *In a system with p pools, the point $\forall j,k : x_j^k = 0$ is not an equilibrium.*

Proof: Assume towards negation this is not the case, and $\forall j,k : x_j^k = 0$ is an equilibrium point. Now consider a setting with only pools 1 and 2, and treat the other pools as independent miners. This is the setting analyzed above and we have seen there that pool 1 can increase its revenue by performing a block withholding attack on pool 2. Denote pool 1's infiltration rate by $\tilde{x}_{1,2} > 0$. Now, take this values

$$
\begin{aligned}
\bar{x}_{1,2} &= \frac{m_2 - m_1 m_2 - \sqrt{-m_2^2(-1 + m_1 + m_1 m_2)}}{-1 + m_1 + m_2} \\
\bar{r}_1 &= \frac{m_1 + (2 + m_1)m_2 - 2\sqrt{-m_2^2(-1 + m_1 + m_1 m_2)}}{m_1(1 + m_2)^2} \\
\bar{r}_2 &= \frac{m_2(-1 + m_1 + m_2)^2}{\left(m_2^2 - \sqrt{-m_2^2(-1 + m_1 + m_1 m_2)}\right)\left(1 - m_1(1 + m_2) - \sqrt{-m_2^2(-1 + m_1 + m_1 m_2)}\right)}
\end{aligned}
\tag{8}
$$

Figure 4. Stable state where only pool 1 attacks pool 2.



(a) $x_{1,2}$ (b) r_1 (c) r_2

Figure 5. Two pools where one infiltrates the other: Optimal infiltration rate $x_{1,2}$ and corresponding revenues (r_1 and r_2) as a function of pool sizes. The line in (a) shows $x_{1,2} = 0$ and the lines in (b) and (c) show the revenue density of 1.

back to the setting at hand with p pools. The revenue of pool 1 is better when

$$x_{1,2} = \tilde{x}_{1,2}, \forall (j,k) \neq (1,2) : x_{j,k} = 0 \ .$$

Therefore, pool 1 can improve its revenue by attacking pool 2, and no-one-attacks is not an equilibrium point. ■

D. Test-case

As a test case, we take the pool distribution in January 16, 2015 [24], shown in Figure 6. We analyze the cases where each of the pools attacks all other open pools, all of which behave honestly. Note that attacking all pools with force proportional to their size yields the same results as attacking a single pool of their aggregate size. Plugging in the numbers into the analysis above shows that a larger pool needs to use a smaller ratio of its mining power for infiltration and can increase its revenue density more than a small pool. The largest pool, DiscusFish, achieves its optimum attack rate at 25% of the pool's mining power, increasing its revenue by almost 3%. This amounts to a daily revenue increase of ฿26 Bitcoin, or almost 5500 USD at the exchange rate on that date. This represents a considerable increase of the pools net revenue. However, for the smallest pool, Eligius, the attack is much less profitable. To reach the optimum it needs almost a third of its power for attacking but increases its revenue density by merely 0.6%, amounting to ฿0.86 a day or 18 USD.

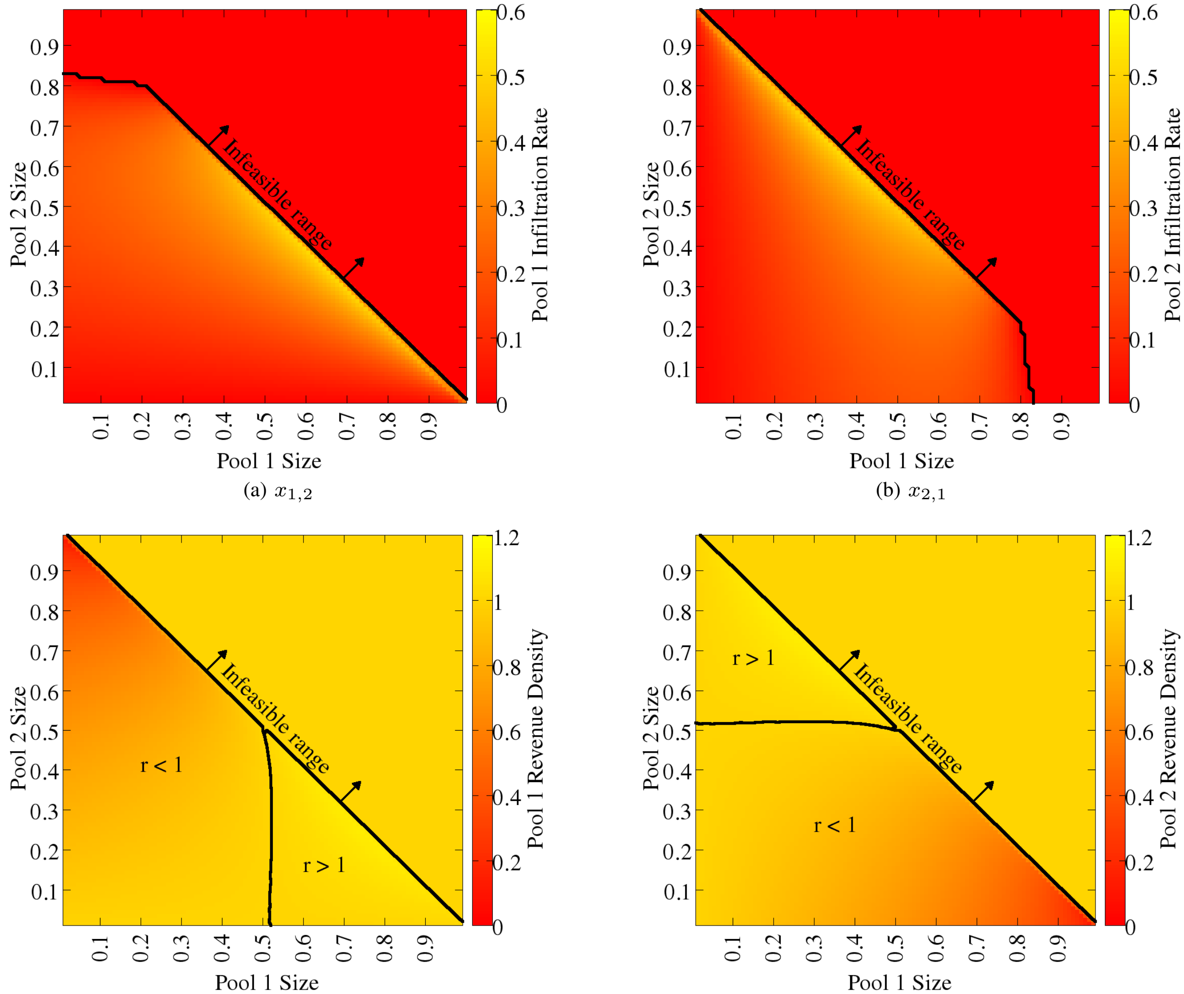
Name	**Size**	**Infiltration Rate**	**Revenue Density**
DiscusFish	24%	25%	102.9%
AntPool	13%	28%	101.8%
GHash.IO	10%	30%	101.5%
BTChine	7%	30%	101.1%
BTCGuild	6%	30%	100.9%
Eligius	4%	32%	100.6%
Others	36%	-	-

Figure 6. The six largest open pool sizes as of January 16, 2015 [24], their optimal infiltration rates (of each pool as a fraction of its size, if it attacked all others without reciprocation), and their revenue density when attacking.

VI. Two Pools

We proceed to analyze the case where two pools may attack each other and the other miners mine solo. Again we have pool 1 of size m_1 and pool 2 of size m_2; pool 1 controls its infiltration rate $x_{1,2}$ of pool 2, but now pool 2 also controls its infiltration rate $x_{2,1}$ of pool 1. This scenario is illustrated in Figure 8

The total mining power in the system is $m - x_{1,2} - x_{2,1}$. The direct revenues R_1 and R_2 of the pools from mining are their effective mining rates, without infiltrating mining



(a) $x_{1,2}$ (b) $x_{2,1}$ (c) r_1 (d) r_2

Figure 7. Two attacking pools system: Optimal infiltration rates (x_1 and x_2) and corresponding revenues (r_1 and r_2) as a function of pool sizes. Lines in (a) and (b) are at $x_{1,2} = 0$ and $x_{2,1} = 0$, respectively. Lines in (c) and (d) are at $r_1 = 1$ and $r_2 = 1$, respectively.

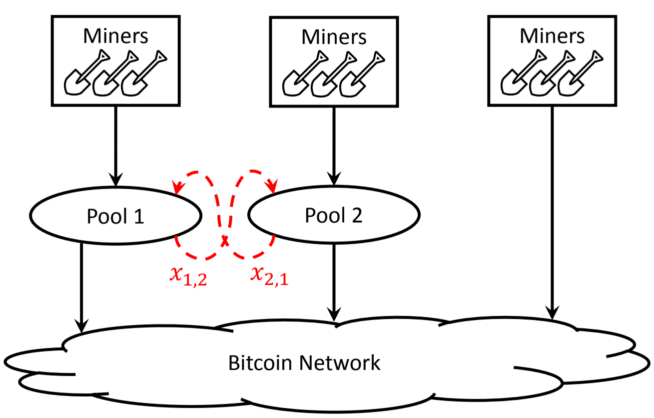


Figure 8. Two pools infiltrating each other.

power, divided by the total mining rate.

$$
\begin{aligned}
R_1 &= \frac{m_1 - x_{1,2}}{m - x_{1,2} - x_{2,1}} \\
R_2 &= \frac{m_2 - x_{2,1}}{m - x_{1,2} - x_{2,1}} \quad .
\end{aligned}
\tag{9}
$$

The total revenue of each pool is its direct mining revenue, above, and the infiltration revenue from the previous round, which is the attacked pool's total revenue multiplied by its infiltration rate. The pool's total revenue is divided among its loyal miners and miners that infiltrated it. At stable state this is

$$
\begin{aligned}
r_1 &= \frac{R_1 + x_{1,2} r_2}{m_1 + x_{2,1}} \\
r_2 &= \frac{R_2 + x_{2,1} r_1}{m_2 + x_{1,2}} \quad .
\end{aligned}
\tag{10}
$$

Solving for r_1 and r_2 we obtain the following closed expressions for each. We express the revenues as functions of $x_{1,2}$ and $x_{2,1}$.

$$\begin{aligned} r_1(x_{1,2}, x_{2,1}) &= \frac{m_2 R_1 + x_{1,2}(R_1 + R_2)}{m_1 m_2 + m_1 x_{1,2} + m_2 x_{2,1}} \\ r_2(x_{2,1}, x_{1,2}) &= \frac{m_1 R_2 + x_{2,1}(R_1 + R_2)}{m_1 m_2 + m_1 x_{1,2} + m_2 x_{2,1}} \quad . \end{aligned} \tag{11}$$

Each pool controls only its own infiltration rate. In each round of the pool game, each pool will optimize its infiltration rate of the other. If pool 1 acts at step t, it optimizes its revenue with

$$x_{1,2}(t) \leftarrow \arg\max_{x'} r_1(x', x_{2,1}(t-1)) \ , \tag{12}$$

and if pool 2 acts at step t, it optimizes its revenue with

$$x_{2,1}(t) \leftarrow \arg\max_{x'} r_2(x', x_{1,2}(t-1)) \ . \tag{13}$$

An equilibrium exists where neither pool 1 nor pool 2 can improve its revenue by changing its infiltration rate. That is, any pair of values x'_1, x'_2 such that

$$\begin{cases} \arg\max_{x_{1,2}} r_1(x_{1,2}, x'_{2,1}) = x'_{1,2} \\ \arg\max_{x_{2,1}} r_2(x'_{1,2}, x_{2,1}) = x'_{2,1} \end{cases} \tag{14}$$

under the constraints

$$\begin{aligned} 0 < x'_1 < m_1 \\ 0 < x'_2 < m_2 \quad . \end{aligned} \tag{15}$$

The feasible region for the pool sizes is $m_1 > 0, m_2 > 0$, and $m_1 + m_2 \leq m$. The revenue function for r_i is concave in x_i for all feasible values of the variables ($\partial^2 r_i / \partial x_i^2 < 0$). Therefore the solutions for equations 12 and 13 are unique and are either at the borders of the feasible region or where $\partial r_i / \partial x_{i,j} = 0$.

From Section V we know that no-attack is not an equilibrium point, since each pool can increase its revenue by choosing a strictly positive infiltration rate, that is, $x_{1,2} = x_{2,1} = 0$ is not a solution to Equations 14–15.

Nash equilibrium therefore exists with $x_{1,2}$, $x_{2,1}$ values where

$$\begin{cases} \dfrac{\partial r_1(x_{1,2}, x_{2,1})}{\partial x_{1,2}} = 0 \\ \dfrac{\partial r_2(x_{2,1}, x_{1,2})}{\partial x_{2,1}} = 0 \end{cases} \quad . \tag{16}$$

Using symbolic computation tools, we see that there is a single pair of values for which Equation 16 holds for any feasible choice of m_1 and m_2.

A. Numerical Analysis

A numerical analysis confirms these observations. We simulate the pool game for a range of pool sizes. For each choice of pool sizes, we start the simulation when both pools do not infiltrate each other, $x_{1,2} = x_{2,1} = 0$, and the revenue densities are $r_1 = r_2 = 1$. At each round one pool chooses its optimal infiltration rate based on the pool sizes and the rate with which it is infiltrated, and we calculate the revenue after convergence with Equation 11. Recall the players in the pool game are chosen with the Round Robin policy, so the pools take turns, and we let the game run until convergence. The results are illustrated in Figure 7.

Each run with some m_1, m_2 values results in a single point in each graph in Figure 7. We depict the infiltration rates of both pools $x_{1,2}, x_{2,1}$ in Figures 7a–7b and the pools' revenue densities r_1, r_2 in Figures 7c–7d. So, for each choice of m_1 and m_2, the values of $x_{1,2}$, $x_{2,1}$, m_1 and m_2 are the points in each of the graphs with the respective coordinates.

For the $x_{i,j}$ graphs we draw a border around the region where there is no-attack by i in equilibrium. For the r_i graphs we draw a line around the region where the revenue is the same as in the no-attack scenario, namely 1.

We first observe that only in extreme cases a pool does not attack its counterpart. Specifically, at equilibrium a pool will refrain from attacking only if the other pool is larger than about 80% of the total mining power.

But, more importantly, we observe that a pool improves its revenue compared to the no-pool-attacks scenario only when it controls a strict majority of the total mining power. These are the small triangular regions in Figures 7c and 7d. In the rest of the space, the trapezoids in the figures, the revenue of the pool is inferior compared to the no-pool-attacks scenario.

B. The Prisoner's Dilemma

In a healthy Bitcoin environment, where neither pool controls a strict majority of the mining power, both pools will earn less at equilibrium than if both pools ran without attacking. We can analyze in this case a game where each pool chooses either to attack and optimize its revenue, or to refrain from attacking.

Consider pool 1 without loss of generality. As we have seen in Section V, if pool 2 does not attack, pool 1 can increase its revenue above 1 by attacking. If pool 2 does attack but pool 1 does not, we denote the revenue of pool 1 by $\tilde{r}_1$. The exact value of $\tilde{r}_1$ depends on the values of m_1 and m_2, but it is always smaller than one. As we have seen above, if pool 1 does choose to attack, its revenue increases, but does not surpass one. The game is summarized in Figure 9.

When played once, this is the classical prisoner's dilemma. Attack is the dominant strategy: Whether pool 2 chooses to attack or not, the revenue of pool 1 is larger when attacking than when refraining from attack, and the same for pool 2. At equilibrium of this attack-or-don't game, when both pools attack, the revenue of each pool is smaller than its revenue if neither pool attacked.

However, the game is not played once, but rather continuously, forming a super-game, where each pool can change its strategy between attack and no-attack. The pools can agree (even implicitly) to refrain from attacking, and in each round

Pool 2 \ Pool 1	no attack	attack
no attack	$(r_1 = 1, r_2 = 1)$	$(r_1 > 1, r_2 = \tilde{r}_2 < 1)$
attack	$(r_1 = \tilde{r}_1 < 1, r_2 > 1)$	$(\tilde{r}_1 < r_1 < 1, \tilde{r}_2 < r_2 < 1)$

Figure 9. Prisoner's Dilemma for two pools. The revenue density of each pool is determined by the decision of both pools whether to attack or not. The dominant strategy of each player is to attack, however the payoff of both would be larger if they both refrain from attacking.

a pool can detect whether it is being attacked and deduce that the other pool is violating the agreement. In this supergame, cooperation where neither pool attacks is a possible stable state [25], [26] despite the fact that the single Nash equilibrium in every round is to attack.

C. Test-case

As an example we take again the pool sizes shown in Figure 6, and study the case where the two largest pools, DiscusFish and AntPool, attack one another. The optimal infiltration rates (out of the total system mining power) are 8% and 12%, respectively, and the pools would lose 4% and 10% of their revenues, respectively, compared to the no-attack scenario.

VII. q Identical Pools

Let there be q pools of identical size that engage in block withholding against one another. Other miners neither attack nor are being attacked. In this case there exists a symmetric equilibrium. Consider, without loss of generality, a step of pool 1. It controls its attack rates each of the other pools, and due to symmetry they are all the same. Denote by $x_{1,\neg 1}$ the attack rate of pool 1 against any other pool. Each of the other pools can attack its peers as well. Due to symmetry, all attack rates by all attackers are identical. Denote by $x_{\neg 1,*}$ the attack rate of any pool other than 1 against any other pool, including pool 1.

Denote by R_1 the direct revenue (from mining) of pool 1 and by $R_{\neg 1}$ the direct revenue of each of the other pools. Similarly denote by r_1 and $r_{\neg 1}$ the revenue densities of pool 1 and other pools, respectively.

The generic equations 3 and 4 are instantiated to

$$\begin{aligned} R_1 &= \frac{m_i - (q-1)x_{1,\neg 1}}{m - (q-1)(q-1)x_{\neg 1,*} - (q-1)x_{1,\neg 1}} \\ R_{\neg 1} &= \frac{m_i - (q-1)x_{\neg 1,*}}{m - (q-1)(q-1)x_{\neg 1,*} - (q-1)x_{1,\neg 1}} \end{aligned} \tag{17}$$

and

$$\begin{aligned} r_1 &= \frac{R_1 + (q-1)x_{1,\neg 1}r_{\neg 1}}{m_i + (q-1)x_{\neg 1,1}} \\ r_{\neg 1} &= \frac{R_{\neg 1} + (q-2)x_{\neg 1,*}r_{\neg 1} + x_{\neg 1,*}r_1}{m_i + (q-2)x_{\neg 1,*} + x_{1,\neg 1}} \end{aligned} . \tag{18}$$

Substituting Equations 17 in Equation 18 and solving we obtain a single expression for any r_i, since in the symmetric case we have $r_1 = r_{\neg 1}$. The expression is shown in Equation 18 (Figure 10).

Given any value of q and m_i (where $qm_i < 1$), the feasible range of the infiltration rates is $0 \le x_{i,j} \le m_i/q$. Within this range r_i is continuous, differentiable, and concave in $x_{1,\neg 1}$. Therefore, the optimal point for pool 1 is where $\partial r_1 / \partial x_{1,\neg 1} = 0$. Since the function is concave the equation yields a single feasible solution, which is a function of the attack rates of the other pools, namely $x_{\neg 1,1}$ and $x_{\neg 1,*}$.

To find a symmetric equilibrium, we equate $x_{1,\neg 1} = x_{\neg 1,1} = x_{\neg 1,*}$ and obtain a single feasible solution. The equilibrium infiltration rate and the matching revenues are shown in Equation 20 (Figure 11).

As in the two-pool scenario, the revenue at the symmetric equilibrium is inferior to the no-one-attacks non-equilibrium strategy.

VIII. Practicalities

A. Ramp-up

Our analysis addresses the eventual revenue of the pools, assuming the mining difficulty is set based on the effective mining power, not including mining power used for withholding. However, difficulty is updated only periodically — every 2016 blocks in Bitcoin. When mining power in the system is regularly increasing, which has been true for the majority of Bitcoin's history [27], no adjustment may be necessary. Specifically, if an attacker purchases new mining hardware and employs it directly for block withholding, this mining power is never included in the difficulty calculation — the system is never aware of it. The difficulty is therefore already correctly calculated and the attack is profitable immediately.

However, if the mining power is static, the attack becomes profitable only after the Bitcoin system has normalized the revenues by adjusting difficulty. Before the adjustment, the revenue of an attacking pool is reduced due to the reduction in block generation of both the attacking and attacked pools.

B. Pool Knowledge

In order to choose its optimal infiltration rate, a pool has to know the rate at which it is attacked, and the revenue density of potential victim pools. A pool can estimate the rate with which it is attacked by comparing the rates of partial and full proofs of work it receives from its miners, as explained in Section II-C. In order to estimate the revenue densities of the other pools, a pool can use one of two methods. First, pools

$$r_i = -\frac{m_i^2 + m_i x_{1,\neg 1} - (q-1)x_{1,\neg 1}((q-1)x_{\neg 1,*} + x_{1,\neg 1})}{((q-1)x_{1,\neg 1} + (q-1)^2 x_{\neg 1,*} - 1)\,((m_i + x_{1,\neg 1})(m_i + (q-1)x_{\neg 1,1}) - (q-1)x_{1,\neg 1}x_{\neg 1,*})} \tag{19}$$

Figure 10. Expression for r_i in a system with pools of equal size.

$$\begin{aligned} \bar{x}_{1,\neg 1} = \bar{x}_{\neg 1,1} = \bar{x}_{\neg 1,*} &= \frac{q - m_i - \sqrt{(m_i - q)^2 - 4(m_i)^2(q-1)^2 q}}{2(q-1)^2 q)} \\ \bar{r}_1 = \bar{r}_{\neg 1} &= \frac{2q}{q - m_i + 2m_i q + \sqrt{(m_i - q)^2 - 4(m_i)^2(q-1)^2 q}} \end{aligned} \tag{20}$$

Figure 11. Symmetric equilibrium values for a system of q pools of equal sizes.

often publish this data to demonstrate their honesty to their miners [28], [29], [30]. Second, a pool can infiltrate each of the other pools with some nominal probing mining power and measure the revenue density directly by monitoring the probe's rewards from the pool.

C. Block Withholding Recycling

We assume that the infiltrating miners are loyal to the attacker. However, some of the pool's members may be disloyal infiltrators. When sending disloyal miners to perform block withholding at other pools, an attacker takes a significant risk.

For example, pool 1 can use a loyal miner w to infiltrate pool 2, and pool 2, thinking the miner is loyal to it, might use it to attack pool 1. The miner m can perform honest mining for pool 1, rather than withhold its blocks, and not return any revenue to pool 2. Moreover, it will take its share of pool 2's revenues (which thinks the miner is loyal to it) and deliver it back to pool 1.

To avoid such a risk, a pool needs a sufficient number of verified miners — miners that it knows to be loyal. In general, the optimal infiltration rate may be as high as 60% of the pool size, but this is only in extreme cases when pools are large. For practical pool sizes, as we saw, a pool may need up to 25% of its mining power for infiltration. In Bitcoin, pools typically have loyal mining power — either run directly by the pool owners or sold as a service but run on the pool owners' hardware [31], [32]. However the size of this mining power is considered a trade secret and is not published.

D. Countermeasures

As in the case of classical block withholding explained in Section II-C, a pool might detect that it is being attacked, but cannot detect which of its miners is the attacker. Therefore a pool cannot block or punish withholding miners.

Nevertheless, various techniques can be used to encourage miners to submit full blocks. A pool can pay a bonus for submitting a full proof of work. This would increase the revenue of the miner that found a block while reducing the revenue of the other miners from this block. While the average revenue of each miner would stay the same, small miners will suffer from higher variance in revenue. Another approach is to introduce a joining fee by paying new miners less for their work until they have established a reputation with the pool. Miners that seek flexibility may not accept this policy and choose another pool. Finally, the pool can use a honeypot trap by sending the miners tasks which it knows will result in a full proof of work [10]. If a miner fails to submit the full proof of work it is tagged as an attacker. To prevent the attacker from learning them, the honeypot tasks have to be regularly refreshed, consuming considerable resources. Pools can also incorporate out of band mechanisms to deter attacks, such as verifying the identity of miners or using trusted computing technologies [33] that assure no block withholding is taking place. This would require miners to use specialized hardware and software, an overhead miners may not accept.

In summary, there is no known silver bullet; all these techniques reduce the pool's attractiveness and deter miners.

E. Block Withholding in Practice

Long term block withholding attacks are difficult to hide, since miners using an attacked pool would notice the reduced revenue density. Nevertheless, such attacks are rarely reported, and we can therefore conclude that they are indeed rare. A recent exception is an attack on the Eligius pool performed in May and June 2014 [34]. The pool lost 300 Bitcoin before detecting the attack, at which point payouts to the attackers were blocked. The attackers continued the attack, accumulating 200 more Bitcoin before realizing they were not receiving their payout.

The reasons the attack was so easily subverted is the limited efforts of the attackers to hide themselves. They have only used two payout addresses to collect their payouts, and so it was possible for the alert pool manager to cluster the attacking miners and obtain a statistically significant proof of their wrongdoing.

It is unknown whether this was a classical block withholding attack, with the goal of sabotage, or a more elaborate scheme. To verify the effectiveness of block withholding for profit, Luu et al. [35] implemented an experimental Bitcoin test network and demonstrated the practicality of the attack.

IX. Discussion

A. Bitcoin's Health

Large pools hinder Bitcoin's distributed nature as they put a lot of mining power in the hands of a few pool managers. This has been mostly addressed by community pressure on miners to avoid forming large pools [21]. However such recommendations had only had limited success, and mining is still dominated by a small number of large pools. As a characteristic example, in the period of November 2–8, 2014, three pools generated over 50% of the proofs of work [36].

The fact that block withholding attacks are rarely observed may indicate that the active pools have reached an implicit or explicit agreement not to attack one another. However, an attacked pool cannot detect which of its miners are attacking it, let alone which pool controls the miners. At some point a pool might miscalculate and decide to try and increase its revenue. One pool might be enough to break the agreement, possibly leading to a constant rate of attacks among pools and a reduced revenue.

If open pools reach a state where their revenue density is reduced due to attacks, miners will leave them in favor of other available options: Miners of sufficient size can mine solo; smaller miners can form private pools with closed access, limited to trusted participants.

Such a change may be in favor of Bitcoin as a whole. Since they require such intimate trust, private pools are likely to be smaller, and form a fine grained distribution of mining power with many small pools and solo miners.

B. Miners and Pools

1) Direct Pool Competition: A pool may engage in an attack against another pool not to increase its absolute revenue, but rather to attract miners by temporarily increasing its revenue relative to a competing pool.

Recent work has investigated the motivation of pools to utilize part of their resources towards sabotage attacks against each other [37], [38]. The model of those works is different from the pool game model in two major ways — a sabotage attack does not transfer revenue from victim to attacker, and migrating miners switch to less attacked pools, changing pool sizes and hence revenues until convergence. The model is parametrized by the cost of the attack and by the mobility of the miners, and the analysis demonstrates that when considering only sabotage attacks there are regions where no-attack is the best strategy. The miner's dilemma is therefore not manifested in that model.

Pool competition for miners is an incentive in and of its own for mutual attacks, and a pool may therefore choose to perform block withholding even if its revenue would increase only after the next difficult adjustment. The two models are therefore complimentary; the analysis of their combination is left for future work.

2) Pool Fees: We assumed in our analysis that pools do not charge fees from their members since such fees are typically nominal (0 – 3% of a pool's revenue [39]). The model can be extended to include pools fees. Fees would add a friction element to the flow of revenue among infiltrated and infiltrating pools. Specifically, Equation 4 would change to take into account a pool fee of f

$$r_i(t) = \frac{R_i(t) + \sum_{j=1}^{p} x_{i,j}(t)(1-f)r_j(t)}{m_i + \sum_{j=1}^{p} x_{j,i}(t)} \ . \tag{21}$$

A pool with a fee of f is a less attractive target for block withholding, since the attacker's revenue is reduced by f. However it is also less attractive for miners in general. Trading off the two for best protection is left for future work, as part of the treatment of the miner-pool interplay.

X. Related Work

A. The Block Withholding Attack

The danger of a block withholding attack is as old as Bitcoin pools. The attack was described by Rosenfeld [10] as early as 2011, as pools were becoming a dominant player in the Bitcoin world. The paper described the standard attack, used by a miner to sabotage a pool at the cost of reducing its own revenue. A more general view of fairness in proof of work schemes was discussed in 2002 by Adam Back [40] in the context of the HashCash system [41]. Early work did not address the possibility of pools infiltrating other pools for block withholding.

In concurrent work, Luu et al. [35] experimentally demonstrate that block withholding can increase the attacker's revenue. They do not address the question of mutual attacks. Courtois and Bahack [42] have recently noted that a pool can increase its overall revenue with block withholding if all other mining is performed by honest pools. We consider the general case where not all mining is performed through public pools, and analyze situations where pools can attack one another. The discrepancy between the calculations of [42] and our results for the special case analyzed there can be explained by the strong approximations in that work. For example, we calculate exactly how infiltrating miners reduce the revenue density of the infiltrated pool.

B. Temporary Block Withholding

In the Block withholding attack discussed in this work the withheld blocks are never published. However, blocks can be withheld temporarily, not following the Bitcoin protocol, to improve an attacker's revenue.

A miner or a pool can perform a selfish mining attack [23]. With selfish mining the attacker increases its revenue by temporarily withholding its blocks and publishing them in response to block publication by other pools and miners.

This attack is independent of the block withholding attack we discuss here and the two can be performed in concert.

An attacker can also perform a double spending attack as follows [10]. He intentionally generates two conflicting transactions, places one in a block it withholds, and publishes the other transaction. After the recipient sees the published transaction, the attacker publishes the withheld block to revoke the former transaction. This attack is performed by miners or pools against service providers that accept Bitcoin, and it not directly related to this work.

C. Block Withholding Defense

Most crypto-currencies use a proof-of-work architecture similar to Bitcoin, where finding proof of work is the result of solution guessing and checking. All of the algorithms we are aware of are susceptible to the block withholding attack, as in all of them the miner can check whether she found a full or a partial proof of work. Prominent examples are Litecoin [5], Dogecoin [6] and Permacoin [7].

It is possible to use an alternative proof of work mechanism in which miners would not be able to distinguish partial from full proofs of work [40], [10], [43]. Such a solution could reduce or remove the danger of block withholding. However, making such a change may not be in the interest of the community: Pool block withholding, or even its potential, could lead to a reduction of pool sizes, as explained in Section IX-A.

D. Decentralized Pools

Although most pools use a centralized manager, a prominent exception is P2Pool – a distributed pool architecture with no central manager [19]. But the question of whether a pool is run by a centralized manager or with a decentralized architecture is almost immaterial for the attack we describe. An open P2Pool group can be infiltrated and attacked, and the P2Pool code can be changed to support attacks against other pools.

On the other hand, P2Pool can be used by groups of miners to easily form closed pools. These do not accept untrusted miners, and are therefore protected against block withholding.

XI. Conclusion

We explored a block withholding attack among Bitcoin mining pools — an attack that is possible in any similar system that rewards for proof of work. Such systems are gaining popularity, running most digital currencies and related services.

We observe that no-pool-attacks is not a Nash equilibrium: If none of the other pools attack, a pool can increase its revenue by attacking the others.

When two pools can attack each other, they face a version of the Prisoner's Dilemma. If one pool chooses to attack, the victim's revenue is reduced, and it can retaliate by attacking and increase its revenue. However, when both attack, at Nash equilibrium both earn less than they would have if neither attacked. With multiple pools of equal size a similar situation arises with a symmetric equilibrium.

The fact that block withholding is not common may be explained by modeling the attack decisions as an iterative prisoner's dilemma. However, we argue that the situation is unstable since the attack can be done anonymously. Eventually, one pool may decide to increase its revenue and drag the others to attack as well, ending with a reduced revenue for all. The inferior revenue would push miners to join private pools, which can verify that their registered miners do not withhold blocks. This would lead to smaller pools, and so ultimately to a better environment for Bitcoin as a whole.

Acknowledgements: For their valuable advice, the author is grateful to Ken Birman, Fred B. Schneider, Emin Gün Sirer, Eva Tardos, and the paper shepherd Joseph Bonneau.

References

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

[2] S. Soper and O. Kharif, "EBay's PayPal unit to start accepting bitcoin payments," *Bloomberg*, September 2014.

[3] A. Chowdhry, "Google adds Bitcoin currency conversion to search," *Forbes*, July 2014.

[4] blockchain.info, "Bitcoin market capitalization," http://blockchain.info/charts/market-cap, retrieved Nov. 2014.

[5] Litecoin Project, "Litecoin, open source P2P digital currency," https://litecoin.org, retrieved Nov. 2014.

[6] Dogecoin Project, "Dogecoin," https://dogecoin.org, retrieved Nov. 2014.

[7] A. Miller, E. Shi, A. Juels, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2014.

[8] Namecoin Project, "Namecoin DNS – DotBIT project," https://dot-bit.org, retrieved Sep. 2013.

[9] V. Buterin, "A next generation smart contract & decentralized application platform," https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf/, retrieved Feb. 2015, 2013.

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

[11] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Research perspectives on bitcoin and second-generation cryptocurrencies," in *IEEE Symposium on Security and Privacy*. IEEE, 2015.

[12] Bitcoin community, "Protocol specification," https://en.bitcoin.it/wiki/Protocol_specification, retrieved Sep. 2013.

[13] Bitcoin community, "Protocol rules," https://en.bitcoin.it/wiki/Protocol_rules, retrieved Sep. 2013.

[14] Bitcoin community, "Bitcoin source," https://github.com/bitcoin/bitcoin, retrieved Sep. 2013.

[15] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *13th IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2013.

[16] M. B. Taylor, "Bitcoin and the age of bespoke silicon," in *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. IEEE Press, 2013.

[17] N. Popper, "Into the bitcoin mines," http://dealbook.nytimes.com/2013/12/21/into-the-bitcoin-mines/, 2013.

[18] E. Swanson, "Bitcoin mining calculator," http://www.alloscomp.com/bitcoin/calculator, retrieved Sep. 2013.

[19] forrestv, "p2pool: Decentralized, DoS-resistant, Hop-Proof pool," https://bitcointalk.org/index.php?topic=18313, 2013.

[20] The Bitcoin community, "Stratum mining protocol," https://en.bitcoin.it/wiki/Stratum_mining_protocol, retrieved Nov. 2014.

[21] G. Andresen, "Centralized mining," https://bitcoinfoundation.org/2014/06/centralized-mining/, retrieved Nov. 2014, June 2014.

[22] I. Eyal and E. G. Sirer, "How a mining monopoly can attack bitcoin," http://hackingdistributed.com/2014/06/16/how-a-mining-monopoly-can-attack-bitcoin/, 2014.

[23] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*, 2014.

[24] blockchain.info, "Bitcoin hashrate distribution," http://blockchain.info/pools, retrieved Jan. 2015.

[25] J. W. Friedman, "A non-cooperative equilibrium for supergames," *The Review of Economic Studies*, pp. 1–12, 1971.

[26] R. J. Aumann and L. S. Shapley, *Long-term competition — a game-theoretic analysis*. Springer, 1994.

[27] blockchain.info, "Bitcoin hash rate," https://blockchain.info/charts/hash-rate, retrieved Feb. 2015.

[28] slush, "System statistics," https://mining.bitcoin.cz/stats/, retrieved Nov. 2014.

[29] GHash.IO, "Ghash.io — bitcoin mining pool," http://organofcorti.blogspot.ca/2014/11/november-9th-2014-weekly-bitcoin.html, retrieved Nov. 2014.

[30] F2pool, "F2Pool," https://www.f2pool.com/, retrieved Nov. 2014.

[31] GHash.IO, "Ghash.io," https://ghash.io/, retrieved Nov. 2014.

[32] KnCMiner, "Kncminer bitcoin mining – cloud mining," https://www.kncminer.com/categories/cloud-mining, retrieved Feb. 2015.

[33] E. G. Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider, "Logical attestation: an authorization architecture for trustworthy computing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011.

[34] wizkid057, "Eligius," https://bitcointalk.org/index.php?topic=441465.msg7282674#msg7282674, retrieved Nov. 2014.

[35] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, "On power splitting games in distributed computation: The case of bitcoin pooled mining," Cryptology ePrint Archive, Report 2015/155, 2015, http://eprint.iacr.org/.

[36] Neighborhood Pool Watch, "November 9th 2014 weekly bitcoin network statistics," http://organofcorti.blogspot.ca/2014/11/november-9th-2014-weekly-bitcoin.html, retrieved Nov. 2014.

[37] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-theoretic analysis of ddos attacks against bitcoin mining pools," in *Workshop on Bitcoin Research*, 2014.

[38] A. Laszka, B. Johnson, and J. Grossklags, "When bitcoin mining pools run dry," in *Workshop on Bitcoin Research*, 2015.

[39] The Bitcoin community, "Comparison of mining pools," https://en.bitcoin.it/wiki/Comparison_of_mining_pools, retrieved Nov. 2014.

[40] A. Back, "Hashcash — amortizable publicly auditable cost-functions," http://hashcash.org/papers/amortizable.pdf, retrieved Jan. 2015, 2002.

[41] A. Back, "Hashcash — a denial of service counter-measure," http://www.cypherspace.org/hashcash/hashcash.pdf, retrieved Jan. 2015, 2002.

[42] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," *arXiv preprint arXiv:1402.1718*, 2014.

[43] I. Eyal and E. G. Sirer, "How to disincentivize large bitcoin mining pools," http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/, retrieved Jan. 2015.