Incentive Compatibility of Pay Per Last N Shares in Bitcoin Mining Pools

Yevhen Zolotavkin, Julian García[(✉)], and Carsten Rudolph

Faculty of IT, Monash University, Clayton, Australia
{yevhen.zolotavkin,julian.garcia,carsten.rudolph}@monash.edu

Abstract. Pay per last N shares (PPLNS) is a popular pool mining reward mechanism on a number of cryptocurrencies, including Bitcoin. In PPLNS pools, miners may stand to benefit by delaying reports of found shares. This attack may entail unfair or inefficient outcomes. We propose a simple but general game theoretical model of delays in PPLNS. We derive conditions for incentive compatible rewards, showing that the power of the most powerful miner determines whether incentives are compatible or not. An efficient algorithm to find Nash equilibria is put forward, and used to show how fairness and efficiency deteriorate with inside-pool inequality. In pools where all players have comparable computational power incentives to deviate from protocol are minor, but gains may be considerable in pools where miner's resources are unequal. We explore how our findings can be applied to ameliorate delay attacks by fitting real-world parameters to our model.

1 Introduction

Blockchain is a distributed ledger technology with demonstrated potential to revolutionize industry and commerce [10]. A number of popular cryptocurrencies based on blockchains have been launched in recent years to unprecedented adoption. These include Bitcoin (BTC) [11], Litecoin (LTC) and Zcash (ZEC) [13], among others [4]. The main technological innovation behind this drive is the proof-of-work consensus mechanism [7], which allows for the ledger integrity to be maintained in a distributed fashion. To achieve this level of decentralization, the system relies on miners who are incentivized to verify transactions. When incentives are compatible, rational players will find it in their best interest to stick to protocol. This paper uses game theory to derive conditions under which a popular mining reward mechanism, Pay per last N shares (PPLNS), is incentive compatible.

The Blockchain is a public ledger that keeps transaction information in a sequence of transaction blocks. Each block contains a hash of the previous block, and the chain grows as new transactions are verified and added to the chain. Any agent can add a block to the chain, so the approach relies on cryptographic puzzles, known as proofs of work, in order to reach consensus. The longest chain, as measured by computational effort exerted, is assumed to be the consensus chain. The agents solving the cryptographic puzzles are known as miners, and

they exchange their computational power for new currency and transaction fees. The puzzle is randomized in such a way that each miner has a probability of discovering the next block proportional to their share of computational power in the network [7].

The mining market is very competitive. Individual miners face large variances in income. Consequently, most miners pool their computational resources, sharing the rewards of the pool amongst all members in proportion to the computational effort invested in mining [9]. Through pooling, miners ensure a more stable income flow. Mining pools are managed by an administrator who will often collect fees from miners, distributing the rewards when blocks are discovered in the pool. Miners prove their work on behalf of the pool by discovering "shares", which are partial proofs of work. It is assumed that every share requires equal computational effort. In addition to satisfying the requirement for partial proof of work, every computed share may in addition qualify as full proof of work. In the latter case, the pool is rewarded by the Bitcoin network, which issues new coins and transfers them to the pool's account. The reward obtained by the pool is then distributed to the members of the pool, according to its reward scheme and the submission behaviour of all the pool members. Reward mechanisms serve to aggregate shares reported in the pool, so as to perform a fair distribution according to work.

Early reward mechanisms often rewarded miners in proportion to the amount of shares submitted by a miner in each round [14]. However, since the distribution of rewards is exponential, under this scheme, miners may increase their reward expectation by changing pools frequently. This attack is known as pool hopping, and discourages honest mining to unsustainable rates [3]. Pay per last N shares (PPLNS) addresses this issue.

In PPLNS, each miner gets a reward that is proportional to the effort exerted during the last shares preceding a submitted solution. Since solutions are not predictable, this reward scheme discourages hoppers who risk losing shares outside the range given by N. A simplified scheme of PPLNS is shown in Fig. 1.

In Fig. 1, time flows from left to right, so that the right-most share is the most recent. A discovered block is marked with a \$ sign, and not counted as a share in PPLNS. In this simple example, we consider only two miners forming the pool with power $\alpha_1 = 0.6$ and $\alpha_2 = 0.4$ for *Miner 1* and *Miner 2*, respectively. The length of the window N is 8 shares.

PPLNS is used by many Bitcoin pools, such as Kano [8], P2Pool [12], AntPool, BCMonster [2], among others. While this reward scheme is resilient to pool hopping attacks, other vulnerabilities are hypothesized to encourage dishonest mining [5]. In other words, the incentive compatibility of PPLNS is questionable [15].

We investigate a new type of attack for PPLNS pools. The idea is that miners can dishonestly increase their revenue by delaying reports of some of the shares that were obtained during a round. Instead of submitting share(s) to the pool manager when these are discovered, an attacker submits them at the end of the mining round, which will happen only if she finds a full solution. The
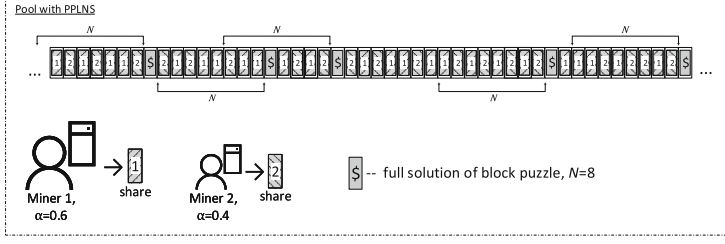
Fig. 1. Schematic explanation of mining in PPLNS pool with 2 miners.

purpose of this paper is to model the strategic incentives behind this kind of attack, as well as to estimate how damaging it can be to the pool. To do so, we formulate a simple game capturing the incentives of pool mining, and solve for Nash equilibria. A PPLNS scheme is incentive compatible if there are Nash equilibria in which miners do not delay their reports.

The rest of this paper is organized as follows: Sect. 2 contains detailed description of the attack and model that can be used to find equilibria. Conditions for incentive compatibility are discussed in Sect. 3, followed by Sect. 4, which addresses how severe attacks may be in pools that are not incentive compatible. We discuss our results and their implications in Sect. 5.

2   Model

Our model starts by computing the expected revenue of a pool member, given the pool composition, pool parameters as well as the rest of Bitcoin network. We consider the puzzle difficulty, D, to be pre-set at the network level. The PPLNS window size, N, is set by the pool manager. We also assume a given distribution of mining power τ_i for i in $1, \ldots, m$, where m is the size of the pool.

Each miner has two actions upon mining every single share: delay or report. For every miner i, we compute how the expected monetary reward changes given these options. The marginal profit for every share depends on the previous decisions made by the miner as well as the strategies of other miners in the pool.

For an attacking miner, there are two separate phases during every round. During the first phase, a miner collects shares for delay (does not report any single share). During the second phase, she reports every newly mined share immediately. For every miner, there is an individual turning point between these phases, which depends on the marginal profit of the two actions (delay or not). The turning point corresponds to the condition when the marginal profit for both actions is equal, or, when the strategy of the miner reaches its natural limit. The rationale behind these limits dictates that the number of delayed shares cannot be less than 0 and cannot exceed N. As soon as the individual turning condition is satisfied, the miner is in the second phase. In terms of time flow, equilibrium arises when every miner is beyond their turning point. Throughout the paper, we
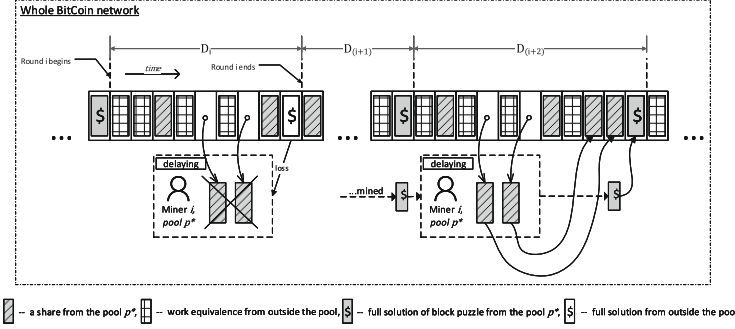
Fig. 2. Whole network schematic picture of a successful and an unsuccessful delay attack.

assume that rounds end some time after this turning point (the validity of this assumption is addressed in Sect. 4). For simplicity, we define an honest miner as one who is always in the second phase (delays 0). Likewise, attacking miners are those who delay at least 1 share in the first phase.

We also assume the following order for the submission of the delayed shares: if an attacker discovers a full solution of the Bitcoin puzzle, she reports all her delayed shares first, and reveals the full solution immediately after that. In our model, reporting shares collected during the first phase happens without time delays in revealing the full solution.

For an honest miner, the expected reward depends on N and D. Parameter D is the complexity of finding a full solution and can be expressed as the average number of shares that need to be mined to discover a full block. Every miner submits a share that he/she has mined and expects that a number of payments will be received for that share during the period in which the next N shares are sent by the pool members (a share will carry no value after this period). The expectation for that number of payments is $\frac{N}{D}$ and the value of a single payment is $Rew * \frac{1}{N}$, where Rew is a standard monetary reward for discovering a block. For simplicity, we omit the constant Rew. Therefore, every miner expects that every submitted share is worth $\frac{N}{ND}$.

These honest expectations for share payments change under delay attacks. A player j can delay an amount of $x_j \in \mathbb{N}$ shares. The effective window size is then $\hat{N}$ instead of N; and the effective expected number of shares submitted between two full solutions, found by the pool, is $\hat{D}$ instead of D. The reasons causing this are illustrated on Figs. 2 and 3. There are several immediate observations: (1) if an attacker is successful in finding a full solution she will report her delayed shares first; (2) due to delaying, the majority of the attackers will lose all the shares collected during the first phase.

Every reported share will be rewarded in a form of monetary payoff from the pool manager within the next $\hat{N}$ subsequent steps. Observation (1) above, implies
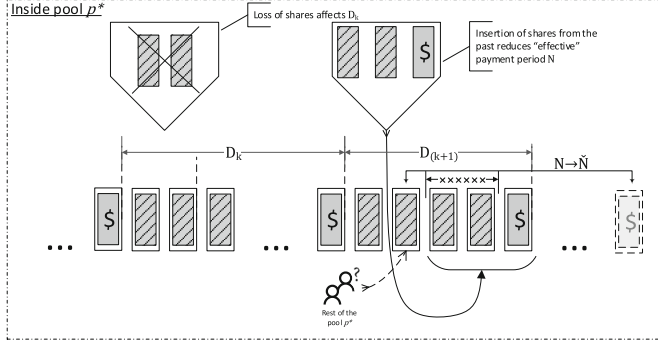
Fig. 3. Inside-pool schematic picture showing how D and N are affected by the delay attack.

that the expected number of steps when a potential reward can be received will be reduced (Fig. 3). This quantity can be computed as follows:

$$\hat{N} = \sum_{j=1}^{m} (N - x_j)\tau_j + (1 - p^*)N \leq N, \tag{1}$$

here, p^* is the probability that the solution is discovered by someone inside the miner's pool, i.e., $p^* = \sum_{j=1}^{m} x_j$. Expression (1) can be explained as follows: The first term, $\sum_{j=1}^{m}(N - x_j)\tau_j$, accounts for the probability τ_j, that miner j finds the full solution and will reduce the effective period for payment to $N - x_j$. The second term, $(1 - p^*)N$, accounts for the probability of finding the full solution outside the pool, $(1 - p^*)$. In this case, all the attacking miners lose their delayed shares and the effective period for payment is N.

Because the majority of the attackers will lose all the shares collected during the first phase, we can conclude that the amount of shares submitted between the nearest two full solutions is less than D. This is reflected in expression (2), which specifies the effective expected number $\hat{D}$ of shares submitted in the pool between the full solutions.

$$\hat{D} = \sum_{j=1}^{m}\left(x_j - \sum_{k=1}^{m} x_k + D\right)\tau_j + (1 - p^*)\left(D - \sum_{j=1}^{m} x_j\right) = D - \sum_{j=1}^{m} x_j + \sum_{j=1}^{m} x_j\tau_j. \tag{2}$$

Expression (2) can be explained as follows. Miner j will be able to publish her delayed shares with probability τ_j. In this case, all shares delayed by other attackers will be lost, and, expected number of shares (submitted in the pool since the last full solution was reported) is $(x_j - \sum_{k=1}^{m} x_k + D)$. Summing up such expectation for all the miners in the pool, we obtain $\sum_{j=1}^{m}(x_j - \sum_{k=1}^{m} x_k + D)\tau_j$. In addition, with probability $1 - p^*$ all delayed shares in the pool will be lost (because the full solution is found by miners outside the pool). This is expressed via term $(1 - p^*)(D - \sum_{j=1}^{m} x_j)$. From (1) and (2) it can be noted that when $x_j = 0, \forall j$, then $\hat{D} = D$ and $\hat{N} = N$.

Previously, it has been stated that if everybody in the pool is honest, the expected revenue from reporting a share is $\frac{N}{ND}$. In contrast, when delaying is possible any miner expects to be paid $\frac{\hat{N}}{N\hat{D}}$ by sending her share to the pool.

Nonetheless, for the share obtained during the first phase (and retained until the end of the mining round) the expectation of the revenue is different. A player j delaying $x_j - 1$ shares, expects the following reward from delaying one more share:

$$\frac{\tau_j}{N}\left(1 + \frac{N - x_j}{\hat{D}}\right).$$

This expression balances the expectation $\frac{\tau_j}{N}$ to be paid once for a share, when j finds a full solution (with probability τ_j). If that happens, she will also be paid $\frac{N-x_j}{\hat{D}}$ times in the subsequent rounds.

Now, we can sum up: some of the miners may never delay because it is not profitable for them to delay a single share; some can delay every mined share until they collect N; and, some will collect a number between 0 and N. Thus, a situation in which miners have no incentive to deviate is found by solving:

$$\frac{\hat{N}}{N\hat{D}} = \begin{cases} \frac{\tau_i}{N}\left(1 + \frac{N-x_i}{\hat{D}}\right), & \text{if } 0 \le x_i < N, \\[2ex] \frac{\tau_i}{N}\left(1 + \frac{N-x_i}{\hat{D}}\right) + C_i, & \text{if } x_i = 0, \qquad \forall i \, (C_i \ge 0). \\[2ex] \frac{\tau_i}{N}\left(1 + \frac{N-x_i}{\hat{D}}\right) - C_i, & \text{if } x_i = N, \end{cases} \tag{3}$$

This equation can be explained by the following constraints: *(i)* x_i cannot be negative – it is impossible to delay a negative number of shares; *(ii)* x_i cannot exceed N because under PPLNS, only the most recent N shares preceding the full solution (found by that pool) can be paid. The parameter C_i here compensates unequal profitability of delaying versus honest reporting. One can see that at $x_i = 0$, reporting may be more profitable for the i-th miner. On the other hand, at $x_i = N$, delaying can be more profitable than reporting.

The symbols listed in Table 1 will be used to define incentive compatibility and to estimate changes in parameters of PPLNS in case the pool is not incentive compatible (Sects. 3 and 4, respectively).

3   Incentive Compatibility

In this section, we will investigate a condition that guarantees honest mining. From Eq. (3), the only kind of incentive compatible equilibrium is described as $\frac{\hat{N}}{N\hat{D}} = \frac{\tau_i}{N}\left(1 + \frac{N-x_i}{\hat{D}}\right) + C_i, \forall i \, (x_i = 0, C_i \ge 0)$ which is equivalent to the following inequality:

$$\frac{\hat{N}}{N\hat{D}} \ge \frac{\tau_i}{N}\left(1 + \frac{N - x_i}{\hat{D}}\right), \forall i, x_i = 0. \tag{4}$$

Table 1. Notation and parameters

Notation	Meaning
p^*	Total mining power of the pool
α_i	Power of miner i relative to the mining power of the pool
τ_i	Absolute power of miner i, e.g., $\tau = \alpha_i p^*$
N	Window size, parameter of PPLNS
D	Complexity Bitcoin network expressed in (average) number of shares
$\hat{N}$	Expected number of steps when a reported share can be rewarded by the pool (case of more than 2 miners that may delay more than 1 share)
$\hat{D}$	Expected number of shares submitted into the pool during the period between reporting two consequent full solutions in the same pool (case of more than 2 miners that may delay more than 1 share)
x_j	Number of shares delayed by miner j
m	Number of miners in the pool
m'	Number of miners who delay shares in the pool

Inability to satisfy expression (4) for a single i, would mean that the pool will not mine honestly. For a pool of size m, there are $2^m - 1$ possible types of deviations from the mining protocol (each miner can either delay or always report). This yields a brute force search unfeasible for large values of m. Nonetheless, we will show that in order to verify incentive compatibility, we do not require exhaustive search. Instead, we derive a condition that can be checked in a linear time.

To derive conditions for incentive compatibility, it is useful to observe the following:

1. The set of all deviations needs to be reduced to a set $\mathcal{F}$, $|\mathcal{F}| \leq m$, of the deviations which (and only which) may produce an equilibrium (based on *Lemma 1*)
2. We show that if there is an incentive compatible equilibrium as described by (4), this equilibrium is unique (*Lemma 2*).
3. A single condition is sufficient and necessary to guarantee (4) (*Lemma 3*).

We start discussing cases that differ from (4). It will be demonstrated that there are only m other profiles that can be equilibria. We point to the fact that a delay attack requires that at least one miner delays a positive number of shares. Further, we show that an equilibrium where for a miner with power τ_i delays are only possible when all other miners with $\tau_k \geq \tau_i$ delay too.

Lemma 1. *If there is an equilibrium and a set $\mathcal{M}$ of delaying miners with power τ_i, $i \in \mathcal{M}$, delaying positive number of shares, then a miner with power τ_k is also delaying if $\exists k \notin \mathcal{M}, \tau_k \geq \tau_i$.*

(see Appendix for the proof).

As result, a miner with power τ_k should also be added to the set $\mathcal{M}$ of delaying miners. In the rest of the paper, we assume that miners are assigned indices according to their mining power sorted in descending order, e.g. $\tau_i \geq \tau_{i+1}$. This allows us to label an equilibrium compactly – specifying the index of the least powerful miner who can delay profitably. Since there are only m miners, we have at most m types of equilibria that differ from (4). The result from *Lemma 1*, showing that $x_i \geq x_{i+1}$ will be used in *Lemmas 2* and *3*.

Lemma 2. *The conditions that support incentive compatibility are inconsistent with any other kind of deviation represented by $\mathcal{F}$.*

For delaying miners included in set $\mathcal{M}$ information about other delaying miners may be incomplete. *Lemma 2* implies that: under certain conditions, a miner with power τ_i will delay a non-negative number of shares irrespectively of its inclusion in the set of delaying miners $\mathcal{M}$; expressions (8) and (9) (Appendix) can be used to calculate directly the number of shares delayed by miner i.

For incentive compatibility, it is necessary that for the most powerful miner (with power τ_1) the delay is not profitable. Using *Lemmas 1* and *2*, we will show that a sufficient and necessary condition for incentive compatibility can be expressed in terms of τ_1.

Lemma 3. *For incentive compatible mining under PPLNS it is sufficient and necessary that $\tau_1 \leq \frac{N}{N+D}$.*

In other words, an incentive compatible pool requires a bound on the computational power of the most powerful miner. This condition for honest mining is important, but even if pools are not incentive compatible the incentives to deviate may be small. The next section explores how these incentives change when we instantiate our model with realistic parameters.

4   Severity of Delay Attacks in the Real World

We propose an algorithm for equilibrium search, and this allows us to show how the parameters of the pool affect the likelihood of delaying attacks. The precondition for our algorithm is existence of equilibria.

To quantify the effect of incentive compatibility it is important to find equilibrium in the form of (3). The main obstacle here is that (3) represents a system of piece-wise expressions. For every single expression with index i, the choice of one out of three different domains affects all expressions in the system.

We use an iterative approach. Consider the schematic illustration on Fig. 4. Here, pool miners are classified into 3 classes ($x = \{0, (0, N), N\}$) according to the power they have. As it has been discussed previously in *Lemma 1*, miners with more power can profitably delay a greater number of shares, which cannot exceed the size of reward window N. Also, the number of shares cannot be negative. According to (3), to make C_i non-negative, for separate cases $x_i = 0$
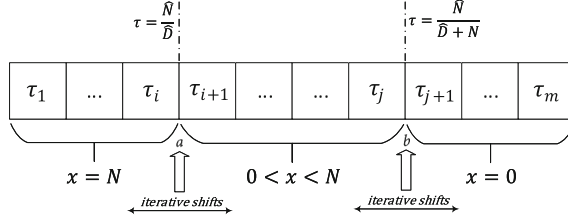
Fig. 4. Illustration of iterative algorithm for equilibrium search.

and $x_i = N$ the mining power should be $\tau_i \leq \frac{\hat{N}}{\hat{D}+N}$ and $\tau_i \geq \frac{\hat{N}}{\hat{D}}$, respectively. However, both $\hat{N}$ and, $\hat{D}$ depend on the selection of points a and b (see Fig. 4).
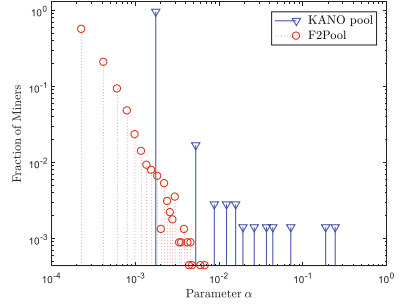
As soon as a, b are known, values of x for the domain $(0, N)$ can be calculated by solving a system of linear equations:

$$N - \sum_{j=1}^{m} x_j \tau_j = \tau_i \left(D - \sum_{j=1}^{m} x_j + \sum_{j=1}^{m} x_j \tau_j + N - x_i \right), \forall i, \tau_i \in (a, b),$$

where one should first substitute $x = 0$ and $x = N$ for corresponding indices.
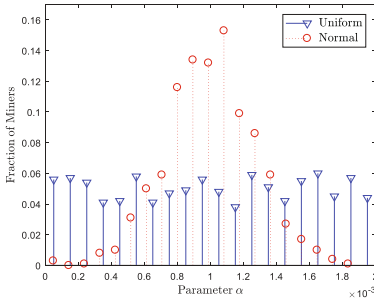
(a) Normalized distribution for uniform and normal ($\mu = 10^{-3}$, $\sigma = 10^{-3}$) cases.

(b) Normalized distribution for Kano pool and F2Pool.

Fig. 5. Distribution of mining power.

The size of the window (a, b) can potentially change from 0 to m. Therefore, the left endpoint a can be placed in any position between 1 and $m - l$, $l = $ length$[(a, b)]$. This requires $\sum_{l=0}^{m}(m - l)$ iterations with each requiring at most 2 computations (at the endpoints) to check validity of the assumption about a and b for that iteration. If the assumption is correct, the other $l - 2$ roots inside the window should be calculated. In terms of computation complexity, the whole procedure requires $O(m^2)$.

In our experiments, we used synthetic as well as real-world data for mining power distributions. In particular, we consider uniform and normal distributions. For real-world data, we collected distributions of mining power from Kanopool and F2pool (see Fig. 5).

In the first part of experiment, we compared the number of miners, who delay exactly N shares. In most cases of delay attacks it turns out that $a = b$ which means that miners are either delaying N shares or not delaying at all. The number of delaying miners is plotted for the left ordinate versus parameter k, where $N = kD$. In addition, the right ordinate scale was used to represent dependency of parameter $\frac{\hat{D}}{D}$ from k (Figs. 6 and 7).
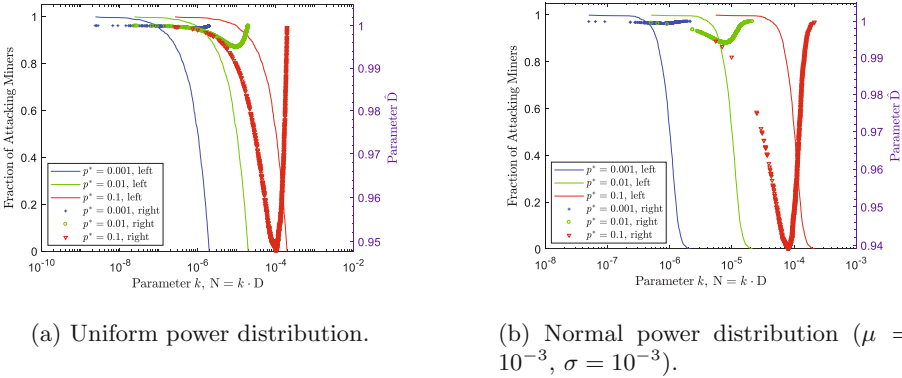

(a) Uniform power distribution.

(b) Normal power distribution ($\mu = 10^{-3}$, $\sigma = 10^{-3}$).

Fig. 6. Synthetic data. Fraction of attacking miners (left ordinate) and parameter $\hat{D}$ (right ordinate) for different k. Modelled for pool power p^* being 0.1%, 1% and 10% of the whole Bitcoin network, respectively. Equilibrium is symmetric, $\forall i \, (x_i \in \{0, N\})$.

Nonetheless, the question of cumulative extra profit (for the group of attackers) is, perhaps, the most important for honest miners. Because pool mining is a zero-sum game, extra profit for one group cause loses for another group of honest miners in that pool. There are several important differences with the concept of marginal profit for a share that has been used to find equilibrium [6]. In order to calculate cumulative extra profit one should consider: (a) extra profit is collected from those rounds where the full solution is submitted by honest miners of that pool; (b) an assumption about the duration of mining round is important and its validity is expressed with certain level of confidence (Fig. 8).

Extra profit is examined for the case when every attacker delays exactly N shares to the end of a round. Since extra profit is discussed in the context of successful solving of a puzzle by the pool, for each miner i we will refer to the power α_i in relation to the pool (not the whole Bitcoin network).

If one considers only the circumstances when attackers win a round, their expected profit is proportional to their power and is equal to what they can earn in fair mining. This is due to the fact that every miner submits N shares before
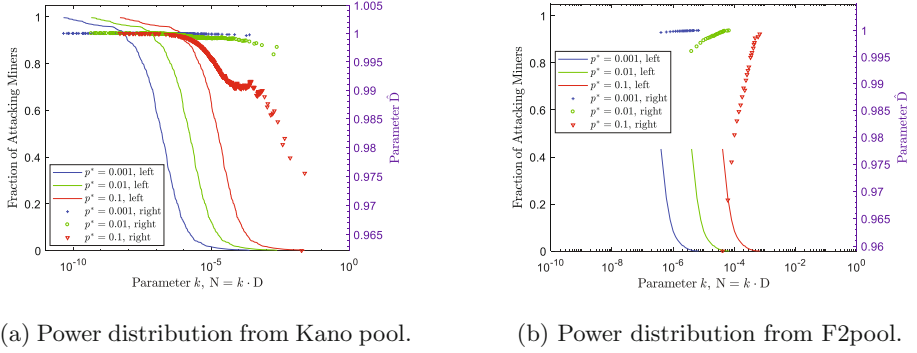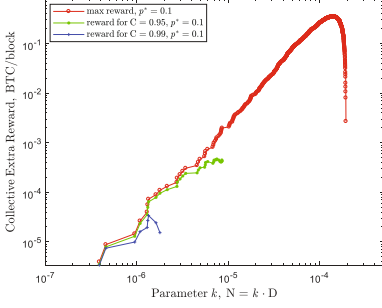
(a) Power distribution from Kano pool.   (b) Power distribution from F2pool.

Fig. 7. Real-world mining pools. Fraction of attacking miners (left ordinate) and parameter $\hat{D}$ (right ordinate) for different k. Modelled for pool power p^* being 0.1%, 1% and 10% of the whole Bitcoin network, respectively. Equilibrium is symmetric, $\forall i\, (x_i \in \{0, N\})$.

releasing a full solution. Such reward distribution is equivalent to solo mining when a miner collects all the revenue in the case of success.

However, if one considers circumstances when honest miners win, it is clear that each attacker collects a fraction of the reward which is proportional to her power in that pool. This can be seen as an additional profit (because they have already collected their fair portion). Such model of extra profit has one limitation: we assume that every attacker manages to collect her N shares (for the delay attack), and, after that, submits no less than $\alpha_i N$ shares to the pool. Therefore, a round should last the time which exceeds that estimation. For a subgroup of attackers, this happens with a probability determined by the least powerful miner in that subgroup (because collecting N shares for the attack takes her the most time). Hence, collective extra payment of any subgroup of attackers can be obtained with certain level of confidence.

It is assumed that a subgroup of attackers of size l includes all miners with power greater or equal than α_l (see *Lemma* 1 for support of this assumption). For every integer $l \in [1, m']$ (m' is the number of attackers in the pool) we will calculate: *(a)* collective extra profit E_l; *(b)* the conditional probability for a round to last longer than it takes for the l-th miner (time t_l) to mine $N + \alpha_l N$ shares, given that the round is won by that pool (i.e., probability $p(t_l|p^*)$). In Fig. 8, for every value of N we calculated maximum extra profit E_l where conditional probability $p(t_l|p^*)$ is greater than or equal to the corresponding confidence level C.

The subgroup of attackers exploits honest miners, who earn $Rew \sum_{m'+1}^{m} \tau_i$, where Rew is the current reward for discovering a full solution in the network (consisting of 12.5 BTC and transaction fees of up to 13.9 BTC on average). For the subgroup (size l) of attackers whose total power is $\sum_{1}^{l} \alpha_i$, the expected

(a) Uniform distribution of mining power.
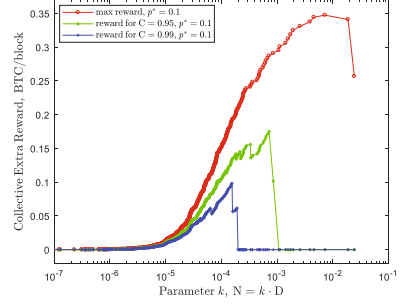
(b) Power distribution from Kano pool.

Fig. 8. Cumulative extra profit versus parameter k. Pool power is 10% of total network. Different colors represent profit for infinite length of mining round (max), for an average round with confidence levels 0.95 and 0.99, respectively. (Color figure online)

collective extra profit E_l is

$$E_l = Rew \left(\sum_{m'+1}^{m} \tau_i \times \sum_{1}^{l} \alpha_i \right).$$

The value of $p(t_l|p^*)$ is calculated as follows:

$$p(t_l|p^*) = 1 - \int_0^{t_l} f(t|p^*)dt,$$

where $f(t|p^*) = \frac{1}{Dp^*} e^{-\frac{t}{Dp^*}}$ is the conditional *pdf* for finding a full solution. The time t_l, necessary for l-th miner to collect $N + \alpha_l N$ shares is specified as $t_l = \frac{N+\alpha_l N}{\alpha_l}$. Hence, $p(t_l|p^*) = e^{-N\frac{1+\alpha_l}{\alpha_l Dp^*}}$, and, requiring that $p(t_l|p^*) \geq C$ we arrive to $\alpha_l (p^* \ln C + k) \leq -k$, $N = kD$. Considering that α_l is positive, there is an additional requirement $k < -p^* \ln C$ (it can be seen from Fig. 8 that blue and green plots are rising from zero level only for $k < 5 \times 10^{-3}$). If the latter is satisfied, we further require that $\alpha_l \geq -\frac{k}{p^* \ln C + k}$, $N = kD$.

For every k and corresponding C, we find l, such that $\max_{(\alpha_l \geq \frac{-k}{p^* \ln C + k})}[l] \leq m'$, and compute E_l (other attackers with indices $\leq l$ also pass the test and form the subgroup that has C-confident cumulative extra profit).

As one can see from the graphs, the extra profit of attackers can be quite substantial in terms of BTC. Remarkably, real-world power distributions (e.g., from Kano pool) lead to sufficiently higher levels of vulnerability to the attack, when compared with a benchmark uniform distribution of power.

5   Discussion

Incentive compatibility is an easily verifiable condition. It only requires information about the computational power of the most powerful miner in the pool. This verification can thus happen in linear time $O(m)$.

It should be stressed that known PPLNS pools comply with the requirement of incentive compatibility. For the existing majority of the pools, k varies between 1 and 5. Nonetheless, this parameter is under the sole control of the pool administrator who may decide to reduce it in order to satisfy requests from the majority of the miners.

Looking at pool miner forums, one can easily observe that a substantial number of miners would like to collect their payments faster. That aspect is especially important for pools that are not very large and infrequently discover complete solutions. Miners who join such pools during the winning round often find themselves in unfair and underpaid situations. The only way to satisfy their expectations fairly is reducing N, which increases the odds for delay attacks.

In pools that are not incentive compatible, our experimental results show that the fraction of delaying miners decreases with k, regardless of power distribution. Also, the shapes of the plots for the pools of different size (but same power distribution), e.g. 0.1%, 1%, 10% of total network power, are similar. However, a comparison between different pools reveals that for the same value of k, known real-world pools may have a higher proportion of attackers compared to artificially simulated data. This is due to the greater inequality in mining power distribution in real-world pools such as Kano. For instance, the most powerful member of a pool can sometimes account for up to a quarter of the pools total power. This may also be a significant obstacle in satisfying the condition for incentive compatibility, $\tau_1 \leq \frac{k}{k+1}$, in large pools with relatively small k.

Interestingly, $\hat{D}$ is non-monotonic on k. Obviously, $\hat{D}$ cannot be greater than D, however, the position of its minimum reflects differences in distribution of mining power in different pools. In addition, greater pool size (e.g. 10% vs 0.1% of network power) allows for attacks with greater k and that causes a greater decline in $\hat{D}$. The non-monotonic behaviour is due to the following property. For very small k, the changes in $\hat{D}$ (compared to D) are insignificant because the amount of shares that are delayed by every miner is negligibly small. For k close to the maximum, changes in $\hat{D}$ are also small due to the fact the number of attackers is small. Interestingly, the position of minimum in $\hat{D}$ for Kano pool (modelling 10% of network power) corresponds to the attack when only two most powerful miners delay. In contrast to that, for simulated data the same effect is achieved only when a majority of pool miners attack. Drops in the number of submitted shares (around 5% for large pools) can serve as a flag feature for pool administrators, who might detect the anomaly even before the attackers collect their first extra revenue.

Our plots for cumulative extra profit for a subgroup of attackers are also non-monotonic. That is because attackers exploit honest miners: when honest miners earn most the fraction of attackers is small; when fraction of attackers is

large, honest miners earn little. It should be noted that the red plot (for the both types of power distribution) stands for maximum collective revenue of attackers when the whole group of attackers can exploit honest miners. That may happen only if a round is unlimited in time. Comparing extra profit in real pools with synthetic data one can notice that for high confidence of estimation, uniform distribution produces insignificant incentives for dishonest miners (even though the pool is large, 10%). On the other hand, incentives for dishonest miners may be quite substantial (up to 0.17 BTC) for a pool with power a distribution that is like that of the Kano pool.

A PPLNS variant that is adopted in several large pools uses the concept of sharechain [12]. This assumes that every share is included in a simplified version of the main Blockchain, making delay attacks impossible by protocol. On the other hand, it may also cause a negative effect on honest miners. If for some reason (e.g., network latency) a share is out of sync, it is lost. Dead on Arrival rates can reach up to 15% of all submitted shares with this scheme. This is a disadvantage for miners whose network connection is unreliable. In that sense, traditional PPLNS has an advantage and is unlikely to be replaced in the near future. Hence, aspects of traditional PPLNS scheme should be analysed with greater attention. Our model shows, in summary, that equitable pools and smaller pools are more resilient. This in sharp contrast to the state of the Bitcoin network.

The analysis of incentive compatibility and related strategic models provide an opportunity to better understand reward functions in the Blockchain. The mechanism design of reward functions is a nascent and promising application of non-cooperative game theory. These models are also useful to evaluate implementation trade-offs. For example, the so-called Block Withholding Attack [1], may become less attractive for an attacker who can benefit from delaying. An adversary delaying shares until the end of the round would be unwilling to discard complete solutions. Also note, for example, that the average number of shares submitted per discovered block, $\hat{D}$, decreases with positive delays. This reduction may be significant from the perspective of computational and network load on pool administrators.

A   Appendix

A.1   Remarks

In the proofs, several aspects related to the concept of incentive compatibility are discussed. For that purpose, it is important to show that:

(1) for the current proofs, we will distinguish only two cases (instead of 3 in Eq. 3) $0 < x \leq N$ and $x = 0$. That can be explained by the fact that pool mining is either entirely honest or not (incentive compatibility questions only that aspect). The state of incentive compatibility when nobody delays can be derived from Eq. 3, $x_i = 0, \forall i$:

$$\frac{\hat{N}}{N\hat{D}} = \frac{\tau_i}{N}\left(1 + \frac{N - x_i}{\hat{D}}\right) + C_i, \ C_i \geq 0.$$

That is equivalent to

$$\hat{N} \geq \tau_i \left(\hat{D} + N - x_i \right),$$

or, this is equivalent to the requirement

$$\hat{N} = \tau_i \left(\hat{D} + N - x_i \right), \ x_i \leq 0, \ \forall i. \tag{5}$$

The latter notation will be used as it allows to analyse conditions for incentive compatibility using the roots of a system of linear equations.

(2) One should distinguish between two different situations: a miner i may have incentives to delay a positive number of shares even if $i \notin \mathcal{M}$; or, a miner i is included in $\mathcal{M}$ and definitely has an incentive to delay. It is assumed that miners in $\mathcal{M}$ do not have information about other delaying miners from outside $\mathcal{M}$. As a result of inclusion (or not inclusion) in the group of delaying miners $\mathcal{M}$, the incentive may be different. That is easy to see on the following example: the amounts of the shares delayed by miners in $\mathcal{M}$ depend on their information about $\mathcal{M}$, but, for i-th miner who is not in $\mathcal{M}$ the amount of delayed shares depends on the information about himself (τ_i) and the information about the number of shares that are delayed by miners in $\mathcal{M}$. However, in case i is the only miner in $\mathcal{M}$, e.g. $\mathcal{M} = \{i\}$ the incentive of the miner i is the same as if $\mathcal{M} = \varnothing$.

According to the definition, incentive compatibility is an equilibrium when $\mathcal{M} = \varnothing$ and nobody has an incentive to delay. Nonetheless, it is not clear if a pool with incentive compatible conditions can be in a state of another equilibrium when $\mathcal{M} \neq \varnothing, |M| > 1$. Information about $\mathcal{M}$ may be incomplete, and, answer to the question about other (delaying) equilibrium may require certain assumption about $\mathcal{M}$. In order to resolve that obstacle, we will produce some intermediate results in *Lemmas* 1 and 2.

A.2   Lemmas

Lemma 1. If there is an equilibrium and a set $\mathcal{M}$ of delaying miners τ_i, $i \in \mathcal{M}$, delaying positive number of shares, then miner with power τ_k is also delaying if $\exists k \notin \mathcal{M}, \ \tau_k \geq \tau_i$.

Proof. Let's assume that $l = \arg \min_{\mathcal{M}} \tau_i$. Considering ONLY delaying by miners in the system described by set $\mathcal{M}$, we rewrite (5) and express x_l as

$$x_l = \frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_l} + D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j. \tag{6}$$

Now, we investigate incentive of a miner with τ_k, $k \notin \mathcal{M}$, who has information about delaying miners from $\mathcal{M}$. As previously, we use (5), however, in that case additional components with index k is included:

$$x_k \tau_k = \sum_{j \in \mathcal{M}} x_j \tau_j + x_k \tau_k - N + \tau_k \left(D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j - x_k + x_k \tau_k \right),$$

$$x_k\left(1-\tau_k\right) = \frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_k} + D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j. \qquad (7)$$

Right hand sides of (6) and (7) are identical except of the difference in denominators of terms $\frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_l}$ and $\frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_k}$, respectively. Nominator $\sum_{j \in \mathcal{M}} x_j \tau_j - N$ is definitely negative. In the opposite case it would mean that at least one miner $g \in \mathcal{M}$, has incentive to delay $x_g > N$ shares. One can conclude this from the fact that $\sum_{j \in \mathcal{M}} \tau_j < p^* \le 1$. Delaying $x_g > N$ is clearly irrational because PPLNS reward scheme considers only the last N submitted shares.

Therefore, $\frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_k} \ge \frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_l}$ as long as $\tau_k \ge \tau_l$. Finally, we arrive to $x_k\left(1-\tau_k\right) \ge x_l$, and because $x_l,\left(1-\tau_k\right)$ are non-negative, x_k is non-negative. $\qquad \square$

Lemma 2: Conditions that support incentive compatibility are inconsistent with any other kind of deviation represented by $\mathcal{F}$.

Proof. We organize our proof in the following order. First, some $\mathcal{M}$, $|\mathcal{M}| = l$, is considered. That can be expanded by adding index $l + 1$ which represents a miner who can delay profitably. As a result, $\mathcal{M} \to \mathcal{M}'$, $|\mathcal{M}'| = l + 1$. Two cases of delay attack will be accounted for a miner with τ_{l+1}: attack with $\mathcal{M}$, attack with $\mathcal{M}'$. Expressions for the number of delayed shares ($x_{l+1}^{\mathcal{M}}$ and $x_{l+1}^{\mathcal{M}'}$, respectively) will be elaborated for the both cases. It will be demonstrated that if $x_{l+1}^{\mathcal{M}}$ is positive, then $x_{l+1}^{\mathcal{M}'}$ is positive too, and, vice versa.

Second, we are going show that by reducing $\mathcal{M}$ we will arrive to $\mathcal{M}^1$, $|\mathcal{M}^1| = 1$, containing only the most powerful miner of that pool with power τ_1. That would mean that a single deviation from incentive compatibility is profitable, which contradicts with the requirement for equilibrium. This conflicts with our assumption about incentive compatibility.

(1) Recalling (5) and (6) we can write

$$x_j \tau_j = \sum_{j \in \mathcal{M}} x_j \tau_j - N + \tau_j \left(D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j \right),$$

$$x_j = \frac{\sum_{j \in \mathcal{M}} x_j \tau_j - N}{\tau_j} + D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j.$$

There are l possible variants for the first and the second equation, respectively, where $j = 1, 2, \ldots, l$. Summing up all the l variations for each of the equations, one will obtain:

$$\sum_{j \in \mathcal{M}} x_j \tau_j = l \left(\sum_{j \in \mathcal{M}} x_j \tau_j - N \right) + \left(D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j \right) \sum_{j \in \mathcal{M}} \tau_j,$$

$$\sum_{j \in \mathcal{M}} x_j = \left(\sum_{j \in \mathcal{M}} x_j \tau_j - N \right) \sum_{j \in \mathcal{M}} \frac{1}{\tau_j} + l \left(D + N - \sum_{j \in \mathcal{M}} x_j + \sum_{j \in \mathcal{M}} x_j \tau_j \right),$$

respectively. For simplicity, we use the following substitutions: $X = \sum_{j \in \mathcal{M}} x_j \tau_j$, $Y = \sum_{j \in \mathcal{M}} x_j$, $\dot{p} = \sum_{j \in \mathcal{M}} \tau_j$, $S = \sum_{j \in \mathcal{M}} \frac{1}{\tau_j}$. Solving system

$$\begin{cases} X = l\,(X - N) + \dot{p}\,(D + N + X - Y) \\ Y = S\,(X - N) + l\,(D + N + X - Y) \end{cases},$$

in respect to X and Y we will arrive to the answers $X = N + \frac{N(l+1-2\dot{p})-D\dot{p}}{l^2-1-\dot{p}(S-1)}$, $Y = 2N + D + \frac{N(2-l+S-2\dot{p})+D(1-l-\dot{p})}{l^2-1-\dot{p}(S-1)}$. The obtained results are for the system of configuration $\mathcal{M}$ and dimensionality l. In order to re-calculate X, Y for configuration $\mathcal{M}'$ (dimensionality $l+1$) one would need to replace l with $l+1$, $\dot{p}$ with $\dot{p} + \tau_{l+1}$, S with $S + \frac{1}{\tau_{l+1}}$. For configuration $\mathcal{M}$ we express variable $x_{l+1}^{\mathcal{M}}$ (which is not yet included in the system) in terms of $X^{\mathcal{M}}, Y^{\mathcal{M}}$ using (7):

$$\begin{aligned}
x_{l+1}^{\mathcal{M}}\,(1 - \tau_{l+1}) &= \frac{X^{\mathcal{M}} - N}{\tau_{l+1}} + D + N + X^{\mathcal{M}} - Y^{\mathcal{M}} \\
&= \frac{1}{\tau_{l+1}} \frac{N\,(l+1-2\dot{p}+\tau_{l+1}\,(2l-S-1)) - D\,(\dot{p}+\tau_{l+1}\,(1-l))}{l^2 - 1 - \dot{p}\,(S-1)}.
\end{aligned}$$

(8)

For configuration $\mathcal{M}'$ we express $x_{l+1}^{\mathcal{M}'}$ as an in terms of $X^{\mathcal{M}'}, Y^{\mathcal{M}'}$ using (6):

$$\begin{aligned}
x_{l+1}^{\mathcal{M}'} &= \frac{X^{\mathcal{M}'} - N}{\tau_{l+1}} + D + N + X^{\mathcal{M}'} - Y^{\mathcal{M}'} \\
&= \frac{1}{\tau_{l+1}} \frac{N\,(l+1-2\dot{p}+\tau_{l+1}\,(2l-S-1)) - D\,(\dot{p}+\tau_{l+1}\,(1-l))}{(l+1)^2 - 1 - (\dot{p}+\tau_{l+1})\left(S + \frac{1}{\tau_{l+1}} - 1\right)}.
\end{aligned}$$

(9)

Now, we are going to compare right-hand sides of Eq. (8) and (9). In the both cases nominators $N\,(l+1-2\dot{p}+\tau_{l+1}\,(2l-S-1)) - D\,(\dot{p}+\tau_{l+1}\,(1-l))$ are identical. Our task is to prove that denominators in (8) and (9) $l^2 - 1 - \dot{p}\,(S-1)$ and $(l+1)^2 - 1 - (\dot{p}+\tau_{l+1})\left(S + \frac{1}{\tau_{l+1}} - 1\right)$, respectively, are of the same sign.

We show that expression $l^2 - 1 - \dot{p}\,(S-1) = l^2 - \dot{p}S - (1-\dot{p})$ is negative. Clearly, $-(1-p)$ is negative. Further, it will be proven that $l^2 - \dot{p}S \le 0$. That expression can be represented as $l^2 - \sum_{j=1}^{l} \tau_j \times \sum_{j=1}^{l} \frac{1}{\tau_j}$. Component $\sum_{i=1}^{l}\sum_{j=1}^{l} \frac{\tau_i}{\tau_j}$ has l^2 terms. Exactly l out of l^2 terms are $\frac{\tau_j}{\tau_j} = 1$. Among the rest $l^2 - l$ (this number is obviously even for any natural l) terms, there are $\frac{l^2-l}{2}$ pairs $\left(\frac{\tau_i}{\tau_j}, \frac{\tau_j}{\tau_i}\right)$, $i \ne j$. We conclude that $\frac{\tau_i}{\tau_j} + \frac{\tau_j}{\tau_i} = \frac{\tau_i^2 + \tau_j^2}{\tau_i \tau_j} \ge 2$ because $(\tau_i - \tau_j)^2 \ge 0$.

Denominator $(l+1)^2 - 1 - (\dot{p}+\tau_{l+1})\left(S + \frac{1}{\tau_{l+1}} - 1\right)$ from (9) is obtained from $l^2 - 1 - \dot{p}\,(S-1)$ by substituting l with $l+1$, $\dot{p}$ with $\dot{p} + \tau_{l+1}$, S with $S + \frac{1}{\tau_{l+1}}$. Therefore, its sign is identical to $l^2 - 1 - \dot{p}\,(S-1)$ from (8) because in the proof we generalized values for $l, \dot{p}, S$. Hence, the both of $x_{l+1}^{\mathcal{M}}$ and $x_{l+1}^{\mathcal{M}'}$ are the numbers of the same sign.

(2) Further, the following technique will be used. Posit that the same conditions that provide incentive compatibility may be exploited by a set of miners $\mathcal{M}$, $|\mathcal{M}| = l$, to delay profitably. Also, let us assume another case of a set $\mathcal{M}^{l-1}$, $|\mathcal{M}^{l-1}| = l - 1$, and a miner with power τ_l who has information about $\mathcal{M}^{l-1}$. In those two cases, miner with power τ_l delays profitably according to the proof provided above. For the latter case, the configuration for delaying equilibrium can be represented as $\{\mathcal{M}^{l-1}, l\}$. According to the results from *Lemma* 1, miner $(l - 1) \in \mathcal{M}^{l-1}$ also delays profitably. Therefore, we may consider another possible configuration $\{\mathcal{M}^{l-2}, l - 1\}$ for whom delaying is definitely profitable. Finally, we may arrive to the configuration $\{\mathcal{M}^1, 2\}$ where $\mathcal{M}^1$ contains only 1-st miner with power τ_1, who can delay profitably. In such case he has an incentive to deviate from honest mining even though the information about actions of others is not taken into account. That clearly contradicts with the assumption that incentive compatibility is an equilibrium.   □

Lemma 3: For incentive compatible mining under PPLNS it is sufficient and necessary that $\tau_1 \leq \frac{N}{N+D}$.

Proof. Condition $\tau_1 \leq \frac{N}{N+D}$ can be derived from the requirement $\hat{N} \geq \tau_1 \left(\hat{D} + N - x_1 \right)$, $x_1 = 0$, for special case when $\mathcal{M} = \varnothing$ meaning that for the most powerful miner it is not profitable to delay. From the second part of *Lemma* 2 it is easy to see why such condition is necessary for incentive compatibility. In addition, it will be illustrated that it is sufficient. We consider $\mathcal{M}^1$ which includes only the 1-st miner. According to *Lemma* 1, the number of delayed shares for the second powerful miner with power of τ_2 (who is not yet included in $\mathcal{M}^1$) is not positive either, $x_2^{\mathcal{M}^1} (1 - \tau_2) \leq x_1^{\mathcal{M}^1} \leq 0$. If we consider $\mathcal{M}^2$ that includes the 1-st and 2- miners, according to *Lemma* 2, sign of x_2 does not change. Hence, neither further expansion of $\mathcal{M}$ nor considering delay from miners that are not included in $\mathcal{M}$ can produce roots that are entirely positive. This means that no delaying configuration can be in a state of equilibrium.   □

References

1. Bag, S., Ruj, S., Sakurai, K.: Bitcoin block withholding attack: analysis and mitigation. IEEE Trans. Inf. Forensics Secur. **12**(8), 1967–1978 (2017)
2. BCmonster: Mining statistics (2017). http://www.bcmonster.com/. Accessed 22 Mar 2017
3. Chávez, J.J.G., da Silva Rodrigues, C.K.: Automatic hopping among pools and distributed applications in the bitcoin network. In: 2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA), pp. 1–7, August 2016
4. Dziembowski, S.: Introduction to cryptocurrencies. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015, pp. 1700–1701, NY, USA (2015). http://doi.acm.org/10.1145/2810103.2812704
5. Fisch, B.A., Pass, R., Shelat, A.: Socially optimal mining pools. ArXiv e-prints March 2017

6. Fudenberg, D., Tirole, J.: Game Theory, 11th edn. The MIT Press, Cambridge (1991)
7. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 3–16, NY, USA (2016). http://doi.acm.org/10.1145/2976749.2978341
8. Kano pool: Pool payout (2017). https://kano.is/index.php?k=payout. Accessed Mar 23 2017
9. Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., Rosenschein, J.S.: Bitcoin mining pools: A cooperative game theoretic analysis. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, pp. 919–927, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2015). http://dl.acm.org/citation.cfm?id=2772879.2773270
10. Morabito, V.: Business Innovation Through Blockchain, vol. 1. Springer International Publishing AG, Heidelberg (2017)
11. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf. Accessed 29 Jan 2016
12. P2Pool: P2Pool bitcoin mining pool global statistics (2017). http://p2pool.org/stats/index.php. Accessed 19 Mar 2017
13. Peck, M.: A blockchain currency that beat s bitcoin on privacy [news]. IEEE Spectr. **53**(12), 11–13 (2016)
14. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint (2011). arXiv:1112.4980
15. Schrijvers, O., Bonneau, J., Boneh, D., Roughgarden, T.: Incentive compatibility of bitcoin mining pool reward functions. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 477–498. Springer, Heidelberg (2017). doi:10.1007/978-3-662-54970-4_28