Incentive Compatibility of Bitcoin Mining Pool Reward Functions

Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden

Stanford University

Abstract. In this paper we introduce a game-theoretic model for reward functions in Bitcoin mining pools. Our model consists only of an unordered history of reported shares and gives participating miners the strategy choices of either reporting or delaying when they discover a share or full solution. We defined a precise condition for incentive compatibility to ensure miners strategy choices optimize the welfare of the pool as a whole. With this definition we show that proportional mining rewards are not incentive compatible in this model. We introduce and analyze a novel reward function which is incentive compatible in this model. Finally we show that the popular reward function pay-per-last-N-shares is also incentive compatible in a more general model.

1 Introduction

By almost any measure, Bitcoin [1] has become the most successful cryptocurrency in history. While Bitcoin has evolved into a very complex sociotechnical system which we will not describe in detail here,[1] at its core lies a decentralized consensus protocol allowing all participants to agree on a common global ledger of transactions to prevent double-spends and other disallowed behavior. The key to Bitcoin's consensus protocol (sometimes more broadly called *Nakamoto consensus* after its founder) is a group of entities called *miners* who race to solve a challenging cryptographic puzzle for the right to append a new block of transactions to Bitcoin's ledger, the *blockchain*. A system of incentives encourages these miners to follow the protocol faithfully in exchange for the ability to earn newly-minted coins and transaction fees in proportion to the amount of computational effort they have expended (also called hashing power or mining power).

Finding a single Bitcoin block is very rewarding (today worth at least Ƀ25, over US$6,000), yet it is also very difficult for smaller miners who might find a block on expectation only every few months or even every few years. As a result, the majority of mining power now consists of miners participate in *mining pools* in which they agree to divide rewards from blocks found by any member of the pool and thus receive a steadier stream of income. Choosing the exact algorithm used to divide up mining pool rewards (the *reward function*) however, turns out to be a challenging incentive design problem.

[1] For an academic overview of Bitcoin we refer the reader to [2].

Pools are sometimes controversial in the Bitcoin community as they represent a form of centralization. Miller et al. proposed a future cryptocurrency which attempts to prevent their formation [3]. As of today though they are an indispensable part of Bitcoin as well as many related cryptocurrencies (to which our work will also apply). Despite this, relatively little work has focused on the reward functions underlying pools since Rosenfeld's initial overview of the space [4]. Several papers have studied the interaction *between* pools and found that in some plausible circumstances pools will be incentivized to attack each other [5,6,7]. Yet the incentives underlying reward functions have not been rigorously studied.

In this paper we introduce a formal game-theoretical framework to study these reward functions. We are motivated by a very natural question: if individual miners are interested in maximizing their expected utility, is their behavior optimal for the pool as a group? For example, if miners are incentivized to delay reporting full solutions to the pool, this may lower the pool's overall rewards and even make it more vulnerable to external sabotage [5]. We introduce a simplified model with only a single mining pool and define basic properties that a good reward function should exhibit. Although our model is deliberately simplified, we still show somewhat surprisingly that some reward functions used in practice such as simple proportional payments are not incentive compatible.

We introduce a novel reward function which is incentive compatible within our model while still maintaining other desirable properties. Our reward function will remain incentive compatible even in a more complex informational model (although it may need to be extended if the definition of incentive compatibility is extended to include more complicated attacks).

While our model cannot capture all reward functions used in practice, we consider it an important milestone in analyzing mining pools in Bitcoin and related systems. We further take the first step to analyzing reward functions in more general informational models by carefully examining the popular pay-per-last-N-shares reward function, and show that it is incentive compatible. This indicates that our approach is not limited to the informational assumptions, but can be more generally applied.

2 Preliminaries

In this paper we look at a simple model in which miners are bound to working for a particular pool and where their strategic choice is the following: if a miner finds a solution to the cryptographic puzzle, *when* does it report this to the pool. The pool is run by a *pool operator* and contains a fixed number n of miners. Each miner i has a fraction α_i of the total mining power. For most of this paper we will assume that $\sum_{i=1}^{n} \alpha_i = 1$, meaning that the pool has all the available mining power; there are no other pools or solo miners. In Appendix C we look at the case where the pools total hashing power $\alpha_P = \sum_{i=1}^{n} \alpha_i < 1$ and show that while this makes a quantitative difference, qualitatively our results carry over.

The time it takes for a miner to find a share is an exponentially distributed random variable with parameter α_i; hence in expectation it takes time $1/\alpha_i$ to find a share. Each share is also a full solution with probability $1/D$.

2.1 Reward Functions and History Transcripts

Miners report their shares and solutions to the pool operator. When a solution is reported, the operator who collects the block reward from the Bitcoin network and subsequently divides the reward among the n miners according to a *reward function* R. The game then restarts. For the mathematical model we assume no variability in the block reward or the transaction fees, although the work can be extended to include this.

The reward function is the only way in which the miners receive any payout and therefore the reward function completely drives the behavior of miners. A perfectly equitable reward function would simply give each miner i a fraction $\frac{\alpha_i}{\alpha_P}$ of the reward in proportion to the fraction of the pool's total mining power to which that miner contributed.

However, the pool operator does not know the actual α_i of each miner. The challenge in designing a reward function R stems from the necessity of estimating this based on reported shares and solutions. The operator's ability to estimate α_i depends on the precise information it has access to. We model this as a *history transcript* $\mathcal{H}$. A reward function $R : \mathcal{H} \to [0,1]^n$ is a function from a history transcript to an allocation $\{a_i\}_{i=1}^n$ with $\sum_i a_i = 1$. We use $R_i : \mathcal{H} \to [0,1]$ to denote the function that yields the i^{th} component of R.

In most of this paper we analyze the case of an unordered history transcript:[2] $\mathcal{H}$ contains for each miner i the total number of shares $b_i \in \mathbb{N}$ that have been reported in that round.[3] Thus, the history transcript is given by a vector $\mathbf{b} \in \mathbb{N}^n$ that contains for each of the n players the total number of shares that she found during the round (where the full solution is also counted as a share). We use vector notation for $\mathbf{b}$, so $\mathbf{b}_1 + \mathbf{b}_2$ means the component-wise addition of $\mathbf{b}_1$ and $\mathbf{b}_2$, and $||\mathbf{b}||_1 = \sum_{i=1}^n b_i$ is the sum of the components of $\mathbf{b}$.

This model is perhaps the simplest possible[4] which enables a mining pool to function, yet it captures several basic reward function schemes used in practice. There are also reward functions which require additional information, such as the order in which shares were reported or reports from previous rounds of the game. In Section 8 we briefly discuss how to generalize this model and the challenges with characterizing incentive compatibility for them. However, we stress that positive results demonstrated incentive compatibility in our simple model extend to any more complicated model, as the pool operator can always decline to use additional information in its reward function.

[2] In Section 6 we consider a strictly more general informational model, which will be described there.

[3] We adopt the convention that $\mathbb{N}$ includes the number 0.

[4] A simpler format such as only receiving information about which miner reported a full solution would only allow a replication of solo mining.

2.2 Miner Strategy

Now that we defined a model and the reward function R, let's look at how the choice of R impacts the behavior of miners. The goal of any mining pool is to earn as many rewards as possible for its members.[5] If miners delay in reporting blocks to the pool, this imposes a risk that an external pool may find the block first, undermining the pool's potential rewards.[6] Note that while we are only modeling a single pool, we build in the assumption that this pool wants to report solutions as fast as possible to the wider network to avoid getting scooped by the competition. Thus we will want our reward function to ensure solutions to be reported and processed as soon as they are found.[7]

In response to a reward rule R, miners choose a strategy $\sigma(R)$ which dictates what a miner does when it finds a share or full solution. Ideally the strategy $\sigma(R)$ is to report any share or solution immediately. However, the pool operator cannot directly tell miners what to do; rather they should choose an R such that the miners corresponding strategy $\sigma(R)$ is to immediately report. Formally, let t be the time since miner i started mining, let T be the number of rounds that have been completed at time t, and let $\mathbf{b}_j$ be the number of shares per player in round j. Miner i is interested in maximizing their throughput:

$$\sigma(R) := \max_{\sigma} \lim_{t\to\infty} \frac{\sum_{j=1}^{T} R_i(\mathbf{b}_j)}{t}. \tag{1}$$

Here σ impacts the number T of rounds that were completed, as well as the number of reported shares $\mathbf{b}_j$ in each round.

2.3 Reward Function Desiderata

We define three properties which are important for a reward function. The first is a formalization of the intuition above:

Property 1 (Incentive Compatibility). A reward function R is *incentive compatible* when every miner's best response strategy $\sigma(R)$ reports full solutions immediately.

In Section 3 we give a mathematical condition that characterizes Property 1, and that can easily be verified for reward functions.

---

[5] In this work we are only considering a pool which follows the default mining strategy and does not attempt to implement an deviant strategies to earn disproportionately more rewards than competing pools, such as temporary block withholding [8].

[6] Another way of saying this is that a reward function which does not compel participants to report solutions immediately is *not* welfare maximizing, since the selfish behavior of individuals can hurt the total reward of the group.

[7] While we do not consider fees in this paper, note that a pool operator would also want to optimize throughput if collects a fraction of the reward.

Next, we require that the pool pays miners in proportion to the amount of work they have performed. Miners form pools to reduce the variance in revenue. In practice they might accept losing a small fraction f of their expected value in fees, but we would like miner performing an α_i fraction of the work to receive an α_i fraction of the reward.

Property 2 (Proportional Payments). A reward function R provides *proportional payments* whenever for each miner i

$$\mathbb{E}_{\mathbf{b}}[R_i(\mathbf{b})] = \alpha_i.$$

Finally, we would like the pool operator to never incur a deficit. That is, the reward function R should precisely divide the reward among the n miners at the end of a round. If this is not the case, then either miners may leave some value on the table, or the pool operator may be liable for more then she received herself. This latter condition is particularly dangerous, as it leaves the pool exposed to sabotage attacks [5] in which competing miners purposely withhold full solutions to damage the pool.

Property 3 (Budget Balanced). A reward function R is (γ, δ)-budget balanced when for all $\mathbf{b}$:

$$\gamma \leq \sum_{i=1}^{n} R_i(\mathbf{b}) \leq \delta.$$

In particular, an $(\gamma, 1)$-budget balanced reward function will never pay more to the miners than the pool operator received. Our goal will be $(1, 1)$-budget balanced reward functions which share the reward exactly among the n miners.

2.4 Common examples

Perhaps the most obvious reward function is the *proportional* reward function: $R_i(\mathbf{b}) = \mathbf{b}_i/K$, where $K = ||\mathbf{b}||_1 = \sum_{i=1}^{n} b_i$. That is, the reward is shared proportional to the number of shares each miner reported. We show that the proportional rule is not incentive compatible in Section 4.1.

Another reward function is the *pay-per-share* reward function: $R_i(\mathbf{b}, s) = b_i/D$, where participants are rewarded a fixed amount per share. In Section 4.2 we show that while this method is incentive compatible, it is not budget balanced (defined in Section 2.3), which means that the pool operator may be liable to pay out more to miners than she collects from the Bitcoin protocol.

2.5 Ensuring Steady Rewards

Miners are interested in maximizing the total reward they receive per time unit, but they join pools primarily to achieve a more consistent stream of revenue. Our goal will be to build a reward function which is as consistent as possible

which satisfies the three properties above. It is tempting to isolate one metric, such as the variance or standard deviation of the distribution of rewards, but we will discuss in Section 7 why these metrics are probably not the best measures of consistency in practice and provide different simulation results to compare reward functions.

3 Incentive Compatibility

We stated that for a reward function R to be incentive compatible, it needs to incentivize miners to report full solutions immediately. In this section we express that as a condition that can easily be checked for any given reward function.

We do this by looking at the strategic choice that a miner faces when she finds a full solution. Either she reports the full solution immediately, or she decides to delay reporting the solution until d more shares have been found. In this section we do not take into account the possibility of another miner finding and reporting a solution during this delay. In Appendix B we show that this is virtually without loss of generality.

Consider the situation when at time t, miner i finds a full solution. At this point $\mathbf{b}_t$ shares have been reported to the pool operator (for notational simplicity we assume that the full solution is already included in $\mathbf{b}_t$). The action space of miner i is $d \in \mathbb{N}$ (including 0) where the miner waits for d additional shares to be reported before reporting the full solution. If miner i decides to wait for d shares before reporting, her expected reward at the end of the delay is:

$$\mathbb{E}_{(\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d)}[R_i(\mathbf{b}_t + \mathbf{b})] = \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d} \Pr(\text{seeing } \mathbf{b}) \cdot R_i(\mathbf{b}_t + \mathbf{b})$$

On the other hand, if she decides to report the full solution immediately, she will receive the reward she was entitled to at that moment and a new round will start. So she will additionally get d times the expected reward per share. That is, delaying her report imposes an opportunity cost by not beginning the next round. So her expected reward in this situation after d more shares is:

$$\begin{aligned} R_i(\mathbf{b}_t) + d \cdot \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{\mathbb{E}_{\mathbf{b}}[||\mathbf{b}||_1]} &= R_i(\mathbf{b}) + d \cdot \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{\sum_{k=1}^{\infty} k \left(1 - \frac{1}{D}\right)^{k-1} \frac{1}{D}} \\ &= R_i(\mathbf{b}_t) + \frac{d}{D} \cdot \mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \end{aligned}$$

Reporting the solution immediately will be more profitable than delaying for d shares if and only if:

$$\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d} \Pr(\text{seeing } \mathbf{b}) \cdot \left(R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)\right) \;\leq\; \frac{d}{D} \cdot \mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \qquad (2)$$

The miner's best strategy is to report immediately if this condition holds for all $d \in \mathbb{N}\backslash\{0\}$. The following lemma states that there exists a $d \in \mathbb{N}\backslash\{0\}$ for which this condition holds if and only if it holds for $d = 1$. This is a very powerful statement: to determine the incentive compatibility of a reward function, we only need to see if it is profitable to delay reporting for a single additional share. In the following, let $\mathbf{e}_j$ be the j^{th} standard basis vector that is 0 everywhere except for the j^{th} component which is 1. Due to space limitations the proofs for all lemmas appear in Appendix A.

Lemma 1. *For a reward function R, a player i has an incentive to report full solutions immediately, iff the following condition holds for all $\{\alpha_i\}_{i=1}^n, \mathbf{b}_t, D, i$:*

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) \;\leq\; \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{D} \tag{3}$$

So to show that a reward function is incentive compatible, we need to show that condition (3) holds, and conversely when we show that for a reward function condition (3) is not guaranteed to hold, it cannot be incentive compatible.[8]

While for incentive compatibility we do not care about miners reporting shares immediately, this is important in ensuring proportional payments.

Lemma 2. *Miners report shares immediately if and only if the reward function R is monotonically increasing each component. That is: for all i, and $\mathbf{b}$:*

$$R_i(\mathbf{b} + \mathbf{e}_i) > R_i(\mathbf{b}).$$

4 Incentive Compatibility of Existing Methods

Now we will apply our characterization of incentive compatibility to reward functions which are in use today. In this section we restrict ourselves to reward functions that can be modeled by our definition of history transcript as described in Section 2.

4.1 Proportional Reward Function $R^{(\text{prop})}$

One of the earliest reward functions that is still in use is the proportional reward function. The idea is to divide the reward according to the proportion of shares of a miner compared to all shares that were reported to the pool:

$$R_i^{(\text{prop})}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}.$$

[8] In Appendix B we show that the possibility of another miner reporting a solution does not materially change the characterization here and in Appendix C we extend this to include the possibility of another pool reporting a full solution.

In expectation, the reward per share for each player is α_i/D. This approach is clearly proportional and budget-balanced. Previous work [4] has shown that in the presence of multiple pools, miners can be incentivized to change pools after a certain number of shares has been found. In this section we present a new problem that exists even in the absence of other mining pools and means that the proportional reward function is not incentive compatible.

Lemma 3. *The proportional rule* $R_i^{(prop)}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}$ *is not incentive compatible.*

This result shows that the proportional reward function is *not* incentive compatible for a fundamental reason distinct from previous criticism. Even in the absence of other pools, it does not always compel miners to report solutions immediately. The intuition behind Lemma 3 is that if a player discovers a full solution early but has been unlucky and reported a lower number of shares than they would expect based on their mining power, it is in their incentive to delay reporting their solution since on expectation their fraction of all reported shares will go up. We can draw a number of corollaries immediately from Lemma 3:

- If the current ratio of blocks exceeds the expected ratio, then a player i would report any full solution immediately.
- If the current ratio of blocks is *lower* than a player's expected ratio, then she might hold off to make up for this discrepancy.
- With fewer shares found, it's easier for a player to catch up, hence she is more willing to hold off reporting.
- After D shares have been found, any player will always report a full solution immediately, even if she has not found a single share herself.

4.2 Per-Per-Share Reward Function $R^{(\mathrm{pps})}$

The pay-per-share reward function pays a fixed amount for every share that is reported. Recall that each share is a full solution with probability $1/D$, in expectation the pool operator sees D shares for every full solution. Therefore the payout per share is $1/D$ leading to the following reward function:

$$R^{(\mathrm{pps})}(\mathbf{b}) = \frac{b_i}{D}.$$

It's easy to see that pay-per-share *is* incentive compatible.

Lemma 4. *The pay-per-share rule* $R_i^{(pps)}(\mathbf{b}) = \frac{b_i}{D}$ *is incentive compatible.*

Proof. The left hand side of (3) evaluates to

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left(R_i^{(\mathrm{pps})}(\mathbf{b}_t + \mathbf{e}_j) - R_i^{(\mathrm{pps})}(\mathbf{b}_t) \right) \\
&= \alpha_i \cdot \left(\frac{b_i + 1 - b_i}{D} \right) + (1 - \alpha_i) \cdot \left(\frac{b_i - b_i}{D} \right) \\
&= \frac{\alpha_i}{D}
\end{aligned}$$

and the right hand side evaluates to

$$\frac{\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{pps})}(\mathbf{b})\right]}{D} = \frac{\alpha_i}{D}.$$

□

This result comes at no surprise: with pay-per-share there is no benefit to delay reporting a full solution as you receive a constant payment for every reported share (or full solution). As discussed before though, it is not budget balanced.

Proposition 1. *The pay-per-share rule* $R_i^{(pps)}(\mathbf{b}) = \frac{b_i}{D}$ *is no better than* $(1/D, \infty)$*-budget balanced.*

Proof. On one extreme, if a full solution is reported before any other shares have been reported, then $R^{(\text{pps})}$ pays out $\sum_{i=1}^{n} R_i^{(\text{pps})}(\mathbf{b}) = 1/D$. On the other extreme there is no bound on how many shares can be found before a full solution must be obtained. Hence $R^{(\text{pps})}$ cannot be $(1/D, C)$-budget balanced for any finite C. Therefore it is $(1/D, \infty)$-budget balanced. □

In the absence of sabotage attacks, with the pay-per-share rule the pool operator pays out no more than it takes on *on expectation*, but keeping the probability of bankruptcy low requires large reserves for the pool operator [4]. There are several variations that ameliorate the budget balance problem [4, Sec 4], none of them quite satisfactorily.

5 A New Incentive Compatible Reward Function

In the last section we saw that two common existing methods which are possible in our model both lack one of the desiderata for reward functions. The proportional reward function may incentivize miners to delay reporting of solutions, whereas the pay-per-share function may make the pool operator liable for more than she receives from the protocol. In this section we demonstrate a reward function that satisfies all three desiderata of reward functions, while still guaranteeing a steady stream of rewards for all participants.

To satisfy the proportional payments property, it is necessary to estimate the proportion of work that each miner has done. The only information the pool operator receives within our informational model is the total number of shares per miner in the round. When only a few shares have been found in the round, every additional share may change this estimation quite significantly. When satisfying the budget-balanced property, this must translate into a large change in the payout. When there is the possibility of a large payout for an extra share, this may lead to incentive compatibility issues. Note that in practice pay-per-share reward schemes usually avoid this problem by lowering the payment amount in these cases. So to give a scheme that meets all three desiderata, we need to take an additional estimator for α_i into account. In the next subsection we show that we can use the identity of the discoverer of the full solution as this estimator.

5.1 The IC Reward Function

We propose the reward function $R_i^{(\text{ic})} : \mathbb{N}^n \times \{1, ..., n\} \to [0, 1]$, that in addition to a count of the shares per miner also includes the identity of the discoverer of the full solution. In the following let $\mathbb{1}\{c\}$ be the indicator function that is 1 if c is true, and 0 otherwise.

$$R_i^{(\text{ic})}(\mathbf{b}, s) = \frac{b_i}{\max\{||\mathbf{b}||_1, D\}} + \mathbb{1}\{i = s\} \cdot \left(1 - \frac{||\mathbf{b}||_1}{\max\{||\mathbf{b}||_1, D\}}\right)$$

There are two cases to consider for the reward function. The easiest is when the total number of reported shares $||\mathbf{b}||_1 \geq D$. In that case $\left(1 - \frac{||\mathbf{b}||_1}{\max\{||\mathbf{b}||_1, D\}}\right) = 0$, hence the reward function is identical to the proportional function. When $||\mathbf{b}||_1 < D$ each share receives a fixed reward of $1/D$, like in the pay-per-share function. However, this would leave some money on the table as the total payout would be $||\mathbf{b}||_1/D$ and $||\mathbf{b}||_1 < D$. So the remainder of the reward is given to the discoverer of the full solution.

Lemma 5. *The reward function $R^{(ic)}$ provides proportional payments.*

Proof. We first split the expression into the two relevant cases.

$$\begin{aligned}\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{ic})}(\mathbf{b}, s)\right] = \Pr(||\mathbf{b}||_1 < D) \cdot \mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 < D\right] \\ + \Pr(||\mathbf{b}||_1 \geq D) \cdot \mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 \geq D\right].\end{aligned}$$

We now show that in both cases the expected reward for miner i is α_i. When $||\mathbf{b}||_1 \geq D$ the IC rule is no different than the proportional rule, hence

$$\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 \geq D\right] = \alpha_i.$$

Now for the case where $||\mathbf{b}||_1 < D$:

$$\begin{aligned}&\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 < D\right] \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k | ||\mathbf{b}||_1 < D) \cdot \left(\frac{\mathbb{E}[b_i | ||\mathbf{b}||_1 = k]}{D} + \Pr(i = s) \cdot \left(1 - \frac{k}{D}\right)\right) \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k | ||\mathbf{b}||_1 < D) \cdot \left(\frac{\alpha_i \cdot k}{D} + \alpha_i \cdot \left(1 - \frac{k}{D}\right)\right) \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k | ||\mathbf{b}||_1 < D) \cdot \alpha_i \\ &= \alpha_i\end{aligned}$$

Here we used the fact that when a full solution is found, the probability that it was discovered by miner j is exactly its power α_j. □

Theorem 1. *The reward function $R^{(ic)}$ is incentive compatible.*

Proof. By Lemma 5, the right-hand side of condition (3) is α_i/D. Now for the left-hand side; if $||\mathbf{b}||_1 \geq D$ the rule is identical to $R^{(\text{prop})}$, so the left-hand side is at most α_i/D, hence condition (3) holds in this case. So the only case left to prove is when $||\mathbf{b}||_1 < D$.

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left(R_i^{(\text{ic})}(\mathbf{b}_t + \mathbf{e}_j, i) - R_i^{(\text{ic})}(\mathbf{b}_t, i) \right) \\
&= \alpha_i \cdot \left(\frac{b_i + 1 - b_i}{D} \right) + (1 - \alpha_i) \cdot \left(\frac{b_i - b_i}{D} \right) + \left(1 - \frac{||\mathbf{b}||_1 + 1}{D} \right) - \left(1 - \frac{||\mathbf{b}||_1}{D} \right) \\
&= \frac{\alpha_i}{D} - \left(\frac{||\mathbf{b}||_1 + 1}{D} - \frac{||\mathbf{b}||_1}{D} \right) \\
&= \frac{\alpha_i - 1}{D} \\
&\leq 0
\end{aligned}$$

So when $||\mathbf{b}||_1 < D$ the miner is expected to *lose* utility by delaying, and certainly condition (3) holds. So in all cases condition (3) holds, hence by Lemma 1 $R^{(\text{ic})}$ is incentive compatible. □

So $R^{(\text{ic})}$ satisfies proportional payments and incentive compatibility. Finally, it is also a $(1, 1)$-budget balance reward function.

Proposition 2. *$R^{(ic)}$ is a $(1, 1)$-budget balanced reward function.*

Proof. When $||\mathbf{b}||_1 < D$ the total payout is $\sum_{i=1} \frac{b_i}{D} + \sum_{i=1} \frac{b_i}{D} + \frac{D - ||\mathbf{b}||_1}{D} = \frac{||\mathbf{b}||_1}{D} + \frac{D - ||\mathbf{b}||_1}{D} = 1$. When $||\mathbf{b}||_1 \geq D$ the total payout is $\sum_{i=1} \frac{b_i}{||\mathbf{b}||_1} = \frac{||\mathbf{b}||_1}{||\mathbf{b}||_1} = 1$. □

5.2 Providing a Steady Payment Stream

While $R^{(\text{ic})}$ satisfies all three desiderata for reward functions, it might be a concern that the reward function pays out a potentially large fraction of the reward to a single miner. Miners join a pool because they prefer a steady stream of small payments over periodic large payments. We show here that the majority of the reward is paid out for shares and not full solutions, and hence that the majority of the pool's rewards are redistributed in a steady stream. This ameliorates a large part of the problem of mining alone, while guaranteeing incentive compatibility. In the Section 7 we give simulation results suggesting that this payment stream is sufficiently steady in practice.

Lemma 6. *In expectation, a fraction $1 - e^{-1} \approx 0.63$ of the rewards are given based on shares under $R^{(ic)}$.*

Proof. The fraction of the reward given to the discoverer of the full solution is

$$\begin{aligned}\sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k) \cdot \left(1 - \frac{k}{D}\right) &= \sum_{k=1}^{D-1} \frac{1}{D} \cdot \left(1 - \frac{1}{D}\right)^{k-1} \cdot \left(1 - \frac{k}{D}\right) \\ &= \left(1 - \frac{1}{D}\right)^{D} \\ &\leq e^{-1}\end{aligned}$$

The remainder of the reward is split among the reported shares, hence the payout to shares in expectation is $1 - e^{-1} \approx 0.63$. □

6 Incentive Compatibility of Pay-Per-Last-N-Shares

In previous sections we have given an overview of incentive compatibility for any reward function based on access to a history transcript $\mathcal{H}$ consisting of a count of all reported shares. By deriving incentive compatibility at this high level of abstraction allowed us to easily prove incentive compatibility for any function within this informational model.

In this section we look at incentive compatibility of a *particular* reward function that require a *more general* informational model: the Pay-Per-Last-N-Shares (PPLNS) reward function, that is widely used in practice. We first discuss the required changes in the informational model, and how the PPLNS function works, and then we show that the function is incentive compatible.

6.1 The PPLNS Reward Function

The PPLNS reward function $R^{(\text{pplns})}$ differs from the reward functions seen so far in two important ways. Firstly, it maintains a history of reported shares that spans *multiple* rounds. So what happens in round T is no longer isolated from what happens in round $T+1$. Secondly, the method takes the order of reported shares into account in a specific way: it maintains a sliding window of length N and divides the reward proportionally over these N shares. So the history transcript $\mathcal{H}$ that $R^{(\text{pplns})}$ uses is $\mathbf{s} = [s_{t-N}, s_{t+1-N}, ..., s_t]$ (an ordered list of N elements) and the reward function is:

$$R^{(\text{pplns})}(\mathbf{s}) = \frac{\#\{s_j : s_j \in \mathbf{s} \wedge s_j = i\}}{N}$$

Since the order of reports matter, we say that shares fall into *slots*. Each slot states if the report contains either a full solution or a share, and who the miner was that reported it.

6.2 Incentive Compatibility of PPLNS

We do not have a general condition under which reward functions in this informational model are incentive compatible, so we argue incentive compatibility directly. In this section we consider both reporting of shares as well as full solutions. For each, we consider a binary strategy space: either report the share/solution immediately, or delay reporting until one more share is found.[9]

Lemma 7. *For the reward function $R^{(pplns)}$; a miner i reports shares immediately when her mining power $\alpha_i < 1 - \frac{D}{N}$.*

In the previous section there were potential benefits, and harms to delaying the report of a share, but there was no opportunity cost. Since miner i did not have a full solution, her delay did not cause unnecessary work for all miners in the pool. For full solutions we have to take the opportunity cost of letting all miners work on a block for which a full solution is already found into account. This will guarantee incentive compatibility whenever $N > D$.

Lemma 8. *For the reward function $R^{(pplns)}$; a miner i reports full solutions immediately when $N \geq D$.*

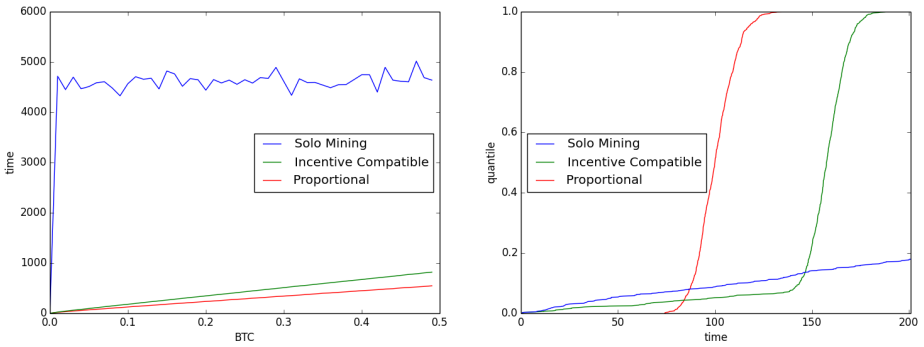
7 Simulations

The typical way to compare different reward function is to look at the variance of payout for a single share [4], with a lower variance considered better. However, the raw variance can be quite misleading for a reward function like $R^{(\mathrm{ic})}$ which allocates some revenue in a steady stream and some in a lumpy stream, assigning a variance almost as high as solo mining.

In Figure 1a we plot the time it takes for a miner to gain a given number of bitcoins with 99% certainty. We run a simulation for a miner i with $\alpha_i = 0.001$, $D = 1,000,000$. A unit of time corresponds to the expected time it takes for all miners combined to find a full solution (in reality this is about 10 minutes) and we normalize the reward for finding a full solution to be Ƀ 1 (in reality this currently about Ƀ 25, although it changes over time). The lines indicate for each of three reward functions how long one has to wait to gain a given amount of Ƀ with 99% probability. First of all, observe that for solo mining the time is about 4500 rounds and it does not increase with time. This is because whether a miner wants to obtain Ƀ 0.001, or Ƀ 0.9, they have to find a full solution to reach this target. So the blue line really indicates the time it takes to find a full solution. Even though in expectation this takes 1000 rounds (for $\alpha_i = 0.001$), in 1% of cases a miner has to wait in excess of 4500 rounds.

It can be seen that the incentive compatible scheme requires somewhat longer to reach the same target than the proportional scheme. This is because not all reward is shared according to the reported shares, but is partly distributed over

[9] This makes our results slightly less general in this setting than for the reduced information setting, where the miner could delay for any delay d.



(a) 99^{th} percentile time to earn rewards (b) CDF of time to earn ₿0.1

Fig. 1: Simulation results for our new incentive-compatible reward function.

the discoverers of full solutions. However, no matter what the target is, the difference in time required differs no more than a small multiplicative factor. Finally, note that since it takes longer to reach targets with high probability, the expected payouts between all three functions is the same.

In Figure 1b we plot the CDF of the time needed to earn ₿0.1. With overwhelming probability $R^{(\text{prop})}$ pays out at least ₿0.1 within 150 rounds, and $R^{(\text{ic})}$ pays out at least ₿0.1 within 200 rounds. Solo mining does not fit on this scale and it wouldn't be until around round 7,000 before a miner makes at least ₿0.1 with overwhelming probability.

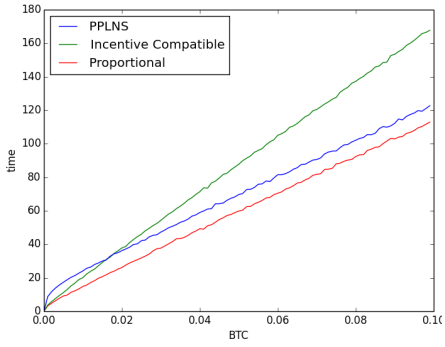


Fig. 2: Comparison of the new incentive compatible scheme to PPLNS.

We compare the new incentive compatible scheme to the PPLNS scheme in Figure 2. Here we can see that the new incentive compatible scheme performs worse by a small multiplicative factor, the PPLNS scheme performs worse by a small additive factor. This means that for small Bitcoin targets it would be

faster to use the IC reward function, whereas for larger target the PPLNS reward function performs better.

From these simulations we can conclude that the trade-off for using the incentive compatible or PPLNS reward function compared to the proportional reward function is a modest delay in the time it would take miners to reach a minimal amount of bitcoin with high probability. In return we get a scheme in which it is obvious for miners what the most profitable strategy for them is.

8 Conclusions & Open Problems

We set out with a simple question: as a mining pool operator, in the absence of other mining pools or outside options, which reward functions will incentivize miners to report full solutions immediately? In this simple model it would be reasonable to assume that miners always have an incentive to report immediately. However, we show that for proportional rewards, there are situations in which miners prefer to hold on to a full solution temporarily in order to improve their payout, harming the entire pool in the process. We also defined a novel reward function that is incentive compatible in this model (and remains so even in more powerful models). While this new scheme is not quite as efficient as proportional rewards in terms of smoothing the miners' revenue streams, it comes reasonably close in practice. We have also showed that the PPLNS reward function is incentive compatible. For a pool operator there are some tradeoffs in deciding to use our new incentive compatible scheme versus the PPLNS scheme. The latter requires a certain lead-up time, where the rewards to miners are below their fraction of the mining power. It also requires pool operators to maintain a more complex state and the payouts are arguably somewhat less transparent. On the other hand, our new incentive compatible method sometimes pays out a rather large amount to the discoverer of the full solution.

We have given a first informational model for which we can characterize incentive compatibility for all reward functions that fall in the model. We've also looked at a particular reward function that falls outside this model, and proved incentive compatibility from first principles. The next enticing question is to see if we can characterize incentive compatibility in this larger informational model at a high level, so that we can quickly identify which other reward functions would be incentive compatible. There are many reward functions in use today [4] that are not covered by any of our results. For example, the Geometric Method weights shares differently according to the order of shares in a round and Slush's Method takes the time of reported shares in a round into account. Defining a common informational model, characterizing incentive compatibility in this model, and classifying these methods remains an interesting open problem.

We stress that our incentive-compatible reward function will remain so even in a model with more extensive history transcripts. Our goal was to introduce the first rigorous, although simplified by omitting notions of time or order of share reporting, model of Bitcoin mining pools and demonstrate that even this simple model can lead to non-intuitive results.

References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.
2. Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, May 2015.
3. Andrew Miller, Elaine Shi, Ahmed Kosba, and Jonathan Katz. Nonoutsourceable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions (preprint), 2014.
4. Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.
5. Ittay Eyal. The Miner's Dilemma. In *IEEE Symposium on Security and Privacy*, 2015.
6. Aron Laszka, Benjamin Johnson, and Jens Grossklags. When Bitcoin Mining Pools Run Dry: A Game-Theoretic Analysis of the Long-Term Impact of Attacks Between Mining Pools. In *Workshop on Bitcoin Research*, 2015.
7. Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *Workshop on Bitcoin Research*, 2014.
8. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

A Proofs

A.1 Proof of Lemma 1

For a reward function R, a player i has an incentive to report full solutions immediately, iff the following condition holds for all $\{\alpha_i\}_{i=1}^n, \mathbf{b}_t, D, i$:

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) \;\leq\; \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{D} \tag{4}$$

Proof. ($\Rightarrow$) This direction is straightforward: when it is beneficial to delay until 1 more share is reported, then there exists a profitable delay (namely $d = 1$).

($\Leftarrow$) We need to prove that for all d, the following inequality holds:

$$\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1 = d} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)) \;\leq\; \frac{d}{D}\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right].$$

We prove this by induction on d, where the induction hypothesis is equation (2). For the base case $d = 1$ the statement follows directly from condition (3). So consider the case $d > 1$:

$$
\begin{aligned}
&\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1 = d} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)) \\
&= \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \left(R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t) \right. \\
&\qquad \left. + \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1 = d-1} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j + \mathbf{b}) - R_i(\mathbf{b}_t + \mathbf{e}_j)) \right) \\
&\leq \frac{1}{D} \mathbb{E}_{\mathbf{b}} \left[R_i(\mathbf{b}) \right] \\
&\qquad + \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1 = d-1} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j + \mathbf{b}) - R_i(\mathbf{b}_t + \mathbf{e}_j)) \\
&\leq \frac{1}{D} \mathbb{E}_{\mathbf{b}} \left[R_i(\mathbf{b}) \right] + \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \frac{d-1}{D} \mathbb{E}_{\mathbf{b}} \left[R_i(\mathbf{b}) \right] \\
&= \frac{d}{D} \mathbb{E}_{\mathbf{b}} \left[R_i(\mathbf{b}) \right]
\end{aligned}
$$

where the first inequality follows from condition (3), and the second from the induction hypothesis. □

A.2 Proof of Lemma 2

Miners report shares immediately if and only if the reward function R is monotonically increasing each component. That is: for all i, and $\mathbf{b}$:

$$R_i(\mathbf{b} + \mathbf{e}_i) > R_i(\mathbf{b}).$$

Proof. Since the order or timing of shares does not matter, for analysis purposes we can assume the following scheme: as soon as a full solution is reported the pool operator asks all miners for the shares that they found. If the reward function R is monotonically increasing then each additional share that i reports increases her share, hence she will report all shares. Conversely, if R is not monotonically increasing at some $\mathbf{b}$, then if miner i has $b_i + 1$ shares, and all other miners have reported shares according to $\mathbf{b}$, then she will not report her last share.

Now consider the original problem: when a miner finds a share, will she report it immediately? If she finds a share and the reward function is monotonically increasing, then reporting it immediately can only increase her reward, whereas delaying it may mean that someone else reports the full solution before she reports her share, in which case she loses the opportunity to report. Thus she will report immediately. □

A.3 Proof of Lemma 3

The proportional rule $R_i^{(\text{prop})}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}$ is not incentive compatible.

Proof. Instantiate (3) for the proportional rule. For the right hand side we have:

$$\begin{aligned}
\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{prop})}(\mathbf{b}, s)\right]/D &= \frac{1}{D}\sum_{k=1}^{\infty}\Pr[\text{full solution is found at } k^{\text{th}} \text{ block}]\frac{\mathbb{E}[b_i|k]}{k} \\
&= \frac{1}{D}\sum_{k=1}^{\infty}\left(1-\frac{1}{D}\right)^{k-1}\frac{1}{D}\frac{k\cdot\alpha_i}{k} \\
&= \frac{\alpha_i}{D}\sum_{k=1}^{\infty}\left(1-\frac{1}{D}\right)^{k-1}\frac{1}{D} \\
&= \frac{\alpha_i}{D}
\end{aligned}$$

Now for the left hand side. In the following let $k = ||\mathbf{b}_t||_1$:

$$\begin{aligned}
&\sum_{j=1}^{n}\alpha_j\cdot\left(R_i^{(\text{prop})}(\mathbf{b}_t+\mathbf{e}_j) - R_i^{(\text{prop})}(\mathbf{b}_t)\right) \\
&= \alpha_i\cdot\left(\frac{b_i+1}{k+1}\right) + (1-\alpha_i)\cdot\left(\frac{b_i}{k+1}\right) - \frac{b_i}{k} \\
&= \frac{\alpha_i b_i + \alpha_i + b_i - \alpha_i b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i + b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i}{k+1} + b_i\left(\frac{1}{k+1} - \frac{1}{k}\right) \\
&= \frac{\alpha_i}{k+1} - \frac{b_i}{k(k+1)} \\
&= \frac{\alpha_i - \frac{b_i}{k}}{k+1}
\end{aligned}$$

Recall that for an incentive compatible scheme we need:

$$\begin{aligned}
\frac{\alpha_i - \frac{b_i}{k}}{k+1} &\leq \frac{\alpha_i}{D} \\
\alpha_i - \frac{b_i}{k} &\leq \alpha_i\frac{k+1}{D} \\
\frac{b_i}{k} &\geq \alpha_i\left(1-\frac{k+1}{D}\right).
\end{aligned}$$

This condition is not guaranteed to be satisfied. In particular, for every $\alpha_i > 0$ there exist positive values b_i, k, D such that the condition is violated. □

A.4 Proof of Lemma 7

For the reward function $R^{(\text{pplns})}$; a miner i reports shares immediately when her mining power $\alpha_i < 1 - \frac{D}{N}$.

Proof. We directly calculate the expected revenue for delay versus reporting. When the miner decides to delay reporting a share until one more share/solution is found, she aims to move the sliding window of slots for which the share is eligible to receive reward one further into the future. This means that –as long as no other miner finds a full solution and reports it– the share is active for $N-1$ of the same slots, so any reward she receives from full solutions in those slots she will get regardless of her choice to report immediately versus delaying. On the upshot, it could be the case that the one additional slot she's eligible for in the future yields a full solution. This will happen with probability $1/D$ (since a share constitutes a full solution with probability $1/D$) and in that case the share gets an extra payout of $1/N$ for the delayed share, yielding an expected benefit for delaying of $1/ND$.

However, there is also a risk associated with delaying. With probability $1-\alpha_i$ a share will be found by a different miner, and with probability $1/D$ it will constitute a full solution. When this happens, miner i will no longer be able to report the share as it was discovered for a previous round. The expected value per share is $1/D$ (as it's active for N rounds, in which in expectation N/D full solutions will be reported for a value of $1/N$ each) hence the expected harm for delaying the report is $(1-\alpha_i)\frac{1}{D^2}$.

So the miner will report the share immediately iff $\frac{1}{ND} < (1-\alpha_i)\cdot\frac{1}{D^2}$. Plugging in $\alpha_i < \frac{D}{N}$ leads to $(1-\alpha_i)\cdot\frac{1}{D^2} \geq \frac{D}{N}\cdot\frac{1}{D^2} = \frac{1}{ND}$ so the condition holds, and miners report shares immediately. □

A.5 Proof of Lemma 8

For the reward function $R^{(\text{pplns})}$; a miner i reports full solutions immediately when $N \geq D$.

Proof. In delaying a full solution, the hope is to get another share to report before the miner reports the full solution. This happens with probability $\alpha_1 \cdot \frac{D-1}{D}$ (we need the share to *not* be a full solution) and the additional value to this share would be $\frac{1}{N}$ compared to it being reported after the full solution. However, while waiting for a share, with probability $\frac{1}{D}$ the next share will be a full solution, either found by miner i, or one of the other miners. Regardless of who finds the solution, the previous full solution that miner i was sitting on has become worthless: either a different miner reported the full solution ending the round and thus making the delayed full solution worthless, or miner i now has 2 full solutions of which she can report only one. When this happens, she loses the solution whose expected value is $1/D$ (as this is counted as a share for future). So the expected upshot for delaying the solution is $\alpha_1\frac{D-1}{D}\frac{1}{N}$ and the expected harm is $\frac{1}{D^2}$.

In addition to this, when the miner chooses to delay until one more share is found, she lets all miners in the pool work on a block for which she already has a solution. If everyone were to spend that effort on a new block, that work would in expectation constitute $1/D$ of the work for a new block, of which in expectation miner i would receive α_i of the reward. Thus, the opportunity cost is $\frac{\alpha_i}{D}$. Therefore, a miner will report a full solution immediately iff $\alpha_i \frac{D-1}{D} \frac{1}{N} - \frac{1}{D^2} \leq \frac{\alpha_i}{D}$ which holds whenever $N \geq D$.

B Incentive Compatibility When Other Miners Can Find a Block Before You Report

In Section 3 we showed that there is a simple condition that precisely characterizes when a reward function R is incentive compatible, under the assumption that no other miner finds and reports a full solution during this delay. In reality a miner does have to take this possibility into account, so in this section we show exactly how the IC condition changes when we drop this assumption.

If we decide to delay reporting the full solution until one additional share is found, then with probability $1/D$ that share will actually be a full solution itself. Without loss of generality we may assume that this solution will be reported immediately (otherwise we could simply ignore its effect). Recall that $\mathbf{b}_t$ is the number of reported shares per miner *including* the unreported full solution that miner i has, and that $\mathbf{e}_j$ is the vector that has zeros everywhere except its j^{th} component, where it is 1. So the expected payout for delaying for one round becomes:

$$\frac{1}{D} \sum_j \alpha_j R_i(\mathbf{b}_t - \mathbf{e}_i + \mathbf{e}_j) + \frac{D-1}{D} \sum_j \alpha_j R_i(\mathbf{b}_t + \mathbf{e}_j)$$

Thus the condition of incentive compatibility is:

$$\begin{aligned} &\frac{1}{D} \sum_j \alpha_j \left(R_i(\mathbf{b}_t - \mathbf{e}_i + \mathbf{e}_j) - R_i(\mathbf{b}_t)\right) \\ + &\frac{D-1}{D} \sum_j \alpha_j \left(R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)\right) \quad \leq \quad \frac{\mathbb{E}_{\mathbf{b}}[R_i(\mathbf{b})]}{D} \end{aligned}$$

For the reward functions that are monotonic increasing in each component (which by Lemma 2 are precisely the reward functions where miners always report all shares) this additional term is negative. Therefore, the IC condition is only easier to satisfy. This means that reward functions that are proven to be incentive compatible using Lemma 1 are still incentive compatible. However, one might worry that our proof that the proportional reward function is not incentive compatible might break. We show next that the thread of being scooped actually does not impact the result qualitatively.

B.1 Proportional

For the proportional reward function we can instantiate the left-hand side as (taking $||\mathbf{b}_t||_1 = k$):

$$\begin{aligned}
&\frac{1}{D}\left(\alpha_i\left(\frac{b_i}{k}-\frac{b_i}{k}\right)+(1-\alpha_i)\left(\frac{b_i-1}{k}-\frac{b_i}{k}\right)\right)\\
&+\frac{D-1}{D}\left(\alpha_i\left(\frac{b_i+1}{k+1}-\frac{b_i}{k}\right)+(1-\alpha_i)\left(\frac{b_i}{k+1}-\frac{b_i}{k}\right)\right)\\
&=\frac{1}{D}\frac{1-\alpha_i}{k}+\frac{D-1}{D}\frac{\alpha_i-\frac{b_i}{k}}{k+1}
\end{aligned}$$

So the proportional reward function is IC if and only if

$$\frac{1}{D}\frac{1-\alpha_i}{k}+\frac{D-1}{D}\frac{\alpha_i-\frac{b_i}{k}}{k+1}\leq\frac{\alpha_i}{D}.$$

Again this is not guaranteed to be satisfied, in fact the same parameters as last time, i.e. $b_i = 2$, $k = 10$, $\alpha_i = 1/2$ and $D = 20$. So including the possibility of another miner finding a full solution does not qualitatively change the incentive compatibility results, although quantitatively there may be situations where a miner would choose to delay if she does not fear being scooped, but choose to report if she does include this possibility.

C Multiple Pools

In the main text we've assumed that there are no other pools that compete for finding solutions to the cryptographic puzzle. This is reasonable from the perspective of proving positive results: any incentive compatible scheme should be incentive compatible regardless of how much mining power other pools have.

However, to convincingly reject the proportional rule as not incentive compatible, we should take the effect of other pools into account. In the following let $\sum_{i=1}^{n}\alpha_i = \alpha_P < 1$ be the total mining power of the pool, so all other mining power —of both other pools and solo miners— is $1-\alpha_P$. For notational simplicity we do not consider being scooped by a different miner in our own pool; it's obvious how this can be included by comparing the results to the one in Appendix B. When we consider to delay reporting a full solution until one more share is found —either inside or outside the pool— then our expected utility for doing so is

$$\sum_j \alpha_j R_i(\mathbf{b}_t+\mathbf{e}_j)+(1-\alpha_P)\left(\frac{1}{D}\cdot 0+\frac{D-1}{D}R_i(\mathbf{b}_t)\right).$$

We don't really care if some other pool finds another share. This does not affect us. But if another pool finds a full solution and reports it, then our mining pool

misses out on a complete payment that it could have received. So the condition for incentive compatibility becomes

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) - (1 - \alpha_P)\frac{R_i(\mathbf{b})}{D} \;\leq\; \alpha_P \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b}_t)\right]}{D}.$$

Under the assumption that the pool in expectation will collect α_P of the total reward among pools, and that miner i collects $\frac{\alpha_i}{\alpha_P}$ of the pool she is in, the right-hand side will remain $\frac{\alpha_i}{D}$. The new term on the left-hand side is simply $(1 - \alpha_P)\frac{b_i}{kD}$. The other term on the left-hand side changes slightly:

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left(R_i^{(\text{prop})}(\mathbf{b}_t + \mathbf{e}_j) - R_i^{(\text{prop})}(\mathbf{b}_t)\right) \\
&= \alpha_i \cdot \left(\frac{b_i + 1}{k+1}\right) + (\alpha_P - \alpha_i) \cdot \left(\frac{b_i}{k+1}\right) - \frac{b_i}{k} \\
&= \frac{\alpha_i b_i + \alpha_i + \alpha_P b_i - \alpha_i b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i + \alpha_P b_i}{k+1} - \frac{b_i}{k}.
\end{aligned}$$

This cannot be simplified to the same convenient expression we had in Section 3. Combining these terms the condition for incentive compatibility of the proportional reward function becomes:

$$\frac{\alpha_i + \alpha_P b_i}{k+1} - \frac{b_i}{k} - (1 - \alpha_P)\frac{b_i}{kD} \leq \frac{\alpha_i}{D}$$

and after rewriting this:

$$\frac{\alpha_i}{k+1} + \alpha_P b_i \left(\frac{1}{kD} + \frac{1}{k+1}\right) - \frac{b_i}{k}\left(1 + \frac{1}{D}\right) \leq \frac{\alpha_i}{D}$$