Majority Is Not Enough: Bitcoin Mining Is Vulnerable

Ittay Eyal(✉) and Emin Gün Sirer

Department of Computer Science, Cornell University, Ithaca, USA
ittay.eyal@cornell.edu, egs@systems.cs.cornell.edu

Abstract. The Bitcoin cryptocurrency records its transactions in a public log called the blockchain. Its security rests critically on the distributed protocol that maintains the blockchain, run by participants called miners. Conventional wisdom asserts that the mining protocol is incentive-compatible and secure against colluding minority groups, that is, it incentivizes miners to follow the protocol as prescribed.

We show that the Bitcoin mining protocol is not incentive-compatible. We present an attack with which colluding miners obtain a revenue larger than their fair share. This attack can have significant consequences for Bitcoin: Rational miners will prefer to join the selfish miners, and the colluding group will increase in size until it becomes a majority. At this point, the Bitcoin system ceases to be a decentralized currency.

Unless certain assumptions are made, selfish mining may be feasible for any group size of colluding miners. We propose a practical modification to the Bitcoin protocol that protects Bitcoin in the general case. It prohibits selfish mining by pools that command less than 1/4 of the resources. This threshold is lower than the wrongly assumed 1/2 bound, but better than the current reality where a group of any size can compromise the system.

1 Introduction

Bitcoin [23] is a cryptocurrency that has recently emerged as a popular medium of exchange, with a rich and extensive ecosystem. The Bitcoin network runs at over 42×10^{18} FLOPS [9], with a total market capitalization around 12 billion US Dollars as of January 2014 [10]. Central to Bitcoin's operation is a global, public log, called the *blockchain*, that records all transactions between Bitcoin clients. The security of the blockchain is established by a chain of cryptographic puzzles, solved by a loosely-organized network of participants called *miners*. Each miner that successfully solves a cryptopuzzle is allowed to record a set of transactions, and to collect a reward in Bitcoins. The more *mining power* (resources) a miner applies, the better are its chances to solve the puzzle first. This reward structure provides an incentive for miners to contribute their resources to the system, and is essential to the currency's decentralized nature.

This research was supported by the NSF Trust STC and by DARPA.

N. Christin and R. Safavi-Naini (Eds.): FC 2014, LNCS 8437, pp. 436–454, 2014.
DOI: 10.1007/978-3-662-45472-5_28

The Bitcoin protocol requires a majority of the miners to be *honest*; that is, follow the Bitcoin protocol as prescribed. By construction, if a set of colluding miners comes to command a majority of the mining power in the network, the currency stops being decentralized and becomes controlled by the colluding group. Such a group can, for example, prohibit certain transactions, or all of them. It is, therefore, critical that the protocol be designed such that miners have no incentive to form such large colluding groups.

Empirical evidence shows that Bitcoin miners behave strategically and form pools. Specifically, because rewards are distributed at infrequent, random intervals, miners form mining pools in order to decrease the variance of their income rate. Within such pools, all members contribute to the solution of each cryptopuzzle, and share the rewards proportionally to their contributions. To the best of our knowledge, such pools have been benign and followed the protocol so far.

Indeed, conventional wisdom has long asserted that the Bitcoin mining protocol is equitable to its participants and secure against malfeasance by a non-majority attacker (Sect. 7). Barring recently-explored Sybil attacks on transaction propagation [4], there were no known techniques by which a minority of colluding miners could earn disproportionate benefits by deviating from the protocol. Because the protocol was believed to reward miners strictly in proportion to the ratio of the overall mining power they control, a miner in a large pool was believed to earn the same revenue as it would in a small pool. Consequently, if we ignore the fixed cost of pool operation and potential economies of scale, there is no advantage for colluding miners to organize into ever-increasing pools. Therefore, pool formation by honest rational miners poses no threat to the system.

In this paper, we show that the conventional wisdom is wrong: the Bitcoin mining protocol, as prescribed and implemented, is not incentive-compatible. We describe a strategy that can be used by a minority pool to obtain more revenue than the pool's fair share, that is, more than its ratio of the total mining power.

The key idea behind this strategy, called Selfish Mining, is for a pool to keep its discovered blocks private, thereby intentionally forking the chain. The honest nodes continue to mine on the public chain, while the pool mines on its own private branch. If the pool discovers more blocks, it develops a longer lead on the public chain, and continues to keep these new blocks private. When the public branch approaches the pool's private branch in length, the selfish miners reveal blocks from their private chain to the public.

This strategy leads honest miners that follow the Bitcoin protocol to waste resources on mining cryptopuzzles that end up serving no purpose. Our analysis demonstrates that, while both honest and selfish parties waste some resources, the honest miners waste proportionally more, and the selfish pool's rewards exceed its share of the network's mining power, conferring it a competitive advantage and incentivizing rational miners to join the selfish mining pool.

We show that, above a certain threshold size, the revenue of a selfish pool rises superlinearly with pool size above its revenue with the honest strategy. This fact has critical implications for the resulting system dynamics. Once a

selfish mining pool reaches the threshold, rational miners will preferentially join selfish miners to reap the higher revenues compared to other pools. Such a selfish mining pool can quickly grow towards a majority. If the pool tips the majority threshold (due to the addition of malicious actors aimed at undermining the system, rational actors wishing to usurp the currency, perhaps covertly, or due to momentum in pool popularity), it can switch to a modified protocol that ignores blocks generated outside the pool, to become the only creator of blocks and reap all the mining revenue. A majority pool wishing to remain covert may remain a benign monopolist, accepting blocks from third-parties on occasion to provide the illusion of decentralization, while retaining the ability to reap full revenue when needed, as well as the ability to launch double-expenditure attacks against merchants. Either way, the decentralized nature of the currency will have collapsed, and a single entity, the selfish pool manager, will control the system.

Since a selfish mining pool that exceeds threshold size poses a threat to the Bitcoin system, we characterize how the threshold varies as a function of message propagation speed in the network. We show that, for a mining pool with high connectivity and good control on information flow, the threshold is close to zero. This implies that, if less than 100% of the miners are honest, the system may not be incentive compatible: The first selfish miner will earn proportionally higher revenues than its honest counterparts, and the revenue of the selfish mining pool will increase superlinearly with pool size.

We further show that the Bitcoin mining protocol will never be safe against attacks by a selfish mining pool that commands more than 1/3 of the total mining power of the network. Such a pool will always be able to collect mining rewards that exceed its proportion of mining power, even if it loses every single block race in the network. The resulting bound of 2/3 for the fraction of Bitcoin mining power that needs to follow the honest protocol to ensure that the protocol remains resistant to being gamed is substantially lower than the 50% figure currently assumed, and difficult to achieve in practice. Finally, we suggest a simple modification to the Bitcoin protocol that achieves a threshold of 1/4. This change is backwards-compatible and *progressive*; that is, it can be adopted by current clients with modest changes, does not require full adoption to provide a benefit, and partial adoption will proportionally increase the threshold.

In summary, the contributions of this work are:

1. Introduction of the Selfish-Mine strategy, which demonstrates that Bitcoin mining is not incentive compatible (Sect. 3).
2. Analysis of Selfish-Mine, and when it can benefit a pool (Sect. 4).
3. Analysis of majority-pool formation in face of selfish mining (Sect. 5).
4. A simple backward-compatible progressive modification to the Bitcoin protocol that would raise the threshold from zero to 1/4 (Sect. 6).

We are unaware of previous work that addresses the security of the blockchain. We provide an overview of related work in Sect. 7, and discuss the implications of our results in Sect. 8.

2 Preliminaries

Bitcoin is a distributed, decentralized crypto-currency [6–8,23]. The users of Bitcoin are called *clients*, each of whom can command accounts, known as *addresses*. A client can send Bitcoins to another client by forming a transaction and committing it into a global append-only log called the *blockchain*. The blockchain is maintained by a network of *miners*, which are compensated for their effort in Bitcoins. Bitcoin transactions are protected with cryptographic techniques that ensure only the rightful owner of a Bitcoin address can transfer funds from it.

The miners are in charge of recording the transactions in the blockchain, which determines the ownership of Bitcoins. A client owns x Bitcoins at time t if, in the prefix of the blockchain up to time t, the aggregate of transactions involving that client's address amounts to x. Miners only accept transactions if their inputs are unspent.

2.1 Blockchain and Mining

The blockchain records the transactions in units of blocks. Each block includes a unique ID, and the ID of the preceding block. The first block, dubbed *the genesis block*, is defined as part of the protocol. A valid block contains a solution to a cryptopuzzle involving the hash of the previous block, the hash of the transactions in the current block, and a Bitcoin address which is to be credited with a reward for solving the cryptopuzzle. This process is called Bitcoin *mining*, and, by slight abuse of terminology, we refer to the creation of blocks as *block mining*. The specific cryptopuzzle is a double-hash whose result has to be smaller than a set value. The problem difficulty, set by this value, is dynamically adjusted such that blocks are generated at an average rate of one every ten minutes.

Any miner may add a valid block to the chain by simply publishing it over an overlay network to all other miners. If two miners create two blocks with the same preceding block, the chain is *forked* into two *branches*, forming a tree. Other miners may subsequently add new valid blocks to either branch. When a miner tries to add a new block after an existing block, we say it *mines on* the existing block. This existing block may be the head of a branch, in which case we say the miner mines on the head of the branch, or simply on the branch.

The formation of branches is undesirable since the miners have to maintain a globally-agreed totally ordered set of transactions. To resolve forks, the protocol prescribes miners to adopt and mine on the longest chain.[1] All miners add blocks to the longest chain they know of, or the first one they heard of if there are branches of equal length. This causes forked branches to be pruned; transactions in pruned blocks are ignored, and may be resubmitted by clients.

[1] The criterion is actually the most difficult chain in the block tree, i.e., the one that required (in expectancy) the most mining power to create. To simplify presentation, and because it is usually the case, we assume the set difficulty at the different branches is the same, and so the longest chain is also the most difficult one.

We note that block dissemination over the overlay network takes seconds, whereas the average mining interval is 10 min. Accidental bifurcation is therefore rare, and occurs on average once about every 60 blocks [12].

When a miner creates a block, it is compensated for its efforts with Bitcoins. This compensation includes a per-transaction fee paid by the users whose transactions are included, as well as an amount of new Bitcoins that did not exist before.[2]

2.2 Pool Formation

The probability of mining a block is proportional to the computational resources used for solving the associated cryptopuzzle. Due the nature of the mining process, the interval between mining events exhibits high variance from the point of view of a single miner. A single home miner using a dedicated ASIC is unlikely to mine a block for years [31]. Consequently, miners typically organize themselves into mining *pools*. All members of a pool work together to mine each block, and share their revenues when one of them successfully mines a block. While joining a pool does not change a miner's expected revenue, it decreases the variance and makes the monthly revenues more predictable.

3 The Selfish-Mine Strategy

First, we formalize a model that captures the essentials of Bitcoin mining behavior and introduces notation for relevant system parameters. Then we detail the selfish mining algorithm.

3.1 Modeling Miners and Pools

The system is comprised of a set of miners $1, \ldots, n$. Each miner i has mining power m_i, such that $\sum_{i=1}^{n} m_i = 1$. Each miner chooses a chain head to mine, and finds a subsequent block for that head after a time interval that is exponentially distributed with mean m_i^{-1}. We assume that miners are rational; that is, they try to maximize their revenue, and may deviate from the protocol to do so.

A group of miners can form a pool that behaves as single agent with a centralized coordinator, following some strategy. The mining power of a pool is the sum of mining power of its members, and its revenue is divided among its members according to their relative mining power [30]. The *expected relative revenue*, or simply the *revenue* of a pool is the expected fraction of blocks that were mined by that pool out of the total number of blocks in the longest chain.

[2] The rate at which the new Bitcoins are generated is designed to slowly decrease towards zero, and will reach zero when almost 21 million Bitcoins are created. Then, the miners' revenue will be only from transaction fees.

3.2 Selfish-Mine

We now describe our strategy, called Selfish-Mine. As we show in Sect. 4, Selfish-Mine allows a pool of sufficient size to obtain a revenue larger than its ratio of mining power. For simplicity, and without loss of generality, we assume that miners are divided into two groups, a colluding minority pool that follows the selfish mining strategy, and a majority that follows the honest mining strategy (others). It is immaterial whether the honest miners operate as a single group, as a collection of groups, or individually.

The key insight behind the selfish mining strategy is to force the honest miners into performing wasted computations on the stale public branch. Specifically, selfish mining forces the honest miners to spend their cycles on blocks that are destined to not be part of the blockchain.

Selfish miners achieve this goal by selectively revealing their mined blocks to invalidate the honest miners' work. Approximately speaking, the selfish mining pool keeps its mined blocks private, secretly bifurcating the blockchain and creating a private branch. Meanwhile, the honest miners continue mining on the shorter, public branch. Because the selfish miners command a relatively small portion of the total mining power, their private branch will not remain ahead of the public branch indefinitely. Consequently, selfish mining judiciously reveals blocks from the private branch to the public, such that the honest miners will switch to the recently revealed blocks, abandoning the shorter public branch. This renders their previous effort spent on the shorter public branch wasted, and enables the selfish pool to collect higher revenues by incorporating a higher fraction of its blocks into the blockchain.

Armed with this intuition, we can fully specify the selfish mining strategy, shown in Algorithm 1. The strategy is driven by mining events by the selfish pool or by the others. Its decisions depend only on the relative lengths of the selfish pool's private branch versus the public branch. It is best to illustrate the operation of the selfish mining strategy by going through sample scenarios involving different public and private chain lengths.

When the public branch is longer than the private branch, the selfish mining pool is behind the public branch. Because of the power differential between the selfish miners and the others, the chances of the selfish miners mining on their own private branch and overtaking the main branch are small. Consequently, the selfish miner pool simply adopts the main branch whenever its private branch falls behind. As others find new blocks and publish them, the pool updates and mines at the current public head.

When the selfish miner pool finds a block, it is in an advantageous position with a single block lead on the public branch on which the honest miners operate. Instead of naively publishing this private block and notifying the rest of the miners of the newly discovered block, selfish miners keep this block private to the pool. There are two outcomes possible at this point: either the honest miners discover a new block on the public branch, nullifying the pool's lead, or else the pool mines a second block and extends its lead on the honest miners.

In the first scenario where the honest nodes succeed in finding a block on the public branch, nullifying the selfish pool's lead, the pool immediately publishes

Algorithm 1. Selfish-Mine

```
1  on Init
2      public chain ← publicly known blocks
3      private chain ← publicly known blocks
4      privateBranchLen ← 0
5      Mine at the head of the private chain.

6  on My pool found a block
7      Δ_prev ← length(private chain) − length(public chain)
8      append new block to private chain
9      privateBranchLen ← privateBranchLen + 1
10     if Δ_prev = 0 and privateBranchLen = 2 then          (Was tie with branch of 1)
11         publish all of the private chain                 (Pool wins due to the lead of 1)
12         privateBranchLen ← 0
13     Mine at the new head of the private chain.

14 on Others found a block
15     Δ_prev ← length(private chain) − length(public chain)
16     append new block to public chain
17     if Δ_prev = 0 then
18         private chain ← public chain                     (they win)
19         privateBranchLen ← 0
20     else if Δ_prev = 1 then
21         publish last block of the private chain          (Now same length. Try our luck)
22     else if Δ_prev = 2 then
23         publish all of the private chain                 (Pool wins due to the lead of 1)
24         privateBranchLen ← 0
25     else                                                 (Δ_prev > 2)
26         publish first unpublished block in private block.
27     Mine at the head of the private chain.
```

its private branch (of length 1). This yields a toss-up where either branch may win. The selfish miners unanimously adopt and extend the previously private branch, while the honest miners will choose to mine on either branch, depending on the propagation of the notifications. If the selfish pool manages to mine a subsequent block ahead of the honest miners that did not adopt the pool's recently revealed block, it publishes immediately to enjoy the revenue of both the first and the second blocks of its branch. If the honest miners mine a block after the pool's revealed block, the pool enjoys the revenue of its block, while the others get the revenue from their block. Finally, if the honest miners mine a block after their own block, they enjoy the revenue of their two blocks while the pool gets nothing.

In the second scenario, where the selfish pool succeeds in finding a second block, it develops a comfortable lead of two blocks that provide it with some cushion against discoveries by the honest miners. Once the pool reaches this point, it continues to mine at the head of its private branch. It publishes one block from its private branch for every block the others find. Since the selfish pool is a minority, its lead will, with high probability, eventually reduce to a single block. At this point, the pool publishes its private branch. Since the private branch is longer than the public branch by one block, it is adopted by all miners as the main branch, and the pool enjoys the revenue of all its blocks. This brings the system back to a state where there is just a single branch until the pool bifurcates it again.

4 Analysis

We can now analyze the expected rewards for a system where the selfish pool has mining power of α and the others of $(1-\alpha)$.

Figure 1 illustrates the progress of the system as a state machine. The states of the system represent the lead of the selfish pool; that is, the difference between the number of unpublished blocks in the pool's private branch and the length of the public branch. Zero lead is separated to states 0 and 0'. State 0 is the state where there are no branches; that is, there is only a single, global, public longest chain. State 0' is the state where there are two public branches of length one: the main branch, and the branch that was private to the selfish miners, and published to match the main branch. The transitions in the figure correspond to mining events, either by the selfish pool or by the others. Recall that these events occur at exponential intervals with an average frequency of α and $(1-\alpha)$, respectively.

We can analyze the expected rewards from selfish mining by taking into account the frequencies associated with each state transition of the state machine, and calculating the corresponding rewards. Let us go through the various cases and describe the associated events that trigger state transitions.

If the pool has a private branch of length 1 and the others mine one block, the pool publishes its branch immediately, which results in two public branches of length 1. Miners in the selfish pool all mine on the pool's branch, because a subsequent block discovery on this branch will yield a reward for the pool. The honest miners, following the standard Bitcoin protocol implementation, mine on the branch they heard of first. We denote by γ the ratio of honest miners that choose to mine on the pool's block, and the other $(1-\gamma)$ of the non-pool miners mine on the other branch.

For state $s = 0, 1, 2, \ldots$, with frequency α, the pool mines a block and the lead increases by one to $s+1$. In states $s = 3, 4, \ldots$, with frequency $(1-\alpha)$, the honest miners mine a block and the lead decreases by one to $s-1$. If the others mine a block when the lead is two, the pool publishes its private branch, and the system drops to a lead of 0. If the others mine a block with the lead is 1, we arrive at the aforementioned state 0'. From 0', there are three possible transitions, all leading to state 0 with total frequency 1: (1) the pool mines a block on its previously private branch (frequency α), (2) the others mine a block on the previously private branch (frequency $\gamma(1-\alpha)$), and (3) the others mine a block on the public branch (frequency $(1-\gamma)(1-\alpha)$).

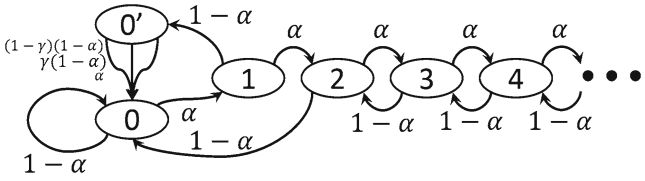


Fig. 1. State machine with transition frequencies.

4.1 State Probabilities

We analyze this state machine to calculate its probability distribution over the state space. We obtain the following equations:

$$\begin{cases} \alpha p_0 = (1-\alpha)p_1 + (1-\alpha)p_2 \\ p_{0'} = (1-\alpha)p_1 \\ \alpha p_1 = (1-\alpha)p_2 \\ \forall k \geq 2 : \alpha p_k = (1-\alpha)p_{k+1} \\ \sum_{k=0}^{\infty} p_k + p_{0'} = 1 \end{cases} \tag{1}$$

Solving (1) (See our full report for details [14]), we get:

$$p_0 = \frac{\alpha - 2\alpha^2}{\alpha(2\alpha^3 - 4\alpha^2 + 1)} \tag{2}$$

$$p_{0'} = \frac{(1-\alpha)(\alpha - 2\alpha^2)}{1 - 4\alpha^2 + 2\alpha^3} \tag{3}$$

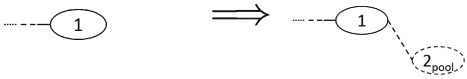
$$p_1 = \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1} \tag{4}$$

$$\forall k \geq 2 : p_k = \left(\frac{\alpha}{1-\alpha}\right)^{k-1} \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1} \tag{5}$$

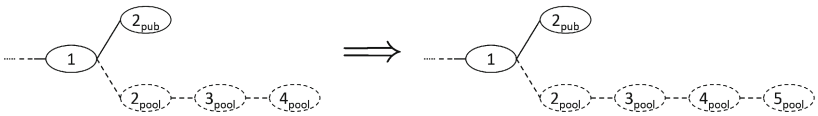
4.2 Revenue

The probability distribution over the state space provides the foundation for analyzing the revenue obtained by the selfish pool and by the honest miners. The revenue for finding a block belongs to its miner only if this block ends up in the main chain. We detail the revenues on each event below.

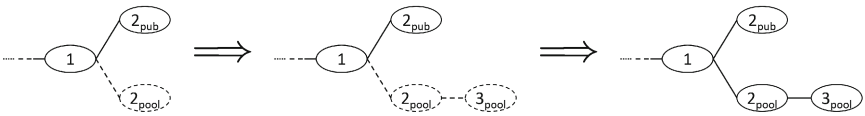
(a) *Any state but two branches of length 1, pools finds a block.* The pool appends one block to its private branch, increasing its lead on the public branch by one. The revenue from this block will be determined later.



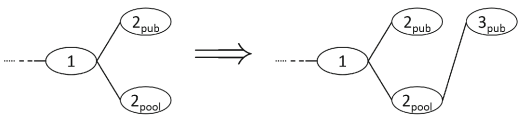
or



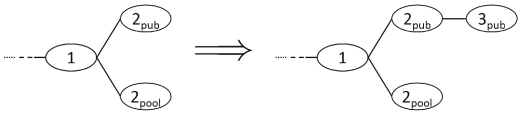
(b) *Was two branches of length 1, pools finds a block.* The pool publishes its secret branch of length two, thus obtaining a revenue of two.



(c) *Was two branches of length 1, others find a block after pool head.* The pool and the others obtain a revenue of one each — the others for the new head, the pool for its predecessor.

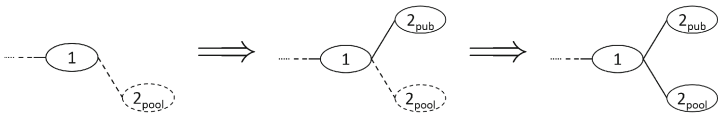


(d) *Was two branches of length 1, others find a block after others' head.* The others obtain a revenue of two.

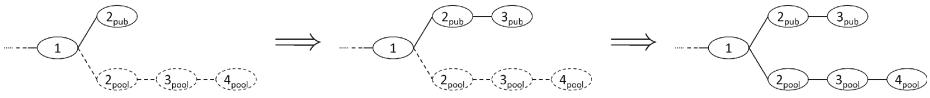


(e) *No private branch, others find a block.* The others obtain a revenue of one, and both the pool and the others start mining on the new head.

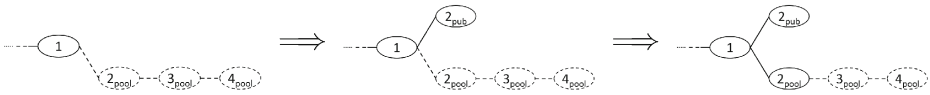
(f) *Lead was 1, others find a block.* Now there are two branches of length one, and the pool publishes its single secret block. The pool tries to mine on its previously private head, and the others split between the two heads. Denote by γ the ratio of others that choose the non-pool block.
The revenue from this block cannot be determined yet, because it depends on which branch will win. It will be counted later.



(g) *Lead was 2, others find a block.* The others almost close the gap as the lead drops to 1. The pool publishes its secret blocks, causing everybody to start mining at the head of the previously private branch, since it is longer. The pool obtains a revenue of two.



(h) *Lead was more than 2, others win.* The others decrease the lead, which remains at least two. The new block (say with number i) will end outside the chain once the pool publishes its entire branch, therefore the others obtain nothing. However, the pool now reveals its i'th block, and obtains a revenue of one.



We calculate the revenue of the pool and of the others from the state probabilities and transition frequencies:

$$r_{\text{others}} = \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_{0'} \cdot (1-\gamma)(1-\alpha) \cdot 2}^{\text{Case (d)}} + \overbrace{p_0 \cdot (1-\alpha) \cdot 1}^{\text{Case (e)}} \tag{6}$$

$$r_{\text{pool}} = \overbrace{p_{0'} \cdot \alpha \cdot 2}^{\text{Case (b)}} + \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_2 \cdot (1-\alpha) \cdot 2}^{\text{Case (g)}} + \overbrace{P[i>2](1-\alpha) \cdot 1}^{\text{Case (h)}} \tag{7}$$

As expected, the intentional branching brought on by selfish mining leads the honest miners to work on blocks that end up outside the blockchain. This, in turn, leads to a drop in the total block generation rate with $r_{\text{pool}}+r_{\text{others}} < 1$. The protocol will adapt the mining difficulty such that the mining rate at the main chain becomes one block per 10 min on average. Therefore, the actual revenue rate of each agent is the *revenue rate ratio*; that is, the ratio of its blocks out of the blocks in the main chain. We substitute the probabilities from (2)–(5) in the revenue expressions of (6)–(7) to calculate the pool's revenue for $0 \leq \alpha \leq \frac{1}{2}$:

$$R_{\text{pool}} = \frac{r_{\text{pool}}}{r_{\text{pool}} + r_{\text{others}}} = \cdots = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)} \,. \tag{8}$$

4.3 Simulation

To validate our theoretical analysis, we compare its result with a Bitcoin protocol simulator. The simulator is constructed to capture all the salient Bitcoin mining protocol details described in previous sections, except for the cryptopuzzle module that has been replaced by a Monte Carlo simulator that simulates block discovery without actually computing a cryptopuzzle. In this experiment, we use the simulator to simulate 1000 miners mining at identical rates. A subset of 1000α miners form a pool running the Selfish-Mine algorithm. The other miners follow the Bitcoin protocol. We assume block propagation time is negligible compared to mining time, as is the case in reality. In the case of two branches of the same length, we artificially divide the non-pool miners such that a ratio of γ of them mine on the pool's branch and the rest mine on the other branch. Figure 2 shows that the simulation results match the theoretical analysis.

4.4 The Effect of α and γ

When the pool's revenue given in Eq. 8 is larger than α, the pool will earn more than its relative size by using the Selfish-Mine strategy. Its miners will therefore earn more than their relative mining power. Recall that the expression is valid only for $0 \leq \alpha \leq \frac{1}{2}$. We solve this inequality and phrase the result in the following observation:

Observation 1. *For a given γ, a pool of size α obtains a revenue larger than its relative size for α in the following range:*

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}. \tag{9}$$

We illustrate this in Fig. 2, where we see the pool's revenue for different γ values with pool size ranging from 0 (very small pool) to 0.5 (half of the miners). Note that the pool is only at risk when it holds exactly one block secret, and the honest miners might publish a block that would compete with it. For $\gamma = 1$, the pool can quickly propagate its one-block branch if the others find their own branch, so all honest miners would still mine on the pool's block. In this case, the pool takes no risk when following the Selfish-Mine strategy and its revenue is always better than when following the honest algorithm. The threshold is therefore zero, and a pool of any size can benefit by following Selfish-Mine. In the other extreme, $\gamma = 0$, the honest miners always publish and propagate their block first, and the threshold is at $1/3$. With $\gamma = 1/2$ the threshold is at $1/4$. Figure 3 shows the threshold as a function of γ.

We also note that the slope of the pool revenue, R_{pool}, as a function of the pool size is larger than one above the threshold. This implies the following observation:

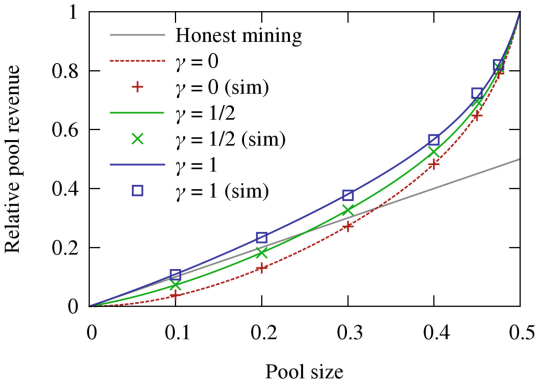


Fig. 2. Pool revenue using the Selfish-Mine strategy for different propagation factors γ, compared to the honest Bitcoin mining protocol. Simulation matches the theoretical analysis, and both show that Selfish-Mine results in higher revenues than the honest protocol above a threshold, which depends on γ.

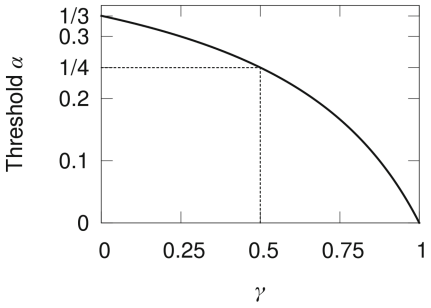


Fig. 3. For a given γ, the threshold α shows the minimum power selfish mining pool that will trump the honest protocol. The current Bitcoin protocol allows $\gamma = 1$, where Selfish-Mine is always superior. Even under unrealistically favorable assumptions, the threshold is never below 1/3.

Observation 2. *For a pool running the Selfish-Mine strategy, the revenue of each pool member increases with pool size for pools larger than the threshold.*

5 Pool Formation

We have shown that once a selfish pool's mining power exceeds the threshold, it can increase its revenue by running Selfish-Mine (Theorem 1). At this point, rational miners will preferentially join the selfish pool to increase their revenues. Moreover, the pool's members will want to accept new members, as this would increase their own revenue (Observation 2). The selfish pool would therefore increase in size, unopposed by any mechanism, towards a majority. Once a miner pool, selfish or otherwise, reaches a majority, it controls the blockchain. The Selfish-Mine strategy then becomes unnecessary, since the others are no longer faster than the pool. Instead, a majority pool can collect all the system's revenue by switching to a modified Bitcoin protocol that ignores blocks generated outside the pool; it also has no motivation to accept new members. At this point, the currency is not a decentralized currency as originally envisioned.

6 Hardening the Bitcoin Protocol

Ideally, a robust currency system would be designed to resist attacks by groups of colluding miners. Since selfish mining attacks yield positive outcomes for group sizes above the threshold, the protocol should be amended to set the threshold as high as possible. In this section, we argue that the current Bitcoin protocol has no measures to guarantee a low γ. This implies that the threshold may be as low as zero, and a pool of any size can benefit by running Selfish-Mine. We suggest a simple change to the protocol that, if adopted by all non-selfish miners, sets γ to 1/2, and therefore the threshold to 1/4. This change is backward

compatible; that is, any subset of the miners can adopt it without hindering the protocol. Moreover, it is progressive; that is, any ratio of the miners that adopts it decreases γ, and therefore increases the threshold.

6.1 Problem

The Bitcoin protocol prescribes that when a miner knows of multiple branches of the same length, it mines and propagates only the first branch it received. Recall that a pool that runs the Selfish-Mine strategy and has a lead of 1 publishes its secret block P once it hears of a competing block X found by a non-pool block. If block P reaches a non-pool miner before block X, that miner will mine on P.

Because selfish mining is reactive, and it springs into action only after the honest nodes have discovered a block X, it may seem to be at a disadvantage. But a savvy pool operator can perform a sybil attack on honest miners by adding a significant number of zero-power miners to the Bitcoin miner network. These virtual miners act as advance sensors by participating in data dissemination, but do not mine new blocks. (Babaioff et al. also acknowledge the feasibility of such a sybil attack [4]). The virtual miners are managed by the pool, and once they hear of block X, they ignore it and start propagating block P. The random peer-to-peer structure of the Bitcoin overlay network will eventually propagate X to all miners, but the propagation of X under these conditions will be strictly slower than that of block P. By adding enough virtual nodes, the pool operator can thus increase γ. The result, as shown in Eq. 9, is a threshold close to zero.

6.2 Solution

We propose a simple, backwards-compatible change to the Bitcoin protocol to address this problem and raise the threshold. Specifically, when a miner learns of competing branches of the same length, it should propagate all of them, and choose which one to mine on uniformly at random. In the case of two branches of length 1, as discussed in Sect. 4, this would result in half the nodes (in expectancy) mining on the pool's branch and the other half mining on the other branch. This yields $\gamma = 1/2$, which in turn yields a threshold of $1/4$.

Each miner implementing our change decreases the selfish pool's ability to increase γ through control of data propagation. This improvement is independent of the adoption of the change at other miners, therefore it does not require a hard fork. This change to the protocol does not introduce new vulnerabilities to the protocol: Currently, when there are two branches of equal length, the choice of each miner is arbitrary, effectively determined by the network topology and latency. Our change explicitly randomizes this arbitrary choice, and therefore does not introduce new vulnerabilities.

7 Related Work

Decentralized digital currencies have been proposed before Bitcoin, starting with [11] and followed by peer-to-peer currencies [32,34]; see [5,22] for short

surveys. None of these are centered around a global log; therefore, their techniques and challenges are unrelated to this work.

Several dozen cryptocurrencies have followed Bitcoin's success [17,18,33], most prominently Litecoin [21]. These currencies are based on a global log, which is extended by the users' efforts. We conjecture that the essential technique of withholding blocks for selfish mining can be directly applied to all such systems.

It was commonly believed that the Bitcoin system is sound as long as a majority of the participants honestly follow the protocol, and the "51 % attack" was the chief concern [1,20,23]. The notion of soundness for a nascent, distributed, Internet-wide, decentralized system implies the presence of incentives for adoption of the prescribed protocol, for such incentives ensure a robust system comprised of participants other than enthusiastic and altruistic early adopters. Felten [15] notes that "there was a folk theorem that the Bitcoin system was stable, in the sense that if everyone acted according to their incentives, the inevitable result would be that everyone followed the rules of Bitcoin as written." Others [25] have claimed that "the well-known argument – never proven, but taken on intuitive faith – that a minority of miners can't control the network is a special case of a more general assumption: that a coalition of miners with $X\%$ of the network's hash power can make no more than $X\%$ of total mining revenues." A survey [5] on the technical features responsible for Bitcoin's success notes that the Bitcoin design "addresses the incentive problems most expeditiously," while Bitcoin tutorials for the general public hint at incentives designed to align participants' and the system's goals [27]. More formally, Kroll, Davey and Felten's work [19] provides a game-theoretic analysis of Bitcoin, without taking into account block withholding attacks such as selfish mining, and argues that the honest strategy constitutes a Nash equilibrium, implying incentive-compatibility.

Our work shows that the real Bitcoin protocol, which permits block withholding and thereby enables selfish mining-style attacks, does not constitute an equilibrium. It demonstrates that the Bitcoin mining system is not incentive compatible even in the presence of an honest majority. Over 2/3 of the participants need to be honest to protect against selfish mining, under the most optimistic of assumptions.

A distinct exception from this common wisdom is a discussion of maintaining a secret fork in the Bitcoin forums, mostly by users[3] btchris, ByteCoin, mtgox, and RHorning [28]. The approach, dubbed the Mining Cartel Attack, is inferior to selfish mining in that the cartel publishes two blocks for every block published by the honest nodes. This discussion does not include an analysis of the attack (apart from a brief note on simulation results), does not explore the danger of the resulting pool dynamics, and does not suggest a solution to the problem.

The influential work of Rosenfeld [30] addresses the behavior of miners in the presence of different pools with different rewards. Although it addresses revenue maximization for miners with a set mining power, this work is orthogonal to the discussion of Selfish Mining, as it centers around the pool reward system.

[3] In alphabetical order.

Both selfish pools and honest pools should carefully choose their reward method. Since a large-enough selfish pool would earn more than its mining power, any fair reward method would provide more reward to its miners, so rational miners would choose it over an honest pool.

Recent work [4] addresses the lack of incentives for disseminating transactions between miners, since each of them prefers to collect the transaction fee himself. This is unrelated to the mining incentive mechanism we discuss.

A widely cited study [29] examines the Bitcoin transaction graph to analyze client behavior. The analysis of client behavior is not directly related to our work.

The Bitcoin blockchain had one significant bifurcation in March 2013 due to a bug [2]. It was solved when the two largest pools at the time manually pruned one branch. This bug-induced fork, and the one-off mechanism used to resolve it, are fundamentally different from the intentional forks that Selfish-Mine exploits.

In a *block withholding attack*, a pool member decreases the pool revenue by never publishing blocks it finds. Although it sounds similar to the strategy of Selfish-Mine, the two are unrelated, as our work that deals with an attack by the pool on the system.

Various systems build services on top of the Bitcoin global log, e.g., improved coin anonymity [22], namespace maintenance [24] and virtual notaries [3,16]. These services that rely on Bitcoin are at risk in case of a Bitcoin collapse.

8 Discussion

We briefly discuss below several points at the periphery of our scope.

System Collapse. The Bitcoin protocol is designed explicitly to be decentralized. We therefore refer to a state in which a single entity controls the entire currency system as a collapse of Bitcoin.

Note that such a collapse does not immediately imply that the value of a Bitcoin drops to 0. The controlling entity will have an incentive to accept most transactions, if only to reap their fees, and because if it mines all Bitcoins, it has strong motivation that they maintain their value. It may also choose to remain covert, and hide the fact that it can control the entire currency. An analysis of a Bitcoin monopolist's behavior is beyond the scope of this paper, but we believe that a currency that is de facto or potentially controlled by a single entity may deter many of Bitcoin's clients.

Detecting Selfish Mining. There are two telltale network signatures of selfish mining that can be used to detect when selfish mining is taking place, but neither are easy to measure definitively.

The first and strongest sign is that of abandoned (orphaned) chains, where the block race that takes place as part of selfish mining leaves behind blocks that were not incorporated into the blockchain. Unfortunately, it is difficult to definitively account for abandoned blocks, as the current protocol prunes and discards such blocks inside the network. A measurement tool that connects to the network from a small number of vantage points may miss abandoned blocks.

The second indicator of selfish mining activity is the timing gap between successive blocks. A selfish miner who squelches an honest chain of length N with a chain of length $N + 1$ will reveal a block very soon after its predecessor. Since normal mining events should be independent, one would expect block discovery times to be exponentially distributed. A deviation from this distribution would be suggestive of mining activity. The problems with this approach are that it detects only a subset of the selfish miner's behavior (the transition from state 2 to state 0 in the state machine), the signature behavior occurs relatively rarely, and such a statistical detector may take a long time to accrue statistically significant data.

Measures and Countermeasures. Although miners may choose to collude in a selfish mining effort, they may prefer to hide it in order to avoid public criticism and countermeasures. It is easy to hide Selfish-Mine behavior, and difficult to ban it. A selfish pool may never reveal its size by using different Bitcoin addresses and IP addresses, and by faking block creation times. The rest of the network would not even suspect that a pool is near a dangerous threshold.

Moreover, the honest protocol is public, so if a detection mechanism is set up, a selfish pool would know its parameters and use them to avoid detection. For instance, if the protocol was defined to reject blocks with creation time below a certain threshold, the pool could publish its secret blocks just before this threshold.

A possible line of defense against selfish mining pools is for counter-attackers to infiltrate selfish pools and expose their secret blocks for the honest miners. However, selfish pool managers can, in turn, selectively reveal blocks to subsets of the members in the pool, identify spy nodes through intersection, and expel nodes that leak information.

Thieves and Snowballs. Selfish mining poses two kinds of danger to the Bitcoin ecosystem: selfish miners reap disproportionate rewards, and the dynamics favor the growth of selfish mining pools towards a majority, in a snowball effect. The system would be immune to selfish mining if there were no pools above the threshold size. Yet, since the current protocol has no guaranteed lower bound on this threshold, it cannot automatically protect against selfish miners.

Even with our proposed fix that raises the threshold to 25 %, the system remains vulnerable: there already exist pools whose mining power exceeds the 25 % threshold [26], and at times, even the 33 % theoretical hard limit. Responsible miners should therefore break off from large pools until no pool exceeds the threshold size.

Responsible Disclosure. Because of Bitcoin's decentralized nature, selfish mining can only be thwarted by collective, concerted action. There is no central repository, no push mechanism and no set of privileged developers; all protocol modifications require public discussion prior to adoption. In order to promote a swift solution and to avoid a scenario where some set of people had the benefit of selective access, we published a preliminary report [14] and explained both the problem and our suggested solution in public forums [13].

9 Conclusion

Bitcoin is the first widely popular cryptocurrency with a broad user base and a rich ecosystem, all hinging on the incentives in place to maintain the critical Bitcoin blockchain. Our results show that Bitcoin's mining protocol is not incentive-compatible. We presented Selfish-Mine, a mining strategy that enables pools of colluding miners that adopt it to earn revenues in excess of their mining power. Higher revenues can lead new miners to join a selfish miner pool, a dangerous dynamic that enables the selfish mining pool to grow towards a majority. The Bitcoin system would be much more robust if it were to adopt an automated mechanism that can thwart selfish miners. We offer a backwards-compatible modification to Bitcoin that ensures that pools smaller than 1/4 of the total mining power cannot profitably engage selfish mining. We also show that at least 2/3 of the network needs to be honest to thwart selfish mining; a simple majority is not enough.

Acknowledgements. We are grateful to Raphael Rom, Fred B. Schneider, Eva Tardos, and Dror Kronstein for their valuable advice on drafts of this paper, as well as our shepherd Rainer Böhme for his guidance.

References

1. andes: Bitcoin's kryptonite: the 51% attack, June 2011. https://bitcointalk.org/index.php?topic=12435
2. Andresen, G.: March 2013 chain fork post-mortem. BIP 50. https://en.bitcoin.it/wiki/BIP_50. Accessed September 2013
3. Araoz, M.: Proof of existence. http://www.proofofexistence.com/. Accessed September 2013
4. Babaioff, M., Dobzinski, S., Oren, S., Zohar, A.: On Bitcoin and red balloons. In: ACM Conference on Electronic Commerce, pp. 56–73 (2012)
5. Barber, S., Boyen, X., Shi, E., Uzun, E.: Bitter to better — how to make Bitcoin a better currency. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 399–414. Springer, Heidelberg (2012)
6. Bitcoin community: Bitcoin source. https://github.com/bitcoin/bitcoin. Accessed September 2013
7. Bitcoin community: protocol rules. https://en.bitcoin.it/wiki/Protocol_rules. Accessed September 2013
8. Bitcoin community: protocol specification. https://en.bitcoin.it/wiki/Protocol_specification. Accessed September 2013
9. bitcoincharts.com: Bitcoin network. http://bitcoincharts.com/bitcoin/. Accessed November 2013
10. blockchain.info: Bitcoin market capitalization. http://blockchain.info/charts/market-cap. Accessed January 2014
11. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO, vol. 82, pp. 199–203 (1982)
12. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: IEEE P2P (2013)

13. Eyal, I., Sirer, E.G.: Bitcoin is broken (2013). http://hackingdistributed.com/2013/11/04/bitcoin-is-broken/
14. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable (2013). arXiv preprint arXiv:1311.0243
15. Felten, E.W.: Bitcoin research in Princeton CS, November 2013. https://freedom-to-tinker.com/blog/felten/bitcoin-research-in-princeton-cs/
16. Kelkar, A., Bernard, J., Joshi, S., Premkumar, S., Sirer, E.G.: Virtual notary. http://virtual-notary.org/. Accessed September 2013
17. King, S.: Primecoin: cryptocurrency with prime number proof-of-work (2013). http://primecoin.org/static/primecoin-paper.pdf
18. King, S., Nadal, S.: PPCoin: peer-to-peer crypto-currency with proof-of-stake (2012). https://archive.org/details/PPCoinPaper
19. Kroll, J.A., Davey, I.C., Felten, E.W.: The economics of Bitcoin mining or, Bitcoin in the presence of adversaries. In: Workshop on the Economics of Information Security (2013)
20. Lee, T.B.: Four reasons Bitcoin is worth studying, April 2013. http://www.forbes.com/sites/timothylee/2013/04/07/four-reasons-bitcoin-is-worth-studying/2/
21. Litecoin Project: Litecoin, open source P2P digital currency. https://litecoin.org. Accessed September 2013
22. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from Bitcoin. In: IEEE Symposium on Security and Privacy (2013)
23. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
24. Namecoin Project: Namecoin DNS - DotBIT project. https://dot-bit.org. Accessed September 2013
25. Narayanan, A., Miller, A.: Why the Cornell paper on Bitcoin mining is important, November 2013. https://freedom-to-tinker.com/blog/randomwalker/why-the-cornell-paper-on-bitcoin-mining-is-important/
26. Neighborhood Pool Watch: October 27th 2013 weekly pool and network statistics. http://organofcorti.blogspot.com/2013/10/october-27th-2013-weekly-pool-and.html. Accessed October 2013
27. Pacia, C.: Bitcoin mining explained like you're five: part 1 - incentives, September 2013. http://chrispacia.wordpress.com/2013/09/02/bitcoin-mining-explained-like-youre-five-part-1-incentives/
28. RHorning, mtgox, btchris, ByteCoin: mining cartel attack, December 2010. https://bitcointalk.org/index.php?topic=2227
29. Ron, D., Shamir, A.: Quantitative analysis of the full Bitcoin transaction graph. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 6–24. Springer, Heidelberg (2013)
30. Rosenfeld, M.: Analysis of Bitcoin pooled mining reward systems (2011). arXiv preprint arXiv:1112.4980
31. Swanson, E.: Bitcoin mining calculator. http://www.alloscomp.com/bitcoin/calculator. Accessed September 2013
32. Vishnumurthy, V., Chandrakumar, S., Sirer, E.G.: Karma: a secure economic framework for peer-to-peer resource sharing. In: Workshop on Economics of Peer-to-Peer Systems (2003)
33. Wikipedia: List of cryptocurrencies. https://en.wikipedia.org/wiki/List_of_cryptocurrencies. Accessed October 2013
34. Yang, B., Garcia-Molina, H.: PPay: micropayments for peer-to-peer systems. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 300–310. ACM (2003)