One Weird Trick to Stop Selfish Miners: Fresh Bitcoins, A Solution for the Honest Miner.

Ethan Heilman
Boston University
heilman@bu.edu

***Abstract*—A recent result in Bitcoin is the selfish mining strategy in which a selfish cartel withholds blocks they mine to gain an advantage. This strategy is both incentive-compatible and harmful to Bitcoin. In this paper we introduce a new defense against selfish mining that improves on the previous best result, we raise the threshold of mining power necessary to profitably selfishly mine from 25% to 32% under all propagation advantages. While the security of our system uses unforgeable timestamps, it is robust to their compromise. Additionally, we discuss the difficulty a mining conspiracy would face attempting to keep the compromise of our scheme secret and we analyze incentives for getting miners to adopt these changes.**

I. Introduction

Bitcoin is a virtual currency [8], [3] that has rapidly increased in user share and value. As of November 2013 it has a market capitalization of 13 billion USD [6]. In the last year it has undergone rapid commercial deployment in the online retail [4] and international remittances sector [5].

In "Majority is not Enough: Bitcoin Mining is Vulnerable", Eyal and Sirer study a mining strategy called selfish mining [14] (see Subsection II). Miners are members of the Bitcoin network that securely process collections of transactions into blocks in exchange for rewards (see Section II-A). This strategy defeats the incentive system of Bitcoin, because a mining pool which follows the selfish protocol can obtain a greater share of mining rewards than those miners which follow the Bitcoin protocol. Taken as a whole, selfish mining represents a 'tragedy of the commons' in which selfish behavior is incentivized over honest behavior, eventually causing most actors to adopt the selfish behavior despite it not being in the group's global interest (see Subsection II-F).

The success of selfish mining depends on two parameters: α, the mining power of the selfish cartel and γ, the ratio of honest mining power that, during a block race, mines on a block released by the selfish cartel. Mining power is the percentage of computational power that a particular miner or mining pool controls out of the total computational power of all the miners. We can view the minimum value of α such that selfish mining is successful as the security threshold for a particular γ. The greater the value γ, the smaller α needs to be and vice versa. Thus, as Eyal and Sirer show, if $\gamma = 0$, then selfish mining is profitable at $\alpha \geq 0.33$ or 33%, whereas if $\gamma = 0.99$ then selfish mining is profitable at $\alpha \geq 0.009$. Eyal and Sirer propose a defense against selfish mining which fixes $\gamma = 0.5$. This raises the threshold for a selfish cartel to be profitable to at least 25% or $\alpha \geq 0.25$ (see Subsection II-G).

In Section III we present and analyze a novel defense, called Freshness Preferred, which raises the security bound to 32%. As selfish mining is based on strategic withholding of blocks, Freshness Preferred decreases the profitability of selfish mining by using unforgeable timestamps to penalize miners that withhold blocks. Under our scheme γ depends on α, therefore we are unable to fix γ for all α as was done by Eyal and Sirer. Instead we show that for all $\alpha \leq 0.32$, γ is low enough such that selfish mining does not win more than its fair share.

The remainder of the paper is organised as follows. In Subsection III-C we analyze the security of our scheme under partial compromise. In Subsection III-F we look at the incentizes of deploying Freshness Preferred and develop an incentive compatible deployment plan. In Section IV we examine a necessary component of Freshness Preferred, unforgeable timestamps.

II. Selfish Mining Background

A. Bitcoin

Bitcoin is collection of cryptographic protocols which allows users to securely transact with each other.

These transactions are recorded in a distributed public ledger known as the block chain. To ensure the irreversibility of the public ledger, Bitcoin miners process transactions by discovering new blocks to place at the head of the blockchain. This act of block discovery is known as mining and is done by performing a large number of simple but intense computations.

B. Block Races

When a block is discovered it is transmitted to neighbors in a p2p overlay network. If this block extends a miner's blockchain, the miner will begin mining on the announced block and retransmit the block to its neighbors.

A block race occurs when two blocks with the same parent block are announced at roughly the same time. According to the Bitcoin protocol, miners should always choose the block they received first, and not retransmit the second block. Since the blocks were announced at roughly the same time, some miners will see one block first and other miners will see the other block first. This causes the blockchain to fork into two branches.

A block race continues until a miner discovers a block which extends one of the competing branches. Because miners that follow the Bitcoin protocol always prefer the longest blockchain, the branch that was just extended becomes the canonical branch for all miners.

C. A History of Selfish Mining Strategies

An adversarial mining strategy based on withholding blocks was first proposed in 2010 by Bitcoin talk member RHorning [12], further analysis was done by ByteCoin showing that, all other things being equal, the attack is successful if the selfish mining cartel controls 33% of the mining power [13]. This 33% threshold assumed that the cartel won half of all block races.

In 2013 Eyal and Sirer published a deep and formal analysis of the incentives of the selfish mining strategy. They show two interesting results. First, if the selfish cartel can gain an advantage in block races, then the selfish cartel would need to control very little mining power to win more than its fair share of mining rewards[14]. Second, that with control of 33% of the mining power, or $\alpha = 0.33$, a selfish cartel will win more than its fair share of the mining rewards, even if honest miners never mine on a selfish block during a block race (that is, $\gamma = 0$).

D. Selfish Mining Assumptions

Eyal and Sirer's selfish mining scenario assumes that there are only two camps of miners: a single cartel of selfish miners and the community of honest miners. The selfish cartel functions as a single coordinated group, the honest community is not coordinated and consists of many actors. The honest community is assumed to represent over 50% of the Bitcoin mining power.

This may not be an accurate description of how selfish mining may play out, for instance there could be multiple selfish cartels competing with each other[1], various miners could play both sides or the honest community could coordinate a response to selfish mining. We will use this set of assumptions in our paper.

E. A Description of the Selfish-Mine Strategy

In the selfish mining strategy introduced by Eyal and Sirer, the share of the mining resources (computing power) controlled by the cartel is denoted by α and the share of the resources owned by the honest miners is $(1-\alpha)$. This strategy relies on creating an informational asymmetry between the cartel's private branch and the honest miners' public branch.

Formally, the strategy Selfish-Mine is played according to the following five rules:

Event: The Honest community discovers a block.
1: If the public branch of the blockchain is longer than the private branch of the cartel, set the private branch equal to the public branch.
2: If the private branch is zero or one block longer than the public branch, publish the entire private branch[2].
3: If the private branch is more than one block longer than the public branch publish the first unpublished block in the public chain.

Event: The Selfish cartel discovers a block.
4: Add the new block to the private branch.
5: If a block race is occurring between the cartel and the honest miners, publish the private branch to win the race.

Thus the cartel does not publish blocks from its private branch except in reaction to the honest miners discovering a block. The Selfish-Mine strategy is said to be successful when the cartel wins more than its fair share of mining rewards.

This success depends on the parameters α and γ. During a block race between the selfish cartel and an honest miner, members of the honest community may learn about the block mined by the selfish cartel before they learn about the block mined by the honest miner, thus some percentage of the honest mining community will be mining on the selfish cartels block. γ is the percentage of honest mining power that mines on a block released by the selfish cartel. Eyal and Sirer relate α and γ using a formula they refer to as Observation 1[14]. The observation is that "a pool of size α obtains a revenue larger than its relative size for α in the following range":

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

This observation shows that as γ increases, the mining resource threshold for selfish mining decreases, but as γ decreases, the threshold increases, raising the difficulty of selfish mining.

F. Selfish Mining Is Harmful

Selfish mining, if put into practice, would be extremely harmful for Bitcoin. As Eyal and Sirer[14] argue, if selfish mining became a more profitable mining strategy, then increasing numbers of miners would join the selfish cartel. This could lead to a situation in which the cartel controls over 50% of the mining resources and thereby centralizes Bitcoin (known as the 51% attack) [17], [1].

Furthermore a selfish mining cartel with significant mining resources would cause two negative effects. First, it would increase transaction approval times, because transactions approved by the selfish private branch would not be public and, due an increased number of block races, transactions approved in blocks that lost a block race would need to be reapproved in a later block. Second, it would make Bitcoin more vulnerable to double spending, as both cartels and non-aligned users could add mutually exclusive transactions to the private and public branches [20].

[1]In simple incentive terms it would be in selfish miners interests to pool their resources, but coordination and secrecy dynamics could lead to multiple cartels.

[2]If there were more than one selfish cartel this would lead to a block "cook off", or chain reaction, as each cartel's block publication would trigger a block publication from the other cartels, until only the cartel with the longest private branch had any private branches remaining. This suggests that selfish mining might be unprofitable with many uncoordinated cartels.

G. The 25% Defense

Eyal and Sirer propose [14] changing the Bitcoin protocol to raise threshold of the minimum successful mining resource share to 25% for arbitrary propagation advantages. Their proposed change is that when a miner learns of two branches of equal length, as occurs during a block race, the miner should retransmit both branches and randomly choose one branch to mine on. They use Observation 1 to show that this raises the threshold of the minimum successful mining resource share to 25% or $\alpha = 0.25$.

III. Freshness Preferred (FP) :The 32% Defense

In this section we present Freshness Preferred (FP), our new mining strategy designed to defend against selfish mining. As selfish mining is based on strategic withholding of blocks, Freshness Preferred decreases the profitability of selfish mining by using unforgeable timestamps to penalize miners that withhold blocks. In Subsection III-B we develop our scheme under the assumption that we have unforgeable timestamps. In Subsection III-C we analyze our scheme when we don't have unforgeable timestamps.

We discuss the design of these unforgeable timestamps in detail in Section IV. Finally in Subsection III-E, we show that Freshness Preferred raises the threshold of the minimum share of mining power necessary to profitably selfishly mine from 25% to 32%. In Subsection III-F we develop a incentive-compatible deployment plan for Freshness Preferred.

A. Description

A miner that plays the FP (Freshness Preferred) strategy uses the following rules:

Event: FP miner receives two blocks within w seconds of each other.

1: If the two blocks are come from branches of equal length the miner accepts the block with the most recent valid timestamp[3] and rejects the other block. If both blocks have equal timestamps the miner prefers the block it received first.

2: Otherwise, the miner accepts the block from the branch of greater length.

All Other Cases:

3: The miner behaves according to the Bitcoin protocol.

That is, we change the Bitcoin protocol so that, rather than a miner preferring the block which arrives earliest, the FP-miner prefers the block which has the most recent timestamp.

The intuition behind FP is if we ensure that block races are won by the party that has the most recently created blocks, then withholding blocks reduces the percentage of honest miners that will mine on the withheld block. The percentage of honest miners that mine on a selfish block during a block race is denoted as, γ. Using Observation 1, made by Eyal and Sirer and discussed in Subsection II-G, reducing γ increases the threshold for selfish mining to be successful.

[3]The lack of a timestamp is treated as an invalid timestamp

B. Modeling a Block Race

We model a block race between a selfish block and an honest block. We assume in this section that the entire honest community has adopted the FP strategy. The selfish block, B_s, is discovered at time D_s, at some later time, D_h, an honest miner discovers the block B_h. The selfish cartel reacts to the publication of B_h and releases B_s in response. We evaluate this block race from the perspective of a FP miner who learns about B_s at time L_s and D_h at time L_h.

Using the heuristic that the we should "overestimate the attacker and underestimate the defender", we assume the cartel has no propagation delay, that it learns about the honest block at discovery time D_h (this is, instantly), and that the honest miner has a lengthy propagation delay of pd_h. Furthermore, we allow the cartel to win all timestamp ties.

Under these assumptions the FP-miner learns about B_s the instant that B_h is discovered, but doesn't learn about B_h until sometime later due to the propagation delay.

$$L_s = D_h$$

$$L_h = D_h + pd_h$$

A block race can only occur when both the blocks are released in the same window of time. We formalize this notion with the parameter w, the block race window. If two blocks are released more than w seconds apart, the first block released, regardless of timestamp, will always win the block race (this is a direct result of our Freshness Preferred Strategy given in Subsection III-A).

As justified in Subsection III-D, we assume that the block race window, w, is larger than the propagation delay pd_h. The difference between the time a miner learns about B_h and B_s will never be greater than w, or put another way, $L_h < L_s + w$. Thus, regardless of when B_h happens, we will always be within the block race window, w. Therefore, as described in Subsection III-A, the FP-miner will decide which block to accept based on the timestamps of the two competing blocks. If the timestamp, T_s, for the selfish block is older than the timestamp for the honest block, T_h, then the FP-miner will choose the honest block. On the other hand, if the timestamps are equal, $T_s = T_h$, then it will prefer the selfish block since we allow the selfish miner to win all ties. By definition the selfish block was discovered prior to the honest block, so the selfish block can never have a more recent timestamp than the honest block.

The probability that a particular selfish block is accepted by the FP-miner is equal to the probability that a honest miner discovers B_h within the same timestamp as B_s. This depends on two factors: the increment of the timestamp in seconds and the per second rate at which honest miners discover new bitcoins.

We introduce Eq. 3, which gives us the probability, p_h, that the honest mining community discovers at least one block during a period of t seconds. Eq. 3 is derived from Eq. 2, which determines the probability of an exponentially distributed event[4] occurring during a time period t and at a particular rate. In our case the rate is the block discovery rate of the honest community. By adjusting the difficulty of mining, Bitcoin maintains a total network rate of block discovery of roughly 1 block every 10 minutes or $\frac{1}{600}$ blocks per second [7]. Using the percentage of the mining power under the control of the honest mining community, $(1-\alpha)$, we get an honest mining rate of $\frac{(1-\alpha)}{600}$.

$$\Pr(\text{event}) = 1 - e^{-\text{rate}\times t} \tag{2}$$

$$p_h = 1 - e^{-\frac{(1-\alpha)}{600}\times t} \tag{3}$$

Plugging in the the increment of the timestamp for t, gives the probability that a honest miner will discover a block within the increment. We call this probability p_h. Therefore during a block race, honest miners will only mine on selfish blocks if the honest block was discovered within the increment of the timestamp, in all other cases the honest miners will not mine on the selfish block.

Thus, p_h is the probability that each time a selfish miner discovers a block that block will be mined on by the honest community. That is, p_h of the time $\gamma = 1$ and $(1-p_h)$ of the time $\gamma = 0$. Using p_h we can compute the average γ as $\gamma = p_h$.

C. Forged Timestamps

Using the notation and methods from the previous section we will examine the case in which our unforgeable timestamp scheme is compromised and a selfish mining cartel is able to forge timestamps from the future. We show that the FP strategy is still robust to selfish mining because selfish miners must still commit to a timestamp within the blocks they discover, even if this timestamp is for a time which is in the future.

As before, we are concerned with a block race in which a selfish miner is instantly reacting to the publication of B_h, but in this case the cartel can choose a timestamp, T_s such that $T_s \geq D_s$, thereby allowing the case $T_s > T_h$.

We assume that the cartel can predict the propagation delay for the honest miner, pd_h, and chooses T_s to maximise their chance of winning a block race. That is, the cartel forges a timestamp of $T_s = D_s + w + pd_h$ to ensure that any honest block that reaches the FP-miner prior to T_s will be older than T_s but still within the window w[5].

To model the probability that a FP-miner accepts B_s over B_h, we need to examine two time intervals.

Interval I: $D_s \leq D_h \leq T_s$ The selfish cartel can release B_s at anytime after T_s and still have a more recent timestamp than any honest block discovered between the time D_s and T_s. Due to the propagation delay, pd_h, blocks that would normally be outside the window of B_s are within w. Thus, all honest miners will choose to mine on B_s if B_h is discovered during this interval.

Interval II: $T_s < D_h$ Any honest block discovered after T_s will be preferred to B_s.

We calculate the probability that B_s is accepted by the FP-miners as the probability that D_h is within Interval I using Eq. 3 and setting $t = (w+pd_h)$. As in the previous subsection, this probability is also the value γ, as it determines the percentage of honest miners, under Freshness Preferred, that will adopt B_s

[4]As is standard in Bitcoin mining analysis[1], [2], block discovery is assumed to be exponential.

[5]To see why consider how the success probability for the cartel would change if T_s deviated from $T_s = D_s + w + pd_h$

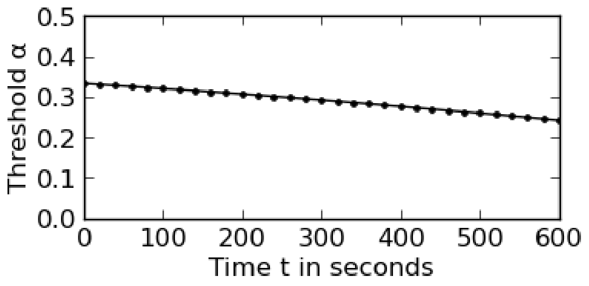

Fig. 1. The value of α necessary for successful selfish mining using Eq. 3 and Observation 1.

D. Choosing Parameters

Based on the timestamp increment used by the NIST Random Beacon[18] (see Subsection IV-A) we adopt an increment of $s = 60$ seconds.

In theory we can choose any value we please for the block race window size w as it is merely used as a filter by FP miners to discard competing blocks that arrive late, but our analysis in Subsection III-C depends on the w being larger than the propagation delay for most of the network. Within Bitcoin the mean propagation time before a node sees a block is 12.6 seconds, and after 40 seconds 95% of the nodes have seen the block[20]. We suggest a conservative value of $w = 120$ seconds.

E. Security of Freshness Preferred

Using these parameters we calculate the security threshold Freshness Preferred offers. In both Subsection III-B and Subsection III-C, we derived γ by determining the appropriate value t for Eq. 3. Combining γ with Observation 1, which we borrow from the “Majority is not Enough: Bitcoin Mining is Vulnerable.” (see Subsection II-E), we calculate the threshold for successful selfish mining. In Figure 1 we graph the relationship between t and the value of α necessary for the selfish cartel to win more its fair share.

In Subsection III-B, we showed that, using Eq. 3, γ depends on the timestamp increment s (this is, $t = s$).

$$\gamma = 1 - e^{-\frac{(1-\alpha)}{600}\times t}$$

In Eq. 4, we plug the equation for γ into Observation 1 to get an equation for the threshold needed for successful selfish mining.

$$\text{threshold needed} = \frac{1-(1-e^{-\frac{(1-\alpha)}{600}\times t})}{3-2\times(1-e^{-\frac{(1-\alpha)}{600}\times t})} \quad (4)$$

Note that the threshold depends on α but that if $\alpha <$ 'threshold needed' the threshold is not met. Thus, we solve for values where $\alpha \geq$ 'threshold needed' to arrive at the threshold (the smallest α that which satifies the threshold needed for the equation). The values of α and t such that they satify the threshold at given in Figure 1. Supplying the timestamp increment, $t = s = 60$, from Subsection III-D, we calculate the threshold for successful selfish mining within FP when $s = 60$ to be 32.5%. This improves on the 25% protection threshold offered by Eyal and Sirer.

Next, we consider the threshold for successful selfish mining within FP, assuming that the selfish cartel has gained the ability to forge timestamps. In Subsection III-C, we showed that, γ depended on the block race window, w, and the propagation delay, pd_h. That is, $t = (w + pd_h)$. Assuming a propagation delay of $pd_h = 100$ seconds, and the block race window $w = 120$, as chosen in Subsection III-D, using Eq. 4, we find that the threshold for successful selfish mining with forgeries is 30%.

F. Incentives for Deploying Freshness Preferred

First we consider the incentives of adding timestamps to blocks. Default miners ignore timestamps, and FP miners prefer them. Adding timestamps is always incentivized as long as some FP miners exist, since timestamps will always improve the chance of winning a block race.

Now we consider the incentives of FP miners preferring blocks with more recent timestamps when they are in the minority. If the default miners significantly outnumber the FP miners, FP miners are at a disadvantage because if there is a block race between default miners and FP miners, the FP miners will likely lose.

To see why, consider the cases of a block race between a block with a recent timestamp and a block with an older timestamp: (1). the more recent block has a higher γ, (2). the older block has a higher γ, (3). both blocks have even γ. In case (1) and (3) FP miners behave exactly the same as default miners, but in case (2) FP miners behavior differs from default miners. In the small chance that an FP miner discovers a block at the same time as the default miners discover a block, the FP miner will likely lose the ensuing block race as most miners will mine on the block discovered by the default miners. Thus, although the disincentive is small, FP miners are not incentive-compatible until they reach over 50% of the honest mining pool.

To solve the incentive problem for FP miners we propose a system for coordinating FP mining only when it is in the miners' best interest. Freshness preferred miners initially use the default block preference behavior (prefer the first block you see), but they always add timestamps. When more than half of the most recent blocks in the blockchain for 30 days include unforgeable timestamps, then FP-miners know that they have reached majority resources with high probability[6]. At which point they switch to FP since it has become incentive-compatible.

G. Slothful Mining

In Subsection III-C we showed the FP mining was robust to forgeable timestamps. This would suggest that perhaps we only use timestamps, not unforgeable timestamps. Unfortunately this is not the case, as timestamps alone, while resistant to selfish mining, enable a new attack we call slothful mining. A slothful miner exploits the fact that if $(w = T_s) > D_s$, that is if the timestamp T_s is greater than the discovery time of the slothful block, the miner can withhold the slothful block for $T_s - D_s$ seconds and still defeat any completing branches of the same height discovered prior to time $D_s + w = T_s$. The advantage of such an attack is constrained by the size of w, but a slothful mining pool of any size wins slightly more than its fair share of resources. We are in the process of analyzing the scope of this attack, but it should not effect FP mining as slothful mining is not possible under unforgeable timestamps.

IV. Unforgeable Timestamps

As shown in Section III, our solution makes use of unforgeable timestamps to ensure a particular block was generated no later than the timestamp.

While the block specification already includes a timestamp[9][10] it is subject to manipulation by network peers performing Timejacking attacks[11]. To categorically rule out any such problems and thereby simplify our analysis, we will assume that the our timestamp is independent from the block timestamp.

We propose a method whereby a miner can prove that a block was mined recently.

Consider a publicly known and verifiable, periodically published, source of random strings such as the NIST beacon [18]. Every s seconds it publishes a new random value R, such that R is completely unpredictable prior to R's publication. Given a particular R and a timestamp T of when that R was published, we have a tuple (R, T) which can be verified by anyone. If a miner includes (R, T) as an input into a block, that miner can prove that the block was mined no earlier than T. This follows from the fact that R is completely unpredictable prior to its publication. We refer to this (R, T) input block as the provable timestamp of the block.

Such sources of randomness are known as random beacons [15] and are well studied. We discuss possible random beacons in Subsection IV-A.

Verifying a timestamp requires that the miner store the output of the beacon for the last few minutes. A further benefit of our scheme is that the random beacon removes many of the attacks inherent in Network Time Protocol [21]. Any block whose timestamp can't be verified either due to the timestamp being too old, or because it is incorrect, is treated as if it has no timestamp.

[6]This is a result of the fact that percentage of blocks mined is a direct function of mining power

Any block whose timestamp T is from the future is a sign that the random beacon has been compromised by an attacker. The verifier must immediately retransmit a hash of this block to some trusted public forum that provides timestamping as proof that the random beacon can no longer be trusted. The miner can then prove to all concerned parties that appropriate measures need be taken to fix and restore the beacon.

A. Random Beacons

Random beacons can provide unforgeable timestamps for our scheme. To avoid single points of trust we can combine n different beacons such that a timestamp is unforgeable as long as at least one of the n beacons does not collude in an attack.

One candidate is the NIST beacon [18]. The NIST beacon is designed for very similar cryptographic applications and satisfies both the property of unpredictability and public verifiability. It generates 512-bit full entropy random strings every 60-seconds using physical randomness.

Another candidate would be to use publicly known, unpredictable information as a source of entropy to an entropy mixer such as a cryptographic hash function. It has been shown that computations on public financial information, such as the values of various stocks, are effective random beacons [16].

Various mining pools could deploy their own random beacons. While each mining pool may have a incentive to behave selfishly, they are unlikely to have a common interest in cheating because they are competing with each other.

B. Compromised Random Beacon Discussion

Consider the incentives at play if a selfish mining cartel has successfully compromised a random beacon and wishes to exploit this advantage by strategically withholding blocks. The cartel must keep the compromise secret but, as we will show, it is very difficult to maintain the secrecy of the compromise of the random beacon.

Any member of the cartel can provably and safely leak that the cartel have compromised a beacon. The leaker just encrypts a withheld block with a forged R' and posts the encrypted block to a public forum prior to time T'. Now from any point in the future, the leaker can post the decryption key and prove that the block was created prior to time T'.

This makes it extremely unlikely that members of a cartel could work together since each member is able to provably and anonymously blackmail the other dishonest miners, "give the following Bitcoin address 50 bitcoins or I release the key". A honest miner need only post a reward for proof that the beacon has been compromised. If any dishonest miner takes the reward, all dishonest miners lose the long term advantage of the compromised beacon. Therefore, as long as any member of the cartels believes another member will defect, the rational choice is to defect first and get the reward.

Once the Bitcoin community learns about a compromise of the random beacon they can take actions to restore its security.

V. Conclusion

In this paper we introduced a new defense against selfish mining that improves on the previous best result, raising the minimum share of mining power necessary to profitably selfishly mine from 25% to 32%. We show that while the security of our system uses unforgeable timestamps, it is robust to their compromise. Finally we showed the difficulty of selfish miners cooperating against our defense due to incentives for members of the selfish mining cartel to leak the fact they have compromised part of the infrastructure our scheme rests on, allowing the Bitcoin community to respond and fix the damage.

As future work we could explore an alternative strategy to Freshness Preferred (FP), called Freshness Required. Under this strategy, the Bitcoin protocol is modified to treat as invalid, any block without a timestamp or with a timestamp that is older than w seconds. This does not prevent older blocks from existing in the block chain, merely that any new blocks that a miner receives are dropped if they too old. More work is needed to quantify how much this strategy raises the mining resource share, α, for selfish miners to be successful. We are in the process of evaluating this strategy.

We have done some preliminary work to validate the FP strategy using Bitcoin discrete event simulators. At this point we are not convinced that existing discrete event simulators offers the necessary fidelity to provide additional validation of our arguments. We are working on a more accurate Bitcoin simulator to fill this role.

VI. Acknowledgements

We thank Sharon Goldberg, Anna Krohn and Jen Rich for comments and suggestions on drafts of this paper.

References

[1] J. A,. Kroll, I. C. Davey, and E. W. Felten., *The Economics of Bitcoin Mining or Bitcoin in the Presence of Adversaries*, WEIS, 2013 http://www.weis2013.econinfosec.org/papers/KrollDaveyFeltenWEIS2013.pdff

[2] R, Meni., *Analysis of Bitcoin Pooled Mining Reward Systems.*, arXiv preprint arXiv:1112.4980, 2011 http://arxiv.org/abs/1112.4980

[3] D. Drainville., *An Analysis of the Bitcoin Electronic Cash System*, 2013 https://math.uwaterloo.ca/combinatorics-and-optimization/sites/ca.combinatorics-and-optimization/files/uploads/files/Drainville,\%20Danielle.pdf

[4] S. Perez., *Over 400 Retailers Are Offering Deals On New Bitcoin Black Friday Website*, Tech Crunch, 2013 http://techcrunch.com/2013/11/28/over-400-retailers-are-offering-deals-on-new-Bitcoin-black-friday-website/

[5] E. Ombok., *Bitcoin Service Targets Kenya Remittances With Cut-Rate Fees*, Bloomberg, 2013 http://www.bloomberg.com/news/2013-11-28/Bitcoin-service-targets-kenya-remittances-with-cut-rate-fees-1-.html

[6] *Bitcoin Market Capitalization*, The Bitcoin Wiki, 2013 https://blockchain.info/charts/market-cap

[7] *Bitcoin Mining Reward*, The Bitcoin Wiki, 2013 https://en.Bitcoin.it/wiki/Mining#Reward

[8] S. Nakamoto., *Bitcoin: A Peer-to-Peer Electronic Cash System*, The Cryptography Mailing List, 2008 http://Bitcoin.org/Bitcoin.pdf

[9] *Block timestamp*, The Bitcoin Wiki, 2013 https://en.Bitcoin.it/wiki/Block_timestamp

[10] *Bitcoin Protocol specification*, The Bitcoin Wiki, 2013 https://en.Bitcoin.it/wiki/Protocol_specification#block

[11] Culubas., *Timejacking & Bitcoin: The Global Time Agreement Puzzle*, Culubas Blog, 2011 http://culubas.blogspot.ca/2011/05/timejacking-Bitcoin_802.html

[12] R. Horning., *Mining Cartel Attack.*, Bitcoin Talk, 2010 https://Bitcointalk.org/index.php?topic=2227.0

[13] ByteCoin., *Re: Mining Cartel Attack.*, Bitcoin Talk, 2010, https://Bitcointalk.org/index.php?topic=2227.msg30083#msg30083

[14] I. Eyal, E. G. Sirer., *Majority is not Enough: Bitcoin Mining is Vulnerable.*, arXivpreprintarXiv:1311.0243, 2013, http://arxiv.org/abs/1311.0243

[15] M. Rabin., *Transaction protection by beacons.*, Journal of Computer and System Sciences, 27(2), 1983.

[16] C. Jeremy, and U. Hengartner, *On the Use of Financial Data as a Random Beacon.*, In Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections, pp. 1-8. USENIX Association, 2010. http://eprint.iacr.org/2010/361.pdf

[17] G. Andresen, *Neutralizing a 51% attack.*, Galvin-Tech 2012 http://gavintech.blogspot.com/2012/05/neutralizing-51-attack.html

[18] M. Iorga, NIST, *NIST Randomness Beacon*, http://www.nist.gov/itl/csd/ct/nist_beacon.cfm, 2013

[19] I. Eyal, E. G. Sirer., *Some Frequently Asked Questions on Selfish Mining*, Hacking Distributed, 2013, http://hackingdistributed.com/2013/11/05/faq-selfish-mining/

[20] C. Decker, R. Wattenhofer., *Information propagation in the Bitcoin network.*, IEEE P2P., 2013, http://www.tik.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf

[21] M. Bishop., *A security analysis of the NTP protocol version 2.*, Computer Security Applications Conference, 1990., Proceedings of the Sixth Annual. IEEE, 1990. APA http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=143746