Publish or Perish: A Backward-Compatible Defense against Selfish Mining in Bitcoin

Ren Zhang and Bart Preneel

KU Leuven, ESAT/COSIC and imec, Leuven, Belgium
{ren.zhang, bart.preneel}@esat.kuleuven.be

Abstract. The Bitcoin mining protocol has been intensively studied and widely adopted by many other cryptocurrencies. However, it has been shown that this protocol is not incentive compatible, because the selfish mining strategy enables a miner to gain unfair rewards. Existing defenses either demand fundamental changes to block validity rules or have little effect against a resourceful attacker. This paper proposes a backward-compatible defense mechanism which outperforms the previous best defense. Our fork-resolving policy neglects blocks that are not published in time and appreciates blocks that incorporate links to competing blocks of their predecessors. Consequently, a block that is kept secret until a competing block is published contributes to neither or both branches, hence it confers no advantage in winning the block race. Additionally, we discuss the dilemma between partition recovery time and selfish mining resistance, and how to balance them in our defense.

Keywords: bitcoin, selfish mining, incentive compatibility

1 Introduction

Bitcoin [15], a decentralized cryptocurrency system, attracts not only many users but also significant attention from academia. Bitcoin critically relies on an incentive mechanism named *mining*. Participants of the mining process, called *miners*, compete in producing and broadcasting *blocks*, hoping to get their blocks into the *main chain* and receive block rewards. When more than one block extends the same preceding block, the main chain is decided by a *fork-resolving policy*: a miner adopts and mines on the chain with the most work, which is typically the longest chain, or the first received block when several chains are of the same length. We refer to this *forked* situation as a *block race*, and to an equal-length block race as a *tie*. Bitcoin's designer implicitly assumed the *fairness* of the mining protocol: as long as more than half of the mining power follows the protocol, the chance that a miner can earn the next block reward is proportional to the miner's computational power [15].

Unfortunately, this assumption has been disproven by a *selfish mining attack* highlighted by Eyal and Sirer [10]. In this attack, the *selfish miner* keeps discovered blocks secret and continues to mine on top of them, hoping to gain a larger lead on the public chain, and only publishes the selfish chain to claim

the rewards when the public chain approaches the length of the selfish chain. Though risking the rewards of some secret blocks, once the selfish chain is longer than its competitor, the selfish miner can securely invalidate honest miners' competing blocks. Accordingly, the overall expectation of the selfish miner's relative revenue increases. An attacker with faster block propagation speed than honest miners can profit from this attack no matter how small the mining power share is. Furthermore, since the revenue of a malicious miner rises superlinearly with the computational power, rational miners would prefer to act collectively for a higher input-output ratio, damaging the decentralized structure of Bitcoin.

Existing defenses can be categorized into two approaches: making fundamental changes to the block validity rules, as suggested by Bahack [6], Shultz [21], Solat and Potop-Butucaru [22]; or lowering the chance of honest miners working on the selfish miner's chain during a tie, as suggested by Eyal, Sirer [10] and Heilman [12]. The former approach demands a backward-incompatible upgrade on both miners and non-miner users; while the latter approach, which we refer to as *tie breaking defenses*, has no effectiveness when the selfish chain is longer than the public chain, therefore cannot defend against resourceful attackers.

This paper proposes another defense against selfish mining. We observe that two policies are involved in deciding which blocks receive mining rewards: the fork-resolving policy demands that the main chain is the longest chain, and the reward distribution policy demands that all blocks in the main chain and no other block receive rewards. Because changing the latter leaves the protocol backward-incompatible, our defense aims to change only the fork-resolving policy.

Our defense replaces the original Bitcoin fork-resolving policy, denoted by *length FRP*, with a *weighted FRP*. By asking miners to compare the *weight* of the chains instead of their length, weighted FRP puts the selfish miner in a dilemma: if the selfish miner keeps a block secret after a competing block is published, the secret block does not contribute to the weight of its chain; if the secret block is published with the competing block, the next honest block gains a higher weight by embedding a proof of having seen this block. In both scenarios, the secret block does not help the selfish miner win the block race. Consequently, our scheme is the first backward-compatible defense that is able to disincentivize block withholding behavior when the selfish chain is longer.

Comparing with existing defenses, our defense has the three-fold advantage of backward-compatible, decentralized and effective. For evaluation, we extend the method developed by Sapirshtein et al. [18] to compute the optimal selfish mining strategy and its relative revenue within our defense. For a selfish miner with more than 40% of mining power, our defense outperforms *the optimal tie breaking defense*: an imaginary defense in which the selfish miner loses every tie.

As an additional contribution, we point out that the core reason of selfish miner's profitability is the high tolerance of the Bitcoin protocol towards network partition. To resolve this dilemma, our scheme introduces a fail-safe parameter k and demands miners to adopt the longest chain if it is at least k blocks ahead of its competitors. Adjusting this parameter allows us to balance between partition recovery time and selfish mining resistance. In the extreme forms, when $k = \infty$,

although sacrificing the ability to recover from temporary network split, our defense is effective against a 51% attacker; and when $k = 1$, the partition recovery time remains unaffected and the performance of our defense, which is reduced to a tie breaking defense, is close to the optimal tie breaking defense against a resourceful attacker. Through evaluating both the effectiveness of our scheme and its partition recovery performance in the presence of an attacker, we recommend the value $k = 3$, which can effectively defend against selfish mining attack and keep the partition recovery time within acceptable range.

We also reflect upon our modeling of the Bitcoin network and discuss the limitations and possible future work.

2 Preliminaries

2.1 Bitcoin Blockchain and Mining

We summarize here essential characteristics of Bitcoin for our discussion and refer to Nakamoto's original paper [15] and the textbook by Narayanan et al. [16] for a complete view of the system. To ensure participants have a consensus on valid transactions, all nodes follow the same *block and transaction validity rules*. A typical transaction in Bitcoin consists of at least one input and one output. The difference between the total amount of inputs and outputs in a transaction, the *transaction fee*, goes to the miner who includes the transaction in the blockchain.

The *blockchain* is a ledger containing all transactions organized as a chain of blocks. Each block contains its distance from the first block, called *height*, the hash value of the preceding *parent block*, a set of transactions, and a nonce. Information about the parent block guarantees that a miner must choose which chain to mine on before starting. A special empty-input *coinbase* transaction in the block allocates a fixed amount of new bitcoins to the miner, thus incentivizing miners to contribute their resources to the system. The miner can embed an arbitrary string in this transaction. To construct a valid block, miners work on finding the right nonce so that the hash of the block is smaller than the *block difficulty target*. This target is adjusted every 2016 blocks so that on average a block is generated every ten minutes. The protocol demands miners to publish valid blocks to the overlay network the moment they are found. If a block ends up being in the longest chain, the coinbase output and all transaction fees in the block belong to the miner. The discarded blocks do not receive any block rewards. To decrease the variance of mining revenues, miners often form *mining pools* to work on the same puzzle and split the rewards according to their contributions.

2.2 A History of Selfish Mining Strategies

The idea of selectively delaying publication of blocks to gain an unfair advantage of block rewards appears as early as 2010 [7]. In 2013, Eyal and Sirer [10] defined and analyzed a specific strategy which they called "selfish mining". Bahack presented a family of selfish mining strategies and evaluated their relative

revenue [6]. Sapirshtein et al. [18] and Nayak et al. [17] showed that under certain conditions, the selfish miner can obtain higher expected relative revenue by working on the selfish chain even when the public chain is longer.

2.3 Existing Defenses against Selfish Mining

Backward-Incompatible Defenses. Bahack proposed a *fork-punishment rule*: competing blocks receive no block reward. The first miner who incorporates a proof of the block fork in the blockchain gets half of the forfeited rewards [6]. However, honest miners suffer collateral damage of this defense, which constitutes another kind of attack. Shultz [21], Solat and Potop-Butucaru [22] recommended that each solved block be accompanied by a certain number of signatures or dummy blocks, proving that the block is witnessed by the network and a competing block is absent, before miners are able to work on it. However they did not provide a mechanism to evaluate whether the number of proofs is adequate to continue working. Neither did they mention how to prevent the selfish miner from generating a dominant number of proofs and releasing them when necessary. In addition, these three defenses require fundamental changes on the block validity and reward distribution rules, consequently network participants who do not upgrade their clients will not understand the new protocol.

Tie Breaking Defenses. Eyal and Sirer proposed that a miner chooses which chain to mine on uniformly at random in a tie [10]. The defense is referred to as *uniform tie breaking* in [18]. The authors showed that this defense raises the *profit threshold*, namely the minimum mining power share to earn unfair block rewards, to 25% within their selfish mining strategy. The threshold was later shown to be 23.21% under the optimal selfish mining strategy [18]. Heilman suggested each miner incorporate the latest unforgeable timestamp issued by a trusted party into the working block [12]. The publicly accessible and unpredictable timestamp is issued with a suggested interval of 60 seconds. When two competing blocks are received within 120 seconds, a miner prefers the block whose timestamp is fresher. The author claimed that this *freshness preferred* mechanism can raise the profit threshold to 32%. However, introducing an extra trusted party is inconsistent with Bitcoin's decentralized philosophy. At last, we note that the rules of tie breaking defenses do not apply when the selfish miner's chain is longer than the public chain, rendering the defenses ineffective against resourceful attackers. A selfish miner with 48% of total mining power earns 89% and 83% of mining rewards within uniform tie breaking and freshness preferred, respectively, given that all other miners follow honest mining strategy.

2.4 Properties of an Ideal Defense

Acknowledging the weaknesses of existing defenses, we enumerate here the desirable properties of an ideal defense.

Decentralization. Introducing a trusted server would open a new single point of failure. Moreover, it violates Bitcoin's fundamental philosophy.

Incentive compatibility. The expected relative revenue of a miner should be proportional to the mining power.

Backward compatibility. Non-miners who cannot upgrade their clients can still participate in the network. This is important for hardware products such as Bitcoin ATMs. Specifically, the following rules should not be changed:

- **Block validity rules.** A valid block in the current Bitcoin protocol should also be valid within the defense, and vice versa.
- **Reward distribution policy.** All blocks in the main chain and no other block receive block rewards.
- **Eventual consensus.** Even when an attack happens, old and new clients should reach a consensus on the main chain eventually. We will further discuss this notion in Sect. 5 and Sect. 6.

3 Our Defense Mechanism

3.1 Threat Model

We follow the threat model of most selfish mining studies [6,10,12,18,21,22]. In this model, there is only one colluding pool of selfish miners. This is considered to be the strongest form of attack because malicious miners can achieve higher input-output ratio by acting together. For brevity we use "the selfish miner" or "the attacker" instead of "the colluding pool of miners". The other miners follow the honest mining strategy. The goal of the attacker is to maximize the expected relative revenue. The selfish miner controls less than half of the total mining power so that it is infeasible for the attacker to simply generate a longer chain to invalidate the work of all the other miners.

In terms of network connectivity, we assume the selfish miner receives and broadcasts blocks with no propagation delay. However, the attacker does not have the power to downgrade the propagation speed of blocks found by other miners. Moreover, we assume there is an upper bound on honest miners' block propagation time among each other. We believe this assumption is realistic for the following reasons. First, the Bitcoin developers and the community have investigated a substantial effort to defend against network attackers and to decrease the block propagation time. This can be seen from the number of dedicated Bitcoin Improvement Proposals [5] and the fast response time to new attacks [1]. Second, miners endeavor to receive new blocks as quickly as possible, because a few more seconds of delay may render the mining effort unprofitable. The majority of large miners utilize Corallo's relay network to send and receive blocks faster [8], which consists of eight well-connected Internet traffic hubs around the globe. Other methods are observed by researchers, such as using multiple gateways for more reliable connection to the public Bitcoin network [14]. Third, Miller et al. discovered some very well-connected nodes that can increase network propagation speed [14]. In early September 2016, 50% of publicly reachable Bitcoin nodes receive a new block within 10 seconds of its creation [2].

At last, although transaction fees are supposed to substitute mining rewards in the long run, we do not consider them in our model because the amount is still quite small comparing with block rewards at this moment.

3.2 Mining Algorithm and Fork-Resolving Policy

We use τ to denote the upper bound on the block propagation time. In reality we expect miners to have a rough consensus on its value. The value can be computed either by a deterministic mechanism like block difficulty adjustment or from a miner's local information similar to network adjusted time [3].

In line with [6, 10, 12, 21, 22], we believe nodes should broadcast all competing valid blocks, instead of just the first one they receive, as in the current implementation.

Definition 1. *From a miner's perspective, a valid block is considered* ***in time*** *if (1) its height value is bigger than the miner's local chain head or (2) its height is the same as the local chain head and it is received no later than τ after receiving the first block of this height. Conversely, a valid block is* ***late*** *if the receiving miner has received a block of the same height τ before receiving this block.*

Definition 2. *A block B_1 is considered to be the* ***uncle*** *of another block B_2 if B_1 is a competing in time block of B_2's parent block.*

Notably, our definition of an uncle has two differences with the better-known uncle definition of Ethereum [4], the cryptocurrency with the second largest market capitalization: (1) the uncle has to be in time; (2) the height of a block B's uncle must be one less than the height of B.

Mining Algorithm Modification. A miner should embed in the working block the hashes of all its uncles.

We now introduce our fork-resolving policy *weighted FRP*. Since two competing chains always have a common prefix, our weight calculation only considers the last part of the chains, i.e., excluding the common prefix.

Definition 3. *From a miner's perspective, the* ***weight*** *of a chain is the number of its in time blocks plus the number of in time uncle hashes embedded in these in time blocks. Whether a block is in time is evaluated from the miner's local perspective.*

Fig. 1 illustrates two different choices of the selfish miner in the same mining sequence. In the left graph, both chains have the same weight three. Although the honest miners have only two blocks, the second block contains the hash of its uncle S because S is published in time. While in the right graph, both chains are weighted two, because the selfish miner does not publish S in time.

Weighted FRP. In a block race,

1. if one chain is longer than the others by no less than k blocks, a miner mines on the longest chain;
2. otherwise the miner chooses the chain with the largest weight;
3. if the largest weight is achieved by multiple chains simultaneously, the miner chooses one among them randomly.

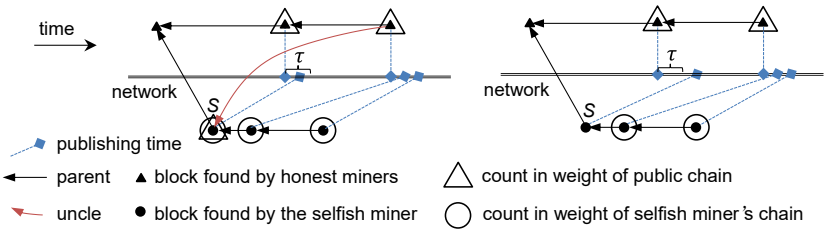

Fig. 1. Two choices of the selfish miner in the same mining sequence. In each graph, blocks are mined from left to right. A blue square indicates the time its connecting block is published. Uncle relation is represented by a red curve with an arrow. No matter whether the selfish miner publishes the first secret block in time or not, the two chains have the same weight.

A fail-safe parameter k is introduced here. When $k = 1$, our defense is reduced to a tie breaking defense: honest miners mine on the heavier chain during a tie. When $k = \infty$, the first rule in weighted FRP never applies. The implication of this parameter will be discussed further in Sect. 5.2.

It can be seen from Fig. 1 that weighted FRP puts the selfish miner in a dilemma. When a competitor of the first secret block S is released, the selfish miner has two options: if publishing S, it will be an uncle of the next honest block; if not, it will be considered a late block by honest miners. In neither way could S contribute only to the weight of the selfish chain. Furthermore, because the second selfish block is mined before the first honest block, it is impossible for the former to embed the hash of this uncle. Hence the latter block is guaranteed to contribute only to the honest chain. As a result, our defense lowers the selfish miner's incentive to withhold a discovered block. This defense is fully decentralized. As for backward compatibility, keeping the current block validity rules and reward distribution policy allows a smooth transition; non-miners who cannot upgrade their clients can also be tolerated. Miners and most publicly reachable network participants need to upgrade in order to implement our defense. Next we will show that we are the closest defense to achieving incentive compatibility to date.

4 Evaluation

Sapirshtein et al. developed an algorithm to convert a mining model into an undiscounted average reward Markov decision process (MDP), which makes it possible to compute the optimal selfish mining policy and its relative revenue [18]. In this part we first formally model the mining process of a selfish miner within our defense, then present the results output by the MDP solver, in the end compare our results with uniform tie breaking and a variant of freshness preferred.

4.1 Modeling a Block Race

We consider only the simple case of one honest miner and one selfish miner. This is based on the assumption that τ is carefully calibrated so that all honest miners have the same view on whether a block is in time. We do not consider an attacker who publishes blocks right before they are late to create inconsistent views here and discuss this attack later in Sect. 6. Mining proceeds in steps. In each step, the selfish miner first makes a decision on how many secret blocks to publish. Then the publicly visible weight of the selfish chain is updated and both miners start mining. The honest miner follows weighted FRP and compares the length and weight of both chains before starting, whereas the selfish miner always works on the selfish chain before giving up. A new block is then mined with the probability α by the selfish miner, and with $1 - \alpha$, the honest miner. The honest miner publishes the new block immediately and updates the weight, whereas the selfish miner always decides whether to publish the new block at the beginning of the next step. At the end of each step, if the block race is concluded or partially concluded, the block rewards are allocated. The rationale behind this publish-mine-found-reward sequence is that rational decisions may only change when a new block is available [6, 18].

Actions. The selfish miner has five possible rational actions in our model.

Adopt. Give up the selfish chain and mine on the honest chain. This action is always available.

Override. Publish enough blocks so that the published selfish chain is k blocks longer than the honest chain or the selfish chain's public weight is heavier than the honest chain's. The honest miner would start mining on the published selfish chain. Feasible when the selfish miner has enough blocks.

Match. If the weight difference is 0, the selfish miner keeps mining without publishing anything; otherwise publishing enough blocks so that two published chains are of the same weight. If both chains are non-empty, half of the honest mining power, namely $(1 - \alpha)/2$, would mine on the published selfish chain, while the other half works on the honest chain. Feasible when the selfish miner has enough blocks except the following scenario: there is a secret block that contains a hash of an honest uncle; if published, the selfish chain's public weight would exceed the honest chain's weight by one, otherwise the former would be smaller than the latter by one.

Even. Publish enough blocks so that the published selfish chain is no shorter than the honest chain but the selfish chain's public weight is smaller than that of the honest chain. Feasible when the selfish chain is no shorter than the honest chain, but publishing until the chains are the same length does not result in a *match*.

Hide. Do not publish anything new so that the next honest block will not embed an uncle hash. Feasible when the published selfish chain is strictly shorter than the honest chain.

These five actions are adequate because they cover all combinations of selfish miner's chain selection and reasonable public weight that might affect the honest

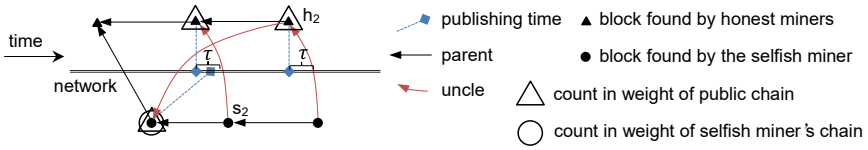

Fig. 2. A block race. Selfish block s_2 is not published in time after its competitor h_2 is mined, thus becomes a late block.

miner's chain selection. All other actions either result in the same weight calculation and chain selection or are obviously irrational. Notably, our definitions of the actions are different from those of [18]. A *wait* action in their model can either be *match*, *even* or *hide* in our model. Moreover, sometimes a *match* or *even* operation do not lead to the publication of any block.

State Space. A state is represented as a 6-tuple (B_h, B_s, $Diff_w$, *luck*, *last*, *published*). B_h and B_s denote the total length of the honest and selfish chain, respectively. $Diff_w = W_\mathrm{h} - W_\mathrm{s}$ is the weight difference between two chains. The Boolean value *luck* indicates whether there is a secret non-late block with an honest uncle. We refer to this block as *the lucky block*. There can be at most one lucky block because if there are two, the uncle with larger height would force the first lucky block to be published or convert it into a late block. There are two possible values of *last*: h or s, indicating the miner of the block mined in the last step. Finally, *published* denotes the number of published selfish blocks.

In our notation, the state in Fig. 2 is described as (2, 3, 2, 1, s, 1). The lucky block is the last selfish block because s_2 is already late. In this state, a *hide* action publishes no more block; choosing *even* publishes s_2, the resulting temporary state before mining is (2, 3, 2, 1, s, 2); publishing the entire chain would be *match*, the resulting state before mining is (2, 3, 0, 0, s, 3). The *luck* value is updated to 0 because the lucky block is no longer secret.

Reward Allocation and State Transition. The reward of each step is a 2-tuple $(R_\mathrm{h}, R_\mathrm{s})$, indicating the number of block rewards for the honest and the selfish miner, respectively. The honest miner gets $R_\mathrm{h} = B_\mathrm{h}$ only in *adopt*. In two situations the selfish miner gets rewards: *override* and at the end of *match* if the honest miner finds a block on the published selfish chain. In both situations, all selfish blocks that are published before and in this step goes to R_s.

Values in the transition matrix are assigned straight-forwardly according to weighted FRP and the action definitions. We only highlight two tricky details here. First, the selfish miner compares which rule in weighted FRP requires less blocks published before *override*. Second, the state tuple indicates which selfish block is the lucky block: if the latest block is honest, the lucky block is the competitor of this block; otherwise the last honest block is an uncle of the lucky block.

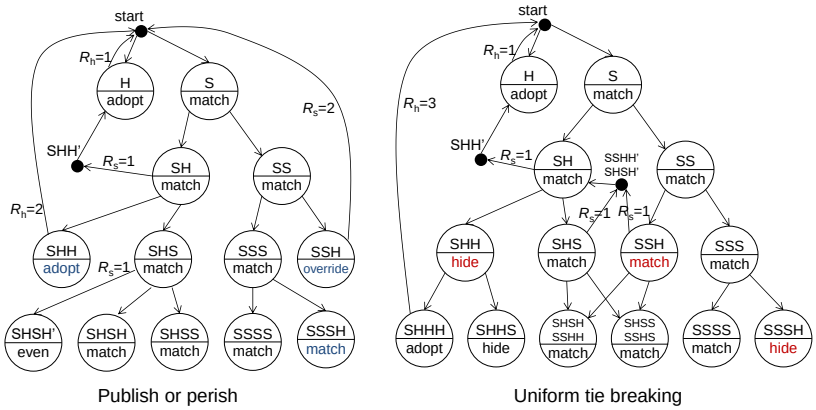

Fig. 3. The optimal selfish mining strategy when $\alpha = 0.48$.

Solving for the Optimal Policy. We compute the optimal policy and its relative revenue with the goal of maximizing the objective function

$$\frac{\sum R_{\mathrm{s}}}{\sum R_{\mathrm{s}} + \sum R_{\mathrm{h}}} \ .$$

Due to the limitation of computational resources, we set the truncating threshold of B_{h} and B_{s} to 13. Once this number is reached, the selfish miner can only choose between *adopt* and *override*. We believe this only affects the results marginally, because most block races end long before reaching this threshold. For $\alpha \leq 0.4$, lowering the threshold to 12 alters the result less than 2×10^{-4}; for $\alpha = 0.48$, lowering the threshold to 12 changes the result no more than 10^{-3}.

4.2 The Optimal Selfish Mining Strategy and its Relative Revenue

The left part of Fig. 3 displays the optimal strategy within our defense when $k = 3$, $\alpha = 0.48$. All the states of the strategy and the transitions among them are shown when the first four blocks are mined in a block race. Each circle represents a state. The string above the horizontal line in a circle describes the mining history resulted in this state. For example, "SH" means that so far two blocks are mined, the first by the selfish miner, the second by the honest miner. An apostrophe after H means the published selfish and honest chains are of the same weight, and the honest miner mined a new block on the selfish chain. The optimal action is written below the horizontal line in each state. The transition probabilities are omitted. The resulted reward is listed next to the transition arrow if it is not zero. A black dot indicates a temporary state which deteriorates into one or several other states.

For comparison, we listed the optimal selfish mining strategy within uniform tie breaking for $\alpha = 0.48$ next to our result. The action names are changed to

match our terminology. In three colored mining sequences, uniform tie breaking encourages more risky behavior for the selfish miner. Specifically, if the honest miner mines two blocks on the honest chain after a selfish block is found, a rational selfish miner would be forced to *adopt* in our scheme whereas in uniform tie breaking, the miner tries to catch up from behind. This is because in our scheme, if the selfish miner keeps working on the selfish chain, the next selfish block would be late and an extra selfish block is required to catch up the honest chain's weight. However in uniform tie breaking, only one selfish block is enough. Similarly, if the honest miner finds one block after the selfish miner finds two blocks, a rational selfish miner chooses *override* in our scheme and claim the block rewards instead of risking them.

For a complete picture on the performance, we computed the relative revenue for α between 0.20 and 0.45 with interval 0.05 within our defense, plus a powerful selfish miner with $\alpha = 0.48$. Four different k values are chosen: $1, 2, 3$ and ∞.

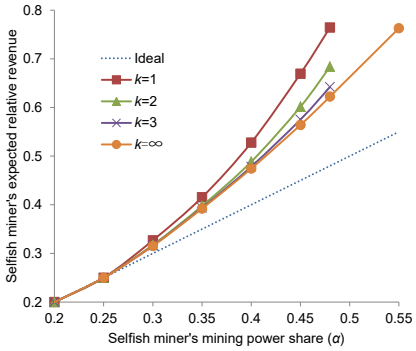

Fig. 4. Relative revenue of the selfish miner within our defense

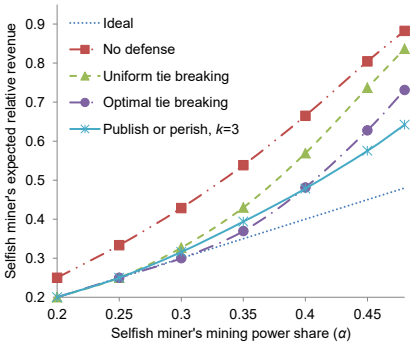

Fig. 5. Comparison with other defenses

The results can be found in Fig. 4. In all four settings, the profit threshold, minimum α to gain unfair rewards, is larger than 0.25. The relative revenue for $\alpha = 0.48$ is 0.764, 0.684, 0.642, 0.622 when $k = 1, 2, 3$ and ∞, respectively. The effectiveness of our defense increases as k grows.

An interesting result is that when $k = \infty$, our defense can prevent a malicious miner with more than 50% of mining power from taking over the network. In Fig. 4, the selfish miner with 55% of mining power only earns 76.3% of block rewards. When the blockchain integrity is more important than partition recovery time, this variant of our protocol can be useful.

4.3 Comparison with Other Defenses

The optimal selfish mining strategy with no defense is used as the base line for the comparison. Two other defenses are implemented besides our own. The first

is uniform tie breaking. The second is an imaginary defense called *optimal tie breaking*, in which the selfish miner loses every tie. This defense can be considered as the strongest form of freshness preferred, in which timestamps are issued with unlimited granularity. For our own defense we choose $k = 3$. In all defenses the selfish miner follows the optimal strategy with truncating threshold 13 to facilitate the comparison. We do not consider the defenses due to Bahack [6], Schultz [21], Solat and Potop-Butucaru [22] because the authors provided no guideline on choosing the parameters or evaluating the performance.

The results are displayed in Fig. 5. The relative revenue for uniform tie breaking and optimal tie breaking when $\alpha = 0.48$ is 0.837 and 0.731, respectively. The numbers become 0.891 and 0.831 if we set the truncating threshold to 160. The difference is because block races with a resourceful attacker usually last for dozens of blocks in these defenses. Neither defense has any effect for $\alpha \geq 0.5$. Our defense has the best performance for all α values except when $\alpha = 0.3$ and 0.35. The performance of our defense can be boosted by including a trusted timestamp server or using local time to identify potential selfish miner's blocks, however we gave up these ideas to maintain the decentralized nature of Bitcoin and avoid opening new attack vectors such as the time jacking attack [9]. Moreover, optimal tie breaking is just imaginary.

5 Balancing Partition Recovery Time and Selfish Mining Resistance

5.1 The Dilemma

If the Bitcoin network is partitioned for at least a moderate amount of time, say several hours, every part of the network would continue to mine new blocks and maintain its own version of the blockchain during the partitioned period. Upon reunion, a malicious miner may strategically mine and publish blocks to keep the network partitioned, in order to lower the quality of service, or to perform double-spending attacks. Deploying such an attack is relatively difficult in Bitcoin, since honest miners would converge to the same history as soon as one chain becomes longer than the others.

Unfortunately, Bitcoin's high tolerance of network partition and fast recovery time is one of the main causes of the selfish miner's profitability. Bitcoin allows the longer chain to claim all the block rewards after the reunion. Withholding blocks receives no punishment because of the indistinguishability between selfish mining and network partition. As a result, in the default Bitcoin protocol, there is no mechanism incentivizing the selfish miner to publish blocks as long as the selfish chain is longer.

Confronted with the dilemma, weighted FRP addresses the selfish mining problem at the price of sacrificing partition recovery time. In our scheme, previously separated groups of miners would consider blocks mined by the other groups during the partitioned period late and only recognize the weight of their own blocks. Consequently, every group works on its own chain until the first rule in weighted FRP applies.

Table 1. Average number of blocks mined before two parts of the network converge

$k \setminus \alpha$	0	0.1	0.2	0.3	0.4	0.5
2	4.00	4.45	4.99	5.69	6.67	7.98
3	12.97	15.35	18.48	23.02	29.90	40.96
4	28.96	36.27	47.08	65.03	95.12	152.14

This problem is not fundamental for the following reasons. First, large scale network partition is relatively easy to detect and hard to forge. Therefore it is possible for the protocol to switch to a different fork-resolving policy when such an incident happens. Second, the partition attack is not deemed a substantial threat in the cryptocurrency community. The GHOST protocol [23], whose variant is adopted by Ethereum, is also vulnerable to this attack. In both length FRP and weighted FRP, a newly released malicious block must be on top of its chain in order to affect honest miners' decisions, whereas in GHOST, honest miners may switch to a different chain when a secret uncle is released. Many popular cryptocurrencies, including Stellar [13] and Ripple [20], do not tolerate network partition at all. Third, our defense can achieve a good balance between effectiveness and partition recovery time. Next we explain how our protocol allows a designer to fine-tune between these two goals.

5.2 A Tradeoff

We demonstrate here the resistance of our scheme towards a network partition attack in a setting highly favorable to the attacker. In this setting, the malicious miner controls α fraction of mining power, while the honest network was partitioned into two parts with the same mining power, $(1 - \alpha)/2$ each. The two versions of main chains have the same length upon reunion. The partition lasts long enough so that the second rule of weighted FRP persistently results in different choices on the main chain. We use the Monte Carlo method to compute how long it takes on average for the two chains to converge by the first rule of weighted FRP.

At the beginning of each simulation, the lengths of the two chains are both 0. A block is generated in each round according to the mining power on both sides. The malicious miner always works on the shorter chain. The goal of the attacker is to keep the honest mining power segregated as long as possible. The simulation terminates when one chain is longer than the other by k blocks. For every k, the simulation is repeated 10^5 times and the average numbers of blocks generated on both sides before convergence are recorded.

The results of our simulation are shown in Table 1. The number of blocks can be roughly converted to time, if we assume on average a block is generated every 10 minutes. For example, when $k = 2$, a malicious miner with 30% of mining power can keep the network segregated for less than one hour after reunion. It can be seen that although a larger k value means better effectiveness of our defense, the partition recovery time also grows exponentially. However, this simulation

only reflects the worst case. If the partition period is short, honest miners are more likely to converge according to the second rule of weighted FRP; if the partition period is long, there is a large probability that one chain will be longer than the other by at least k blocks shortly after reunion.

We believe when $k = 3$, a good balance is achieved: as can be seen from Fig. 4, the effectiveness of our defense is close to optimal; the partition recovery time is no more than a few hours. For a higher-level protocol that has a contingency plan against network partition, a few hours of recovery time should not cause extra problems.

6 Limitations and Future Work

We acknowledge our scheme's limitation as it is designed and evaluated in a specific threat model, while the reality could be far more complicated.

First, Bitcoin is designed with the goal of functioning in an asynchronous network, yet we designed our defense within the synchronous model by assuming an upper bound of block propagation time. Designing secure protocols in an asynchronous network is a known hard problem. The only research that proves the security properties of Bitcoin [11] is conducted within the synchronous model. Several other selfish mining defenses also require a fixed upper bound on the block propagation time in order to be effective [12, 21, 22]. Recognizing this possible divergence between this wide-used assumption and the reality, we are unable to solve this conundrum and remove it from our model.

Second, when the fail-safe parameter $k > 1$, an attacker may broadcast blocks right before they are late to cause inconsistent views among honest miners. This is a common problem for all protocols in the synchronous model [13]. One solution is to ask miners of, e.g., 100 predecessor blocks to broadcast signatures that a block is in time, and the successor block's miner to incorporate them in the block to justify the choice of parent block. Another solution is to establish an open trust network similar to Stellar [13] to reach consensus among miners on controversial blocks. The network can be silent when no attack happens.

Third, although eventual consensus is achieved, an attacker may still utilize the temporary inconsistency between weighted FRP and old clients who follow length FRP to launch double-spending attacks. Therefore non-miners who are susceptible to this attack should also upgrade their clients. For example, a merchant running SPV client should receive and verify information about uncle blocks in order to calculate the weight value. However, to deploy this attack, the attacker needs to create a competing chain longer than the public chain and discard it, which costs at least two block rewards, around $14,000 in early September 2016.

Fourth, our model of the mining process neglects some real-world factors. Our model does not permit the occurrence of natural forks, neither did we consider the influence of transaction fees on the selfish miner's strategy or how multiple selfish miners colluding and competing with each other.

At last we note that our protocol does not achieve incentive compatibility, though it is the closest scheme to date. Achieving incentive compatibility is not an impossible task [19], we hope to reach this point in the near future.

7 Conclusion

The selfish mining attack is a fundamental challenge faced by Bitcoin, for it not only breaks the fairness assumption in the original analysis by the Bitcoin designer, but also posts potential threats to the decentralized structure of Bitcoin. Existing backward-compatible defenses can only deal with equal-length block race, but are powerless when facing a selfish chain longer than the public one. In this study, we proposed a decentralized backward-compatible defense by replacing the current length with our weighted FRP. It can defend against selfish mining when the selfish miner's private chain is longer than the honest miner's chain. Under our weighted FRP, miners evaluate which chain to mine on based on the number of blocks that are published in time and the knowledge of competing blocks that are published in time. As a result, the selfish miner's chain would have disadvantages either because the blocks are not published in time or the lack of knowledge of competing blocks. Our defense outperforms existing defenses under the optimal selfish mining strategy. Additionally, we observed that the selfish mining attack is made possible in Bitcoin as a result of Bitcoin's high tolerance towards network partition. Reflecting upon this dilemma, our scheme introduced a fail-safe parameter k. Through adjusting k, we achieved a balance between effectiveness and partition recovery time.

In contrast to existing defenses, our work attempts to defend against selfish mining through revising the fork-resolving policy rather than the reward distribution policy. This direction promises the advantage of backward-compatibility. We believe it contributes to the discussion on defending against selfish mining as it generates an alternative approach and therefore more possibilities. Our study also contributes to an in-depth understanding on the origins of selfish mining attack, namely Bitcoin's high partition tolerance. By highlighting this dilemma, we hope to raise the awareness on the trade-off between service availability and security, and therefore to open discussions on a series of choices in front of us in designing and improving Bitcoin and other blockchain technologies.

We also acknowledge the limitations of our scheme. Particularly, our defense is still only a mitigation solution towards selfish mining. A truly incentive compatible proof-of-work mining protocol is yet to be discovered.

Acknowledgements. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the imec High Impact initiative Distributed Trust project on Blockchain and Smart contracts. The authors would like to thank Yonatan Sompolinsky for pointing out several potential attacks on an earlier version of the protocol. We would also like to thank Kaiyu Shao, Güneş Acar, Alan Szepieniec, Danny De Cock, Michael Herrmann and the anonymous reviewers for their valuable comments and suggestions.

References

1. Bitcoin core version 0.10.1 released, https://bitcoin.org/en/release/v0.10.1
2. Bitcoin stats - data propagation, http://bitcoinstats.com/network/propagation/
3. Block timestamp, https://en.bitcoin.it/wiki/Block_timestamp
4. Ethereum white paper: Modified Ghost implementation, https://github.com/ethereum/wiki/wiki/White-Paper♯modified-ghost-implementation
5. Bitcoin improvement proposals (2016), https://github.com/bitcoin/bips/blob/master/README.mediawiki
6. Bahack, L.: Theoretical Bitcoin attacks with less than half of the computational power (draft). arXiv preprint arXiv:1312.7013 (2013)
7. Btchris, Bytecoin, Mtgox, RHorning: Mining cartel attack (2010), https://bitcointalk.org/index.php?topic=2227
8. Corallo, M.: Bitcoin relay network, http://bitcoinrelaynetwork.org/
9. culubas: Timejacking & bitcoin (2011), http://culubas.blogspot.be/2011/05/timejacking-bitcoin_802.html
10. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security, pp. 436–454. Springer (2014)
11. Garay, J.A., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: Analysis and applications. In: EUROCRYPT. pp. 281–310 (2015)
12. Heilman, E.: One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner. Cryptology ePrint Archive, Report 2014/007 (2014), https://eprint.iacr.org/2014/007
13. Mazieres, D.: The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation (2015)
14. Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoins public topology and influential nodes (2015)
15. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), http://www.bitcoin.org/bitcoin.pdf
16. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies. Princeton University Pres (2016)
17. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: IEEE European Symposium on Security and Privacy (EuroS&P). pp. 305–320. IEEE (2016)
18. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in Bitcoin. Financial Cryptography and Data Security (2016)
19. Schrijvers, O., Bonneau, J., Boneh, D., Roughgarden, T.: Incentive Compatibility of Bitcoin Mining Pool Reward Functions. In: Financial Cryptography and Data Security (2016)
20. Schwartz, D., Youngs, N., Britto, A.: The Ripple protocol consensus algorithm. Ripple Labs White Paper (2014)
21. Shultz, B.L.: Certification of witness: Mitigating blockchain fork attacks (2015), http://bshultz.com/paper/Shultz_Thesis.pdf
22. Solat, S., Potop-Butucaru, M.: Zeroblock: Preventing selfish mining in bitcoin. arXiv preprint arXiv:1605.02435 (2016)
23. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in Bitcoin. Financial Cryptography and Data Security (2015)