Selfish Mining Re-Examined

Kevin Alarcón Negy[1], Peter Rizun[2], and Emin Gün Sirer[1]

[1] Computer Science Department, Cornell University
[2] Bitcoin Unlimited

Abstract. Six years after the introduction of selfish mining, its counterintuitive findings continue to create confusion. In this paper, we comprehensively address one particular source of misunderstandings, related to difficulty adjustments. We first present a novel, modified selfish mining strategy, called *intermittent selfish mining*, that, perplexingly, is more profitable than honest mining even when the attacker performs no selfish mining after a difficulty adjustment. Simulations show that even in the most conservative scenario ($\gamma = 0$), an intermittent selfish miner above 37% hash power earns more coins per time unit than their fair share. We then broadly examine the profitability of selfish mining under several difficulty adjustment algorithms (DAAs) used in popular cryptocurrencies. We present a taxonomy of popular difficulty adjustment algorithms, quantify the effects of algorithmic choices on hash fluctuations, and show how resistant different DAA families are to selfish mining.

1 Introduction

Twelve years ago, the Bitcoin (BTC) white paper [26] introduced a novel consensus protocol that kicked off an era of permissionless blockchains. In describing this protocol, Nakamoto asserted that the system was secure as long as a majority of miners were honest [25]. To encourage honest participation, Bitcoin offers financial incentives in the form of newly minted bitcoins as well as transaction fees. These incentives, Nakamoto argued, would be more profitable than defying the protocol. Tantamount to an incentive compatibility claim for the protocol, these assertions were adopted widely and became folk theorems, and even garnered justification from formal modeling [18].

Nonetheless, these assertions were proven false. In a counterintuitive result, Eyal and Sirer showed that there existed an alternative strategy, known as selfish mining, whose financial incentive surpassed that of mining honestly [11]. Selfish mining involves withholding mined blocks and releasing them only after honest miners have wasted resources mining alternative blocks. Until a difficulty adjustment, wasting competitors' blocks confers no benefit to the selfish miner (SM). Following a difficulty adjustment, however, the selfish miner can collect much more than its fair share of block rewards, depending on its percentage of total network hash power (α) and what proportion of honest miners mine on a SM's block during a fork in the network (γ). Counterintuitively, the selfish mining strategy returns excess profits for any miner or pool with more than 1/3rd of the global hash power ($\alpha > 33\%$), even with the assumption that no honest miner

mines on a selfish block in a fork ($\gamma = 0$). As $\alpha \to 50\%$, a selfish miner collects close to 100% of rewards in the network, a doubling of its honest income.

Ever since its introduction, the selfish mining paper has attracted a cult of denialism [8,16,35]. Leaving aside claims that stem from an inaccurate model of how the Bitcoin protocol and pooling work [12], the resulting arguments revolve around the issues of difficulty adjustments.

First, critics have asserted that selfish mining is unprofitable because time spent forking blocks only serves to reduce the speed at which the main chain grows. Hence, the argument goes, an increase in relative revenue is meaningless because profit per time-unit decreases. Second, critics have claimed that selfish mining must necessarily involve long-duration attacks that persist past a difficulty adjustment in order to be profitable.

In this paper, we show both of these claims to be false. We illustrate a surprising selfish mining variant where the attacker ceases to act selfishly immediately after a difficulty adjustment, yet, paradoxically, still earns more than an honest miner. We call this strategy *intermittent selfish mining*. We then investigate the profitability of selfish mining under different difficulty adjustment algorithms. In particular, we quantify the benefits of selfish mining on Bitcoin Cash, Bitcoin SV, Ethereum, and Monero, and show the conditions under which profit per time-unit exceeds honest mining.

Overall, this paper introduces intermittent selfish mining (§3) and quantifies its benefits when applied to the Bitcoin protocol (§3.2). This protocol is, even more counterintuitively than the original selfish mining strategy, profitable even without performing any attack past the difficulty adjustment. Second, this work provides a taxonomy of difficulty adjustment algorithms (DAAs) (§4.1). Finally, it examines selfish mining profitability under the DAAs of Bitcoin Cash, Bitcoin SV, Ethereum, and Monero (§4.3). Overall, the paper provides a more complete picture of selfish mining's implications, and can inform the design of future proof-of-work (PoW) systems.

2 Background

This work describes an adversary employing selfish mining in proof-of-work cryptocurrencies. In this section, we first describe the PoW mining process and then outline the selfish mining algorithm.

2.1 Mining in PoW Cryptocurrencies

At their core, cryptocurrencies allow clients to publish transactions which are collated and placed into blocks by miners. In PoW cryptocurrencies, these blocks are mined by hashing the block data with a nonce until the resulting hash value is below a target value. The target value is determined by a coin's difficulty adjustment algorithm (DAA). *Difficulty* describes how difficult it is to generate a hash below this target value. Once a miner obtains a valid hash, it broadcasts the block to receive newly minted coins and collect transaction fees. Once a block resides on the longest chain, it is considered accepted. *Orphans* are blocks that do not reside on the longest chain. Accepted blocks must then be buried under

a sufficiently long suffix of the blockchain for their transactions to be considered finalized.

2.2 Selfish Mining Strategy

In selfish mining, the selfish miner with a hash rate of α withholds newly-mined blocks instead of immediately publishing them. As a result, honest miners are unaware of these blocks and, unknowingly, are coerced into wasting hash power mining blocks that are likely to be replaced in the chain. In this way, a selfish miner probabilistically earns more block rewards than honest miners.

There are three scenarios in which a selfish miner publishes a block.

First, if the SM has a private chain of length two and the next block is found by an honest miner, the new chain height difference is one. At this point, the SM publishes its entire private chain to ensure a fork win.

Second, if the SM has a private chain of length greater than two and the next block is found by an honest miner, the SM publishes only one block, the oldest block in its private chain, while keeping the rest of its private chain hidden.

Third, if the SM has found a single block and the next block is found by the honest miner, the SM will publish its block immediately. At this point, the network is in a forked state. The SM will try to mine on its own block, while the honest miners choose whether to mine on the honest or selfish block. The proportion of honest miners that mine on the selfish block is referred to as γ. Zero represents the most pessimistic γ value; it cannot be negative.

3 Intermittent Selfish Mining

We now introduce the *intermittent selfish mining* strategy. Intermittent selfish mining is a modification of selfish mining in which a miner alternates between selfish and honest mining at every difficulty adjustment in Bitcoin. The Bitcoin DAA targets a block time of ten minutes and adjusts after every 2016 blocks on the main chain. We assume the worst-case scenario for the attacker and omit transaction fees and mining costs from our analysis.

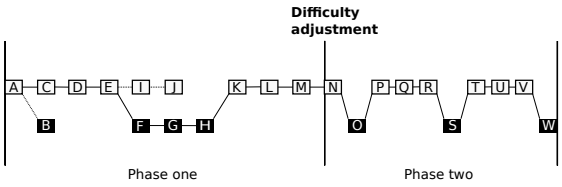


Fig. 1: Intermittent selfish mining timeline. Black indicates ISM-mined blocks. An ISM engages in selfish mining pre-difficulty adjustment, sometimes losing blocks (e.g. block B), but causes difficulty to drop by excluding blocks (e.g. I and J). The ISM mines honestly post-difficulty adjustment and collects more rewards per unit time than it would normally.

Intermittent selfish mining is comprised of two phases. In phase one, an intermittent selfish miner (ISM) employs selfish mining. The goal of phase one is to knock out the honest miners' blocks and set up the attacker to profit in the epoch directly following the attack. As pointed out by critics of selfish mining, although this results in an increase in the number of

blocks won relative to the honest miner, this by itself does not lead to increased profit for the selfish miner, not taking into account transaction fees. Phase one merely extends the time it takes to reach 2016 blocks on the main chain; it does not increase the number of blocks per minute produced by the selfish miner. In fact, the profit of the attacker slightly decreases in this phase because at every fork of equal height, the selfish miner risks losing its forked block if the honest miners are able to mine on the honest block before the selfish miner can extend its chain, when $\gamma < 1$. Nonetheless, it will force the network to lower the mining difficulty to make up for slower block times and what it perceives as a lower hash rate.

In phase two, following the difficulty adjustment, the ISM switches to honest mining. With the new lower difficulty, honest mining results in a faster mining rate than normal for all miners. Though this increased rate of minting profits all miners equally in phase two, over the two phases, the ISM profits more relative to the honest miner and per time unit. Surprisingly, this is sufficient for the attacker to gain an advantage over honest miners. This lays to rest the claim that a selfish mining attack must be launched and remain active past a difficulty adjustment. Even though no selfish mining activity takes place after a difficulty adjustment, the attacker still gains an economic advantage. Further, this change in strategy also lowers the likelihood that the honest community will detect selfish mining. An ISM could repeat this strategy over multiple periods and profit more than honest mining in each iteration.

Figure 1 shows an example of intermittent selfish mining over the length of a single iteration. The example involves an ISM with about 30% hash power. In the diagram, blocks are mined in alphabetical order. White blocks represent non-ISM, honest blocks and black blocks are ISM-mined blocks. Selfish mining is employed only in phase one. The ISM mines block B, withholds it, and then is forced to publish it to compete with honest block C. The honest miners mine block D faster than any block is mined on B and therefore B is orphaned. Later, the ISM succeeds in knocking out blocks I and J by withholding its private chain of blocks F, G, and H until the latest possible moment to guarantee a win. Once the difficulty adjustment is reached, selfish mining results in a lower difficulty to compensate for the slower build of the public chain. After this adjustment, in phase two, the ISM mines honestly. Although it wins blocks at its expected rate, the lower difficulty results in more blocks per time unit.

Intermittent selfish mining dispels a misconception about the profitability of selfish mining. Selfish mining is often argued to be impractical because the attack, it is erroneously claimed, must be maintained for several difficulty periods in order for the attacker to earn a profit. The crux of the argument is that because the selfish miner earns less revenue per unit time during the first difficulty period due to its elevated orphan rate, it must maintain the attack for several additional difficulty periods to compensate. Grunspan & Perez-Marco [16] formalized such a time-based revenue model for selfish mining and calculated that an attacker with 10% of the network hash rate and with a γ parameter of 0.9 must maintain the attack for $\sim$10 weeks. Their calculation is incorrect because they fail to

account for the profit earned by the attacker in the difficulty period following the attack. Their revenue calculation stops one difficulty period too early, when selfish mining ends. Initiating the attack results in less revenue per unit time during the first difficulty period, but this should be considered a loan rather than a loss: the attacker gets paid back at the conclusion of the attack.

It is easy to show that the selfish miner in the previous example ($\alpha = 10\%$, $\gamma = 0.9$) will earn a profit even if conducting the attack for only a single, 2016-block difficulty period and then switching to honest mining, exactly as would an intermittent selfish miner, using the state probabilities and state transitions from [11]. The presence of an ISM in phase one drives up the orphan rate for the honest miners to 8.61% and its own to 6.74%. We can use the equation

$$2016 = (n * (1 - i) * \alpha) + (n * (1 - h) * (1 - \alpha))$$

to calculate expected, non-orphaned block wins, where i and h are the ISM and honest orphan rate, respectively, and n is the number of blocks that need to be found to complete a difficulty period. In expectation, the ISM will have won 205 blocks, which is 3 blocks more than if employing honest mining, and the honest miners will have won 1811 blocks. The expected time until the difficulty adjustment stretches from 14 days to 15.29 days. Although the ISM earns more than its fair share of block rewards, its revenue per day falls from 14.40 blocks/day to 13.43 blocks/day during phase one. The following difficulty period makes up for this temporary dip.

To compensate for the fact that the last set of 2016 blocks took 15.29 days to find rather than 14 days, the Bitcoin network adjusts the difficulty parameter downwards, making the next 2016 blocks come faster. At this point, the intermittent selfish miner returns to mining honestly in phase two, thereby reducing the network orphan rate back to normal. Because of the lower difficulty, the next 2016 blocks take 12.82 days in expectation. Although the intermittent selfish miner, now mining honestly, earns only 202 blocks in expectation, its expected revenue per day increases to 15.72 blocks/day during phase two.

During these two phases, the equivalent of two difficulty periods, the intermittent selfish miner wins in expectation $205 + 202 = 407$ blocks over the course of $15.29 + 12.82 = 28.11$ days, for an average revenue rate of 14.47 blocks per day. Since its expected revenue per unit time is greater than if it were mining honestly (14.40 blocks per day), intermittent selfish mining is profitable for attack durations significantly shorter than the 70 days computed in [16].

3.1 Intermittent Selfish Mining Evaluation

We examine the intermittent selfish mining strategy using a Monte Carlo simulation to generate a chain of 8064 blocks excluding orphaned blocks, the equivalent of two intermittent selfish iterations. At each simulated second, each miner has a random chance of finding a block, set to the miner's hash rate divided by the difficulty of its most recent block.

We design our experiments to answer the following questions. (1) How does intermittent selfish mining affect difficulty? (2) What is the ISM's block-win rate (blocks/minute) and how does it fluctuate in each phase of intermittent selfish

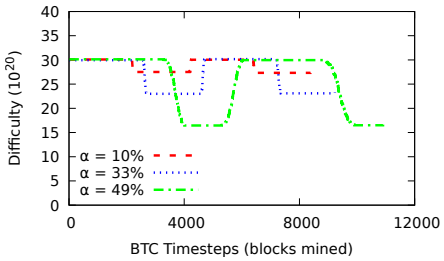


Fig. 2: An ISM causes the difficulty to lower after selfish phases and rise after honest phases ($\gamma = 0$).

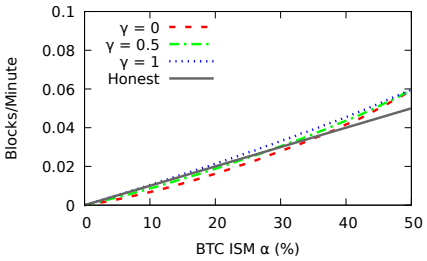


Fig. 3: ISM block-win rate.

mining? (3) Does the overall chain-growth rate (i.e. number of blocks added to the longest chain per minute) change in the presence of an ISM?

To answer these questions, we analyze difficulty, block-win rate, and chain-growth rate as each block is generated in a given run under various α and γ levels. We simulate each combination of parameters 100 times and then calculate averages and standard deviations for the data points.

3.2 Results

First, Figure 2 shows how an ISM with $\gamma = 0$ affects difficulty throughout two periods. We only show data for $\alpha = 10\%$, 33%, and 49%, which in the original selfish mining paper were minority rates that incurred losses, broke-even, and profited, respectively. As α increases, the number of blocks necessary to reach the end of the two iterations increases. Additionally, the effects of an ISM are more apparent when $\alpha = 49\%$. Selfish mining in phase one requires almost double the normal 2016 blocks to reach a difficulty adjustment. Once the difficulty lowers, honest mining in phase-two occurs for about 2016 blocks, then the period ends.

Next, Figure 3 shows the intermittent selfish miner's block-win rate for three γ levels: 0, 0.5, and 1. In comparison with the original selfish mining strategy, the potential rewards of intermittent selfish mining are more modest. Nonetheless, expected profits for an ISM surpass the profits of honest mining above certain hash rates depending on the γ value. Even in the most conservative estimation, when $\gamma = 0$, the ISM profits if its hash rate is above 37%. The block-win rates at each γ level converge at around 0.06 as α reaches 50%, which surpasses its expected 0.05 block-win rate.

The corresponding cumulative block-win rate (blocks/minute) by the ISM is shown in Figure 4. Due to the way the ISM alternates strategies, the win rate fluctuates between phases. In phase one, which includes timesteps 0 to about 4000 for $\alpha = 49\%$, selfish mining has a win rate of about 0.047, which is lower than the expected 0.049 win rate. In phase two, the win rate increases to about 0.057 at its peak, before the next phase shift. Of course, the difficulty adjustment rising back to a higher level combined with resuming selfish mining brings the cumulative block-win rate to 0.053. The win rate will continue to fluctuate, but

it will converge to about 0.0568. Figure 4 shows that a miner only has to engage in intermittent selfish mining for a little over one difficulty period to immediately win more blocks per minute than it would under honest mining.

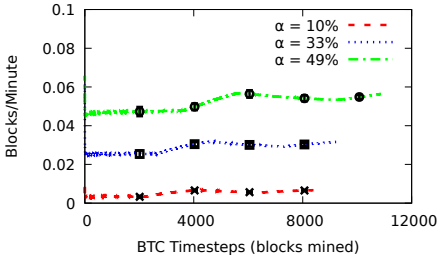


Fig. 4: ISM block-win rate after some number of timesteps ($\gamma = 0$).

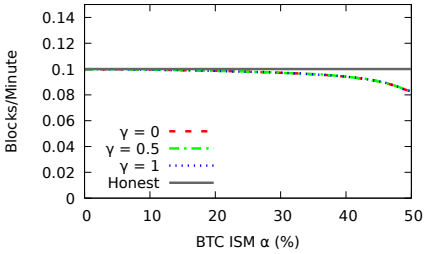


Fig. 5: Chain-growth rate in the presence of an ISM. The three γ curves are superimposed since γ values do not affect growth rate.

Finally, Figure 5 shows the chain-growth rate in the presence of an ISM. In BTC, the expected chain-growth rate is 0.1 blocks/minute. Initially, one might predict that intermittent selfish mining causes deflation by using difficulty to increase the chain-growth rate and, therefore, the supply of bitcoins. Yet, this figure shows that increasing an ISM's α rate lowers the chain-growth rate. Intermittent selfish mining, surprisingly, slows the coin mint rate. Each slow phase-one outweighs the rapid phase-two, which, as an unintended side-effect, leads to a lower chain-growth rate overall, despite increasing profits for the ISM.

4 Difficulty Adjustment Algorithms

We now focus our analysis on difficulty adjustment algorithms (DAAs). Given that there now exist various PoW coins with diverse protocols, an analysis of several current DAAs and their impact on selfish mining was necessary.

The main goal of a difficulty adjustment algorithm is to set a difficulty that causes blocks to be mined at regular target time intervals. A responsive DAA allows a cryptocurrency to quickly adjust difficulty to prevent blocks from being mined at levels too high or low compared to the target rate. On the other hand, being too responsive would allow difficulty levels to be easily manipulated by large miners entering and exiting whenever it benefits them.

In this section we classify various DAAs and evaluate their responsiveness to increases in hash power, both from an honest miner and a selfish miner.

4.1 DAA Taxonomy

Existing approaches to DAA can be classified into three categories:

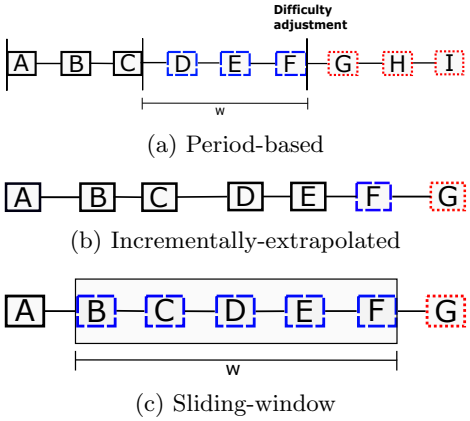
Period-based. Period-based DAAs are algorithms in which difficulty is adjusted only at the end of a typically fixed period. A period is defined as the amount of time it takes to generate w blocks on the main chain. The period

width, w, can be chosen to be large enough to minimize extreme difficulty fluctuations, but must be small enough to adjust to major hash rate changes.

Figure 6a shows a period-based DAA with $w = 3$. After block F is mined, the period ends and the difficulty is recalculated based on the block times of blocks D - F. The difficulty is then set for the next period of blocks G - I.

Our evaluation of DAAs uses Bitcoin as the period-based cryptocurrency. The Bitcoin DAA targets an average mining time of ten minutes per block [6]. After 2016 blocks are mined on the main chain, which takes roughly two weeks, the difficulty is adjusted to get closer to the target block time.

Incrementally-extrapolated. The incrementally-extrapolated DAA is one in which difficulty is incremented/decremented depending on how far outside of the block timing bounds a new block is. In contrast to Bitcoin, where a proportion can be used to calculate a new difficulty, incrementally-extrapolated difficulties increase/decrease the current difficulty by a fractional amount. This DAA limits the responsiveness of the cryptocurrency to hash rate changes.



(a) Period-based

(b) Incrementally-extrapolated

(c) Sliding-window

Fig. 6: DAA Taxonomy. Blue blocks are used in the difficulty calculation. Red blocks are mined using the new difficulty.

Figure 6b shows an example of an incrementally-extrapolated DAA. To mine block G, the DAA only looks at the elapsed time since the parent block F was generated and then adds or subtracts from the parent difficulty depending on how close the elapsed time was to the target block time.

Ethereum (ETH), as of Byzantium, uses an incrementally-extrapolated DAA that adjusts at every new block [9]. Each new block difficulty is calculated by measuring the time difference between the current timestamp and that of its parent, then incrementing or decrementing the parent difficulty depending on whether the time difference is outside of the desired bounds of 9–17 seconds. As will be shown in the results, a sudden doubling of the hash rate in the network will cause a slow difficulty change compared to a DAA which uses proportion calculations and can adjust difficulty completely within one block.

Sliding-window. The sliding-window DAA is similar to period-based except its difficulty recalculation occurs at every new block. To calculate the current difficulty, a block-window of width w consisting of ancestor blocks is used. A new difficulty is calculated based on the amount of time it took to generate the blocks in the block-window compared to the expected time.

Figure 6c shows a sliding-window DAA with $w = 5$. To mine block G, the DAA slides its window over blocks B - F and recalculates the difficulty based on how long it took to generate the blocks within the window compared to a target value. Once G is mined, it will be included in the next window.

Cryptocurrencies that use sliding-window DAAs are Bitcoin Cash (BCH), Bitcoin SV (BSV), and Monero (XMR). Given that BSV branched off BCH [1], both share the same DAA and for this paper we refer to them collectively as BCH/BSV. BCH/BSV targets ten minutes per block [3], while Monero targets two minutes per block [23]. The sliding window widths are ∼144 and 600 blocks for BCH/BSV and XMR, respectively. To avoid timestamp-based attacks, BCH/BSV chooses the median of the three most recent blocks and the median of the blocks 144–146 behind the current block based on timestamp to use as the beginning and end of the window, respectively [4]. XMR, on the other hand, orders the last 745 blocks, excludes the most recent 15, then omits the outer 120 blocks (i.e. 60 recent and 60 oldest) leaving 600 blocks in its window [24].

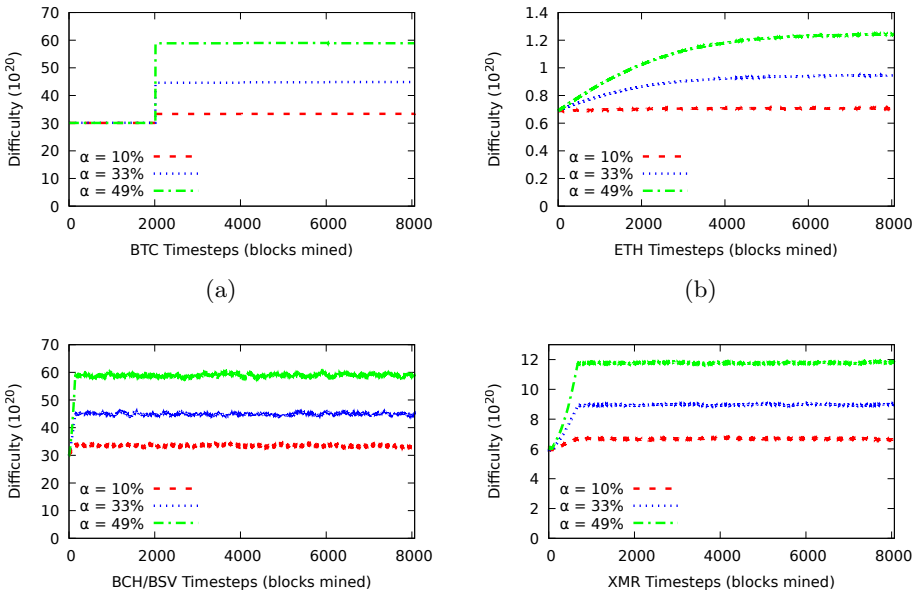


Fig. 7: Difficulty once a new honest miner enters the system ($\gamma = 0$).

4.2 DAA Evaluation

To evaluate these difficulty adjustment algorithms, we examine attacks launched by renting extraneous hash power and ask the following questions. (1) How effective are DAAs at adjusting difficulty if a substantial amount of hash power

is introduced to the network? (2) How much can a new miner profit in terms of block-win rate upon entering a new cryptocurrency?

First, we analyze how the various DAAs of Bitcoin, ETH, BCH/BSV, and Monero alter difficulty once a new honest miner enters the system. We also evaluate the block-win rate of the new miner while it takes advantage of the old difficulty. Second, we compare the profitability of a new selfish miner under the different DAA schemes. As before, each experiment simulates the generation of 8064 blocks on each blockchain, not including orphans.

Thus, our simulations start with a network of a given hash power, and then add additional mining power belonging to the adversary. For instance, to introduce a miner with a 30% hash rate, we give the miner enough hash power, S, such that $S/(S+H) = 0.3$, where H is the initial hash power in the network.

We disregard timestamp manipulation attacks by miners because they are an orthogonal concern and their full treatment is beyond the scope of this paper. So, when choosing median BCH/BSV outer blocks, the middle block is always the outer block since the three blocks are guaranteed to be in timestamp order. Recent data from these systems indicate that the timestamp in new blocks matches global time to within seconds.

Our simulations also take into account the difficulty clamps of Bitcoin and BCH/BSV. If the blocks used in a difficulty adjustment were mined too slowly or too quickly, the difficulty adjusts only to the limits set by the difficulty clamps. Bitcoin has a difficulty clamp of 4× or 0.25× the target time, while BCH/BSV has a clamp of 2× and 0.5×. As Section 4.3 will show, selfish miners under 50% hash power will be unaffected by these clamps.

4.3 Results

We first examine how difficulty adjusts if a new honest miner enters the network, shown in Figure 7. With the exception of ETH, period/window width is the most significant determining factor in the adjustment period. This width is the amount of blocks necessary to completely adjust difficulty and reach equilibrium when hash power is added to the network. For this reason, BSV and BCH, with their 144 block-window, are the fastest to adjust. Monero takes longer at 675 blocks, which come from the 600 block-window width and the 75 newest blocks that are omitted from the sliding-window. Finally, Bitcoin takes longest of the three with a period width of 2016. ETH differs in that there is no concept of width in its DAA. Since it is incremental, difficulty takes about 10,000 blocks to stabilize.

We omit showing difficulty graphs when a new SM enters the network because difficulty does not adjust much, if at all, for any of the schemes analyzed. A new selfish miner spends time trying to create forks in the network instead of using its hash power to help grow the longest chain. As such, DAAs will not adjust in this scenario since the chain will grow at roughly the same rate as before despite having extra hash power in the network.

Next, we analyze the cumulative block-win rate for a new honest miner in Figure 8. The key takeaway is that upon entering a cryptocurrency, powerful new miners can take advantage of an initial low difficulty to mine blocks faster than normal. Noticeably, a new miner can leverage this initial period in Bitcoin

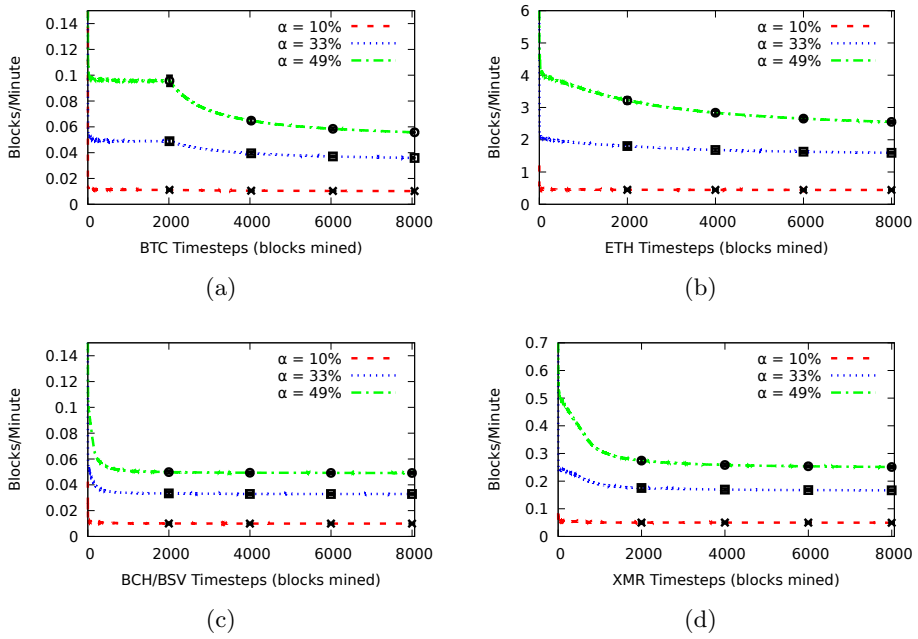


Fig. 8: Cumulative block-win rate of an new honest miner after some number of timesteps ($\gamma = 0$).

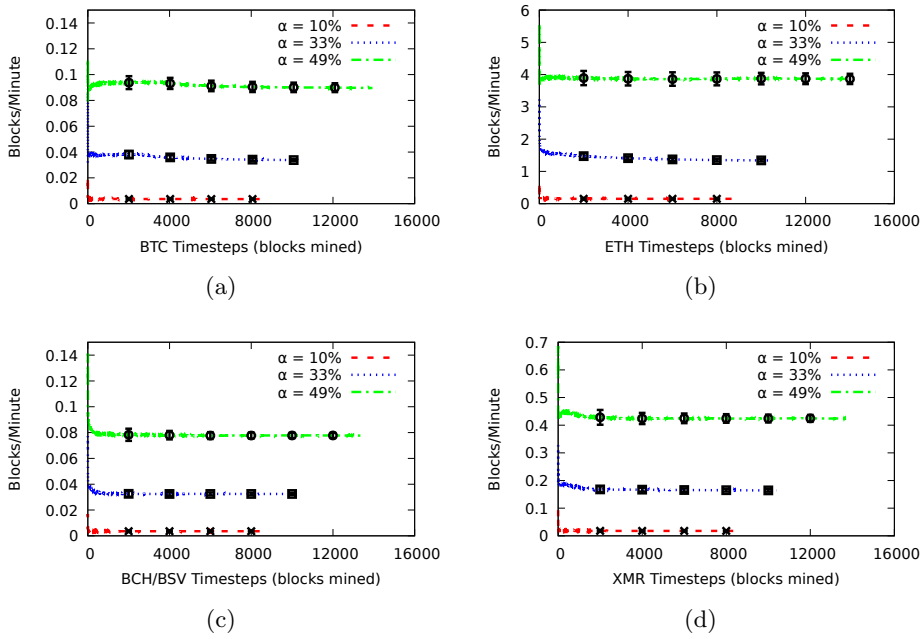


Fig. 9: Cumulative block-win rate of a new selfish miner after some number of timesteps ($\gamma = 0$).

for about two weeks. The 49% miner wins 0.1 blocks per minute for the duration of the period immediately upon entering. Once the difficulty adjusts, the win rate gradually declines. ETH, BCH/BSV, and Monero, on the other hand, begin adjusting difficulty and lowering the miner block-win rate almost instantly. These graphs imply that there is a benefit to a miner alternating between cryptocurrencies and profiting from low difficulties in each, only abandoning a coin to allow its difficulty to revert back to profitable levels.

We now evaluate the same measure if the new miner in the system is a selfish miner. Figure 9 shows corresponding block-win rates for a new selfish miner. In the four coins, the new SM with $\alpha = 49\%$ earns about double the amount of blocks per minute than what it should with honest mining. These graphs corroborate the fact that a new SM orphans enough blocks that the chain-growth rate and, therefore, difficulty barely changes. Hence, with the DAAs under analysis, DAA choice is irrelevant when it comes to selfish mining.

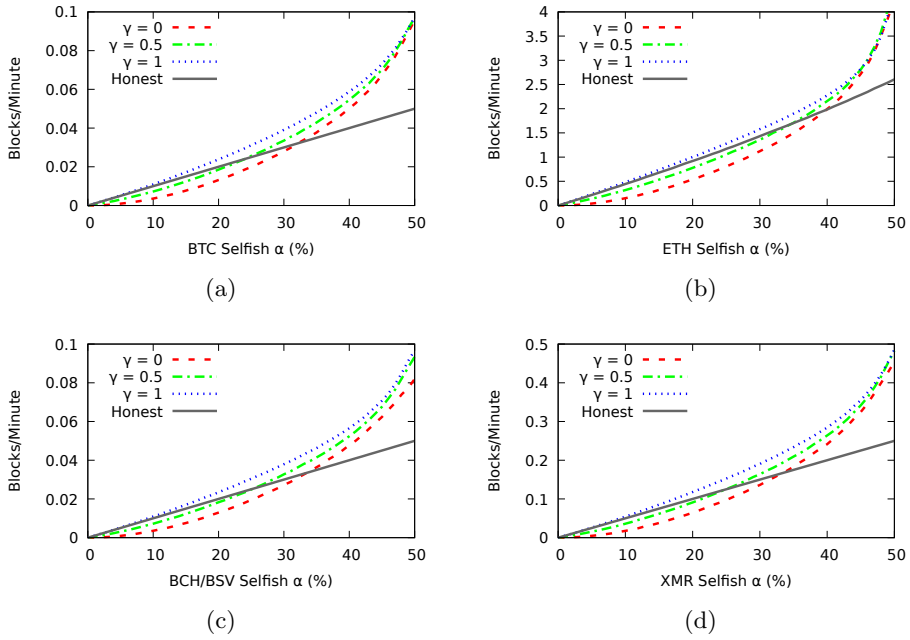


Fig. 10: Block-win rate earned by a new selfish miner.

The next graphs in Figure 10 show the block-win rate at the end of the simulation. As expected, higher γ leads to higher win rate. It also lowers the threshold of hash power needed to break even. $\gamma = 1$ allows a new selfish miner with any amount of hash power to enter any coin and at the very least make what it was expected to make if it had employed honest mining. A more reasonable low γ rate would require about 33% of the global hash rate to break even.

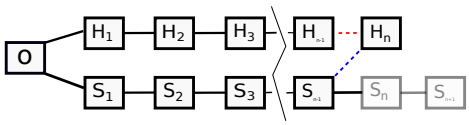


Fig. 11: After a lengthy chain-race, the honest miner chooses whether to build on H_{n-1} or S_{n-1}. In this scenario, the honest chain will have a lower difficulty than the selfish chain, since it took longer to produce. Blocks S_n and S_{n+1} are unpublished.

As shown in Figure 11, suppose the SM takes an initial hidden lead of 3 blocks (S_1, S_2, and S_3). The honest miner (HM) then spends time mining on the original parent and creates its own chain with block H_1. The SM publishes block S_1 to compete with H_1. In $\gamma = 0$, the HM will try to mine on its own H_1 block while it thinks it is a winnable chain. Assume the SM mines a fourth block, S_4, in its private chain and then the HM mines on its own block H_2, and both alternate some number of blocks. When mining block H_n, the HM has a choice: in $\gamma = 0$, it chooses to mine on its own block, H_{n-1}; in $\gamma = 1$, it chooses to mine on the selfish block, S_{n-1}. This choice has a significant effect on difficulty. Since S_{n-1} was mined long before H_{n-1} was mined, mining on S_{n-1} should have a higher difficulty than mining on H_{n-1}. Therefore, choosing to mine on the selfish block is the same as choosing to mine on a more difficult chain.

If n is on the order of hundreds, as can happen when α is close to 50%, the honest chain could have a significantly lower difficulty than the selfish chain. Since the HM will inevitably lose, the decision of which chain to mine on determines how fast an HM can force the SM to release all private blocks. In $\gamma = 0$, honest miners can catch up to the SM much faster by taking advantage of a lower difficulty. When $\gamma = 1$, however, honest miners choose to mine on the harder chain and will more slowly catch up to the selfish miner. In the long run, honest miners choosing the more difficult chain means more honest blocks are orphaned and the SM is able to extend its winning chain for longer than if $\gamma = 0$. Thus, the selfish block-win rate and the proportion it earns in $\gamma = 0$ is significantly less than in $\gamma = 1$ for high hash rates.

The gap seen in Figure 10c between $\gamma = 0$ and 1 will be more or less pronounced depending on the length n the two chains reach before merging and the DAA block width, w. In our experiments, we observed n to be on the order of hundreds. If $n < w$, the difference in difficulties between the two chains will be less significant as with Bitcoin, whose block width is 2016. For this reason, the original selfish mining analysis did not exhibit a gap at higher α values. BCH/BSV, on the other hand, has a block width of 144. When the value of n is on the order of hundreds it significantly impacts the difficulty in BCH/BSV. Finally, Monero exhibits a gap that is greater than Bitcoin but smaller than BCH/BSV since its block width of about 675 falls in the middle of the other two widths. Although ETH should have the widest gap, since it only looks at the difficulty of the parent, it does not exhibit this gap due to its incremental nature that causes difficulty to adjust gradually as seen in Figure 7b.

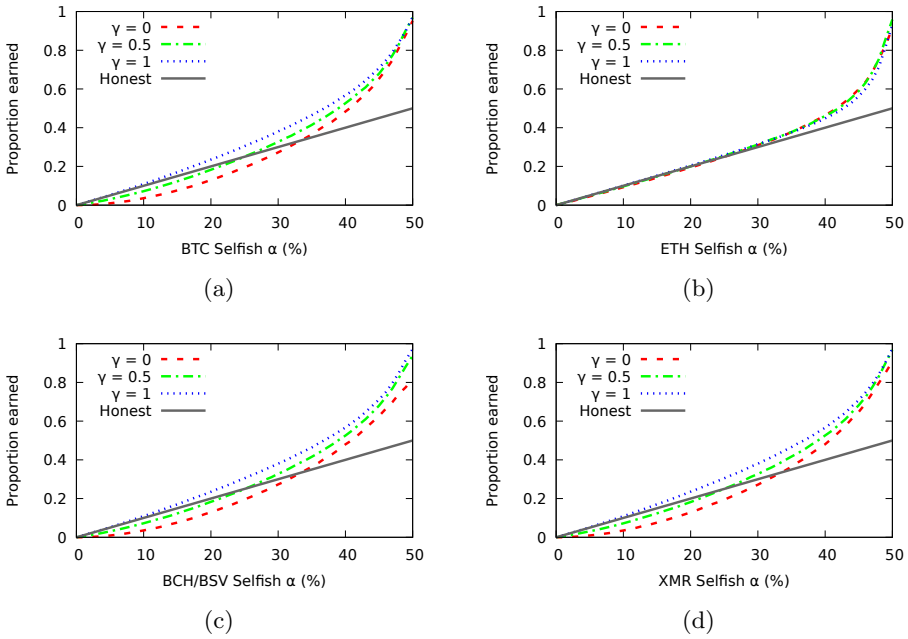


Fig. 12: The proportion of profit earned by a new SM. Uncle rewards in ETH practically nullifies any penalties an SM would pay from creating forks.

Finally, the graphs in Figure 12 show the proportion earned by the selfish miner relative to the honest miner. For BTC, BCH/BSV, and XMR, the results are similar to the results from the original selfish mining results, with the exception of the BSV γ gap, mentioned above. ETH, on the other hand, shows that a new selfish miner with any hash rate and any γ value can at least break-even. This finding is entirely due to the uncle block reward system that exists in ETH, which is described in more detail in Appendix A.

5 Related Work

This work covers two areas of research: difficulty adjustment algorithms and deviant mining behavior. We discuss related work below.

Difficulty Adjustment Algorithms Prior work in DAAs focuses on the relationship between hash power and difficulty. Kraft [17] analyzed the Bitcoin DAA in the presence of exponentially-increasing hash rate and found that it resulted in lower average block times than desired. Noda et al. [28] compared several DAAs and concluded that Bitcoin's DAA could not stabilize block times in the face of fluctuating hash power. Neither work considers how difficulty adjusts in the presence of deviant mining behavior.

A few recent studies have looked into leveraging DAAs to increase profits. Fiat et al. [13] examined equilibria when miners are allowed to throttle their hash power to bring down the difficulty level. Smart mining [15] is similar to intermittent selfish mining in that it also employs alternating strategies. It alternates

between honest mining and remaining idle at every difficulty adjustment. Our work differs from both studies in that our adversary exerts its full hash power and it actively attempts to harm the profits of other miners by employing selfish mining. We leave quantitative comparisons between intermittent selfish and smart mining as future work.

Deviant Mining Behavior Since its introduction, selfish mining researchers have looked at various contexts and strategy modifications to see how its profitability is affected [14, 27, 31, 32]. Sapirshtein et al. [31] and Nayak et al. [27] showed that small modifications to selfish mining lead to higher profits depending on the α and γ rates. Göbel et. al. [14] analyzed selfish mining using a propagation-delay model. These studies all assume a constant difficulty level.

The selfish mining strategy is one of several attacks that creates forks in the blockchain. Liao and Katz [21] present a strategy where so-called whale transactions with large fees are used to convince miners to fork the network. Kwon et al. [19] introduce the fork-after-withholding attack where a mining pool participant only tries to fork blocks from miners in competing pools.

Most previous work falls under the larger umbrella of mining attacks and deviant strategies. For example, research has looked at unintended mining behavior once Bitcoin no longer confers block rewards [7, 34], strategies employed by mining pool participants [10, 20, 22, 30], and coin-hopping strategies [2, 5, 33].

6 Conclusion

This paper examined the controversy around selfish mining and evaluated its application to a range of popular cryptocurrencies. Specifically, it introduced intermittent selfish mining and examined several difficulty adjustment algorithms with selfish mining in action. With intermittent selfish mining, this paper showed that selfish mining under the Bitcoin DAA can be profitable without extending the attack past a difficulty adjustment. Separately, this work quantified the envelope within which selfish mining is a feasible strategy against the various DAAs present in BTC, ETH, BCH, BSV, and XMR.

Selfish mining is an instance of game-theoretic attacks that take advantage of information asymmetry in distributed systems. Such attacks tend to be subtle, unexpected, and at times, counterintuitive. We caution laypeople against accepting folk theorems at face value.

7 Acknowledgements

We thank Soumya Basu, Ittay Eyal, Kai Mast, and the anonymous reviewers for their insightful suggestions. We also thank Vitalik Buterin for providing inspiration for intermittent selfish mining. This work is supported by Alfred P. Sloan Foundation Grant 2016-20166039.

References

1. Bitcoin SV Version 0.1 Goes Live. `https://bitcoinsv.io/2018/10/21/bitcoin-sv-version-0-1-goes-live/`, accessed: 2018-11-7

2. Altman, E., Reiffers, A., Menasché, D., Datar, M., Dhamal, S., Touati, C., El-Azouzi, R.: Mining competition in a multi-cryptocurrency ecosystem at the network edge: A congestion game approach (2019)
3. Bitcoin-ABC Community: Bitcoin-ABC Source Chainparams. `https://github.com/Bitcoin-ABC/bitcoin-abc/blob/master/src/chainparams.cpp`, accessed: 2018-11-02
4. Bitcoin-ABC Community: Difficulty Adjustment Algorithm Update. `https://www.bitcoinabc.org/2017-11-01-DAA/`, accessed: 2018-11-5
5. Bonneau, J.: Why buy when you can rent? In: International Conference on Financial Cryptography and Data Security. Springer (2016)
6. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Sok: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In: Security and Privacy (SP), 2015 IEEE Symposium on (2015)
7. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016)
8. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718 (2014)
9. Ethereum Community: Byzantium Hard Fork changes. `https://github.com/ethereum/wiki/wiki/Byzantium-Hard-Fork-changes`, accessed: 2019-01-2
10. Eyal, I.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy (2015)
11. Eyal, I., Sirer, E.G.: Majority is not Enough: Bitcoin Mining is Vulnerable. CoRR **abs/1311.0243** (2013), `http://arxiv.org/abs/1311.0243`
12. Felten, E.: Bitcoin isn't so broken after all. `https://freedom-to-tinker.com/2013/11/07/bitcoin-isnt-so-broken-after-all/`, accessed: 2019-04-10
13. Fiat, A., Karlin, A., Koutsoupias, E., Papadimitriou, C.: Energy equilibria in proof-of-work mining. In: Proceedings of the 2019 ACM Conference on Economics and Computation (2019)
14. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. Performance Evaluation **104** (2016)
15. Goren, G., Spiegelman, A.: Mind the mining. In: Proceedings of the 2019 ACM Conference on Economics and Computation (2019)
16. Grunspan, C., Pérez-Marco, R.: On profitability of selfish mining. arXiv preprint arXiv:1805.08281 (2018)
17. Kraft, D.: Difficulty control for blockchain-based consensus systems. Peer-to-Peer Networking and Applications **9**(2) (2016)
18. Kroll, J.A., Davey, I., Felten, E.W.: The Economics of Bitcoin Mining , or Bitcoin in the Presence of Adversaries (2013)
19. Kwon, Y., Kim, D., Son, Y., Vasserman, E., Kim, Y.: Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017)
20. Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., Rosenschein, J.S.: Bitcoin mining pools: A cooperative game theoretic analysis. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (2015)
21. Liao, K., Katz, J.: Incentivizing blockchain forks via whale transactions. In: International Conference on Financial Cryptography and Data Security (2017)

22. Luu, L., Saha, R., Parameshwaran, I., Saxena, P., Hobor, A.: On power splitting games in distributed computation: The case of bitcoin pooled mining. In: 2015 IEEE 28th Computer Security Foundations Symposium (2015)
23. Monero Community: Monero Source Cryptonote-Config. https://github.com/monero-project/monero/blob/master/src/cryptonote_config.h, accessed: 2018-11-02
24. Monero Community: Monero Source Difficulty. https://github.com/monero-project/monero/blob/master/src/cryptonote_basic/difficulty.cpp, accessed: 2018-12-7
25. Nakamoto, S.: Bitcoin P2P e-cash paper. https://satoshi.nakamotoinstitute.org/emails/cryptography/threads/1/?view=satoshi#014849, accessed: 2019-04-11
26. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
27. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on (2016)
28. Noda, S., Okumura, K., Hashimoto, Y.: A Lucas Critique to the Difficulty Adjustment Algorithm of the Bitcoin System. SSRN (2019), https://ssrn.com/abstract=3410460, accessed: 2019-09-15
29. Ritz, F., Zugenmaier, A.: The Impact of Uncle Rewards on Selfish Mining in Ethereum. In: 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (2018)
30. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint arXiv:1112.4980 (2011)
31. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal Selfish Mining Strategies in Bitcoin. In: Financial Cryptography and Data Security (2017)
32. Shomer, A.: On the phase space of block-hiding strategies. IACR Cryptology ePrint Archive (2014)
33. Spiegelman, A., Keidar, I., Tennenholtz, M.: Game of coins. arXiv preprint arXiv:1805.08979 (2018)
34. Tsabary, I., Eyal, I.: The gap game. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (2018)
35. Wright, C.S.: The Fallacy of the Selfish Miner (1): Economic Argument Critiqued (2017)

A Uncle blocks and selfish mining

An *uncle block* is an orphaned block whose parent resides on the main chain. Uncle blocks in ETH can be referenced by later blocks on the main chain and are rewarded according to the equation $(8-h)*b/8$, where b is the block reward and h is the height difference, up to 6, between the uncle block and the referencing block. Additionally, the creator of the referencing block is rewarded with an extra $b/32$ per uncle that is included, up to two uncles. Note that if a losing fork is longer than one block long, only the first block in the losing chain will be rewarded as an uncle. This system incentivizes miners to reference uncle blocks to gain extra rewards, while disincentivizing small miners from joining mining pools by rewarding, albeit minimally, these losing blocks.

This reward structure has the unintended consequence of nullifying the risks and penalties of selfish mining. As has been noted before [29], uncle block rewards allow selfish miners to fork without suffering a massive loss if the fork loses. As shown in Figure 12b, uncle block rewards in ETH allow selfish miners to at minimum break even, no matter what amount of hash power they possess.

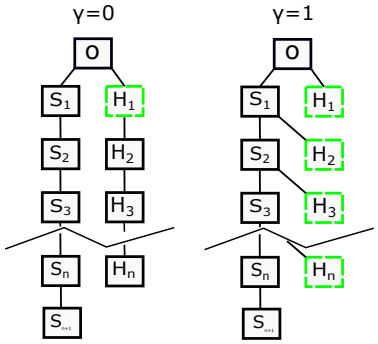


Fig. 13: In ETH, $\gamma = 0$ results in only one honest uncle block, H_1, whereas $\gamma = 1$ results in n possible uncles.

Interestingly, the graph shows unexpected behavior when α is around 45%. Around this hash rate, a selfish miner earns less relative to the honest miner under $\gamma = 1$ than if $\gamma = 0$. Normally, $\gamma = 1$ allows an SM to win all forks and earn more than if $\gamma = 0$ and the SM has to compete to win a fork. Yet, here we find the reverse to be true. This counter-intuitive finding stems directly from the uncle reward structure.

Figure 13 shows an example that provides the intuition behind this finding. Suppose an SM has mined a private chain of length 3 (S_1, S_2, S_3) on block o, the origin block. By not publishing any block, the SM allows the honest miner to waste resources mining on block o. If the honest miner is able to mine a block, H_1, then the SM releases S_1 to create an artificial fork. Assume the SM then mines a block, S_4, to maintain a chain length difference of 3. Then the honest and selfish miner begin alternating mining blocks. In $\gamma = 0$, the honest miner chooses to mine on its own blocks. No matter how long the two chains become, the honest miner will lose all its blocks, but H_1 will be eligible to be rewarded as an uncle block since its parent o is on the winning chain. However, when $\gamma = 1$, once the artificial fork of S_1 and H_1 is created, the honest miner will abandon H_1 and mine on S_1. As the two miners alternate finding blocks, the honest miner will always abandon its own block and mine on the selfish block. The key is that now all honest blocks from H_1 to H_n can be included as uncle

blocks. Since the selfish miner was ahead throughout the entire chain buildup, most of the uncle blocks will be included by the selfish miner if following a greedy inclusion strategy. Although both uncle creator and including miner are mutually rewarded for the uncle block, the inclusion of each additional uncle block gives a relatively higher reward to the creator than the including miner. Thus, in the best case scenario for the selfish miner, where b is the block reward, the honest miner would receive $2/8 * b$ and the selfish miner that has included the uncle block receives $1/32 * b$, if the height difference is 6. Normally, the honest miner would be rewarded even more since the uncle block would likely be included within a few block generations. In the example above, if H_1 is included in the block S_4, which is the earliest that the selfish miner could know about H_1, the height difference between the uncle and the new block would be three and the honest miner would then receive $5/8 * b$. Ritz and Zugenmaier [29] discuss other uncle inclusion strategies for the selfish miner.