On the Profitability of Selfish Mining Against Multiple Difficulty Adjustment Algorithms

Michael Davidson[1] and Tyler Diamond[2]

[1] National Institute of Standards and Technology, michael.davidson@nist.gov
[2] National Institute of Standards and Technology tyler.diamond@nist.gov

January 16, 2020

Abstract. The selfish mining attack allows cryptocurrency miners to mine more than their "fair share" of blocks, stealing revenue from other miners while reducing the overall security of payments. This malicious strategy has been extensively studied in Bitcoin, but far less attention has been paid to how the strategy may impact other cryptocurrencies. Because selfish mining is an attack against the difficulty adjustment algorithm (DAA) of a cryptocurrency, it may have a different effect when used on coins with different DAAs. In this work, we study the degree to which selfish mining can increase the revenue of miners for a wider variety of cryptocurrencies than have been studied before, including Bitcoin, Litecoin, Bitcoin Cash, Dash, Monero, and Zcash. To do so, we generalize the selfish mining strategy to blockchains with variable difficulty, and use simulations to measure how profitable the strategy is. We find that the other cryptocurrencies under consideration are far more susceptible to selfish mining than Bitcoin is, and that the strategy is profitable for miners with a lower hash rate. We also show that by dishonestly reporting block timestamps, selfish miners can generate enormously disproportionate revenues up to 2.5 times larger than they would through honest mining for some DAAs. For each DAA, we consider what happens when parameters are changed, and suggest parameter sets that would improve the algorithm's resilience against selfish mining.

Keywords: Bitcoin · Cryptocurrency · Selfish Mining · Difficulty Adjustment Algorithm · Blockchain

1 Introduction

Bitcoin, the first cryptocurrency, was initially described by the pseudonymous Satoshi Nakamoto in 2008 [23]. Since then, thousands of additional cryptocurrencies have been created. The primary innovation that allowed cryptocurrencies to exist securely was to incentivize a distributed, pseudonymous set of "miners" to produce proof of work puzzle solutions in order to agree on a consistent ordering of transactions into blocks, and then adjust the "difficulty" of these puzzles to approximately track changes in mining activity ("hash rate"). These puzzles typically involve hashing a block header and nonce repeatedly until finding a hash that is below a certain target value determined by the difficulty level. In the literature, this is commonly described as an attempt to hash until the resulting digest has "a certain number of leading zeros".[1]

By including the previous block header hash in each block header, this proof of work process creates a chain of blocks ("blockchain") that are cryptographically linked, and would require even greater work to overwrite. As such, it was widely believed that

[1] This is an oversimplification which ignores the mantissa; changes to the target are not performed in neat multiples of two.

cryptocurrencies such as Bitcoin were secure as long as the majority of the hash rate is following the protocol honestly. In particular, if a given miner has $x\%$ of the hash rate, they should in expectation receive $x\%$ of the total block rewards over an extended period of time.

However, as Bitcointalk user RHorning pointed out in 2010, miners with less than half of the hash rate can force other miners to waste computing power by not honoring their blocks [29]. Later, Eyal and Sirer (and Bahack independently) formalized the idea through what they dubbed the "selfish mining" strategy [2, 7]. An honest miner is expected to broadcast any newly-found blocks to his peers so they quickly propagate across the network. However, by strategically or "selfishly" withholding blocks to build a private blockchain, he can choose to broadcast his private chain at a time that will maximize his revenue; essentially, he can force other miners to waste work on blocks that are destined to be "orphaned" and become stale; that is, those blocks will not be in the canonical blockchain recognized as valid. The miners of those now stale blocks will not be compensated for their work.

This work investigates the profitability of selfish mining against difficulty adjustment algorithms other than Bitcoin's and Ethereum's, while suggesting parameter changes that may improve the algorithms' resistance to selfish mining. To this end, we generalize the selfish mining strategy for chains of variable difficulty. Furthermore, we demonstrate the relative importance of manipulating block timestamps as part of the selfish mining strategy space.

Section 2 provides necessary background information. Section 3 covers related work on selfish mining and difficulty adjustment algorithms. Section 4 describes our methodology, and section 5 reports the results of our simulations for each difficulty adjustment algorithm. Section 6 provides some general discussion, and section 7 concludes.

2 Background

2.1 Selfish Mining

Typically, when a miner mines a new block, they will broadcast that block to their peers with the intention of having it propagate to the rest of the network as quickly as possible. A miner does not get to spend the block reward until a certain number of blocks have been built on top of his, so it is generally in the miner's best interest to get any new blocks into the hands of competing miners quickly, so other miners can mine on top of the newly found block.

Under some conditions, however, a deviant strategy will allow a miner with $x\%$ of the global hash rate to receive more than $x\%$ of the total reward. The strategy works by forcing honest miners to waste effort mining on top of blocks that are destined to become stale when the selfish miner broadcasts the blocks he has withheld. The following description of the strategy assumes that each block has the same difficulty; we describe the necessary modifications for chains of variable difficulty in section 4. It is parameterized by α, the fraction of the total hash power controlled by the selfish miner, and γ, the fraction of honest hash power that mines on top of a selfish block if there is a race between honest and selfish blocks (that is, the network influence of the selfish miner, when competing blocks are broadcast at the same time). Figure 1 shows the algorithm that a selfish miner will use to determine whether or not they publish their block.

A selfish miner will have "more than their fair share" of blocks added to the canonical blockchain; however, this alone is insufficient to be profitable. As long as the difficulty of the mining puzzle remains constant, the selfish miner loses with this strategy, but honest miners lose even more. Only when the difficulty adjusts downward does the strategy become profitable in the traditional sense. Most existing work captures this idea of the

Algorithm 1: Selfish-Mine

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

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

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

Figure 1: The original selfish mining strategy [7]

share of blocks added to the canonical blockchain by the selfish miner as "relative revenue". For example, if the selfish miner has 35% of the hash rate but mines 40% of the blocks that are accepted by the network, their relative revenue is 0.40. However, fewer total blocks will have been mined per unit of time (before the difficulty adjusts), and the selfish miner will mine fewer blocks than they otherwise would have, had they mined honestly.

In this work, we take the impact of time into account in order to create a revenue metric with more applicability to the real world. To do so, we adjust the relative revenue by the ratio of the elapsed time of the attack over the expected time for honest mining, and then use the percent change relative to the selfish miner's α. This metric, which we call time-adjusted relative gain (TARG), more accurately captures the revenue benefit to a selfish miner compared to mining honestly. Note that, consistent with prior literature, we sometimes conflate this measure of revenue with profitability in the text; in this case, we do so here to avoid confusion with the term "relative revenue". However, cryptocurrency mining is a low margin business, so even a small increase in revenue can correspond to a substantial increase in profitability.

The hash rate where a selfish miner can improve their relative revenue above and beyond what they would earn from mining honestly is given in the following formula. If $\gamma = \frac{1}{2}$, selfish mining is profitable when $\alpha \geq \frac{1}{4}$, and if $\gamma = 0$, selfish mining is profitable when $\alpha \geq \frac{1}{3}$.

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

2.2 Difficulty Adjustment Algorithms

Because proof-of-work cryptocurrencies do not have a central authority that determines who can mine and at what rates, the total amount of hash power supplied to the network will vary over time. However, to maintain a planned monetary policy and better user experience, new blocks should be found at predictable times regardless of the hash power (for instance, Bitcoin targets 10 minute block intervals). Without a difficulty adjustment algorithm (DAA), an increasing hash rate would have blocks found more and more frequently, inflating the currency more quickly and making payments less predictable and secure. The role of the DAA is to change the difficulty of the mining puzzle to adjust to changes in hash rate in order to have blocks be produced at a constant rate.

While the primary purpose of a DAA is to keep the interblock arrival times consistent over the long term in the face of fluctuations in hash rate in order to enforce the monetary policy of the cryptocurrency, there are a variety of other considerations that may go into its design. For example, the DAA should avoid sudden difficulty changes when the hash rate remains constant, discourage wild oscillations from the feedback between hash rate and difficulty, and avoid exceptionally long intervals between new blocks.

2.3 Time and Timestamps

Maintaining accurate clocks in distributed systems is a challenging problem, but relatively accurate timekeeping is a prerequisite for having the difficulty adjustment algorithm maintain its desired interblock arrival time in cryptocurrencies.

Some cryptocurrencies have different timestamp rules, but the ones studied here are largely identical. There are three notions of time that a node cares about: system clock time, the block timestamp, and the network adjusted time. When nodes connect, they each send a timestamp to each other. A node's network adjusted time is the local system clock time plus the median offset of the reported timestamps from all nodes they are connected to, bounded by a maximum 70 minute adjustment in either direction away from the system time. Monero is the only cryptocurrency studied here that doesn't use network adjusted time.

Because block timestamps are the only times that nodes can objectively agree upon, it is these timestamps that are used in DAA calculations. There are two rules to determine whether a node will consider a block valid based on its timestamp:

- The block timestamp must be less than 2 hours ahead of the network adjusted time (or in Monero's case, the system clock time).
- The timestamp must be greater than the median timestamp of the previous 11 blocks.

Together, these rules should prevent the block timestamps from diverging more than a couple hours from real time, and provide nodes with an agreed-upon notion of time for the purposes of difficulty adjustments. However, if the DAA is poorly designed (or poorly implemented), malicious miners may be able to strategically set block timestamps that "confuse" the algorithm and rapidly drive the difficulty down, allowing them to mine blocks more quickly than intended by the coin's monetary policy. The is known as a timewarp attack, and has successfully been executed on several cryptocurrencies, inflating the coin supply.

Another possible attack that utilizes timestamps is the timejacking attack, which takes advantage of the network adjusted time. By connecting to a target node multiple times and reporting incorrect timestamps, an attacker that maintains more than half of the target's connections can move the victim's network adjusted time forward or backward by up to 70 minutes. This can be used to force a target node to temporarily consider a block valid or invalid, depending on the block timestamp and the direction the attacker moves the victim's clock [5].

3 Related Work

Since the initial selfish mining papers [2, 7] were published, extensive work has been done to investigate selfish mining attacks in a variety of scenarios.

3.1 Selfish Mining

While it was generally known that selfish mining wasn't profitable until a difficulty adjustment occurred, this was proven in [12]. This may be why no selfish mining attacks have been seen against Bitcoin; although Neudecker and Hartenstein did notice that the observed frequency of short block intervals between two consecutive blocks mined by the same miner is conspicuously large, which could be caused by selfish mining, it was more likely other factors [25].

Nayak et al showed that a variety of "stubborn mining" strategies could enhance miner profits beyond naïve selfish mining [24]. Furthermore, combining these strategies with an eclipse attack (partitioning a node from the remainder of the network) could enhance these profits, and counterintuitively even benefit the eclipsed "victim". Sapirshtein et al further improved the strategy by using a Markov Decision Process to derive optimum selfish mining strategies, and showed that using an optimal strategy, miners can lower the fraction of the total hash rate needed to be profitable from 25% to 23.21% [31].

Others have studied the performance of selfish mining with more detailed models or realistic environments. For instance, [10] included block propagation delays in their model, and showed that γ can increase dramatically as the variance in propagation delays increases, which further increases the effectiveness of selfish mining. Carlsten et al show that selfish mining is more effective in an environment where there is a large maximum block size and transaction fees secure the chain instead of mostly being secured by a block subsidy [6]. In this environment, selfish miners will tend to build bigger blocks and thus collect more in fees. Gervais et al incorporate block propagation times, block size, expected block times, and the possibility of eclipse attacks into their model, and show that larger block sizes and shorter expected block times increase the relative revenue of selfish miners, but that advanced block propagation techniques can minimize this [9].

The above studies only consider models where a single selfish miner exists, but others expand this to consider multiple selfish miners acting simultaneously [3, 11, 20, 21]. A formalization of multiple selfish miners is provided in [21], which concludes that Bitcoin's security is further degraded when more than one selfish miner exists. For example, with two independent selfish miners, the threshold for profitably selfish mining drops to 21.48% [3]. The more selfish miners that exist, the lower the threshold for profitability, according to [20], which also demonstrates a Nash equilibrium of multiple selfish miners earning disproportionate rewards if they have equal and sufficiently large hash rates. The simulations from [11] show that a few selfish miners, each with $\alpha > 0.1$, can simultaneously earn a $10 - 30\%$ relative revenue increase, but that relative revenue decreases when there are too many selfish miners at once.

Ethereum, unlike Bitcoin, has a different consensus mechanism that includes the existence of stale blocks ("uncles") as part of its difficulty adjustment algorithm and reward scheme. The "uncle blocks" mechanism in Ethereum lowers the threshold for selfish mining profitability, because their own stale blocks still give the selfish miner some reward, making the strategy less risky. Ritz and Zugenmaier showed that, using the observed uncle block ratio from Ethereum in December 2017, the profitability threshold for selfish mining was $\alpha = 0.185 \pm 0.012$ [30]. Niu and Feng's Markov model found that with $\alpha > 0.163$, selfish mining is profitable, and that beneath this value, the selfish miner loses far less than they would on Bitcoin [26]. Furthermore, the revenue for both the selfish miner and the honest miners increase with α due to uncle rewards, and thus could lead to higher inflation of the Ether asset. Finally, [13] more formally analyzes Ethereum's susceptibility

to selfish mining, and proposes new variants of the strategy.

There are other mining attacks that are related to but distinct from selfish mining. For example, the Fork After Withholding attack presented in [19] involves withholding a proof of work solution from the mining pool that the attacker belongs to, and then only propagating it when an external honest miner publishes their own solution, creating deliberate forks. This strategy is always profitable, and is in fact a dominant strategy for large pools to attack small pools with. Coin-hopping is another attack where an adversarial miner switches from mining one coin to another at the beginning of an epoch, leaving honest miners with a higher difficulty chain, and then switches back when the difficulty decreases [22]. This allows a miner to mine at the lowest cost possible while saddling more "loyal" miners with a higher electricity bill.

3.2 Countermeasures

In their initial paper, Eyal and Sirer propose that honest miners, when presented with two competing chains, choose randomly instead of prioritizing the first one seen [7]. This is the equivalent of setting $\gamma = 0.5$, and thus makes selfish mining unprofitable if $\alpha < 0.25$. If the attacker didn't have that high of a γ initially, this countermeasure will improve the selfish miner's performance. Heilman proposes a technique, Freshness Preferred, where instead of miners accepting the first seen block, they accept the block with the most recent timestamp from a trusted source embedded within it [14]. He suggests using the NIST Randomness Beacon [16] for "unforgeable timestamps", which raises the profitability threshold for selfish mining to 0.32. ZeroBlock attempts to prevent selfish mining by having miners append "dummy" blocks to the tip of their local chain if they haven't seen a new block within a certain period of time [31]. Zhang and Preneel propose a backwards compatible defense against selfish mining that punishes miners for publishing their blocks later by changing the fork choice rule from most work to one that accounts for competing blocks that were seen later [32]. The primary disadvantage of the scheme in [32] is that it would take longer for the network to recover from a partition.

3.3 Difficulty Adjustment Algorithms

There has been surprisingly little research performed on difficulty adjustment algorithms, despite their prominent status in proof of work cryptocurrencies.

In [8], the security of the Bitcoin blockchain and its Nakamoto Consensus algorithm are proven in the context of blockchains where the difficulty varies. Kraft shows that Bitcoin's DAA performs poorly when the hash rate is growing exponentially, leading to blocks being found too quickly [17]. In [27], the Bitcoin DAA is shown to be susceptible to a collapse in the rate of finding new blocks if Bitcoin's price decreases significantly in a short period of time. The original CryptoNote DAA was analyzed in [28], finding that it was susceptible to a time warp attack, which was then fixed. Aggarwal and Tan investigated the behavior of miners switching their hash power between Bitcoin and Bitcoin Cash to take advantage of Bitcoin Cash's temporary "Emergency Difficulty Adjustment" algorithm [1]. Hovland and Kucera take a "feedback control engineering" approach to DAAs, and show that if the difficulty adjusts every block, it will quickly adjust to "disturbances" in the hash rate [15]. Furthermore, they claim that moving averages are optimal for reducing random noise while maintaining this quick response. The same authors propose an alternative DAA that reduces the difficulty if it has been a long time since the last block was seen, in order to reduce the long tail of interblock arrival times [18], which is similar to how the Bitcoin Testnet works. Finally, [4] proposes Bobtail, a DAA that would dramatically reduce the variance in block arrival times and would reduce the profitability of selfish mining, but would require substantially increasing the size of block headers (3KB for Bitcoin, as opposed to 80 bytes).

Table 1: Algorithm notation

Notation	Meaning
$chain[x]$	The block in the blockchain at height x
$chain[-1]$	The top (most recent) block in the blockchain, or the chain tip
$chain[x:y]$	The blocks in the blockchain from height x to height y
$chain[x:]$	The blocks in the blockchain from height x to the chain tip.
	If x is negative, then returns the most recent $\|x\|$ blocks
$sumWork(chain)$	A function that will compute the total work of every block in the chain
$filter$(x)	Bound a value x to a minimum and maximum value, dependent upon the DAA used
$len(chain)$	Return the length of the chain
$median(chain)$	Return the block with the median timestamp of the chain
$expectedTime$	Target block solving time $*$ Lookback period
α^+	The lowest α where TARG > 0

4 Methodology

We implemented a simulator that uses Monte Carlo methods in order to establish the profitability of selfish mining against a variety of DAAs.[2] The DAAs selected correspond to those used in the top proof of work cryptocurrencies by market capitalization with the exception of Ethereum, which uses a proof of work consensus mechanism that is more complicated and thus out of scope. The coins considered here are Bitcoin, Bitcoin Cash, Litecoin, Monero, Dash, and Zcash. Inter-block arrival times are modeled as an exponential distribution, consistent with the literature. For each variant of each DAA, we ran simulations for all even α's between 0.06 and 0.48, all γ's between 0 and 0.9 by multiples of 0.05 (although this paper only reports on 0 and 0.5 due to space considerations), and timestamps reported honestly, 1 hour in the future, or 2 hours in the future. For each of these parameter sets, we ran 30 simulations of 10000 blocks each, and averaged the results.

While we selected DAAs based on their use in specific cryptocurrencies, it is important to note that the results from our simulations do not necessarily reflect the realities of mining on those chains. Factors beyond the DAA that impact consensus are not accounted for in our models. As such, the results here should not be broadly applied to the cryptocurrency as a whole, but rather just the DAA employed. The notation used in describing the various DAAs in the paper are in Table 1.

In particular, for Bitcoin Cash, we do not consider the reorg protection added to the Bitcoin ABC client[3] (one of several prominent Bitcoin Cash implementations), which added a proof of work penalty for reorgs of three blocks or more in depth, and won't reorg more than ten blocks by default. Enforcing these rules may reduce the efficacy of selfish mining, but we consider this out of scope, because our focus is on the DAA in isolation. This conveniently allows us to sidestep the more philosophical question of whether consensus-related client-side mitigations that may not be universally enforced constitute a consensus rule for the cryptocurrency system as a whole. For Dash, we do not consider the ChainLocks feature which finalizes blocks and prevents reorgs, and would thus largely neutralize the impact of selfish mining.[4] These techniques have their own tradeoffs, and may make it easier for an attacker to cause permanent or longer-lasting chain splits, despite protecting against selfish mining.

Our simulator makes a number of simplifying assumptions: 1) constant block reward, 2) constant hash rate (no new miners come online or disappear), 3) there is no propagation delay for blocks, 4) the exchange rate of the cryptocurrency remains constant despite the attack, and 5) only one selfish miner (or mining pool) exists. Furthermore, we do not take

[2] https://github.com/usnistgov/SelfishMiningSim
[3] https://github.com/Bitcoin-ABC/bitcoin-abc/releases/tag/v0.18.5
[4] https://github.com/dashpay/dips/blob/master/dip-0008.md

into account how honest miners would respond upon detecting selfish mining. In theory, the honest miners may be able to take actions that would reduce the effectiveness of selfish mining. However, existing research suggests that selfish mining tends to be more profitable when multiple miners apply the strategy simultaneously, so we do not believe that this exaggerates our results [3, 11, 20]. To bootstrap each simulation, we assume there exists a sufficiently long chain of blocks with a constant baseline difficulty of 1, where each block timestamp matches the expected interblock arrival time.

A limitation of our simulator is in the case where honest miners mine a block on top of a selfish miner's block after the attacker won a block race. With constant difficulty, the probability of this event should be determined solely by γ. When the difficulty changes every block, however, it could be the case that the tie-breaking block's required difficulty will differ depending on the timestamp in the racing blocks. In this case, the winner of the tie depends on both γ and the two difficulties, but our simulator only accounts for γ. This also adds a strategic aspect to the choice of which chain tip to mine on for the honest miners, because the first seen one isn't necessarily the "best" choice.[5] Fixing this would require redefining γ, but we leave this for future work; for this paper, the reader should think of γ as representing a range $(\gamma - \epsilon_1, \gamma + \epsilon_2)$ for an unknown ϵ_1 and ϵ_2. It is unclear to us whether this limitation biases the results in any particular direction, but due to the possible strategic aspects, we ignore this effect. This should have no impact on results when $\gamma = 0$, however, if we make the typical assumption that miners mine on top of the first valid block they see.

Another limitation arises with respect to timestamps on simulated blocks. We make the unrealistic assumption that all miners have their local system clocks synchronized. In the networks studied in this paper, a miner will not accept a block as valid if its timestamp is more than 2 hours (7200 seconds) past either its network adjusted time or the local system clock. Thus, in real world scenarios, if a miner puts a timestamp 7200 seconds past their system time into their block, it is likely that some other miners will consider that block invalid and would not mine on top of it until the block is reconsidered and the timestamp is within those bounds. Incorporating this factor into the simulation would significantly complicate the analysis. Our results only attempt a maximum of 7200 seconds of timestamp manipulation in each block. However, combined with a timejacking attack, even longer manipulations are possible, so a clever attacker may be able to profit even more. This is similar to an Eclipse attack, which increases profit from selfish mining [24], and would be a promising avenue for future research. In addition, when our selfish miners manipulate timestamps, they do this for every block they mine, even though there may be more intricate and profitable strategies than this.

Crucially, we needed to alter the selfish mining strategy described above in order to accommodate chains with variable difficulty. In traditional selfish mining models, where difficulty is assumed to be constant, each block has an equal impact on the state change. In our setting, where each block has a different difficulty, a new notion of state is required. Because we take the variable difficulty of blocks into account, our selfish mining strategy does not perfectly match existing literature, nor do we claim to have found the optimal strategy. The details of the adjusted algorithm are in Algorithm 1.

The metrics that we used to evaluate the selfish mining strategy are relative revenue (RR), time-adjusted relative revenue (TARR), and, most importantly, the time-adjusted relative gain (TARG), defined as follows:

$$RR = \frac{blocks\ mined\ by\ selfish\ miner\ that\ make\ the\ main\ chain}{total\ blocks\ mined\ on\ main\ chain}$$

$$TARR = \frac{RR}{\frac{elapsed\ time}{expectedTime}}$$

$$TARG = \frac{TARR - \alpha}{\alpha}$$

[5] For instance, whichever block requires higher difficulty might be the superior choice, because the lower difficulty block would lose if there was another race, or the low difficulty one might be best just to be able to solve it faster in expectation.

Algorithm 1 Selfish Mining: Chains of Variable Difficulty

DEFINITIONS:
$effectiveState := sumWork(privateChain) - sumWork(mainChain)$
$ifLose := effectiveState - nextMainChainBlockDifficulty$
INIT:
$effectiveState \leftarrow 0$
$ifLose \leftarrow -(nextMainChainBlockDifficulty)$
ON SELFISH MINER FINDS BLOCK:
append new block to private chain and continue mining on private chain
$effectiveState \leftarrow effectiveState + newPrivateBlockDifficulty$
$ifLose \leftarrow ifLose + newPrivateBlockDifficulty$
ON OTHER MINERS FIND BLOCK:
append new block to public, main chain
$effectiveState \leftarrow effectiveState - newPublicBlockDifficulty$
$ifLose \leftarrow ifLose - newPublicBlockDifficulty$
if $ifLose \leq 0$ **and** $len(privatechain) > 0$ **and** $effectiveState > 0$ **then**
  publish private chain, overtake main chain, and mine on top of new public chain tip
else if $effectiveState = 0$ **and** $len(privatechain) > 0$ **then**
  publish private chain and enter race
else if $ifLose > 0$ **then**
  continue mining on private chain
else if $len(privatechain) = 0$ **then**
  mine on top of new public chain tip
end if

Recall that we frequently use the term "profitability" when we actually mean revenue, in order to reduce the possible confusion with the term "relative revenue". In the tables of data throughout this paper, if no timestamp is mentioned, the result displayed is the most significant one, regardless of the timestamp used.

5 Results

5.1 Bitcoin and Litecoin

Unlike the other DAAs investigated in this paper, Bitcoin's DAA does not adjust the difficulty level each block. Instead, it changes every 2016 blocks by looking at the timestamps for the first and last blocks of that period, taking the difference, dividing by 1209600 (the expected number of seconds for the interval, or two weeks), and then limiting the difficulty change to at most a factor of 4 in either direction, if necessary. The actual Bitcoin implementation has an off-by-one error; it is clear that the intention was to include the timestamp of the last block of the previous period rather than the first block of the existing period, but it would require a hard fork to fix this error. Litecoin uses the same DAA, but fixed the off-by-one bug, and scaled back their target block time from Bitcoin's 600 seconds to just 150 seconds (and thus divides by 302400, or approximately 3.5 days).

In Bitcoin and Litecoin, we found the selfish mining strategy to be far riskier than prior work would suggest, and substantially less lucrative than selfish mining against other algorithms. We suspect that the primary reason for this is that approximately 20% of the simulation occurs before the first difficulty adjustment, which acts as a "loss leader" for the strategy against these two coins (if the simulator mined more than 10000 blocks, the strategy would be more profitable). When γ is zero, selfish mining was not profitable until α was approximately 0.42 (6.31% $TARG$); even the relative revenue doesn't exceed α until

Table 2: TARG results of Bitcoin. Default parameters in gray.

	$\gamma = 0$			$\gamma = 0.50$		
Lookback	TARG (α^+)	$\alpha = 0.40$	$\alpha = 0.48$	TARG (α^+)	$\alpha = 0.40$	$\alpha = 0.48$
2016/OB1	6.31% (0.42)	-3.03%	37.56%	0.59% (0.32)	17.73%	48.89%
2016	6.54% (0.42)	-3.36%	37.7%	1.69% (0.32)	17.63%	47.94%
Litecoin	5.38% (0.42)	-5.03%	38.59%	1.31% (0.32)	17.56%	47.48%
1008	7.33% (0.40)	7.33%	53.3%	1.61% (0.30)	25.8%	63.64%
288	6.63% (0.38)	14.22%	69.14%	0.93% (0.28)	32.04%	74.43%
144	6.71% (0.38)	14.66%	72.15%	0.33% (0.28)	32.66%	75.42%
60	0.64% (0.36)	18.15%	75.54%	1.73% (0.28)	35.23%	79.54%
24	1.41% (0.36)	21.23%	78.74%	1.59% (0.28)	35.65%	80.49%
12	5.15% (0.36)	24.18%	79.93%	3.22% (0.28)	39.18%	82.83%

$\alpha \geq 0.38$. With $\gamma = 0.5$, the strategy becomes profitable with approximately 1/3 of the hash rate. For both coins, a miner with 48% of the hash rate and no network influence, the selfish mining strategy is approximately 37.5% more profitable than honest mining. This increases to 46.7% and 47.5% for Bitcoin and Litecoin, respectively, when $\gamma = 0.5$. We hypothesize that if block propagation time were factored into the simulation, Litecoin would be more susceptible to selfish mining than Bitcoin due to shorter block intervals.

We also performed the same simulations for Bitcoin, but with the off-by-one bug fixed and lookback periods of 2016, 1008, 288, 144, 60, 24, and 12 blocks, in order to see if difficulty responsiveness could be improved without making the coin more susceptible to selfish mining. The results are in Table 2. By itself, fixing the off-by-one bug had no discernible impact. Unfortunately, any substantial reduction in the lookback period results in selfish mining becoming dramatically more profitable. It may be the case that a lookback of 1008 could be a compromise between selfish mining resilience and the ability to recover from sudden hash rate collapses: with $\gamma = 0$, selfish mining became profitable when $\alpha = 0.4$, with a 7.33% $TARG$.

Timestamp manipulation was, overall, not an effective strategy for increasing the profitability of selfish mining against Bitcoin's DAA. Despite a slightly higher TARG from fiddling with block timestamps, fewer of the simulation runs did better than honest mining when α was small, which suggests a high variance in effectiveness and thus high risk in applying the strategy. For lookback periods of 1008 and 288, manipulation has an ambiguous impact. For lookback of 144 blocks, timestamp manipulation provides a similar and sometimes better RR, but lowers TARG. For lookback of 60, 24, or 12, timestamp strategies improve RR but dramatically reduce TARG, with a decreasing impact as α increases.

Summary: Selfish mining is probably a less significant risk to Bitcoin in practice than has been emphasized in the literature, due to the lengthy period leading up to the difficulty adjustment. Furthermore, timestamp manipulation strategies are either ineffective or risky with little benefit. Reducing the lookback period to 1008 blocks (approximately 1 week) can be done with only a small sacrifice in selfish mining resilience.

5.2 Bitcoin Cash (D601)

Bitcoin Cash adopts a very different DAA from Bitcoin, which adjusts every block and is parameterized by both a lookback period and the "MedianTimePast" (MTP), with a default lookback period of 144 blocks and MTP of 3 blocks. Bitcoin Cash uses the same 600 second expected block time as Bitcoin. The algorithm is a slightly modified simple moving average of the difficulties over the prior 144 blocks (or more generally, the lookback period), with changes bounded by a factor of 2. Algorithm 2 shows how to calculate the next required difficulty.

Algorithm 2 Bitcoin Cash DAA

Require: chain : array of blocks
Require: MTP : number of blocks to check for the median timestamp of
Require: LOOKBACK : number of blocks in difficulty adjustment window
Require: filter() : bounds input between expectedTime/2 and 2*expectedTime
$topBlocks \leftarrow chain[-MTP:]$
$bottomBlocks \leftarrow chain[-(LOOKBACK + MTP) : -LOOKBACK]$
$topMed \leftarrow median(topBlocks)$
$botMed \leftarrow median(bottomBlocks)$
$actualTime \leftarrow filter(topMed.timestamp - botMed.timestamp)$
$newDiff \leftarrow sumWork(chain[botMed : topMed]) * \frac{expectedTime}{actualTime}$

With a default lookback period of 144 blocks and MTP of 3 blocks, a selfish miner with $\gamma = 0$ becomes profitable when $\alpha = 0.34$, with a TARG of 6.22%, which increases to 30.9% when $\alpha = 0.4$, and 76.75% when $\alpha = 0.48$. If the selfish miner has $\gamma = 0.5$, then the strategy becomes profitable when $\alpha = 0.26$, with a TARG of 2.5%. This increases to 41.72% when $\alpha = 0.4$, and 80.47% when $\alpha = 0.48$.

In addition to the default parameter set, we ran simulations for (lookback, MTP) of (12, 3), (24, 3), (24, 11), (48, 3), (48, 11), (48, 17), (144, 11), (144, 17), (144, 45), (288, 3), (288, 11), (288, 17), (288, 45), (288, 73), (2016, 3), (2016, 11), (2016, 17), (2016, 45), and (2016, 505). While we expected the profitability of selfish mining to be highest with shorter lookback periods, the strategy performed worst for a lookback of 12, improved as the lookback period increased to 288, and then became less profitable at 2016. Increasing the MTP forces the selfish miner to have a greater hash rate to be profitable, and significantly lowers the profitability when α is low. As α increases, the protective effect of higher MTPs becomes smaller. Timestamp manipulation may sometimes make up some of the difference for the selfish miner, but their best options get worse as MTP increases. Finally, the MTP should not be set too high relative to the lookback period; doing so leads to wild oscillations and difficulty drops, because the top and bottom blocks are more likely to be close together.

The effectiveness of manipulating timestamps depends on the parameter set. With default parameters, reporting dishonest timestamps severely decreases the profitability of selfish mining. When MTP = 3, timestamp manipulation tends to be profitable for lower lookback periods and harmful for larger ones, but the optimal timestamp isn't necessarily the one furthest in the future. Furthermore, while consistently leading to higher RR with short lookback periods, manipulation tended to have a larger positive impact on the TARG for lower α, and a more muted impact when α was large. This suggests that selfish mining with timestamp manipulation could be a profitable strategy for smaller miners on a coin that uses relevant difficulty algorithm parameters, like a shorter lookback period.

Interestingly, manipulating timestamps was more consistently beneficial with MTPs greater than 3, which we did not expect. Our intuition was that a smaller MTP would make it easier for the selfish miner to "capture" the median timestamp and decrease the difficulty of their next private block to a greater degree. It's possible that this is still the case, but because the lower difficulty also makes their private chain have less work, it counts against the selfish miner. The relationship between DAA parameters and the effectiveness of timestamp manipulation requires more study, but since a higher MTP made selfish mining less effective, it is still recommended to use a higher MTP.

Results for various parameter sets can be found in Table 3. The parameter set that was most resistant to selfish mining, particularly when $\gamma = 0.5$, was (12, 3). The selfish miner has consistently higher relative revenue even at lower α's, but the strategy is less profitable when time-adjusted. While we do not know for sure why this is, one possibility

Table 3: TARG for Bitcoin Cash DAA. Default parameters in gray. Recommended parameters in blue.

	$\gamma = 0$			$\gamma = 0.50$		
Lookback/MTP	TARG (α^+)	$\alpha = 0.40$	$\alpha = 0.48$	TARG (α^+)	$\alpha = 0.40$	$\alpha = 0.48$
144/3	6.22% (0.34)	30.9%	76.75%	2.5% (0.26)	41.72%	80.47%
144/11	0.78% (0.34)	24.69%	75.58%	0.31% (0.26)	37.37%	78.19%
144/45	4.59% (0.36)	20.33%	72.04%	4.01% (0.28)	34.52%	76.27%
12/3	1.99% (0.36)	16.40%	58.85%	7.92% (0.34)	24.48%	60.45%
288/3	6.52% (0.34)	30.2%	78.74%	3.17% (0.26)	41.93%	81.25%
2016/3	3.73% (0.34)	26.81%	68.39%	1.22% (0.26)	37.81%	72.07%
2016/505	1.67% (0.36)	15.82%	60.07%	1.67% (.28)	34.53%	76.6%

is that selfish mining is risky with short lookback periods because of the high variance in inter-block arrival time, which is smoothed out with larger lookback periods. We have greater confidence in the more modest improvements seen with (2016, 505), which was the best parameter set at resisting selfish mining when $\gamma = 0$.

Summary: The default DAA parameters for D601 as used by Bitcoin Cash are among the least optimized for selfish mining resistance. This could be improved by making the lookback period substantially longer or shorter, as well as increasing the MTP. Timestamp manipulation isn't effective against the default parameters, but is effective against other configurations; however, this is mitigated by the reduced profitability of selfish mining strategies against other configurations and thus shouldn't discourage a change in parameters.

5.3 Dash (Dark Gravity Wave)

Dash uses a DAA dubbed Dark Gravity Wave, which adjusts every block and is parameterized by the lookback period. While the implementation itself is convoluted,[6] it ultimately ends up being a simple moving average that assigns double weight to the most recent block, and bounds changes by a factor of 3.[7] There is also an off-by-one bug in Dash's implementation, so that there is one fewer timestamp counted in the calculation than there should be. Dash targets 150 seconds between blocks, but the off-by-one bug biases this such that it ends up being closer to 159 seconds in practice. To calculate the next block's difficulty, Dash uses Algorithm 3.

Algorithm 3 Dash DAA (Dark Gravity Wave)

Require: chain : array of blocks
Require: LOOKBACK : number of blocks in difficulty adjustment window
Require: OB1 : if the implementation has an off-by-one error, this value is 1. Otherwise, this value is 0.
Require: filter() : bounds input between (old difficulty)/3 and 3*(old difficulty)

$WORK \leftarrow 2 * chain[-1].difficulty + \frac{sumWork(chain[-LOOKBACK:])}{LOOKBACK+1}$

$newDiff \leftarrow filter(WORK * \frac{chain[-1].timestamp - chain[-(LOOKBACK-OB1)].timestamp}{expectedTime})$

For Dark Gravity Wave, we tested the default Dash parameters – that is, a lookback period of 24 blocks and an off-by-one error – as well as lookback periods of 24, 48, 60, 144, and 2016 blocks, with the off-by-one error fixed. The most striking result was the degree to which timestamp manipulation made selfish mining more profitable, even with very low α, as can be seen in Figure 2a.

[6] https://github.com/dashpay/dash/blob/master/src/pow.cpp
[7] https://github.com/zawy12/difficulty-algorithms/issues/31

Table 4: TARG results for Dash. The first column represents α^+ for any timestamp, and the second column represents α^+ when all timestamps are profitable. Default parameters in gray. Recommended parameters in blue.

Lookback	TARG (α^+) any	TARG (α^+) all	$\alpha = 0.40$	$\alpha = 0.48$
$\gamma = 0$				
24/OB1	15.5% (0.12)	0.69% (0.32)	131.38%	97.17%
24	0.53% (0.10)	170.15% (0.24)	143.75%	105.59%
48	1.13% (0.16)	6.59% (0.32)	82.03%	93.31%
60	1.46% (0.18)	7.93% (0.32)	71.98%	91.56%
144	0.62% (0.26)	0.63% (0.30)	43.3%	82.31%
2016	-	4.71% (0.32)	34.51%	71.84%
$\gamma = 0.50$				
24/OB1	11.42% (0.08)	1.94% (0.26)	131.66%	97.1%
24	15.50% (0.08)	156.10% (0.24)	142.29%	105.69%
48	0.24% (0.10)	3.36% (0.24)	85.29%	94.82%
60	1.53% (0.12)	3.46% (0.24)	76.63%	92.93%
144	3.67% (0.20)	0.33% (0.22)	51.57%	84.40%
2016	-	1.88% (0.24)	41.47%	73.8%

With default parameters and a selfish miner with $\gamma = 0$, the selfish mining strategy provides a 15.5% TARG when $\alpha = 0.12$ and timestamps are moved forward by 2 hours. It isn't until $\alpha = 0.32$ that selfish mining becomes profitable without manipulating timestamps, at which point the TARG for this miner becomes 0.69%, 57.06%, or 155.89% for honest timestamps, 1 hour in the future, or 2 hours in the future, respectively. With $\alpha = 0.4$, this changes to 29.98%, 71.74%, and 131.38%, and for $\alpha = .48$, we have 72.71%, 84.17%, and 97.17%. If the selfish miner has $\gamma = 0.5$, then selfish mining with timestamps 2 hours in the future provides 11.42% of extra profitability with α as low as .08. At $\alpha = 0.26$, selfish mining becomes more profitable to the tune of 1.94%, 52.92%, or 152.13% for honest timestamps, 1 hour in the future, or 2 hours in the future, respectively. For $\alpha = 0.4$, these numbers are 38.66%, 75.56%, 131.66%, and for $\alpha = .48$, these numbers rise to 75.16%, 84.72%, and 97.11% profitability increases. Notably, the off-by-one bug provides slight protection against the selfish mining attack. Recall that TARG is scaled to α, which explains why it is lower for the highest α values.

To contextualize these numbers, consider a selfish miner with 32% of the global hash rate for a Dark Gravity Wave coin and no network influence ($\gamma = 0$) that normally collects \$1M per month in revenue, with \$950,000 per month in expenses, resulting in a \$50,000 per month profit. If they were to employ the selfish mining strategy while manipulating timestamps by 2 hours, their revenue would increase to \$2.5589M, for a total profit of \$1.6089M. In this example, while the revenue increased by a factor of 2.5, the actual profits increased by a factor of 32.

Increasing the lookback period to 48, 60, or 144 has a neutral or even a slightly positive influence on selfish mining profitability, as we saw with Bitcoin Cash above. However, this is more than offset by the reduction in the effectiveness of timestamp manipulation as the lookback period increases. With a lookback period of 2016, timestamp manipulation is no longer effective. With shorter lookback periods, as α grows, RR may be higher without timestamp manipulation, but TARR and TARG are significantly higher with timestamp manipulation despite this. Stated differently, while the selfish miner who manipulates timestamps may not get a particularly large fraction of the blocks, their impact on the difficulty will speed up the time between blocks such that they mine more blocks per unit time. Finally, observe that timestamp manipulation appears as though it can be used as a substitute for network power; γ matters far less to the profitability of selfish mining when reporting dishonest timestamps.

We recommend adopting a radically longer lookback period of 2016 blocks in order to mitigate the impact of selfish mining on Dark Gravity Wave coins.

Summary: Dark Gravity Wave with a short lookback period is dangerously susceptible to selfish mining attacks, particularly when combined with timestamp manipulation. Even miners with very low hash rates can profit substantially with this strategy, but this is mitigated as the lookback period increases. Dash's Chainlocks mitigate this issue, but timestamp manipulation is likely still a profitable strategy for miners. This DAA demonstrates that timestamp manipulation should be considered in future analysis of selfish mining.

5.4 Monero

Monero uses a DAA that adjusts for every new block, and is parameterized by a delay period, a quantity of outliers to exclude, and a lookback period. To calculate the next block's difficulty, use Algorithm 4.

Algorithm 4 Monero DAA

Require: chain : array of blocks
Require: LOOKBACK : number of blocks in difficulty adjustment window
Require: DELAY : number of blocks to ignore, starting from the chain tip
Require: OUTLIERS : number of array entries to remove from both ends of the array
  $window \leftarrow chain[-(LOOKBACK + DELAY) : -DELAY]$
  for block in window **do**
    $timestamps.append(block.timestamp)$
  end for
  $timestamps \leftarrow timestamps.sort()$
  $filteredTimestamps \leftarrow timestamps[OUTLIERS : LOOKBACK - OUTLIERS]$
  $filteredWork \leftarrow window[OUTLIERS : LOOKBACK - OUTLIERS]$
  $newDiff \leftarrow \frac{sumWork(filteredWork) * expectedTime}{filteredTimestamps[-1] - filteredTimestamps[0]}$

Monero has an expected inter-block arrival time of 120 seconds, a delay of 15 blocks, 60 outliers (which means that 120 total outliers are actually removed), and a lookback period of 720 blocks. When $\gamma = 0$, a selfish miner has a 3.96% TARG for $\alpha = 0.36$, a 19.15% TARG for $\alpha = 0.4$, and a 68.99% TARG for $\alpha = 0.48$. If $\gamma = 0.5$, the selfish miner can get a TARG of 3.67% when $\alpha = 0.28$, 33.84% when $\alpha = 0.4$, and 74.0% when $\alpha = 0.48$. Timestamp manipulation usually provides a small benefit to the selfish miner with these default parameters.

In addition to the default (15, 60, 720) parameter set, we tested performance for the following parameter sets: (60, 60, 720), (15, 120, 720), (60, 120, 720), (15, 60, 240), (5, 20, 240), (5, 20, 2160), (15, 60, 2160), and (45, 180, 2160). Simulation results for each parameter set are in Table 5. Holding the lookback period constant at 720, changing the other parameters had at best a minor impact on the profitability of selfish mining. Lowering the lookback period to 240 without changing the delay or number of outliers removed led to a slight reduction in selfish mining profitability; however, scaling all parameters down by an equal factor made the algorithm substantially more susceptible to selfish mining, particularly when reporting dishonest timestamps. The higher lookback period of 2160 blocks was most resistant to selfish mining, and scaling all parameters up by a factor of 3 was the best parameter set tested. That said, the differences between most parameter sets were not large enough to warrant a specific recommendation.

For a lookback period of 720 blocks, timestamp manipulation was often helpful, but not substantially so. For the (15, 60, 240) parameter set, timestamp manipulation had an ambiguous but small impact. In contrast, the (5, 20, 240) set gave the selfish miner a

Table 5: TARG results of Monero, DOL = Delay/Outliers/Lookback. Default parameters in gray. Most resistant parameters in blue.

	$\gamma = 0$			$\gamma = 0.5$		
DOL	$\alpha = 0.36$	$\alpha = 0.40$	$\alpha = 0.48$	$\alpha = 0.28$	$\alpha = 0.40$	$\alpha = 0.48$
15/60/720	3.96%	19.15%	68.99%	3.67%	33.84%	74.00%
60/60/720	4.16%	18.29%	68.48%	3.99%	33.28%	72.52%
15/120/720	4.32%	18.64%	69.07%	4.31%	33.87%	72.66%
60/120/720	3.02%	18.36%	67.63%	3.18%	32.44%	71.97%
15/60/240	2.53%	18.07%	70.0%	3.02%	32.94%	73.25%
5/20/240	11.70%	31.24%	76.91%	8.64%	46.72%	81.02%
5/20/2160	2.23%	17.1%	62.24%	1.95%	29.81%	65.10%
15/60/2160	1.52%	16.0%	61.73%	1.73%	29.52%	66.32%
45/180/2160	0.46%	14.06%	60.42%	1.48%	28.8%	66.06%

substantial advantage when setting timestamps 2 hours in the future. When the lookback period was 2160 blocks, timestamp manipulation had a small benefit to the selfish miner when α was large, but not to a substantial degree, for the (15, 60, 2160) and (45, 180, 2160) parameter sets. For (5, 20, 2160), timestamp manipulation consistently provided a slight benefit to the selfish miner. Short delay periods and removing fewer outliers tended to exaggerate the impact of timestamp manipulation in favor of the selfish miner.

Summary: Monero's DAA (and its variants) was fairly resistant to selfish mining, and there wasn't substantial room for improvement. By including a delay and removing outliers, the actions of selfish miners take time to have an impact on the difficulty, thus reducing the ability of the miner to benefit from short term strategic manipulations.

5.5 Zcash (Digishield v3)

The Digishield v3 DAA is fairly popular, and is most notably used in Zcash. This algorithm is a "tempered" simple moving average, and is parameterized by the lookback period, "MedianTimePast" (MTP), a damping factor, and maximum upward and downward adjustments. The next block's difficulty is calculated using Algorithm 5.

The default parameters in Zcash are (lookback, MTP, damping, maxup, maxdown) = (17, 11, 4, 16, 32), with a target 150 second block interval. We also ran simulations for the following (lookback, MTP) pairs with damping of 2, 4, or 8, and maxup and maxdown of 16 or 32: (17, 3), (17, 11), (144, 3), (144, 11), (144, 45), (2016, 3), (2016, 11), and (2016, 505). The results can be found in Table 6.

With Zcash's default parameters and $\gamma = 0$, TARG turns positive when $\alpha = 0.32$ and timestamps are manipulated by 1 or 2 hours, with gains of 1.53% and 8.06%, respectively. When $\alpha = 0.36$, the TARG is positive for all timestamps: 5.05%, 23.44%, and 35.35%. For $\alpha = 0.4$, the TARGs are 22.58%, 46.30%, and 62.0%, and for $\alpha = 0.48$, this increases to 76.03%, 84.36%, and 87.59%. If the selfish miner has $\gamma = 0.5$, using timestamps 2 hours in the future achieves a TARG of 1.51% when $\alpha = 0.24$. It isn't until $\alpha = 0.28$ that selfish mining is profitable without timestamp manipulation, but manipulation helps substantially: for honest, 1 hour ahead, and 2 hour ahead timestamps, selfish mining generates TARGs of 4.14%, 11.66%, and 16.67%, respectively. When $\alpha = 0.4$, the TARG increases to 37.48%, 60.23%, and 80.43%, and for $\alpha = 0.48$, the TARG is 76.66%, 85.35%, and 89.21%.

Clearly, timestamp manipulation is extremely effective against Zcash's current DAA, as can be seen in Figure 2b. However, making the maxup and maxdown parameters symmetric - or even allowing faster upward adjustments and slower downward ones – negates this effect. When (maxup, maxdown) is (16, 16), timestamp manipulation is sometimes slightly effective, but it doesn't lower the needed α for profitability. Timestamp manipulation

Algorithm 5 Zcash DAA (Digishield v3)

Require: chain : array of blocks
Require: LOOKBACK : number of blocks in difficulty adjustment window
Require: MTP : number of blocks to check for the median timestamp of
Require: DAMPING : reduce the impact of time variations by this factor
Require: MAXUP : used to determine minTimespan for maximum upward adjustment
Require: MAXDOWN : used to determine maxTimespan for maximum downward adjustment
Require: filter() : bounds input between minTimespan and maxTimespan

$B \leftarrow chain[-1]$
$A \leftarrow chain[len(chain) - LOOKBACK - 1]$
$bMedian \leftarrow median(chain[B.height - MTP : B.height])$
$aMedian \leftarrow median(chain[A.height - MTP : A.height])$
$actualTimespan \leftarrow bMedian.timestamp - aMedian.timestamp$
$actualTimespan \leftarrow expectedTime + \frac{actualTimespan - expectedTime}{DAMPING}$
$minTimespan \leftarrow expectedTime * \frac{100.0 - MAXUP}{100.0}$
$maxTimespan \leftarrow expectedTime * \frac{100.0 + MAXDOWN}{100.0}$
$actualTimespan \leftarrow filter(actualTimespan)$
$avgTarget \leftarrow 0.0$
for block in chain[B.height-LOOKBACK: B.height] **do**
  $avgTarget \mathrel{+}= \frac{1.0}{block.difficulty}$
end for
$avgTarget \leftarrow \frac{avgTarget}{LOOKBACK}$
$bnNew \leftarrow avgTarget * \frac{actualTimespan}{expectedTime}$
$newDiff \leftarrow \frac{1.0}{bnNew}$

is ineffective for (32, 32) and extremely counterproductive for (32, 16). However, selfish mining performance without timestamp manipulation is nearly identical regardless of the maximum adjustments. For otherwise default parameters, lowering the MTP to 3 made selfish mining significantly more profitable, especially with a lower α. The damping factor has a strange impact on the effectiveness of timestamp manipulation for otherwise default parameters: an increase in the damping factor protects against selfish mining with 1 hour future timestamps, but makes selfish mining far more profitable than otherwise with 2 hour future timestamps. A lower damping factor has little impact in this case.

Adjusting the lookback period to 144 blocks but otherwise keeping default parameters has comparable resistance to selfish mining as the default lookback of 17 blocks, but it negates the impact of timestamp manipulation. With this higher lookback, increasing the damping factor to 8 makes the DAA slightly more resistant to selfish mining and makes timestamp manipulation counterproductive. Decreasing the damping factor to 2 makes selfish mining slightly more profitable and makes timestamp manipulation mildly effective. With a lookback of 144 and a lower MTP of 3, selfish mining is more effective, but timestamp manipulation still doesn't help. When the MTP is 45 blocks, selfish mining is less effective, but timestamp manipulation is sometimes slightly more profitable. In either case, the various combinations of maximum adjustments had only a small impact on the profitability of selfish mining, without an easily discernible pattern.

Increasing the lookback period to 2016 blocks but using otherwise default parameters, the DAA is substantially more resistant to selfish mining as well as timestamp manipulation. Decreasing the MTP to 3 makes selfish mining more profitable, but is still a dramatic improvement over the default. When MTP = 505, selfish mining is noticeably less profitable. With (lookback, MTP) of (2016, 11), lowering the damping factor to 2 makes selfish mining

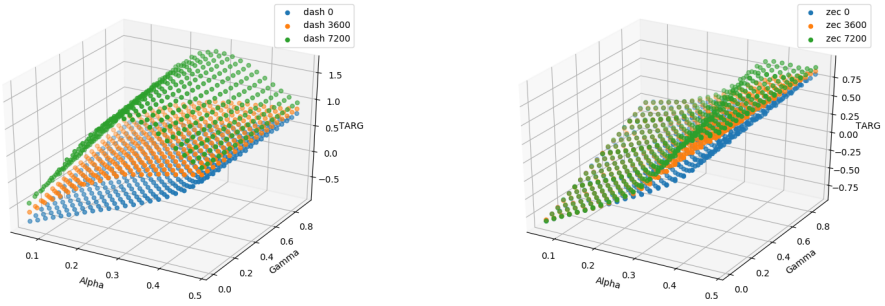
Table 6: Zcash TARG for select parameter sets. LMDUN = Lookback, MTP, Damping, maxup, maxdown. The first column represents α^+ for any timestamp, and the second column represents α^+ when all timestamps are profitable. Default parameters in gray. Recommended parameters in blue.

LMDUN	TARG (α^+) any	TARG (α^+) all	$\alpha = 0.40$	$\alpha = 0.48$
$\gamma = 0$				
17/11/4/16/32	8.06% (0.32)	35.35% (0.36)	61.99%	87.59%
17/11/4/32/32	6.08% (0.36)	6.08% (0.36)	22.71%	74.21%
17/3/4/16/32	0.73% (0.28)	4.55% (0.34)	72.60%	92.64%
144/11/4/16/32	6.30% (0.36)	6.30% (0.36)	23.75%	73.40%
144/11/2/16/32	0.56% (0.34)	9.03% (0.36)	24.78%	75.05%
144/11/8/16/32	5.46% (0.36)	5.46% (0.36)	21.97%	71.28%
144/3/4/16/32	3.57% (0.34)	3.57% (0.34)	27.39%	74.39%
144/45/4/16/32	4.80% (0.36)	4.80% (0.36)	20.35%	72.16%
2016/505/8/32/16	2.04% (0.42)	2.04% (0.42)	-3.17%	25.10%
$\gamma = 0.50$				
17/11/4/16/32	1.51% (0.24)	16.67% (0.28)	80.43%	89.21%
17/11/4/32/32	0.06% (0.26)	3.93% (0.28)	36.66%	78.19%
17/3/4/16/32	5.37% (0.22)	22.18% (0.26)	86.85%	94.74%
144/11/4/16/32	0.66% (0.26)	4.63% (0.28)	36.47%	76.47%
144/11/2/16/32	0.42% (0.26)	4.90% (0.28)	38.03%	78.36%
144/11/8/16/32	3.84% (0.28)	3.84% (0.28)	35.03%	73.15%
144/3/4/16/32	0.96% (0.26)	0.96% (0.26)	39.09%	78.26%
144/45/4/16/32	0.28% (0.26)	4.65% (0.28)	34.53%	75.78%
2016/505/8/32/16	1.58% (0.36)	1.58% (0.36)	8.36%	28.26%

significantly more profitable, and increasing it to 8 makes selfish mining far less profitable. More specifically, when the damping factor is 2 and $\gamma = 0.5$, the TARG turns positive when $\alpha = 0.28$. For $\gamma = 0$, gains begin at the same α as it would with a damping factor of 4 (that is, $\alpha = 0.38$), but with a far higher TARG. In contrast, with a damping factor of 8 and $\gamma = 0.5$, gains start at $\alpha = 0.36$, and for $\gamma = 0$, the TARG doesn't become positive until $\alpha = 0.42$. These improved results are comparable to Bitcoin's level of selfish mining resistance. The parameter set most resistant to selfish mining was (2016, 505, 8, 32, 16), although the results for other maximum adjustment values are comparable. In this case, the Digishield DAA was even more resistant to selfish mining than Bitcoin, which is notable.

While the interactions between parameters are somewhat complex in Digishield, there are a few patterns. Increasing the lookback period improves the algorithm's resistance to selfish mining, first by making timestamp manipulation less effective, and then in general when the lookback period is sufficiently long. Increasing the MTP is also protective. When lookback = 17, decreased damping has little impact, but increased damping makes 2 hour timestamps significantly more profitable (and 1 hour timestamps less profitable than with damping = 4). This changes when lookback = 144, where decreased damping is slightly harmful and increased damping is slightly protective. When lookback = 2016, lower damping makes selfish mining more profitable, and a higher damping factor offers significant protection.

The maximum adjustment parameters had very little impact when the lookback period was increased. This makes sense, because each new block contributes a smaller amount of work relative to the total, and thus would adjust the difficulty by a smaller amount. The result is that these filters don't come into play as often. For higher lookback periods, these parameters would need to be scaled down significantly in order to be relevant. That said, it does appear that reducing the maximum downward adjustment helps prevent selfish mining



(a) The significance of manipulating the timestamp for Dash.

(b) The significance of manipulating the timestamp for Zcash.

Figure 2: Timestamp manipulation effectiveness in regards to α and γ.

by forcing the difficulty to take longer to adjust to the selfish miner's suddenly "missing" hash rate. Similarly, a higher maximum upward adjustment can reduce the profitability by allowing the difficulty to "recover" from the selfish miner's "absence" more quickly. In the case of Zcash, the asymmetric fast adjustment down with slow adjustments back up is what makes timestamp manipulation so effective, and making the adjustments symmetric closes this attack vector. Unfortunately, changing the maximum adjustment parameters makes the coin more susceptible to sudden hash rate drops as well as fly-by-night miners that rapidly change which coin they are mining based on profitability ("coin-hopping").

Summary: The Digishield DAA has many parameters that interact with each other in potentially complex ways. In general, a higher MTP is protective against selfish mining, and a higher damping factor is also protective – but only when the lookback period is increased. The Zcash developers should consider making the maximum adjustment parameters symmetric in order to improve selfish mining resistance with minimal changes to the algorithm.

6 Discussion

Algorithm comparison: Table 7 summarizes the above results for the default parameter sets for each algorithm. We emphasize again that the results here are for the DAA itself, and not necessarily the coin employing it, because some cryptocurrencies like Bitcoin Cash and Dash have other mitigations in place that may make selfish mining more challenging or less profitable.

That said, it is clear from this comparison that Bitcoin forces the selfish miner to have a substantially larger fraction of the global hash rate in order to be profitable, and with a substantially lower TARG when it is profitable, than other coins' algorithms. With 40% of the Bitcoin hash rate and no network influence, a selfish miner will still lose money (over 10000 blocks); for the next best competitor, Monero, the same selfish miner increases its time-adjusted revenue by 19.15%. However, the gap tends to close as the selfish miner's network influence increases. On the other side of the spectrum are Dash's Dark Gravity Wave algorithm and zCash's Digishield, with DGW in particular being highly susceptible to timestamp manipulation. Bitcoin Cash's D601 algorithm performs somewhere in between.

We believe that the wildly divergent results across algorithms and their parameterizations lends support to the idea that selfish mining is fundamentally an attack on the DAA itself.

Table 7: TARG results of each coin with default parameters. The first column represents α^+ for any timestamp, and the second column represents α^+ when all timestamps are profitable.

Coin	TARG (α^+) any	TARG (α^+) all	$\alpha = 0.40$	$\alpha = 0.48$
$\gamma = 0$				
Bitcoin	6.31% (0.42)	6.31% (0.42)	-0.87%	37.56%
Bitcoin Cash	6.22% (0.34)	6.22% (0.34)	30.9%	76.75%
Dash	15.5% (0.12)	155.89% (0.32)	131.38%	97.17%
Monero	3.80% (0.36)	3.80% (0.36)	19.15%	68.99%
Zcash	8.06% (0.32)	35.35% (0.36)	61.99%	87.59%
$\gamma = 0.50$				
Bitcoin	0.59% (0.32)	4.63% (0.34)	17.73%	46.89%
Bitcoin Cash	2.5% (0.26)	2.5% (0.26)	41.72%	80.47%
Dash	11.42% (0.08)	152.13% (0.26)	131.66%	97.11%
Monero	3.67% (0.28)	3.67% (0.28)	33.84%	74.0%
Zcash	1.51% (0.24)	16.67% (0.28)	80.43%	89.21%

Timestamps: Our results indicate that timestamp manipulation and a cryptocurrency's timestamp validity rules must be considered when choosing a DAA, and particularly when considering how the DAA will perform against selfish mining attacks. Timestamp manipulation in some cases can significantly reduce the hash rate required by a selfish miner in order to profit, as observed for variants of Dark Gravity Wave, Digishield, and some non-default variants of Bitcoin Cash's DAA. This means that even small miners may attempt to use the selfish mining strategy, to the network's detriment. In some cases, even, selfish miners may perform poorly based on the relative revenue metric used in prior research, but still have a positive *time-adjusted* gain – an inversion of the standard selfish mining results as have been historically applied to Bitcoin.

While we expected timestamp manipulation to be advantageous to the selfish miner in many cases, we were surprised to see that naively setting timestamps 2 hours into the future was not universally the best strategy, even when manipulation was beneficial. That is, there were cases where setting timestamps 1 hour into the future outperformed 2 hours for the selfish miner; this suggests that there may be a rich strategy space for miners to consider when setting block timestamps.

Furthermore, we propose that the combination of block timestamp manipulation and strategically timejacking nodes can have a synergistic effect for the attacking selfish miner. Recall that timejacking is a variant of Sybil attack where nodes report faulty system times in order to change a target node's "network adjusted time". Combining this with block validity rules requiring blocks to be no more than 2 hours ahead of the network adjusted time can allow a miner to artificially increase their γ; adjust a target node's clock backwards, and blocks that have future timestamps will be considered temporarily invalid and won't be relayed to other nodes or built on top of by timejacked miners.

Lookback periods and MedianTimePast: We had originally hypothesized that shorter lookback periods would be more susceptible to selfish mining, and that the effectiveness of selfish mining would decrease as the lookback period lengthened. However, our results indicate that that both short and long lookback periods can be protective, and that selfish mining is most profitable somewhere in the middle. This was the case for Bitcoin Cash, Dash (with honestly reported timestamps), and to a lesser extent, Monero (when the delay period and number of outliers removed remains constant). One possible reason for this is that when lookback periods are sufficiently short, there will be a larger observed variance in interblock arrival times, and this gets "smoothed out" when the lookback period increases. This higher variance for short lookback periods may make selfish mining riskier. Therefore, increasing the lookback period may at first help a selfish

miner due to reducing this variance, but beyond a certain point, longer lookback periods dilute the impact of the selfish miner on the difficulty, reducing profits.

We expected that increasing the MTP would improve an algorithm's resistance to selfish mining, which indeed was the case. What we did not expect, however, was for timestamp manipulation to be more effective for higher MTPs. It is unclear to us why this is the case, but the reduced profitability of selfish mining in general from higher MTPs outweighs this effect.

Finally, we note that there exists a relationship between the lookback period and MTP. For instance, we simulated selfish mining against the Bitcoin Cash DAA with lookback of 12 blocks and MTP of 11 blocks, and the difficulty had wild swings of 90% or more in a single block due to sometimes having very little work occur between the median blocks chosen, which may be close together. If using a variant of Bitcoin Cash's D601 algorithm, cryptocurrency designers should be careful not to set these values too close. This was less of an issue for Zcash/Digishield, because of the damping factor.

DAA tradeoffs: This work only investigated the relationship between difficulty adjustment algorithms and selfish mining. However, the choice of DAA impacts many other aspects of a cryptocurrency. In particular, there appears to be a direct tradeoff between resistance to selfish mining and responsiveness to changes in hash rate. The DAA is unable to ascertain, for instance, whether a sudden increase in the time between blocks is due to a selfish miner directing their hash rate to a private chain or if a flood destroyed a miner's hardware. Similarly, the DAA doesn't know if a sudden decrease in the time between blocks is a result of new, honest miners coming online, or a temporary coin-hopping attack.

Cryptocurrency communities should recognize that these tradeoffs may be valued differently at different stages of a coin's "life cycle". For example, coins that share proof of work mining algorithms and have the minority of the hash rate for that algorithm (like Bitcoin Cash at the time of writing), or "ASIC resistant" coins that can be mined with general purpose hardware (like Monero) may consider coin-hopping attacks as more serious than selfish mining, but then switch when they become majority hash rate, when specialized hardware is made for the mining algorithm, or when they become substantially bigger.

7 Conclusion and Future Work

This paper is the first (that we are aware) comparative investigation into the efficacy of selfish mining against a variety of difficulty adjustment algorithms. Furthermore, we have demonstrated that some algorithms are far more susceptible to selfish mining than others, and that selfish miners should include block timestamp manipulation as a new component of their strategy space.

There remain many questions to investigate in future work. For instance, how does a miner determine the optimal timestamp, and are there more advanced strategies than naively setting the timestamp to be an offset from the miner's system time? How does timejacking influence the profitability of selfish mining? What about multiple small selfish miners simultaneously mining on a cryptocurrency where timestamp manipulation dramatically lowers the profitability threshold?

There are also many other potential difficulty adjustment algorithms that can be analyzed, including simple combinations such as Dark Gravity Wave but including a delay (as in Monero). Finally, future work should examine how effective certain mitigations are, such as those currently employed by Bitcoin Cash and Dash.

References

[1] Vipul Aggarwal and Yong Tan. A structural analysis of bitcoin cash's emergency difficulty adjustment algorithm. *Available at SSRN 3383739*, 2019.

[2] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*, 2013.

[3] Qianlan Bai, Xinyan Zhou, Xing Wang, Yuedong Xu, Xin Wang, and Qingsheng Kong. A deep dive into blockchain selfish mining. *arXiv preprint arXiv:1811.08263*, 2018.

[4] George Bissias and Brian Neil Levine. Bobtail: A proof-of-work target that minimizes blockchain mining variance (draft). *arXiv preprint arXiv:1709.08750*, 2017.

[5] Alex Boverman. Timejacking & bitcoin, 2011.

[6] Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.

[7] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[8] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[9] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16. ACM, 2016.

[10] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[11] Jake A Gober. *The Dynamics of a" Selfish Mining" Infested Bitcoin Network: How the Presence of Adversaries Can Alter the Profitability Framework of Bitcoin Mining.* PhD thesis, 2018.

[12] Cyril Grunspan and Ricardo Pérez-Marco. On profitability of selfish mining. *arXiv preprint arXiv:1805.08281*, 2018.

[13] Cyril Grunspan and Ricardo Pérez-Marco. Selfish mining in ethereum. *arXiv preprint arXiv:1904.13330*, 2019.

[14] Ethan Heilman. One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer, 2014.

[15] Geir Hovland and Jan Kucera. Nonlinear feedback control and stability analysis of a proof-of-work blockchain. 2017.

[16] John Kelsey. The new randomness beacon format standard: An exercise in limiting the power of a trusted third party. In Cas Cremers and Anja Lehmann, editors, *Security Standardisation Research - 4th International Conference, SSR 2018, Darmstadt, Germany, November 26-27, 2018, Proceedings*, volume 11322 of *Lecture Notes in Computer Science*, pages 164–184. Springer, 2018.

[17] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9(2):397–413, 2016.

[18] Jan Kucera and Geir Hovland. Tail removal block validation: Implementation and analysis. 2018.

[19] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. 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*, pages 195–209. ACM, 2017.

[20] Tin Leelavimolsilp, Long Tran-Thanh, and Sebastian Stein. On the preliminary investigation of selfish mining strategy with multiple selfish miners. *arXiv preprint arXiv:1802.02218*, 2018.

[21] Francisco J Marmolejo-Cossío, Eric Brigham, Benjamin Sela, and Jonathan Katz. Competing (semi)-selfish miners in bitcoin. *arXiv preprint arXiv:1906.04502*, 2019.

[22] Dmitry Meshkov, Alexander Chepurnoy, and Marc Jansen. Short paper: Revisiting difficulty control for blockchain systems. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 429–436. Springer, 2017.

[23] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[24] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[25] Till Neudecker and Hannes Hartenstein. Short paper: An empirical analysis of blockchain forks in bitcoin.

[26] Jianyu Niu and Chen Feng. Selfish mining in ethereum. *arXiv preprint arXiv:1901.04620*, 2019.

[27] Shunya Noda, Kyohei Okumura, and Yoshinori Hashimoto. A lucas critique to the difficulty adjustment algorithm of the bitcoin system. *Available at SSRN 3410460*, 2019.

[28] Surae Noether and Sarang Noether. Difficulty adjustment algorithms in cryptocurrency protocols. 2014.

[29] RHorning. Mining cartel attack, 2010.

[30] Fabian Ritz and Alf Zugenmaier. The impact of uncle rewards on selfish mining in ethereum. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 50–57. IEEE, 2018.

[31] Siamak Solat and Maria Potop-Butucaru. Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. *arXiv preprint arXiv:1605.02435*, 2016.

[32] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer, 2017.