Not So Predictable Mining Pools: Attacking Solo Mining Pools by Bagging Blocks and Conning Competitors

Jordan Holland, R. Joseph Connor, J. Parker Diamond, Jared M. Smith, and Max Schuchard

Department of Electrical Engineering and Computer Science
UT Computer Security Laboratory
University of Tennessee, Knoxville, TN, USA
{jholla19,rconnor6,jdiamon3,jms,mschucha}@utk.edu
https://volsec.eecs.utk.edu

Abstract. In this paper we present three attacks against the predictable solo mining (PSM) scheme. In PSM, miners receive shares for submitting partially valid solutions to the current Proof of Work, adding those shares to their account. When the pool successfully mines a block, the block is awarded to the miner with the most shares, and the rewarded miner "pays" an amount of shares equal to the next highest miner's to claim the block. Our attacks take advantage of the fact that the amount of shares expended winning two different blocks, which have the *same monetary value*, can vary by up to a factor of *four*. We show that by strategically spreading its shares across multiple accounts, a malicious miner can generate more revenue than a naive miner of the same computational power by only claiming blocks with a low share cost. By doing so, a miner can reduce computational power it must expend to win a block by more than 30%. Our other two attacks reduce the profitability of victim miners in the pool by minimizing the gap between first and second place when the victim wins a block. This drives up the average amount of computational power the victim must contribute to receive a reward. An adversary not concerned with cost can reduce the number of shares a victim retains after winning a block by up to 26%. We also find that an adversary with more computational power than their victim can reduce the number of shares the victim retains after winning a block by more than 8% with only limited impact on the adversary's profitability.

1 Introduction

Cryptocurrency mining represents a several hundred million dollar a year industry. As a point of reference, Bitcoin miners alone generated 563 million US dollars worth of revenue in 2016 [10]. The increasing popularity and profitability of cryptocurrency mining has resulted in an exponential increase in the total computational power dedicated to the task. For example, the cryptocurrency Ethereum has seen a more than 18 fold increase in overall network hashrate between September 2016 and September 2017 [5]. This increase in overall network compute power means that individuals possessing small amounts of computational resources find themselves unable to consistently generate mining revenue. As a result, miners often opt to join mining pools where they aggregate their computational resources in an effort to receive a consistent, reliable income proportional to their contribution to the pool.

The *payout scheme* of a mining pool decides how to distribute the pool's revenue between individual miners. Ideally, a payout scheme should demonstrate *proportional fairness*, where miners receive rewards proportional to their contribution to the pool. Additionally, pool operators want a payout scheme that is *incentive compatible* between themselves and their miners. Specifically, the miner's best strategy should be that they dedicate all of their computational resources towards the pool, resulting in maximized revenue by both miner and pool operator. With the number of different mining pools increasing, so too has the number of different payout schemes that are implemented by these pools. Many of these schemes are not vetted for incentive compatibility or fairness, resulting in attacks against deployed payout schemes which break either proportional fairness or incentive compatibility [17,7,15].

In this paper we present three attacks against a relatively new payout scheme, the predictable solo mining (PSM) scheme. Unlike other payout schemes that split each won block between all mining participants, PSM awards the entire block to the miner that currently has saved the most "computational credit" or *shares*. The winning miner's shares are then adjusted to be the difference between their current shares and the next highest miner's shares. This payment scheme results in a variance in the number of shares "needed" to win a block, even when the pool's computational power is held constant. Our attacks take advantage of the fact that the amount of shares expended winning two different blocks, which have the *same monetary value*, can vary by up to a factor of *four*. By strategically adjusting our malicious miner's behavior, we can exclusively take advantage of highly efficient blocks, while at the same time forcing victims to more frequently accept less efficient blocks. The attacks we develop present evidence that this payout scheme is neither incentive compatible nor fair.

In our first attack, we show that by launching a Sybil attack against a pool using PSM, a malicious miner can generate more revenue than an honest miner of equivalent computational power. Additionally, we show that in an ecosystem of PSM pools, the PSM payout scheme is not incentive compatible, and encourages miners to spread computational resources between several pools to maximize returns of their computational power investment. We find that an adversary can trivially increase their profits by 10%, even if they just interact with a single PSM pool, and can increase their profits by more than 30% when they apply our strategy across multiple PSM pools. This means that miners are incentivized to spread their computational resources across several PSM pools, making the PSM mining scheme not incentive compatible between miner and pool operator. Our second attack shows that miners can *degrade* the profitability of victim miners by driving up the average amount of computational power a victim must invest to receive a reward at a financial cost to themselves. We demonstrate than an adversary can reduce the number of shares retained by a victim after winning a block by up to 26%, reducing the victim's mining efficiency. Lastly, our third attack utilizes an altered variant of our original Sybil attack to degrade competitor profitability while at the same time maintaining the adversary's.

The rest of this paper is laid out as follows: Section 2 discusses the process of mining cryptocurrencies, mining pools, and payout schemes across mining pools; Section 3 examines the attacks we have constructed and lays out how they work; Section 4 discusses the simulator we built and the data we collected to test our various attacks; Section 5 examines our evaluation in carrying out our attacks; Section 6 dives into existing attacks against mining pools and where our contributions lie and the current state of integrity in the cryptocurrency ecosystem; finally, Section 7 summarizes what we have discovered and presents potential future work.

2 Background

2.1 Mining Cryptocurrencies

Cryptocurrency mining is the process of adding transaction records to to the currency's public ledger of past transactions. This ledger of past transactions is called the *blockchain*, and serves as a means to tracking valid transactions in the network. In order to add transaction records to the blockchain, miners must compute a resource-intensive *proof-of-work* (PoW) that serves to provide security guarantees against double-spending and ensure that transactions are tamper-resistant. In exchange for computing the PoW, miners are rewarded with a number of units of currency to compensate the miner [1].

In order to control the rate at which records are added to the blockchain, cryptocurrency networks can adjust the what is considered a valid solution to the PoW in an effort to require miners to spend more computational cycles to find one. The *network difficulty* is a relative measure of how difficult it is to find a new block, where the difficulty is adjusted periodically as a function of how much hashing power has been deployed by the network of all miners for a cryptocurrency.

2.2 Mining Pools

As a result of the large number of miners working on the most profitable cryptocurrencies, new miners often face a high barrier to entry due to the variability associated with successfully computing solutions to PoW. Miners with small hashrates relative to the largest miners have a lesser probability of mining a block, which causes the variance in its overall profitability to increase. Additionally, as the mining difficulty of the network increases so too does the variability, meaning that individual miners can expect to mine for long periods without any reward.

To mitigate this variability in rewards, cryptocurrency miners can join a *mining pool* which aggregates computing power (i.e. collective hashing power), thus lessening the variability in finding blocks. Pools distribute the rewards of successfully mined blocks to their member miners regardless of whether the miner was the actual node to solve the PoW. This scheme allows smaller-scale miners who would otherwise receive only sporadic rewards to instead receive a steady, reliable income. In exchange, pool owners will collect a small fee from each block found by the pool before distributing the remaining rewards to pool participants.

Pools attempt to distribute rewards proportionally to each miner based on their contribution to the pool's overall hashrate. Therefore, pools need a secure manner to estimate the computational power of each miner. To accomplish this, miners occasionally submit *shares*, solutions to the PoW for the current block that are at a lower difficulty than the network difficulty, to their pool. Miners find shares in the process of hunting for a full solution to the PoW. The frequency and difficulty of the shares submitted to the pool serves as a proxy for the miner's computational power. When a block is found, the pool uses a pre-determined *payout scheme* that determines how to allocate the reward of the found block to the miners in the pool based on submitted shares.

2.3 Mining Pool Payout Schemes

Various payout schemes have been used in mining pools, as explored by Rosenfeld et. al. [17]. The pay-per-last-N shares (PPLNS) scheme and predictable solo mining (PSM) are two examples of current mining pool payout schemes. PPLNS is used in the largest Ethereum mining pool [4], `Ethermine.org`, and PSM is used in another major Ethereum mining pool [6], `Ethpool.org`, as well as Bitcoin, Litecoin, ZCash, DogecCoin, ZClassic, Komodo, and Hush solo mining pools [16,2,19,9]. The two schemes work as follows:

– **PPLNS Scheme:** This reward system is round based, where one round is an arbitrary number of minutes. When a block is found by the pool, the block reward is distributed according to the number and difficulty of the shares submitted by each during the last hour. Payout takes place immediately after the minimum payout amount of 1 coin has been reached.
– **PSM Scheme:** In this scheme, each submitted share will increase the credits of the miner who submitted the share by the share difficulty. The miner who accumulates the most credits receives the reward of the next mined block and their credits will be reset to their current credits minus the credits of the runner-up miner. Re-setting the credits of the miner who did receive the block reward to 0 was abandoned as it did penalize miners having an above average hashrate. Typically, a miner will receive a full block reward as soon as their accumulated credits equals the current block difficulty (+/- pool luck) [6].

3 Attacking PSM Pools

In a PSM pool, the entire block reward is awarded to the miner with the most shares any time the pool finds a block, and the winner's shares are then decreased by the number of shares held by the miner with the second-most shares. Therefore, the ”cost” of a block reward can be characterized by the number of shares held by the second-place miner at the time the block is mined by the pool. Likewise, the efficiency of a miner can be characterized by the number of blocks won per share contributed.

The average cost of a block in a PSM pool is equal to the network difficulty [6]. However, because the mining process is random, the time between mined blocks in the pool varies. As a result, the number of shares that miners have when a block is found (and by extension, the cost for the block reward) also varies.

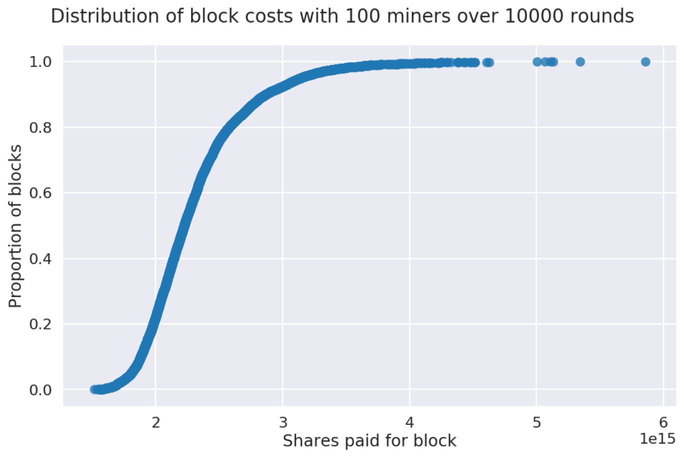


Fig. 1: The average cost per block is equal to the network difficulty (approximately 2.3×10^{15} in Ethereum at the time of writing), but block costs are distributed randomly.

3.1 Minimizing the Cost of Winning Blocks

A key observation to make about a PSM pool is that a miner can never pay more shares for a block than the miner current has in their account. Miners who naively submit all found shares to a single account in a single PSM pool can expect to see block costs drawn from the population seen in Figure 1, resulting in an average cost that is approximately equal to the network difficulty. However, by refusing to place more shares into their account, a malicious miner will only ever win the cheapest cheapest blocks produced by the pool. A miner who uses this strategy can expect to pay less per block, on average, compared to a naive miner with a comparable hash rate.

A miner who employs this strategy may have leftover computing power after filling its account to the target number of shares. When the miner succeeds at filling an account to the target number of shares, the miner in turn begins crediting shares into a new account, filling it the target number. Our adversaries goal is to always have an account at the target number, meaning that if a block would be claimed for that value or lower, they will always win it, paying exactly their targeted cost.

A natural risk of the cost minimization strategy is that blocks at or bellow the adversary's particular cost threshold will be insufficiently frequent. If our adversary can fill accounts to the target number of shares faster than blocks at that cost appear then some of those accounts will be wasted. In other words, although a miner may decrease their average cost per block, they may win fewer total blocks if any of their hashing ability goes unused. Since these shares do not directly contribute to the adversaries revenue, the adversary has no motivation to generate them; we term such shares *withheld shares.*

In order to counteract this loss of throughput the adversary can adopt one of two strategies. First, the adversary can increase the cost it is willing to pay for blocks, reducing its efficiency, but increase its throughput. Alternatively, the adversary can spread computational power across multiple pools, attempting to claim inexpensive blocks from several different pools. As long as the miner can submit shares to alternate pools, it may use the cost minimization strategy without a loss in throughput. In order to show that the cost-minimization strategy is advantageous for miners in practice, we also demonstrate that realistic miners can achieve a significant reduction in per-block cost without withholding too many shares. In particular, for a miner with access to N mining pools, the maximum acceptable proportion of wasted shares is $1 - \frac{1}{N}$.

The total profit of a miner for a given time period can be calculated as:

$$\text{Profit} = \frac{\text{(Shares earned)}}{\text{(Share cost per block)}} \times \text{(Block reward)}$$

Assuming that a miner is never idle (that is, that there are sufficient alternate pools where leftover computing power can be used to earn shares), then an individual miner's shares are roughly proportional to the miner's hashing power (barring the effects of luck), which is assumed to be constant. The block reward is likewise assumed to be constant, as is the case in Ethereum. Therefore, a miner who lowers their average cost per block for a fixed time period can expect a greater total profit during that time.

Mining pools profit from the pool fees that are deducted for blocks mined by the pool, so the pool is solely interested in miners contributing as much hashing power as possible to the pool. If miners can obtain a greater payout by contributing certain shares to alternate pools, then the PSM payout model cannot be incentive-compatible. Furthermore, since the pool cannot expect to find blocks at an average cost that is less than the network difficulty, a single miner who wins blocks at an average cost that is less than the network difficulty necessarily increases the average cost for the other miners in the pool.

3.2 Increasing a Miner's Block Costs by Donating Shares

Malicious miners in PSM pools can abuse the lack of integrity on share submissions in order to inflate a target miner's average block cost. Consider the "cost" of each block as the number of shares that the second place miner has when a block is found. An adversary can submit shares under another miner's public key to the corresponding pool, inflating that miner's share count. Our adversary can increase a victim's costs by artificially closing the gap between the victim and the miner in second place at the time that the victim wins a block. By consistently minimizing the difference between the target miner and this runner up we can effectively define the cost of the block for the target, and increase the target miner's average block cost. Since the malicious miner is donating shares to other miners, this attack will cost the adversary money, while also reducing the victim's profitability.

3.3 Multiple Account Idling to Drive Up Target Miner Block Cost

The previous attack is intuitive and effective, but a malicious miner can do almost as well in driving up the cost of a target miner without donating all of their work to other miners. Furthermore, we can do this even the pool has implemented integrity-checking measures. This attack involves having at least two accounts in the same pool and spreading one's submitted shares among those accounts. Consider the "cost" of a block being the number of shares the second place miner has when a block is found. Increasing the average cost of a target miner then involves minimizing this difference. By driving an attack account up near the top of leaderboard of the pool and then "idling" at a specific rank, the attack account can wait until the target is in range to attempt to position himself in second by a minimum number of shares in the case of a target miner victory.

While idling the attack account, the adversary can offloads extra shares to an offload account, slowly driving that account up the leaderboard. Once the target passes the attacker, the malicious miner then uses all shares it finds to constantly minimize the gap between itself and the target. By riding the target miner up the few spots left in the leaderboard, the attacker is able to minimize the gap between it and the target at the time of the target miner's victory, and thus define the cost of the block for the target.

Once the target miner has won, on a successful attack the attacker should be at the top of the leaderboard at the start of the new block. The attacking account now uses all of its hashing power to win the first block it can. After the attacker wins a block he switches to the offloading account, making it the new attacker account. Due to the offloading of shares while "idling" the new attacking account is in a much better position to get back to the "idling" state where the attack on the target can be launched. By spreading out across multiple accounts (ultimately as many as needed) the attacker is able to almost constantly be in an "attacking" position given a higher hash rate than the target.

4 Experimental Setup

4.1 Simulation Methodology

To evaluate our attacks against the integrity of PSM pools, we have built a generic discrete mining pool simulator, which is publicly available at [11]. We use random probabilistic distributions to model both the rounds in which blocks are found as well as the shares distributed to each miner based on their total hash rate, the average difficulty, and the number of shares we wish each miner to receive per second. Most documented configurations of real-world mining pools can be configured in our simulator by manipulating the following parameters:

- **Number of Honest Miners**: the number of miners which model generic, non-malicious miners that contribute all shares available per round to their own total.
- **Number of Malicious Miners**: the number of miners which behave maliciously in one of several considerations, depending on the chosen attack. These miners are global adversaries and can see all members of the mining pool, including each miner's current shares, and have the ability to make predictions about the next round's accumulated shares for a given miner. Malicious miners can freely distribute their valid shares to other miners or completely separate pools in any manner they see fit. Finally, malicious miners can make accurate guesses on when the next block will occur, and use this information to make informed decisions about what to do with their currently valid shares.
- **Miner hash rates**: the distribution of hash rates both the honest and malicious miners should pull from, which come from both real-world PPNS and PSM pools. The collection of this data will be discussed in the next section, Section 4.2.
- **Number of block rounds to simulate**: the number of blocks our simulator should simulate being found by the combined power of the miners in the pool. Note, there is not necessarily a 1-to-1 mapping of simulated rounds and rounds where blocks are found; in general, there can be thousands of rounds for every block round.
- **Round length**: the round length in simulated seconds that each round should last. This value informs the calculation of the block rounds and the share distribution to each miner per round by influencing the difficulty to find blocks.
- **Shares per second**: the number of shares per second on average that a miner should find. We use this value when calculating share distribution per round for each miner.
- **Network difficulty**: the total difficulty of finding a block for the given pool, which is pulled from the real-world pool being simulated. If we are simulating a PSM pool, we pull this from the Ethpool API; otherwise, if it is a PPLNS pool, we pull this from the Ethermine API.

To represent realistic pools, we use known statistical distributions for modeling both the rounds where blocks will be found and the distribution of shares to miners per round. These distributions are calculated as follows:

- **Finding Blocks**: We model finding blocks using a geometric distribution based on the configured round length, the total pool hash rate as given by summing the hash rates of all honest and malicious miners in the pool, and the average network difficulty of the Ethereum network measured as the number of hashes required on average to find a block. We calculate the probability of finding a block in a given round as:

$$P(finding\ block) = \frac{round\ length \times total\ pool\ hash\ rate}{network\ difficulty} \tag{1}$$

  We then sample n samples from a geometric distribution of Bernoulli trials, where n is the number of blocks we wish to find from the parameters above, and the probability is calculated from Equation 1.
- **Distributing Shares to Miners**: We model the distribution of shares per round to each miner with a binomial distribution based on the miner's hash rate, the desired shares per second, and the configured round length. We first calculate the difficulty of getting shares by Equation 2.

$$D(shares) = \left\lfloor \log_2 \frac{miner\ hash\ rate}{shares\ per\ second} \right\rfloor \tag{2}$$

  Then, we get the actual probability of the miner finding a valid hash in one second by Equation 3 using the difficulty we just computed.

$$P(finding\ hashes) = \frac{miner\ hash\ rate}{2^{difficulty}} \tag{3}$$

  Next, we derive the shares the miner should find each round by sampling from a binomial distribution based on n trials, where n is the round length in seconds, and the probability computed in Equation 3 and multiplying it by $2^{difficulty}$, where the difficulty comes from Equation 2.

This generic simulator gives us a way to simulate the interactions of honest and malicious miners over any arbitrary number of blocks as well as various pool types. In the next section, we discuss our method to collect data from existing, real-world pools to use as our miner hash rates. In the sections following, we discuss how we have used our simulator to find novel attacks on PSM pools that are not incentive-compatible and can be or may already be actively exploited on the largest Ethereum mining pools.

4.2 Collecting Real-World Miner Data

Our simulator uses hash rates and difficulties collected from the Ethpool and Ethermine APIs, where Ethpool is a PSM mining pool and Ethermine is a

PPLNS mining pool. Using the miner addresses that Ethpool and Ethermine use to pay miners, we were able to find miners that were paid by the mining pools on the Ethereum block-chain via geth [8], the official Ethereum command-line interface. The paid miners' addresses were then queried against the Ethpool and Ethermine APIs to locate active miners, their hash rates, and the difficulty of the Ethereum network. At the time of writing, Ethpool has roughly 1,000 active miners, and Ethermine has approximately 40,000 active miners.

5 Evaluation

5.1 Cost Minimization Attack

32 simulations were run for pools of 100 miners for with hash rates drawn at random from the real distribution of miners in Ethpool. In each case, the simulation was allowed to run until the pool found 10000 blocks. In each run, a single miner used the "cost minimization" strategy by mining until reaching a fixed target (75% of the expected block cost) and then stopping. The remainder of the miners naively contributed every share found immediately. Miners whose hash rates were so small as to never win a block for the duration of the simulation are omitted. Results are shown in Figure 2.

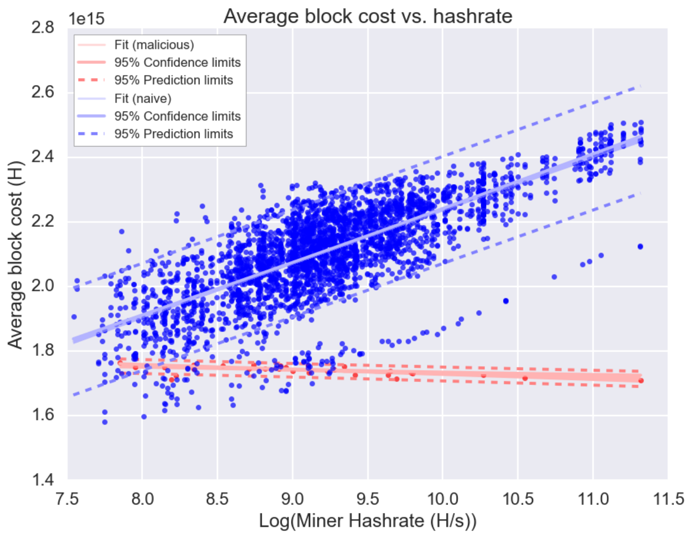


Fig. 2: Average cost per block as a function of miner hash capacity, comparing naive (blue) and malicious (red) miners.

Figure 2 shows that the PSM payout scheme already gives an advantage to miners with lower hash rates. The average block cost for an honest miner increases proportionally with the logarithm of the miner's hash rate. But even many lower-than-average miners can expect to decrease their average cost per block with the cost minimization attack, compared to honest miners with the same hash rate. Unlike normal PSM schemes where miners with higher hash rates are less efficient, our malicious miner becomes more efficient as hash rate grows. This is a result of the miner being able to more rapidly fill accounts to the target share value, allowing the miner to take advantage of several inexpensive blocks in short succession.

A second set of simulations was run to demonstrate that this attack is practical at a significant advantage for a typical miner without reducing throughput, assuming access to only a small number of alternate pools. Each simulation from this set used a fixed set of 100 miners with 1 malicious and the remainder honest. The malicious miner's hash rate for these simulations was fixed at the median hash rate from Ethpool, to represent a typical miner. The malicious miner's target block cost was varied across simulations to observe the relationship between the average block cost and the proportion of shares not submitted to the pool.

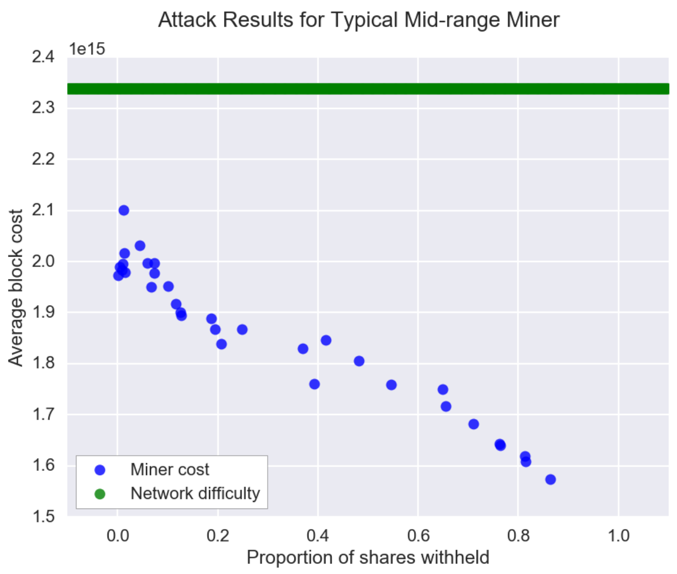


Fig. 3: Average block cost as a function of shares not contributed to the pool for a median Ethpool miner (1.3 GH/s) with network difficulty 2.3×10^{15}. A naive miner would have an average block cost equal to the network difficulty in PSM pools.

Figure 3 shows that a typical miner can expect to reduce their average cost per block from approximately 2.1×10^{15} hashes to 1.8×10^{15} hashes by withholding only 50% of its shares. Assuming the existence of only a single alternative mining pool of comparable size, this malicious miner can achieve this reduction without any wasted hashing power, by contributing the leftover hashing power to the alternate pool.

5.2 Malicious Donation Attack

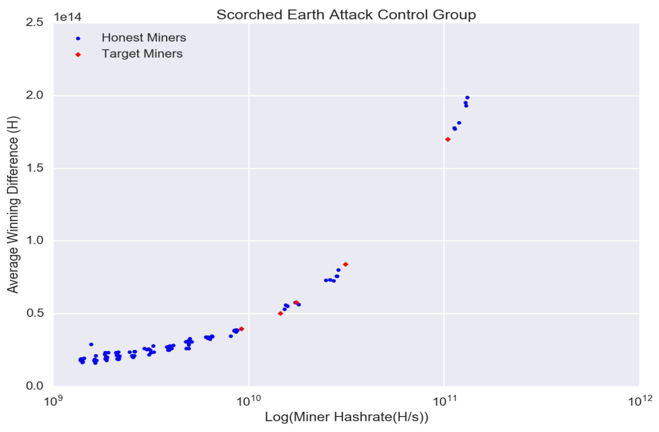
An antagonistic miner can inflate the cost of a victim miner by allocating shares to miners immediately below the victim in shares when the victim is about to win a block in the PSM pool. As the victim miner increases its shares, the malicious miner can give its own shares to the runner-up miner in order to decrease the difference in shares between the victim and the runner-up to 1. This ensures that the victim almost always has no shares to put towards the next block. Under this scenario, the attacker saves no shares for itself and gives shares to any miner so long as it closes the gap between the victim – when the victim is about to win a block – and the next miner.

Figure 4 shows that by using the above strategy, an attacker can consistently force a victim to spend all shares available to win a block, or to put it differently the gap between the victim and the second place miner is always a difference of 1 share. The amount of shares left over after winning a block generally follows an exponential increase with respect to hash rate. The aforementioned figure shows that the victim miner's left-over shares decrease by roughly 26% in the presence of this attack, given this distribution of 100 hash rates. Over time this decrease in the amount of shares available for the next block will ultimately increase the average block cost required for the victim to obtain a block. This assumes that the malicious miner values punishing the victim more than making money, so this study disregards the performance of the attacker under this scenario.

5.3 Idling Attack

A malicious miner can increase the average block cost of a target miner in PSM mining pools even if the pool has integrity measures implemented, provided the attacking miner has a sufficiently large hash rate in comparison with the target. By situating multiple accounts strategically in the pool, the malicious miner can be in second place almost every time the target miner is close to winning a block. This in turn means that the adversary can ensure that the victim miner *always* has their shares reset to zero after winning a block.

Table 1 shows that by using this attack a miner can cause the target to have a smaller average number of leftover shares after winning a block. The data for this table was gathered by running a control group of 100 miner's for 10,000 blocks, 10 times. Afterwards, we ran the same group of 100 miners, taking control of the miner with the highest hashrate in the pool, making it our attacker. Each attack scenario was then ran 10 times for 10,000 blocks, and then averaged out over the different runs. This results in raising the target miner's average block cost, and therefore reduces the miner's profits. The only cost incurred by the malicious miner is when the malicious miner accidentally exceeds the victim's shares essentially attack themselves. Our reported results are with a



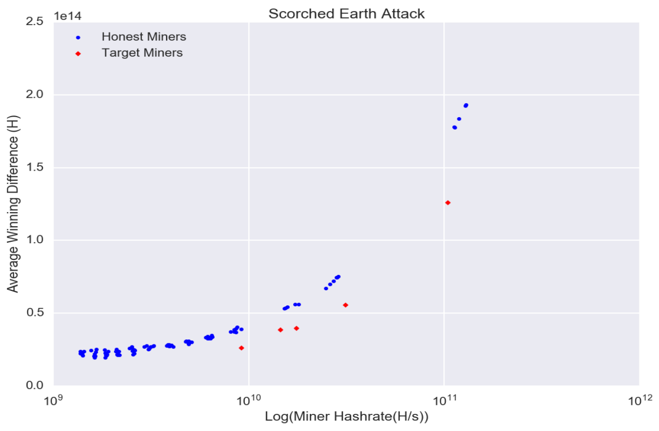


Fig. 4: Average shares remaining after winning a block without (left) and with (right) a scorched-earth attacker

Attacker / Target Ratio	% Decrease in Average Winning Difference
1.2	.03
4.2	5.02
7.5	6.31
9.0	5.6
14.2	8.36

Table 1: The average decrease in a victim's gap when it wins a block by a computationally stronger adversary launching our "idling" attack.

hand optimized algorithm. We believe that with some machine optimization the attacker could further drive up the target's average block cost at a very small overall cost to himself.

6 Related Work

Other attacks have been proposed against different mining pool payout schemes which break incentive compatibility and fairness. Rosenfeld et. al.s's work [17] examines several payout schemes including Pay-Per-Share (PPS) and Pay-Per-Last-N-Shares (PPLNS), exploring their vulnerability to *pool-hopping* attacks. In a pool hopping attack an adversarial miner jumps between several pools hoping to capitalize on instances where any one of those pools successfully mines a block earlier than expected. This attack has a similar end goal to our cost minimizing attack in Section 3.1, however our attack can achieve greater efficiency if multiple pools exist to spread attacker resources across, it can still provide the adversary gains *even if only one pool exists.*

Concurrent work recently published by Zamyatin et. al. [20] also examined PSM pools [1]. Several key differences exist between our work and Zamyatin et. al.'s. First, our attacks utilize different properties than Zamyatin's work. Our cost-based attack described in Section 3.1 successfully wins blocks at a minimum cost, rather than building up a large gap between the malicious miner and the second-place miner. This allows our attacks to be more efficient than Zamyatin's in terms of average work performed per winning block by between 10% to 30%. Increasing other miner's block costs via multiple idling accounts, presented in Section 3.3, is a new attack not proposed by Zamyatin and does not require donating shares to other miners not controlled by the adversary. As a result, our idling attack, unlike their tactical donation attack, can not be trivially defeated by adding authentication to the mining pool. Lastly, in Zamyatin. et. al.'s work, the actual attack simulations were conducted with a mining pool consisting of only *two* miners, a high hashrate and low hashrate miner. As described in Section 4, our work evaluates attacks with realistic simulations based on representative population of miners and hashrates taken the Ethpool API. Our more accurate model better captures the dynamics of what an adversary needs to accomplish to realize the attacks presented in our work.

Besides these related attacks, a wide body of literature on payout schemes exists. Many related papers, for example work by Lewenberg et. al. [14] and Schrijvers et. al. [18], examine cryptocurrency mining pools from a game-theoretic perspective. These works often focus on theoretical constructions of mining pools involving limited players. To our knowledge, no such game theoretic analysis has been conducted specifically on PSM pools. Other studied attacks include denial-of-service attacks [12,13] and withholding attacks [15,3,7], where malicious pools attack rival pools and damage their reward by withholding valid blocks.

[1] In Zamyatin's work, these pools were called *queue-based* pools

7 Conclusions

In this paper we have presented three separate attacks against the predictable solo mining scheme. The variability in the cost of the blocks in this scheme leave it vulnerable to a malicious miner strategically gaining an advantage. First, we showed that a miner that strategically spreads out its shares across multiple accounts can generate more revenue than a naive miner with the same computational power by only claiming blocks below the average share cost. The second attack presented shows that an adversary, not concerned with cost, can reduce the shares a victim retains after winning a block, reducing their profitability. Our third attack on the PSM scheme shows that even on pools with integrity implemented, an adversary can reduce the number of shares a victim retains limited impact to his overall profitability. In conclusion, we recommend the PSM scheme not be implemented in any new pools and any pools currently using the scheme change to another, more incentive compatible and fair, payout scheme.

References

1. Bitcoin.it: Bitcoin Mining (2017)
2. Con Kolivas: Anonymous Solo Bitcoin Mining for Everyone (2017)
3. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv.org (2014)
4. Ethermine.org: Ethermine (2017)
5. Etherscan.io: Ethereum Network HashRate Growth Rate (2017)
6. Ethpool.org: Credits on Ethpool (2017)
7. Eyal, Ittay: The Miner's Dilemma. 2015 IEEE Symposium on Security and Privacy (2015)
8. Go-ethereum: Geth (2017)
9. Hellcatz: Solo Mining Pool for ZCash, Hush, Komodo, Zclassic, and Zen (2017)
10. Hileman, G., Rauchs, M.: Global cryptocurrency benchmarking study. Cambridge Centre for Alternative Finance (2017)
11. Holland, J., Connor, J., Diamond, P., Smith, J., Schuchard, M.: aminingpoolsimulator - Mining Pool Simulator. `https://github.com/VolSec/aminingpoolsimulator` (2017)
12. Johnson, B., Laszka, A., Grossklags, J., Vasek, M.: Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. Lecture Notes in Computer Science pp. 72–86 (2014)
13. Laszka, A., Johnson, B., Grossklags, J.: When bitcoin mining pools run dry. International Conference on Financial . . . (2015)
14. Lewenberg, Yoad, Bachrach, Yoram, Sompolinsky, Yonatan, Zohar, Aviv, Rosenschein, Jeffrey S: Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis. International Foundation for Autonomous Agents and Multiagent Systems (May 2015)
15. Luu, L., Saha, R., Parameshwaran, I.: On power splitting games in distributed computation: The case of bitcoin pooled mining. 2015 IEEE 28th Computer Security Foundations Symposium (2015)
16. NiceHash: NiceHash Solo Mining Pool for Multiple Currencies (2017)
17. Rosenfeld, M.: Analysis of Bitcoin Pooled Mining Reward Systems (Dec 2011)

18. Schrijvers, O., Bonneau, J., Boneh, D.: Incentive compatibility of bitcoin mining pool reward functions. ... Conference on Financial ... (2016)
19. TBDice: Anonymous solo Litecoin and Dogecoin mining pool based on ckpool (2017)
20. Zamyatin, A., Wolter, K., Werner, S., Mulligan, C.: Swimming with Fishes and Sharks: Beneath the Surface of Queue-based Ethereum Mining Pools. sba-research.org (September 20, 2017)