Uncle Traps: Harvesting Rewards in a Queue-based Ethereum Mining Pool

Sam M. Werner
Imperial College London
London, United Kingdom
sam.werner16@imperial.ac.uk

Paul J. Pritz
Imperial College London
London, United Kingdom
paul.pritz18@imperial.ac.uk

Alexei Zamyatin
Imperial College London and SBA Research
London, United Kingdom
alexei.zamyatin17@imperial.ac.uk

William J. Knottenbelt
Imperial College London
London, United Kingdom
wjk@imperial.ac.uk

ABSTRACT

Mining pools in Proof-of-Work cryptocurrencies allow miners to pool their computational resources as a means of reducing payout variance. In Ethereum, *uncle blocks* are valid Proof-of-Work solutions which do not become the head of the blockchain, yet yield rewards if later referenced by main chain blocks. Mining pool operators are faced with the non-trivial task of fairly distributing rewards for both block types among pool participants.

Inspired by empirical observations, we formally reconstruct a Sybil attack exploiting the uncle block distribution policy in a queue-based mining pool. To ensure fairness of the queue-based payout scheme, we propose a mitigation. We examine the effectiveness of the attack strategy under the current and the proposed policy via a discrete-event simulation. Our findings show that the observed attack can indeed be obviated by altering the current reward scheme.

CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**.

KEYWORDS

Ethereum Mining Pools, Discrete-event Simulation, Mining Pool Attacks

ACM Reference Format:
Sam M. Werner, Paul J. Pritz, Alexei Zamyatin, and William J. Knottenbelt. 2019. Uncle Traps: Harvesting Rewards in a Queue-based Ethereum Mining Pool. In *12th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2019), March 12–15, 2019, Palma, Spain.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3306309.3306328

1 INTRODUCTION

As the first decentralised cryptocurrency, Bitcoin [14] has successfully demonstrated how incentive mechanisms can be applied to reach agreement in a dynamically changing set of pseudonymous and mutually distrusting participants. The underlying consensus protocol, termed *Nakamoto consensus*, builds on a random leader election process where participating nodes are required to invest computational power in solving cryptographically hard, memoryless and non-invertible puzzles. The latter is referred to as *Proof-of-Work* (PoW). The process of searching for solution candidates is termed *mining*, with participating nodes being referred to as *miners*. Each miner that generates a valid solution to the PoW puzzle becomes the leader and is allowed to determine the set of unconfirmed transactions. These are then appended in a data structure referred to as a *block* to the *blockchain*, a publicly accessible and immutable distributed ledger. The PoW is a partial pre-image attack on SHA256 in Bitcoin, where generating a valid solution via brute-forcing is hard, while verifying a hash against a pre-image is easy. To incentivise honest participation and compensate miners for the computational effort invested, the leader is rewarded a predefined amount of newly minted units of the underlying cryptocurrency.

Unlike Bitcoin, Ethereum [2], the cryptocurrency with the second highest market capitalisation[1], has substantially faster block generation intervals. This results in the more frequent occurrence of so-called forks, where multiple blocks/PoW solutions are generated around the same time and compete for becoming the head of the chain. To incentivise miners whose blocks did not become part of the main chain to extend the head of the main chain instead of working on their fork, Ethereum introduced the notion of *uncle blocks*. Instead of restricting the ancestors of a block to one, Ethereum follows an *inclusive* approach [12] where main chain miners can reference uncle blocks. For each such referenced uncle block, additional rewards are distributed to both the main chain miner and the creator of the uncle block.

Due to the memorylessness of PoW puzzle, the number of solutions found per time interval follows a Poisson process. Hence, miners collude to form *mining pools* with the aim of reducing the variance of received payouts. In a mining pool, participants jointly work towards finding PoW solutions and share the earned revenue according to some reward distribution policy. Usually, such organisations are run by a single operator, who maintains the necessary infrastructure, monitors the contribution of each miner and determines how and where the computational power of the pool is to

VALUETOOLS 2019, March 12–15, 2019, Palma, Spain

ACM ISBN 978-1-4503-6596-3/19/03…$15.00
https://doi.org/10.1145/3306309.3306328

[1] https://coinmarketcap.com. Accessed: 09-30-2018

be employed. In order to track the amount of work performed by miners in a pool, operators ask miners to submit *shares*. These are PoW solutions with a lower difficulty than for the network. Based on the number of shares a miner submits over a specific time period, the miner's *hash rate* can be derived. Hash rate refers to the number of unique attempted PoW solutions generated over a period of time.

Mining pools not only embody a form of centralisation within a decentralised network [7, 9], but are also faced with the non-trivial task of fair reward distribution. In Ethereum mining pools, the latter problem also extends to the distribution of uncle block rewards, which is the focus of this paper. As a result, several different reward payout schemes have been implemented by mining pool operators, further examined in [11, 13, 15, 16].

In this paper, we focus on mining pools that employ a queue-based reward payout scheme. We show how an adversary may leverage this uncle reward distribution mechanism in order to increase her expected payouts. Additionally, we discuss an observed reward-increasing strategy employed by a miner in a queue-based mining pool and reconstruct the observed attack via a discrete-event simulation to examine the effect on the mining pool. We also propose a modification to the studied uncle reward distribution policy, which would obviate reward-increasing strategies rooted in the former.

The remainder of this paper is structured as follows. Section 2 provides a more detailed overview of reward allocation in Ethereum and of mining pool reward schemes. In Section 3, we show and discuss the existence of reward-increasing opportunities in the uncle reward distribution mechanism of a queue-based mining pool. Section 4 proposes a mitigation policy. We examine the effects of the reconstructed attack in Section 5, prior to concluding in Section 6.

2 BACKGROUND

2.1 Block Rewards in Ethereum

In Ethereum, each full block is currently rewarded with a static reward of 3 ETH[2], and any fees paid by users for transactions included in the block. Block arrival times follow a Poisson distribution with the rate parameter $\lambda = \frac{H}{D}$, where H is the network's total hash rate and D the network difficulty of the PoW. In Bitcoin, the PoW difficulty is adjusted every two weeks to maintain a target block generation interval of 10 minutes. In Ethereum the difficulty is adjusted dynamically after every block while the target block interval amounts to approximately only 15 seconds [2].

Multiple PoW solutions found roughly around the same time can create two (or more) parallel competing branches in the underlying blockchain due to network latency [5, 6]. Eventually, the branch supported by the majority of the computational power in the network becomes the main chain, i.e. the sequence of blocks accumulating the most PoW effort since the genesis block. All other branches are discarded and receive no rewards. To achieve a faster convergence to a single chain, Ethereum leverages a reward scheme for forked blocks, similar to the notion of *inclusive blockchain protocols* [12]. Miners of main chain blocks can reference forked blocks, which are then referred to as uncle blocks. Each referenced uncle block results in an additional reward being distributed to both the main chain miner and its creator. Participants mining on conflicting branches are hence incentivised to rejoin the main chain, as they are guaranteed reimbursement for otherwise wasted computational effort.

Each block included in the main chain can reference up to two uncle blocks and receive a small reward of $\frac{1}{32}$ of a full block reward per referenced uncle. However, the reward paid to the miner of each referenced uncle block varies. The reward diminishes depending on how distant of an ancestor the uncle block is relative to the main chain block it was referenced by. An uncle block is rewarded for being up to six generations away from the included block. Hence, the reward is computed as

$$U = (U_n + 8 - B_n) \cdot \frac{B}{8}, \tag{1}$$

where U is the uncle block reward, U_n the uncle block number, B_n the number of the block included in the main chain, and B the reward for a full block [3].

2.2 Mining Pool Reward Schemes

Mining pools allow individual miners to reduce their payout variance by pooling together their computational resources with other miners. Computational effort of a miner is tracked via the submission of shares, where each share has a probability $\frac{d}{D}$ of being a valid solution to the network PoW, with d being the share difficulty. Over the years, mining pools have introduced a range of innovative reward schemes, the majority of which have been formally examined by Rosenfeld [15]. In the following subsections we briefly compare some of these existing schemes.

2.2.1 Proportional Payout Scheme. In this simple scheme, a miner is rewarded proportionally to the number of shares submitted in the time interval between two blocks found by the pool. Consequently, a block reward B is split between the N miners in the pool based on their respective number of shares submitted. Rosenfeld [15] shows this reward scheme to be susceptible to various forms of manipulation.

2.2.2 Pay-Per-Last-N-Shares. The *Pay-Per-Last-N-Shares* (PPLNS) scheme distributes a reward proportional to the last N shares that have been submitted. Resilience advantages of PPLNS are more closely examined by Rosenfeld [15].

2.2.3 Queue-based Payout Scheme. Ethpool[3] was the first Ethereum mining pool to introduce a queue-based reward scheme. Under such a reward scheme, miners accumulate *credits* for each share submitted to the pool operator. Each time a full block is mined by the pool, the block reward is allocated to the miner in the pool with the highest accumulated credit balance. The top miner then has her credit balance reset to the difference between her own and the second highest credit balance in the pool.

2.3 Mining Pool Attacks

While mining pools are intended to benefit pool participants through a more steady payout stream, they are subject to different types

[2]ETH is the underlying unit of exchange in Ethereum

[3]https://ethpool.org

of attacks. Previous research [4, 13] has shown that *block withholding* attacks, whereby a malicious miner withholds blocks from the pool, yet still receives a reward from the pool operator for her overall performed work, can cause mining pools to suffer tremendous financial losses. Vasek, Thornton and Moore [17] have shown how mining pools are negatively affected by Distributed Denial of Service (DDoS) attacks. Laszka, Johnson and Grossklags [10] have examined game-theoretic aspects of attacks between mining pools.

Primarily, adversaries will try to increase their own expected payouts at the cost of other miners by exploiting vulnerabilities of the underlying reward scheme. With regards to the queue-based payout scheme, different attack strategies aimed at exploiting the non-uniform credit reset mechanism have been identified. Zamyatin et al. [18] propose attack scenarios following real-world observations in Ethpool, whereby an adversary may strategically donate her computational power to other miners in the pool in order to manipulate the queue constellation with the aim of receiving larger credit differences. These attacks were further studied by Holland et al. [8]. However, the aforementioned attacks, both explicitly focus on the non-uniform credit reset mechanism and ignore any potential vulnerabilities stemming from the uncle reward distribution policy of the pool.

3 ATTACK DESCRIPTION, OBSERVATION AND RECONSTRUCTION

We propose a Sybil attack that leverages the reward-increasing opportunities under a scheme of random distribution of uncle block rewards. According to this scheme, uncle block rewards are distributed randomly among miners in the pool without accounting for differences in the work performed. The selected miner receives the full uncle block reward. For instance, a miner with 2 GH/s has the same chance of receiving a found uncle block as a miner with 10 MH/s.

The attack entails the division of hashing power between a set of smaller miners and the deliberate increase of the pool uncle rate. We introduce the notion of *uncle traps*. These describe miners with conspicuously small hash rates that serve the sole purpose of increasing the likelihood of receiving uncle block rewards in the aforementioned scheme.

We define a fair pool, where fairness refers to the expected reward of an individual miner being proportional to the shares submitted to the pool by that miner in relation to the total shares submitted to the pool. We use the notion of a *round* as the time interval between two blocks found by the pool – these may be uncle blocks or full blocks.

3.1 Attack Modelling

3.1.1 Definitions. Define D as the network difficulty and B and U as the block and uncle block rewards, respectively. The fairness assumption means that a miner with hash rate h, mining in a pool of N miners, where the total hash rate of the participating miners is

$$H = \sum_{i=1}^{N} h_i \quad , i = 1, \ldots, N ,$$

and the expected duration of a round t_R is equal to

$$E[t_R] = \frac{D}{H} ,$$

should have an expected reward per round of

$$E[R] = \frac{h \cdot E[t_R]}{D} \cdot E[R_P] = \frac{h}{H} \cdot E[R_P] , \tag{2}$$

where $E[R_P]$ is the expected reward of the pool.

3.1.2 Attack Model. The total number of miners in the pool is defined as N, consisting of the miner i and all other miners, denoted N_O. Each miner can earn a reward by either being at the top of the queue when a block is found, i.e. having the highest number of accumulated credits, or by receiving an uncle reward. Define p as the network probability of finding an uncle block in each round and $(1 - p)$ as the probability of finding a regular block. From the pool fairness (2), we know that the individual miner i will receive an expected reward from full blocks of

$$E[R_B] = \frac{h_i}{H} \cdot (1 - p) \cdot B . \tag{3}$$

Additionally, uncle blocks are distributed randomly between the miners, adding an expected reward from uncle blocks of

$$E[R_U] = \frac{1}{N_O + 1} \cdot p \cdot U . \tag{4}$$

We now consider a scenario where a miner in the pool, defined as the attacker has a hash rate of h_A. The attacker controls a number of miners, denoted N_A, where N_A refers to all miners controlled by the attacker, including uncle traps. The expected reward function from full blocks remains unaltered for the attacker. However, the expected uncle block reward changes due to the introduction of uncle traps. The expected reward from uncle blocks is now

$$E[R_U] = \frac{N_A}{N_O + N_A} \cdot p \cdot U , \tag{5}$$

per round. Numerical examples of Equation 5 can be found in Appendix A (Figure 7). Combining the two possible sources of reward from (3) and (5), we find the attacker's total expected reward as

$$E[R_A] = \frac{h_A}{H} \cdot (1 - p) \cdot B + \frac{N_A}{N_O + N_A} \cdot p \cdot U . \tag{6}$$

Hence, an attacker could substantially increase her expected reward, by dividing her mining power between a large number of smaller miners, thereby creating uncle traps. In Ethereum, additional addresses can be created at no financial cost, making this strategy feasible in practice. However, in order to be recognised by the pool operator, a minimum hashing power has to be maintained, imposing a lower limit on the divisibility of computational power.

In addition to splitting hashing power, an attacker can increase her reward by intentionally increasing the pool's uncle rate. This can be achieved by withholding a full block from the pool operator until some other miner has found a block and only then submitting it. The probability of event F that the attacker finds a block in any given round is

$$P(F) = \frac{h_A}{H} . \tag{7}$$

For a proportion of some long period of time, the attacker will be at the top of the queue. This proportion P_{TOP} is found to be

$$P_{TOP} = \frac{E[R_B]}{B} . \tag{8}$$

Conversely, the proportion of time the attacker is not at the top of the queue is $P_{\overline{TOP}} = (1 - P_{TOP})$. To pursue the outlined strategy, the attacker will withhold blocks, unless she is at the top of the queue. Hence, she will only force an uncle block, if she finds a block and is not top of the queue. The number of blocks I the attacker finds over time period T, consisting of T/t rounds of duration t, for which she is not top of the queue is equal to the expected maximum number of intentional uncle blocks. We find the expected value of I to be

$$E[I] = (1 - P_{TOP}) \cdot P(F) \cdot \frac{T}{t} .$$

From Equations (7) and (8) we find the expected number of intentional uncle blocks as

$$E[I] = \frac{h_A}{H} - \frac{{h_A}^2}{H^2} \cdot (1 - p) \cdot \frac{T}{t} .$$

Therefore an adversary has two levers to exploit the random uncle block reward distribution scheme. Firstly, she may split her hashing power between multiple miners to increase the likelihood of receiving an uncle block. Secondly, she can intentionally increase the pool uncle rate, increasing the total number of uncle blocks she can attempt to gather.

3.2 Attack Execution

We now proceed to reconstruct the observed attack step by step.

(1) An attacker generates N_A Ethereum addresses, where $N_A - 1$ will be the number of uncle traps
(2) The attacker iterates over the generated uncle trap addresses and computes s shares per address. We assume $s = 1$ for simplicity, although the actual number may be higher in reality.
(3) Each time a generated share is the valid solution to the network's PoW puzzle:
   - If one of the attacker's miners is at the top of the queue, publish block to pool instantly.
   - Else, withhold the block until another block is found in the network, then publish to the pool, generating an uncle.

Additionally, the attacker maintains one larger miner in the pool. For that miner, she follows the behaviour outlined in (3) above. Hence, she will force uncle blocks if none of her miners are at the top of the queue and immediately publish the block otherwise.

3.3 Real World Observations

Historic observations about the state of Ethpool suggest that there has been at least one occasion where a large miner did pursue the aforementioned reward-increasing behaviour over a longer period of time [1]. It was observed that a swarm of 7 500 miners with conspicuously small hash rates, reportedly ranging from 2 MH[4] per second to 6 MH/s, were mining in the pool, all of which were orchestrated by the same adversary[5]. We shall refer to this pseudonymous beneficiary account as the attacker. Even though it is not possible to retrieve historic hash rate information for Ethpool participants, it was possible to verify that the aforementioned adversary was a major beneficiary of the rewards collected by the observed family of smaller miners during the period of mid-May to mid-July 2018 by tracing payments on the Ethereum blockchain. We extracted the addresses of all recipients of uncle and full block rewards from Ethpool for the two month period and checked if any of these addresses made a transaction to the attacker. Over the course of this period, we found that the attacker managed to receive 19.14% of the total uncle blocks found by Ethpool spread across a set of 148 unique miners participating in the pool. Figure 1 displays the total number of uncle blocks found per day by Ethpool over the stated time period and compares it to the share of uncle blocks received by the attacker. The attacker's share of the total received uncle block rewards is substantially higher compared to her 5.55% share of the total full blocks mined by the pool throughout the same time period.

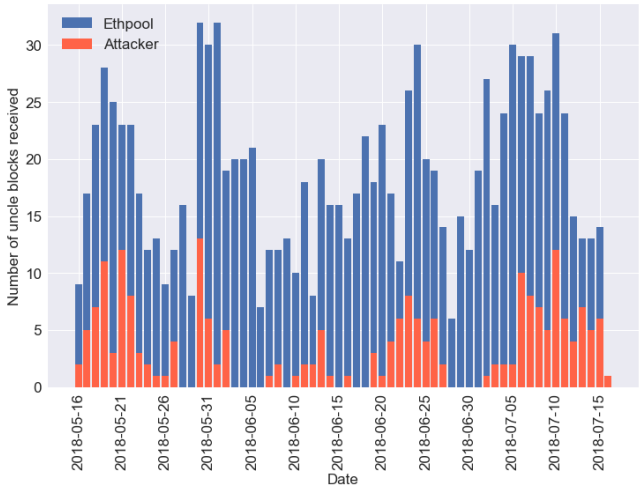

Figure 1: The uncle blocks the attacker received between mid May and mid July 2018 in Ethpool compared to all uncle blocks found by the pool during that period.

For the examined time period, the average total hash rate of the Ethereum network was 276.16 TH/s[6]. During that time, Ethpool found 1.54% of the total number of blocks found by the network and thus Ethpool's overall hash rate at that time was approximately 4.24 TH/s. Hence, from the attacker's share of the total number of full blocks mined by Ethpool, we know that the attacker also accounted for approximately 5.55% of the pool's total hash rate, or 185.74 GH/s[7]. A total of 7 500 miners with an average hash rate of approximately 4 MH/s per miner, operated by the attacker, would require 30 additional GH/s, invested in uncle traps. These derivations are in line with the observations concerning uncle blocks shown in Figure 1, which presumably reflect the performance of the uncle traps harvesting the uncle block rewards. An overview

[4] 1 Megahash = 10^6 hashes
[5] miner address: 0x7dE44b1F1527486a16FF586eF301B6b62dA6aC11
[6] 1 Terahash = 10^{12} hashes
[7] 1 Gigahash = 10^9 hashes

of the total reward distribution of the attacker may be found in Appendix A (Figure 5).

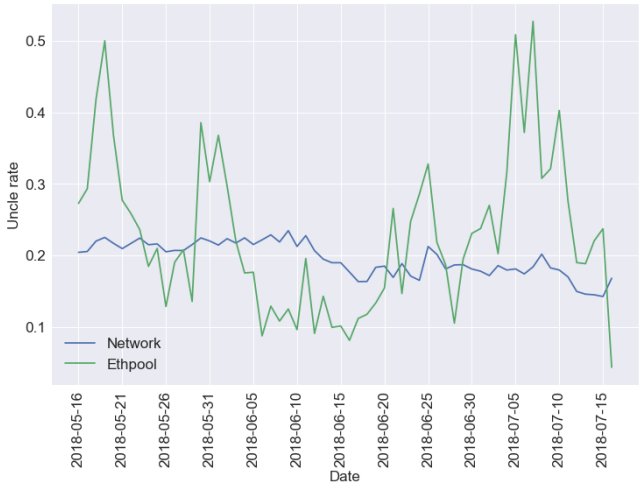

Figure 2: The uncle rate of the Ethereum network compared to the uncle rate of Ethpool between mid May and mid July 2018.

A further interesting observation lies in the uncle rate of Ethpool compared to the network's uncle rate for the examined time period, as shown in Figure 2. With an average uncle rate of 22.92%, the pool lies slightly above the network's rate of 19.52%. Ethpool's uncle rate standard deviation of 0.1090 notably exceeds the network's uncle rate standard deviation of 0.0245. This is explained by the deliberate increase of the uncle rate by the attacker, as described above.

4 PROPOSED MITIGATION

Apart from the general challenge of random number generation in a publicly verifiable manner, the main problem of the studied uncle reward policy lies in not accounting for hash rate differences between miners in a pool. Distributing uncle blocks following the same single queue-based scheme as for full blocks would lower the expected reward of mining in the pool below that of solo mining and does thus not serve as a mitigation. Our proposed solution to the problem entails a hash rate-weighted random allocation of uncle blocks to participating miners. Let the probability of receiving an uncle block reward be $\frac{h_i}{H}$ for each miner i, where h_i is the hash rate of that miner and H is the total hash rate of the pool. As in a PPLNS scheme, miner hash rates h_i are computed as $h_i = \frac{s_i}{S}$, where S is a large number – e.g. a multiple of the network difficulty – and s_i are the shares submitted by a miner i over the period of S shares. Further, we define h_A as the total hash rate of the attacker. The attacker's expected reward per round from receiving uncle blocks, when dividing her hashing power between N_A miners will then be

$$E[R_U] = \sum_{i=1}^{N_A} \frac{h_i}{H} \cdot p \cdot U \,. \tag{9}$$

From Equations (9) and (3), we find the attacker's total expected reward as

$$\begin{aligned} E[R_A] &= \frac{h_A}{H} \cdot (1-p) \cdot B + \frac{h_A}{H} \cdot p \cdot U \\ &= \frac{h_A}{H}((1-p) \cdot B + p \cdot U) \end{aligned}$$

By introducing a hash rate-weighted allocation of uncle block rewards, we eliminate the number of miners N_A – operated by the attacker – from the expected reward function. Hence, an attacker is no longer able to increase her profit function from dividing hashing power between many miners and fair uncle block distribution is assured. It should be noted that this mitigation does not preclude the deliberate increase of a pool's uncle rate.

5 DISCRETE-EVENT SIMULATION OF UNCLE TRAPS

In this section we use a discrete-event simulation to examine the effectiveness of the reconstructed attack strategy. We compare the performance of an attacker between different simulated mining pool scenarios.

5.1 Simulation Set Up

We simulate a queue-based mining pool for a duration of 200 000 blocks using a discrete-event simulator developed in C++. All simulations were executed on an Ubuntu server with an AMD EPYC 7401P processor. For the constant uncle rate, the average uncle rate of the network during the observed attack in Ethpool, namely 0.195235, is taken. Additionally, we assume static network and share difficulties of 200 trillion and 3.6 billion, respectively. The total computational power of Ethpool during the observed attack was equal to approx. 4.24 TH/s, yet the exact hash rate distribution remains unknown. However, Zamyatin et al. [18] show that the hash rates in Ethpool resemble a log-normal distribution. Hence, we construct a mining pool by sampling from a log normal distribution for the hash rates of the pool participants until the overall pool size is equal to approx. 4.24 TH/s, including the prespecified hash rate of the attacker. The distribution of sampled miners can be seen in Appendix A (Figure 6).

We examine three specific scenarios. The first scenario is a reference scenario referred to as *Honest*, which represents regular mining under a queue-based payout scheme, absent any attack strategy. In this reference scenario, the attacker employs her full computational power of 215.73 GH/s for a single account. For the second scenario, called *Attack*, we simulate the reconstructed attack. In this scenario, the attacker deploys 185.73 GH/s from one account and intentionally increases the uncle rate as outlined in Section 3. Additionally, the attacker spreads a total of 30 GH/s equally across 7 500 uncle traps she controls. Lastly, we simulate the same attack behaviour under the hash rate-weighted uncle block reward distribution policy we proposed in the previous section. We refer to the third simulated mining scenario as the *Mitigation* case.

5.2 Simulation Results

To evaluate the performance of the attacker in the three distinct mining pool scenarios, we focus on the number of full and uncle blocks mined and rewarded. Additionally we assess the success of

Table 1: Simulation results for a honest mining pool scenario and an attack scenario with uncle traps.

		Scenarios		
		Honest	Attack	Mitigation
Full Blocks	*Mined*	8 126	329	329
	Rewarded	8 065	6 651	6 651
	Ratio	0.9925	20.2158	20.2158
Uncle Blocks	*Mined*	1 968	9 785	9 785
	Rewarded	72	43 865	2 353
	Ratio	0.0366	4.4829	0.2405
Reward per invested MH (ETH$\times 10^{-8}$)		1.2075	4.6812	1.1921

the attacker by examining the reward per invested MH in ETH. We can compute the former as

$$\textit{Reward per invested MH} = \frac{(B \cdot b_r) + (U \cdot u_r)}{(TS \cdot d)/1\,000\,000}, \tag{10}$$

where for a given miner m_i, TS is the total number of shares submitted, b_r the total number of full blocks received by a miner, and u_r the total number of uncle blocks received by a miner.

5.2.1 Honest Scenario. For the *Honest* scenario, Table 1 shows that the attacker was rewarded for a substantially lower number of uncle blocks relative to the number of uncles she mined. Assuming the average uncle block reward was 1.6875 ETH[8], this would amount to a total loss of 3 199.5 ETH, or 684 213.01 USD[9] for the attacker compared to mining solo. Based on the uncle rate of 19.52%, the block duration of 200 000 blocks, as well as on the constructed pool consisting of 498 miners, the expected number of uncles rewarded per miner regardless of hash rate is 76.31. This shows how large miners with above average hash rates in the pool are inherently at a disadvantage, compared to mining solo, due to the uncle block reward distribution.

5.2.2 Attack Scenario. In the *Attack* scenario, the attacker mined a significantly lower number of full blocks, a logical consequence of the uncle rate spiking. The uncle traps have resulted in 43 865 uncle blocks being rewarded to the attacker, an increase by a factor of 609.24 relative to the number of received uncle blocks in the *Honest* scenario. The attacker was rewarded 34 080 uncle blocks more than she actually mined. When assessing the overall performance of the attacker, it can be seen that the attacker was able to increase her reward per invested MH by a factor of 3.88 from 1.2075×10^{-8} ETH to 4.6812×10^{-8} ETH overall.

[8]the average of the possible uncle block rewards
[9]with an ETH price of 213.85 USD. www.coinmarketcap.com. Accessed: 10-03-2018

5.2.3 Mitigation Scenario. For the *Mitigation* case, the attacker received only 5.36% of the uncle block rewards she received in the *Attack* scenario. This is in line with our formal evaluation of the reward scheme in Section 4. The number of uncles received is still substantially higher than for the *Honest* scenario. However, the attacker was rewarded 17.53% fewer full blocks, due to the continued use of the uncle rate spiking strategy. Overall, the reward per invested MH is the lowest for the Mitigation case, being slightly lower than for the *Honest* scenario.

5.3 Other Interesting Observations

Given that the attack strategy under the current uncle block reward distribution policy proved to be very profitable, we investigate its effect on other pool participants. The top plot of Figure 3 displays the reward distribution for full and uncle blocks among miners of different hash rates over the course of 200 000 blocks, absent any attack strategy. The bottom plot shows the block distribution for the uncle trap attack scenario. When comparing these two plots, one can see in the attack scenario (bottom) that the majority of uncle block rewards are being absorbed by miners of very small size (red line), the uncle traps. The high drop in uncle reward density for medium and large sized miners can be explained by the altered hash rate distribution for the pool under the presence of uncle traps (Appendix A Figure 6), which absorbed 43 860 uncle block rewards.

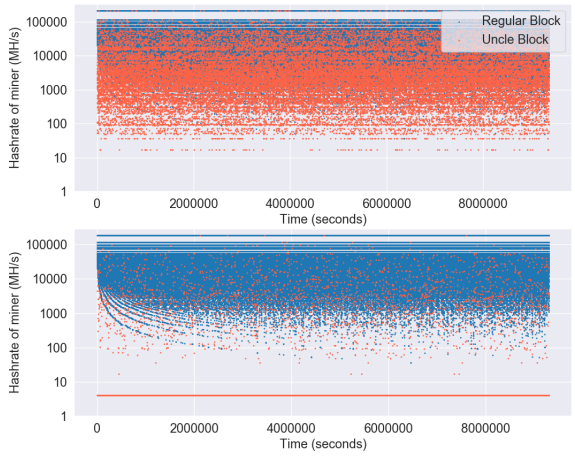

Figure 3: Small miners (red line in bottom plot) absorbing the uncle block rewards in the *Attack* scenario, compared to the *Honest* case (top).

The effect of uncle traps on the performance of all pool participants can also be seen by comparing the performance of miners based on their computational efforts, as shown in Figure 4. As the current uncle reward distribution mechanism does not account for differences in hash rates, miners of different hash rates are affected equally negatively. This is different from the previously studied attacks described in Section 2.3. Hence, an adversary orchestrating

an uncle trap attack can not only increase her own reward, but also harms the pool participants as a collective.

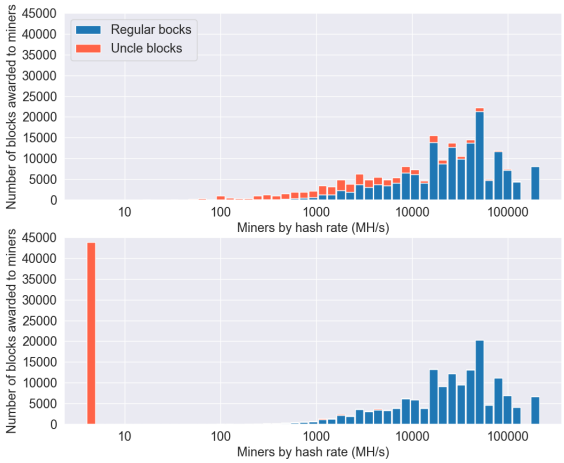

Figure 4: Number of full and uncle blocks awarded to miners based on hashrate (MH/s) on a logarithmic scale in a pool with no uncle traps (top) and with uncle traps (bottom) for 200 000 blocks.

6 CONCLUSION AND FUTURE WORK

Building on anomalies observed in Ethpool data, we have evaluated a Sybil attack strategy, devised to exploit a system of random uncle block reward distribution in a queue-base mining pool. By formally reconstructing the observed attack, we have identified two levers that an attacker can employ to increase her reward. In a discrete-event simulation, we have further demonstrated the effectiveness of the aforementioned attack strategy. We show that the significant negative effect of an uncle trap attack on mining pool participants can be mitigated by accounting for hash rate differences in the uncle block reward policy. Hence, we recommend an altered distribution mechanism, whereby the likelihood of a miner receiving the full uncle reward should be directly proportional to the hash rate of a miner. Such a scheme would work towards ensuring pool fairness and obviate the use of the outlined Sybil attack strategy.

A further investigation into the overall fairness of queue-based mining pools including a game-theoretic evaluation of mining pool equilibria under different attack scenarios could be a fruitful avenue for further research. A formal exploration of schemes to mitigate the deliberate increase in a mining pool's uncle rate could further help devising fair reward distribution policies. Formal verification of reward schemes would help to increase transparency and guarantee fairness with respect to reward allocation.

ACKNOWLEDGMENTS

The authors would like to thank Dragos Ilie for helpful feedback. This research is supported by funding from the Brevan Howard Centre for Financial Analysis, Blockchain.com, Bridge 1 858561 SESC, Bridge 1 864738 PR4DLT (all FFG), the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), and the competence center SBA-K1 funded by COMET.

REFERENCES

[1] BitcoinTalk. 2018. Don't be a fool - Stop mining on Ethpool right now. https://bitcointalk.org/index.php?topic=3970569.0. Accessed: 2018-09-17.
[2] Vitalik Buterin. 2014. Ethereum: A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper. https://github.com/ethereum/wiki/wiki/White-Paper Accessed: 2016-08-22.
[3] Ethereum community. 2018. Mining. https://github.com/ethereum/wiki/wiki/Mining. Accessed: 2018-09-13.
[4] Nicolas T Courtois and Lear Bahack. 2014. On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718. https://arxiv.org/pdf/1402.1718.pdf Accessed: 2018-09-22.
[5] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. 2016. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, Vol. 9604. Springer, Berlin, Heidelberg, 106–125.
[6] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the Bitcoin network. In *IEEE P2P 2013 Proceedings*. IEEE Computer Society, Trento, Italy, 1–10. https://doi.org/10.1109/P2P.2013.6688704
[7] A. Gervais, G. O. Karame, V. Capkun, and S. Capkun. 2014. Is Bitcoin a Decentralized Currency? *IEEE Security Privacy* 12, 3 (May 2014), 54–60. https://doi.org/10.1109/MSP.2014.49
[8] Jordan Holland, Joseph Connor, Parker Diamond, Jared M. Smith, and Max. Schuchard. 2018. Not So Predictable Mining Pools: Attacking Solo Mining Pools by Bagging Blocks and Conning Competitors. https://fc18.ifca.ai/preproceedings/79.pdf. Accessed: 2018-10-19.
[9] Aljosha Judmayer, Alexei Zamyatin, Nicholas Stifter, Artemios G Voyiatzis, and Edgar Weippl. 2017. Merged Mining: Curse or Cure? In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, Berlin, Heidelberg, 316–333.
[10] Aron Laszka, Benjamin Johnson, and Jens Grossklags. 2015. When Bitcoin Mining Pools Run Dry. In *Financial Cryptography and Data Security*, Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 63–77.
[11] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. 2015. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, ACM, Istanbul, Turkey, 919–927. http://www.cs.huji.ac.il/~avivz/pubs/15/fp245-lewenbergA.pdf
[12] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. 2015. Inclusive block chain protocols. In *Financial Cryptography and Data Security*. Springer, Puerto Rico, 528–547. http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive_btc_full.pdf
[13] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. 2015. On power splitting games in distributed computation: The case of bitcoin pooled mining. In *Proc. 28th IEEE Computer Security Foundations Symposium (CSF 2015)*. IEEE, IEEE Computer Society, Verona, Italy, 397–411.
[14] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. https://bitcoin.org/bitcoin.pdf Accessed: 2015-07-01.
[15] Meni Rosenfeld. 2011. Analysis of bitcoin pooled mining reward systems. arXiv preprint:1112.4980. https://arxiv.org/abs/1112.4980 Accessed:2018-10-16.
[16] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. 2016. Incentive Compatibility of Bitcoin Mining Pool Reward Functions. In *FC '16: Proceedings of the the 20th International Conference on Financial Cryptography*. Springer Berlin Heidelberg, Berlin, Heidelberg, 477–498. http://www.jbonneau.com/doc/SBBR16-FC-mining_pool_reward_incentive_compatibility.pdf
[17] Marie Vasek, Micah Thornton, and Tyler Moore. 2014. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *International Conference on Financial Cryptography and Data Security*. Springer, Springer Berlin Heidelberg, Berlin, Heidelberg, 57–71. http://secon.utulsa.edu/vasek/vasekbtc14.pdf
[18] Alexei Zamyatin, Katinka Wolter, Sam Werner, Peter G Harrison, Catherine EA Mulligan, and William J Knottenbelt. 2017. Swimming with Fishes and Sharks: Beneath the Surface of Queue-based Ethereum Mining Pools. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, IEEE Computing Society, Banff, Canada, 99–109. https://doi.org/10.1109/MASCOTS.2017.22

A SUPPLEMENTARY FIGURES

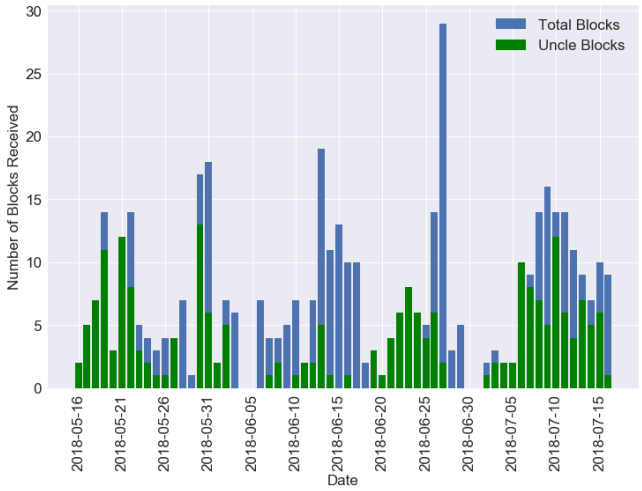

Figure 5: The number of uncle blocks the attacker received per day relative to the total number of blocks she received per day in Ethpool between mid May to mid July 2018.

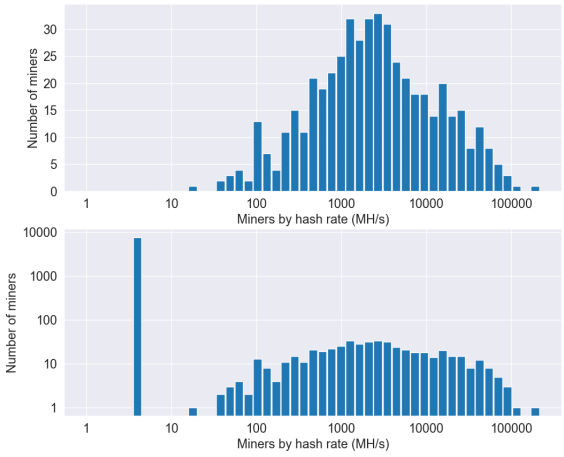

Figure 6: The miner distribution in a mining pool with no uncle traps (top) opposed to with uncle traps (bottom).

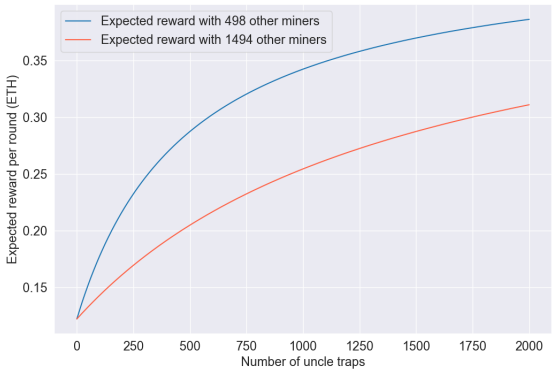

Figure 7: The attacker's expected reward per round as a function of the number of uncle traps employed.

B VARIABLES AND SYMBOLS

Symbol	Description
D	Network difficulty
d	Share difficulty
B	Full block reward
U	Uncle block reward
H	Aggregate hash rate of the mining pool
h_A	Hash rate of the attacker
S	A large number, e.g. a multiple of the network difficulty
s_i	Shares submitted by a miner i over a period of S shares
p	Probability of the pool finding an uncle block in a given round
$(1-p)$	Probability of the pool finding a full block in a given round
P_{TOP}	Proportion of time the attacker is at the top of the queue
$P_{\overline{TOP}}$	Proportion of time the attacker is not at the top of the queue
$E[R_P]$	Expected reward for the total pool per round
$E[R_A]$	Expected reward for the attacker per round
$E[t_R]$	Expected duration of a round
$E[R_U]$	Expected reward from uncle blocks per round
U_n	Uncle block number
B_n	Block number
N	Total number of miners in the pool
N_O	Number of miners in the pool not controlled by the attacker
N_A	Number of miners in the pool controlled by the attacker
I	Number of intentional uncle blocks forced by the attacker