Swimming with Fishes and Sharks: Beneath the Surface of Queue-based Ethereum Mining Pools

A. Zamyatin*,‡, K. Wolter†, S. Werner‡, C.E.A. Mulligan‡, P.G. Harrison ‡ and W.J. Knottenbelt‡
* SBA Research, Austria
† Freie Universität Berlin, Germany
‡ Imperial College London, United Kingdom
Email: azamyatin@sba-research.org, katinka.wolter@fu-berlin.de,
{sam.werner16, c.mulligan, pgh, wjk}@imperial.ac.uk

***Abstract*—Cryptocurrency mining can be said to be the modern alchemy, involving as it does the transmutation of electricity into digital gold. The goal of mining is to guess the solution to a cryptographic puzzle, the difficulty of which is determined by the network, and thence to win the block reward and transaction fees. Because the return on solo mining has a very high variance, miners band together to create so-called mining pools. These aggregate the power of several individual miners, and, by distributing the accumulated rewards according to some scheme, ensure a more predictable return for participants.**

In this paper we formulate a model of the dynamics of a queue-based reward distribution scheme in a popular Ethereum mining pool and develop a corresponding simulation. We show that the underlying mechanism disadvantages miners with above-average hash rates. We then consider two-miner scenarios and show how large miners may perform attacks to increase their profits at the expense of other participants of the mining pool. The outcomes of our analysis show the queue-based reward scheme is vulnerable to manipulation in its current implementation.

I. Introduction

The field of cryptocurrencies has experienced a rapid growth in popularity since the introduction of Bitcoin [19] in 2008. Today, over 750 alternative cryptocurrencies or *altcoins*[1] exist. Ethereum [6] is the most highly capitalised cryptocurrency after Bitcoin. Its primary innovation is a Turing-complete scripting language allowing the creation of programs governing the transfer of value, known as *smart contracts*.

A fundamental data structure underpinning many cryptocurrencies is the *blockchain*. This provides an append-only immutable record of digitally-signed *transactions*. Transactions, each of which represents the transfer of some token of value from source wallets to recipient wallets, are consolidated into *blocks*. Each block is identified by a unique hash over all included transactions and the block header, which contains (amongst other things) the hash of the previous block and a nonce. The fact that the hash of the previous block is referenced in the next block effectively *chains* the blocks together, such that it is impossible to change the contents of a block without also updating every subsequent block.

Participating nodes in a cryptocurrency network communicate in a peer-to-peer fashion using a gossip protocol, broadcasting blocks so that each node stores a complete copy of the blockchain. Since there is no central point of control, a key element of this system is the distributed consensus mechanism used to agree on the content accepted into the blockchain.

In Bitcoin and Ethereum, and most altcoins, the mechanism used is referred to as *Nakamoto consensus* and involves nodes competing to solve a challenging cryptographic puzzle, known as Proof-of-Work (PoW)[2]. The latter is designed such that there exists no better strategy than enumerating all possible candidates, while the verification of a potential solution is trivial. The process of attempting to solve this puzzle is defined as *mining* and the participating nodes are referred to as *miners*. Each attempt at a solution is known as a *hash* and the computational power of a miner is given by its *hash rate*.

Miners collect all transactions they receive over the peer-to-peer network and consequently try to generate a new block by brute-forcing the solution to the required PoW puzzle. Each time a miner succeeds in creating a new block, the latter is appended to the public blockchain and propagated through the network. As reward for investing computational effort, the miner is granted a fixed amount of newly generated or minted units of the underlying currency. Furthermore, transactions include a small fee to incentivise the winning miner to include them in the latest block.

In Ethereum, the PoW consists of finding a nonce input to the Ethash [8] algorithm, such that the result is below a certain threshold depending on the *difficulty* [10]. Since miners can leave or join the race for generating the next block at any time, Ethereum implements a mechanism to dynamically adjust the difficulty of the PoW, such that a new block is found on average approximately every fifteen seconds. At the time of writing, the difficulty, i.e. the expected number of hashing operations required to find a solution to the PoW, amounts to approximately 740 trillion hashes [1].

To reduce the variance of the time between finding blocks and hence stabilise revenue over time, miners cooperate to create so-called *mining pools*. The hash rate of such mining pools usually significantly exceeds that of single miners and, as a result, the average interval between finding blocks is reduced.

[1] Source: http://coinmarketcap.com. Accessed: 2017-04-11

[2] While other consensus mechanisms, such as Proof-of-Stake [14], [11], are currently being researched and developed, PoW, as of this writing, remains by far the most adopted consensus approach in permissionless blockchains.

Taken together with a scheme which ideally distributes block rewards proportionally to the effort invested by miners, this allows each participant to more accurately predict the overall accumulated revenue over time and ensures a steadier payment stream. In return, miners are usually charged a small proportion of their revenue by the pool. To measure the effort invested by miners, the mining pool accepts solutions to a cryptographic puzzle that has a considerably relaxed difficulty threshold; these solutions are known as *shares*.

The schemes used for dividing rewards among miners can differ substantially from pool to pool. Some pools split each mined block into fractions and award each miner the part of the block that corresponds to their mining investment in terms of shares, while other pools rank miners according to the invested work as evidenced by shares and award a mined block always to the top-ranked miner. Previous research work by Rosenfeld provided an overview of such reward schemes [20] and introduced so-called *pool-hopping* attacks, where miners dynamically switch between pools to increase their profit. Lewenberg et al. pointed towards problems in preventing pool-hopping [16], while Schrijvers et al. conducted a study on incentive compatibility of common reward schemes [21]. Further research evaluated potential attack scenarios between pools, including denial-of-service attacks [13], [15] and less direct *withholding attacks* [12], [7], [17], where pools infiltrate competitors and cause damage by withholding valid blocks.

In this paper we focus on a recently-introduced approach for distributing block rewards among miners [4] which we refer to as a *queue-based* reward payout scheme. Under this scheme, the block reward is paid to the miner residing at the first position of a priority queue sorted by credits received for submitted shares over time. We evaluate the expectation and variance of miners' revenues under this scheme, comparing the results to the *PPLNS (Pay-Per-Last-N-Shares)* reward payout scheme. Thereby we aim to show which type of miners, in terms of different hash rates, benefit the most from the queue-based reward payout scheme as opposed to an alternative 'fair' reward scheme. To this end, we introduce a discrete event-based simulation, which allows us to model the ecosystem of a single mining pool including the dynamics of miner behaviour. We make use of data extracted from Ethpool [2], a popular Ethereum mining pool, which was the first to implement the queue-based payout scheme. For comparative purposes we use a conventional PPLNS scheme, as implemented by the Ethermine Ethereum mining pool. Furthermore, we highlight a potential vulnerability rooted in the uneven distribution of credits in the queue-based approach, which can be exploited in several ways by miners with above average hash rates to increase their long-term revenue. A real-world scenario, in which this vulnerability is being exploited to the benefit of a small group of miners, has been observed in Ethpool.

The remainder of this paper is organised as follows. Section II outlines useful background and notation. Section III explains the mechanics of the queue-based reward payout scheme, while Section IV discusses the numerical results obtained from a simulation of this model, highlighting how large miners are at a natural disadvantage in such a scheme and comparing the economic benefit of different simulated attack scenarios. Section V introduces three different attack scenarios arising from the nature of the queue-based reward scheme. Section VI concludes and outlines future work.

II. Preliminaries

In this section we will provide an overview of different mining approaches, more specifically solo and pooled mining, as well as explain the structure of miner rewards and the most common conventional reward payout schemes.

A. Notation

In the process of mining for a cryptocurrency a miner m_i will perform the necessary hashing operations at rate h_i. Commonly, the hash rate of a miner will range between a few hundred megahashes[3] per second (MH/s), and several gigahashes per second (GH/s). The total hash rate of a group of miners is the sum of the individual hash rates, denoted by H. The difficulty D indicates the expected number of hashes needed to find the next block. The pool difficulty d indicates the expected number of hashes needed to find a share that is submitted to the pool; such shares enable the mining pool to objectively assess miner hash rate and may also be candidates for a new block.

B. Miner rewards

The Ethereum mining protocol differentiates between full and so-called *uncle* blocks. The latter represent valid blocks, which did not become the new head of the blockchain [9]. This occurs if a block submitted by a competing miner is propagated faster to the majority of all nodes in the network. We denote the probability of a block being an uncle, which depends on the miner's network connectivity γ, as p_u.

In contrast to Bitcoin, miners receive payouts not only for full, but also for uncle blocks in Ethereum. As of this writing, the reward for a full block R_b is 5 ETH[4], while the reward R_u for an uncle block is 3.75 ETH, excluding transaction fees. On the Ethereum public blockchain, as in most other cryptocurrencies, the revenue generated by finding a block consists of the block reward and fees collected from the included transactions. Since the transaction fees are hard to model, while representing only a small share of the total block reward, they are omitted in this paper for simplification. Hence, we denote the expected revenue per block as:

$$R_e = R_b(1 - p_u) + R_u p_u \tag{1}$$

C. Solo mining

Miners participating in the consensus finding mechanism on an individual basis, and thereby receiving the entire reward for each found block, are referred to as *solo miners*. The number of blocks found by a solo miner per time unit follows a Poisson distribution with rate parameter $\lambda = h/D$, where D is the current difficulty and h represents the solo miner's hash rate.

[3] i.e. 10^8 hashes

[4] Ether – abbreviated ETH – is the underlying currency in Ethereum.

The expected revenue per performed hashing operation can be hence formulated as

$$E[R_h] = \frac{R_e}{D} \tag{2}$$

The variance of the revenue per hashing operation is

$$\mathrm{Var}[R_h] = R_e^2 \lambda = \frac{R_e^2}{D} \tag{3}$$

D. Mining pools

Mining pools are a way for solo miners to join their resources (mining power) together in order to increase their probability of finding a block. These pools are run by so-called *operators*, whose main task, apart from maintaining the mining software and scripts, is to estimate each participating miner's hash rate and their contribution to the generated blocks. To this end, the mining pool operator sends to each miner a PoW problem, identical to the network PoW puzzle, but with a lower difficulty[5].

Miners participate in the pool by continuously employing computational power in solving the pool's problem. Each time a miner finds a solution, i.e. finds a nonce input to the Ethash algorithm yielding a result below the required threshold, she submits a *share* to the block to be found next by the mining pool. Sometimes, the submitted solution to the mining pool's problem will also be a solution to the more difficult network problem. As a result, the mining pool will generate the next block and collect the block reward. The latter is then distributed among the pool's miners based on each miner's contribution and according to the reward payout scheme.

The time between submitted shares is exponentially distributed with mean d/h_i, where h_i is the hash rate of miner m_i and d is the pool problem difficulty. Each share is the solution to the current network PoW puzzle with probability d/D. The time it takes the mining pool as a whole to find a block can be modelled as an exponentially distributed random variable with mean D/H, where $H = \sum_{i=1}^{n} h_i$ is the sum of the hash rates of the n individual miners in the pool and D is the network difficulty. The number of blocks found in a specific time period in turn follows a Poisson distribution with rate parameter $\lambda = H/D$.

Shares can be *stale*, i.e. valid but no longer applicable to the current PoW puzzle of the network. In other words, if a miner submits a share for block b_j only after it has been found and the pool is already mining on block b_{j+1}, the share is ignored. The rate of stale shares depends on each miner's network connectivity γ.

E. Conventional reward payout schemes

The first mining pools were created around 2011, mostly implementing reward schemes that equally distribute each block reward among all or a subset of miners, based on shares submitted during a specific time period. To compensate their administrative effort, mining pools charge a fee f, which is a small proportion of the revenue. Below, we briefly summarize some well known reward payout schemes, as initially described by Rosenfeld [20].

1) Proportional Payout: In a proportional scheme, also referred to as *Round-based Pay-Per-Share*, miners receive payouts each time the mining pool finds a block, according to their contribution to this block, as measured by the number of valid shares s_i submitted by miner m_i since the last block. Hence, the expected reward per block of a miner m_i is

$$E[R_i] = (1-f) R_e \frac{s_i}{\sum_{j=1}^{n} s_j} \tag{4}$$

where n is the size of the mining pool. The counted shares of each miner are then reset to zero, as the mining pool starts with computations for the next block.

The number of expected shares per block is

$$E[S] = \frac{D}{d} \tag{5}$$

The expected revenue per hashing operation is the same as in solo mining, decreased by the mining pool's fee f:

$$E[R_h] = \frac{(1-f) R_e}{D} \tag{6}$$

The variance, however, is lower than in solo mining by approximately a factor $D/(\ln D)$. Thereby, only small miners effectively profit from the reduced variance: for large miners, accounting for significant portions of the mining pool's hash rate, the variance can only be multiplied by factor h_i/H, representing the miner's portion of the pool's hash rate [20].

2) Pay-Per-Last-N-Shares (PPLNS): The PPLNS payout scheme is a modified version of the proportional scheme, aiming at the prevention of pool hopping. This is achieved by performing the calculation of the reward payout only after the miners have submitted $N > E[S]$ shares in total. Hence, the expected revenue of miner m_i per payout is

$$E[R_i] = (1-f) B R_e \frac{s_i}{N} \tag{7}$$

where B is the number of blocks found by the pool during the last N shares and s_i the number of shares miner m_i contributed to N.

Due to the proportionality of the reward payouts, miners are incentivised to employ their maximal mining capacity for as long as possible in both of the above schemes.

III. Modelling the Queue-based Payout Scheme

This section provides a detailed overview of the queue-based reward payout scheme. We showcase the structure of the scheme, as implemented by Ethpool, as well as point out the workings and problems of the underlying mechanisms for credits accounting.

[5] If the problem difficulty were equal to the network difficulty, each submitted share would also be generating the next block. Hence, the mining pool operator would know how much work the finder of the block has performed, but would have no information on other miners' contributions.

A. Structure of reward payouts

The queue-based reward payout scheme was first implemented by Ethpool in late 2015 and introduces a new way of handling payouts, while relying on a mechanism to account for each miner's contribution similar to that of conventional payout schemes. By submitting valid shares, miners earn so-called *credits*. Each valid share increments the miner's credits by d, the expected number of hashes required to solve the mining pool's PoW problem.

The mining pool maintains a ranking in form of a priority queue, where the priority of each miner is defined by her earned credits. A miner's priority increases with each of her submitted shares. Based on their hash rate and network connectivity, miners race for the top position in the queue. As a result, the ordering of the priority changes dynamically after each submitted share.

Each time the mining pool finds a block, the *complete*[6] reward is allocated to the miner $m_{(1)}$ currently positioned at the top of the queue. Consequently, the winning miner's credits are reset to the difference between her and the second placed miner's credits:

$$c(m_{(1)}) := c(m_{(1)}) - c(m_{(2)}) \tag{8}$$

Uncle blocks are considered differently, in the sense that they do not lead to a re-calculation of the winner's credit balance. Hence, the winner of an uncle block will receive uncle reward R_u and continue to reside on top of the queue, until being overtaken or winning a full block.

While in theory, the possible range of miners' credits is $[0, +\infty)$, Ethpool states each miner is expected to collect approximately D credits, before winning a block [4]. However, due to the special treatment of uncles, the expected amount of credits of the miner at the top of the priority queue is:

$$E[c(m_{(1)})] = (1 + p_u)D \tag{9}$$

B. Discussion of potential problems

We now move on to highlight two potential problems of the queue-based payout scheme.

1) Unequal variance impacts: As we can see, this scheme only takes into account the credits and hence the work performed by the second placed miner, instead of looking at the amount of shares submitted/credits earned by all miners, for the credit resetting policy. This particularly impacts the revenue of large miners, which account for significant portions of the mining pool's hash rate. Since these miners produce significantly more shares than the average miner, they reach the top of the queue more frequently. Thereby, large miners also end up absorbing more of the mining pool's variance caused by lucky/unlucky streaks[7]. Small miners, on the other hand, reach the top of the queue less frequently. Thus, the probability of a small miner reaching the top at a time when the pool is having an unlucky streak is comparatively low to that of a large miner. As a result, small miners will be earning over-proportional revenue shares with regards to their invested computational effort.

TABLE I: Priority queue showing a gap between large and medium/small miners.

(a) Before Block i

Position	Miner	Credits
1	Alice	110
2	Bob	105
3	Eve	60
4	Dave	30

(b) After Block i

Position	Miner	Credits
1	Bob	105
2	Eve	60
3	Dave	30
4	Alice	5

(c) Before Block i+1

Position	Miner	Credits
1	Bob	115
2	Eve	65
3	Dave	35
4	Alice	15

(d) After Block i+1

Position	Miner	Credits
1	Eve	65
2	Bob	50
3	Dave	35
4	Alice	15

2) Non-uniform credits redistribution: In a scenario where two or more miners maintain significantly high hash rates compared to the rest of the miners in the pool, the large miners will be rapidly moving up the queue and overtaking slower miners. Consequently, there is a high probability of at least two large miners being positioned at the top of the queue with far more credits than smaller miners, when the pool finds the next block. Such a scenario is illustrated in Table I. Here the two large miners, Alice and Bob, consistently earn 10 credits per round, while the small miners, Eve and Dave, earn 5.

The calculation of the new credits for the winner of the block, in our case Alice, takes into account only the credits earned by Bob, placed second (cf. Table Ia). Since Bob too collected a high amount of credits, Alice will find herself re-positioned at the end of the queue (cf. Table Ib). In the next round, Bob will win the block reward. However, due to the significant gap between Bob's and Eve's credits, Bob will be re-positioned both in front of Dave and Alice, thus receiving an advantage (cf. Table Id).

We see the redistribution of credits is only fair if the credits difference between each two consecutive miners is constant. However, since real world observations show the logarithm of mining power in Ethpool resembles a Gaussian distribution (cf. Section IV-B), we argue that this is very unlikely to be the case in reality.

IV. Simulating the Queue-based Payout scheme

In this section we present simulation results for our model of the queue-based reward payout scheme and show how large miners are disadvantaged in Ethpool's current implementation. In Subsection IV-A, we provide simulation results for a pool containing only two miners, one with large hash rate and one with small hash rate, while simulation results for a realistic population of miners (as sampled from Ethpool) are given in Subsection IV-B.

[6] Less pool fees.

[7] Given some time interval, pool luck is the ratio of blocks actually mined by a pool to the mathematical expectation of the number of mined blocks.

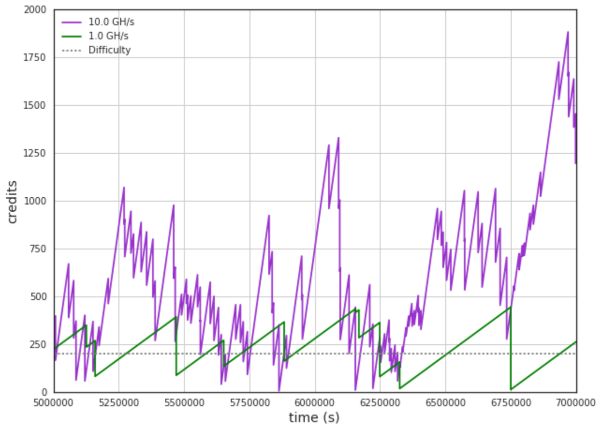

Fig. 1: The evolution of Ethpool credits in a two-miner scenario.

Our event-based simulator constructs the time between shares submitted by miners as a random number[8] following an exponential distribution with rate parameter $\lambda = h/d$. All simulations run for 500 000 blocks and are performed under constant network difficulty $D = 200$TH (trillion hashes), network connectivity $\gamma = 1$ (no uncle blocks), pool problem difficulty $d = 3.6$b and proportional pool fee $f = 0.01$. We further assume an uptime of 100% for all miners (with no stale or invalid shares).

A. Simulating a two-miner pool

First, we simulate the simple model of a mining pool containing only two miners, one large miner m_l with a hash rate of 10 GH/s and a miner m_s with a significantly lower hash rate of 1 GH/s. Although arguably such a scenario will not be observed in reality, we use this set-up to better understand the benefits and disadvantages of large and small miners in the queue-based reward payout scheme.

The development of earned credits is visualised in Figure 1. We can see the credits of the small miner $c(m_s)$ lie only slightly above the network difficulty when winning a block. The credits of the 10 GH/s miner $c(m_l)$, on the other hand, by far exceed the credits expected according to the network difficulty and are subject to high variance.

Closer observations of the credits development in Figure 1 show a repeating pattern. The large miner remains on top of the queue with significantly high credits for prolonged periods. However, this trend changes once the small miner has accumulated more credits than the large miner earns between winning two blocks. We can see that the credits of the large miner start to decrease quickly, while those of the small miner continue to grow. At some point, the small miner takes over the lead and wins a block. Consequently, $c(m_s)$ is reset to $c(m_s) - c(m_l)$ and the small miner is overtaken by the large, whose credits then start increasing significantly.

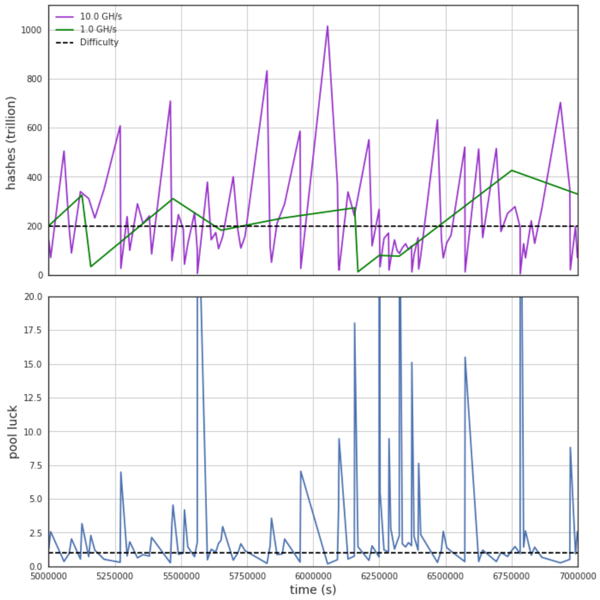

Fig. 2: Performed work per block (top) in comparison to pool luck in the two-miner scenario (bottom). Each peak in pool luck correlates with a drop in required work per block and vice versa.

As mentioned in Section II, it has been found in earlier work that miners responsible for significant portions of the total hash rate H of a mining pool can adjust their revenue variance by a maximum factor h_i/H. Hence, the revenue variance of m_l can be improved only by 9%, while the small miner profits from a variance reduction of over 90%. This is also evident from the top graph in Figure 2, which shows the development of performed work per block and its correlation with pool luck. As we can see, the variance of performed work is significantly higher for the large miner: 35 872.5 compared to 21 137.0 for the small miner. Furthermore, the large miner on average has to invest more computational effort per block than the small miner: 201.52TH in contrast to 186.82TH, which is surprisingly less than the network difficulty.

These observations are also mirrored in the miner's generated revenues: the small miner received rewards for 3 442 blocks more than she mined, as shown in Table II.

TABLE II: Blocks mined and rewarded in the queue-based scheme in the two-miner scenario.

Miner	Blocks			Average performed work (trillion hashes)
	Rewarded	Mined	Ratio	
Large (10 GH/s)	451,316	454,758	0.9924	201.52
Small (1 GH/s)	48,684	45,242	1.0761	186.82

[8] The generation of random numbers is accomplished by the Mersenne Twister [18] algorithm.

B. Simulating Ethpool

Next, we use a large data set of miners as input for our simulation to generate realistic results. In particular, the data set consists of 729 miners extracted from Ethpool's public API [3] in the period between 2017-02-21 and 2017-04-09, accumulating a total hash rate of 699.18 GH/s. Figure 3 shows the distribution of hash rates in the extracted data set on a logarithmic scale, which resembles a Gaussian curve.

Figure 4 shows the number of credits necessary to win a block. While it is not clear from visual inspection whether the credits form a stationary random process with normal variation, Figure 5 very clearly illustrates the high autocorrelation in the number of credits necessary to win a block.

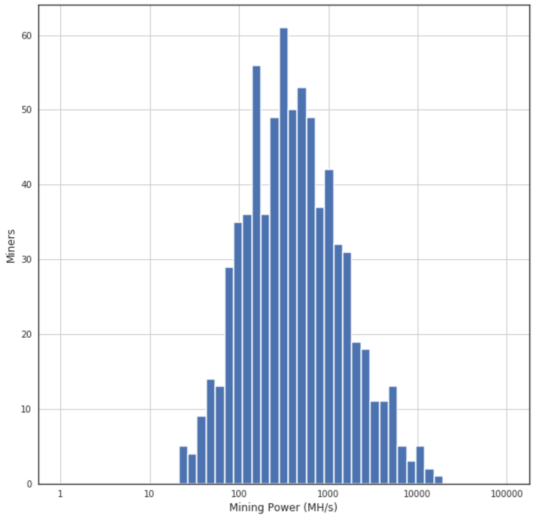

Fig. 3: Distribution of mining power in Ethpool (logarithmic scale). The largest miner controls 18.07 GH/s, the smallest 22 MH/s. The average mining power is approximately 960 MH/s, while the median amounts to 380 MH/s. Standard deviation is 1.74 GH/s.

Figure 6 visualizes the development of performed work for small, medium and large miners in Ethpool. As already seen in the two-miner scenario, the variance of the necessary work per block decreases with the respective miner's hash rate, i.e., miners accounting for significant portions of the overall hash rate are most affected by lucky/unlucky streaks.

Since the list of credits for Ethpool is public the interested miner is advised to study pool luck and the current level of credits when deciding which pool to join. Given the strong autocorrelation, the observed credit levels when the pool wins a block may also serve as a decision criterion whether or not to leave a pool.

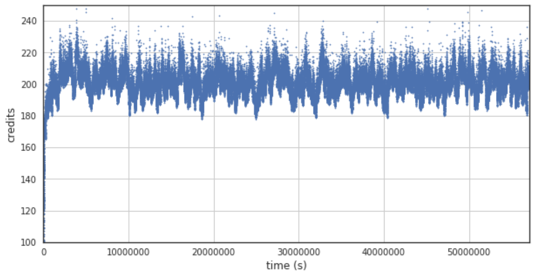

Fig. 4: Development of credits when winning a block in the multi-miner scenario.

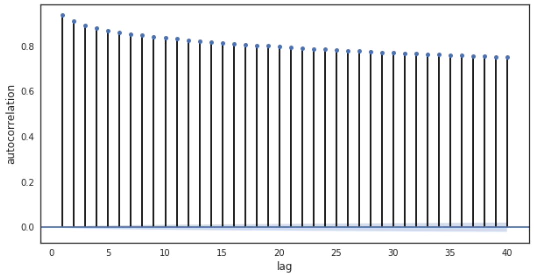

Fig. 5: Autocorrelation of credits when winning a block for lags 1:40.

Furthermore, the large miners must invest more computational power on average to win a block than small miners, as is evident in Figure 7. While miners with hash rates of more than 10 GH/s perform slightly more work than required by the network difficulty, miners in the 10th percentile of mining power evade approximately 5–7 trillion hashes (2.5–3.5% of total) of work per block. When put in relation to the average computational effort, the relative difference between of the smallest and largest miner amounts to nearly 5%. The results yielded by the simulation of Ethpool confirm the observations made in the two-miner scenario.

As in the two-miner scenario, the disadvantage of large miners is reflected in their economic performance, since small miners are rewarded for more blocks than they are entitled to. To better express the deviation of economic output between small and large miners, we compare the performance of miners in the queue-based scheme implemented by Ethpool, to the PPLNS scheme implemented by Ethermine, where N is equal to the shares submitted in the last 60 minutes [5]. We introduce *return per computed MH* as a performance metric and illustrate our results in Figure 8. It can be seen that there is a clear bias towards small and medium-sized miners with regards to profitability in Ethpool, while large miners have higher return on investment in Ethermine.

Figure 9 sorts the return on invested work by the hash rate and clearly shows that in a queue-based scheme like Ethpool miners with low hash rate receive above average return on investment, while miners with large hash rate are at a disadvantage.

We note that while the numerical difference between Ethpool and Ethermine is very small in this performance metric, the absolute bias scales up quickly over time. For example, a miner with a hash rate of 18.07 GH/s loses 1.406×10^{-10} ETH every million hashes, when choosing Ethpool over Ethermine.

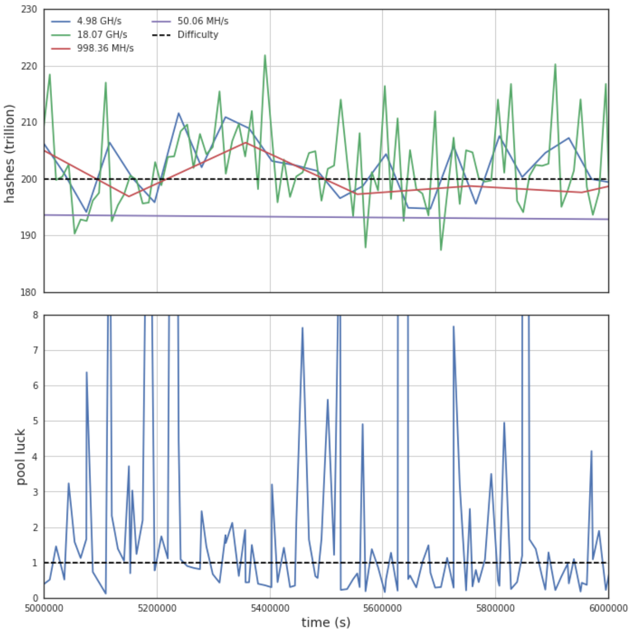

Fig. 6: Performed work per block in comparison to pool luck for small, medium and large miners. The large miner absorbs most of the variance.

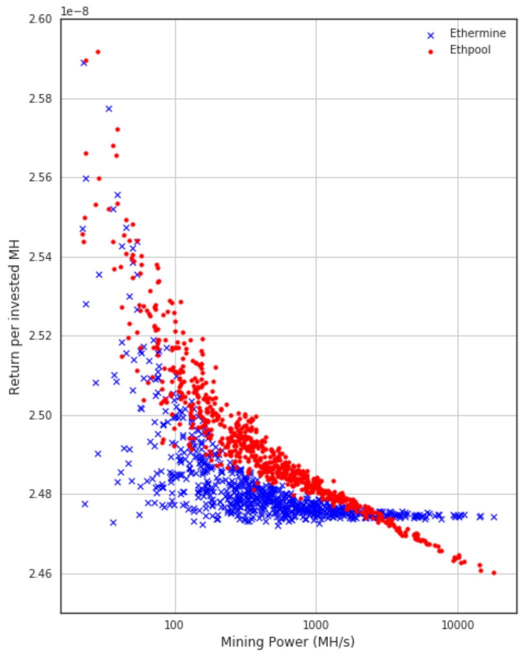

Fig. 8: Return per computed MH in a multi-miner scenario (logarithmic x-axis).

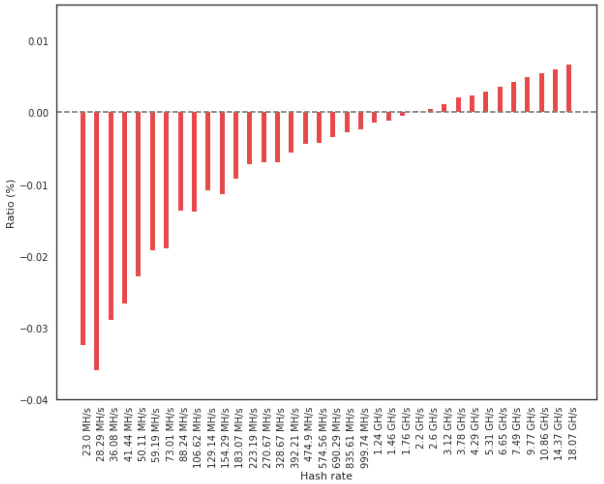

Fig. 7: Ratio of performed work per block by miners in Ethpool, relative to the work performed on average. Miners are grouped according to their hash rate.

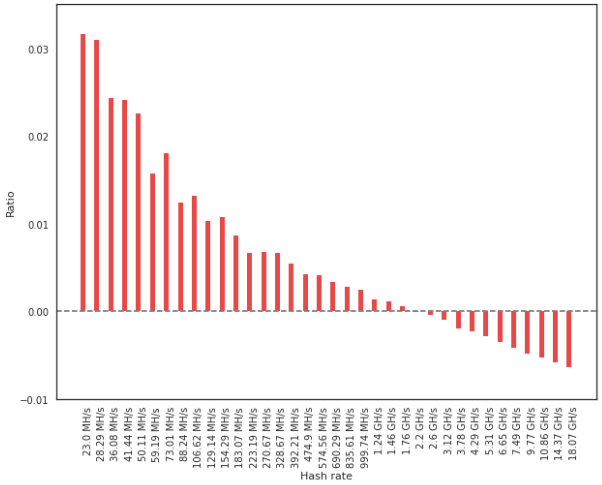

Fig. 9: Ratio of return per computed MH for miners in Ethpool, relative to the average return per computed MH. Miners are grouped according to their hash rate.

Consequently, her loss per day will amount to 0.2187 ETH[9] and approximately 79 ETH per year. We observe that small fish have a happier life in Ethpool than in Ethermine and are better off than the large sharks, at least if the latter do not attack in some way.

A summary of the simulation results with regards to economic performance of miners in Ethpool is provided in Table III. We find that miners with low hash rate benefit considerably from joining a queue-based mining pool. They can reduce the variance in their gained revenue and even receive better return on investment than the average miner. This must come at the expense of miners with large hash rate, who are at disadvantage with respect to both criteria.

C. Exponential Difficulty

The presented simulations were conducted under the assumption of a constant difficulty of Ethereum's PoW. However, in practice the difficulty is adjusted after every block and has

[9] At the time of writing this amounts to approximately US$ 78.

TABLE III: Miner performance in the multi-miner scenario.

Hash rate	Miners	Blocks			Average performed work (trillion hashes)	Average revenue per computed MH (10^{-8} ETH)		
		Rewarded	Mined	Ratio		Ethpool	Ethermine	Ratio
23.0 MH/s	5	16	12	1.3387	193.42	2.5540	2.5228	1.0124
29.84 MH/s	4	21	18	1.1944	193.08	2.5475	2.5112	1.0145
42.27 MH/s	15	30	31	0.9640	194.74	2.5338	2.5032	1.0122
57.75 MH/s	21	41	41	0.9965	195.77	2.5217	2.5022	1.0078
78.97 MH/s	33	56	54	1.0309	196.64	2.5140	2.4902	1.0096
107.79 MH/s	48	77	76	1.0148	197.19	2.5073	2.4873	1.0080
152.11 MH/s	75	108	106	1.0211	197.75	2.5011	2.4854	1.0063
203.06 MH/s	44	144	142	1.0169	198.25	2.4952	2.4803	1.0060
284.16 MH/s	78	202	200	1.0096	198.48	2.4927	2.4789	1.0056
384.63 MH/s	74	273	266	1.0258	198.76	2.4896	2.4782	1.0046
538.74 MH/s	73	383	379	1.0095	199.06	2.4861	2.4771	1.0036
729.49 MH/s	54	518	509	1.0186	199.27	2.4837	2.4760	1.0031
988.34 MH/s	55	703	694	1.0127	199.43	2.4818	2.4758	1.0024
1.39 GH/s	48	987	977	1.0107	199.67	2.4788	2.4754	1.0014
1.84 GH/s	24	1 305	1 294	1.0088	199.83	2.4769	2.4751	1.0007
2.54 GH/s	27	1 802	1 779	1.0125	200.01	2.4748	2.4749	1.0000
3.59 GH/s	18	2546	2519	1.0107	200.27	2.4716	2.4747	0.9987
4.99 GH/s	15	3 531	3 483	1.0140	200.46	2.4692	2.4745	0.9979
6.72 GH/s	8	4 744	4 739	1.0010	200.66	2.4668	2.4745	0.9969
9.91 GH/s	6	7 046	7 044	1.0002	200.92	2.4637	2.4745	0.9956
13.47 GH/s	3	9 457	9 436	1.0023	201.07	2.4618	2.4744	0.9949
18.07 GH/s	1	12 844	12 864	0.9984	201.22	2.4602	2.4742	0.9943

been observed to increase at a high rate. In fact, between March and June 2017 the difficulty has increased from 200 to 740 trillion hashes, resembling exponential growth at approximate rate $k = 2.726 \times 10^{-6}$.

We simulate both the two-miner and multi-miner scenarios under exponentially increasing PoW difficulty, using an initial difficulty $d = 200$ trillion hashes and the measured growth rate k. Apart from an expected increase in performed work, the results in the two-miner scenario remain approximately the same as described in Section IV-A. In the multi-miner case, large miners remain disadvantaged, however at a slightly smaller scale. The relative difference between the average computational effort of the smallest and largest miner decreases from to 5% to approximately 3%, while the effects on the return per computed MH are negligible. Detailed simulation results are provided in Appendix VII-A.

V. Modelling Attacks

In observations of the Ethpool mining pool we have noticed behavioural artefacts, such as occasional donations of hashing power by one miner to another or sudden drop of hash rate of a top ranked miner. In this section we want to explore the motivations behind such behaviour. To this end, we extend our model by allowing miners to withhold valid shares, donate their mining power in a tactical manner and maintain multiple wallets in a pool. We further provide simulations for the introduced attacks in a scenario with two miners and discuss their effectiveness in Subsection V-D.

A. Share withholding

We assume that the queue-based reward scheme introduces a new attack scenario, allowing malicious miners to increase their profits at the expense of other miners in the pool.

Looking at other schemes, it may seem that reaching the top of the miner ranking as often as possible appears to be the highest rewarding strategy. However, since the credits resetting policy and hence the new credits of a miner winning a block depend solely on the credits of the miner ranked second, the optimal strategy is different. Instead of simply trying to win the next block, a miner can increase her long term revenue by winning the next block when there is a large gap between her and the second placed miner's credits. We describe a possible attack strategy in the example below.

TABLE IV: Profit improvement by exploiting a non-uniform credits dispersion. Alice stops submitting shares (a), allows Bob to pass (b) and profits from the new queue constellation (c)–(f).

(a) Before block i

Position	Miner	Credits
1	**Alice**	**110**
2	Bob	105
3	Dave	55

(b) Block i found

Position	Miner	Credits
1	Bob	115
2	**Alice**	**110**
3	Dave	60

(c) After block i

Position	Miner	Credits
1	**Alice**	**110**
2	Dave	60
3	Bob	5

(d) Block i+2 found

Position	Miner	Credits
1	**Alice**	**120**
2	Dave	65
3	Bob	15

(e) After block i+2

Position	Miner	Credits
1	Dave	65
2	**Alice**	**55**
3	Bob	15

(f) After block i+3

Position	Miner	Credits
1	**Alice**	**65**
2	Bob	25
3	Dave	10

We make use of a simplified version of our example from Section III-B. The modified setup is shown in Table IVa: Alice, in our case the attacker, is ranked first, only a few credits ahead of Bob. We observe a significant gap between the credits of the second and third placed miners. Again, we assume the two large miners, Alice and Bob, constantly earn 10 credits per round, while the small miner, Dave, earns 5. Furthermore, we assume the ranking is visible to all miners, as in the case of Ethpool [4].

By comparing the differences in credits between the first and second (Alice vs Bob) and second and third (Bob vs Dave) ranked miners, it can be seen that Bob will benefit a lot more from the credit-resetting mechanism than Alice, should this order sustain, namely 50 in contrast to 5 credits. Therefore, Alice is incentivized to stop submitting shares, this way allowing Bob to win the next block (cf. Table IVb). In the next round, Alice will be ranked first and now profits from the large difference between her and the next miner's credits (cf. Table IVc). As a result, Alice will be re-positioned in the ranking ahead of both Dave and Bob, gaining a significant advantage for the next few rounds (cf. Table IVe). This theoretical example shows that although the underlying motivation for such a credit resetting policy is to reward large miners for their above average work, Bob finds himself in a situation of having been cheated out of a significant amount of credits, which negatively impacts his long-term revenue.

B. *Tactical donation of mining power*

In order to further improve the chances of Bob overtaking her, Alice can dedicate her mining power to Bob, by spoofing the payout address she uses when submitting shares. According to the current implementation of the mining protocol, the mining pool operator will believe Bob has increased his mining power, hence rewarding him with more credits. Assuming Bob does not respond to this “forced-donation”, Alice will end up even more likely in the same favourable position as discussed in the previous sub-section. An observed real-world occurrence of such a donation strategy is shown in Figure 10, where a miner in Ethpool receives an unexpected boost of mining power when they are about to win the next block.

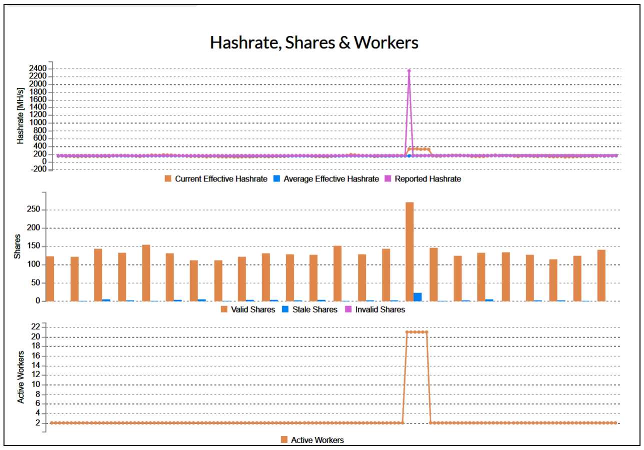

Fig. 10: Screenshot: A miner in Ethpool receives an unexpected donation of computing power at around the time at which they win a block – generosity or self-interest on behalf of the donor?

C. *Using multiple payout addresses*

Due to the pseudo-anonymity constraints in permissionless blockchains such as Ethereum, the mining pool operator cannot prevent miners from using multiple payout addresses in parallel. While miners gain no advantage from this in conventional schemes such as PPLNS, maintaining multiple accounts can be used to optimize revenues in the queue-based scheme.

Since in the current implementations, miners are not fairly rewarded for hash rate overhead, a potential solution is to start mining for other payout addresses, once reaching the top of the queue. This strategy be can be applied as an improvement to the exploitation of the non-uniform credits distribution: instead of ceasing to submit shares or investing mining power in another miner, an attacker can dedicate overhead mining capacity to her other addresses.

D. *Simulating attack scenarios*

In order to model the three different attack scenarios introduced in Section V, we use the same initial simulation setup as for the two-miner case from Subsection IV-A. For each of the three different scenarios, the following *attacking condition* prevails: the 10 GH/s miner attacks as soon as the 1 GH/s miner’s credits reach the 90% threshold of the attacker’s credits[10].

TABLE V: Attack simulation results in a two-miner scenario.

Attack strategy	Miner	Average performed work (trillion hashes)	Blocks		
			Rewarded	Mined	Ratio
Share withholding	Attacker Victim	194.398 260.105	456 433 43 567	443 476 56 524	1.0292 0.7707
Tactical donation of mining power	Attacker Victim	189.85 349.775	468 678 31 322	445 227 54 773	1.0527 0.5719
Using a second wallet	Attacker Victim	220.61 253.53	457 161 42 839	445 668 54 332	1.0258 0.7885

Recall that the first attack described was share withholding, whereby the attacker stops the submission of shares with the aim to significantly increase the probability of the victim surpassing her before the next block is found. Once the small miner wins the block, the large miner would continue her work. After 500 000 blocks, following this strategy the attacker was rewarded an additional 12 957 blocks more than the number of blocks she actually mined (Table V).

The second attacking scenario simulated is the tactical donation of mining power, whereby the large miner directs his submitted shares to the smaller miner’s payout address. This has the effect of temporarily increasing the hash rate and thereby the credits of the small miner. Following such a behaviour, the attacker received an extra 23 451 blocks.

The third attacking strategy discussed was the systematic use of multiple payout addresses. For simplicity, we simulated only one additional payout address, or a second wallet, for the 10 GH/s miner. Each time the attacking condition was met, the large miner redirected her hash rate/mining power to her second wallet. Thereby the attacker did not have to give away any of her credits to the victim. For applying this attack strategy, the 10 GH/s miner received rewards for 11 493 additional blocks.

The reason why this last attack performs worse in terms of extra blocks rewarded than the first two attack scenarios is due to the second wallet itself. By including the attacker’s second wallet we are essentially adding a third miner to the simulation. However, this can lead to the scenario in which the attacker’s own wallet eats into her first wallet’s credits. The attacker therefore requires a strategy explicitly for protecting herself against such unfortunate queue constellations in order to increase her profits. However, we do not provide detailed suggestions for an optimal solution in this paper.

[10] After having tested and compared results using multiple thresholds, 90% proved to be the most rewarding.

E. Other attack vectors

Two other attack vectors include *pool-hopping* and the withholding of blocks from the mining pool. As described by Rosenfeld [20], pool-hopping refers to a miner's practice of dynamically switching between different pools in order to increase profits. In such scenarios, miners join a mining pool only when the expectation of earning rewards is high and leave as soon as the expectation drops, thereby increasing the variance of the mining pool's total hash rate. As a consequence, the expected revenues of non-pool-hopping miners decrease, making the mining pool less attractive compared to pool-hopping resistant pools.

In the current implementation of the Ethereum PoW protocol, miners are able to determine whether a found solution to the mining pool's problem also represents a solution to the network's PoW puzzle. Consequently, a miner can decide to withhold such blocks from the mining pool, which is generally referred to as block withholding. Depending on the pursued goal, an attacker can either simply damage the mining pool as a whole [20], or gamble to increase their own revenue at the cost of other miners or the mining pool operator [7].

We note these two attack strategies are not specific to the queue-based reward distribution scheme. Rather, they are applicable to mining pools regardless of the underlying payout mechanism. We therefore leave the evaluation of these attack scenarios to future work.

VI. Conclusion and Future Work

We have conducted what we believe to be the first academic study of the queue-based reward payout approach implemented by Ethpool, and compared it to the Pay-Per-Last-N-Shares approach implemented in Ethermine. We have created a discrete event-based simulation model to analyse whether the queue-based scheme offers a fair return on investment, both for miners with large hash rates (the sharks) and those with small hash rates (the fish). From our simulation results we have seen that in a two-miner scenario and, more significantly, in the case of Ethpool, a large miner is at a disadvantage compared to the small miner(s). When compared to Ethermine's PPLNS scheme it could be seen that a large miner in Ethpool had to perform significantly more hashing operations per block won than a small miner. Obviously, miners find strategies to optimise their revenue. Real-world data from Ethpool indicates that some miners with high mining power have noticed this disadvantage and attempt to compensate for it through the exploitation of the credit resetting policy. We highlighted three different potential attacking scenarios stemming from this non-uniformity: the stalling of mining power, a tactical donation of mining power, and the use of multiple payout addresses. From our attack simulation it could be seen that attackers can indeed strategically manipulate queue constellations, receive a substantial number of additional blocks and thereby offset their initially skewed work per block ratio.

It should be noted that we have demonstrated the existence of attacks specific to the current implementation of the queue-based reward payout scheme. We modelled these attack scenarios assuming the victim miners do not defensively respond and have ignored possible pool-hopping scenarios as part of a second wallet strategy. A thorough game-theoretic analysis of such behaviour entailing multiple attackers in a multi-miner scenario, as well as an investigation into protective mechanisms to resist such exploitation attempts, could thus prove to be a fruitful avenue of future research.

VII. Appendix

A. Results under Exponential Difficulty Growth

TABLE VI: Miner performance in the multi-miner scenario under exponential difficulty increase with growth rate $k = 2.726 \times 10^{-6}$.

Hash rate	Miners	Blocks			Average performed work (trillion hashes)	Average revenue per computed MH (10^{-8} ETH)		
		Rewarded	Mined	Ratio		Ethpool	Ethermine	Ratio
22.99 MH/s	5	15	13	1.1449	414.30	1.1919	1.2350	0.9651
29.9 MH/s	4	20	21	0.9765	414.63	1.1914	1.2148	0.9807
42.29 MH/s	15	29	29	1.0091	417.87	1.1804	1.1956	0.9873
57.75 MH/s	21	40	40	0.9907	417.34	1.1838	1.1932	0.9921
78.96 MH/s	33	55	55	1.0165	420.02	1.1769	1.1798	0.9975
107.81 MH/s	48	76	74	1.0234	420.80	1.1749	1.1745	1.0003
152.1 MH/s	75	107	106	1.0161	422.06	1.1719	1.1702	1.0015
203.07 MH/s	44	144	143	1.0073	422.53	1.1707	1.1677	1.0026
284.15 MH/s	78	202	200	1.0101	423.14	1.1693	1.1661	1.0027
384.66 MH/s	74	273	272	1.0023	423.22	1.1692	1.1656	1.0031
538.73 MH/s	73	383	382	1.0020	423.74	1.1679	1.1643	1.0031
729.49 MH/s	54	518	519	0.9996	423.98	1.1673	1.1638	1.0030
988.35 MH/s	55	703	702	1.0018	424.39	1.1662	1.1634	1.0024
1.39 GH/s	48	987	982	1.0054	424.78	1.1652	1.1631	1.0018
1.84 GH/s	24	1306	1302	1.0031	425.15	1.1642	1.1629	1.0011
2.54 GH/s	27	1802	1808	0.9969	425.60	1.1630	1.1626	1.0003
3.59 GH/s	18	2548	2551	0.9987	426.05	1.1618	1.1623	0.9996
4.99 GH/s	15	3534	3529	1.0013	426.52	1.1605	1.1622	0.9985
6.72 GH/s	8	4748	4772	0.9949	426.88	1.1596	1.1621	0.9978
9.91 GH/s	6	7053	7119	0.9908	427.37	1.1582	1.1621	0.9966
13.47 GH/s	3	9464	9510	0.9952	427.81	1.1570	1.1620	0.9957
18.07 GH/s	1	12858	12960	0.9921	428.05	1.1564	1.1620	0.9952

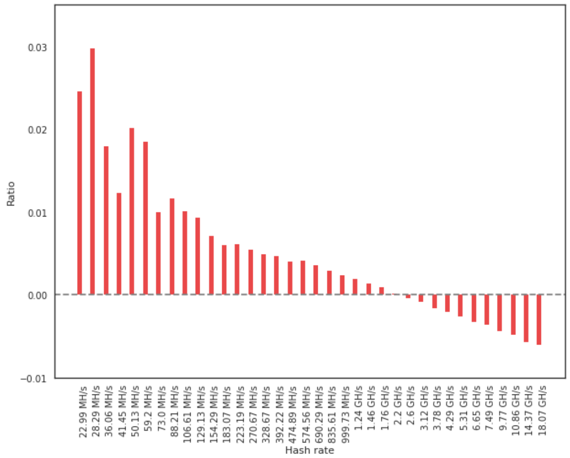

Fig. 11: Ratio of return per computed MH for miners in Ethpool, relative to the average return per computed MH under exponentially increasing difficulty with growth rate $k = 2.726 \times 10^{-6}$. Miners are grouped by hash rate.

VIII. Acknowledgements

The authors would like to thank Iain Stewart for helpful discussions and insightful observations. Katinka Wolter contributed to this work while on sabbatical leave at Imperial.

References

[1] Ethereum statistics. https://ethstats.net/. Accessed: 2017-06-18.

[2] Ethpool mining pool. http://ethpool.org/. Accessed: 2017-06-18.

[3] Ethpool public API. http://ethpool.org/api/credits. Accessed: 2017-06-18.

[4] Ethpool reward payout scheme. http://ethpool.org/credits. Accessed: 2017-06-18.

[5] bitfly e.U. Terms of service. http://bitfly.at/GTS_v1.0.pdf. Accessed: 2017-06-18.

[6] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper, 2014. Accessed: 2017-06-18.

[7] N. T. Courtois and L. Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

[8] Ethereum community. Ethash. https://github.com/ethereum/wiki/wiki/Ethash. Accessed: 2017-06-18.

[9] Ethereum community. Ethereum mining rewards. https://github.com/ethereum/wiki/wiki/Mining#mining-rewards. Accessed: 2017-06-18.

[10] Ethereum community. Mining. https://github.com/ethereum/wiki/wiki/Mining. Accessed: 2017-06-18.

[11] Ethereum community. Proof of stake FAQ. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ. Accessed: 2017-06-18.

[12] I. Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[13] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*, pages 72–86. Springer, 2014.

[14] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[15] A. Laszka, B. Johnson, and J. Grossklags. When bitcoin mining pools run dry. In *International Conference on Financial Cryptography and Data Security*, pages 63–77. Springer, 2015.

[16] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[17] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 397–411. IEEE, 2015.

[18] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, Dec 2008. Accessed: 2017-06-18.

[20] M. Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[21] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. *Financial Cryptography and Data Security*, 2016.