Nash Equilibrium of Multiple, Non-Uniform Bitcoin Block Withholding Attackers

Sean Elliott
Southern Methodist University
Dallas, TX, USA
sjelliott@smu.edu

***Abstract*—This research analyzes a seemingly malicious behavior known as a block withholding (BWH) attack between pools of cryptocurrency miners in Bitcoin-like systems featuring blockchain distributed databases. This work updates and builds on a seminal paper, The Miner's Dilemma, which studied a simplified scenario and showed that a BWH attack can be rational behavior that is profitable for the attacker. The new research presented here provides an in-depth profit analysis of a more complex and realistic BWH attack scenario, which includes mutual attacks between multiple, non-uniform Bitcoin mining pools. As a result of mathematical analysis and MATLAB modeling, this paper illustrates the Nash equilibrium conditions of a system of independent mining pools with varied mining rates and computes the equilibrium rates of mutual BWH attack. The analysis method quantifies the additional profit the largest pools extract from the system at the expense of the smaller pools. The results indicate that while the presence of BWH is a net negative for smaller pools, they must participate in BWH to maximize their remaining profits, and the results quantify the attack rates the smaller pools must maintain. Also, the smallest pools maximize profit by not attacking at all — that is, retaliation is not a rational move for them.**

***Index Terms*—Distributed databases, Bitcoin, cryptocurrency, blockchain**

I. Introduction

Virtual currencies such as Bitcoin perform the same fundamental functions as traditional money. Instead of transactions involving cash, gold, and banks, however, this revolutionary digital money is powered by mathematics, code, and computers located in any country and under any government around the globe [1]. In the ten years since its inception, Bitcoin has grown into the largest and most successful cryptocurrency ever. The Bitcoin blockchain, a distributed public ledger of every Bitcoin transaction ever performed, is secured by a decentralized process called block mining. Today, Bitcoin participants worldwide, known as miners, altogether burn over 2 gigawatts of power running cryptographic hash functions in a race to generate the next block of transactions on the blockchain [2]. As of December 2018, approximately $6,000 per minute worth of Bitcoin revenue is available for winning miners, though that is down from a peak of over $30,000 per minute in 2017 [3].

The Bitcoin system depends on thousands of participating network computer nodes quickly and robustly achieving distributed, decentralized consensus on the validity of a new block of transactions to add to the blockchain, a public ledger of all Bitcoin transactions since inception. Many sources describe the details of the process [1] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]. This paper focuses on specialized computation nodes known as miners, and in particular the behavior of pools of Bitcoin miners. Bitcoin miners compile a block of transactions and race to solve a Proof of Work (PoW) puzzle so the network will adopt their block over all others. Reports of a new block are propagated in seconds throughout the network, while the difficulty of the PoW is such that on average a new block will be found every 10 minutes. Therefore, consensus about a new block is very likely to be achieved before the next PoW is solved. In this way, the miners are essential to securing the blockchain and are rewarded for their participation. In 2018, the miner who solves the PoW receives 12.5 new Bitcoins and typically an additional fraction of a Bitcoin in transaction fees. This reward for supporting and securing the system with mining is worth about $60,000 and occurs on average every 10 minutes around the clock.

The PoW puzzle solving process is a repeated cryptographic hash on the transaction block contents. The miner tries different 256 bit random seeds until the hash result meets the PoW criteria. Motivated by the block reward, miners run specialized hardware to compute hashes as quickly as possible, often many trillions of times a second, essentially converting electricity into Bitcoin. Viewed as a random process, the miners in effect search through a uniform distribution of random seed numbers, and the result is a Poisson process with an exponential distribution of time between successful searches. Thus, while there is substantial motivation to become a miner, the payoff variance in time is high and memoryless: a single mining unit, a solo miner, can work for months or years without receiving a reward. For example, with a state-of-the-art mining hardware "rig," such as the Antminer S15, it would take a solo miner a year on average to solve a single PoW puzzle [15]. However, since the process is both random and memoryless, a solo rig could mine for a year unsuccessfully and still be no closer to solving a PoW puzzle, earning nothing in the meantime [12]. The vast majority of miners now join pools as depicted in Figure 1 and share rewards to reduce this reward time variance and instead obtain much more steady payments [6]. In fact, mining pools, which are an ad hoc addition to the system not present in the original Bitcoin design, have increased in popularity along with the growth of world wide hash power such that today about 99% of all hashing is performed within

978-1-7281-2080-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDIS.2019.00029

pools [3] [17] [14]. However, pools are a fundamental change to the Bitcoin ecosystem because they impose centralization on a system that was intended to be decentralized, and they require trust in a system that was designed to be trustless [12].

Researchers have theorized that these changes toward pooled hash power have led to vulnerabilities in the system. For example, if one entity controls more than 50% of the worldwide hash power, it has long been believed that the system would be susceptible to a host of malicious actions like double spending and selfish mining [5] [6] [10]. Furthermore, researchers have shown that pools can launch successful attacks even if they control less than the majority of worldwide hash [5] [18] [19] [20]. With pooling, there is a new risk of one entity controlling even more than 50% of mining power because the pool manager does not have to make the capital investment to purchase the hardware directly. At this time, the only incentive a pool has to refrain from owning more than 50% is to satisfy community complaints and garner goodwill. One pool, GHash.IO, already controlled over 50% of all worldwide hash power at one point, but reduced its market share in response to industry pressure [21] [7] [17]. Unfortunately, this is not a stable or incentive compatible situation, meaning that the choices of rational miners do not optimize the welfare of the pool as a whole [5] [11] [12]. There are many ways for multiple pools to collude behind the scenes, outside of the Bitcoin system, either by one pool-managing-entity actually controlling multiple pools, or by managers of multiple pools acting in unison. This type of activity, which hides the true hash power of a Bitcoin entity, is known as laundering hashes [8] [17] [9]. Many of the types of attacks arising from pooled mining described above would be easy to perform, and there are no safeguards in place to prevent them. [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [22].

This paper focuses on the block withholding (BWH) attack between open pools in a Bitcoin-like blockchain system as depicted in Figure 2 [7] [6] [13]. In a BWH attack, a pool of miners sends a fraction of mining power to another pool, a BWH victim pool, and does not report any PoW solutions found for the victim pool (*withholding* them) while still collecting the shared reward generated by non-attacking miners in the victim pool. In essence, the BWH attack reduces the effectiveness of the victim pool while increasing the effectiveness of the attacker's normal pool. Note that the attacker can not obtain the system reward for the withheld solution himself because the pool's public key is included in the blocks mined [15]. BWH attacks are not simply malicious hacks. They can be the actions of rational pool managers because the attacks can be quantifiably profitable for the attackers, especially for large pools. Using profit models of simplified representations of Bitcoin-like mining pools, researchers have shown that various Nash equilibrium states exist of engaging in, rather than refraining from, BWH attacks [7] [8]. Furthermore, BWH behavior within a victim pool is easy to disguise because the solutions are rare. A victim pool manager can detect the likely presence of an attacker in the pool, but it is very difficult to identify the culprit [7]. To prevent BWH, some researchers have proposed disincentivizing BWH by paying pool miners a bonus when they solve a block, adding a 2-step validation to the pool mining protocol so that miners are unaware when they have solved a block, using only small and closed pools of trusted miners, and other methods [4] [7] [11] [13]. At this time, there is not a consensus among users on which approach to adopt and how to best prevent BWH. The BWH attack remains an ongoing and unsolved problem in Bitcoin and other newer cryptocurrencies that are mined in pools and use a traditional PoW consensus protocol in which a miner is aware he has completed a PoW [22]. Decentralized mining pools such as Smartpool and P2Pool are also susceptible to BWH [14] [13].

The profitability of BWH attacks was analyzed in [7], where researchers presented games with one pool conducting a BWH attack, and with two symmetric pools attacking each other, using a maximum pool size of 50% of the worldwide hash. Building on and updating [7], this present paper expands the analysis to include BWH attacks among multiple mining pools of non-symmetric sizes, including pools attacking back. The methods used in this research also allow expanding a pool size out near the limit of 100% since, as mentioned earlier, pools can hide their true size and secretly control more than 50% of worldwide hash. These equations help in analyzing whether a BWH attack will be profitable and how much hash power is required, as a function of attacker hash power, with both closed and open pools. The resulting analysis is more general purpose and gives a more complete picture of profitability than other existing studies. Section II provides further background on pooled mining, highlighting assumptions made for the subsequent analysis of BWH. A detailed look at the empirical and mathematical analysis of BWH attacks is given in section III. Finally, conclusions and ideas for future work are covered in section IV.

II. Pool Mining Background

A Bitcoin mining pool manager sends out work to his miners in the form of hashing assignments consisting of a Block Template and a range of random numbers to use in the hash function. The Block Template contains the public key of the pool manager and the transactions he selected for inclusion in the block. Any PoW solution found by any miner with that Block Template will result in the block reward going to the holder of the included public key: the pool manager. Miners must independently trust the manager to fairly reward the miner for his work, including the miner who found the PoW. Also, the pool manager runs a full Bitcoin node on the open, pseudonymous, peer-to-peer network, alleviating his miners from the tasks of communicating with the full system and maintaining a copy of the full blockchain, which is 195 GB as of December 2018 [23]. In fact, with the prevalence of pooling, a tiny percentage of Bitcoin nodes represent a large majority of the mining power [17]. When a solution is broadcast on the network, all Bitcoin network nodes verify that the solution, all transactions in the block, and the block itself are valid. This is the step that authenticates and secures

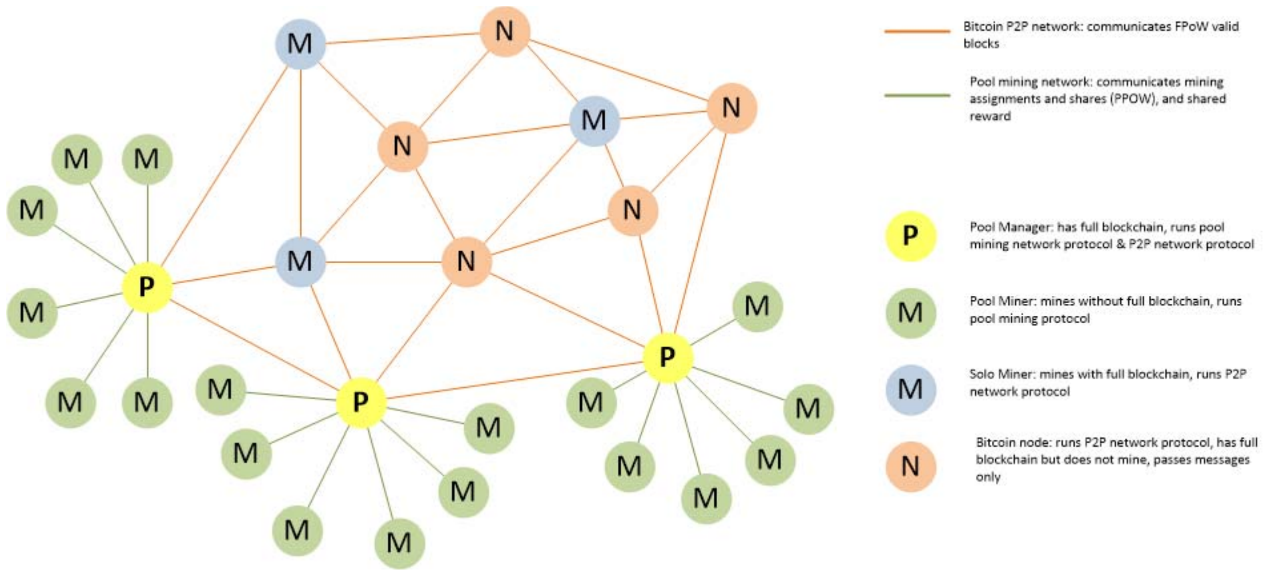
Bitcoin P2P network: communicates FPoW valid blocks
Pool mining network: communicates mining assignments and shares (PPOW), and shared reward
Pool Manager: has full blockchain, runs pool mining network protocol & P2P network protocol
Pool Miner: mines without full blockchain, runs pool mining protocol
Solo Miner: mines with full blockchain, runs P2P network protocol
Bitcoin node: runs P2P network protocol, has full blockchain but does not mine, passes messages only
M
N
P

Fig. 1. Bitcoin network with pooled and solo miners

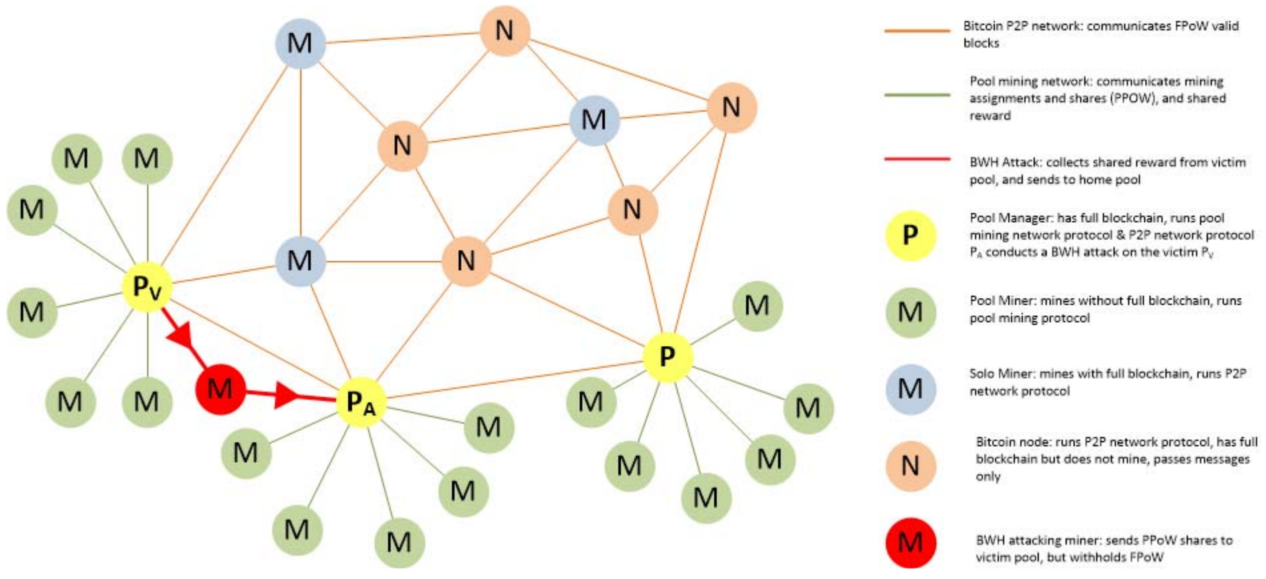
Bitcoin P2P network: communicates FPoW valid blocks
Pool mining network: communicates mining assignments and shares (PPOW), and shared reward
BWH Attack: collects shared reward from victim pool, and sends to home pool
Pool Manager: has full blockchain, runs pool mining network protocol & P2P network protocol P_A conducts a BWH attack on the victim P_V
Pool Miner: mines without full blockchain, runs pool mining protocol
Solo Miner: mines with full blockchain, runs P2P network protocol
Bitcoin node: runs P2P network protocol, has full blockchain but does not mine, passes messages only
BWH attacking miner: sends PPoW shares to victim pool, but withholds FPoW
P_V
P_A
P
M
N

Fig. 2. Bitcoin network with pool P_A attacking pool P_V

the blockchain. Message propagation through the network takes on the order of seconds, and the network nodes update their copies of the blockchain, resulting in system consensus on the next block in the chain [24] [10] [16]. After that, all pool managers and solo miners begin the next round by selecting and assembling pending transactions into another candidate Block Template. Pool managers then send their new blockchain hashing assignments to their miners, and all mining nodes start searching for a new PoW solution [4].

The Bitcoin PoW puzzle problem is expressed as:

$$SHA256(SHA256(BlockTemplate||Nonce)) \leq D \quad (1)$$

where D is the system difficulty target, the nonce is a one-time use random number, and $||$ is a concatenation function. Secure Hash Algorithm 256 (SHA256) is a cryptographic hash function from the National Security Agency with many applications besides Bitcoin [25]. The algorithm has the properties that the operations themselves are easy to calculate, but working backwards from a given hash result to determine the original string is computationally infeasible. In the case of Bitcoin, this means that it is virtually impossible for a miner to take a shortcut and directly compute the nonce that solves the puzzle. Instead, the miner must use trial-and-error to find a suitable input nonce. However, given the nonce and the Block

Header, it is computationally easy for any Bitcoin participant to confirm that they produced the desired hash meeting the requisite difficulty, D. The value of D, the maximum number that can fulfill the difficulty requirement, as of December 2018 is approximately 5.28×10^{54}, which means a suitable hash result will be a 256-bit integer with at least 73 leading zeros. In other words, 21.9×10^{21} hash must be computed on average to find a full PoW, which translates to a 37.7×10^{18} hash/second rate system-wide for a winner every 10 minutes.

While searching for a PoW, a miner within a pool reports easier to find "shares," also called partial PoW (PPoW), to the pool manager. A share is a PoW solution meeting the pool's target difficulty d, which is much easier than the system difficulty, therefore $d \gg D$. Typically 32 leading zeros are required to be considered a share, in contrast to the Full PoW (FPoW) which has many more leading zeros required for the system block reward. Pool miners report PPoW shares continuously to the pool manager while they search for the FPoW. The purpose of the share is to provide the pool manager a mechanism to monitor miner hash rates. The shares will show that a miner is legitimately working and provide a quantitative basis for sharing the eventual block reward. The pool manager gives each miner a range of nonce values in which to search, many of which will result in PPoW [8] [17]. There is a probability D/d that any share solution will be less than D as well, meaning it is a solution that also fulfills the system-wide target. For example, using current figures, if d has 32 leading zeros and D has approximately 73, then a share would have a probability 2^{41} of being a solution for the system-wide puzzle as well. For the analysis in this paper, the shared reward of pooled miners is assumed to be continuous and proportional to miner hash rate, which is very nearly the case with a high rate of share/PPoW production.

Pool managers use various algorithms for dividing the reward among miners [4] [11] [12]. In earlier, simpler payment methods such as the Proportional method, after one miner solved a FPoW, the pool manager divided that reward among pool participants, weighted by share contribution since the last reward. That led to a behavior known as pool-hopping where miners joined a pool that had a recent win and then hopped to another winner after a short time. The objective in the pool-hopping behavior was to minimize diluted shares for the individual miner by leaving pools with a long time since a win, but it reduced the expected reward rate for the pool manager and miners who did not hop. Two more recent reward schemes are PayPerShare and the modified proportional scheme PayPerLastNShares, which are the two most popular payment schemes among large pools [15] [26]. In the PayPerShare reward scheme, the miner receives a set fee whenever he produces a share, without waiting for the pool to solve a FPoW. While the scheme avoids pool-hopping — there is no benefit or penalty for a miner leaving or joining this pool — it requires the pool manager to invest in a reserve of Bitcoin to use for rewards when the time between FPoW rewards is long [4] [9] [11] [12]. The large pools as of this writing use the PayPerShare model [15]. In PayPerLastNShares, which is a hopping-resistant, modified proportional scheme, when a block reward comes to the pool, the manager divides the reward evenly over the last N shares, which is a moving window of the recent participants. Miners have an incentive to stay with a pool to help it win more quickly and to keep their shares within the window. Since the reward is paid out after a win as with the original Proportional method, no reserve of reward Bitcoins is required [9] [12]. Some pool managers offer other promotions and prizes such as a bonus to a miner for submitting a full solution. In any algorithm, the pool manager is compensated for his services, typically charging his members fees of 0-3% of his pool's revenue [15]. For the analysis in this paper we will assume equilibrium conditions such that pool membership reaches steady state, participants do not hop, and the reward is exactly proportional to the hash rate without additional fees or rewards. More complex and time dependent activities and reward schemes are left for future work.

III. Model, Analysis and Results

A. Equilibrium Equations for Block Withholding

Consider a Bitcoin-like system consisting of both open and closed pools and a world-wide total hash rate normalized to 1. A miner can join any open pool anonymously, and with reasonable diligence, withhold full Proof of Work (FPoW) solutions without detection by the pool manager while still collecting rewards for partial Proofs of Work (PPoW) shares, creating a Block Withholding (BWH) attack on the victim pool. In contrast, a miner can join a closed pool only after establishing trust outside of the blockchain system, the assumption being that trusted miners will not conduct BWH attacks. The system under consideration has P open pools, each with a hash capability of h_i for a total open pool system wide hash rate of $\sum_{i=1}^{P} h_i$. For all closed pools, not participating in BWH, the total hash rate is h_C. The total system (world-wide) hash rate consists of only open or closed pools, so $h_C + \sum_{i=1}^{P} h_i = 1$. The special case of closed pools that conduct BWH attacks is left for future work.

In the absence of BWH in a system, the expected value of the mining reward rate of pool i would equal the hash rate h_i in the normalized system considered here. When a BWH attack occurs, any pool reward rate is diluted by the hash rate of all attackers joining the victim pool. The victim pays out for shares without getting the full rate of system reward due to discarded FPoW results. If h_v is the hash rate of a victim pool under normal conditions, and h_a is the BWH hash rate allocated to the victim by attackers, then the dilution from BWH is given by (2):

$$BWH\ dilution = \frac{h_v}{h_v + h_a} \tag{2}$$

A victim may also be an attacker, and vice versa, in a system of many open pools. A rational pool manager needs to decide how much hash to allocate to normal mining, and how much hash to use to attack other open pools, and then how to allocate hash among those victims. Thus, a pool manager conducting a BWH attack will reduce his normal mining rate, which reduces his reward from that source, as well as reducing the overall world wide hash rate, while increasing his reward from block withholding.

The notation h_{ij} is used to indicate the allocation of pool i used for attacking pool j, with the special case of h_{ii} meaning the amount of hash pool i allocates for normal mining. So the expected value of the portion of the reward rate coming from normal mining, r_{Ni} , for pool i, taking into account the dilution of all other attacking pools, is given by (3), which is the normal hash rate times the dilution factor for that pool as a victim:

$$r_{Ni} = \frac{h_{ii}^2}{\sum_{k=1}^{P} h_{ki}} \tag{3}$$

Similarly, the expected value of the reward rate due to BWH attacks, r_{Bi}, of pool i against all other open pools is given by (4), where the attacking pool gets its share of the dilution caused by BWH:

$$r_{Bi} = \sum_{\substack{j=1 \\ j \neq i}}^{P} \frac{h_{jj} h_{ij}}{\sum_{k=1}^{P} h_{kj}} \tag{4}$$

Finally, the effective hash rate of any participant is increased due to all BWH attack hash removed from the competition for FPoW. Assuming the system difficulty level has adjusted such that only non-attacking hash, h_{ii}, are considered, the effectiveness of h_{ii} at finding FPoW is higher compared to a system with no BWH attacks. Equation (5) shows the effective reward rate on the system under consideration which increases due to BWH:

$$r_{eff} = \frac{1}{1 - \sum_{k=1}^{P} \sum_{\substack{l=1 \\ l \neq k}}^{P} h_{kl}} \tag{5}$$

It is important to note that removing hash from FPoW efforts to use for BWH attack will not result in an improved reward *rate* until the 2 week system difficulty adjustment occurs [7]. Until the D value adjusts to the removed hash, although the fraction of blocks won will reflect the normal FPoW competition, the mean system period will be longer than 10 minutes. A more realistic scenario for a pool manager may be the case where new hash hardware has been acquired and the decision is where to allocate: BWH attack or FPoW. The analysis for this paper assumes an equilibrium is reached after all hash allocations have been made, such that the difficulty adjustment has a $\Delta = 0$. Therefore, the effect on system difficulty is factored out, and reward rates can be calculated based on expected values using the constant hash allocation weight h_{ij}. Also, the total hash hardware is assumed to stay constant during the analysis period.

Combining all the contributors to the reward for pool i, the total reward for pool i is given in (6):

$$\begin{aligned} r_{Ti} &= r_{eff}(r_{Ni} + r_{Bi}) \\ &= \frac{1}{1 - \sum_{k=1}^{P} \sum_{\substack{l=1 \\ l \neq k}}^{P} h_{kl}} \times \\ &\left(\frac{h_{ii}^2}{\sum_{k=1}^{P} h_{ki}} + \sum_{\substack{j=1 \\ j \neq i}}^{P} \frac{h_{jj} h_{ij}}{\sum_{k=1}^{P} h_{kj}} \right) \end{aligned} \tag{6}$$

The state of BWH at any given moment in the system can be expressed by the $P \times P$ matrix of h_{ij} hash values shown in (7). The diagonal contains the h_{ii} values which represent normal hashing, and all the other h_{ij} values represent hash applied to i attacking j:

$$H = \begin{matrix} h_{11} & h_{12} & h_{13} & \dots & h_{1p} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2p} \\ h_{31} & h_{32} & h_{33} & \dots & h_{3p} \\ \vdots & \vdots & \vdots & \ddots & h_{(p-1)p} \\ h_{p1} & h_{p2} & h_{p3} & h_{p(p-1)} & h_{pp} \end{matrix} \tag{7}$$

At any given time, a rational pool manager will adjust his row of hash values, changing the allocation of hash between normal and attack, as well as changing the relative amount of hash applied to his BWH victims. We will show that there is a Nash equilibrium where all pool managers have maximized their total reward with h_{ij} adjustments, and no further changes are possible that improve the reward. We will show the equilibrium point is where the partial derivative of the reward r_{Ti} with respect to each hash attack adjustment h_{ij}, $i \neq j$ is zero as expressed in (8). Demonstrating also that the 2nd derivative is negative is left for future work. Given a constant total hash $h_i = \sum_{k=1}^{P} h_{ik}$ for the pool — the total amount of hash hardware stays constant during the time period studied in this analysis — there are $P - 1$ independent hash variables to adjust in a row. For the analysis method described here, the allocation of non-attacking hash h_{ii} is considered a dependent variable.

$$\text{Equilibrium: } \forall i, j; i \neq j; \frac{\delta r_{Ti}}{\delta h_{ij}} = 0 \tag{8}$$

Some additional definitions are made here in order to produce more compact equations when taking the derivative of the reward equations. Considering partial derivatives of r_{Ti} with respect to h_{ij} one at a time with all other BWH attack values in the H matrix constant, the normal hash value h_{ii} of attacker i has the previously explained dependence on changes in h_{ij} such that $\delta h_{ii} = -\delta h_{ij}$. With the rest of H constant, to find $\delta r_{Ti} / \delta h_{ij}$, the hash available to allocate between h_{ii} and h_{ij} is defined to be $h_{ij0} = h_{ii} + h_{ij}$. This paper defines $u, 0 \leq u \leq 1$ to be the allocation of h_{ij0} between h_{ii} and h_{ij} for the purposes of taking the derivative, such that $h_{ii} = h_{ij0} u$ and $h_{ij} = h_{ij0}(1 - u)$. Substituting for h_{ii} and h_{ij} and using the scaling factors A_1 to A_3 and coefficients C_1 to C_3 defined below, the reward can be cast as (9) and results in (10) for the derivative:

$$r_{Ti} = \frac{A_1}{(C_1+u)}\left(\frac{A_2u^2}{(C_2+u)} + \frac{A_3(1-u)}{(C_3-u)} + C_4\right) \tag{9}$$

$$\begin{aligned}\frac{\delta r_{Ti}}{\delta h_{ij}} &= \frac{-\delta r_{Ti}}{\delta u} \\ &= \frac{-A_1}{(C_1+u)}\left(\frac{A_2u(2C_2+u)}{(C_2+u)^2} + \frac{A_3(1-C_3)}{(C_3-u)^2}\right. \\ &- \left.\frac{1}{(C_1+u)}\left(\frac{A_2u^2}{(C_2+u)} + \frac{A_3(1-u)}{(C_3-u)} + C_4\right)\right)\end{aligned} \tag{10}$$

$$A_1 = \frac{1}{h_{ij0}} \tag{11}$$

$$C_1 = \frac{1 + h_{ij} - \sum_{k=1}^{P}\sum_{\substack{l=1\\l\neq k}}^{P} h_{kl}}{h_{ij0}} - 1 \tag{12}$$

$$A_2 = h_{ij0} \tag{13}$$

$$C_2 = \frac{\sum_{k\neq i}^{P} h_{ki}}{h_{ij0}} \tag{14}$$

$$A_3 = h_{jj} \tag{15}$$

$$C_3 = \frac{\sum_{k\neq i}^{P} h_{kj}}{h_{ij0}} + 1 \tag{16}$$

$$C_4 = \sum_{\substack{l=1\\l\neq i,j}}^{P} \frac{h_{ll}h_{il}}{\sum_{k=1}^{P} h_{kl}} \tag{17}$$

B. Matlab Analysis

This section describes two methods to model and analyze non-uniform BWH attacks using Matlab.

1) BWH reward game: The first modeling method casts the BWH attacks as a multivariate Nikaido-Isoda function [27] with and without relaxation [28] to iterate to an approximate Nash equilibrium point for all modeled Bitcoin pools. In the Nikaido-Isoda function, a collective action is denoted by the vector $\mathbf{x}$, which for this analysis will be represented by the state matrix H defined in (7). In other words, the collective action $\mathbf{x}$ represents all pools in the system selecting their hash allocation h_{ij}, with each individual action x_i equivalent to an update of the row h_i in the state matrix H. Nash equilibrium is achieved when all individual actions x_i are maximized for the reward function $r(\mathbf{x})$, and no further actions are possible that improve the individual reward.

The Nikaido-Isoda function given in (18) represents progress toward Nash equilibrium as the summation of the change in individual rewards as each individual (in this case individual pools) change their strategy based on the current state of the system. In this equation, y_i is the new action for individual i based on state $\mathbf{x}$, and $\mathbf{y}$ is the new system state after all individual actions y_i are taken.

$$\Psi(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{P}\left[r_i(y_i|\mathbf{x}) - r_i(\mathbf{x})\right] \tag{18}$$

The Nash equilibrium point is defined as $\mathbf{x}^*$ such that (19) holds.

$$\max_{y} \Psi(\mathbf{x}^*,\mathbf{y}) = 0 \tag{19}$$

In other words any change away from $\mathbf{x}$ results in a decrease of the summed reward function.

As a practical matter, in computation the Nash equilibrium point is only estimated, iterating until the Nikaido-Isoda function reaches a parameter ϵ.

$$\Psi(\mathbf{x}^*,\mathbf{y}) \leq \epsilon \tag{20}$$

The MATLAB implementation consists of a state matrix H and the total reward function for hashing with BWH described in (6). Modeling a Bitcoin-like system of P open pools of varying total hash power, the $P \times P$ matrix H captures the allocation of hash power from each pool, potentially attacking any other pool, and $\mathbf{x}$ represents the current contents of H at any time during the algorithm. $\mathbf{y}$ represents the next state after each pool, i, has updated their BWH allocation from x_i to y_i. The system starts with an initial $\mathbf{x}$, for example normal hashing only with no BWH attacks. At each round of the algorithm, each pool updates their y_i allocation and then the collective $\mathbf{y}$ is assigned to $\mathbf{x}$ and the round repeats.

During a round each pool uses the same strategy, which is a multivariate optimization of the reward function by modifying all self hash allocations, x_i, while other hash allocations $x_{j\neq i}$ stay constant. The optimization is constrained such that all hash allocations are positive values, and the sum of hash allocations for the given pool is a constant — i.e. the hash hardware is unchanged, only the allocation changes. This is implemented with the MATLAB *fmincon* function using the sequential quadratic programming (SQP) optimization algorithm [29] applied to the negative reward function, with inequality and equality constraints [30]. The minimization function is summarized in (21)

$$\min_{\substack{y_i\in[0,1]^P\\ \sum_{j=1}^{P} y_{ij}=h_i}} -r(y_i|\mathbf{x}) \tag{21}$$

To potentially improve convergence, a relaxation algorithm is applied which scales down the change of each step by a factor α. At each step s, the new system state is a weighted sum of the existing state and the recalculated hash weights of each pool as shown in (22). Although each pool optimizes their reward for the given state, the reward will be different once all the pool moves are applied. The relaxation algorithm approaches the equilibrium point more gradually as the state converges to the equilibrium point. The α term, which ranges from 0 to 1, controls the rate of change per step, and a value of 1 is equivalent to the algorithm without relaxation.

$$\mathbf{x}^{s+1} = (1-\alpha)\mathbf{x}^s + \alpha\mathbf{y}^s \tag{22}$$

TABLE I
H matrix and rewards for 3 non-uniform pools without BWH

	P1	P2	P3
Hash Allocations	0.45	0	0
	0	0.30	0
	0	0	0.25
Reward	0.45	0.30	0.25

2) First Derivative Minimization: The second method of finding a Nash equilibrium point is using MATLAB to find a numerical solution for the system such that all $P(P-1)$ derivatives described in (10) are equal to zero. The problem is set up as an error minimization problem with the objective to minimize the sum of the squares of all derivatives.

$$\epsilon = \sum_{i=1}^{P} \sum_{\substack{j=1 \\ j \neq i}}^{P} \left(\frac{\delta r_i}{\delta h_{ij}}\right)^2 \tag{23}$$

Once again the MATLAB *fmincon* function is used with the SQP optimizations and this time minimizing (23) with similar inequality and equality constraints. The minimization function is summarized in (24).

$$\min_{\substack{y_i \in [0,1]^P \\ \sum_{j=1}^{P} y_{ij} = h_i}} \sum_{i=1}^{P} \sum_{\substack{j=1 \\ j \neq i}}^{P} \left(\frac{\delta r_i}{\delta h_{ij}}\right)^2 \tag{24}$$

C. Results

As examples of BWH MATLAB analysis, Table I shows the starting H matrix and reward for 3 non-uniform pools with no BWH, and Tables II to IV show three methods to reach equilibrium with very similar results. Tables II and III demonstrate Nikaido-Isoda reward maximization without relaxation and then with relaxation applied. Table IV shows the result with derivative sum-squared minimization.

Next a study of 15 pools with a hash distribution in alignment with the top 15 Bitcoin pools [3] is shown in Table V. Table VI shows the hash allocation between BWH and FPoW of the top 5 pools. Notice the 5th pool does not participate in BWH as an attacker, which is true for the remaining 10 smaller pools as well. Table VII shows the change in reward due to BWH. Note that only the two largest pools benefit from BWH. The next two pools lose in the presence of BWH, but must participate in attacking in order to maximize their profit — their profit would be less if they did not attack at all. For the smaller pools, there is nothing to do — attacking does not produce a benefit.

IV. Conclusions and Future Work

This paper details a mathematical representation of BWH attacks among non-uniform attackers in a Bitcoin-like blockchain system. It also demonstrates two methods of numerical modeling for such a system in MATLAB with nearly identical results. The model and analysis presented in this paper show that pools attacking each other reach a Nash

TABLE II
3 non-uniform pools with BWH modeled using Nikaido-Isoda reward maximization without relaxation

	P1	P2	P3
Hash Allocations	0.2678635	0.0671588	0.0444996
	0.0966469	0.2077115	0.0141832
	0.0854895	0.0251298	0.1913172
Reward	0.4592130	0.2980476	0.2427393
Improvement	+2.05%	-0.65%	-2.90%

TABLE III
3 non-uniform pools with BWH modeled using Nikaido-Isoda reward maximization relaxation $\alpha = 0.25$

	P1	P2	P3
Hash Allocations	0.2678639	0.0671604	0.0444984
	0.0966460	0.2077103	0.0141839
	0.0854901	0.0251293	0.1913177
Reward	0.4592130	0.2980476	0.2427394
Improvement	+2.05%	-0.65%	-2.90%

TABLE IV
3 non-uniform pools with BWH modeled using derivative sum-squared minimization

	P1	P2	P3
Hash Allocations	0.2678634	0.0671592	0.0444992
	0.0966469	0.2077108	0.0141829
	0.0854897	0.0251300	0.1913179
Reward	0.4592130	0.2980477	0.2427393
Improvement	+2.05%	-0.65%	-2.90%

TABLE V
Hash Capability of the Top 15 Bitcoin Pools, December 2018 [3]

P1	P2	P3	P4	P5
0.214	0.177	0.130	0.110	0.097
P6	**P7**	**P8**	**P9**	**P10**
0.092	0.070	0.038	0.028	0.020
P11	**P12**	**P13**	**P14**	**P15**
0.010	0.005	0.003	0.002	0.002

TABLE VI
Hash allocation of top 5 Bitcoin Pools at BWH equilibrium

	P1	P2	P3	P4	P5
Hash Alloc.	0.130058	0.012333	0.004752	0.001365	0.000000
	0.016661	0.126962	0.003039	0.000000	0.000000
	0.013820	0.007794	0.121744	0.000000	0.000000
	0.012309	0.006913	0.000166	0.108856	0.000000
	0.010990	0.006172	0.000148	0.000000	0.097194
	0.010424	0.005854	0.000140	0.000000	0.000000
	0.007931	0.004454	0.000107	0.000000	0.000000
	0.004305	0.002418	0.000058	0.000000	0.000000
	0.003172	0.001782	0.000043	0.000000	0.000000
	0.002266	0.001273	0.000031	0.000000	0.000000
	0.001133	0.000636	0.000015	0.000000	0.000000
	0.000567	0.000318	0.000008	0.000000	0.000000
	0.000340	0.000191	0.000005	0.000000	0.000000
	0.000227	0.000127	0.000003	0.000000	0.000000
	0.000227	0.000127	0.000003	0.000000	0.000000

TABLE VII
REWARD PERFORMANCE, TOP 15 BITCOIN POOLS, AT BWH EQUILIBRIUM

	P1	**P2**	**P3**	**P4**	**P5**
Hash	0.214	0.177	0.130	0.110	0.097
Reward	0.217223	0.178894	0.129533	0.109421	0.096452
Change	+1.30%	+0.87%	-0.56%	-0.72%	-0.76%
	P6	**P7**	**P8**	**P9**	**P10**
Hash	0.092	0.070	0.038	0.028	0.020
Reward	0.091480	0.069605	0.037785	0.027842	0.019887
Change	-0.76%	-0.76%	-0.76%	-0.76%	-0.76%
	P11	**P12**	**P13**	**P14**	**P15**
Hash	0.010	0.005	0.003	0.002	0.002
Reward	0.009944	0.004972	0.002983	0.001989	0.001989
Change	-0.76%	-0.76%	-0.76%	-0.76%	-0.77%

equilibrium where large pools in particular are incentivized to attack for increased profits. While the presence of BWH is a net negative for smaller pools, they must participate in BWH to maximize their remaining profits. Other insights include that the smallest pools are instead better off not attacking even when being attacked. Analysis also illustrates what may otherwise be an intuitive assumption that if someone targets a pool that is already being attacked, the attacker's profit will be less. By demonstrating and quantifying Bitcoin's vulnerabilities such as BWH in this way, the hope is that the resulting insights into the technology ultimately will benefit the cryptocurrency ecosystem.

A recent work devised a specialized selfish mining attack, which also required a Sybil attack to be effective [13]. It would be interesting to include this type of attack in future analysis. Also, the equilibrium state of pools seems to tend towards ever larger pools; future work could simulate a system of pools unconstrained by industry pressure limiting pool size, to see what would be likely to happen in reality.

Other future work could model a more complex system representing the real world allocation of pools. Also, a quantitative representation of transient effects could be included such as the two-week difficulty update and bringing new hardware online. Other dynamic system behaviors may emerge since the equilibrium condition of the system is actually constant hash growth. In addition, the profitability of running a closed pool while attacking open pools could be studied. Finally, a similar analysis could be applied to other new cryptocurrencies and pool reward systems and include, for example, the pool manager's fee in the calculations.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
[2] Alex de Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.
[3] Blockchain Luxembourg S.A. Bitcoin hashrate distribution, 2018.
[4] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.
[5] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable. *arXiv preprint arXiv:1311.0243*, 2013.
[6] Nicolas T. Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *CoRR*, abs/1402.1718, 2014.
[7] I. Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103, May 2015.
[8] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas 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.
[9] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.
[10] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
[11] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.
[12] Ben Fisch, Rafael Pass, and Abhi Shelat. Socially optimal mining pools. In *International Conference on Web and Internet Economics*, pages 205–218. Springer, 2017.
[13] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 195–209, New York, NY, USA, 2017. ACM.
[14] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smart pool: Practical decentralized pooled mining. *IACR Cryptology ePrint Archive*, 2017:19, 2017.
[15] Yaron Velner, Jason Teutsch, and Loi Luu. Smart contracts make bitcoin mining pools vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 298–316. Springer, 2017.
[16] Ren Zhang and Bart Preneel. Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers Track at the RSA Conference*, pages 277–292. Springer, 2017.
[17] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. 2015.
[18] Xavier Boyen, Christopher Carr, and Thomas Haines. Blockchain-free cryptocurrencies: A framework for truly decentralised fast transactions. Technical report, Cryptology ePrint Archive, Report 2016/871, 2016.
[19] Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *International Conference on Financial Cryptography and Data Security*, pages 499–514. Springer, 2016.
[20] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.
[21] Jon Matonis. The bitcoin mining arms race: Ghash. io and the 51% issue, 2014.
[22] Alexei Zamyatin, Katinka Wolter, Sam Werner, Peter G Harrison, Catherine EA Mulligan, and William J Knottenbelt. Swimming with fishes and sharks: Beneath the surface of queue-based ethereum mining pools. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2017 IEEE 25th International Symposium on*, pages 99–109. IEEE, 2017.
[23] Blockchain Luxembourg S.A. Blockchain size, 2018.
[24] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *NSDI*, pages 45–59, 2016.
[25] U.S. Department of Commerce, National Institute of Standards, and Technology. *Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4*. CreateSpace Independent Publishing Platform, USA, 2012.
[26] Bitcoin Wiki. Comparison of mining pools, 2018.
[27] Hukukane Nikaidô, Kazuo Isoda, et al. Note on non-cooperative convex games. *Pacific Journal of Mathematics*, 5(Suppl. 1):807–815, 1955.
[28] Jacek B Krawczyk and Stanislav Uryasev. Relaxation algorithms to find Nash equilibria with economic applications. *Environmental Modeling & Assessment*, 5(1):63–73, 2000.
[29] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.
[30] Matlab optimization toolbox, R2015b. The MathWorks, Natick, MA, USA.