[Short Paper:] Revisiting Difficulty Control for Blockchain Systems

Dmitry Meshkov and Alexander Chepurnoy

IOHK Research

Abstract. The Bitcoin whitepaper [1] states that the security of the system is guaranteed as long as honest miners control more than half of the current total computational power. The whitepaper assumes static difficulty case when it is equally hard to solve a cryptographic proof-of-work puzzle dependless on system behavior. However, a real Bitcoin network is using an adaptive difficulty adjustment mechanism.
In this paper we introduce and analyze a new kind of attack on mining difficulty retargeting. A malicious miner is increasing his mining profits from the attack. The average time interval between blocks is increasing as a side-effect of the attack.
We propose an alternative difficulty adjustment algorithm in order to reduce the incentive of such an attack and also to improve stability of inter-block delays. Finally, we evaluate the presented approach and show that the novel approach performs better than original algorithm of Bitcoin.

1 Introduction

Blockchain systems have attracted significant amount of interest after the Bitcoin whitepaper [1] was published in 2008. Bitcoin security relies on a distributed problem solving protocol which maintains a distributed ledger. In the protocol miners are trying to find a partial hash collision in order to generate a valid block by iterating over nonce field values. That is, for a given block $\mathcal{B}$ (with a random nonce included) in order for being valied valid, the condition $hash(\mathcal{B}) < T$ should hold, where *target* parameter T specifies expected hardness of the block generation. This hardness can be also expressed via *difficulty* parameter $D = \frac{1}{T}$.

Alternative systems may rely on other types of computational puzzles rather than finding a partial hash collision, e.g., [2,3]. Nevertheless, all of them assume some algorithm that changes the difficulty of the puzzle dynamically. An algorithm for difficulty readjustment is required in order to make an open blockchain system working stable in the face of participants joining and leaving the system (resulting in constantly changing available computational power for solving the puzzles), and also to stabilize mean latency between blocks.

The difficulty readjustment algorithm in Bitcoin assumes that the total computational power involved in the mining process does not significantly change from epoch to epoch. In contrast, real networks show that significant variance in computational power happens over long periods. For example, we show in

this paper that due to continuous growth of computational power in the Bitcoin network a mean delay between blocks of about 7%, in comparison to the expected value, could be observied. Noteworthy, exponential growth of computational power, often observed in practice, is the absolutely worst case (regarding the mean block delay divergence) for the Bitcoin's difficulty readjustment algorithm [4].

In this paper we also consider a new type of miner behavior with regards to difficulty readjustment problematic for security guarantees of a blockchain system which we call a coin-hopping attackfollowing the "pool-hopping" term raised in [5]. In this attack, an adversarial miner is switching from mining one coin to another in the beginning of an epoch, then he is switching back in the beginning of next epoch when difficulty becomes lower. We show how mining profit is increasing for Bitcoin's difficulty readjustment function, and how inter-block delay suffers from the coin-hopping attack.

As a solution for the significant variance in computational power and also in order to reduce incentive of the described coin-hopping strategy, we propose an alternative difficulty readjustment procedure. The new algorithm is using integer arithmetic for all the steps only, which is important since nodes in peer-to-peer network are running on different platforms. We show that the proposed solution is better suited for exponential growth of the total mining power and it also reduces profit and negative side-effects of the coin-hopping attack.

1.1 Related Work

In this section, we provide an overview about known formal and informal studies of Bitcoin with regard to the dynamic nature of the difficulty parameters. Following the well known paper of Garay et al. [6] generalizing the Bitcoin backbone protocol in a static difficulty setting, a newer paper from the same authors [7] is providing a positive answer on whether basic security properties of the Bitcoin backbone protocol (common prefix, chain quality and chain growth) hold in case of dynamic difficulty, in a cryptographic setting with an arbitrary adversary. Nevertheless, studying concrete attacks against the real protocol is still needed.

The Timejacking attack [8] allows an attacker to first shift the network time at a victims node (which is calculating network time by averaging timestamps it gets regularly from neighbors) and then force the victim node to reject a block with a specially crafted timestamp (other nodes are accepting). The time wrapping attack [9] is exploiting the fact that Bitcoin is using difference in timestamps between last and first block in an epoch, not last block in an epoch and last block in a previous epoch. By using specially crafted timestamps for the last block of each epoch, an attacker can produce more blocks for a time window with more work contributed to his chain. The difficulty raising attack, introduced in [10], allows an attacker to discard n-depth block, for any n, and for any computational power of the attacker, with probability 1 if he is willing to wait long enough.

Another paper [4] is introducing an alternative difficulty readjustment function designed to work better than Bitcoin's not just for almost constant mining

power but also when the power is growing exponentially with time. We provide comparison with this function in the paper in one of the following sections.

The paper is organized as follows: in section 2 we provide a detailed view of Bitcoin's readjustment function. In section 3 we introduce the coin-hopping attack, followed by the definition of an improved difficulty readjustment function described in section 4. Section 5 provides results of evaluation of the presented approach.

2 Bitcoin Mining

The concept of Bitcoin mining was introduced in the Section 4 of the Bitcoin whitepaper [1] and discussed in detail later in the papers [4],[7]. In Bitcoin a miner generates a block by iterating over a *nonce* value and calculating the hash of a block with the nonce value included. For a block $\mathcal{B}$ to be valid, a value of a hash function has to be less than the current *target* T, $hash(\mathcal{B}) < T$, where $hash$ is an ideal cryptographic hash function. Hardness to find a block could be expressed also via *difficulty* D as $D = \frac{1}{T}$. If output of the $hash$ function is μ bits long then the probability to generate a block by doing q requests to the hash function is $\frac{T \cdot q}{2^\mu} = \frac{q}{D \cdot 2^\mu}$. We define miners *hashrate* R as $R = \frac{q_s}{2^\mu}$, where q_s is number of queries done per time unit. The probability to generate a block within a time unit is then $\frac{R}{D}$.

Every M blocks ($M = 2016$ for Bitcoin) difficulty is recalculated as

$$D_{i+1} = D_i \cdot \frac{M \cdot |\Delta|}{S_m} \tag{1}$$

where $|\Delta|$ is the expected time interval between blocks and S_m is the actual time spent to generate M blocks. For the Bitcoin network, the observed time interval of $\approx$*9 minutes 20 seconds* is less than the planned value of $|\Delta| = 10$ *minutes* due to continuous growth of the network computational power. Difficulty recalculation interval $M = 2016$ has been chosen to recalculate difficulty every 2 weeks on average. This time interval is big enough to usually see a change of the computational power of the network, resulting in a mean delay is close to the planned 10 minutes right after target recalculation, whereas at the end of an epoch it is less than 9 minutes in average (see Figure 1).

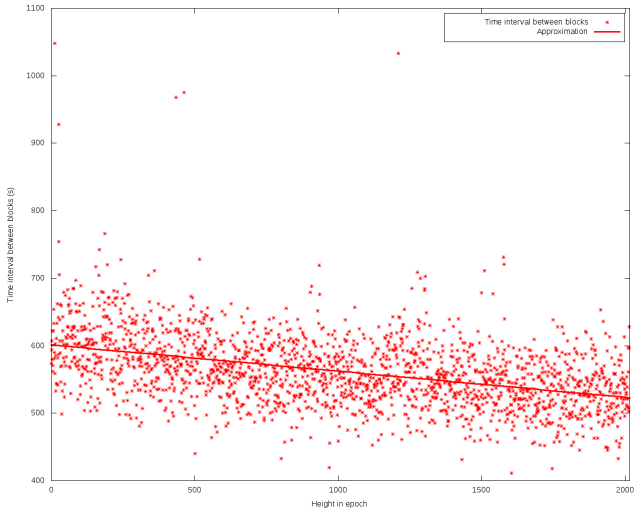

Fig. 1. Average block time between difficulty recalculation

TODO the picture above is a mess

The next section describes an attack that allows to gain benefits by attacking the recalculation algorithm.

3 Coin-hopping Attack

We consider the following attack involving an adversarial miner $\mathcal{A}$:

- There are $N > 1$ possible coins $\mathcal{A}$ can contribute to. Without a loss of generality, we assume that each of them provides about the same profitability of the mining activity.
- $\mathcal{A}$ is mining one of the coins before a beginning of an epoch, say, epoch A. At the beginning of A he switches to mine another coin.
- Without thge contribution of miner $\mathcal{A}$ the total mining power of the network for the epoch decreases.
- For an epoch B right after epoch A, the difficulty gets readjusted to a lower value. So $\mathcal{A}$ starts mining again with a lower difficulty.

We call this strategy a *coin-hopping attack*.

To calculate the profit the adversarial miner gains from this attack, we use Bitcoins' difficulty recalculation function and assume a constant network hashrate (with respect to the rest of the network, without our adversarial miner). We denote the hashrate of miners not participating in the coin-hopping attack as R_0, and we denote the hashrate of the adversarial miner as $R_a = R_0 \cdot p, 0 < p < 1$. Before epoch A the difficulty is $D_0 = (R_0 + R_a) \cdot |\Delta|$, and the adversary mines $\frac{M \cdot R_a}{R_0 + R_a}$ blocks per epoch (of M blocks) in average investing $M \cdot R_a \cdot |\Delta|$

computational power. During the epoch B the adversary mines with difficulty $D_1 = R_0 \cdot |\Delta|$ calculated from honest miners hashrate R_0 only. The adversary mines $\frac{M \cdot R_a}{2(R_0+R_a)}$ blocks per epoch in average investing $\frac{M \cdot T \cdot R_a \cdot R_0}{2 \cdot (R_0+R_a)}$ computational power. Consequently honest miners will spend $(R_0+R_a)|\Delta|$ computational power per block, whereas attacker will just spend $R_0 \cdot |\Delta|$ computational power per block. Considering mining to be at least break-even, the cost of computational resources invested into mining should be around the expected reward. If $\mathcal{W}$ is block reward and $\mathcal{C}$ is hash calculation cost, honest miners profit is

$$(\frac{M \cdot R_a}{R_0 + R_a}) \cdot \mathcal{W} - M \cdot R_a \cdot T \cdot \mathcal{C} = M \cdot R_a \cdot (\frac{\mathcal{W}}{R_0 + R_a} - T \cdot \mathcal{C}) \tag{2}$$

per epoch. At the same time adversarial profit is

$$\frac{M \cdot R_a}{2 \cdot (R_0 + R_a)} \cdot (\mathcal{W} - T \cdot R_0 \cdot \mathcal{C}) \tag{3}$$

Additional profit of the adversary is: [TODO: ???]

$$\frac{\mathcal{W}}{\mathcal{C}} < T \cdot (R_0 + 2 \cdot R_a). \tag{4}$$

Remarkably, under such an attack the mean time between blocks will be

$$T_a = \frac{T}{2}(\frac{R_0 + R_a}{R_0} + \frac{R_0}{R_0 + R_a}) = T(1 + \frac{p^2}{2(1+p)}) \tag{5}$$

which is bigger than the planned time T.

The next section provides a new algorithm for difficulaty adjustment.

4 Improved Difficulty Adjustment

The difficulty adjustment algorithm employed by Bitcoin works as designed: if the hash rate of the network is constant, it yields to the desired block rate. However it does not achieve the desired block rate in other situations and is vulnerable to the attack, described in 3. In this section we are going to propose an alternative difficulty adjustment algorithm.

First, we state properties of an ideal difficulty update algorithm:

1. It should be resistant to known types of attacks based on difficulty manipulation.
2. It should lead to an almost constant desired block rate for random fluctuations in the hash rate.
3. It should be simple enough to use integer arithmetic for all computational steps.

Security is the most important feature of blockchain systems and should be regarded with highest priority. Incorrect block rate is not considered a big

problem in the Bitcoin community but it may be important for more advanced applications of blockchain systems. Implementation of the *ideal* difficulty update algorithm based solely on integer arithmetics is desired for a platform independent approach. This rule is not required necessarily, since as mentioned in [4], it is possible to include non-integer algorithm parameters as part of the block, but it provides another way of difficulty manipulating to an attacker.

In this section we are providing a difficulty adjustment algorithm based on the well-known linear least squares method[11]. In the simpliest case of pair linear regression ($y = kx + b$) coefficients may be calculated as follows:

$$\begin{cases} k = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2} \\ b = \overline{y} - k\overline{x} \end{cases} \tag{6}$$

Note, that for accurate difficulty prediction we should use a few last observed difficulties, rather than just one, as implemented in Bitcoin, but it is still possible to use this algorithm right after second epoch.

The next section provides an evaulation of the presented approach.

5 Evaluation

We now present simulation results that show that the method proposed in Section 4 outperforms Bitcoin's difficulty update algorithm. We will regard difficulty growth in this section, keeping in mind the fact, that it is closely related with network hash rate, which is usually considered in literature.

TODO Linear?

5.1 Exponential Difficulty

First, we observe exponential difficulty growth as it occurres in practice in the Bitcoin network. As we already mentioned, exponential difficulty growth is the absolutely worst case possible for Bitcoins difficulty retargeting algorithm. For simplicity we regard a situation, when hash rate increases by 10% each epoch, more complicated research of exponential difficulty growth can be found in [4]. Figure 2 presents difficulty function over epochs, which is 2016 blocks in Bitcoin.

Note that difficulty calculated from Bitcoin algorithm is always significantly lower than the real one. This leads to *9 min 5 sec* time interval between blocks, which is ≈10% lower then desired *10 min* interval. Difficulty, calculated by the linear algorithm is constantly lower, then the real one, but still much closer to it. Mean time interval between blocks is *9 min 45 sec* (TODO: use the same numbers for this throughout the article), which is much closer to the desired one.

While the difficulty update algorithm, proposed in [4] leads to much better results for exponential difficulty growth with a constant rate, we should note, that our algorithm is much simpler and may be implemented with integer arithmetic only. Moreover, exponential difficulty growth is the simplification of the difficulty growth law, and it may be incorrect to expect it in some situations.

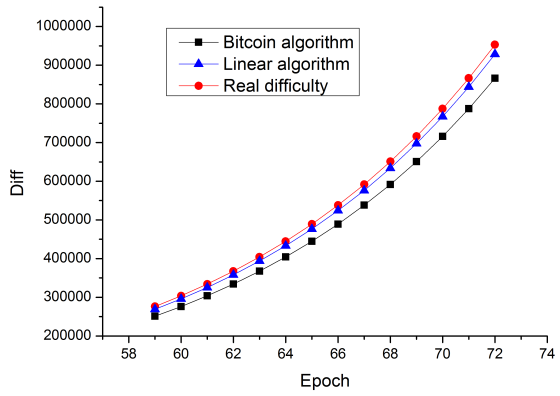

Fig. 2. Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of exponential hash rate growth

5.2 Coin-hopping Attack

We consider a situation as described in the Section 3: an attacker with computational power R_a (for simplicity we suppose $R_a = 0.2 \cdot R_a$ in this section) turn on and turn off his mining to manipulate difficulty and minimize computational power, expended for block mining. Figure 2 (TODO: shouldn't this be figure 3?) represents difficulty as the function over epochs for this situation.

Note that the difficulty calculated with the Bitcoin algorithm is always in antiphase with the real one and the attacker spends his computational power only when difficulty is low. The Bitcoin difficulty update algorithm leads to *10 min 10 sec* mean delay between blocks, which is in good correlation with 5. The linear algorithm also leads to enlarged time interval between blocks equal to *10 min 5 sec*, resulting in a two times lower deviation from the desired time. Obviously, the profit of the attacker is then also 2 times lower while using the the linear difficulty update algorithm, providing a servious improvement.

Thus, linear difficulty control algorithm, proposed in Section 4 is better than the one used in Bitcoin for the given situation, both in terms of block rate and in terms of attacker profit.

References

1. S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System (2008). `arXiv:43543534534v343453`, `doi:10.1007/s10838-008-9062-0`.
   URL `http://s.kwma.kr/pdf/Bitcoin/bitcoin.pdf`
2. A. Miller, A. Juels, E. Shi, B. Parno, J. Katz, Permacoin: Repurposing bitcoin work for data preservation, in: Security and Privacy (SP), 2014 IEEE Symposium on, IEEE, 2014, pp. 475–490.

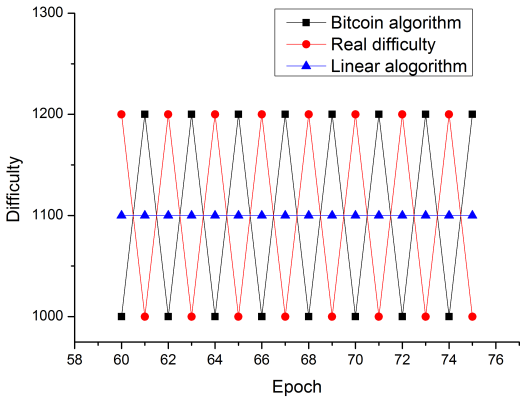

Fig. 3. Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of attack, described in section **??**sec:attack)

3. A. Biryukov, D. Khovratovich, Equihash: Asymmetric proof-of-work based on the generalized birthday problem, Ledger 2.
4. D. Kraft, Difficulty control for blockchain-based consensus systems, Peer-to-Peer Networking and Applications (2015) 1–17.
5. M. Rosenfeld, Analysis of bitcoin pooled mining reward systems, arXiv preprint arXiv:1112.4980.
6. J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Advances in Cryptology-EUROCRYPT 2015, Springer, 2015, pp. 281–310.
7. J. A. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol with chains of variable difficulty.
8. The timejacking attack.
   URL `http://culubas.blogspot.com`
9. ArtForz, The time wrapping attack.
   URL `https://bitcointalk.org/index.php?topic=43692.msg521772#msg521772`
10. L. Bahack, Theoretical bitcoin attacks with less than half of the computational power, arXiv preprint arXiv:1312.7013.
11. C. L. Lawson, R. J. Hanson, Solving least squares problems, Vol. 161, SIAM, 1974.