[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. An 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 an incentive to attack and also to improve stability of inter-block delays. We 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 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 block $\mathcal{B}$ (with a random nonce included) to be valid, condition $hash(\mathcal{B}) < T$ should hold, where *target* parameter T specifies expected hardness of a valid block generation. This hardness can be also viewed via *difficulty* parameter $D = \frac{1}{T}$.

Alternative systems may rely on different than finding a partial hash collision types of computational puzzles [2,3]. Nevertheless, all of them assume some algorithm that changes puzzle difficulty dynamically. The 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, and also to stabilize mean latency between blocks.

A difficulty readjustment algorithm in Bitcoin assumes that total computational power involved in mining process does not significantly change from epoch to epoch. In contrast, real networks show that significant variance in computational power could happen for long periods. For example, we show in this paper that due to continuous growth of computational power in Bitcoin network a mean delay between blocks by 7% less than expected value. Noteworthy,

exponential growth of computational power, often observed in practice, is the absolutely worst case (regarding the mean block delay divergence) for the Bitcoins difficulty readjustment algorithm [4].

In this paper we also consider a new type of miner behavior in regards with 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 suffer from the coin-hopping attack.

As a solution for significant variance in computational power and also in order to reduce incentive to follow 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 total mining power and also reduces profit and negative side-effects of the coin-hopping attack.

1.1 Related Work

Here we observe known formal and informal studies of Bitcoin in regards with dynamic nature of difficulty parameter. Following the well known paper of Garay et al. [6] generalizing Bitcoin backbone protocol in static difficulty setting, a newer paper from the same authors [7] is getting 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.

Timejacking attack [8] allows an attacker first shift network time at a victim node (which is calculating network time by averaging timestamps got 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 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 enough time.

A 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.

1.2 Structure of the Paper

Several parts are organizing the paper. In Section 2 we provide detailed view of Bitcoin readjustment function. In Section 3 we introduce the coin-hopping attack. In Section 4 an improved difficulty readjustment function is proposed and analyzed. Section 5 provides results of simulations.

2 Bitcoin Mining

The concept of Bitcoin mining was introduced in the Section 4 of the Bitcoin whitepaper [1] and discussed then in the papers [4],[7]. In Bitcoin a miner generates a block by iterating over *nonce* value and calculating hash of a block with the nonce value included. For a block $\mathcal{B}$ to be valid, the hash of its header must be less than 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 $hash$ 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 miner *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 time unit then is $\frac{R}{D}$.

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

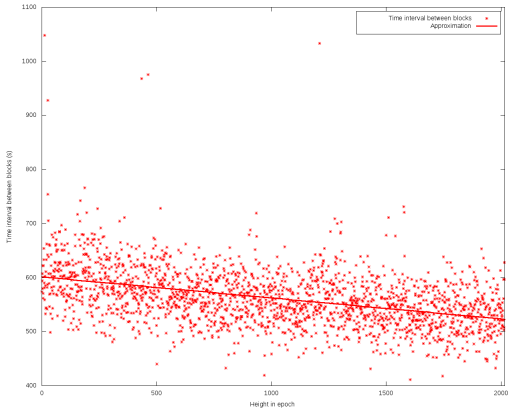

Fig. 1. Average block time between difficulty recalculation

TODO the picture above is a mess TODO transition to next section

3 Coin-hopping Attack

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

- There are $N > 1$ possible coins $\mathcal{A}$ can contribute to. Each of them is about the same mining profitability.
- $\mathcal{A}$ is mining one of the coins before a beginning of an epoch, say, epoch A. At this moment he switches to mine other coins.
- Without a contribution of $\mathcal{A}$ mining power for the epoch goes down.
- For an epoch B next to the epoch A where $\mathcal{A}$ left difficulty re-adjusted to a lower value. So $\mathcal{A}$ starts mining again with a lower difficulty.

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

To calculate adversarial profit from the attack we assume Bitcoin difficulty recalculation function and constant hash rate.

Consider the network has hash rate R_0 provided by honest miners and the adversary has additional hash rate $R_a = R_0 \cdot p$ to turn it on or off. Before epoch A the adversary mines all the time with difficulty $D_0 = (R_0 + R_a) \cdot T$ and will mine $\frac{M \cdot R_a}{R_0 + R_a}$ blocks per epoch (M blocks) in average spending $M \cdot R_a \cdot T$ computational power for them. During the epoch B the adversary mines with difficulty $D_1 = R_0 \cdot T$ is calculated from honest miners R_0 only. He will mine $\frac{M \cdot R_a}{2(R_0 + R_a)}$ blocks per epoch in average spending $\frac{M \cdot T \cdot R_a \cdot R_0}{2(R_0 + R_a)}$ computational power for them. Consequently honest miner will spend $(R_0 + R_a)T$ computational power per block, whereas attacker will just spend $R_0 T$ computational power per block. The cost of computational resources invested into mining should be around the

expected reward. If B is block reward and C is hash calculation cost, the honest miner profit is

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

per epoch. At the same time adversarial profit is

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

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

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

Remarkable, that under such an attack 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 then desired time T. Please notice that we regard p as the ratio between the hash-powers of the attacker and the honest network so it changes from 0 to 1.

TODO transition to next section

4 Improved Difficulty Adjustment

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

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 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 the 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 in subclass of integer programming is desired for different platforms compatibility. This rule is not required, because, 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 going to regard difficulty adjustment algorithm based on 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 few last observed difficulties, rather than just one, as implemented in Bitcoin, but it's possible to use this algorithm right after second epoch.

We regard it as the good candidate for difficulty update algorithm, because:

1. It should reduce profit of the attack, described in Section 3. Calculations of the attacker profit are described in Section 5
2. It leads to desired block rate for linear changes in the hash rate. This means, that regarding block rate, linear algorithm is better, then Bitcoin's one, in all cases, except constant hash rate, when they lead to the same result.
3. It is simple enough to use integer arithmetic for all computational steps with high fidelity.

5 Simulations

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

TODO Linear?

5.1 Exponential Difficulty

First, we observe exponential difficulty growth occurred in practice in 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 growth 10% each epoch, more complicated research of exponential difficulty growth can be found in [4]. Figure 2 presents difficulty as the function of epoch, 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 from Linear algorithm is also always lower, then the real one, while it's much closer to it. Mean time interval between blocks is *9 min 45 sec*, which is much closer to the desired one.

While 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 simplier 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.

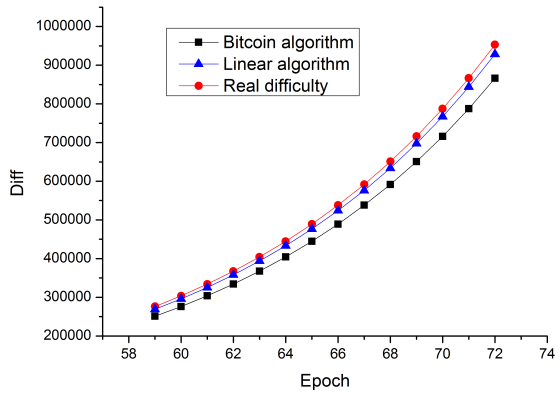

Fig. 2. Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of exponential hash rate growth

5.2 Coin-hopping Attack

We consider a situation 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 represents difficulty as the function of epcoh 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. Bitcoin difficulty update algorithm leads to *10 min 10 sec* mean delay between blocks, which is in good correlation with 5. Linear algorithm also leads to enlarged time interval between blocks *10 min 5 sec*, but it's deviation from desired time is 2 times lower. Obviously, attacker profit is also 2 times lower in situation with linear difficulty update algorithm, which may be regarded as a good result.

Thus, linear difficulty control algorithm, proposed in Section 4 is better than the Bitcoin's in all situations 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.

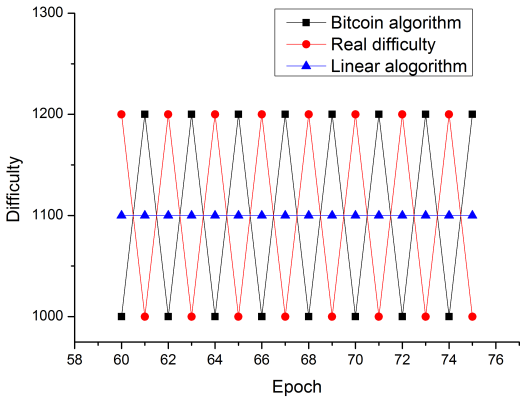

Fig. 3. Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of attack, described in section **??**sec:attack)

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.