Revisiting Difficulty Control for Blockchain Systems

Dmitry Meshkov[b], Alexander Chepurnoy[b]

[a] *IOHK Research*
[b] *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.

In this paper we introduce and analyze a new kind of attack based on mining difficulty manipulations. A malicious miner is increasing his mining profits from the attack. The average time interval between blocks increases as a side-effect.

We propose an alternative difficulty adjustment algorithm in order to reduce the incentive of such an attack while at the same time improve stability of inter-block delays. Finally, we evaluate the presented approach and show that it performs much better than the original algorithm of Bitcoin.

Keywords: Blockchain, Bitcoin, Decentralized consensus, Peer-to-peer networks, Proof-of-Work

1. Introduction

Blockchain systems have attracted a significant amount of interest from various communities after the Bitcoin whitepaper [1] has been published in 2008. Bitcoin security relies on a distributed problem solving protocol, also referred to as mining, that maintains the distributed ledger. In the protocol miners solve moderately hard computational puzzles in order to generate a block []. In Bitcoin solving a puzzle is about finding a partial hash function collision []. Alternative systems may rely on other types of computational puzzles [**?**].

Email address: `dmitry.meshkov@iohk.io` (Dmitry Meshkov)

Nevertheless, all of them use some algorithm that changes a puzzle *difficulty* dynamically.

These algorithms for difficulty retargeting are required in order to make blockchain systems predictable and stabilize mean latency between blocks. The latter is important for several reasons:

- With too frequent blocks it is possible for a lot of miners to have block propagation time bigger than latency between blocks. This decreases security of a blockchain systems [2, 3].
- High latency leads to decreasing network throughput [4] and may be critical for high-loaded blockchain systems like Bitcoin, where blocks are already full today [5]. Increasing latency in the Bitcoin network will lead to some transactions that will never be included into blockchain. Moreover, this will lead to potentially infinite growth of the unconfirmed transactions pool, preventing relaying for most of Bitcoin transactions.

Most of blockchain systems relying on difficulty retargeting algorithms assume total computational power involved in the mining process does not significantly change from epoch to epoch. Using more complicated retargeting algorithms with incorrect assumptions [6] may lead to incorrect time intervals between blocks even for the simple case of constant a hash rate. For example, it is observed that mean time between blocks in Nxt is 2 times bigger than stated in the whitepaper [7]. Moreover, adjusting difficulty to often leads to broader distribution of time intervals between blocks and makes blockchain system unpredictable [6]. Varying network computational power makes this algorithms inefficient for difficulty recalculation, e.g. continuous growth of computational power leads to decreasing mean latency between blocks. In average, the block time in the Bitcoin network is 1.07 times lower, then expected. It is worth noticing that exponential growth of computational power, which is the situation observed in practice in accordance with Moores law [8], is the absolutely worst case (regarding the maximal block rate) possible for Bitcoins difficulty retargeting algorithm [9]. On the other side, target recalculation algorithms should be simple enough to use only integer arithmetic for all computational steps, since all nodes in the peer-to-peer network have to agree absolutely on the calculated difficulty.

The original Bitcoin white-paper, states that the security of the system is guaranteed as long as there is no attacker in possession of half or more of the total computational power used to maintain the system [1]. Most of the models used in the literature to discuss double-spending attacks assume that mining difficulty is constant [**?**]. However, this is usually not the case, the difficulty is not constant and it can be manipulated by an attacker. The Difficulty Raising Attack, introduced in [10], enables the attacker to discard n-depth block, for any n and any attacker hash power, with probability 1 if he is willing to spend enough time on the attack. The fact that there is no way to determine whether a block have been computed on its declared time or not, has been used as part of other attacks [11, 12]. In section 2 we introduce a new attack for blockchain systems which manipulates difficulty for decreasing effective hash rate, required for block generation.

Novel studies of difficulty control proposes better functions for difficulty recalculation. For example, the paper [9] introduces a target recalculation function, designed to work perfectly not just for constant hash rate but also if the hash rate grows exponentially (with a constant but unknown rate). Allthough this seems to meet the situation observed in practice in the Bitcoin network, there are still a lot of open questions for future research. Is it possible to create a function, suitable for random fluctuations in the hash rate? Is it possible to create a function, simple enough to be based on integer arithmetic for all computational steps? Is it possible to create a function, that is stable against attackers manipulations?

The following sections of the paper are organized as follows: section two provides an overview of the mining function in the Bitcoin network, followed by a description of a possible attack againstm Bitcoins' target recalculation algorithm. Afterwards, section four describes a new algorithm for the difficulty recalculation and section five provides an evaluation of the described algorithm. Finally, section six provides our conclusions.

2. Bitcoin Mining

The concept of Bitcoin mining was introduced in section 4 of the Bitcoin whitepaper [1] and discussed then in the papers as [9, 13, 14, 3]. The *chain quality* property for a blockchain system was introduced in [3]. This property states that if a malicious attacker holds (?) of total mining power then it could generate no more than (?) of total blocks in the long run, assuming a static difficulty. In this paper we study the advantage of a malicious by influencing the difficulty parameter.

In Bitcoin a miner generates a block by iterating through *nonce* and calculating SHA-256 hash of a blocks header with the nonce value included. For a block to be valid the hash of its header must be less than current *target* T: $hash(blockheader) < T$. Hence, the *difficulty* is expressed as $p = \frac{1}{T}$. If the output of the *hash* function is μ bits long, the probability to generate a block by doing q requests is $\frac{T \cdot q}{2^\mu} = \frac{q}{D \cdot 2^\mu}$. We choose a second as a time unit, and 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 one time unit then is $\frac{R}{D}$

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

$$D_{i+1} = D_i \cdot \frac{MT}{S_m} \tag{1}$$

where T is the expected time interval between blocks and S_m is actual time spent to generate M blocks. For Bitcoin, the observed time interval is equal to ≈*9 min 20 sec* is less than defined *T=10 min* because of the continuous growth of the computational power of the network. The difficulty recalculation interval $M = 2016$ has been chosen in such a way to recalculate difficulty about every 2 weeks. This time interval is big enough to see an increase in the computational power of the network: directly after target recalculation block time is close to the defined 10 minutes, whereas at the end of *epoch* it tends to be less than 9 minutes (see figure 1).

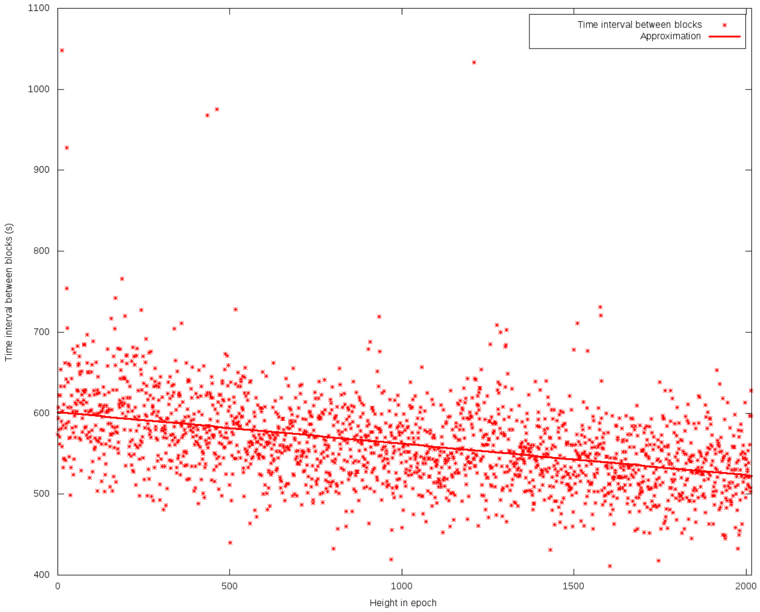

Figure 1: Average block time between difficulty recalculation

TODO the picture above is a mess The next section describes an attack that allows to gain benefits by attacking the recalculation algorithm.

3. Switching 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 $\mathcal{A}$s' 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 *switching attack*.

To calculate the profitan adversary miner gains from the attack we use the Bitcoin difficulty recalculation function and a constant hash rate.

Consider the network has a hash rate of R_0 provided by honest miners and the adversary miner provide an additional hash rate $R_a = R_0 \cdot p$ to turn it on or off. Before epoch A the malicious miner mines all the time at 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 computation power equal to $M \cdot R_a \cdot T$. During the epoch B the attacker mines at difficulty $D_1 = R_0 \cdot T$ calculated based honest miners R_0 solely. He will mine $\frac{M \cdot R_a}{2(R_0 + R_a)}$ blocks per epoch in average investing computational power equal to $\frac{M \cdot T \cdot R_a \cdot R_0}{2(R_0 + R_a)}$. At the same time, honest miners will spend $(R_0 + R_a)T$ computational power per block, while the attacker just spends $R_0 T$ computational power per block. Assuming that mining should at least be break-even, the cost of the computational resources invested into mining should be around the expected reward. If B is the block reward and C describes the cost for hash calculation, 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: ???]

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

Remarkably, 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 p describes the ratio between the hash-powers of the attacker and the honest network, so it is in the range from 0 to 1 (assuming that the attacker does not hold more ten half of the computational power of the network).

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 is constant, it yields the desired block rate. However it does not achieve the desired block rate exactly in other situations and it is vulnerable to attack, as shown 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 a constant block rate, even 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 in subclass of integer programming is desired for different platforms compatibility. This rule is not required, because, as mentioned in [9], 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[15]. In the most simple case of pair linear regression ($y = kx + b$) coefficients may be calculated as follows:

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

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 assume it is a good candidate for the 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 the 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 outperforms Bitcoins difficulty update algorithm. We will evaluate *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.

All calculations have been done based on open-source programs, available at the Scorex project GitHub page [16].

TODO Linear?

5.1. Exponential Difficulty

First, we observe exponential difficulty growth as it occurrs 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 [9]. Figure 2 presents the 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 by the

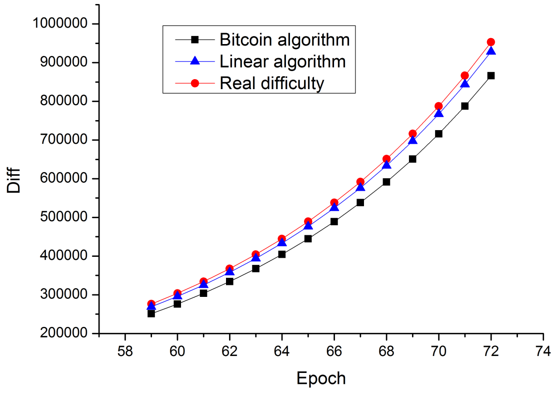

Figure 2: Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of exponential hash rate growth

Linear algorithm is also always lower, then the real one, but still much closer to it. Mean time interval between blocks is *9 min 45 sec*, which is much closer to the desired one.

While the difficulty update algorithm, proposed in [9] 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.

5.2. Switching 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, invested 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. The Bitcoin difficulty update algorithm leads to

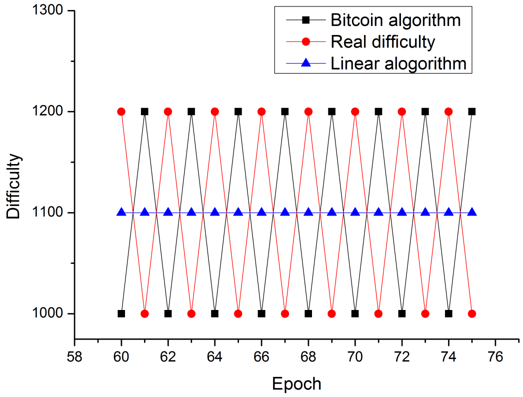

Figure 3: Real difficulty (red) and difficulties calculated from bitcoin (black) and linear (blue) algorithms in situation of attack, described in section **??**sec:attack)

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 also 2 times lower while using the linear difficulty update algorithm, which a serious improvement.

Thus, linear difficulty control algorithm, proposed in Section 4, is better than the algorithm used in Bitcoin in all situations both in terms of block rate and in terms of the attackers' profit.

6. Conclusion

In this paper we analyze a new kind of attacks on the blockchain based on manipulating mining difficulty. This attack decreases computational power spent by an attacker for block mining while increasing mean time interval between blocks. It is especially favorable in situation, when the cost of computational resources invested into mining is around the expected reward and there are enough forks to switch mining between them.

To improve the stability of block times and decrease the attacker profit, we proposed an alternative difficulty update algorithm, based on linear regression. It was found that this algorithm is better then the one used for Bitcoin, both in terms of block rate and in terms of attacker profit, while it's still simple enough to be computed with integer arithmetic only.

Acknowledgments

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] C. Decker, R. Wattenhofer, Information propagation in the bitcoin network, in: Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on, IEEE, 2013, pp. 1–10.

[3] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Advances in Cryptology-EUROCRYPT 2015, Springer, 2015, pp. 281–310.

[4] R. Böhme, T. Okamoto, On scaling decentralized blockchains, 2016. URL http://fc16.ifca.ai/bitcoin/papers/CDE+16.pdf

[5] B. Armstrong, What happened at the satoshi roundtable (2016). URL https://medium.com/@barmstrong/what-happened-at-the-satoshi-roundtable-6c11a10d8cdf#.pvoqt832u

[6] andruiman, Nxt forging algorithm: simulating approach. URL https://www.scribd.com/doc/243341106/Nxt-forging-algorithm-simulating-approach

[7] andruiman, Nxt forging algorithm: simulating approach. URL http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt

[8] G. E. Moore, Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff., IEEE Solid-State Circuits Newsletter 3 (20) (2006) 33–35.

[9] D. Kraft, Difficulty control for blockchain-based consensus systems, Peer-to-Peer Networking and Applications (2015) 1–17.

[10] L. Bahack, Theoretical bitcoin attacks with less than half of the computational power, arXiv preprint arXiv:1312.7013.

[11] The timejacking attack.
URL `http://culubas.blogspot.com`

[12] ArtForz, The time wrapping attack.
URL `https://bitcointalk.org/index.php?topic=43692.msg521772#msg521772`

[13] 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.

[14] I. Eyal, E. G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, in: Financial Cryptography and Data Security, Springer, 2014, pp. 436–454.

[15] C. L. Lawson, R. J. Hanson, Solving least squares problems, Vol. 161, SIAM, 1974.

[16] Scorex blockchain framework.
URL `https://github.com/ScorexProject`