On the trade-offs of Proof-of-Work algorithms in blockchains

Zi Hau Chin, Timothy Tzen Vun Yap, Ian K. T. Tan

Multimedia university, Persiaran Multimedia, 63100 Cyberjaya, Malaysia
zihau27@gmail.com, timothy@mmu.edu.my, ian@mmu.edu.my

Abstract. There are many different protocols that regulate the way in which each node on a blockchain network is able to reach consensus that a newly created block is valid. One of the protocols, Proof-of-Work (PoW) gained popularity when it was implemented in a blockchain-based cryptocurrency known as Bitcoin. However, there are inherent deficiencies in its current implementation. This paper discusses these deficiencies, as well as the parameters that directly and indirectly affect its efficacy and performance so that possible enhancements to the protocol can be investigated.

Keywords: Proof-of-Work, Blockchain, Bitcoin, Difficulty

1 Introduction

A blockchain is based on a distributed, decentralized and public digital ledger technology that is used to record data over numerous computers. By storing data across its peer-to-peer network, the data is not owned or controlled by a single entity, hence it is decentralized and distributed. It is a growing list of records known as "blocks" that are linked together using cryptography. Each block is linked back, referring to the previous block or parent block by including the cryptographic hash of the previous block in the chain. The linked blocks then form a chain, thus any included record cannot be altered without the modification of every following block, which will require consensus of the network majority. Blockchain consensus protocols are keeping all the nodes on a network synchronized, in which the nodes agree on a same state of the blockchain.

2 Blockchain

The idea behind blockchain was originally described by Haber and Stornetta as "a cryptographically secured chain of blocks" [1]. Bayer, Haber and Stornetta then implemented Merkle trees into the previous design in order to improve its efficiency and reliability by allowing multiple documents to be collected into one block [2]. However, the blockchain technology as we know it today was conceptualized by a pseudonymous Satoshi Nakamoto in the Bitcoin white paper during 2008 [3].

R. Alfred et al. (eds.), *Computational Science and Technology*,
Lecture Notes in Electrical Engineering 603,
https://doi.org/10.1007/978-981-15-0058-9_55

The most interesting features of blockchain are immutability and decentralization. Data stored in blockchain is secured using cryptography. Particularly in Bitcoin, a Secure Hash Algorithm 256 (SHA256) cryptographic hash function is applied twice to the block header of each block which contains data such as the cryptographic hash of the previous block, transaction data, timestamp, etc. The output is a 32-byte hash known as the "block hash" or "block header hash" which acts as the primary identifier of a block due to its ability to identify a block unambiguously and uniquely. Additionally, a cryptographic hash function is designed with a one-way function in mind, which means that it is unfeasible to invert, thus contributing to the blockchain's immutability.

In Bitcoin, a distribution computation system known as "Proof-of-Work (PoW)" algorithm is used in order to achieve the state of being completely decentralized. Unlike centralized system or network that require all data to pass through central server, Bitcoin passes all data through a distributed, peer-to-peer (P2P) network, where all nodes are equal and there are no nodes with special rights. Every 10 minutes, a vote will be conducted in order for the nodes to reach consensus about the state of the Bitcoin network.

Dywork and Noar presented the concept of PoW in 1993 to combat junk mail and control access to a shared resource [4] but the term PoW first originated from a 1999 paper by Jakobsson and Juels [5] . The concept is that in order to acquire access to a shared resource, the user is required to compute a moderately hard but feasible function, thus preventing ill-conceived usage of the shared resource. Bitcoin's PoW is actually based on Back's version of HashCash's PoW [6].

3 Literature Review

3.1 Pricing via Processing for Combatting Junk Mail

The main idea introduced by Dywork and Naor is that a computation of a moderately hard but feasible function is required by the user in order to acquire access to a shared resource to prevent frivolous use of the shared resources [4]. Previously, legislation and usage fees have been used to restrain the access to a resource. However, the authors wished to avoid a system where sending an email between friends and families would cost as much as sending a postal mail. Thus, such approaches were deemed to be underutilizing the electronic medium and is unsuitable to act as a discouragement for sending junk email. So instead of charging the "cost of using the medium plus the profit to the provider", they wished to impose a type of cost on the transmission where it will discourage the act of sending junk email while not interfering with the normal usage of the system.

A function known as the pricing function that requires the sender of an email to compute a somewhat expensive but feasible function of the message plus some other additional information was proposed. Additionally, a shortcut would be used by the resource manager to allow cheap access to the resource that will bypass the control mechanism. The idea of a shortcut is the same as the trapdoor one-way permutation proposed by Diffie and Hellman [7], where in this case, a shortcut would allow a pricing function to be computed easily. Using email as an example, the shortcut would allow the email service provider to "permit bulk mailings at a price chosen by the service provider", thus lowering the cost of sending emails as compared to computing the pricing function for each recipient. The system was assumed to be in an environment where computers were connected to a communication network and it did not require human participation. The system would be consisted of a pricing function *fs*, a shortcut *c* and a hash function *h*. The pricing function and shortcut would be determined by an "authority" and all users must obey the authority. A hash function was used so that only the hash value of a pricing function would be applied to a message instead of pricing function which may be too long. The hash function *h* and pricing function *fs* were known to all users, but the shortcut *c* would be known only by the authority and any number of trusted agents chosen by the authority.

In order to send a message *m*, at time *t*, to destination *d*, the sender was required to compute the following pricing function:

$$y = f_s(h(< m, t, d >)) \tag{1}$$

Then proceed to send <*y*, *m*, *t*> to destination *d*. The email client of the recipient will verify that $y = f_s(h(< m, t, d >))$. The message would be discarded when the verification fails or time *t* is notably different from the current time, and the sender might or might not be notified that the transmission failed. On the other hand, the message would be sent to the recipient successfully if the verification succeeded and the time *t* was not significantly different from the current time.

3.2 Bitcoin: A Peer-to-Peer Electronic Cash System

Nakamoto published the Bitcoin whitepaper in 2008 [3] but the actual Bitcoin client was released in 2009. The vision of Nakamoto is to create a "purely peer-to-peer digital cash that would allow one party to send the online payment to another party without going through a financial institution". The idea is to define a digital signature chain as an electronic coin. A coin owner will transfer the coin to the new owner by signing and adding a "hash of the previous transaction and the public key of the new owner" to the end of coin. Verification of the ownership of the chain can be done by a payee by verifying the signatures. To ensure that the owners of the coin did not double-spend without a trusted central authority, transactions need to be announced publicly. Additionally, the participants must follow the same timeline in order for everyone to agree on a single order history in which transactions are received.

A PoW system similar to Hashcash [6] was utilized to implement the distributed peer-to-peer timestamp server in Bitcoin. Roughly speaking, participants or also known as miners are expending their computational power to find the solution to a PoW puzzle, this process is called mining. The goal of the PoW puzzle is to find a value or nonce by incrementing it, that when hashed with a cryptographic hash function such as SHA-256 will output a block hash with the required number of leading zero bits. Every time a solution is found, a new block will be generated and attached to the existing blockchain.

3.3 Revisiting Difficulty Control for Blockchain Systems

In Bitcoin, although the expected time taken to mine a block or also known as block time is set to 10 minutes, the block time is actually determined by the *difficulty* (D) of the PoW. As time goes by, the efficiency of hardware and number of nodes will cause the hash rate (computational power) to increase, thus the time taken to mine a block will decrease. If the current block time is less than 10 minutes, the difficulty will be increased (harder to mine a block) in order to maintain the 10 minutes block time. The difficulty will be adjusted once every 2016 blocks or roughly 2 weeks (2016 * 10 minutes = 20160 minutes = 2 weeks). But there is an adjustment limit of how much the difficulty can increase or decrease per difficulty readjustment. The new difficulty can only increase up to 4 times of the current difficulty, and can only decrease by 1/4 times of the current difficulty. The adjustment limit exists in order to prevent drastic changes to the difficulty of the whole Bitcoin network. The *target* (T) is a value that determines the number of leading zeros required from the computed SHA-256 block hash. The block hash must be smaller than or equal to the current target in order for the newly generated block to be valid and accepted by the network.

There is a relationship between the difficulty and target, where if the difficulty increases (harder to mine), the target will decrease (as the target becomes smaller, it will be harder to find a block hash that is smaller than or equal to it) and vice versa. The equation to calculate the new target T in Bitcoin can be summarized as:
For every N blocks ($N = 2016$)

$$T_{i+1} = T_i * \frac{X_N}{N * B} \tag{2}$$

X_N is the actual time taken to mine N number of blocks while B is the expected block time (10 minutes in Bitcoin). The formula to calculate the new difficulty D can be defined as:

$$D_{i+1} = \frac{1}{T_{i+1}} \tag{3}$$

However, the difficulty readjustment algorithm can be exploited in order to maximize a miner's profit. The *coin-hopping attack* can be summarized as following:

- We assume that miner X can mine at least 2 possible cryptocurrencies (*C1*, *C2*) with the same profitability.
- X is currently mining *C1* before the start of an epoch *E1*. At the beginning of *E1*, X switched to mine *C2*.

– The total hash rate (mining power) of *C1* dropped because X is not contributing his computational power to mine *C1*.
– Then at the beginning of epoch *E2* that is right after epoch *E1*, the difficulty of *C1* decreased. So now X switched back to mine *C1* because the lower difficulty mean it has a higher profitability.

Bitcoin is vulnerable to such attack because Bitcoin's difficulty readjustment algorithm is not suitable to deal with a sudden increase or decrease in hash rate [8]. If the hash rate is constant, the difficulty readjustment algorithm is able to maintain the expected block time. But if the hash rate is growing exponentially, it will not be able to maintain the expected block time. Meshkov, Chepurnoy and Jansen state that an ideal difficulty readjustment algorithm must:

(i) be resistant to known difficulty manipulation-based attacks such as the coin-hopping attack.
(ii) be able to maintain the expected block time even with the random increase or decrease in the network hash rate.

A new difficulty readjustment algorithm called *linear algorithm* that is based on linear least square method [9] was proposed. To obtain a more accurate prediction, rather than using only 1 cycle (previous 2016 blocks) of the actual time taken to mine a block, multiples cycles of 2016 blocks can be used instead. Three different simulation of the proposed algorithm was performed. The first is to observe how well the proposed algorithm is able to deal with exponential difficulty growth. A fixed increase of 10% in the hash rate was applied to every epoch.

From the observations, the difficulty calculated by the Bitcoin algorithm is always lower than the real difficulty and the linear algorithm. Thus, the average block time from the original Bitcoin algorithm is approximately 9 minutes and 5 seconds, which is about 10% lower than the expected 10 minutes. On the other hand, the difficulty calculated by the linear algorithm is closer to the real difficulty, albeit still lower with the average block time of about 9 minutes and 45 seconds. The linear algorithm has an average error of roughly 1.9% while the Bitcoin algorithm is sitting at about 9.1%.

The last simulation measured how well the algorithms deal with the coin-hopping attack. To manipulate the difficulty, the attacker repeatedly turned on and off his mining equipment. Difficulty calculated by the Bitcoin algorithm is always in antiphase with the real difficulty and the average block time is 10 minutes 10 seconds, while the average block time for linear algorithm is 10 minutes 5 seconds.

3.4 Fast(er) Difficulty Adjustment for Secure Sidechains

During the Scaling Bitcoin workshop in Milan 2016, Friedenbach proposed few solutions regarding the Bitcoin difficulty adjustment algorithm [14]. Friedenbach at first, suggested to adjust the difficulty more frequently, or at least every N blocks where $N < 2016$. By adjusting the difficulty earlier, Bitcoin will be able to respond to sudden event such as sudden increase or decrease in hash rate more quickly. However, when the difficulty adjustment interval increases, the error rate also increases. This is due to fact that the algorithm is not reacting fast enough to sudden shifts in hash rate. In addition, when the difficulty adjustment interval is very low, the error rate actually increases exponentially, because the algorithm is overreacting to some lucky blocks (blocks are mined within few seconds or minutes). Thus adjusting the difficulty every block is actually not recommended.

In the second suggestion, the sliding window is decoupled from the adjustment frequency. Currently in Bitcoin where the difficulty will readjust once every 2016 blocks, the actual time taken to mine the previous 2016 blocks were used in order to calculate the new difficulty. Friedenbach proposed to treat the interval of difficulty adjustment and the size of sliding window (actual time taken to mine X blocks) as two independent parameters to tweak. Last but not least, suggestion to change the control algorithm was also made as Bitcoin is currently using a simple and naive control algorithm.

3.5 On the Security and Performance of Proof of Work Blockchains

Gervais, Karame, Wüst, Glykantzis, Ritzdorf and Capkun studied the extent to which consensus and network parameters such as block size, block time, network propagation and etc., affect the performance and security of PoW blockchains [10]. Ever since the release of Bitcoin in 2009, it has been forked multiple times. All the forks include fine tuning the consensus parameters of Bitcoin (block time and hash function) and network parameters (block size and information propagation mechanism) in order to improve the efficiency of blockchain. For example, two of the most well-known forks of Bitcoin, Dogecoin and Litecoin implemented the block time of 1 minute and 2.5 minutes respectively instead of 10 minutes.

Even though many different consensus protocols such as PoS, Practical Byzantine Fault Tolerance (pBFT), PoB, and etc. were introduced, PoW is still the main consensus protocols that many new cryptocurrencies choose to implement. Although the security of Bitcoin has been studied in depth, but that is not the case for other variants of PoW blockchains. A new framework was proposed in order to analyze the relationship between performance and security of various network and consensus parameters of PoW blockchains. The framework proposed by Gervais et al. has two key components: i) a PoW blockchain instance and ii) a blockchain security model.

Bitcoin, Litecoin, Dogecoin and Ethereum were instantiated as blockchain instances with their default network and consensus parameters. A simulator that is able to mimic the blockchain instances are designed and implemented in order to obtain a more realistic result. The PoW blockchain instances will output measured or simulated stale block rate, block propagation times and throughput. The proposed framework enables Gervais et al. to measure the security and performance of PoW blockchains when considering different consensus and network parameters. However, as this paper main focus is on the performance of PoW blockchain, only the performance component is reviewed.

Stale blocks are undesirable because they cause chain forks to happen, and it could affect the performance and security of blockchain. Bitcoin, Litecoin and Dogecoin are all PoW-based blockchain and are using the same propagation system but with different block time and block sizes [11]. Table 1 suggests that block interval (block time) and block sizes heavily affect the stale block rate. For instance, Bitcoin which has larger average block size (534.8KB) but 4 times longer block interval (10 minutes) as compared to Litecoin (6.11 KB & 2.5 minutes) results in a higher stale block rate of 0.41% vs 0.273%. On the contrary, the difference in stale block rate between Litecoin and Dogecoin is mainly caused by the block interval (2.5 minutes vs 1 minute) as their average block size was roughly the same (6.11 KB vs 8KB). In Ethereum, stale blocks are known as uncle blocks. The stale block rate of Ethereum is 6.8% as compared to Bitcoin 0.41%. Although it has the smallest average block size of 1.5KB, but the block interval is also the lowest (10 – 20 seconds) and the number of available public nodes are also higher than both Litecoin and Dogecoin. Based on the observation, we can infer that a very low block interval is not necessarily the best as seen from the example of Dogecoin and Ethereum.

Table 1. Impact of different blockchain parameter choices on the network propagation times/median block propagation time (t_{MBP}). Stale block rate (r_{s}) and average block size (S_{B}) were measured over the last 10000 blocks [10].

	Bitcoin	Litecoin	Dogecoin	Ethereum
Block interval (mins)	10	2.5	1	1/6 to 2/6
Public nodes	6000	900	600	4000
Mining pools	16	12	12	13
t_{MBP} (sec)	8.7	1.02	0.85	0.5 0.75
r_s (%)	0.41	0.273	0.619	6.8
S_B (KB)	534.8	6.11	8	1.5

Impact of block interval Multiple simulations with different block interval ranging from 25 minutes to 0.5 seconds were performed to investigate the effects of the block interval on t_{MBP} and r_s. Four different cases of block request management system method were also defined for the simulation, including:

1. Standard block request management
2. Standard block request management + unsolicited block push from miners
3. Standard block request management + unsolicited block push from miners + relay network
4. Sendheaders mechanism with unsolicited block push and relay network

With unsolicited block push, miner is able to broadcast their mined blocks without advertisement. Whereas in relay network, blocks are propagated immediately after the verification. A lower block size can be achieved because transactions are reference with a 2 bytes transaction ID. On the contrary, Bitcoin adopted sendheaders mechanism since version 0.12. Peers can receive future *block headers* directly from their peers, skipping usage of *inv* messages, thus reducing latency and bandwidth overhead. Each simulation was run for 10000 blocks, for each block request management system method.

From Table 2, the r_s for block interval of 10 minutes with the standard request management method was 1.85%, which is relatively close to the 1.69% reported by Decker and Wattenhofer [11]. On the other hand, the r_s significantly dropped when unsolicited block push for miners were added to the method (Case 2). Unsolicited block push profit the miners since miners were interconnected and the first node with faster propagation method was able to reach the network majority faster. However, r_s is not significantly affected by the addition of the relay network (Case 3). In contrast, the addition of the relay network (Case 3) significantly reduced the r_s when the block size was bigger than 2 MB as shown in Table 3. Gervais et al. stated that although the impact of the sendheader mechanism was limited, it was able to mitigate the partial eclipse attacks [12].

Table 2. Impact of block interval ranging from 25 minutes to 0.5 seconds on t_{MBP} and r_s. Block size was fixed to the current block size of Bitcoin (1MB) [10].

	Case 1		Case 2		Case 3		Case 4	
Block interval	t_{MBP} (s)	r_s (%)	t_{MBP} (s)	r_s (%)	t_{MBP} (s)	r_s (%)	t_{MBP} (s)	r_s (%)
25 mins	35.73	1.72	25.66	0.16	22.50	0.03	22.44	0.02
10 mins	14.7	1.51	10.65	0.13	9.41	0.14	9.18	0.13
2.5 mins	4.18	1.82	2.91	0.16	2.60	0.16	2.59	0.15
1 mins	2.08	2.15	1.34	0.35	1.30	0.25	1.27	0.29
30 secs	1.43	2.54	0.84	0.45	0.84	0.51	0.84	0.52
20 secs	1.21	3.20	0.67	0.86	0.69	0.85	0.68	0.82
10 secs	1.00	4.77	0.35	1.73	0.33	1.41	0.53	1.59
5 secs	0.89	8.64	0.37	2.94	0.45	2.99	0.44	3.05
2 secs	0.84	16.65	0.40	6.98	0.39	7.28	0.38	7.10
1 secs	0.82	26.74	0.53	12.44	0.38	12.59	0.37	12.52
0.5 secs	0.82	38.15	0.61	20.62	0.49	20.87	0.36	21.10

Impact of Block size Simulations with a fixed block interval of 10 minutes and ranging block sizes from 0.1 MB to 8 MB were performed to investigate the effects of block size on the performance. From Table 3, the t_{MBP} increased linearly whenever the block size increased, but when block size was equal to 8 MB, t_{MBP} and r_s actually increased exponentially. With an efficient network propagation mechanism, it is actually able to decrease the t_{MBP} and r_s. But in Case 2, the t_{MBP} is worse than that from the standard block request management

(Case 1) with block size of 8 MB. Thorough analysis of the results in Tables 1 to 3 can be found in [10].

Table 3. Impact of block size ranging from 0.1 MB to 8 MB on t_{MBP} and r_s. Block interval was fixed to the current block interval of Bitcoin (10 minutes) [10].

	Case 1		Case 2		Case 3		Case 4	
Block size	t_{MBP} (sec)	r_s (%)	t_{MBP} (sec)	r_s (%)	t_{MBP} (sec)	r_s (%)	t_{MBP} (sec)	r_s (%)
0.1 MB	3.18	0.32	2.12	0.03	2.02	0.03	2.02	0.2
0.25 MB	7.03	0.88	4.93	0.11	4.49	0.05	4.46	0.17
0.5 MB	13.62	1.63	9.84	0.13	8.65	0.05	8.64	0.06
1 MB	27.67	3.17	20.01	0.38	17.24	0.07	17.14	0.07
2 MB	57.79	6.24	44.6	1.12	35.49	0.08	35.38	0.1
4 MB	133.30	11.85	126.57	5.46	78.01	0.12	78.40	0.13
8 MB	571.50	29.97	875.97	15.64	555.49	0.43	550.25	0.4

4 Discussions and Conclusion

Although the PoW algorithm allows blockchains to achieve consensus in a distributed manner and achieve security without the need of a central authority, it is not without its deficiencies. It is lacking in speed, limiting the throughput blockchains are able to achieve. In addition, the PoW difficulty adjustment in Bitcoin is not responsive to sudden event or differences in hash rate. If there is a loss of this hash rate just after a difficulty adjustment (worst case), it would take months for the miners to slowly grind until the next difficulty adjustment. This will result in several difficulty adjustment cycles to level out the discrepancies due to the adjustment limit.

From this study, we will be proposing a solution to the inherent trade offs of PoW through the consideration of learning models such as expectation maximization and dynamic Bayesian network through reparameterization of the PoW scheme. The need to identify the parameters of the scheme is challenging as the blockchain itself has many facets to its framework, such as network, storage, view (data structure of the full ledger), sidechain as well as consensus [15]. In addition, key parameters that actually significantly alters the performance of the scheme has to be ascertained as well, and they need to fit the optimization performed by the learning models.

5 Acknowledgement

Financial support from the Ministry of Higher Education, Malaysia, under the Fundamental Research Grant Scheme with grant number FRGS/1/2018/ICT02

/MMU/03/6, as well as the Multimedia University Mini Fund with Project ID MMUI/180239, are gratefully acknowledged.

References

1. Haber, S., & Stornetta, W. S. (1990, August). How to time-stamp a digital document. In Conference on the Theory and Application of Cryptography (pp. 437-455). Springer, Berlin, Heidelberg.
2. Bayer, D., Haber, S., & Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time-stamping. In Sequences Ii (pp. 329-334). Springer, New York, NY.
3. Nakamoto, S. (2008, October 31). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved April 25, 2019, from https://nakamotoinstitute.org/bitcoin/
4. Dywork, C., & Naor, M. (1993). Pricing via Processing or Combatting Junk Mail. Advances in Cryptology - CRYPTO'92: Lecture Notes in Computer Science, 740, 139-147.
5. Jakobsson, M., & Juels, A. (1999). Proofs of Work and Bread Pudding Protocols(Extended Abstract). In IFIP — The International Federation for Information Processing (pp. 258-272). Springer.
6. Back, A. (2002). Hashcash - A Denial of Service Counter-Measure. Retrieved April 30, 2019, from https://www.researchgate.net/publication/2482110_Hashcash_-_A_Denial_of_Service_Counter-Measure
7. Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644-654.
8. Meshkov, D., Chepurnoy, A., & Jansen, M. (2017). Short Paper: Revisiting Difficulty Control for Blockchain Systems. In Data Privacy Management, Cryptocurrencies and Blockchain Technology (pp. 429-436). Oslo, Norway.
9. Lawson, C. L., & Hanson, R. J. (1974). Solving Least Squares Problems.
10. A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in CCS '16 Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 2016.
11. Decker, C., & Wattenhofer, R. (2013). Information propagation in the Bitcoin network. IEEE P2P 2013 Proceedings. Trento, Italy: IEEE.
12. Gervais, A., Ritzdorf, H., Karame, G. O., & Capkun, S. (2015). Tampering with the Delivery of Blocks and Transactions in Bitcoin. CCS '15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 692-705). Denver, Colorado, USA: ACM.
13. Investing.com. (n.d.). All cryptocurrencies. Retrieved from Investing.com: https://www.investing.com/crypto/currencies
14. Friedenbach, M. (2016, October 9). Scaling Bitcoin workshop : Milan 2016 - Fast difficulty adjustment. Retrieved from Scalingbitcoin: https://scalingbitcoin.org/transcript/milan2016/fast-difficulty-adjustment
15. Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E. G., Song, D. & Wattenhofer, R. (2016, February). On scaling decentralized blockchains. In International Conference on Financial Cryptography and Data Security (pp. 106-125). Springer, Berlin, Heidelberg.