# A Memory-Hard Blockchain Protocol

Sui Cheng
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
e-mail: cs1995@mail.ustc.edu.cn

Sian-Jheng Lin
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
e-mail: sjlin@ustc.edu.cn

***Abstract*—Bitcoins cryptocurrency is a decentralized digital currency released as open-source software in 2009. In these 9 years of development, the mining devices are changed from the public CPUs into a professional ASICs. The ASICs is a application-specific integrated circuit, which is designed exclusively for mining. Because of its outstanding computing power, the ASICs gradually replaced other devices and monopolies the Bitcoin market. In this paper, we propose a new blockchain chain protocol, which allows us to adjust the loading time of the mining process. This strategy reduces the ASIC and CPU mining rate of the gap. So that the using of ASIC mining is limited. Finally, we analyzed the safety of this new chain structure and compared it to the original chain protocol.**

***Keywords-Blockchain; memory-hard function; multiple related protocols***

## I. INTRODUCTION

Bitcoin is a decentralized, electronically encrypted currency [1]. As a billing system, Bitcoin does not rely on central agencies to issue new money and maintain transactions, but rather on a decentralized system to maintain a list of records, termed as blockchain [2]. Blockchain is a string of related data blocks generated by cryptography method. New blocks are always linked to its previous block. So blockchain is a protocol whose data is ranged by the time block generated [3].

In the bitcoin system, all users can contribute to the block chain maintained by the Proof-of-Work (PoW) protocol [4]. The PoW protocol requires that the hashing of a valid block is less than a predetermined target threshold. Each miner continues to find out the valid block by adjusting a input value of the hashing function, termed as nonce in the block. After obtaining a valid block, the miner will broadcast the block, and other miners will stop the mining after verifying the validness of the block.

In the traditional blockchain, except the creation block, the block header of each block contains the hash value of its parent block header. Each new block is generated by changing the nonce, and constantly calculating the hash value of its block header until the number is less than current difficulty. So mining is just a purely computational process. The faster computing speed you have, the greater possibility to dig into a new block you will have.

The common mining devices have undergone several changes, from PC to GPU and FPGA, until the advent of ASIC. Currently, the market of bitcoin mining has been dominated by ASIC, and it is unprofitable by using other mining devices [5]. The sole purpose of ASIC is dedicated to the mining process. However, the use of ASIC chips also had a series of problems in the early days:

- As the computing power of the entire network continues to rise rapidly, the life of the ASIC chips become very short (around 6 months only).
- Investing in ASIC mining devices increases the costs (electricity costs and cooling costs).
- Due to the immaturity of the industry, the delayed delivery of mining devices has led to the chips being eliminated to the customers.

Although using ASIC chips for mining is now quite mature, and the life of mining devices have become longer expectancy, the mining industry has shifted from the personal field to a large professional mining center. Therefore, the mining industry is still very unfriendly to individual miners. Further, most of the computational power in the entire network is concentrated in the minority, which not only increase the waste of resources, but also increase the possibility of the 51% attack. This runs counter to the original purpose of bitcoin [6].

To break the monopoly of ASIC in mining industry, there exist multiple methods such as memory-hard functions. The memory-hard function is a class of hash functions which is difficult to be parallel. In this function, the time required to complete a given computational problem is primarily determined by the time to load data from memory, and this allows us to reduce the portion of the computational power on the overall cost of time. Therefore, by increasing the number of times to load data from memory, the I/O will dominate the whole processing time, so as to achieve the purpose of resisting ASICs.

Some memory-hard functions have been proposed, such as scrypt and Primecoin [7]. Those algorithms can be divided into two steps, and the first step is to generate an array of pseudo-random numbers from a seed. This array is stored in the main memory. To compute the hash value, the input of the hashing function is the concatenation of the block header with some numbers randomly picked from the the array of pseudo-random numbers.

As several pseudo-random numbers must be loaded, data loading will affect the time of hash computations. However, there exists a strategy, termed as time-memory trade-off. In the scrypt function, each pseudo-random number is

978-1-5386-6410-0/18/$31.00 ©2018 IEEE

calculated from the previous one, ASIC miners can reduce the amount of space used in main memory by half by storing only the pseudo-random numbers with odd indices [7], [8]. If the hashing function requires the pseudo-random numbers with even indices, the value can be computed from the previous one. The time-memory trade-off causes that it is possible to avoid the influences of data loading in hashing computations, and the performance gap between ASICs and general purpose CPUs is still very large possibly.

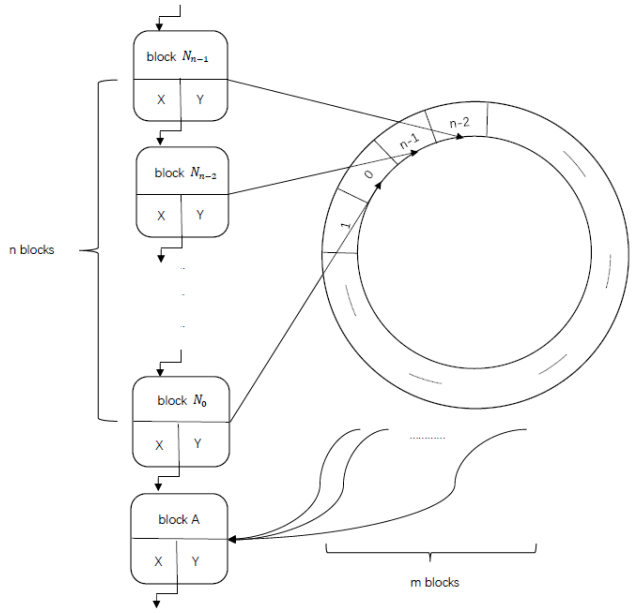


Figure 1. Circular queen.

To solve this issue, a new hashing function is required without the property the time-memory trade-off. That is, if the size of the memory is less than a threshold, the performance is declined significantly, and it does not have obvious way to design a method to possess the time-memory trade-off. Further, we also desire that the times of data loading is controllable in the protocol.

This paper presents a new blockchain protocol by adding some more links in each block. This causes that the time of the mining process is dominated by I/O, rather than the computing power.

The rest of this paper is organized as follows. In Section II, we propose a new blockchain protocol and describe it exactly. In Section III, we analyse the time-memory trade-off properties and loading time compare to traditional memory-hard functions, and also analyse its security exactly. In section IV, we summarize this paper.

## II. PROPOSED PROTOCOL

In this subsection, we describe the proposed protocol in detail. The proposed protocol has two parameters $(n, m)$, where $n$ indicates the number of hash values that may will be used in the mining process, and $m$ indicates the number of hash values chosen in the memory. For performances, those $n$ hash values should be stored in the cache or the memory, or else the processor must spend a lot of time to access these values in the mining process. In the proposed protocol, the hash values of the last $n$ blocks in the blockchain are denoted as $\{N_i | i = 0, \ldots, n-1\}$ in order, where $N_0$ refers to the latest block. The $m$ of those $n$ hash values with $N_0$ will be used in the hashing computation of the new block. Given the value of *nonce*, the following gives the steps to choose those $m$ hash values.

1) Let $t_0 = nonce$ and i = 1. Calculate $t = hash_0(t_0)$;
2) Let $t_i = N_t$. Calculate $t = hash_0(t_i)$. If $i = m$, output $(t_0, t_1, \ldots, t_m)$, or else $i = i + 1$ and repeat this step.

Notice that the codomain of $hash_0$ is $\{0, \ldots, n-1\}$. One can use $hash_0(t_0) = x \ mod \ (n)$. The hashing is calculated by $R = hash(t_0||t_{1,\ \gamma}||\ldots||t_{m,\gamma}||N_0)$. If the value $R$ meets the target difficulty, the mining process is completed, or else choose a new value of *nonce* and repeat the procedure to find out the new $m$ hash values. In the implementation, these $n$ hash values are stored in a circular queue, as shown in Fig. 1, to avoid the data movement to generate a new block.

The proposed protocol requires $n$ hash values $\{N_i | i = 0, \ldots, n-1\}$ in the mining process. However, when the length of the block chain is less than $n$, the proposed protocol cannot be applied. To solve this issue, a possible way is that, the parameter $n$ and $m$ can be reduced to the length of the block chain $t$. That is, we use two parameters $(n', m')$, where $n' = \min\{n, t\}$ and $m' = \min\{m, n'\}$, to substituted $(n, m)$ when $t < n$.

## III. DISCUSSIONS

### A. Time-Memory Trade-off Properties

The PoW protocol scrypt consists of two steps. The first step is to generate $n$ pseudo-random numbers sequentially, and each pseudo-random number is computed by using the prior pseudo-random number. In the second step, the hash value is produced by using a number of pseudo-random numbers picked by a pseudo-random order. This allows the time-memory trade-off by only storing $n/2$ pseudo-random numbers with even indices in memory, and other pseudo-random numbers can be produced by the numbers in the memory. As ASIC is much faster than general purpose CPU in hashing computations, ASIC mining machines can sacrifice the computational power to reduce the memory required. Thus, it is limited to reduce the gap between these two types of mining machines.

However, the proposed PoW protocol does not possess time-memory trade-off property, due to the fact that each hash value cannot be calculated by using other hash values directly. This feature is called Moemry-hard. Thus, if the $n$ hash values cannot be fully stored in memory, some hash values shall be stored in secondary storage devices (e.g. hard disks). When the hash value is required in hashing computations, the processor have to access the hard drives, and this will slow down the entire mining speeds.

### B. Analysis of Loading Time

As the proposed PoW protocol is memory-hard, the mining speed will be reduced significantly when the size of memory is less than $n$. The number of hashing values stored in main-memory is denoted as $s$. This subsection gives the

average loading time to compute a hashing value when $s < n$. The loading time from main-memory is denoted as $t_0$, and the loading time from secondary storage devices is denoted as $t_1$.

When $s = n$, the average loading time of the hash computation is $m * t_0$. If $s < n$, the average loading time of the hash computation is given by $m * (s/n * t_0 + (n - s)/n * t_1)$. In general, $t_1$ is tens of thousands of times larger than $t_0$. Thus, an appropriate value of $n$ can limit the advantages of ASICs in hashing computations.

### C. Security Analysis

In this section, we analyse the security of the block chain by using the purpose PoW protocol. In the head of a block, the hash value of transaction is denoted as $Y$ and the rest of the head is denoted as $X$.

***1) Single block replacing attack*:** We analyse the difficulty of replacing a block in the blockchain. For the conventional blockchain shown in Fig. 2, block $A$ is the parent of block $B$. We consider that the attacker tampers with the transactions in the block, and this will change the value of $Y$. The hash value of the tampered transaction is denoted as $Y'$. If the transactions in Block $A$ is modified, block $A_{Y'}$ must satisfy

$$hash(A_X||A_Y)=hash(A_X||A_{Y'}). \quad (1)$$

Assume the probability of finding a suitable $A_{Y'}$ to meet (1) is $p$.

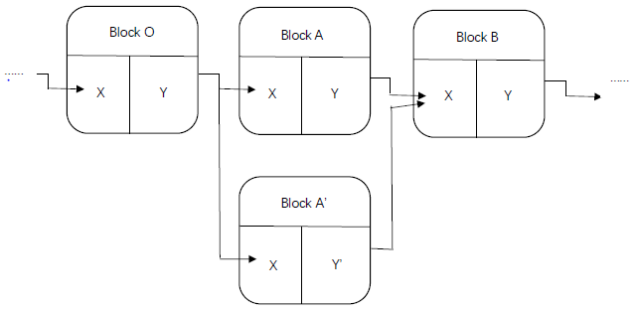


Figure 2. Attack traditional blockchain.

For the proposed protocol shown in Fig. 3, each block is associated with $m$ blocks other than its parent block, (2) should be satisfied when the block $N_0$ is changed:

$$hash(N_{d1,Y}||N_{d2,Y}||...||N_{di,Y}||...||N_{dm,Y}||N_{0,X}||N_{0,Y}) = hash(N_{d1,Y}||N_{d2,Y}||...||N_{di,Y}||...||N_{dm,Y}||N_{0,X}||N_{0,Y'}), \quad (2)$$

where each $d_i$ is computed by the algorithm in Sec.(II). That is, $d_1 = hash_0(A_{nounce})$, and $d_i = hash_0(N_{di-1})$, for $i = 1, \ldots, m$. The probability of finding a suitable $N_{0,Y'}$ to meet (2) is denoted as $q$. In the worst case, if $d_i \neq 0$, for $i = 1, 2, \ldots, m$, the length we have changed in (1) and (2) are the same. By the birthday attacking [9], we have $q = p$.

If the attacker tampers the block $N_0$ to $N_0'$, which is used in the hashing computations of the child block $A$. Further, $N_0$ is also used in the hashing computations of $t$ more blocks denoted as $G_1, G_2, \ldots, G_t$. Let $U_i$ denote the $m + 1$ hash values used in the hashing computation in $G_i$, and let $U_0$ denote the set of the $m + 1$ hash values used in the hashing computation in $A$. Thus, $hash(N_0) \in U_i$, for $i = 0, 1, \ldots, t$. Then we have

$$hash(u_{i,0}||\ldots||u_{i,k}||\ldots||u_{i,m-1}||u_{i,m}) = hash(u_{i,0}||\ldots||u_{i,k}'||\ldots||u_{i,m-1}||u_{i,m}), \quad (3)$$

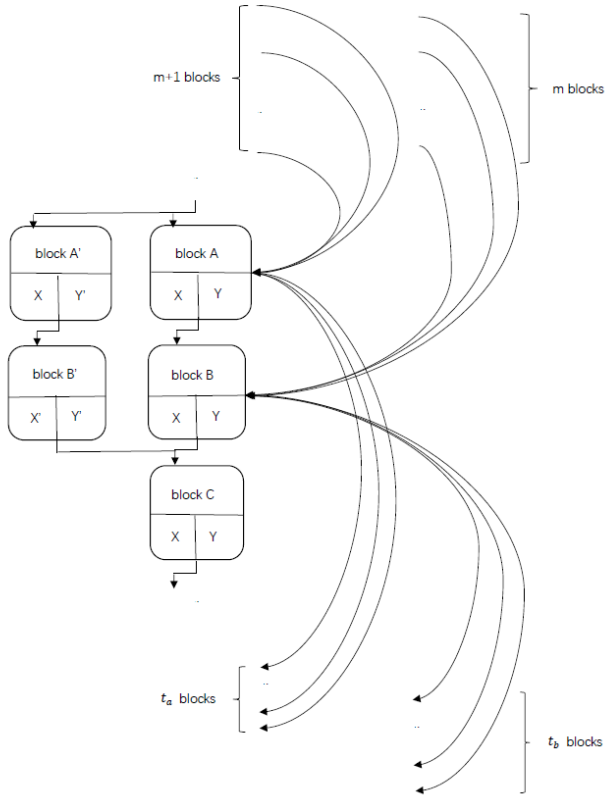


Figure 3. Attack new blockchain.

for each $u_{i,j} \in U_i, j = 0, 1, \ldots, m$. Notably, $u_{i,k} = hash(N_{0,Y}) \in U_i$ and $u_{i,k}' = hash(N_{0,Y'})$ denote the replaced hash value, for $i = 0, 1, \ldots, t$.

These $t + 1$ equations can be treated as independent. In the worst case, only $u_{i,k} = hash(N_0) \in U_i$, thus the probability is $q^{t+1} = p^{t+1} \leq p$. This shows that the proposed protocol is more secure than the conventional blockchain protocol when $t \geq 1$.

***2)Adjacent blocks replacing attacks*:** In Fig.4, if the block $A$ is replaced with $A'$, which is used in the hashing computations of the child block $B$. Further, the block $B$ is replaced with $B' = B_{X'}||B_{Y'}$, which is used in the hashing computations of the block $C$. In the conventional blockchain protocol, when $A$ is replaced with $A'$, a successful attack has to satisfy

$$B_{X'} = hash(A'). \quad (4)$$

When $B$ is replaced with $B'$, a successful attack has to satisfy

$$hash(hash(A')||B_{Y'})=hash(hash(A)||B_Y). \quad (5)$$

The probability of satisfying (5) is denoted as $p_1$.

In the new protocol, blocks $B$ and $C$ are both associated with $m+1$ prior blocks. For $i = 0, 1, \ldots, m$, let $(B_i)_{i=0}^{m}$ denote the vector consisting of the $m+1$ hash values used in the hashing computation in $B$, and let $(C_i)_{i=0}^{m}$ denote the vector consisting of the $m+1$ hash values used in the hashing computation in $C$. Notably, $B_0 = hash(A)$ and $C_0 = hash(B)$. Then we have

$$B_{X'} = hash(B_1||\ldots||B_m||B_0'), \quad (6)$$

for $B_0'=hash(A')$. When $B$ is replaced with $B'$, we have to meet

$$\begin{aligned} &hash(C_1||\ldots||C_m||C_0) \\ =&hash(C_1||\ldots||C_m||C_0'), \end{aligned} \quad (7)$$

for $C_0'=hash(B')$, where $B'=B_{X'}||B_{Y'}$. The probability of satisfying (7) is denoted as $q_1$. Comparing (5) with (7), the length of $Y'$ and the output length of hash function are the same. Thus, by the birthday attacking, we have $q_1 = p_1$.

If block $A$ is used in the hashing computations of block $B$ and $t_a$ more blocks denoted as $GA_1, GA_2, \ldots, GA_{ta}$. Further, block $B$ is used in the hashing computations of block $C$ and $t_b$ more blocks denoted as $GB_1, GB_2, \ldots, GB_{tb}$.

Let $UA_i$ denote the vector of the $m+1$ hash values used in the hashing computation in $GA_i$, for $i = 0, 1, \ldots, t_a$. Notably, $UA_0$ corresponds to the hashing computation in $B$, and $hash(A) \in UA_i$, for $i = 0, 1, \ldots, t_a$. Let $UB_i$ denote the vector of the $m+1$ hash values used in the hashing computation in $GB_i$, for $i = 0, 1, \ldots, t_b$. Notably, $UB_0$ corresponds to the hashing computation in $C$, and $hash(B) \in UB_i$, for $i = 0, 1, \ldots, t_b$. Then we have

$$\begin{aligned} &hash(ua_{i,0}||\ldots||ua_{i,k}||\ldots||ua_{i,m-1}||ua_{i,m}) \\ =&hash(ua_{i,0}||\ldots||ua_{i,k}'||\ldots||ua_{i,m-1}||ua_{i,m}) \end{aligned} \quad (8)$$

for each $ua_{i,j} \in UA_i$, $j = 0, 1, \ldots, m$. Notably, $ua_{i,k} = hash(A_Y) \in UA_i$ and $ua_{i,k}'=hash(A_{Y'})$ denote the replaced hash value, for $i = 0, 1, \ldots, t$. Further, we have to satisfy

$$\begin{aligned} &hash(ub_{i,0}||\ldots||ub_{i,k}||\ldots||ub_{i,m-1}||ub_{i,m}) \\ =&hash(ub_{i,0}||\ldots||ub_{i,k}'||\ldots||ub_{i,m-1}||ub_{i,m}) \end{aligned} \quad (9)$$

for each $ub_{i,j} \in UB_i$, $j = 0, 1, \ldots, m$. Notably, $ub_{i,k} = hash(B_Y) \in UB_i$ and $ub_{i,k}'=hash(A_{B'})$ denote the replaced hash value, for $i = 0, 1, \ldots, t$.

It can be seen that these $t_a + 1$ and $t_b + 1$ equations are independent. Thus, the probability of replacing the block $A$ with $A'$ to keep (8) is $P_1 = q_1^{t_a+1}$, for $t_a \geq 0$, and the probability of replacing the block $B$ with $B'$ to keep is $P_2 = q_1^{t_b+1}$, for $t_b \geq 0$. Therefore, the probability of a successful attack is $P = P_1 * P_2 = q_1^{t_a+t_b+2} < p_1$, which is lower than the probability of the successful attack on conventional blockchain protocols.

## IV. CONCLUSIONS

In this paper, we proposed a memory-hard PoW protocol on blockchain. The proposed protocol is a memory-hard function without the time-memory trade-off. Thus, the memory has to store the last $n$ hash values of the blockchain. Then $m$ of $n$ hash values, as well as the last hash value , are chosen to compute the hash of the new block. So each block is not only associated with its parent block but also with the $m$ of $n$ prior blocks. The analysis shows that the proposed protocol is a pure memory-hard function. Then we show that the security of the proposed protocol is better than the conventional blockchain protocol.

## REFERENCES


[1] Swan M. *Blockchain: Blueprint for a new economy[M]*. " O'Reilly Media, Inc.", 2015.

[2] Zyskind G, Nathan O. *Decentralizing privacy: Using blockchain to protect personal data*. Security and Privacy Workshops (SPW), 2015 IEEE. IEEE, 2015: 180-184.

[3] Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[4] Vukolić M. *The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication*. International Workshop on Open Problems in Network Security. Springer, Cham, 2015: 112-125.

[5] Tapscott D, Tapscott A. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world[M]*. Penguin, 2016.

[6] S. Nakamoto. *"Bitcoin: A Peer-to-Peer Electronic Cash System"*. http://bitcoin.org/bitcoin.pdf

[7] Alwen J, Chen B, Pietrzak K, et al. *Scrypt is maximally memory-hard*, Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2017: 33-62.

[8] Percival C, Josefsson S. *The scrypt password-based key derivation function*. No.RFC 7914. 2016.

[9] Bellare M, Kohno T. *function balance and its impact on birthday attacks*. International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2004: 401-418.