ASIC-resistant Hash Functions

Ainur R. Zamanov[#1], Vladimir A. Erokhin[*], Pavel S. Fedotov[#]
[#] Department of Computer Systems of Technologies; [*] Department of Industrial Economics and Management
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Moscow, Russia
[1]zamanov@ieee.org

***Abstract*— At present, the use of application-specific integrated circuit (ASIC) in electronic payment systems (EPS) based on cryptocurrencies gives a decisive advantage in mining. In the future, this can lead to the fact that the computing power of the network will be concentrated in the hands of a limited number of participants that contradicts the ideas of Bitcoin's creators about a distributed electronic payment system. This article describes one of the solutions to this problem, namely the use of ASIC-resistant hash functions in the proof-of-work system, which will help make EPS more distributed and, accordingly, more stable. The purpose of the study is to analyze ASIC-resistant hash functions. Modified algorithms of the proof-of-work mechanism are proposed.**

Keywords— cryptocurrency; hash function; proof-of-work.

I. INTRODUCTION

The decentralized approach is popular in applications. Such systems are characterized by:

- independence from a single transaction processing center, which makes it very difficult to rollback them;
- high speed of transactions;
- low transaction costs.

Bitcoin is one of the technologies using this paradigm. It is a set of concepts underlying a distributed electronic payment system. Bitcoin is a peer-to-peer network, whose members communicate with each other using a specific protocol.

A transaction in the Bitcoin network describes the transfer of a certain number of the accounting units - bitcoins - from one or more accounts (called input addresses) to one or more accounts (called output addresses). In addition to addresses, the transfer amount, the electronic digital signature of the sender, transactions contain some metadata.

All information about transactions is stored in a distributed data structure - blockchain. The blockchain is a sequence of blocks containing information about the transactions carried out. Each block contains a fingerprint hash value of the previous block, which makes it impossible to change the data in the block without changing the subsequent ones.

Creation of the new block (mining) is carried out by using the proof-of-work mechanism. It is made in order that the probability of creating the new block at a given node is proportional to its computing power. Proof-of-work in Bitcoin is the search for an unknown random number (nonce), whose hash of combining with block data (block_data) satisfies the following inequality:

$$H(nonce \,||\, block\_data) < target$$

Where H is a hash function, $target$ is a predetermined number, called the difficulty of the network, $||$ is a concatenation operation. This problem is the problem of enumeration of integers when using cryptographic hash functions.

SHA-256 is used as a hash function in Bitcoin [2]. It has good cryptographic properties, but it has a huge drawback: it is essentially easy to calculate on the application-specific integrated circuit (ASIC). This is a problem for the community because using ASIC gives a huge advantage in mining. Accordingly, the computing power of the network is increasingly consolidated in a limited range of nodes and the system becomes less distributed, which contradicts the ideas of the creators of Bitcoin [1].

This article is devoted to the analysis of ASIC-resistant hash functions.

II. SHA-256 ALGORITHM

The SHA-2 family of algorithms, which includes SHA-256, was published in 2002 [3].

The SHA-256 algorithm consists of two steps: preprocessing and calculating the hash.

During preprocessing, the bit representation of a message of length l is padded by one, k zeros, and a 64-bit binary representation of the number l so that the following equality holds:

$$k + l + 1 \equiv 448 \bmod 512$$

The length of the padded message is a multiple of 512 bits. After the addition, the message is parsed into N blocks of 512 bit $M^{(0)}, M^{(1)}, \cdots, M^{(N)}$. $H^{(0)}$ is a constant consisting of eight 32-bit words and being the initial hash value.

Next, the step of directly computing the hash value begins. In $i = 1$ to N:

978-1-5386-4340-2/18/$31.00 ©2018 IEEE

- Prepare the message schedule $\{W_t\}$ where $0 \le t \le 63$;
- Initialize the eight working variables, a,b,c,d,e,f,g and h with the $(i-1)$ th hash value;
- 64 function rounds are performed over these 8 variables. A diagram of the SHA-256 round function is shown in Fig. 1;
- compute the i th intermediate hash value $H^{(i)}$.

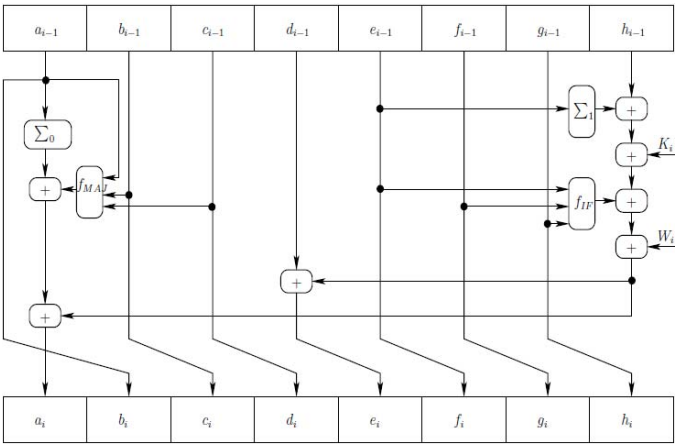

Fig. 1. Round function of SHA-256 hash function

The output value of the hash function is the concatenation of intermediate hash values $H_0^{(N)},\cdots,H_7^{(N)}$.

III. Analysis of ASIC-resistant Algorithms

A. Equihash

In 2017, the concept of Equihash was proposed. It involves the use of the proof-of-work mechanism on the basis of the generalized birthday problem [4].

In contrast to the proof-of-work used in Bitcoin, in addition to an unknown integer, it is required to find a sequence of numbers satisfying the conditions of complexity and generalized collision.

According to the concept of [5], for every integer iterated by means of the Wagner algorithm for the generalized paradox of birthdays, a search for a sequence $\{x_1,x_2,\cdots,x_{2^k}\}$ of numbers of length $(\frac{n}{k+1}+1)$ bits satisfying equality:

$$H(I \,\|\, V \,\|\, x_1) \oplus \cdots \oplus H(I \,\|\, V \,\|\, x_{2^k}) = 0$$

Where H is a cryptographic hash function, I is a given message, V is a nonce, k,d,n are the specified network parameters.

Then, the difficulty of the network is verified that the hash of the concatenation $I,V,x_1,\cdots,x_{2^k}$ starts with d zeros. Fig. 2 shows the general scheme of Equihash.

Equihash time and memory requirements:

- memory is $2^{\frac{n}{k+1}+k}$ bytes;
- time – $(k+1)\cdot 2^{\frac{n}{k+1}+d}$ calls of H hash function;
- solution size – $2^k \cdot (\frac{n}{k+1}+1)+160$ bit;
- verification cost – 2^k calls of hash function and XOR operation.

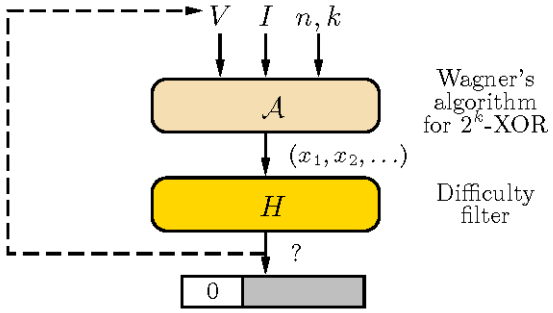

Fig. 2. Equihash's general scheme

B. Ethash

Another alternative algorithm is Ethash [6]. It is based on the need to store a huge data set in memory, which is unprofitable for ASIC devices.

Main stages of the algorithm:

- a seed is calculated from the block headers;
- a 16 MB pseudorandom cache is calculated from the message, which depends on the block number, while clients store this cache;
- from the cache, a 1 GB size data set is generated with the property that each element in the data set depends only on a small number of cache entries. The size of these data set grows linearly with time;
- mining consists in selecting random data set fragments and hashing their concatenation;
- for verification, it is sufficient to store the cache and compute certain fragments of the data set.

As we can see from the algorithm, for mining it is required to store a huge amount of data in RAM, while verification requires a small amount of memory.

C. Performance tests

In order to analyze practical results of these algorithms benchmark tests was provided (Table 1).

TABLE I. PERFORMANCE TESTS

Miner	Average performance		
	Classic Bitcoin proof-of-work (MH/s)	***Ethash using Claymore v 10.0 (MH/s)***	***Equihash using EWBF's miner (H/s) [k = 9, n = 80]***
Intel Core i7-5820K	10.5	0.93	39
Nvidia GeForce GTX 1080 Ti with 11 GB GDDR5X	32	30.2	657
Antminer S9	13243	-	-

The tests were conducted in the operating system Ubuntu 14.04 LTS. Antminer S9 is not compatible with Ethash and Equihash. Due to unprofitable mining, there is no ASIC for these algorithms Difficulty parameter d was the same for all algorithms. The test results show that proposed algorithms are advantageous for the GPU.

D. Proposals for improving proof-of-work algorithm

Based on the analysis of ASIC-resistant hash functions, changes were proposed to the existing Bitcoin proof-of-work algorithm:

- make the algorithm more demanding on memory;
- use an algorithm that would be more advantageous for CPU and GPU calculations;
- to avoid the threat of ASIC dominance, it is necessary to modify the algorithm using the solutions used in Equihash and Ethash.

IV. CONCLUSIONS

One of the main requirements for the hashing algorithm used is the demand for RAM. This is done to avoid the dominance of ASIC. The algorithm should not only require large computing powers, but also a large amount of RAM. The most effective algorithms are Equihash and Ethash. Both algorithms require memory. Equihash is based on the generalized birthday problem, while Ethash is based on the task of search of combinations from a given data set.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf. Accessed on: Nov. 17, 2017.

[2] A. Narayanan, J. Bonneau, . Felten, A. Miller and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton Univ. Press, 2016, pp. 75-98.

[3] *Secure Hash Standard*, FIPS PUB 180-4, 2015. DOI: 10.6028/NIST.FIPS.180-4

[4] D. Wagner, "A Generalized Birthday Problem," in *Advances in Cryptology - CRYPTO 2002*, Santa Barbara, CA, USA, pp. 288-304. DOI: 10.1007/3-540-45708-9_19

[5] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem," *Ledger*, vol. 2, pp. 1-30, 2017. DOI: 10.5915/LEDGER.2017.48

[6] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2015. [Online]. Available: http://gavwood.com/Paper.pdf. Accessed on: Dec. 04, 2017.