Performance of Hash Algorithms on GPUs for Use in Blockchain

Alexandr Kuznetsov
Department of information systems and technologies security
V. N. Karazin Kharkiv National University
JSC "Institute of Information Technologies"
Kharkov, Ukraine
kuznetsov@karazin.ua

Kyryl Shekhanin
Department of information systems and technologies security
V. N. Karazin Kharkiv National University
Kharkov, Ukraine
kyryl.shekhanin@nure.ua

Andrii Kolhatin
Department of information systems and technologies security
V. N. Karazin Kharkiv National University
JSC "Institute of Information Technologies"
Kharkov, Ukraine
kolgatin-a@yandex.ua

Diana Kovalchuk
Department of information systems and technologies security
V. N. Karazin Kharkiv National University
Kharkov, Ukraine
dianakovalhyk@ukr.net

Vitalina Babenko
Department of International E-Commerce and Business
V. N. Karazin Kharkiv National University
Kharkiv, Ukraine
vitalinababenko@karazin.ua

Iryna Perevozova
Department of Entrepreneurship and Marketing
Ivano-Frankivsk National Technical University of Oil and Gas
Ivano-Frankivsk, Ukraine
dianakovalhyk@ukr.net

***Abstract*—The main cryptographic primitives in blockchain networks are hashing functions that are designed to form short and unpredictable digests for the message entered. In blockchain networks, hashing is used to build linked block lists, which provide safe and secure storage of important information in a distributed repository. The peculiarity of the hash search problem in blockchain networks allows applying the maximum parallelization of calculations, what good are multithreaded graphics processors (GPUs). In this paper, we explore the performance of GOST 34.311, STRIBOG, KECCAK, SHA2, RIPEMD160, Blake2b, and Whirlpool cryptographic hashing algorithms. HashCat software and various GPUs were used for comparative analysis of efficiency. GPUs were used: Geforce 740M 2GB; Geforce GTX1050ti 4GB; Rx580 Aorus 4GB; Rx580 Sapphire Pulse 8GB; Sapphire Vega 56 8GB.**

Keywords—cryptographic hashing; blockchain technology; GPUs; efficiency

I. INTRODUCTION

One of the most important crypto-primitives used on blockchain networks, through which users are provided with a service of the integrity of user data blocks, is hashing functions. Hashing functions are primitives used in various cryptographic and non-cryptographic applications to provide information security services. But in blockchain networks, they are the primary primitive that allows the provision of such a security service as integrity [1]. In particular, the creation of linked lists that can securely store information in distributed repositories is implemented by hashing. Therefore, the properties of this primitive are extremely important for the construction and development of blockchain networks, namely to investigate the efficiency of hashing functions and is aimed at this paper.

By definition, hashing is the transformation of an input array of arbitrary length data into a fixed-length output string [2-4]. Such transformations are also called hash functions or collapsing functions, and their results are called a hash, a hash code, a hash sum, or a message digest [3].

Currently, a large number of hash functions, different in purpose and characteristics and capabilities, are developed and used [5-10]. In this work, we restrict ourselves to an analysis of GOST 34.311 [11-13], STRIBOG [13], KECCAK [14-16], SHA2 [17], RIPEMD160 [18], Blake2b [19, 20] and Whirlpool [21] algorithms. The purpose of the article is comparative analysis of efficiency of these algorithms using Graphics Processors (GPU) - devices of a personal computer or game console, which does the graphical rendering. Modern GPUs very efficiently process and portray computer graphics. Due to a specialized conveyor architecture, they are much more efficient in processing graphical information than a typical central processor. The peculiarity of the hash search in blockchain networks allows applying the maximum parallelization of calculations. Multithreaded GPUs that are hundreds of times faster than CPUs are well suited for this purpose [22, 23]. Therefore, comparative analysis of the efficiency of hash functions on different GPUs are relevant.

II. HASHING FUNCTIONS RESEARCHED

A. GOST34.11-94

To calculate the cryptographic hashing function in Russia, the standard GOST P 34.11-94 was introduced in 1994 [11] (which has already been abolished) [13]. This standard was later reissued as an international CIS standard GOST 34.311-95 [12]. On January 1, 2013, it was replaced by GOST P 34.11-2012 « Stribog» [13].

The standard defines an algorithm and procedure for calculating a hash function for a sequence of characters. When processing blocks, transformations according to the algorithm of cryptographic transformation of GOST 28147-89 are used [24]. A 256-bit block is being processed, the output value is also 256-bit length. The algorithm determines the checksum calculated across all outbound blocks, which is part of the final hash calculation, which somewhat complicates the collision search attack. Anti-collision measures are also being implemented based on the

incompleteness of the last block. Block processing is done according to the encryption algorithm GOST 28147-89 , which contains transformations on S-blocks, which significantly complicates the application of the method of differential cryptanalysis to search for collisions.

B. STRIBOG

The current Russian cryptographic standard GOST P 34.11-2012 "Information technology. Cryptographic protection of information. Hashing function" determines the algorithm and procedure for calculating the hash function Stribog. The algorithm was put into effect on January 1, 2013 [13].

The standard defines an algorithm and procedure for calculating a hash function for a sequence of characters. The major parameters of the algorithm are: hash size - 256 or 512 bits; the size of the input block is 512 bits.

The izzz.io blockchain platform with open source and smart contracts uses cryptographic libraries with the algorithm STREEBOG. This platform is used by BigNet, BitCoen, Buzcoin, Baikalika, NWP Solution, SBS Platform, NS Platform [25].

C. KECCAK

The Keccak cryptographic hashing algorithm was developed in [14]. On October 2, 2012, Keccak won the SHA-3 Cryptographic Algorithm Competition, conducted by the National Institute of Standards and Technology of the United States [15]. On August 5, 2015, the algorithm was approved and published as FIPS 202 standard [16].

FIPS 202 defines the SHA-3 family of cryptographic hash functions:

SHA3-224 – hash length of 224 bits;

SHA3-256 – hash length of 256 bits;

SHA3-384 – hash length of 384 bits;

SHA3-512 – hash length of 512 bits.

Today, the SHA-3 algorithm is one of the most common cryptographic hash functions that is used in many cryptocurrencies, such as Nexus (NXS), SmartCash (SMART), X-Cash (XCASH), MaxCoin (MAX), SecureCoin (SRC), Bitcoin File (BIFI), CreativeCoin, Slothcoin (SLOTH), 365Coin (365), Galleon (GLN), Helix Coin (HXC), CryptoMeth (METH), BitcointalkCoin (TALK) and others. [26, 27].

D. SHA2

The SHA-2 hash function was developed by the US National Security Agency (NSA) and published by the National Institute of Standards and Technology in the Federal FIPS PUB 180-2 in August 2002 [17]. In March 2012, the latest edition of FIPS PUB 180-4 was released, which added the SHA-512/256 and SHA-512/224 functions based on SHA-512 (since 64-bit SHA-512 architectures run faster than SHA-256) [72]. Thus, SHA-2 (Secure Hash Algorithm Version 2) is a family of cryptographic algorithms - unidirectional hash functions that includes the algorithms SHA-224, SHA-256, SHA-384, SHA-512, SHA -512/256 i SHA-512/224.

The SHA-2 family hash functions are based on the Merkle - Damgard structure. After the message is divided into blocks, each block contains 8 words. The algorithm passes each message block through a loop with 64 or 80 rounds. At each round, 2 words out of eight are converted, the transformation function is set by other words. The results of processing each block are summed up, the sum is the result of the hash function.

The US Government's SHA-224, SHA-256, SHA-384, and SHA-512 hashing algorithms are permitted for use in some government programs, including the use of other cryptographic algorithms and protocols, to protect non-confidential information. The standard also permits the use of SHA-2 by private and commercial entities. Therefore, the SHA-2 family algorithms are perhaps the most common hashing functions used, including in distributed decentralized blockchain systems and a large number of cryptocurrencies [28].

E. RIPEMD160

The cryptographic hash function of RIPEMD-160 was developed in [18]. For an arbitrary input message, the function generates a 160-bit hash value. The RIPEMD-160 is an upgraded version of RIPEMD, which in turn used MD4 principles and is compared to the more popular SHA-1.

RIPEMD-160 is an iterative hash function that works on 32-bit words. The round function accepts a 5-word link variable and a 16-word message block and translates it into a new link variable. All operations are defined over 32-bit words. The bit size of the hash result and the binding variable for RIPEMD-160 is increased to 160 bits (five 32-bit words), the number of rounds is increased from three to five, more differences are made between the two rows (not only steel values, but also Boolean functions and word order).

The RIPEMD algorithm is one of the most common hashing functions used in many modern cryptocurrencies [29].

F. Blake2b

The BLAKE hash function is proposed in [19, 20]. BLAKE is a family of four hash functions: BLAKE-224, BLAKE-256, BLAKE-384 and BLAKE-512. Like the SHA-2 algorithm, the BLAKE hash function has a 32-bit version (BLAKE-256) and a 64-bit version (BLAKE-512), from which other instances are output using different initial values, another padding, and reduced output.

The BLAKE algorithm is used in the Blakecoin cryptocurrency [30]. In addition, this algorithm is used as a component in algorithms X11, X12, X13, X14, X15, and X17 for mining various cryptocurrencies and distributed decentralized systems [31].

G. Whirlpool

The Whirlpool cryptographic hash function was developed in [21]. Published in November 2000. Hashes an incoming message up to 256 bits length. The output value of the Whirlpool hash function is 512 bits.

The WHIRLPOOL hash function is used as a component of cryptocurrency mining algorithms X14, X15, X17 [31].

III. Performance of Hashing Algorithms on GPU

The following hardware has been selected for comparative explorings of the performance of hash algorithms on graphical computing systems:

- Geforce 740M 2GB;
- Geforce GTX1050ti 4GB;
- Rx580 Aorus 4GB;
- Rx580 Sapphire Pulse 8GB;
- Sapphire Vega 56 8GB.

HashCat software was used for the research [32]. HashCat is a utility that provides password recovery. It is most actively used to recover WPA / WPA2 passwords, as well as keys to encrypted office documents. Since 2015, it has been distributed open source under the MIT license. The utility allows you to use any device that implements the OpenCL standard (OpenCL provides instruction-level and data-level parallelism and is an implementation of the GPGPU technique. OpenCL is a completely open standard, its use is not subject to license deductible).

The hashcat v5.1.0 version supports the ability to recover passwords for more than 100 algorithms, including hash functions such as: GOST 34.311, СТРІБОГ256, СТРІБОГ512, KECCAK 256, KECCAK 512, SHA2 256, SHA2 512, RIPEMD160, Blake2b, Whirlpool. The following methods can be used to recover the original password:

- Brute-force attack;
- Mask attack;
- Combinator attack;
- Dictionary attack;
- Fingerprint attack;
- Hybrid attack;
- Permutation attack;
- Rule-based attack;
- Table-lookup attack;
- Toggle-Case attack.

The general idea behind such attacks is to find the prototype by hashing / encrypting the data and comparing the result with the searched hash.

The test results are given in Tables 1 and 2.

Table 1 summarizes Benchmark results, that is, estimates of hash rate (number of generated hash codes per second) by sequential hashing of the dataset.

Table 2 shows the specific complexity indicators, namely the hash rate (number of generated hash codes per second), which accounts for one OpenCL computational kernel of the used graphical calculator.

TABLE I. Benchmark result on graphical computing systems using HashCat software (KHash/s)

	Geforce 740M 2GB	Geforce GTX1050ti 4GB	Rx580 Aorus 4GB	Rx580 Sapphire Pulse 8GB	Sapphire Vega 56 8GB
GOST 34.311	10442,9	65337,9	89450	91932,3	233100
STRIBOG256	3512.6	13613,5	56751,7	55162,9	99485
STRIBOG512	3518,4	13569,5	55207,8	56635,9	99641,5
KECCAK256	38149,7	260400	320400	327800	529600
KECCAK512	38285	262400	326400	334400	538700
SHA2 256	133400	889600	1700900	1755900	3088100
SHA2 512	38794,4	297300	405600	421600	710100
RIPEMD160	177000	1383100	2356200	2437700	4129100
Blake2b	96515,9	585100	1103300	1132800	1738000
Whirlpool	12516,8	64773,3	324500	333200	591400

TABLE II. The hashing rate (number of generated hash codes per second) that is attributed to one OpenCL computing core of the GPU used (KHash/s)

	Geforce 740M 2GB	Geforce GTX1050ti 4GB	Rx580 Aorus 4GB	Rx580 Sapphire Pulse 8GB	Sapphire Vega 56 8GB
GOST 34.311	5221,45	10889,65	2484,7	2553,6	4162,5
STRIBOG256	2706.3	2268,9	1576,4	1532,3	1776,5
STRIBOG512	1759,2	2261,5	1533,5	1573,2	1779,3
KECCAK256	19074,8	43400	8900	9105,5	9457,1
KECCAK512	19142,5	43733,3	9066,6	9288,8	9619,6
SHA2 256	66700	148266,6	47247,2	48775	55144,6
SHA2 512	19397,2	49550	11266,6	11711,1	12680,3
RIPEMD160	88500	230516,6	65450	67713,8	73733,9
Blake2b	48257,9	97516,6	30647,2	31466,6	31035,7
Whirlpool	6258,4	10795,5	9013, 9	9255,5	10560,7

Figure 1 illustrates the hash rate diagram of Table 10 for clarity. As we can see, the fastest is the algorithm RIPEMD160, followed by the algorithm SHA2, Blake2b and others.

Figure 2 shows the diagrams of the specific velocity, that is, the hash rate attributable to one computational core of the graphical calculator. As you can see, the ranking of algorithms for performance is almost the same (RIPEMD160, SHA2, Blake2b, etc.), but the native speed values have changed. That is, each graphics device has a different number of computational cores and the fastest by the criterion of specific performance are Geforce calculators.

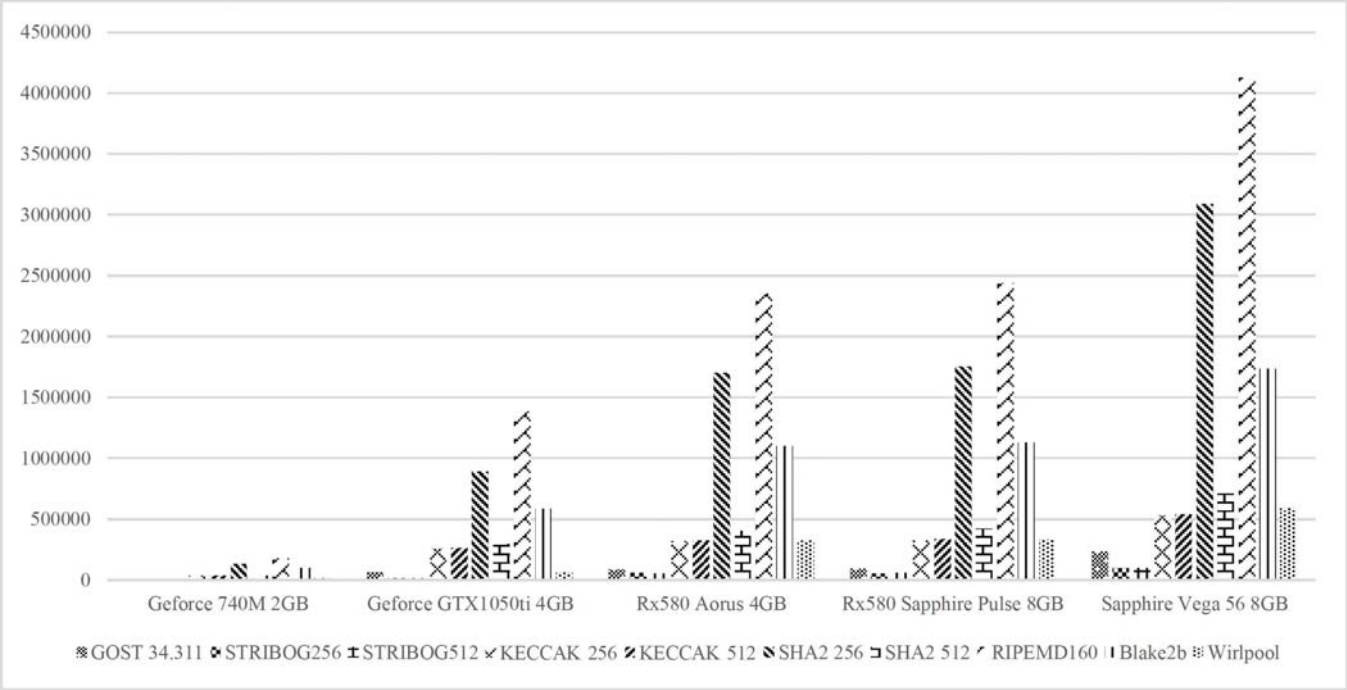

Fig. 1. Comparison of performance of hash algorithms on GPUs

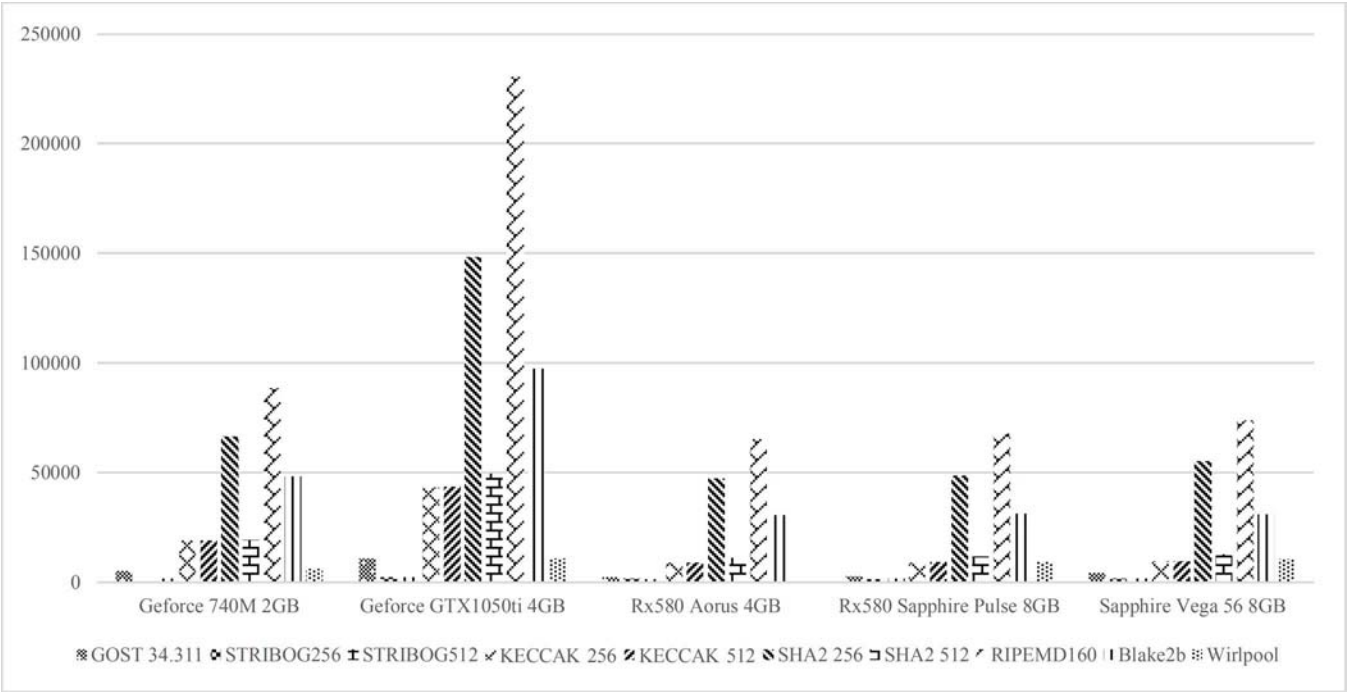

Fig. 2. Comparison of specific velocity of hash algorithms on GPUs

Thus, the conducted researches show that cryptographic hashing functions different in structure and mathematical transformations give different acceleration on computer systems. The most attractive are graphical or specialized computing devices.

This study may be useful for the development of decentralized systems using blockchain technology, as well as for improving cryptographic algorithms [33-36]. In addition, it can be useful in various applications in the field of information protection, data processing and transmission, including in modern telecommunication systems and networks [37-40].

IV. Conclusions

In this work, comparative studies of the performance of cryptographic hashing algorithms that are or may be applied in the consensus protocols of modern blockchain networks are carried out. The results of the comparative analysis make it possible to select cryptographic hashing functions according to performance criteria on different devices and explain their practical application for the construction of decentralized blockchain type systems. This also applies to the results of benchmarking on GPUs, especially regarding the possible development of a national blockchain segment.

A promising area for further research is to study and compare the statistical properties of the hash functions considered, especially with the use of the NIST STS and DIEHARD statistical security special packages.

REFERENCES

[1] NISTIR 8202. Blockchain Technology Overview, 2018, 68 p. Access mode: https://doi.org/10.6028/NIST.IR.8202.

[2] B. Preneel, “The State of Cryptographic Hash Functions,” Lecture Notes in Computer Science, pp. 158–182, 1999.

[3] C. Paar and J. Pelzl, “Kryptografie verständlich,” eXamen.press, 2016.

[4] P. Rogaway and T. Shrimpton, “Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance,” Lecture Notes in Computer Science, pp. 371–388, 2004.

[5] B. Preneel, “Cryptographic hash functions,” European Transactions on Telecommunications, vol. 5, no. 4, pp. 431–448, Sep. 2010.

[6] S. Contini, R. Steinfeld, J. Pieprzyk, and K. Matusiewicz, “A Critical Look at Cryptographic Hash Function Literature,” Coding and Cryptology, Jul. 2008.

[7] P. Gauravaram and L. R. Knudsen, “Cryptographic Hash Functions,” Encyclopedia of Information Assurance, pp. 1–10, Dec. 2010.

[8] W. E. Burr, “Cryptographic hash standards: where do we go from here?,” IEEE Security & Privacy Magazine, vol. 4, no. 2, pp. 88–91, Mar. 2006.

[9] J. Tchórzewski and A. Jakóbik, “Theoretical and Experimental Analysis of Cryptographic Hash Functions,” Journal of Telecommunications and Information Technology, vol. 1, pp. 125–133, Mar. 2019.

[10] A. Regenscheid, R. Perlner, S. Chang, J. Kelsey, M. Nandi, and S. Paul, “Status report on the first round of the SHA-3 cryptographic hash algorithm competition,” 2009.

[11] V. Dolmatov, Ed., “GOST R 34.11-94: Hash Function Algorithm,” Mar. 2010.

[12] GOST 34.311-95. Information technology. Cryptographic protection of information. Hash function. Date of introduction 1995-01-01. Access mode: http://docs.cntd.ru/document/gost-34-311-95

[13] A. Degtyarev, “GOST R 34.11-2012: Hash Function,” Aug. 2013.

[14] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “The Making of KECCAK,” Cryptologia, vol. 38, no. 1, pp. 26–60, Jan. 2014.

[15] NIST. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register, 72(112), November 2007. Access mode: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

[16] NIST Releases SHA-3 Cryptographic Hash Standard. August 05, 2015. Access mode: https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard

[17] Secure Hash Standard. Federal Information. Processing Standards Publication 180-2. 2002 August 1. (FIPS PUB 180-2). Access mode: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[18] The hash function RIPEMD-160. Access mode: http://homes.esat.kuleuven.be/~bosselae/ripemd160.html

[19] SHA-3 proposal BLAKE. Access mode: https://131002.net/blake/

[20] BLAKE2 – fast secure hashing. Access mode: https://blake2.net/

[21] LARC - Laboratório de Arquitetura e Redes de Computadores. Access mode: http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html

[22] Malware mints virtual currency using victim's GPU. Access mode: https://www.theregister.co.uk/2011/08/16/gpu_bitcoin_brute_forcing/

[23] More Bitcoin malware: this one uses your GPU for mining. Access mode: https://arstechnica.com/tech-policy/2011/08/symantec-spots-malware-that-uses-your-gpu-to-mine-bitcoins/

[24] V. Dolmatov, Ed., “GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms,” Mar. 2010.

[25] IZZZIO. Access mode: https://en.bitcoinwiki.org/wiki/IZZZIO

[26] SHA-3 Coins. Access mode: https://cryptorival.com/algorithms/sha3/

[27] Keccak hashing algorithm (SHA-3) – Keccak Coins and miner for Keccak. Access mode: https://coinguides.org/keccak-algorithm-miner-coins/

[28] SHA-256 Coins. Access mode: https://cryptorival.com/algorithms/sha256/

[29] Cryptography behind top 20 cryptocurrencies. Access mode: https://www.susanka.eu/coins-crypto/

[30] About Blakecoin. Access mode: https://blakecoin.org/about-blakecoin/

[31] Alexander Markov. “Algorithm X13 for mining on GPUs”. May 28, 2018. Access mode: https://miningbitcoinguide.com/mining/sposoby/x13

[32] Hashcat. Advanced Password Recovery. Access mode: http://hashcat.net/hashcat/

[33] I. Gorbenko, O. Kuznetsov, Y. Gorbenko, A. Alekseychuk and V. Tymchenko, "Strumok keystream generator," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, Ukraine, 2018, pp. 294-299. DOI: 10.1109/DESSERT.2018.8409147

[34] A. Kuznetsov, V. Frolenko, E. Eremin and O. Zavgorodnia, "Research of cross-platform stream symmetric ciphers implementation," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, Ukraine, 2018, pp. 300-305. DOI: 10.1109/DESSERT.2018.8409148

[35] Andrushkevych A., Gorbenko Y., Kuznetsov O., Oliynykov R., Rodinko M. A (2019) “A Prospective Lightweight Block Cipher for Green IT Engineering*”. In: Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*, vol 171. Springer, Cham, pp. 95-112. DOI: 10.1007/978-3-030-00253-4_5

[36] Kuznetsov O., Potii O., Perepelitsyn A., Ivanenko D., Poluyanenko N. (2019) “Lightweight Stream Ciphers for Green IT Engineering”. *In: Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*, vol 171. Springer, Cham, pp. 113-137. DOI: 10.1007/978-3-030-00253-4_6

[37] P. Gurzhiy, B. Gorodetsky, O. Yudin and Y. Ryabukha, "The Method of Adaptive Counteraction to Viral Attacks, Taking Into Account Their Masking in Infocommunication Systems," *2019 3rd International Conference on Advanced Information and Communications Technologies (AICT)*, Lviv, Ukraine, 2019, pp. 423-425. doi: 10.1109/AIACT.2019.8847893

[38] S. Harkusha, O. Harkusha and O. Yudin, "Model of frequency and time resource allocation WiMAX with taking into account the defined priorities," *2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*, Kharkiv, 2016, pp. 88-91. doi: 10.1109/INFOCOMMST.2016.7905344

[39] Galata, L.P., Korniyenko, B.Y., Yudin, A.K. Research of the simulation polygon for the protection of critical information resources. *XVII International Scientific and Practical Conference on Information Technologies and Security* (ITS 2017), Kyiv, Ukraine, November 30, 2017, pp. 23-31.

[40] Tkach, B. P., & Urmancheva, L. B. (2009). Numerical-analytic method for finding solutions of systems with distributed parameters and integral condition. Nonlinear Oscillations, 12(1), 113-122. doi:10.1007/s11072-009-0064-6