I. J. Computer Network and Information Security, **2021, 2, 1-15**
Published Online April 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2021.02.01

Performance Analysis of Cryptographic Hash Functions Suitable for Use in Blockchain

Alexandr Kuznetsov[1], **Inna Oleshko**[2], **Vladyslav Tymchenko**[3], **Konstantin Lisitsky**[4], **Mariia Rodinko**[5] **and Andrii Kolhatin**[6]
[1,3,4,5,6] V. N. Karazin Kharkiv National University, Svobody sq., 4, Kharkiv, 61022, Ukraine
E-mail: kuznetsov@karazin.ua, tvlad.tyma@gmail.com, konstantin.lisickiy@mail.ru, m.rodinko@gmail.com, kolgatin-a@yandex.ua

[2] Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine
E-mail: inna.oleshko@nure.ua

Received: 30 June 2020; Revised: 04 September 2020; Accepted: 21 October 2020; Published: 08 April 2021

Abstract: A blockchain, or in other words a chain of transaction blocks, is a distributed database that maintains an ordered chain of blocks that reliably connect the information contained in them. Copies of chain blocks are usually stored on multiple computers and synchronized in accordance with the rules of building a chain of blocks, which provides secure and change-resistant storage of information. To build linked lists of blocks hashing is used. Hashing is a special cryptographic primitive that provides one-way, resistance to collisions and search for prototypes computation of hash value (hash or message digest). In this paper a comparative analysis of the performance of hashing algorithms that can be used in modern decentralized blockchain networks are conducted. Specifically, the hash performance on different desktop systems, the number of cycles per byte (Cycles/byte), the amount of hashed message per second (MB/s) and the hash rate (KHash/s) are investigated. The comparative analysis of different hashing algorithms allows us to choose the most suitable candidates for building decentralized systems type of blockchain.

Index Terms: Hashing, Performance, Cryptographic Algorithm, Blockchain, Cryptocurrency.

1. Introduction

Modern decentralized information systems and networks buit on the blockchain technology are increasingly distributes and uses in various applications. Thus, blockchain technologies are used in the construction of cryptocurrencies, for the construction of decentralized systems of different purpose and functional tasks, which unite, for example, key certification centers, in construction of distributed decentralized networks of electronic identification and electronic voting, in building information systems with support so-called smart contracts, etc. Selection of a specific hash function is one of the main task for designing a blockchain system. Till nowadays, there has been no extensive analysis of hash functions, which can be use in blockchain technology.

This work mainly focused on the performance analysis of cryptographic hash functions that can be used in blockchain systems. The most common and applicable cryptographic hashing algorithms, which are standardized internationally and nationally, are considered, as well as algorithms that, although not standardized, are used in most modern decentralized systems built on blockchain technology. In particular, the following cryptographic hashing functions are explored:

- the ARGON family of cryptographic algorithms (ARGON2D and ARGON2I hash functions) [1], used in some cryptocurrencies, such as MMXVI;
- the BALLOON hashing algorithm [2] is used in Deft cryptocurrency;
- the BLAKE cryptographic algorithm family consists of the following hash functions: BLAKE224, BLAKE256, BLAKE384, BLAKE512 [3-5]. This family of algorithms are used in such cryptocurrency as Monero, Blakecoin, Electroneum and other[5].
- the BMW Cryptographic Algorithm family consisting of the following hash functions as BMW224, BMW256, BMW384, BMW512 [6]. This family of algorithms are used in such cryptocurrency’s as CLOAK, Dark, XST, MXT, ONION, etc.;
- the CUBEHASH family of algorithms consisting of the following hash functions: CUBEHASH224, CUBEHASH256, CUBEHASH384, CUBEHASH512 [7-10]. This family of hashing algorithms are used in such cryptocurrency mining algorithms as X17, X13, X11, etc.;

- the DJB-2 hashing algorithm [11]. This hashing function is not cryptographic, but it is still used in the X17 cryptocurrency mining algorithm;
- family of ECHO cryptographic algorithms such as ECHO224, ECHO256, ECHO384, ECHO512 [12]. These algorithms are used in such cryptocurrency's as DarkCoin, Navcoin, PinkCoin, etc.;
- the ED2K hashing algorithm [13, 14], used in file sharing networks and eDonkey2000 and Overnet file sharing programs [14];
- the EDONR family of algorithms such as EDONR256 and EDONR512 [15, 16];
- the DAGGER-HASHIMOTO hashing algorithm and its further development and improvement - ETHASH algorithm [17-19]. It was later replaced by the Ethash algorithm;
- the FUGUE family of cryptographic algorithms (FUGUE224; FUGUE256; FUGUE384; FUGUE512) [20]. These algorithms are used in such cryptocurrency's as Navcoin, MinersCoin, KoboCoin etc.
- cryptographic hashing algorithm GOST34.11-94-256 [21];
- the GROESTL cryptographic algorithm family (the GROESTL224 hash function; GROESTL256; GROESTL384; GROESTL512) [22, 23], is used in some blockchain systems, including the Verge cryptocurrency project;
- the HAMSI family of cryptographic algorithms consisting from the hashing functions HAMSI224, HAMSI256, HAMSI384, HAMSI512 [24]. This algorithms used in such cryptocurrency's as Firecoin, Orlycoin, EverGreenCoin etc.;
- the Has160 hashing algorithm [25];
- the J-H hashing algorithms such as J-H224, J-H256, J-H384, J-H512 [26]. They are used in such cryptocurrency's as Orlycoin, EverGreenCoin, XSH, Dash etc.;
- the KECCAK hashing functions such as KECCAK224, KECCAK256, KECCAK384, KECCAK512, which are standardized as SHA3) [27, 28]. Nowadays this family of algorithms is one of the most widespread in different cryptocurrencies. They are used in such cryptocurrency's as Quark, METH, SMART, BIFI, GLN, TALK etc.;
- the Kupyna hashing algorithm (Kupyna256 and Kupyna512 hash functions) [29]. This algorithm is adopted as the national standard of Ukraine DSTU 7564: 2014;
- the LOSELOSE hashing algorithm [30]. The LOSELOSE hashing function is not cryptographic, but it is used in the X17 cryptocurrency mining algorithm;
- the LUFFA family of cryptographic algorithms (LUFFA224 hash function; LUFFA256; LUFFA384; LUFFA512) [31]. This hashing functions is used in such cryptocurrency's as Dash, Navcoin, Halcyon, Orlycoin etc.;
- the LYRA hashing algorithms: LYRA2RE, LYRA2REV2 [32, 33]. This family of algorithms is used in such cryptocurrency's as ORE, MONA, VTC, RUP etc.;
- the MD family of hashing functions (MD4 and MD5) [34, 35];
- the PANAMA256 cryptographic hashing algorithm [36];
- the PROGPOW hashing algorithm [37]. The first cryptocurrency on the ProgPoW algorithm was Bitcoin Interest (BCI) [38];
- the RIPEMD160 cryptographic hashing algorithm [39], used in many modern cryptocurrencies [40];
- the SCRYPT cryptographic hashing algorithm [41]. Scrypt is used in many cryptocurrencies as a validation algorithm. The algorithm was first implemented for Tenebrix (released in September 2011) and became the basis for Litecoin and Dogecoin. Also, the scrypt algorithm is used by ProsperCoin, CashCoin, MonaCoin, Mooncoin and many others;
- the SHA1 cryptographic hashing algorithm [42];
- the SHA2 family of cryptographic algorithms (SHA2-256 and SHA2-512 hash functions) [43], are used in such cryptocurrencies as Peercoin, Bitcoin, NXT, Namecoin and others [44];
- the SHABAL family of hashing functions such as SHABAL256 and SHABAL512 [45]. This cryptographic algorithm is used in such cryptocurrencies as Firecoin, KoboCoin, Orlycoin etc.;
- the SHAVITE family of hashing functions (SHAVITE224, SHAVITE256; SHAVITE384; SHAVITE512) [46]. These algorithms are used in such cryptocurrencies as Dash, PURA, SNRG, EverGreenCoin etc.;
- the SIMD hashing functions such as SIMD224, SIMD256, SIMD384, SIMD512 [24]. These algorithms are used in such cryptocurrencies as Dash, EverGreenCoin, MLM, DeepOnion etc.;
- the SKEIN family of cryptographic algorithms (SKEIN224; SKEIN256; SKEIN384; SKEIN512) [47]. These algorithms are used in such cryptocurrencies: Hshare, Pura, BitSend, KoboCoin etc.;
- the SNEFRU256 hashing algorithm [48];
- the STREEBOG hashing algorithm (in STREEBOG256 and STREEBOG512 variants) [49]. This algorithm was described in the standard GOST R 34.11-2012. Open-source blockchain izzz.io and intellectual contracts for the use of cryptographic libraries with the STREEBOG algorithm implemented: BitCoen, NWP Solution, Buzcoin, NS Platform etc. [50];

- the TIGER hashing algorithm [51], used in Gnutella, Gnutella2, Direct Connect file sharing protocols, as well as in Phex, BearShare, LimeWire, Shareaza, DC ++ and Valknut file sharing;
- the WHIRLPOOL hashing algorithm [52] is used in such cryptocurrency's as Halcyon, Orlycoin, KoboCoin, Verge etc.;
- hashing algorithms from the "X" family (algorithms X11; X12; X13; X14). The "X" family of algorithms has a high degree of protection against hacker attacks.

The algorithms of the "X" family got their name according to the number of hash functions used in them. For example, the following hash functions are used in the X11 algorithm: Echo, Shavite, Luffa, Keccak, Groestl, Blake, Bmw, Jh, Skein, Cubehash, Simd. The X13 algorithm uses such hash functions as Keccak, Groestl, Blake, Bmw, Jh, Skein, Cubehash, Simd, HAMSI, FUGUE. The X17 algorithm uses the following 17 cryptographic functions: Echo, Shavite, Luffa, Keccak, Groestl, Blake, Bmw, Jh, Skein, Cubehash, Simd, HAMSI, FUGUE, SHABAL, WHIRLPOOL, LOSELOSE, DJB-2 [53].

The first cryptocurrency whose blockchain was built on X11 was DarkCoin. To date, hashing algorithms of the X family are widely used in cryptocurrencies. For example, the X15 algorithm is used in such cryptocurrencies: KOBO, HAL, NMB, SHLD, SOLE, MARYJ, XWT, DUB, EGC, TAGR, ORLY, OMA, FIRE, CRAB, FOREX, XPRO, HTML5, DKD. The X13 algorithm is used as a cryptocurrency mining tool: DANK, AERO, AGS, APEX, BOST, CLOAK, JUDGE, NAV, OPAL, PSEUD, SLG, SSV, UTIL, XST, EKN, KORE, STV, AM, TRI, QBK, BOOM, FUTC, HEDG, CON, BLITZ, and this list is constantly growing.

The above hashing algorithms have been constructed using different information message processing schemes and using different mathematical transformations of data blocks for calculating hash codes. The different ideology of constructing these algorithms is due to the considerations of the research authors and their subjective ideas about the rational construction of the hashing function by the criteria of stability / complexity. Our focus is, first of all, on the possibility of using the studied algorithms in the construction of decentralized blockchain networks, in particular, cryptocurrencies, distributed technology projects, smart contracts, etc. The hashing algorithms considered in this paper are applied to over 90% of existing decentralized systems projects [54-57].

2. Related Work

The article [58] considers the main cryptoalgorithms used in cryptocurrencies to create hash functions. Special attention devoted to the study of security and confidentiality of the blockchain technology.

In the paper [59], was proposed lightweight hash-based blockchain architecture for IIoT. In the proposed architecture, the following hash functions were used - QUARK, PHOTON, and SPONGENT. Authors provides a comparison of security and performance of hash functions QUARK, PHOTON, and SPONGENT. The article also provides top 10 hash functions for blockchain-based coins.

The article [60] considers the peculiarities of the use of cryptographic protocols transactions on blockchain technologies. The advantages and disadvantages of the most popular and used on any blockchain platform algorithms for reaching consensus are given.

In the paper [61] was proposed to use streaming hash functions for blockchain systems in the non-financial sphere. The authors showed that stream hash functions which based on the theory of linear sequential schemes satisfy all the minimum requirements and can be widely used in non-financial areas.

In the article [62] was analyzed the hashing functions for their application in the Blockchain systems. Authors focused their attention on the results of the comparative analysis of the main properties of hash functions and recommendations for their application. For the use in modern blockchain systems, 512-bit hash functions were recommended: SHA-2 512, SHA-3 512 and Kupyna 512.

In this article, we continue the research on the performance of hashing algorithms, which we began in [56]. We are investigating hash rates across multiple computing platforms, including desktop systems and GPUs. In our study, we summarize the test results from [56] and consider over 80 different algorithms, including both internationally standardized hash functions and little-known specialized algorithms [54].

3. Comparison of Explorings on Desktops

For comparative explorings, etalon and (if any) optimized software implementations of information hashing algorithms were used.

The explorings is conducted on the following computer desktops:

- 64-bit computing platform with using AMD Ryzen Threadripper 2970WX 24-Core Processor 3.0 GHz.;
- 64-bit computing platform with using Intel Core i9-7980 2.60 GHz.

These computing platforms are the most common with high computing performance.

Performance studies were conducted on three criteria:
the number of cycles of the computing system per byte (Cycles/byte);

- message volume per second (MB/s);
- the number of generated hash codes per second (KHash/s).

These criteria characterize both absolute (second and third indicators) and specific performance indicators (first indicator). The obtained data contains the results of testing hash rate for different lengths of input. In particular, inputs (consecutive counter values) 2^{10} and 2^{20} bytes were sent to the input of each algorithm.

The following tables summarize the results of comparison of the performance of these algorithms for each of the computing desktops used. In particular, Table 1 summarizes the results of comparison of the performance of different hashing functions on 64-bit computing platform using AMD Ryzen Threadripper 2970WX 24-Core Processor 3.0 GHz. In this case, 2^{10} and 2^{20} bytes of data were supplied as input. In the table 2 we show values of performance of the explored algorithms for the 64-bit computing platform Intel Core i9-7980 2.60 GHz bytes.

The analysis and researches show that the fastest algorithms are DJB-2 and LOSELOSE. But according to their specification these are not cryptographic algorithms. These hash functions calculate the normal checksum and the search for prototypes for such transformations is trivial. Further along the speed of conversion are the algorithms MD4, EDONR-256, EDONR-512, ED2K and other cryptographic transformations, the stability of which is not satisfactory to date. Most cryptographic features have comparable aspects rates of hash code generation, including algorithm Kupyna (Ukraine's national standard). The slowest hashing algorithms were the X-family hash functions and the ARGON2D, ARGON2I, and SCRYPT algorithms.

Table 1. Testing results for different algorithms (AMD Ryzen Threadripper 2970WX 24-Core Processor 3.0 GHz)

Algorithm name	2^{10} byte input text			2^{20} byte input text		
	Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
ARGON2D	2605872,83	0,001149	0,001122	2531,66	1,182493	0,001128
ARGON2I	2232922,54	0,001335	0,001304	2180,86	1,367934	0,001305
BLAKE-224	11,67	256,69	250,67	10,89	274,93	0,26
BLAKE-256	11,77	254,39	248,42	10,98	272,71	0,26
BLAKE-384	8,18	366,63	358,04	7,15	418,59	0,40
BLAKE-512	8,21	364,85	356,30	7,17	417,26	0,40
BMW-224	5,92	505,83	493,97	5,23	572,68	0,55
BMW-256	5,89	508,28	496,36	5,20	575,82	0,55
BMW-384	3,52	851,12	831,17	2,79	1073,26	1,02
BMW-512	3,51	853,89	833,88	2,78	1076,57	1,03
CUBEHASH-224	20,31	147,48	144,02	15,17	197,17	0,19
CUBEHASH-256	20,54	145,53	142,12	15,35	195,16	0,19
CUBEHASH-384	20,32	147,48	144,02	15,17	197,58	0,19
CUBEHASH-512	20,37	147,02	143,58	15,19	197,10	0,19
DJB-2	2,52·10-5	2,13·107	2,08·107	2,48·10-5	1,91·107	18181,8
ECHO-224	27,72	108,21	105,68	24,56	121,91	0,12
ECHO-256	27,40	109,31	106,74	24,32	123,17	0,12
ECHO-384	52,56	57,01	55,67	45,92	65,23	0,06
ECHO-512	51,13	58,53	57,16	45,43	65,90	0,06
ED2K	4,02	747,38	729,86	3,75	798,00	0,76
EDONR-256	3,89	765,94	747,99	3,66	816,65	0,78
EDONR-512	2,17	1377,89	1345,60	1,91	1569,72	1,50
FUGUE-224	19,23	155,81	152,15	17,11	175,14	0,17
FUGUE-256	19,13	156,53	152,86	17,06	175,52	0,17
FUGUE-384	31,62	98,87	96,55	26,40	113,49	0,11
FUGUE-512	37,46	79,89	78,01	33,01	90,72	0,09
GOST34.11-94	43,74	68,44	66,83	42,19	70,97	0,07
GROESTL-224	23,23	128,93	125,91	21,10	141,83	0,14
GROESTL-256	22,98	130,32	127,27	20,88	143,44	0,14
GROESTL-384	31,30	95,62	93,38	26,67	111,92	0,11
GROESTL-512	31,39	95,43	93,19	26,25	114,06	0,11
HAMSI-224	32,69	91,67	89,53	32,51	92,07	0,09

Table 1. Continuation

Algorithm name	2^{10} byte input text			2^{20} byte input text		
	Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
HAMSI-256	32,91	90,99	88,86	32,18	93,02	0,09
HAMSI-384	84,47	35,75	34,91	82,00	36,50	0,03
HAMSI-512	83,53	35,83	34,99	81,97	36,56	0,03
HAS160	5,41	554,22	541,23	5,05	592,42	0,56
JH-224	33,28	90,68	88,56	30,63	97,81	0,09
JH-256	32,74	91,44	89,30	30,72	97,53	0,09
JH-384	33,19	90,15	88,03	31,17	96,25	0,09
JH-512	32,68	91,61	89,46	30,63	97,73	0,09
KECCAK-224	10,94	273,21	266,81	9,59	311,15	0,30
KECCAK-256	10,98	272,85	266,46	10,21	293,14	0,28
KECCAK-384	13,79	218,18	213,07	13,40	223,10	0,21
KECCAK-512	20,54	145,72	142,30	19,33	154,91	0,15
KUPYNA-256	19,45	157,04	153,36	17,83	173,03	0,17
KUPYNA-512	34,29	89,77	87,66	28,72	106,81	0,10
LOSELOSE	2,49·10-5	1,94·107	1,89·107	2,47·10-5	1,97·107	18867,9
LUFFA-224	12,21	240,55	234,92	11,35	265,06	0,25
LUFFA-256	12,04	248,65	242,83	11,23	266,47	0,25
LUFFA-384	17,57	170,58	166,59	15,86	188,42	0,18
LUFFA-512	23,54	127,42	124,44	21,36	140,30	0,13
LYRA2REV2	30,96	96,73	94,46	10,97	273,07	0,26
LYRA2RE	30,22	99,20	96,88	10,90	275,22	0,26
MD4	3,97	756,00	738,28	3,74	802,28	0,77
MD5	5,63	532,00	519,53	5,26	568,03	0,54
PANAMA-256	3,42	873,81	853,33	1,58	1906,50	1,82
RIPEMD-160	13,57	220,71	215,53	12,75	234,84	0,22
SCRYPT	883,21	3,41	3,33	23,40	128,19	0,12
SHA1	8,80	340,12	332,14	8,25	362,95	0,35
SHA2-256	12,60	237,66	232,09	11,80	253,71	0,24
SHA2-512	8,71	344,47	336,40	7,68	389,95	0,37
SHABAL-256	8,28	360,71	352,25	6,58	454,32	0,43
SHABAL-512	8,32	359,96	351,53	6,55	458,49	0,44
SHAVITE-224	16,86	176,32	172,19	15,89	188,29	0,18
SHAVITE-256	16,60	180,42	176,19	15,62	191,73	0,18
SHAVITE-384	27,45	109,06	106,50	24,42	122,78	0,12
SHAVITE-512	27,21	110,04	107,46	24,17	123,90	0,12
SIMD-224	22,27	134,05	130,91	20,97	142,84	0,14
SIMD-256	22,30	134,35	131,20	20,96	142,82	0,14
SIMD-384	27,11	110,73	108,13	24,09	124,24	0,12
SIMD-512	27,04	110,81	108,21	24,00	124,77	0,12
SKEIN-224	4,57	657,41	642,01	4,22	707,54	0,67
SKEIN-256	4,59	654,54	639,20	4,25	703,74	0,67
SKEIN-384	4,69	638,60	623,63	4,34	690,31	0,66
SKEIN-512	4,53	661,15	645,65	4,21	710,90	0,68
SNEFRU-256	108,35	27,63	26,99	105,08	28,49	0,03
STREEBOG-256	35,10	85,31	83,31	29,98	100,02	0,10
STREEBOG-512	35,01	85,52	83,52	29,86	100,28	0,10
TIGER	4,25	705,16	688,63	3,95	758,19	0,72
WHIRLPOOL	18,05	166,18	162,28	16,93	177,24	0,17
X11	45,04	66,50	64,95	7,28	411,37	0,39
X12	52,14	57,40	56,06	7,26	412,83	0,39
X13	58,66	50,81	49,62	7,22	414,62	0,40
X14	61,13	48,97	47,83	7,25	413,15	0,39

Table 2. Performance testing of hashing algorithms for 64-bit computing platforms (Intel Core i9-7980 2.60 GHz)

Algorithm name	2^{10} byte input text			2^{20} byte input text		
	Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
ARGON2D	1904191,27	0,001361	0,001329	1866,23	1,386180	0,001322
ARGON2I	1905170,88	0,001360	0,001328	1880,48	1,378617	0,001315
BLAKE-224	10,25	259,23	253,15	9,20	281,65	0,27
BLAKE-256	9,80	264,93	258,72	9,22	281,12	0,27
BLAKE-384	6,63	392,14	382,95	5,81	446,39	0,43
BLAKE-512	6,74	385,36	376,33	5,92	438,00	0,42
BMW-224	5,57	462,34	451,50	4,96	524,29	0,50
BMW-256	5,53	465,62	454,71	4,92	527,19	0,50
BMW-384	3,01	862,32	842,11	2,39	1073,26	1,02
BMW-512	3,05	851,12	831,17	2,43	1064,54	1,02
CUBEHASH-224	20,20	127,94	124,94	14,94	173,40	0,17
CUBEHASH-256	20,13	128,75	125,74	14,91	173,84	0,17
CUBEHASH-384	20,71	125,16	122,22	14,69	176,59	0,17
CUBEHASH-512	20,22	128,20	125,20	14,95	173,15	0,17
DJB-2	1,53·10-5	2,27·107	2,22·107	1,41·10-5	2,23·107	21276,6
ECHO-224	24,11	107,46	104,94	21,41	121,04	0,12
ECHO-256	24,05	107,86	105,33	21,35	121,46	0,12
ECHO-384	46,74	55,45	54,15	41,32	62,69	0,06
ECHO-512	45,03	57,55	56,21	39,91	64,77	0,06
ED2K	3,32	780,19	761,90	3,11	835,52	0,80
EDONR-256	3,69	699,52	683,12	3,44	756,00	0,72
EDONR-512	1,86	1387,01	1354,50	1,64	1572,08	1,50
FUGUE-224	17,63	147,21	143,76	15,62	165,94	0,16
FUGUE-256	17,58	147,46	144,00	15,69	165,16	0,16
FUGUE-384	28,14	92,07	89,91	25,29	102,47	0,10
FUGUE-512	35,86	72,31	70,62	31,25	82,89	0,08
GOST34.11-94	36,76	70,44	68,79	35,85	72,40	0,07
GROESTL-224	21,74	118,46	115,68	19,22	134,92	0,13
GROESTL-256	20,86	124,28	121,37	19,22	136,11	0,13
GROESTL-384	30,30	85,65	83,64	25,25	102,58	0,10
GROESTL-512	33,12	85,24	83,25	26,06	102,69	0,10
HAMSI-224	30,21	85,33	83,33	29,13	88,95	0,08
HAMSI-256	29,49	87,94	85,88	28,94	89,66	0,09
HAMSI-384	88,41	29,30	28,61	85,88	30,20	0,03
HAMSI-512	87,49	29,31	28,62	85,98	30,33	0,03
HAS160	5,03	514,51	502,45	4,72	548,99	0,52
JH-224	30,40	83,81	81,84	28,42	91,23	0,09
JH-256	29,69	87,48	85,43	27,85	93,26	0,09
JH-384	30,37	85,42	83,42	28,46	91,12	0,09
JH-512	30,35	85,60	83,59	27,86	92,64	0,09
KECCAK-224	9,45	274,07	267,64	8,36	310,51	0,30
KECCAK-256	9,64	268,87	262,56	8,96	289,50	0,28
KECCAK-384	12,08	214,70	209,66	11,82	217,68	0,21

Table 2. Continuation

Algorithm name	2^{10} byte input text			2^{20} byte input text		
	Cycles/byte	MB/s	KHash/s	Cycles/byte	MB/s	KHash/s
KECCAK-512	17,41	149,28	145,79	16,40	158,11	0,15
KUPYNA-256	19,24	134,85	131,69	17,46	148,59	0,14
KUPYNA-512	31,92	81,63	79,72	26,86	96,91	0,09
LOSELOSE	1,41·10-5	2,55·107	2,49·107	1,41·10-5	2,38·107	22727,3
LUFFA-224	11,87	218,09	212,98	11,25	230,56	0,22
LUFFA-256	12,12	213,82	208,81	11,46	226,62	0,22
LUFFA-384	18,07	143,54	140,18	16,34	158,66	0,15
LUFFA-512	23,46	110,97	108,37	20,70	125,23	0,12
LYRA2REV2	28,52	90,86	88,73	9,49	276,67	0,26
LYRA2RE	25,67	100,92	98,56	9,44	274,50	0,26
MD4	3,31	783,69	765,32	3,10	836,19	0,80
MD5	4,91	527,45	515,09	4,62	561,04	0,54
PANAMA-256	3,23	806,60	787,69	1,49	1747,63	1,67
RIPEMD-160	11,85	218,04	212,93	11,15	232,40	0,22
SCRYPT	722,61	1,80	1,75	20,56	125,73	0,12
SHA1	7,81	331,83	324,05	7,41	349,76	0,33
SHA2-256	13,46	192,54	188,03	12,68	204,88	0,20
SHA2-512	7,94	327,37	319,70	7,02	369,87	0,35
SHABAL-256	6,15	420,61	410,75	4,94	524,55	0,50
SHABAL-512	6,31	416,60	406,83	5,01	507,78	0,48
SHAVITE-224	14,19	180,07	175,85	13,46	191,91	0,18
SHAVITE-256	14,33	178,63	174,45	13,53	189,93	0,18
SHAVITE-384	24,41	105,95	103,47	21,67	119,52	0,11
SHAVITE-512	23,77	106,31	103,82	21,04	123,14	0,12
SIMD-224	19,98	130,05	127,00	18,78	138,13	0,13
SIMD-256	19,32	134,62	131,47	18,14	142,92	0,14
SIMD-384	24,52	105,58	103,10	21,74	119,05	0,11
SIMD-512	23,88	108,45	105,91	21,66	119,13	0,11
SKEIN-224	4,12	629,78	615,02	3,88	669,16	0,64
SKEIN-256	4,02	643,69	628,61	3,78	684,45	0,65
SKEIN-384	4,15	627,14	612,44	3,90	666,19	0,64
SKEIN-512	4,02	645,67	630,54	3,78	685,79	0,65
SNEFRU-256	89,21	29,02	28,34	86,62	29,95	0,03
STREEBOG-256	30,40	85,40	83,40	25,96	99,85	0,10
STREEBOG-512	31,59	82,20	80,28	26,84	96,91	0,09
TIGER	3,51	739,48	722,14	3,31	787,81	0,75
WHIRLPOOL	15,57	166,60	162,69	14,62	177,27	0,17
X11	40,95	63,76	62,26	5,91	439,47	0,42
X12	48,72	53,19	51,94	5,98	433,12	0,41
X13	54,36	47,66	46,54	5,93	437,09	0,42
X14	55,84	46,40	45,31	5,93	437,64	0,42

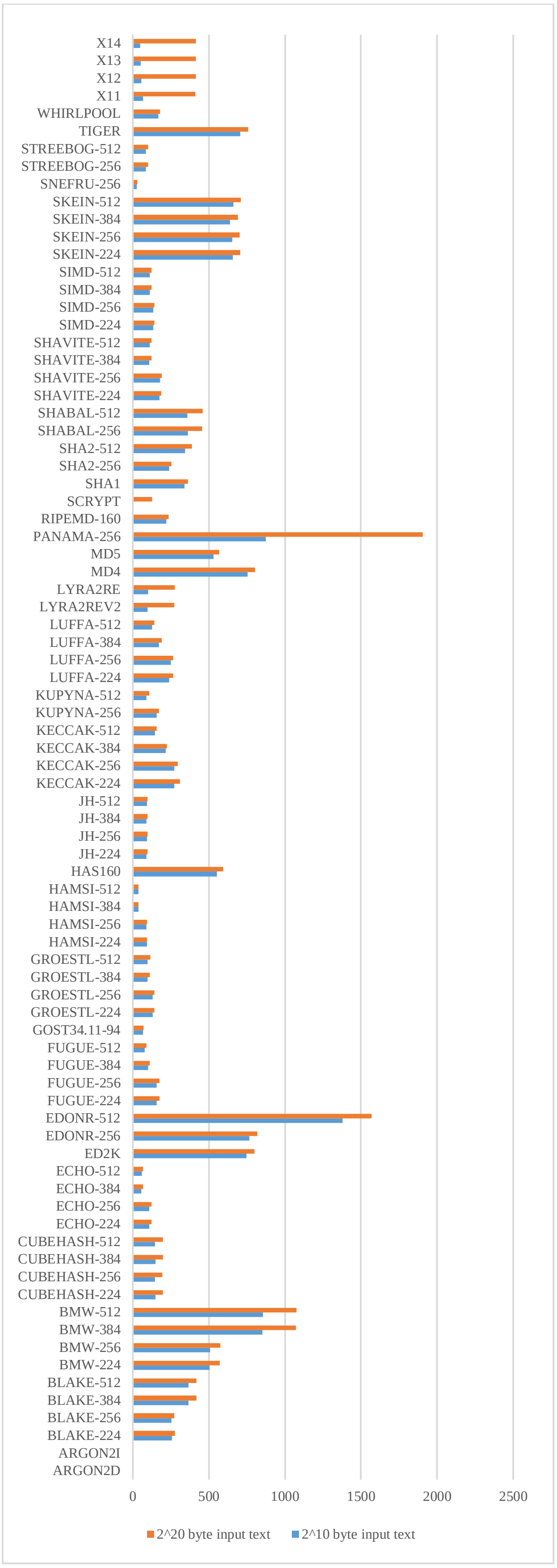

Fig.1. Productivity of hash functions on AMD Ryzen Threadripper 2970WX 24-Core Processor 3.0 Ghz(MB/s)

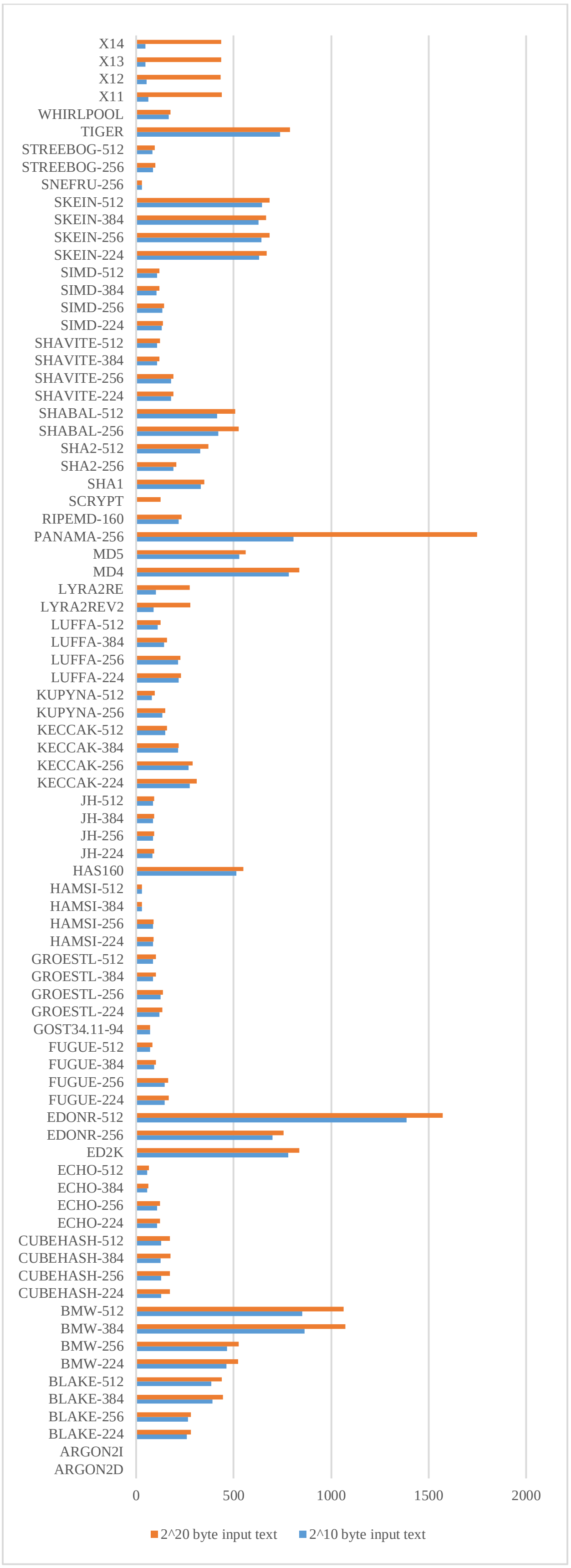

Fig.2. Productivity of hash functions on Intel Corei9-7980 2.60 GHz (MB/s)

4. Performance of Hashing Algorithms on GPU

Performance comparison was produced for the following hashing algorithms: GOST 34.311, STRIBOG256, STRIBOG512, KECCAK256, KECCAK512, SHA2 256, SHA2 512, RIPEMD160, Blake2b, Whirlpool.

We selected the following widespread hardware for analysis the performance of hash algorithms on computer graphics systems:

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

During the comparative analysis, the HashCat program was used [63]. Hashcat is one of the world's fastest password recovery tools. The program was proprietary until 2015, but now released as free software. Versions are available for Linux, macOS and Windows and can be CPU or GPU based. Examples of hashing algorithms supported by hashcats are Microsoft LM, MD4, MD5 hashes, SHA family, Unix Crypt formats, MySQL, and Cisco PIX. Nowadays, Hashcat is actively used for brute-force WPA / WPA2 passwords, crack passwords from MS Office documents, PDF, 7-Zip, RAR, TrueCrypt. The selection of passwords recommended running on the GPU, since the GPU is able to sort out combinations much faster.

Hashcat assumes to use a various types of attacks for effectively crack all kinds of hashes. It could be such attack as Mask attack, Dictionary attack, Permutation attack, Table-lookup attack (only on CPU) etc.

Checking the correctness of the implementation of hashing algorithms in the utility HashCat was performed as follows(for example for Sha2-256):

1. Calculate the hash value of the string "zxcvbn";
2. Sha2-256 ("zxcvbn") =

```
E8F56862D74EF5599AF4EECA73924BFA
44A6773A497AF0C29C48E18729BA6FF0
```

3. As an input to the program HashCat, we use previous value and enable the brute force attack mode:

```
hashcat64.exe -m1400 -a3
E8F56862D74EF5599AF4EECA73924BFA44A6773A4
97AF0C29C48E18729BA6FF0 –O
```

4. As a result, we receive a message about the successful restoration of the prototype;
5. Perform a speed check with the following command:

```
hashcat64.exe -m1400 –b.
```

The results of productivity of hashing algorithms with the use HashCat program are given in Tables 4 and 5.

Table 3 shows the results of the Benchmark,that is, estimates of the hashing speed by consistent hashing of the input information.

Table 4 shows hashing speed for different algorithms for one OpenCL computing core of the applied graphics video card.

Figure 3 illustrates the data from the table 3. From the diagram, we can see that the fastest is the algorithm RIPEMD160. In the second and third places, respectively, are the algorithms SHA2 and Blake2b. Other algorithms are followed.

Figure 4 illustrates the hashing speed for one computing core of the graphics video card. From the diagram we can conclude, that the indicators of performance for the algorithms is almost the same. The algorithms RIPEMD160, SHA2 and Blake2b are take the first places in performance, but their native speed values have changed. Each graphics hardware has a different quantity of cores. The hash speed for computing core of a graphics computer is the highest for Geforce.

Based on the results, we conclude that hashing algorithms RIPEMD160, SHA2 and Blake2b are the fastest algorithms and have the greatest performance. These algorithms are the most common and widely used nowadays. In additional, from the conducted research we can see that different in structure and mathematical transformations hash functions give different acceleration on different devices. Graphical or specialized computing devices give the best results.

The obtained results of comparative analysis make it possible to choose cryptographic hashing functions according to the criteria of performance on different devices and to substantiate their practical application for the construction of decentralized systems such as blockchain.[60-62].

Table 3. Estimates of the hash rate by sequential hashing of the dataset using HashCat utility (KHash/s)

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

Table 4. Estimates of the hash rate which accounts for one OpenCL computing core of the applied graphics video card (KHash/s)

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

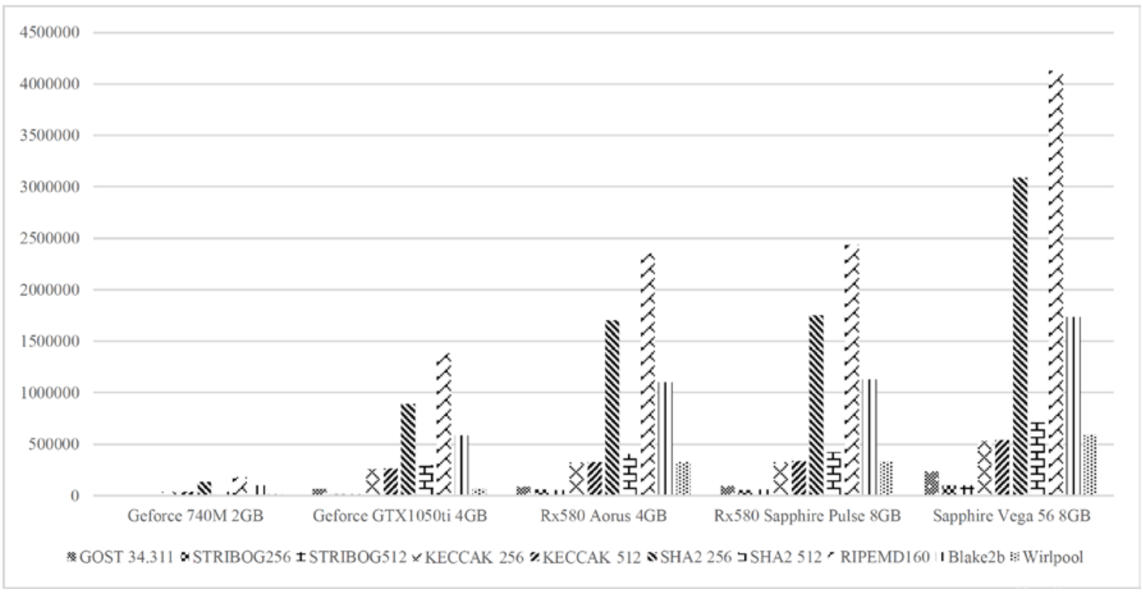

Fig.3. Comparison of productivity of different hashing functions on graphical computing systems

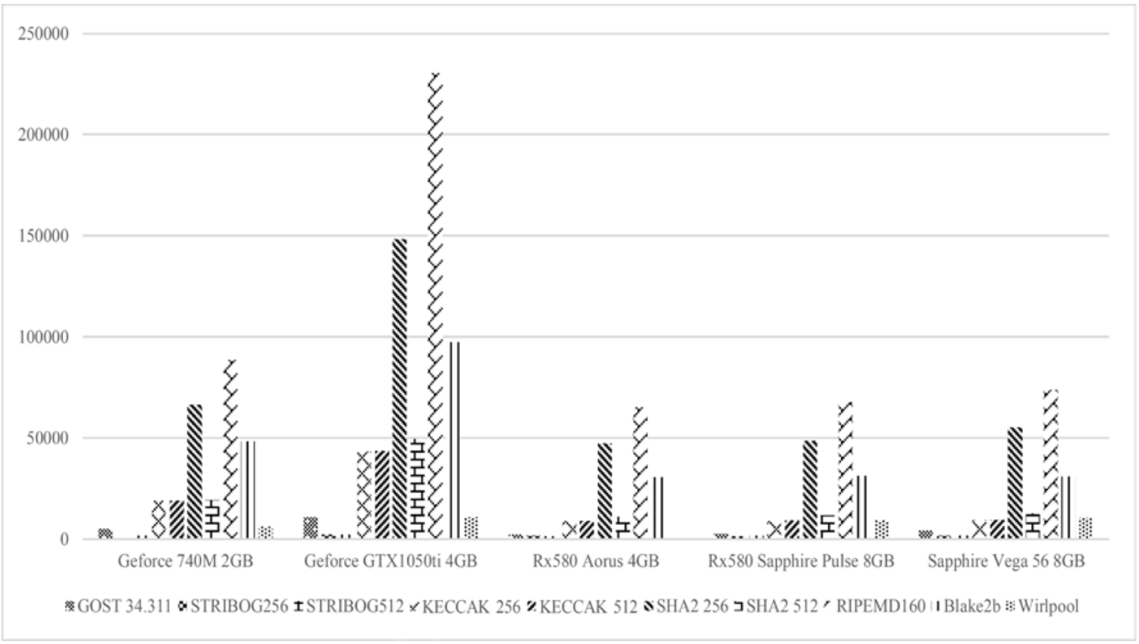

Fig.4. The diagram of the hashing speed for one computing core of the graphics video card

5. Conclusions

Hashing is a procedure for converting an input data array of arbitrary length into an output bit string of fixed length. Hash functions are widely used in cryptography, as well as in many data structures - Hash tables, Bloom filters and Cartesian trees. A good hash function must satisfy two properties: calculate quickly and minimize the number of collisions.

Blockchain technology and the cryptocurrency market demonstrate dynamic development and attract close attention. The innovation of this technology is that transaction information is not stored in a centralized database, but transferred to the computers of all network members who store data locally. In fact, blockchain technology is a universal way to store and process information in almost any field of activity. Hashing ensures the security and reliability of the blockchain system, protects it from collisions and possible hacking. Nowadays there are many hash functions. The actual problem is to choose a hash function for a blockchain system. Protected blockchain systems are built on such properties of hashing functions as unpredictability and irreversibility.

The problem of choosing hash functions compounded by the fact that powerful computers being developed exclusively for mining bitcoins. These computers are able to offer hashes much more often than the average computer, which allows them to find the prototypes much faster. Therefore, the study of hash functions and the formation of recommendations for their choice for the use in modern blockchain systems is certainly important and extremely relevant scientific task.

In our article, we are making a comparison of the productivity of different hash functions. These hash functions are used or promising for use in blockchain systems. For all of the algorithms considered, we made a comparative analyses of performance of hashing algorithms on different computing platforms and with different input parameters. Obtained results are partially summarized in Tables 1-4. In particular, it is found that most distributed network projects use reliable and time-tested cryptographic hashing algorithms (such as KECCAK, SHA2, RIPEMD160, etc.) that have relatively high speed performance. But in recent years, other hash functions have begun to be used to protect ASIC miners, though they may be even faster than KECCAK, SHA2, or RIPEMD160, but have some vulnerabilities to the irreversibility properties. Examples include the use of MD4, EDONR-256, EDONR-512, ED2K, or even the simplest DJB-2 and LOSELOSE features. Due to the simplicity of calculating certain indicators, one cannot neglect the violation of irreversibility properties, especially if they are based on the main advantages of blockchain networks.

The obtained in this study results make it possible to choose hashing functions according to the criteria of performance and use them on different devices for the construction of decentralized systems such as blockchain.

Our studies also include the iterative cryptographic hash function "Kupyna". This function is adopted as the national standard of Ukraine DSTU 7564: 2014 [29]. This research will be useful in the case of development of national blockchain systems.

In our research work we used the most common high-performance hardware and proven software.

The results of this work were reported at international conferences and received positive feedback [55, 56].

Further research can be focused on comparing the statistical properties of hash algorithms. It will also be reasonable to conduct additional research of productivity of hash functions on the most popular computing desktop systems.

In addition, the obtained research results can be useful for various methods of increasing the security of cryptographic systems [64-72].

References

[1] Argon2. By Dmitry Khovratovich. 30 March 2015. https://www.cryptolux.org/index.php/Argon2

[2] Balloon Hashing: A Memory-Hard Function Providing Provable Protection Against Sequential Attacks. 12.05.2017. https://eprint.iacr.org/2016/027.pdf

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

[4] BLAKE2 - fast secure hashing. https://blake2.net/

[5] About Blakecoin. https://blakecoin.org/about-blakecoin/

[6] Blue Midnight Wish. - Trondheim, Norway: Norwegian University of Science and Technology, 2008. - C. 71.

[7] CubeHash specification (2.B.1). http://cubehash.cr.yp.to/submission2/spec.pdf

[8] CubeHash efficiency estimates (2.B.2). http://cubehash.cr.yp.to/submission/estimates.pdf

[9] CubeHash parameter tweak: 16 times faster. http://cubehash.cr.yp.to/submission/tweak.pdf

[10] Single Block Attacks and Statistical Tests on CubeHash. August 21, 2009. http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1986&context=article

[11] Bernstein hash djb2. https://riot-os.org/api/group__sys__hashes__djb2.html

[12] ECHO hash function. https://crypto.orange-labs.fr/echo/

[13] Ed2k-hash. 7 May 2005. https://wiki.anidb.info/w/Ed2k-hash

[14] ed2k-tools. Tools for eDonkey2000 and Overnet. http://ed2k-tools.sourceforge.net/index.shtml

[15] Edon–R, An Infinite Family of Cryptographic Hash Functions. May 2009. https://pdfs.semanticscholar.org/e901/492cbb9d1f8a4365397676da808a9d9415cc.pdf

[16] D. Gligoroski et al., "Cryptographic hash function Edon-R′," Proceedings of the 1st International Workshop on Security and Communication Networks, Trondheim, 2009, pp. 1-9.

[17] The Ethereum Wiki. https://github.com/ethereum/wiki

[18] Dagger Hashimoto. Https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto

[19] Ethash Design Rationale. Https://github.com/ethereum/wiki/wiki/Ethash-Design-Rationale

[20] Hash Function Fugue. https://researcher.watson.ibm.com/researcher/view_group.php?id=3302

[21] GOST R 34.11-2012. Information technology. Cryptographic information security. Hash function. Date introduced 2013-01-01. http://docs.cntd.ru/document/gost-r-34-11-2012

[22] Grøstl – a SHA-3 candidate. Thomsen. March 2, 2011. http://www.groestl.info/Groestl.pdf

[23] Grøstl – a SHA-3 candidate. http://www.groestl.info/team.html

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

[25] Telecommunications Technology Association. Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160). TTAS.KO-12.0011/R1, December 2000.https://www.tta.or.kr/include/Download.jsp?filename=stnfile/TTA-0072.pdf

[26] JH. https://ehash.iaik.tugraz.at/wiki/JH

[27] The sponge and duplex constructions. Keerhttps://keccak.team/sponge_duplex.html

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

[29] A New Standard of Ukraine: The Kupyna Hash Function. https://eprint.iacr.org/2015/885.pdf

[30] The C Programming Language by Brian W. Kernighan (1978-02-22) Paperback, Prentice Hall, 178 p.

[31] The Hash Function Family Luffa (Round 2 Archive). http://www.hitachi.com/rd/yrl/crypto/luffa/index.html

[32] Lyra2RE – A new PoW algorithm for an ASIC-free future. By Vertcoin Developers. https://cryptorating.eu/whitepapers/Vertcoin/Vertcoin_Lyra2RE_Paper_11292014.pdf

[33] Lyra2REv2. https://en.bitcoinwiki.org/wiki/Lyra2REv2

[34] MD4 Message Digest Algorithm. RFC 1186. Last updated 2013-03-02. https://datatracker.ietf.org/doc/rfc1186/

[35] The MD5 Message-Digest Algorithm. https://www.ietf.org/rfc/rfc1321.txt

[36] The Panama Cryptographic Function. http://www.drdobbs.com/security/the-panama-cryptographic-function/184410745

[37] Overview of the ProgPOW algorithm for GPU mining. Alexander Markov. October 10, 2018. https://miningbitcoinguide.com/mining/sposoby/progpow

[38] Company Coinmarket. https://coinmarket.news/2019/01/20/progpow-obzor-svezhih-majnerov-dlya-novogo-algoritma/

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

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

[41] Colin Percival. Stronger key derivation via sequential memory-hard functions. 2009. https://en.bitcoinwiki.org/wiki/Scrypt http://www.tarsnap.com/scrypt/scrypt.pdf

[42] US Secure Hash Algorithm 1 (SHA1). By P. Jones. September 2001. https://www.ietf.org/rfc/rfc3174.txt

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

[44] SHA-256 Coins. https://cryptorival.com/algorithms/sha256/

[45] Shabal, a Submission to NIST’s Cryptographic Hash Algorithm Competition. 28.10.2008 https://www.cs.rit.edu/~ark/20090927/Round2Candidates/Shabal.pdf

[46] The SHAvite-3 Hash Function. By Eli Biham and Orr Dunkelman. http://www.cs.technion.ac.il/~orrd/SHAvite-3/Spec.31.10.08.pdf
[47] The Skein Hash Function Family Version 1.3 - 1 Oct 2010. http://www.skein-hash.info/sites/default/files/skein1.3.pdf
[48] Cryptohash: snefru256. https://snefru256.cryptohash.net/
[49] GOST R 34.11-2012. http://docs.cntd.ru/document/gost-r-34-11-2012
[50] IZZZIO. https://en.bitcoinwiki.org/wiki/IZZZIO
[51] A Tiger Hash Implementation for C#. 10 Mar 2012. https://www.codeproject.com/Articles/149061/A-Tiger-Hash-Implementation-for-C
[52] LARC - Laboratório de Arquitetura e Redes de Computadores. http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html
[53] The X13 algorithm for mining on GPUs. Alexander Markov. May 28, 2018. https://miningbitcoinguide.com/mining/sposoby/x13
[54] Cryptocurrency mining algorithms - table 2019 and summary. https://mining-cryptocurrency.ru/algoritmy-kriptovalyut/
[55] Kuznetsov A, Lutsenko M, Kuznetsova K, Martyniuk O, Babenko V, Perevozova I. Statistical Testing of Blockchain Hash Algorithms. In: Fedushko S, Gnatyuk S, Peleshchyshyn A, Hu Z, Odarchenko R, Korobiichuk I, editors. Proceedings of the International Workshop on Conflict Management in Global Information Networks (CMiGIN 2019) co-located with 1st International Conference on Cyber Hygiene and Conflict Management in Global Information Networks (CyberConf 2019), Lviv, Ukraine, November 29, 2019 [Internet]. CEUR-WS.org; 2019 [cited 2020 Jul 2]. p. 67–79.
[56] Kuznetsov A, Shekhanin K, Kolhatin A, Kovalchuk D, Babenko V, Perevozova I. Performance of Hash Algorithms on GPUs for Use in Blockchain. In: 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT). 2019. p. 166–70.
[57] Poluyanenko N, Kuznetsov A, Lazareva E, Marakushyn A. Extrapolation to calculate the probability of a double spending attack. In: Subbotin S, editor. Proceedings of The Third International Workshop on Computer Modeling and Intelligent Systems (CMIS-2020), Zaporizhzhia, Ukraine, April 27-May 1, 2020 [Internet]. CEUR-WS.org: 2020 [cited 2020 Jul 2]. p. 610–620.
[58] Myronets I., Shkrebtii A. Cryptographic algorithms and features of their use in blockchain systems // Ukrainian Scientific Journal of Information Security, 2019, vol. 25, issue 2, pp. 104-109.
[59] B. Seok, J. Park, J.H. Park. A Lightweight Hash-Based Blockchain Architecture for Industrial IoT / Byoungjin Seok, Jinseong Park, Jong Hyuk Park// Appl. Sci. 2019, 9, 3740; doi:10.3390/app9183740.
[60] Koibichuk V. V. Features of application of cryptographic algorithms during digital money transfer by blockchain technologies / V. V. Koibichuk // Black Sea Economic Studies. - 2020. - Vol. 49. - P. 201-204. - Access mode: http://nbuv.gov.ua/UJRN/bses_2020_49_35
[61] Semerenko V., Коробов А. Blockchain technology implementation based on stream hash functions. Conference: Proceeding of the XLIX scientific and technical conference of Vinnytsia National Technical University (VNTU) Units,At: UKRAINE, Vinnytsia, April 2020
[62] P.V. Kravchuk, I.D. Gorbenko, A.I. Pushkaryov. Analysis of application of hash function in blockchain technology. // Applied radioelectronics, 2018, Vol 17, № 3, 4, pp. 147-151.
[63] Hashcat. Advanced Password Recovery. Access mode: http://hashcat.net/hashcat/
[64] Raihana Syahirah Abdullah, Faizal M.A.,"Block Chain: Cryptographic Method in Fourth Industrial Revolution", International Journal of Computer Network and Information Security, Vol.10, No.11, pp.9-17, 2018.
[65] Hany F. Atlam, Ahmed Alenezi, Madini O. Alassafi, Gary B. Wills, "Blockchain with Internet of Things: Benefits, Challenges, and Future Directions", International Journal of Intelligent Systems and Applications, Vol.10, No.6, pp.40-48, 2018.
[66] Dipti Pawade, Sagar Jape, Rahul Balasubramanian, Mihir Kulkarni, Avani Sakhapara,"Distributed Ledger Management for an Organization using Blockchains", International Journal of Education and Management Engineering, Vol.8, No.3, pp.1-13, 2018.
[67] M. Iavich, S. Gnatyuk, E. Jintcharadze, Y. Polishchuk, A. Fesenko and A. Abisheva, “Comparison and Hybrid Implementation of Blowfish, Twofish and RSA Cryptosystems”, *Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering* (UKRCON), Lviv, Ukraine, 2019, pp. 970-974.
[68] A. Kuznetsov, Y. Gorbenko, A. Andrushkevych and I. Belozersev, "Analysis of block symmetric algorithms from international standard of lightweight cryptography ISO/IEC 29192-2," *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T),* Kharkov, 2017, pp. 203-206. DOI: 10.1109/INFOCOMMST.2017.8246380.
[69] 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
[70] Gnatyuk S., Kinzeryavyy V., Kyrychenko K., Yubuzova Kh., Aleksander M., Odarchenko R. “Secure Hash Function Constructing for Future Communication Systems and Networks”, *Advances in Intelligent Systems and Computing*, Vol. 902, pp. 561-569, 2020.
[71] 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
[72] S. Gnatyuk, M. Kovtun, V. Kovtun, A. Okhrimenko, “Development of a search method of birationally equivalent binary edwards curves for binary Weierstrass curves from DSTU 4145-2002”, *Proceedings of 2nd Intern. Scientific-Practical Conf. on the Problems of Infocommunications. Science and Technology* (PIC S&T 2015), Kharkiv, Ukraine, October 13-15, 2015, pp. 5-8.

Authors' Profiles

Alexandr Kuznetsov
Doctor of Sciences (Engineering), Full Professor, Academician of the Academy of Applied Radioelectronics Sciences, Professor of the Department security information systems and technologies of the V. N. Karazin Kharkiv National University, Ukraine. Areas of scientific interests: applied cryptology and coding theory.
Email: kuznetsov@karazin.ua

Inna Oleshko
Candidate of Technical Sciences, Senior Lecturer of the Department of information technology security, Kharkiv National University of Radio Electronics. Areas of scientific interests: biometric authentication, programming, blockchain.
Email: inna.oleshko@nure.ua

Vladyslav Tymchenko
Graduate student of the Department security information systems and technologies of the V. N. Karazin Kharkiv National University, Ukraine. Areas of scientific interests: post-quantum cryptography and authentication, security information systems and technologies.
Email: tvlad.tyma@gmail.com

Konstantin Lisitsky
Graduate student of the Department security information systems and technologies of the V. N. Karazin Kharkiv National University, Ukraine. Areas of scientific interests: applied cryptography and cybersecurity, block ciphers analysis and development.
Email: konstantin.lisickiy@mail.ru

Mariia Rodinko
Graduate student of the Department security information systems and technologies of the V. N. Karazin Kharkiv National University, Ukraine. Areas of scientific interests: applied cryptography and cybersecurity, block ciphers analysis and development.
Email: m.rodinko@gmail.com

Andrii Kolhatin
Graduate student of the Department security information systems and technologies of the V. N. Karazin Kharkiv National University, Ukraine. Areas of scientific interests: cryptology and steganography, information hiding and steganographic analysis.
Email: kolgatin-a@yandex.ua

How to cite this paper: Alexandr Kuznetsov, Inna Oleshko, Vladyslav Tymchenko, Konstantin Lisitsky, Mariia Rodinko, Andrii Kolhatin, "Performance Analysis of Cryptographic Hash Functions Suitable for Use in Blockchain", International Journal of Computer Network and Information Security(IJCNIS), Vol.13, No.2, pp.1-15, 2021. DOI: 10.5815/ijcnis.2021.02.01