Statistical Testing of Blockchain Hash Algorithms

Alexandr Kuznetsov [1 [0000-0003-2331-6326]], Maria Lutsenko [1 [0000-0003-2075-5796]], Kateryna Kuznetsova [1 [0000-0002-5605-9293]], Olena Martyniuk [2 [0000-0002-0377-7881]], Vitalina Babenko [1 [0000-0002-4816-4579]] and Iryna Perevozova [3[0000-0002-3878-802X]]

[1] V. N. Karazin Kharkiv National University, Kharkiv, Ukraine
kuznetsov@karazin.ua, lutsenko.maria.kh@gmail.com, kate.kuznetsova.2000@gmail.com, vitalinababenko@karazin.ua
[2] International Humanitarian University, Odessa, Ukraine, emartynuk2017@gmail.com
[3] Ivano-Frankivsk National Technical University of Oil and Gas, Ivano-Frankivsk, Ukraine, perevozova@ukr.net

Abstract. Various methods are used for statistical testing of cryptographic algorithms, for example, NIST STS (A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications) and DIEHARD (Diehard Battery of Tests of Randomness). Tests consists of verification the hypothesis of randomness for sequences generated at the output of a cryptographic algorithm (for example, a keys generator, encryption algorithms, a hash function, etc.). In this paper, we use the NIST STS technique and study the statistical properties of the most common hashing functions that are used or can be used in modern blockchain networks. In particular, hashing algorithms are considered which specified in national and international standards, as well as little-known hash functions that were developed for limited use in specific applications. Thus, in this paper, we consider the most common hash functions used in more than 90% of blockchain networks. The research results are given as average by testing data of 100 sequences of 10^8 bytes long, which means that is, the size of the statistical sample for each algorithm was 10^{10} bytes. Moreover, each test (for each of the 100 sequences) was considered as an independent observation. In addition, the article presents statistical portraits for each algorithm under study (diagrams of the numbers of passing each test).

Keywords: statistical testing, hashing algorithms, blockchain technology.

1 Introduction

In this work, statistical studies of the output sequences of cryptographic hash functions are performed, when this functions processing excessive input data. In this case, input data are formed using a regular counter.

The NIST STS (A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications) methodology, which is recommended by The National Institute of Standards and Technology, USA for the study of random and pseudorandom number generators for cryptographic applications [1, 2]. Both

world-famous hashing functions (standardized at the international and/or national levels) [3-7] and little-known algorithms [8, 9], which are designed for use only in certain applications (e.g., in decentralized blockchain systems [10-14]). Specifically, the statistical security results of the following algorithms are given: GOST 34.311, STRIBOG256, STRIBOG512, BALLOON 32, BALLOON 64, BLAKE256, BLAKE512, BMW, CUBEHASH, DJB-2, DJB-2 XOR, ECHO, FUGUE 224, FUGUE 256, FUGUE 384, FUGUE 512, GROESTL 256, GROESTL 512, HAMSI 224, HAMSI 256, HAMSI 384, HAMSI 512, J-H, KECCAK 256, KECCAK 512, LOSELOSE, LUFFA, PROGPOW, RANDOMX, RIPEMD160, SCRYPT 1024, SCRYPT 16384, SHA2 256, SHA2 512, SHABAL 224, SHABAL 256, SHABAL 384, SHABAL 512, SHAVITE, SIMD, SKEIN, WHIRLPOOL, X11.

2 Research methodology and results

The NIST STS statistical test suite recommended by The National Institute of Standards and Technology, USA [1, 2] was used to conduct studies of various hashing algorithms by statistical security criteria's. Methods of statistical testing and processing algorithm of the obtained results are given in [15, 16].

The NIST Statistical Test Suite was developed during the AES competition for the study of random or pseudorandom number generators and is the most common tool for assessing the statistical security of cryptographic primitives. The use of this package allows us to estimate how closely the crypto algorithms under study approximate the generators of "random" sequences, that is, with a high probability to confirm whether the generated sequence is statistically secure. The order of testing of a single binary sequence *S* is as follows:

- the null hypothesis H_0 is advanced - the assumption that this binary sequence S is random;
- the test statistic *c(S)* is calculated according to the sequence S;
- the probability function $P = f\left(c\left(S\right)\right)$ is calculated using a special function and test statistics;
- the probability value *P* is compared with the threshold value $\alpha \in [0,96; 0,99]$. If so $P \geq \alpha$, the hypothesis H_0 is accepted. Otherwise, an alternative hypothesis is accepted.

The test suite contains 15 statistical tests, but in fact, depending on the input parameters, 188 probability values of *P* are calculated, which can be considered as the result of individual tests.

Frequency (Monobits) Test. Aims to determine the relation between zeros and ones in a binary sequence of a certain length. For a truly random binary sequence, the number of zeros and ones is almost the same. The test estimates how close the unit is to 0.5.

Test for Frequency Within a Block. The essence of the test is to determine the fraction of ones inside the block with a length of *m* bits, i.e. it is necessary to find out

whether the repetition rate of ones in the block with a length of *m* bits is approximately equal $m/2$, as might be assumed in the case of a random sequence.

Runs Test. This test searches for rows, that is, continuous sequences of identical bits. A series (runs) of length *k* bits consists of *k* absolutely identical bits, beginning and ending with a bit containing the opposite value. In this test, you need to find out if the number of such rows really matches their number in random order. In particular, it is determined whether the ones and zeros in the initial sequence quickly or slowly alternate.

Test for The Longest Run of Ones in a Block. This test determines the longest row of ones inside the block with a length of *m* bits. It is necessary to find out whether the length of such a row actually meets the expectation of the length of the longest row of ones in the case of a completely random sequence.

Random Binary Matrix Rank Test. Here, we calculate the rank of non-continuous sub-matrices constructed from the initial binary sequence. The purpose of this test is to test for linear dependence of fixed length substrings that make up the initial sequence.

Discrete Fourier Transform (Spectral) Test. The essence of the test is to estimate the peak height of the discrete Fourier transform of the initial sequence. The purpose is to identify periodic properties of the input sequence, for example, closely spaced repetitive sections. The idea is that the number of peaks in excess of the 95% amplitude threshold is much greater than 5%.

Non-Overlapping (Aperiodic) Template Matching Test. This test calculates the number of predefined templates found in the original sequence. It is necessary to identify random or pseudorandom number generators that form too often non-periodic patterns. As in Overlapping Template Matching Test, a window with a length of *m* bits is also used to search for specific patterns with a length of *m* bits. If no pattern is found, the window shifts one bit. If a pattern is found, then the window moves to the bit that follows the pattern found, and the search continues.

Overlapping (Periodic) Template Matching Test. The essence of this test is to calculate the number of predefined templates that were found in the original sequence. As in Non-Overlapping Template Matching Test, a window with a length of *m* bits is also used to search for specific patterns with a length of *m* bits. The search itself is conducted in a similar way. If no pattern is found, the window shifts one bit. The difference between this test and previous test is that when the pattern is found, the window moves only one bit forward, and then the search continues.

Maurer's Universal Statistical Test. In here determines the number of bits between the same patterns in the initial sequence (a measure that is directly related to the length of the compressed sequence). It is necessary to find out whether this sequence can be significantly compressed without loss of information. If this can be done, then it is not truly random.

Linear Complexity Test. The test is based on the principle of the linear shift register feedback. You need to find out if the input sequence is complex enough to be considered completely random. Absolutely random sequences are characterized by long linear shift registers. If such a register is too short, then it is assumed that the sequence is not completely random.

Serial Test. This test is to calculate the frequency of all possible overlaps of the m bit length patterns at the initial bit sequence. The purpose is to determine whether the number of occurrences of overlapping $2m$ patterns by the length of the m bits is approximately the same as in the case of an absolutely random input bit sequence. The latter is known to be monotonous, that is, each pattern with a length of m bits appears in a sequence with equal probability. It is worth noting that when $m=1$, so the periodicity test goes into the frequency bit test.

Approximate Entropy Test. As in the periodicity test, this test focuses on calculating the frequency of all possible overlaps of the m bit length patterns at the initial bit sequence. It is necessary to compare the overlap frequencies of two consecutive blocks of the initial sequence with the lengths m and $m+1$with the overlap frequencies of similar blocks in a completely random sequence.

Cumulative Sum (Cusum) Test. The test is the maximum deviation (from zero) at an arbitrary bypass determined by the cumulative sum of the given digits $(-1,+1)$ in the sequence. It is necessary to determine whether the cumulative sum of the partial sequences occurring in the input sequence is too large or too small compared to the expected behavior of such a sum for a completely random input sequence. Thus, the cumulative amount can be regarded as an arbitrary bypass. For a random sequence, the deviations from the bypass should be near zero.

Random Excursions Test. The essence of this test is to calculate the number of cycles that have strictly *k* excursions with an arbitrary bypass of the cumulative sum. The arbitrary bypass of a cumulative sum begins with partial sums after the sequence $(0,1)$ is translated into the corresponding sequence $(-1,+1)$. An arbitrary bypass cycle consists of a series of single-length steps performed in random order. In addition, such a bypass begins and ends on the same element. The purpose of this test is to determine whether the number of visits to a particular state within a cycle differs from a similar number in the case of a completely random input sequence. In fact, this test is a set consisting of eight tests that are conducted for each of the eight cycle states: -4, -3, -2, -1 and +1, +2, +3, +4.

Random Excursions Variant Test. This test calculates the total number of excursions to a given condition when you randomly bypass the cumulative sum. The purpose is to determine deviations from the expected number of visits to different states at random bypass. In fact, this test consists of 18 tests for each state: -9, -8, ..., -1 and +1, +2, ..., +9.

Thus, as a result of binary sequence testing, a vector $P=\{P_1,P_2...P_{188}\}$ of probability values P_j is formed. The analysis of the components P_j of this vector allow us to point to specific defects in the randomness of the tested sequence.

Passing each of the 15 statistical tests is an important criterion for evaluating a pseudorandom generator. Therefore, not even matching one or more criteria means that the stream cannot withstand cryptanalysis at a high level. If, on the other hand, the generator passes all the tests, this does not indicate the security of the generator, since such tests do not take into account the features of the actual design of the generator.

The accumulated experience of statistical testing shows that the number of tests passed by the generator being tested depends directly on the selected cryptographic algorithm output sequence. To ensure the reliability of the results of statistical testing in the work [15-18], it is proposed to evaluate the mathematical expectation of the number of tests passed X_i by the investigated generator (crypto algorithm), considering each i test as a single observation (experience), i.e. as a specific implementation of some random variable X.

When conducting statistical surveys, 100 sequences with a length of 10^8 bytes were generated for each algorithm, i.e. the size of the statistical sample for each algorithm reached 10^{10} bytes. Each testing (for each of the 100 sequences) was considered as an independent observation. Table 1 summarizes the statistical test results for each algorithm studied.

Table 1. The statistical testing results of hashing algorithms

Algorithm Name	M099	D099	S099	P099	M096	D096	S096	P096	MIN
GOST 34.311	132.90	43.93	6.62	1	186.83	1.46	1.20	1	182
STRIBOG256	131.92	55.93	7.47	1	186.70	1.81	1.34	1	181
STRIBOG512	131.64	48.99	6.99	1	186.76	1.88	1.37	1	182
BALLOON 32	134.20	60.56	7.78	1	187.10	1.09	1.04	1	185
BALLOON 64	126.50	0.25	0.50	1	183.00	1.00	1.00	1	182
BLAKE256	133.31	55.13	7.42	1	186.75	1.90	1.38	1	183
BLAKE512	132.65	55.74	7.46	1	186.73	1.59	1.26	1	183
BMW	132.39	48.99	6.99	1	186.92	1.55	1.24	1	182
CUBEHASH	131.22	55.65	7.46	1	186.80	1.44	1.2	1	183
DJB-2	8.92	1.21	1.10	1	11.64	0.29	0.53	1	10
DJB-2 XOR	2.99	2.16	1.47	1	4.94	1.69	1.30	1	0
ECHO	131.96	53.85	7.33	1	186.51	2.16	1.47	1	182
FUGUE 224	133.07	45.78	6.76	1	186.61	2.11	1.45	1	180
FUGUE 256	132.42	43.74	6.61	1	186.78	2.25	1.50	1	180
FUGUE 384	131.03	57.22	7.56	1	186.66	1.78	1.33	1	182
FUGUE 512	133.02	62.31	7.89	1	186.76	2.14	1.46	1	180
GROESTL 256	133.23	56.01	7.48	1	186.72	2.14	1.46	1	181
GROESTL 512	133.01	58.58	7.65	1	187.14	0.90	0.94	1	184
HAMSI 224	132.84	53.65	7.32	1	186.66	2.36	1.53	1	112
HAMSI 256	131.71	51.42	7.17	1	186.87	1.87	1.36	1	181
HAMSI 384	132.86	51.26	7.15	1	187.19	1.21	1.10	1	182
HAMSI 512	132.17	51.16	7.15	1	186.45	2.68	1.63	1	179
J-H	131.70	71.63	8.46	1	186.69	2.43	1.56	1	180
KECCAK 256	131.27	58.79	7.66	1	186.52	2.70	1.64	1	181
KECCAK 512	132.40	48.12	6.93	1	186.78	1.41	1.18	1	182
LOSELOSE	16.00	0.00	0.00	1	16.00	0.00	0.00	1	16
LUFFA	133.11	47.07	6.86	1	186.71	1.50	1.22	1	183
PROGPOW	130.00	0.00	0.00	1	188.00	0.00	0.00	1	188
RANDOMX	0.00	0.00	0.00	1	0.00	0.00	0.00	1	0
RIPEMD160	84.79	84.04	9.16	1	132.11	80.95	8.99	1	105
SCRYPT 1024	133.80	68.36	8.26	1	187.10	1.49	1.22	1	184
SCRYPT 16384	140.00	0.00	0.00	1	185.00	0.00	0.00	1	185

Algorithm Name	M099	D099	S099	P099	M096	D096	S096	P096	MIN
SHA2 256	132.70	57.39	7.57	1	186.74	1.67	1.29	1	182
SHA2 512	133.01	51.78	7.19	1	186.87	1.99	1.41	1	182
SHABAL 224	132.76	50.12	7.08	1	186.67	2.52	1.58	1	180
SHABAL 256	133.18	61.72	7.85	1	186.61	2.39	1.54	1	115
SHABAL 384	131.63	45.61	6.75	1	186.54	2.00	1.41	1	180
SHABAL 512	132.81	45.41	6.73	1	186.87	1.57	1.25	1	182
SHAVITE	132.01	53.72	7.33	1	186.9	1.53	1.23	1	182
SIMD	133.09	39.54	6.28	1	186.85	1.58	1.25	1	182
SKEIN	131.90	46.97	6.85	1	186.71	1.26	1.12	1	184
WHIRLPOOL	132.29	47.90	6.92	1	186.78	1.59	1.26	1	182
X11	132.46	46.48	6.81	1	186.80	1.28	1.13	1	184
X12	133.10	21.29	4.61	1	186.20	2.36	1.53	1	183
X13	137.00	74.56	8.63	1	186.90	0.69	0.83	1	185
X14	131.40	24.44	4.94	1	186.90	0.69	0.83	1	123
X15	130.10	31.49	5.61	1	186.20	4.56	2.13	1	182
X16	0.90	0.09	0.30	1	1.00	0.00	0.00	1	1
X17	3.20	0.36	0.60	1	5.80	0.16	0.40	1	5

Table 1 provides the following data:

- "M096" and "M099" – estimates of expected value (the sample mean) of the number of passed tests by criterion $P_j \geq 0,96$ and criterion $P_j \geq 0,99$, respectively;
- "D096" and "D099" ("S096" and "S099") – estimates of the statistical dispersion (standard deviation) of the results of testing the number of statistical tests completed by the criterion $P_j \geq 0,96$ and criterion $P_j \geq 0,99$, accordingly;
- "P099" – the confidence value for the number of statistical tests completed by criterion $P_j \geq 0,99$ and accuracy $\varepsilon = 2$;
- "P096" – the value of the confidence probability for the number of statistical tests passed by criterion $P_j \geq 0,96$ and accuracy $\varepsilon = 1$;
- "Min096" – the minimum values of the number of statistical tests passed by the criterion $P_j \geq 0,96$.

The results of statistical studies (statistical portraits) of hashing algorithms are shown in Fig. 1-40. On the abscissa scale the statistical test number (from 1 to 188) is given, on the ordinate scale the fraction of passing of the corresponding test is given.

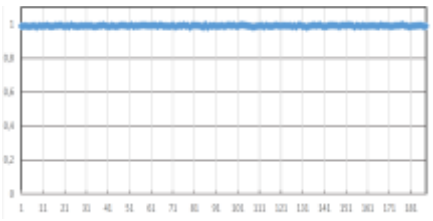

Fig. 1. BALLOON32

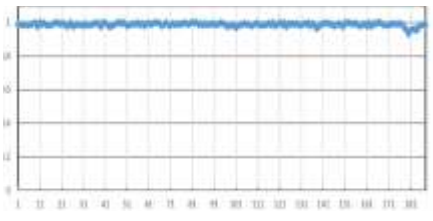

Fig. 2. BALLOON64

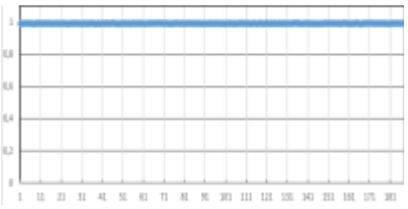

Fig. 3. BLAKE 256

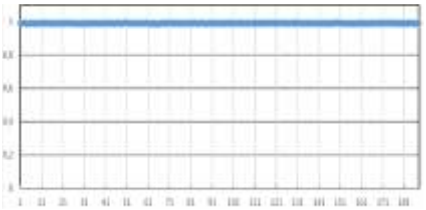

Fig. 4. BLAKE 512

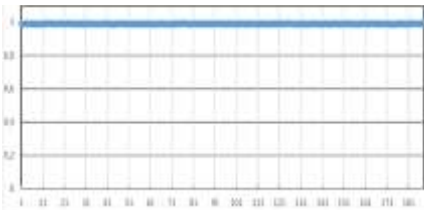

Fig. 5. BMW

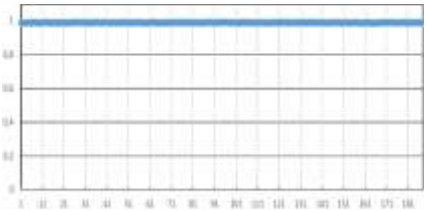

Fig. 6. CUBEHASH

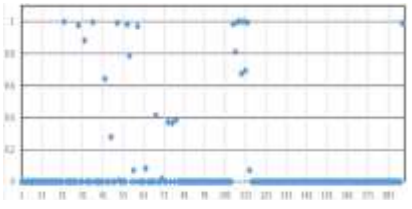

Fig. 7. DJB-2

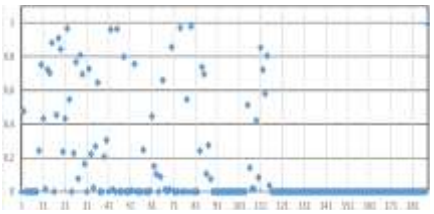

Fig. 8. DJB-2XOR

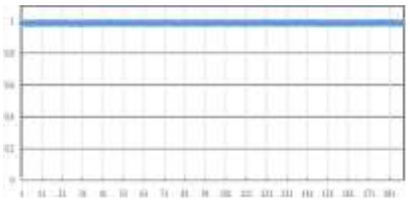

Fig. 9. ECHO

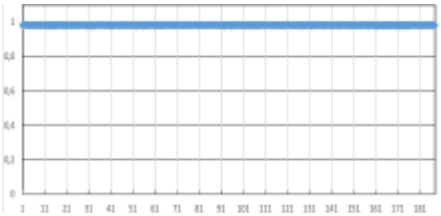

Fig. 10. KECCAK 256

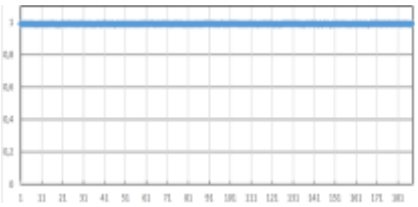

Fig. 11. KECCAK 512

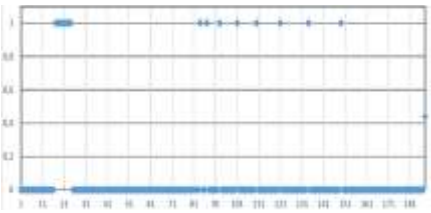

Fig. 12. LOSELOSE

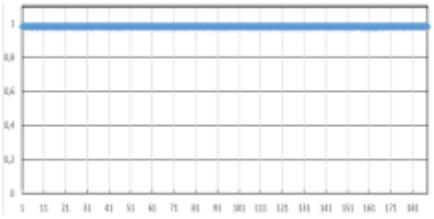

Fig. 13. LUFFA

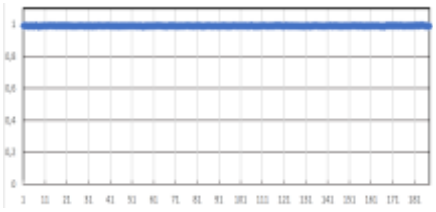

Fig. 18. GOST_256

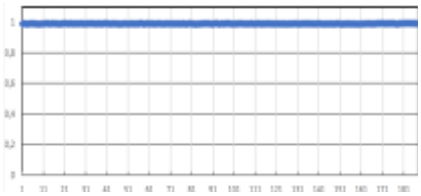

Fig. 14. FUGUE224

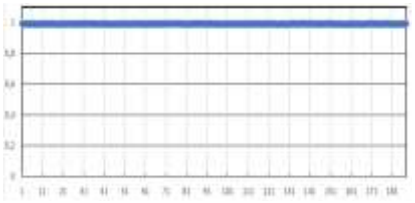

Fig. 19. Stribog_512

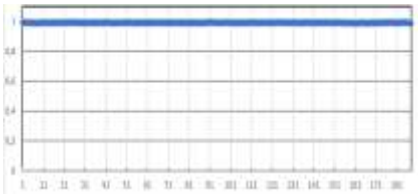

Fig. 15. FUGUE256

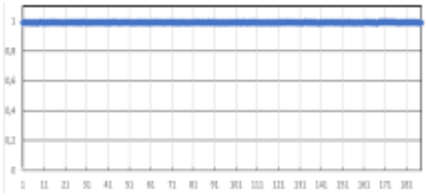

Fig. 20. WHIRLPOOL512

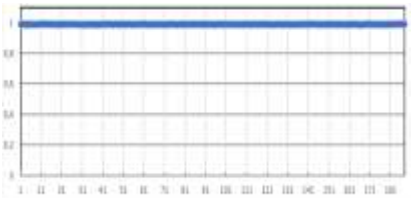

Fig. 16. FUGUE384

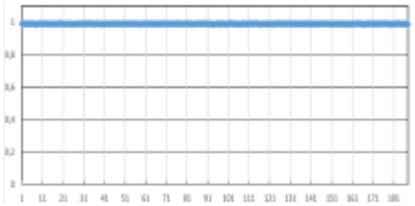

Fig. 21. GROESTL 256

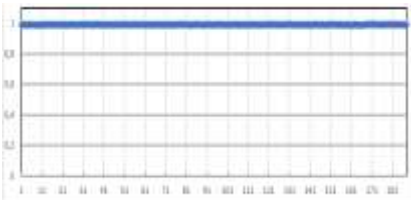

Fig. 17. FUGUE512

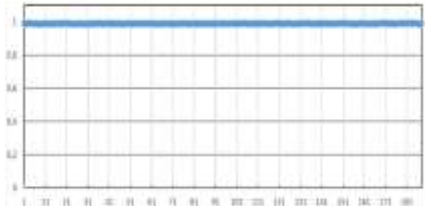

Fig. 22. GROESTL 512

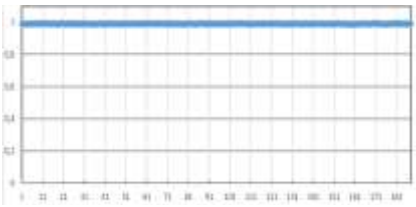

Fig. 23. HAMSI 224

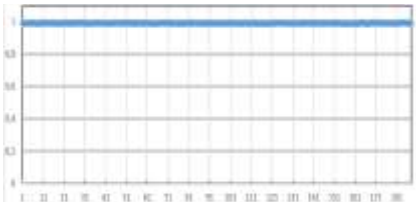

Fig. 24. HAMSI 256

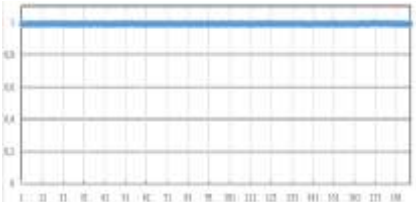

Fig. 25. HAMSI 384

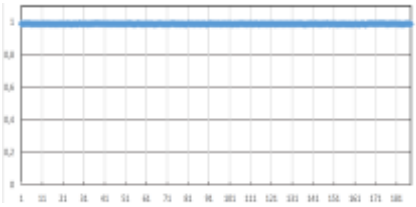

Fig. 26. HAMSI 512

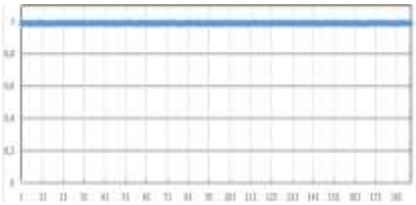

Fig. 27. J-H

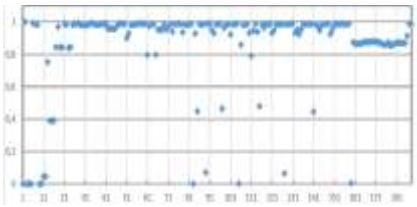

Fig. 28. RIPEMD160

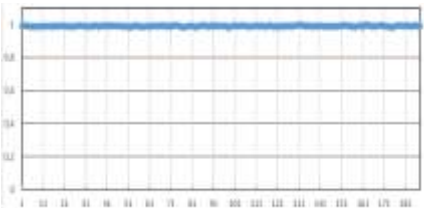

Fig. 29. SCRYPT 1024

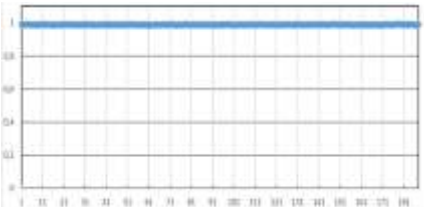

Fig. 30. SHA2 256

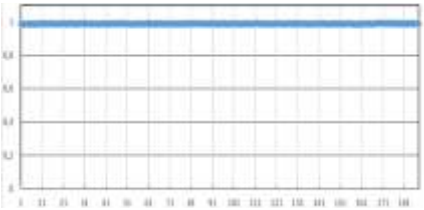

Fig. 31. SHA2 512

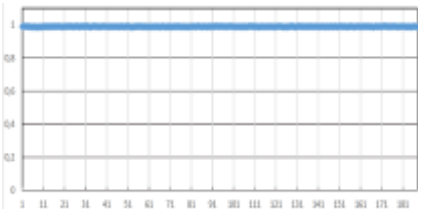

Fig. 32. SHABAL 224

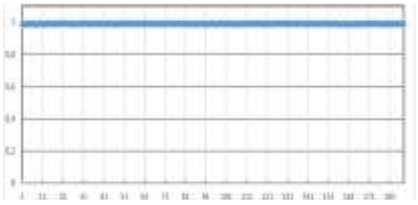

Fig. 33. SHABAL 256

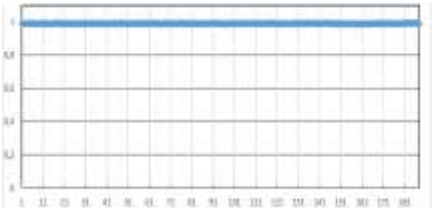

Fig. 34. SHABAL 384

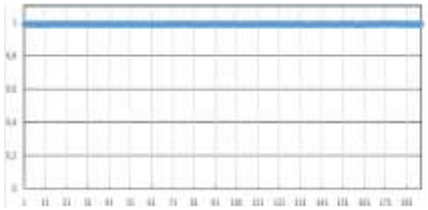

Fig. 35. SHABAL 512

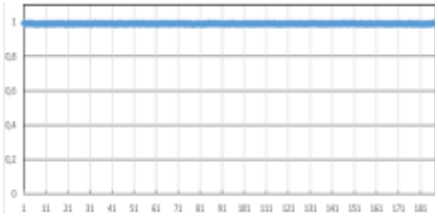

Fig. 39. STREEBOG 256

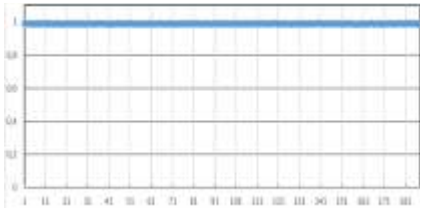

Fig. 36. SHAVITE

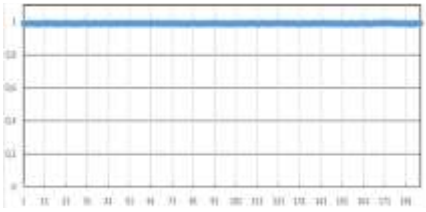

Fig. 37. SIMD

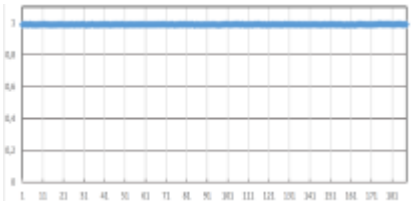

Fig. 38. SKEIN

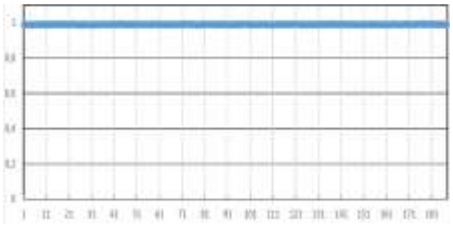

Fig. 40. X11

The results of statistical studies indicate that certain high-speed algorithms cannot be applied in cryptographic applications. This applies, for example, to the algorithms DJB-2, LOSELOSE, and others, because these algorithms, in fact, do not compute a cryptographic checksum. However, most of the cryptographic hashing algorithms, that have been studied, have shown high statistical properties and have high rates of indistinguishability criterion with truly random sequence.

3 Conclusions

Hashing functions are a complex and very important cryptographic primitive that is used in almost all mechanisms and protocols of cryptographic security of information (password generation, encryption, pseudorandom sequence generation, electronic signature generation, etc.). In recent years, the use of hashing has expanded significantly. In particular, with the advent and rapid spread of decentralized distributed systems based on so-called "linked lists" (blockchain) technology, there was an urgent need for fast, safe and reliable hashing functions, because of their unpredictable and irreversible features secure blockchain chains are being built. The task of choosing a hash function is much more complicated due to the proliferation of specialized computants that are being developed and practically used to look for prototypes of preformed hash values (ASIC-mining). By investing in the acquisition of ASICs, individual players can be deliberately advantaged compared to other blockchain users and can, therefore, cause non-trust and compromise of decentralized technologies (e.g., different cryptocurrencies, distributed storage, smart- contracts, etc.). Therefore, the study of the properties of modern hashing algorithms and the rationale for their recommendations for the national blockchain technology segment development is certainly an important and extremely relevant scientific task.

The results obtained shows that most hashing functions satisfy the criteria of statistical security (by the NIST STS method), that is, by different indicators the output sequences (hash values) do not differ (in the statistical sense) from the truly random sequences. These are mainly known and standardized algorithms, which are applied in various cryptographic applications and have already been substantially researched and studied in previous tests. However, among the algorithms in Table 1 there are those whose statistical certainty is either unsatisfactory or completely unacceptable. For example, the well-known hashing algorithm RIPEMD160, which is standardized in ISO/IEC 10118-3:2018 and accepted for use in the European Union, has shown low values of statistical security (the average number of statistical tests with $P_j \geq 0,96$ completed does not exceed 85). That is, if the RIPEMD160 algorithm inputs an excess sequence (in our studies, the input sequence was formed by a regular counter), the generated hash sequences differ from the random sequence, i.e. they have some determinism. Although we have not identified any specific defects in the RIPEMD160 algorithm, the results indicate that some of the generated hash codes are flawed in terms of randomness and unpredictability.

The unsatisfactory performance of the DJB-2, DJB-2 XOR, and LOSELOSE hashing algorithms should be noted separately. In terms of statistical security, they are not acceptable for practical use in cryptographic applications. This conclusion is predictable because the DJB-2, DJB-2 XOR, and LOSELOSE algorithms are essentially not cryptographic, and the calculation of the hash sequences in them is similar to a regular checksum. But, as the results show, even when using statistically dangerous algorithms as part of cascading mining schemes (for example, in the X family hash algorithms), the generated hash sequences also do not satisfy statistical security indicators (see last two lines of the Table 1).

Thus, choosing a hashing algorithm for building blockchain system elements is extremely important and painstaking. In view of the results obtained, in addition to performance, it is also necessary to consider the reliability and security of cryptocurrencies. Also important is the availability of Specialized Computers (ASICs), which greatly accelerate mining in certain consensus protocols. Therefore, to justify the choice of hashing algorithms, it is necessary to consider various factors and performance indicators, including the features of building a specific blockchain system, consensus protocols, processing and messaging algorithms, etc.

This research might be useful for the improvement of various methods of information security, as well as other practical use [17-22].

References

1. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
2. NIST Cryptographic Toolkit. https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software
3. ISO/IEC 10118-1:2016. Information technology – Security techniques – Hash-functions – Part 1: General. (2016-10), 12 p. https://www.iso.org/standard/64213.html
4. Handbook of Applied Cryptography by Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. October 1996, 816 pages, Fifth Printing (August 2001). http://cacr.uwaterloo.ca/hac/
5. 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
6. NISTIR 7896 Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. https://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf
7. A New Standard of Ukraine: The Kupyna Hash Function. https://eprint.iacr.org/2015/885.pdf
8. Ed2k-hash. 7 May 2005. https://wiki.anidb.info/w/Ed2k-hash
9. The C Programming Language by Brian W. Kernighan (1978-02-22) Paperback, Prentice Hall, 178 p.
10. Melanie Swan. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc, 2015, 152p.
11. Marco Iansiti and Karim R. Lakhani (2017). “The Truth About Blockchain”. Harvard Business Review (January–February 2017 issue). pp. 118-127.
12. Dylan Yaga, Peter Mell, Nik Roby, Karen Scarfone. NISTIR 8202 Blockchain Technology Overview. National Institute of Standards and Technology, Internal Report 8202, 66 pages (October 2018). https://doi.org/10.6028/NIST.IR.8202
13. X11 - cryptocurrency mining algorithm with 11 rounds of hashing. Alexander Markov. May 23, 2018. https://miningbitcoinguide.com/mining/sposoby/x11
14. Cryptocurrency mining algorithms - table 2019 and a brief description. https://mining-cryptocurrency.ru/algoritmy-kriptovalyut/
15. Gorbenko, I., Kuznetsov, A., Gorbenko, Y., Vdovenko, S., Tymchenko, V., & Lutsenko, M. (2019). Studies on Statistical Analysis and Performance Evaluation For Some Stream Ciphers. *International Journal of Computing, 18*(1), 82-88.

16. I. Gorbenko, A. Kuznetsov, V. Tymchenko, Y. Gorbenko and O. Kachko, "Experimental Studies Of The Modern Symmetric Stream Ciphers," *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 125-128. doi: 10.1109/INFOCOMMST.2018.8632058
17. 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
18. Krasnobayev V., Kuznetsov A., Koshman S., Moroz S. (2019) Improved Method of Determining the Alternative Set of Numbers in Residue Number System. In: Chertov O., Mylovanov T., Kondratenko Y., Kacprzyk J., Kreinovich V., Stefanuk V. (eds) Recent Developments in Data Science and Intelligent Analysis of Information. ICDSIAI 2018. Advances in Intelligent Systems and Computing, vol 836. Springer, Cham, pp. 319-328, 05 August 2018. DOI: 10.1007/978-3-319-97885-7_31
19. Hu Z., Gnatyuk S., Kovtun M., Seilova N. Method of searching birationally equivalent Edwards curves over binary fields, Advances in Intelligent Systems and Computing, Vol. 754, pp. 309-319, 2019.
20. Iavich M., Gagnidze A., Iashvili G., Gnatyuk S., Vialkova V. Lattice based Merkle, CEUR Workshop Proceedings, Vol. 2470, pp. 13-16, 2019.
21. 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.
22. 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