Method of quick hash functions quality determination[⋆]

Andrii Sahun[1,*,†], Yevheniy Nikitenko[1,†], Pavlo Gikalo[2,†], Olena Panasko[3,†] and Valerii Dudykevych[4,†]

[1] *National University of Life and Environmental Sciences of Ukraine, 15 Heroyiv Oborony str., 03041 Kyiv, Ukraine*
[2] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37 Beresteiskyi ave., 03056 Kyiv, Ukraine*
[3] *Cherkasy State Technological University, 460 Shevchenka ave., 18000 Cherkasy, Ukraine*
[4] *Lviv Polytechnic National University, 12 Stepana Bandery str., 79013 Lviv, Ukraine*

Abstract
To evaluate the effectiveness of the developed method, a set of hash functions was implemented based on the MD5 function (inclusive). It is confirmed that the traditional approach for assessing the quality of hash outputs—based on detecting collisions using brute-force methods—has a significant drawback: high computational complexity. The proposed method is based on a hypothesis confirmed during the study: a higher-quality hash function is one whose statistical variance characteristics meet certain criteria. The method makes it possible to clearly determine whether a given hash function can be considered cryptographic or should be classified as non-cryptographic.

Keywords
hash function, collisions, hash function quality, statistical variance, cryptographic and non-cryptographic hash function

1. Introduction

The current state of information technology development is leading to the emergence of new hash functions, each with its own unique characteristics and areas of application (cryptographic, non-cryptographic, and checksum-related). It is well known that the main parameter used to evaluate a hash function is the number of collisions, the presence or absence of collisions, and overall resistance to collision occurrence [1–3]. Investigating the quality of any crypto-algorithm is an important issue in itself and is covered in many sources [1–4]. As a rule, a separately developed ad-hoc method is used for such studies [5].

In addition to detecting collisions when assessing the quality and robustness of hash functions, mathematical methods can generally be used to evaluate hash functions. A similar approach is described in [3], where the study demonstrates the use of statistical metrics (such as the Strict Avalanche Criterion (SAC)), which are essentially closely related to variance analysis. Research in the context of network equipment, where variance indicators are used to assess the uniformity of hash distribution, is presented in [6].

Identifying and analyzing the collision problem for various hash functions is a non-trivial task. In many cases, different ad hoc algorithms and technologies are used for this purpose. It has been proven that a cryptographic hash function must be resistant to both preimage attacks and collisions. Meanwhile, for non-cryptographic functions and those used for computing checksums, the priority is typically on resisting the occurrence of collisions [5].

Various methods exist for detecting collisions, while the most accurate approach remains the brute-force method of exhaustively testing all possible values. However, such methods are computationally intensive. A faster alternative is the heuristic estimation of collision probability using the birthday paradox, which is based on the number of possible output values N (the hash

[⋆] *CSDP'2025: Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine*
[*] Corresponding author.
[†] These authors contributed equally.

a.sagun@nubip.edu.ua (A. Sahun); e.nikitenko@nubip.edu.ua (E. Nikitenko); gikalo.pavlo@lll.kpi.ua (P. Gikalo); o.panasko@chdtu.edu.ua (O. Panasko); valerii.b.dudykevych@lpnu.ua (V. Dudykevych)
0000-0002-5151-9203 (A. Sahun); 0000-0002-9222-644X (E. Nikitenko); 0000-0002-8058-8267 (P. Gikalo); 0000-0002-0510-7742 (O. Panasko); 0000-0001-8827-9920 (V. Dudykevych)

size) [5, 7]. This approach is considered optimal in terms of time efficiency and acceptable accuracy when assessing the quality of a hash function. Recent advances in cryptographic research emphasize the importance of developing robust hash functions resistant to collision and preimage attacks, which remains a challenging task due to the evolving complexity of cryptographic algorithms [6]. Additionally, the prospects of post-quantum cryptographic algorithms pose new challenges and opportunities for designing hash functions with improved security properties and efficient evaluation methods [7–12]. These studies underscore the necessity for practical, time-efficient approaches to assess hash function quality without compromising accuracy. Nonetheless, in practical scenarios, applying these methods to evaluate hash function quality remains extremely difficult due to their high computational cost.

2. The aim of this research

The aim of this research is to develop and improve methods for detecting hash function quality, which would allow for evaluating their cryptographic properties—such as the avalanche effect, distribution uniformity, and collision resistance.

3. Main part

To verify the above hypothesis regarding a method for evaluating the quality of a given hash function H(x), the authors developed three "simplified" hash functions—$H_1(x)$, $H_2(x)$ to $H_n(x)$—which return output values of fixed length similar to that of the original function. These functions are structurally similar and represent lightweight variants of the MD5 hash function.

The results obtained through the computational experiment for the statistical deviations of the hash functions $H_1(x)$, $H_2(x)$ … $H_n(x)$ та $H_{md5}(x)$ are compared based on the criterion of which function exhibits the smallest deviation from certain variance centers of the output hash values across several input samples. The MD5 function is included in this comparison and is treated as a benchmark of a "robust" cryptographic hash function.

When formulating the final conclusions, the time required to compute hash values for all testing functions is also considered. For the standard MD5 function, a program implementation from a referenced source [13].

All used md5-like hash functions used in the research generate 128-bit output values. Just like the original md5 function, simplified variants can be used to split databases and calculate checksums to verify file integrity.

During hypothesis testing, all three simplified hash functions $H_1(x)$, $H_2(x)$, $H_3(x)$ and the original function $H_{md5}(x)$ produce 128-bit values. They also demonstrate an avalanche effect. Below is a general description of the created demo functions $H_1(x)$, $H_2(x)$, $H_3(x)$ and the original function $H_{md5}(x)$:

1. Function $H_{md5}(x)$ (#1)—MD5Example (original MD5) is a standardized cryptographic hash function with 128-bit output. This function consists of 4 rounds of 16 operations, which include bitwise logical functions, addition modulo 2^{32} and cyclic shifts. Its properties include a good avalanche effect and its initial creation for cryptographic applications. However, it is now considered to be cryptographically compromised. This function is deterministic and fully implements compression of any size to a 128-bit digest.
2. Function $H_1(x)$ (#2)—differs from the original MD5 in that it does not have rounds or a Merkle–Damgård structure. It uses simple bitwise operations (XOR, rotateLeft, rotateRight) without complex logic. Instead of tables of constants or complex functions, simply uses position-based character shifts. The initialization vector used in this hash function (IV) is hardwired and fixed. It supports the avalanche effect, but: (1) does not provide collision resistance; (2) is not cryptographically secure; (3) can be used for simple hashing tasks where security is not critical (for example, checksum, in hash tables).

3. The function $H_1(x)$ (#2) differs from MD5 in that: (1) has a more complex structure than SimpleHash128, but is still much simpler than MD5; (2) uses non-linear bitwise combinations, multiplication by prime numbers (for example, 131, 0x5a5a5a5a)—to enhance the avalanche effect; (3) does not have a clear block structure or rounds like MD5; (4) does not use padding, which is a critical part of secure hashes; (5) provides a tangible avalanche effect, even with a small change of symbols; (6) is not suitable for cryptography due to the fact that it is easily subject to the selection of input data to create collisions and does not have differential stability.

The main differences between these hash functions are listed in Table 1.

Table 1
Characteristics of the researched hash functions

Hash-function properties	Md5(№1)	SimpleHash128(№2)	AvalancheHash(№3)
Crypto-strenght	compromised	insufficient	insufficient
Avalanche-effect	sufficient	notable	sufficient
Round's-structure	standard	custom	custom
Hash length (bits)	128	128	128
Constants and complex operations	standard	custom	custom
Padding	standard	custom	custom
Applying for cryptography	was applied	was not applied	was not applied
Speed of operations	high	higher then №1, №3	high

The hash functions listed in Table 1 can be characterized as follows:

- MD5 is an authentic standardized hash algorithm with complex logic;
- SimpleHash128 is a simplified version of the md5 algorithm, which shows a certain level of avalanche effect, but is not crypto-resistant;
- AvalancheHash is a bit more complex, with non-linear operations for a better avalanche effect.

These simplified analogs are useful for this study and for non-cryptographic tasks (such as hashing in games or databases).

The conclusion regarding the quality of the hash function $H_i(x)$ is made based on the nature of the change in the input data, depending on the change in the input values: the hash function is considered high-quality if the change of at least one bit in the input data leads to changes in a significant number of bits of the output value (avalanche effect). Otherwise, the hash function H(x) is considered to be of poor quality.

To illustrate the operation of this method, a graph is constructed showing the number of changed bits in the hash for each change in the input message.

We will calculate the quality parameters of the obtained hash function according to the following parameters: dispersion, mathematical expectation, number of collisions on a wide sample of initial values of the “hash-128” type, and we will build corresponding illustrative graphs of the quality parameters of the hash function.

4. Testing the statistical characteristics and quality of hash functions

For all three hash functions presented above, a computational experiment was conducted in which each function was supplied with 100 different input values of fixed bit lengths: 4, 6, 8, 10, 12, 14, 16, 32, 64, and 128 bits.

As a result of this computational experiment, a comparison of the three hash functions (MD5, SimpleHash128, and AvalancheHash) was performed. The obtained values of variance, mean (mathematical expectation), and number of collisions for all input lengths are presented in Tables 2 and 3.

Table 2
The comparative table of the results of the analysis of the characteristics of three hash functions (at 4, 6, 8, 10, 12 input values)

Length of input hash value, bits	Name of hash-function	Value of mathematical expectation	Value of variance	Number of collisions
4	MD5	0.5234	0.2299	84
	SimpleHash128	0.5138	0.0596	84
	AvalancheHash	0.4052	0.0470	84
6	MD5	0.5002	0.2441	47
	SimpleHash128	0.4501	0.0874	47
	AvalancheHash	0.4618	0.0564	47
8	MD5	0.4945	0.2462	17
	SimpleHash128	0.5006	0.1081	17
	AvalancheHash	0.5051	0.0689	17
10	MD5	0.4981	0.2472	7
	SimpleHash128	0.5028	0.1194	7
	AvalancheHash	0.4852	0.0743	7
12	MD5	0.4903	0.0876	1
	SimpleHash128	0.4807	0.2472	1
	AvalancheHash	0.4905	0.1342	1

Table 3 is practically a continuation of Table 2, but differs from it in that starting with 14-bit input values, all three studied hash functions begin to demonstrate the actual absence of collision of the output values.

5. Analysis of the computational experiment results

Analyzing the obtained statistical indicators for the hash functions created for the experiment, the following observations can be made:

1. The mean value being close to 0.5 in all cases indicates that the bit values in the hash outputs are approximately uniformly distributed—close to the ideal distribution of 50% zeros and 50% ones in the bit representation. The standardized hash function MD5 consistently demonstrates a stable mean around ~0.5. The function labeled “AvalancheHash” tends to shift slightly toward the range of 0.52–0.53, while the “SimpleHash128” function exhibits slightly greater variability but also approaches 0.5.

Table 3
The comparative table of the results of the analysis of the characteristics of three hash functions (at 4, 6, 8, 10, 12 input values)

Length of input hash value, bits	Name of hash-function	Value of mathematical expectation	Value of variance	Number of collisions
14	MD5	0.4944	0.2472	0
	SimpleHash128	0.5055	0.1342	0
	AvalancheHash	0.4966	0.0921	0
16	MD5	0.4935	0.2475	0
	SimpleHash128	0.4708	0.1431	0
	AvalancheHash	0.4572	0.0996	0
32	MD5	0.5059	0.2474	0
	SimpleHash128	0.4727	0.1958	0
	AvalancheHash	0.5182	0.1431	0
64	MD5	0.5000	0.2476	0
	SimpleHash128	0.4897	0.1953	0
	AvalancheHash	0.5205	0.1425	0
128	MD5	0.5028	0.2481	0
	SimpleHash128	0.4808	0.1956	0
	AvalancheHash	0.5256	0.1453	0

2. The variance for the MD5 function is estimated at approximately 0.247. This value is expected for a uniformly distributed binary sequence. Meanwhile, the SimpleHash128 and AvalancheHash functions show lower variance, indicating a less chaotic (but still relatively strong) distribution of bits in these functions.
3. The number of detected collisions for short input messages (in the 4–10 bit range) is predictably high (ranging from 84% to 47%). However, starting from input lengths of 12 bits, all evaluated functions show practically zero collisions. All functions exhibit the same trend of decreasing collision rates as the input length increases.

Let's make a graph of variance change dependencies, mathematical expectation and collisions for functions No. 1, No. 2, No to illustrate the operation of this method graphs (Figure 1).

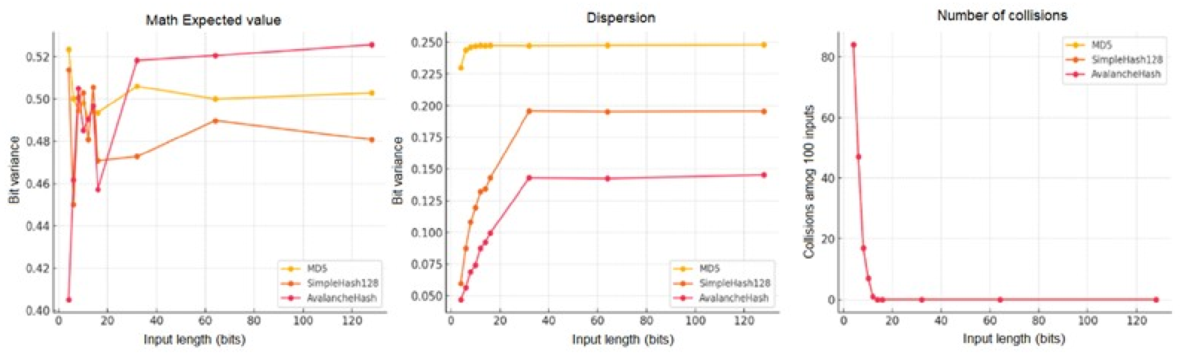

Figure 1: graphs of variance change dependencies, mathematical expectation and collisions for functions No. 1, No. 2, No. 3 Commons

From Figure 1 (left chart), it is evident that the value for MD5 consistently remains around 0.5, which corresponds to a uniform distribution of bits “0” and “1”. At the same time, for the SimpleHash128 and AvalancheHash functions, these values initially fluctuate, but stabilize as the input length increases.

In the center chart of Figure 1, we observe that the variance values for the MD5 function remain approximately constant at ≈ 0.247, which is typical for a uniform binary distribution. The functions labeled “SimpleHash128” and “AvalancheHash” exhibit lower variance, indicating a less chaotic bit distribution (although still considered acceptable).

The right chart in Figure 1 shows the number of detected collisions for all three functions under study. For short input lengths (4–10 bits), all the examined functions exhibit approximately the same number of collisions (the graphs coincide almost in one line). However, starting from an input length of 12 bits, no collisions are observed for any of the functions (which is fully expected for 100 unique input combinations).

To further test the hypothesis regarding the evaluation of hash function quality using statistical indicators, we will calculate the corresponding values for the SHA-1 (Figure 2a) and SHA-2 (Figure 2b) functions.

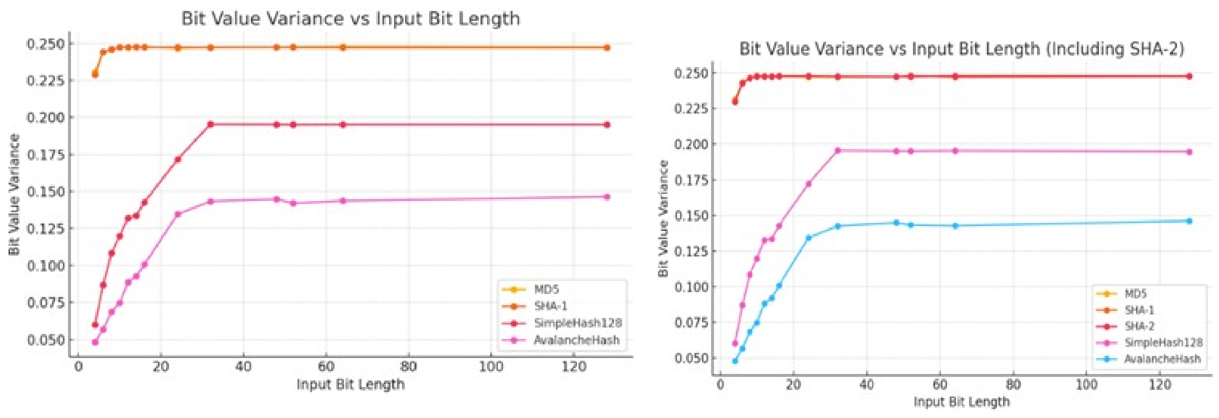

Figure 2: Bit Value Variance vs Input Bit length: a—for MD5, SHA-1, SimpleHash-128 and AvalancheHash functions; b—for MD5, SHA-1, SHA-2, SimpleHash-128 and AvalancheHash functions

As seen in Figure 2, the MD5 and SHA-1 functions exhibit stable variance around 0.25—this is expected for a high-quality cryptographic hash function with a uniform bit distribution. The SimpleHash128 function shows a lower but gradually increasing variance (≈0.15–0.20), indicating a less entropic yet predictable distribution. The output of AvalancheHash varies more significantly—it starts lower but increases to around ≈0.15–0.18, which indicates the presence of an avalanche effect, albeit with lower entropy.

The SHA-2 (SHA-256) function, as expected, demonstrates high and stable variance (~0.25)—nearly an ideal uniform bit distribution. The SHA-1 and MD5 functions also remain close to this ideal value, with minor fluctuations. In contrast, SimpleHash128 and AvalancheHash show lower variance, indicating a less random (but still avalanche-like) bit distribution.

This confirms the higher entropy and uniformity of cryptographic hash functions (SHA-2, SHA-1, MD5) compared to the simplified hash implementations (SimpleHash128 and AvalancheHash).

The computational complexity of calculating variance and detecting collisions for a hash function differs significantly in nature.

A specific comparison of the computational complexity of finding a collision (through brute force and through the birthday paradox) and calculating the variance for the 128-bit hash functions “SimpleHash128” and “AvalancheHash” and MD5 is given in Table 4.

Table 4
The comparative analysis of hash function quality evaluation methods for functions "SimpleHash128", "AvalancheHash" and MD5

Methods for evaluating hash function quality	Complexity	Valuation of results	Suitability for use/speed
Full brute-force	$\theta(2^{128})$	$\approx 3.4*10^{38}$	Impossible (for now)
Birthday Paradox	$\theta(2^{64})$	$\approx 1.8*10^{19}$	Almost impossible
Dispersion (100 input)	$\theta(n*m)$	$\approx 1.3*10^{4}$	Very good (fastest)
Dispersion (200 input)	$\theta(n*m)$	≈12800	Very good (fast)
Dispersion (500 input)	$\theta(n*m)$	≈25600	Very good (fast)
Dispersion (1000 input)	$\theta(n*m)$	≈64000	Very good (up to 1 second)
Dispersion (10000 input)	$\theta(n*m)$	≈1280000	Depends of hardware

So, traditionally, to find a collision, you need to go through all possible output values. Since the output for the functions "SimpleHash128" and "AvalancheHash" and MD5 is 128 bits, the number of possible unique hashes is: $2^{128} \approx 3{,}4\times10^{38}$. This means that in the worst case, $3{,}4\times10^{38}$ unique input values need to be checked to be guaranteed to find a collision. When using the "birthday" attack [14, 15], it is possible to reduce the search to $2^{64} \approx 1{,}8\times10^{19}$ iterations. This is significantly faster, but still very computationally intensive.

If is used a modern video card, for example NVIDIA RTX 4090, to sort through all the possible values of the argument of the developed function. Then, for the well-known and comparable MD5 algorithm developed for its complexity (it has a 128-bit output), the computing power allows you to calculate 200-300 billion hashes per second (i.e. 2×10^{11} до 3×10^{11} hashes/sec) it takes from 1.13×10^{27} seconds to go through all possible values $\approx 3{,}6\times10^{19}$ years.

If a more optimal "birthday" attack is implemented, the computational complexity will be: $2^{64} \approx 1.8\times10^{19}$ iterations, and the calculation time will be: 6×10^{7}seconds =1.9 years.

Thus, under real conditions, determining the number of collisions using existing methods is a significantly more computationally intensive task. Analyzing the results of the computational experiments shows that in order to obtain a statistically significant estimate of collision frequency, it is desirable to test a much larger sample for the developed function—for example, 10^{20} or even more hash values. At the same time, Figure 1 demonstrates that the quality of a hash function is closely correlated with the variance of its output values (like in [16, 17]).

The entropy of md5 function values is quite high compared to SimpleHash128 and AvalancheHash functions. At the same time, these functions also show an avalanche effect, but with lower entropy compared to the md5 function.

As seen in Figure 4, the time gain when evaluating and classifying a hash function in terms of its cryptographic nature using variance—compared to traditional collision detection. This one can reach up to $1.65*10^{34}$ times.

Conclusions

The calculation of variance for sample values is a far simpler computational task. Therefore, the variance metric of a hash function allows for a clear conclusion as to whether a given hash function is cryptographic or non-cryptographic.

Variance calculation is an extremely efficient task with linear computational complexity, typically performed in milliseconds even for hundreds of input samples. In contrast, finding collisions, even when using the birthday paradox, remains a task of exponential complexity, making it orders of magnitude more demanding. Thus, for practical use, especially in evaluating the quality of non-cryptographic hash functions, the use of variance-based indicators is a promising approach.

As we can see from the above researches, the function can be cryptographic hash function if it has a uniform bit distribution functions exhibit stable variance around 0.25.

To enable practical application of the proposed evaluation method, further research is required—including the creation of hash function quality classes ranked by acceptable deviations from an ideal hash function. It may also be necessary to develop classification criteria based on deviation from a reference variance value, in order to reliably categorize a given cryptographic hash function as strong, moderate, or weak in terms of quality.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] X. Wang, H. Yu, Collisions of SHA-1: Second-Preimage and Preimage Attacks, 2005.

[2] V. Lakhno, et al., Evaluation of the Probability of Breaking the Electronic Digital Signature Elements, in: Sustainable Communication Networks and Application, Lecture Notes on Data Engineering and Communications Technologies, 93, 2022, 639–648. doi:10.1007/978-981-16-6605-6_48

[3] M. Pandya, et al., Design and Performance Analysis of a SPECK Based Lightweight Hash Function. Electronics, MDPI, 2024.

[4] X. Jiang, M. Li, K. Makulov, V. Lakhno, A. Sahun, Enhanced Neural Differential Distinguisher for Speck32/64 using Attention Mechanisms and Multi Ciphertext Inputs, Informatica, 49(19) (2025) 197–210. doi:10.31449/inf.v49i19.7889

[5] A. Sahun, et al., Devising a Method for Improving Crypto Resistance of the Symmetric Block Cryptosystem RC5 using Nonlinear Shift Functions, Eastern European J. Enterprise Technol. 5(9(113)) (2021) 17–29. doi:10.15587/1729-4061.2021.240344

[6] S. Yevseiev, et al., Development of Niederreiter Hybrid Crypto-Code Structure on Flawed Codes, Eastern-European J. Enterp. Technol. 1.9(97) (2019) 27–38. doi:10.15587/1729-4061.2019.156620

[7] A. Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of Post-Quantum Cryptographic Algorithms, in: Classic, Quantum, and Post-Quantum Cryptography, 3504, 2023, 39–49.

[8] A. Bessalov, et al., CSIKE-ENC Combined Encryption Scheme with Optimized Degrees of Isogeny Distribution, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 36–45

[9] A. Bessalov, V. Sokolov, S. Abramov, Efficient Commutative PQC Algorithms on Isogenies of Edwards Curves, Cryptography 8(3), iss. 38 (2024) 1–17. doi:10.3390/cryptography8030038

[10] S. Abramov, A. Bessalov, V. Sokolov, Properties of Isogeny Graph of Non-Cyclic Edwards Curves, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3550 (2023) 234–239.

[11] M. Iavich, et al., Classical and Post-Quantum Encryption for GDPR, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3829 (2024) 70–78.

[12] S. Zhebka, et al., Method for Adaptive Allocation of Cryptographic Resources in Distributed Databases, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3991 (2025) 620–628.

[13] R. Rivest, The MD5 Message-Digest Algorithm (RFC 1321), Internet Engineering Task Force, 1992. https://datatracker.ietf.org/doc/html/rfc1321

[14] M. Bellare, P. Rogaway, The Birthday Problem, Introduction to Modern Cryptography (PDF), 2005, 273–274.

[15] A. Kozhukhovskyi, et al., Analysis of the Cryptographic Resistance of Electronic Digital Signature Elements, Bulletin of Cherkasy State Technological University. Series: Technical Sciences, (3) (2013) 80–85.

[16] H. Wang, et al., Design Theory and Method of Multivariate Hash Function, Sci. China Inf. Sci. 53 (2010) 1977–1987.

[17] R. Vaughn, M. Borowczak, Strict Avalanche Criterion of SHA 256 and Sub Function Removed Variants, Security, 8(3) (2022) 40. doi:10.3390/cryptography8030040