mathematics

Article

Developing a New Collision-Resistant Hashing Algorithm

Larissa V. Cherckesova, Olga A. Safaryan *, Nikita G. Lyashenko and Denis A. Korochentsev

Department of Cyber Security of Information Systems, Don State Technical University, 344000 Rostov-on-Don, Russia
* Correspondence: safari_2006@mail.ru; Tel.: +7-(863)-238-15-18

Abstract: Today, cryptographic hash functions have numerous applications in different areas. At the same time, new collision attacks have been developed recently, making some widely used algorithms like SHA-1 vulnerable and unreliable. This article is aiming at the development of a new hashing algorithm that will be resistant to all cryptographic attacks, including quantum collision attacks that potentially pose a threat to some widely used cryptographic hash functions. This algorithm was called Nik-512. The avalanche effect is tested to ensure the cryptographic strength of the developed algorithm. The Nik-512 function is then applied to build a data integrity system which can be used to protect data from malicious users.

Keywords: collision resistance; cryptography; data integrity; hashing; quantum attack

MSC: 00A06

check for updates

Citation: Cherckesova, L.V.; Safaryan, O.A.; Lyashenko, N.G.; Korochentsev, D.A. Developing a New Collision-Resistant Hashing Algorithm. *Mathematics* **2022**, *10*, 2769. https://doi.org/10.3390/math10152769

Academic Editors: Weihua Xu, Jinhai Li and Xibei Yang

Received: 22 June 2022
Accepted: 2 August 2022
Published: 4 August 2022

1. Introduction

In the modern world, information security is becoming increasingly important due to the rapid development of information technologies and the ever-growing amount of data that should be protected. According to the OWASP Web Application Security Project, the most critical security risks in 2021 included errors in implementations of cryptographic algorithms and failures in data integrity systems. Cryptographic hash functions have numerous practical applications, namely, password authentication, electronic signature schemes and blockchain proof-of-work protocols. If vulnerability is discovered in a cryptographic hash function, then it can no longer be used to provide required security.

In recent years, serious vulnerabilities have been found in some hashing algorithms that were considered reliable in the past, and these algorithms can no longer be used to ensure cryptographic security. Steady increases in computing power have already made many cryptographic attacks that were previously considered only theoretical now practical for malicious users. Recent groundbreaking achievements in quantum computing mean that in the nearest future, even more effective attacks on hashing algorithms will be possible.

Hence, there is a need for new hashing algorithms that will provide sufficient cryptographic resistance against the most modern attacks that can be accomplished with the use of supercomputers and quantum computers. Strong cryptographic hash functions make it possible to ensure the integrity of information in spite of continuous increases in the computing power that attackers have on their disposal. Improvements in common hashing algorithms have been proposed recently to make them more efficient and resistant to modern collision attacks. For instance, scientists Nada E. El-Meligy, Tamer O. Diab and Ashraf S. Mohra described and implemented a number of valuable changes to the standard SHA-512 that they concluded could be employed to enhance the security of the SHA-2 algorithm without losses in performance [1]. In another article that was published recently, Torres-Alvarado et al. investigated possible errors in the implementation of the SHA-3 standard and proposed a new architecture that can decrease the probability of failures in the algorithm, which makes it more reliable and also provides better resistance to cryptographic attacks [2]. An important application of the cryptographic hash function that has been actively researched in recent years also includes key generation. As Bai, Jiang and Wu showed in an article that examined modern approaches to key generation, utilizing hash functions is an effective way to save memory and improve the performance of such algorithms [3]. Hashing algorithms have several possible applications in machine learning. A cryptographic hash function can be effectively used in image detection systems when the security of the data involved in the detection process is indispensable. Gadamsetty et al. exhaustively researched an application of SHA2-256 to a system for recognizing ships from images [4] and managed to implement a system that ensured data integrity during the detection and further transmission of the collected data. In 2021, Alotaibi utilized a hash function based on the Miyaguchi–Preneel structure to develop an innovative machine learning technique that can be used for malware detection [5].

The object of this work is a cryptographic hash function, the subject of the work is the collision resistance of cryptographic hash functions and the aim of the research is to develop a cryptographic hash function that is resistant to modern collision attacks.

In accordance with the main aim of the research, the following objectives were determined:

- Study the theory of modern hashing algorithms and collision attacks;
- Build a mathematical model of a hash function resistant to collision attacks;
- Perform an analysis of the developed hash function based on statistical methods;
- Implement a data integrity system that is resistant to collision attacks and based on the developed hash function.

The main scientific methods that were used in the research are analysis, classification, comparison and experiment.

2. Materials and Methods

2.1. Description of Hash Function and Collision Attack

A hash function is a function that takes a string of arbitrary length as an argument and calculates a value of fixed length n. The resulting value is called a hash value of the given string [6].

Hash collision for a hash function H is defined as a pair of different messages X and Y that have the same hash value when this hash function is applied to the messages X and Y. An algorithm that calculates a collision for the hash function H is called a collision attack for this function. The number of hash function calls required to find a collision of the hash function is determined as follows:

$$C = 2^{n/2}, \tag{1}$$

where n is a hash length [7].

There are various classifications of cryptographic hash functions. In one of the possible classifications, all cryptographic hash functions can be divided into two classes: keyed and keyless. Keyless hash functions do not use an additional key and are used to confirm that there are no corruptions that may have been introduced during the transmission of a message. On other hand, keyed hash functions use a secret key K that is known to both the sender and the recipient of the message. The use of the key allows for performing message authentication (confirmation of the fact that the message was received from a trusted sender who knows the secret key K) [8].

Based on the resistance of hashing algorithms to various cryptographic attacks, the following types of hashing algorithm are defined:

(1) Resistance to the first preimage attack, which ensures that generating a message that corresponds to a given hash value is not feasible in practice.

(2) Resistance to the attack of finding the second preimage (also known as weak collision resistance) implies the practical impossibility of generating such a message B for predetermined message A that A and B will be a collision.

(3) Resistance to collision attacks means that the implementation of a collision attack for any message is impossible in practice.

(4) Near collisions resistance means a high computational complexity of the problem of finding two messages A and B, such that for a sufficiently small value δ, the following condition will be satisfied:

$$d(H(A), H(B)) < \delta, \tag{2}$$

where H is a hash function and d is the Hamming distance.

(5) Partial preimage resistance means that obtaining any part of the original message from its hash has significant computational complexity.

A collision attack allows an attacker to edit a protected file or to forge an electronic signature, for instance, a system that checks the integrity of some file A based on its hash. Let H1 be the hash value calculated for file A. If file A has been changed, then the hash value for this file will also change and will be equal to some value H2. In this case, the integrity check will fail. However, when a collision attack can be accomplished, the attacker can create a file B for which the hash value is H1, and thus, later they can replace file A with file B. This scheme of a collision attack is shown in Figure 1.

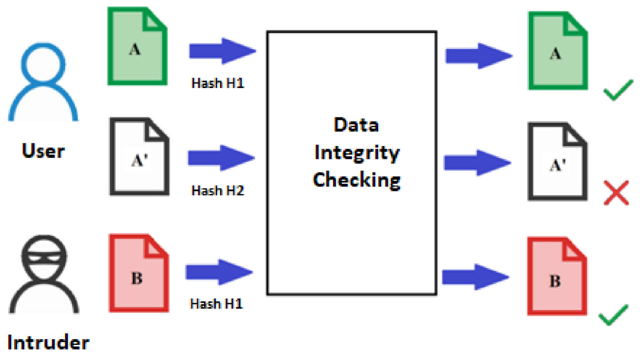


Figure 1. Scheme of a collision attack.

It is important to note that a collision attack allows an attacker to replace a file only if a collision for this file can be generated in advance. Creating a new file with the same hash based on an arbitrary file and a corresponding hash value requires the successful implementation of the second preimage attack. In practice, the most dangerous attack on a cryptographic hash function is the chosen-prefix collision attack, since it allows attackers to impose false information if they choose real data as the first prefix and fake data as the second prefix.

2.2. *Review of Modern Hashing Algorithms and Collision Attacks*

Currently, the most common cryptographic hash functions are algorithms of the SHA family, which includes three standards: SHA-1, SHA-2 and SHA-3. The SHA-1 algorithm was created in 1995 and published in the Federal Information Processing Standard FIPS PUB 180-1 [8]. In 2017, the SHAttered prefix collision attack was developed and implemented in practice [9]. This attack has a computational complexity of 263 [10]. As a consequence, SHA-1 is no longer considered resistant to collision attacks, and SHA-2 or SHA-3 should be used instead [11].

The SHA-2 algorithm is a collision-resistant hashing algorithm, and as of 2022, there is no SHA-2 modification that will allow for any collision attack that is faster than a generic attack [7]. Collision attacks on SHA-2 with a reduced number of rounds have been developed, but they hold only theoretical value since they cannot be applied to the full SHA-2 algorithm [12]. Since SHA-2 is based on the Davis–Meyer structure, there was a serious concern among theoreticians that attacks based on differential cryptanalysis can be found against it. This fact resulted in the emergence of the new SHA-3 standard [13].

The SHA-3 standard was developed by a group of four scientists led by Joan Dymen. In 2012, a special competition was held by the National Institute of Standards and Technology, and the Keccak function won the competition. This function was used for the development of the SHA-3 standard [14].

Currently, there are no collision attacks on the SHA-3 algorithm that would allow a collision faster than a generic collision attack [15]. In 2021, it was proved that the SHA-3 algorithm is resistant to quantum collision attacks. This means that there is no quantum attack on SHA-3 that would allow finding a collision faster than if the BHT quantum algorithm were utilized [16]. From detailed comparisons of SHA-1 and SHA-3, it can be concluded that differential cryptanalysis is ineffective against the SHA-3 algorithm [17].

A common approach to the implementation of collision attacks that can exploit vulnerabilities in the specific hashing algorithm is differential cryptanalysis. This method is based on the analysis of differences between input data values at various rounds of the hashing process [18]. In differential cryptanalysis, a difference is understood to be a binary sequence obtained by applying the XOR operator to the current two messages in a given round of the algorithm [19].

Another important class of collision attacks is quantum collision attacks. For a long time, quantum attacks were only theoretical. However, in recent years, working quantum computers have been developed by Google and IBM. This means that quantum attacks may become a reality in the near future, and when assessing the cryptographic strength of hash functions, it is necessary to take into consideration the possibility of implementing quantum collision attacks [20].

The computational complexity of a generic quantum attack can be calculated as follows:

$$C = 2^{n/3}, \tag{3}$$

where n denotes hash length.

In the case of using a quantum computer to accomplish a collision attack, the Brasard–Hoyer–Tappard (BHT) algorithm, which based on the application of the Grover algorithm can significantly reduce the amount of computation required. The computational complexity of the BHT algorithm is equal to the complexity of the generic quantum collision attack [21].

The BHT algorithm for finding a collision of a function H with hash length n consists of the following steps:

(1) A subset $X \subseteq \{0,1\}^n$ is chosen with cardinality equal to $2^{n/3}$.

(2) The value of the function $f(x)$ is calculated for each element x from the subset X.

(3) A list L is created from elements of the form $\{x, f(x)\}$ for each x.

(4) The list L is stored in quantum memory.

(5) Grover's algorithm is used to find x' such that the following condition will be satisfied:

$$(x' \in \{0,1\}^n \backslash X) \; and \; f(x') = f(x) \tag{4}$$

Thus, the value x' found as a result of the execution of the algorithm and the v corresponding value x will be a collision of the hash function H.

The main disadvantage of the BHT algorithm is the amount of quantum memory required to execute the algorithm. This means that in practice, providing a sufficient amount of quantum memory will be expensive to the extent of the practical impossibility of employing the BHT algorithm to find a collision [22].

The amount of quantum memory q required to apply the BHT algorithm grows exponentially with the hash length n and is calculated by the formula:

$$q = 2^{n/3}, \tag{5}$$

where n denotes the hash value length.

To solve the problem associated with memory requirements, scientists A. Chailloux, M. Naya-Plasencia and A. Schrottenloher in a paper published in 2017 proposed an innovative quantum algorithm for finding hash collisions [23]. This algorithm is called CNS. The main approach in this algorithm is the modification of the collisions search that places most of the data processed in classical memory instead of quantum memory. In this case, it is necessary to minimize the inevitable loss of computational speed [24]. The developed algorithm uses the amount of quantum memory, which linearly depends on the hash length n. In this case, the time required to perform the algorithm is determined by the formula

$$T = 2^{2n/5}. \tag{6}$$

The required amount of classical memory can be calculated as follows:

$$c = 2^{n/5}. \tag{7}$$

2.3. Related Works

Although most of the publications related to hashing algorithms investigate some aspects of already existing hash functions, there have been several articles that propose a completely new hashing algorithms within last three years. In the article published by K. Sakan, S. Nyssanbaeva, N. Kapalova et al. in 2022 [25], a new hashing algorithm called HBC-256 was developed and analysed. The authors used the Devis–Mayer construction to build their algorithm. They also tested an avalanche effect of the developed algorithm and proposed an implementation of the developed algorithm that utilizes parallelization to improve performance. The most notable features highlighted by the authors are the reduction of the number of rounds for improving the performance and the finding that the specific structure of the compression function increases the number of blocks that can be processed in parallel. In 2020, Tutueva, Karimov, Moysis et al. [26] described a new hashing algorithm that uses adaptive chaotic maps and provides resistance to collision attack. A similar approach was used by Yu Wang, Liquan Chen, Xinguyan Wan et al. [27], where a new keyed hash function was proposed. This function is based on the Merkle–Damgard construction with a chaotic map neural network. The authors state that their algorithm is resistant to all modern attacks including multicollision attacks, which are mentioned less frequently in articles on cryptographic hash functions than are classical collision attacks. A completely different approach to the problem was proposed in 2020 by Edimar Verissimo de Silva [28], who developed a new architecture that utilizes a complex sequence of permutations and rotations of separated blocks and leads to strong resistance against attacks at the expense of performance. From the review of related works, it can be concluded that scientists today are exploring innovative and uncommon approaches to building hashing algorithms that will be resistant to modern attacks.

3. Results

3.1. Development of the Collision-Resistant Hashing Algorithm

A cryptographic hash function that is resistant to collision attacks must be a one-way function. That means that it is easy to calculate the hash value for any message but that finding at least one message for which the hash value is equal to the given value must be computationally difficult (computation complexity depends on the hash length at least exponentially).

Another indispensable property of the cryptographic hash function is the avalanche effect. This refers to the property of a hash function that a slight change in the input message will result in a complete change of the hash value. The avalanche effect means that a change in one bit of the message will result in a change in half the bits of the hash.

One of the possible approaches to the development of a collision-resistant hash function is iterative hash construction with a block cipher being employed as a compression function. The main advantage of this approach is the strong cryptographic properties of the developed hash function that can be achieved without additional modifications if the chosen block cipher is secure enough. Another common approach is based on sponge construction, which can be described as an algorithm with internal state that uses a permutation function to produce output (hash value) from a stream of input bits. Authors decided to build a compression function without relying on the existing block ciphers and constructed it as a set of permutations of internal state with bitwise operations applying to its elements. The advantage of this approach in comparison with SHA-1 and SHA-2 is that it allows for building a secure hash function with fewer rounds and without determining specific constants for the internal state. The shortcoming of the chosen approach is that each round requires a significant number of operations to be performed and there is a need for optimization that will take advantage of potential efficiency of bitwise operations in low-level programming languages.

As a result of the conducted research, a cryptographic hash function with a hash output length of 512 bits has been developed. This function is based on the Miyaguchi–Preneel structure. It uses two 512-bit long values A stored as an array of 16 32-bit elements. The message for which the hash should be calculated is divided into blocks of 512 bits. The last block is padded with zeros to a size of 512 bits. There is no limit to the input string length for the developed algorithm.

At the beginning of the algorithm execution, the value of S is initialized as 0 if the function works in keyless mode or as a key value if the key mode is used. The value of M is initialized with the first block of the message being processed.

For each message block, the compression function is called, which takes as parameters the current value of the internal state and the message block that should be processed. The flow chart of the hash function is shown in Figure 2.

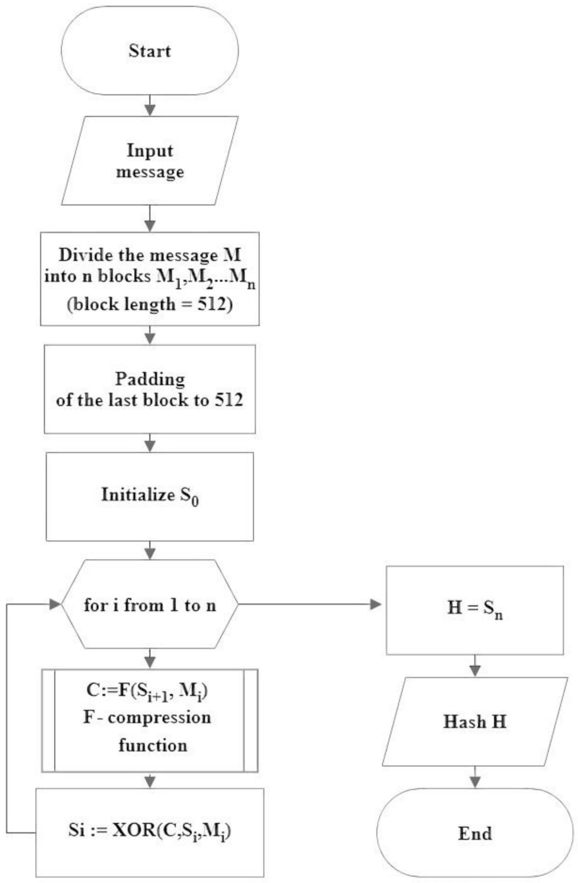

Start
Input message
Divide the message M into n blocks $M_1, M_2 \ldots M_n$ (block length = 512)
Padding of the last block to 512
Initialize S_0
for i from 1 to n
$C := F(S_{i+1}, M_i)$ F - compression function
$Si := XOR(C, S_i, M_i)$
$H = S_n$
Hash H
End

Figure 2. The flow chart of the implemented hash algorithm.

The compression function takes *A* and *B* values of 512 bits as input and calculates a value of 512 bits. Then calculation 32 rounds is performed, during each of the rounds the following formulas are applied to the input:

(1) The input receives the current values of arrays *A* and *B*.

(2) For A_i, the transformation is performed according to formula (7) (the indexes of the elements are taken modulo 16):

$$A_i := (A_i \gg 1) \bigoplus (\neg B_i) \bigoplus (A_{i+6} \wedge \neg B_{i+3}), \tag{8}$$

(3) The array is rotated (the i-th element becomes the $(i-1)$-th element, and the zero element becomes the last element).

(4) For A_i with indices $2 \leq i \leq 16$, the transformation is performed according to the formula:

$$A_i := (\neg A_i \gg 1) \bigoplus B_i \bigoplus (\neg A_{i-6} \wedge B_{i-3}), \tag{9}$$

(5) For all A_i and B_i, transformations are performed according to the formula

$$\begin{cases} A_i := A_i \; mod \; 2^{32} \\ B_i := \left(B_i + A_i \times A_{(i+r^3) mod 16}\right) mod \; 2^{32} \end{cases}, \tag{10}$$

where r is the number of the current round (for the first round r = 0, for the last round r = 31)

The flow chart of the compression function is shown in Figure 3.

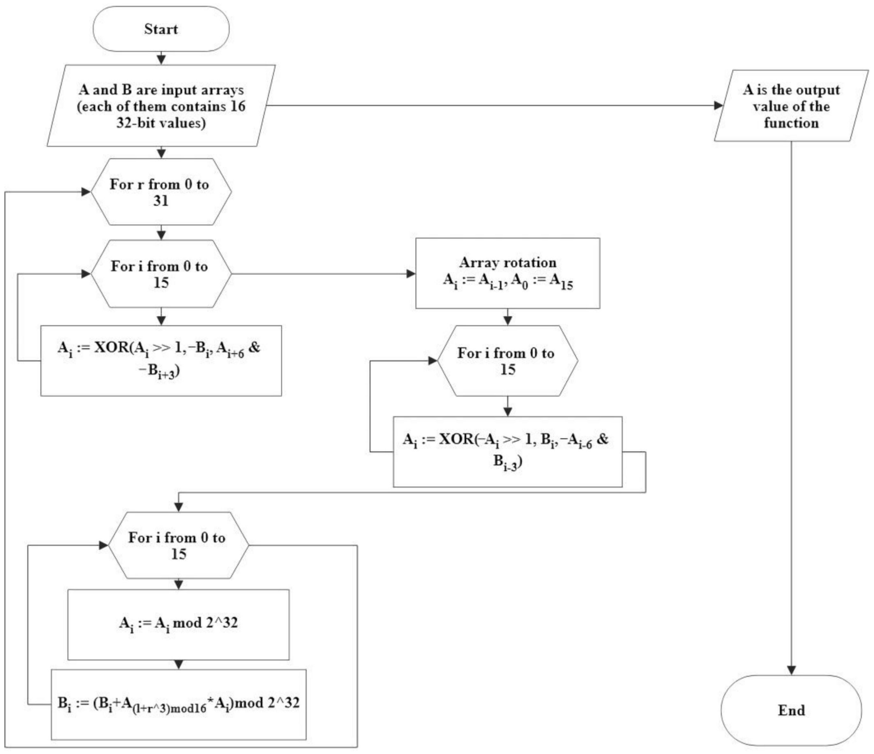

Start
A and B are input arrays (each of them contains 16 32-bit values)
A is the output value of the function
For r from 0 to 31
For i from 0 to 15
Array rotation $A_i := A_{i-1}, A_0 := A_{15}$
A_i := XOR(A_i >> 1, $-B_i$, A_{i+6} & $-B_{i+3}$)
For i from 0 to 15
A_i := XOR($-A_i$ >> 1, B_i, $-A_{i-6}$ & B_{i-3})
For i from 0 to 15
$A_i := A_i$ mod 2^32
$B_i := (B_i + A_{(l+r\text{^}3)mod16} * A_i)$mod 2^32
End

Figure 3. The flow chart of the compression function.

In this work, the authors have developed the hashing algorithm with hash length 512, but it is possible to build a hashing algorithm with even larger hash length following the same approach. For example, a hash function that produces 1024-bit output can be created from Nik-512 by increasing the length of the input arrays A and B (in this case, both arrays will contain 32 values instead of 16). To achieve this, the input message should be divided into blocks with the length 1024, and some adjustments to the formulas using in the compression function will be needed. It is important to note that the authors do not claim that this modification will have the same cryptographic properties as the original algorithm.

To test the avalanche effect, it is necessary to generate many message pairs that differ from each other by only one bit and calculate the hash values for these messages. After that, the Hamming distance should be calculated for each pair of received hashes. In the case of a strong cryptographic hash function, the Hamming distance for pairs of hashes can be considered a random variable with expectation equal to $n/2$, where n is the hash length. In the case of the hash function developed as the result of this research, the mathematical expectation was 256. Then, the deviation of the mathematical expectation and the standard deviation were calculated. As a sample, 65,536 pairs of messages were used.

The test was performed for four different hash functions:

(1) SHA512 modification of the SHA-2 algorithm.

(2) SHA3-512 modification of the SHA-3 algorithm.

(3) Hash function GOST 34.10–2018 with a hash length of 512 bits.

(4) Nik-512 (this name was given by the author to the new hashing algorithm described in this paper).

Figure 4 shows the results of testing the avalanche effect of the Nik-512 hash function.

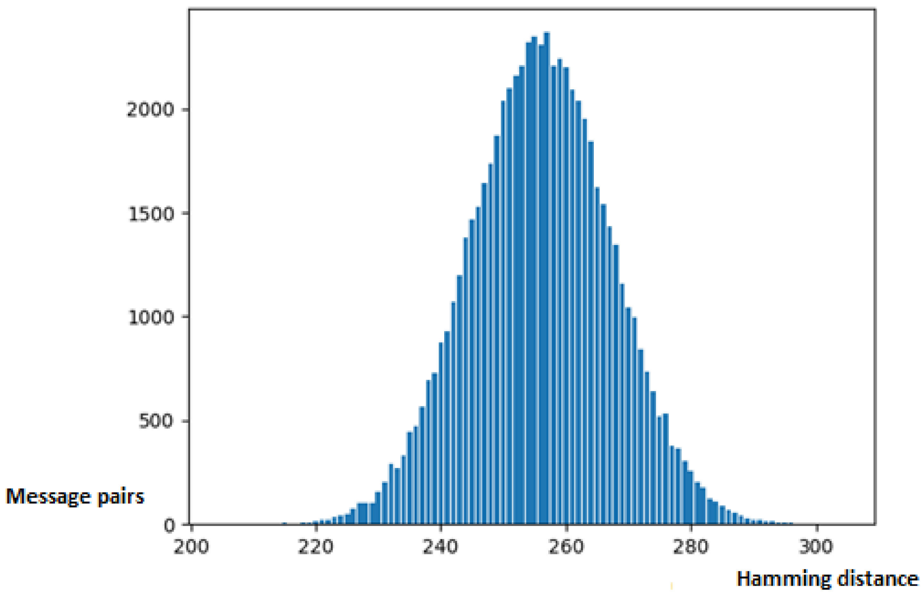


Figure 4. The avalanche effect of the hash function Nik-512.

Table 1 shows the differences between the mathematical expectations and the actual averages and standard deviations for various hash functions. Among the four algorithms examined, the Nik-512 hash function has the smallest differences between the average and expected values and the standard deviations. Based on the data obtained, it can be concluded that the avalanche effect is stronger for the developed hash function than for other algorithms. This means that the behavior of the developed function is closest to the random number generator. Thus, it provides the best collision resistance. The result obtained allows for using the developed Nik-512 algorithm as a cryptographic hash function resistant to collision attacks.

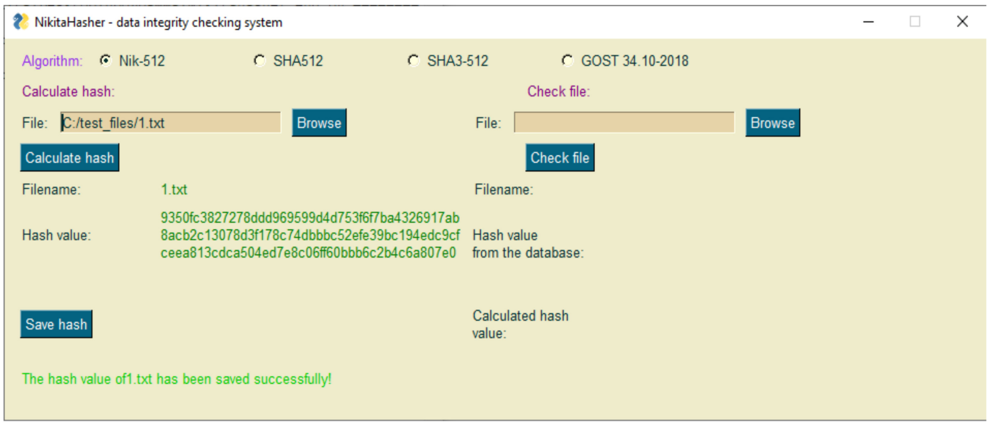
Table 1. The avalanche effect testing results.

Hash Function	Absolute Difference between the Expected Value (256) and the Average Hamming Distance	Standard Deviation of the Hamming Distance
SHA512	0.3786	11.3647
SHA3-512	0.4921	11.3451
ГОСТ 34.11–2018	0.4515	11.2974
Nik-512	0.0467	11.2951

3.2. Demonstration of the Implemented Data Integrity Checking System

Based on the developed hash function, a data integrity system was implemented. The user can choose one algorithm from the list that includes SHA512, SHA3-512, the hash function described in the Russian standard GOST 34.11–2018 (a modification with a hash length of 512 bits is used) and the Nik-512 hash function.

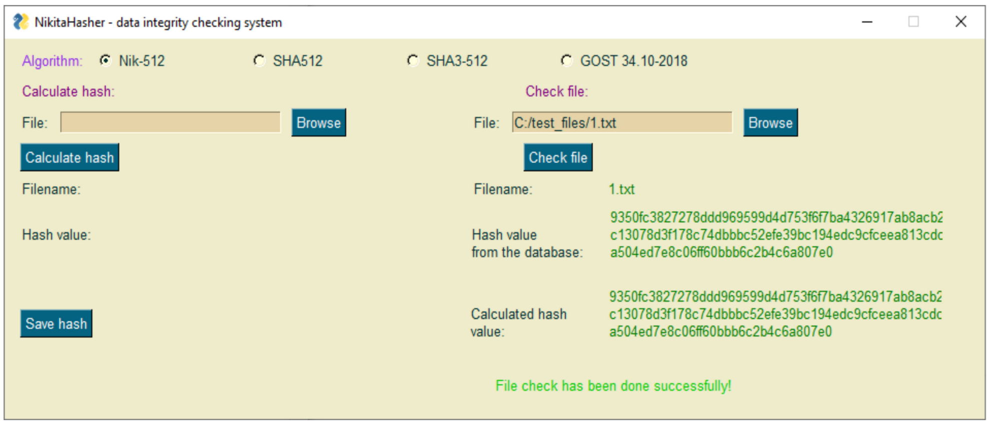
Figure 5 shows an example of calculating the Nik-512 hash function for a file selected by the user. The resulting value can later be stored in the database.



```
Algorithm: [✓ Nik-512]  SHA512  SHA3-512  GOST 34.10-2018
Calculate hash:                                   Check file:
File: C:/test_files/1.txt                         File:
Filename:     1.txt                               Filename:
Hash value:   9350fc3827278ddd969599d4d753f6f7ba4326917ab
              8acb2c13078d3f178c74dbbbc52efe39bc194edc9cf   Hash value
              ceea813cdca504ed7e8c06ff60bbb6c2b4c6a807e0    from the database:
                                                  Calculated hash
                                                  value:
The hash value of1.txt has been saved successfully!
```

Figure 5. Hash value calculation and saving.

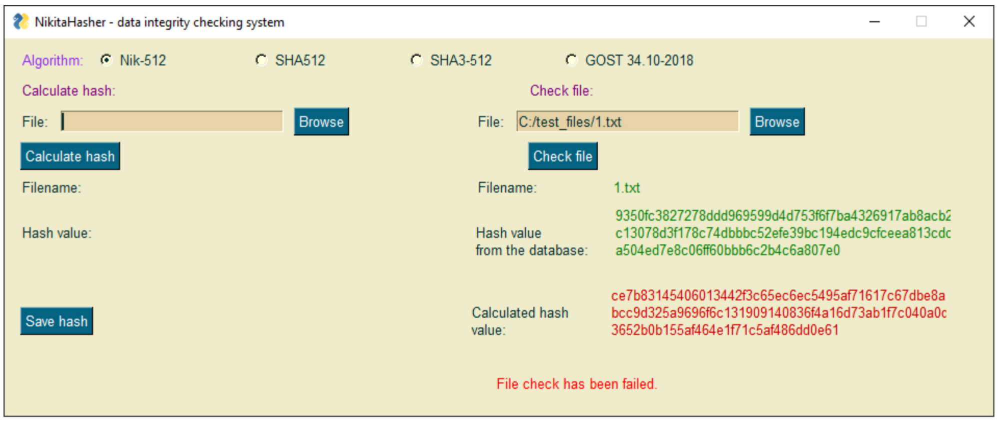
Figure 6 shows an example of a file integrity check. The previously stored hash value is loaded from the database. The hash value is also calculated for the current version of the file, and then a comparison with the loaded value is performed. If the two values are the same, a message is displayed indicating that the verification was successful.



```
Algorithm: [✓ Nik-512]  SHA512  SHA3-512  GOST 34.10-2018
Calculate hash:                     Check file:
File:                               File: C:/test_files/1.txt
Filename:                           Filename:     1.txt
Hash value:                         Hash value    9350fc3827278ddd969599d4d753f6f7ba4326917ab8acb2
                                    from the database: c13078d3f178c74dbbbc52efe39bc194edc9cfceea813cdc
                                                  a504ed7e8c06ff60bbb6c2b4c6a807e0
                                    Calculated hash 9350fc3827278ddd969599d4d753f6f7ba4326917ab8acb2
                                    value:        c13078d3f178c74dbbbc52efe39bc194edc9cfceea813cdc
                                                  a504ed7e8c06ff60bbb6c2b4c6a807e0
File check has been done successfully!
```

Figure 6. A successful data integrity check.

Figure 7 shows an example of a check in which the hash value for the uploaded file differs from the previously saved value, meaning that the file has been changed and the integrity check has failed.



Algorithm: [✓ Nik-512]

File: C:/test_files/1.txt

Filename: 1.txt

Hash value from the database:

```
9350fc3827278ddd969599d4d753f6f7ba4326917ab8acb2
c13078d3f178c74dbbbc52efe39bc194edc9cfceea813cdc
a504ed7e8c06ff60bbb6c2b4c6a807e0
```

Calculated hash value:

```
ce7b83145406013442f3c65ec6ec5495af71617c67dbe8a
bcc9d325a9696f6c131909140836f4a16d73ab1f7c040a0c
3652b0b155af464e1f71c5af486dd0e61
```

File check has been failed.

Figure 7. A failed data integrity check.

Table 2 compares the developed software with existing solutions. Supported hashing algorithms with resistance to collision attacks are shown for each program. The support of each of these algorithms is an advantage of the program. The presence of hashing algorithms with discovered vulnerabilities (MD-5, SHA-1) is a disadvantage of the program, since the user can use this algorithm to protect a file and thus make this file vulnerable to a collision attack. The developed software tool supports modifications with a hash length of 512 bits of the standard algorithms SHA-2, SHA-3 and GOST 34.10-2018. Additionally, the Nik-512 hash function that was developed as part of this work is available.

Table 2. Comparison of hashing programs.

	SHA-2	SHA-3	GOST 34.11-2018	Whirlpool	Nik-512	Vulnerable (SHA-1, MD-5)
HashGenerator	+	+	–	–	–	+
HashMyFiles	+	+	–	–	–	+
HashTab	+	+	–	+	–	+
QuickHash	+	–	–	–	–	+
MultiHasher	+	–	–	–	–	+
NikitaHasher	+	+	+	–	+	–

4. Discussion

The results of this research include the development of the collision-resistant cryptographic hash function Nik-512 and the implementation of the data integrity checking system NikitaHasher. Given that attacks aiming at falsifying data and forging electronic signatures pose a serious threat in the modern world, this system can be used effectively by any company to ensure data integrity. It bears mentioning that its applicability is not restricted by any type of data or any specific properties of the content of the files stored. This fact makes the implemented system promising for coming into widespread use in the nearest future. In the light of the recent advances in the development of fast and secure cryptographic hash functions, the vulnerable algorithms SHA-1 and MD-5 have no practical advantages in modern cryptography and should be replaced with more secure hash functions.

The authors give a review of modern hashing algorithms from the perspective of resistance to modern collision attacks. They conclude that today, the most dangerous type of attacks on hashing algorithms are quantum attacks because they allow for finding a collision significantly more quickly than classical attacks that do not require quantum computations. Upon investigating recent articles on quantum collision attacks and post-quantum security of hashing algorithms, the authors came to a conclusion that a cryptographic hash function could be resistant to quantum attacks if it were constructed and implemented correctly. Thanalakshmi et al., in research on a post-quantum digital signature system, discussed similar issues related to weaknesses of hashing algorithms being applied to digital signature verification taking into consideration recent advancements in quantum technologies [29]. Their research showed that it is possible to build post-quantum hash-based digital signature system.

5. Conclusions

The main result of this work is the development of a cryptographic hash function with a 512-bit hash length that is resistant to collision attacks. The cryptographic strength of the developed hash function was confirmed by testing its statistical properties and comparing them with other hashing algorithms. As a result of the comparison, it was concluded that the developed hash function better fulfills the avalanche effect requirement, and this means better resistance to collision attacks. Additionally, using the constructed hash function, a software tool for data integrity control was developed. The program provides the user with a greater choice of collision-resistant hashing algorithms than existing analogues. Another important advantage of the developed software is the lack of support for vulnerable hashing algorithms.

Further research may include constructing key derivation functions and blockchain applications based on the Nik-512 hashing algorithm and integrating the implemented software with digital signature systems.

Author Contributions: Conceptualization, L.V.C., O.A.S., N.G.L. and D.A.K.; methodology, L.V.C., O.A.S., N.G.L.; software, O.A.S., N.G.L. and D.A.K.; validation, L.V.C., O.A.S. and N.G.L.; formal analysis, L.V.C., O.A.S. and N.G.L.; investigation, L.V.C., O.A.S. and N.G.L.; resources, D.A.K.; data curation, L.V.C., O.A.S. and N.G.L.; writing—original draft preparation, L.V.C. and O.S; writing—review and editing, O.A.S. and D.A.K.; visualization, O.A.S.; supervision, L.V.C. and D.A.K.; project administration, L.V.C., O.A.S. and D.A.K.; funding acquisition, O.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Acknowledgments: Beskopylny Alexey Nikolaevich, Dmitriy Davidovich Gabrielyan.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. El-Meligy, N.E.; Diab, T.O.; Mohra, A.S.; Hassan, A.Y.; El-Sobky, W.I. A Novel Dynamic Mathematical Model Applied in Hash Function Based on DNA Algorithm and Chaotic Maps. *Mathematics* **2022**, *10*, 1333. [CrossRef]
2. Torres-Alvarado, A.; Morales-Rosales, L.A.; Algredo-Badillo, I.; López-Huerta, F.; Lobato-Báez, M.; López-Pimentel, J.C. An SHA-3 Hardware Architecture against Failures Based on Hamming Codes and Triple Modular Redundancy. *Sensors* **2022**, *22*, 2985. [CrossRef] [PubMed]
3. Bai, E.; Jiang, X.-Q.; Wu, Y. Memory-Saving and High-Speed Privacy Amplification Algorithm Using LFSR-Based Hash Function for Key Generation. *Electronics* **2022**, *11*, 377. [CrossRef]
4. Gadamsetty, S.; Ch, R.; Ch, A.; Iwendi, C.; Gadekallu, T.R. Hash-Based Deep Learning Approach for Remote Sensing Satellite Imagery Detection. *Water* **2022**, *14*, 707. [CrossRef]
5. Alotaibi, A.S. Biserial Miyaguchi–Preneel Blockchain-Based Ruzicka-Indexed Deep Perceptive Learning for Malware Detection in IoMT. *Sensors* **2021**, *21*, 7119. [CrossRef] [PubMed]
6. Seth, J.N. *Practical Cryptography in Python: Learning Correct Cryptography by Example*; Apress: Austin, TX, USA, 2019; 380p, ISBN 978-1-4842-4900-0.
7. Imam, R.; Sumagita, M. Analysis of Secure Hash Algorithm (SHA) 512 for Encryption Process on Web Based Application. *Int. J. Cyber-Secur. Digit. Forensics* **2018**, *7*, 373–381.
8. Stevens, M.; Bursztein, E.; Karpman, P.; Albertini, A.; Markov, Y. The First Collision for Full SHA-1. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2017; pp. 570–596. [CrossRef]
9. Leurent, G.; Peyrin, T. From Collisions to Chosen-Prefix Collisions Application to Full SHA-1. In *Advances in Cryptology—EUROCRYPT 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 527–555. [CrossRef]

10. Al-Odat, Z.; Khan, S. Constructions and Attacks on Hash Functions. In Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 5–7 December 2019; pp. 139–144. [CrossRef]
11. Rasjid, Z.E.; Soewito, B.; Witjaksono, G.; Abdurachman, E. A review of collisions in cryptographic hash function used in digital forensic tools. *Procedia Comput. Sci.* **2017**, *116*, 381–392. [CrossRef]
12. Tiwari, H. Merkle-Damgård Construction Method and Alternatives: A Review. *J. Inf. Organ. Sci.* **2017**, *41*, 283–304. [CrossRef]
13. Aumasson, J.P. *Serious Cryptography: A Practical Introduction to Modern Encryption*; No Starch Press: San Francisco, CA, USA, 2017; 434p, ISBN 978-1-59327-826-7.
14. Guo, J.; Liao, G.; Liu, G.; Liu, M.; Qiao, K.; Song, L. Practical Collision Attacks against Round-Reduced SHA-3. *J. Cryptol.* **2020**, *33*, 228–270. [CrossRef]
15. Boissier, R.H.; Noûs, C.; Rotella, Y. Algebraic Collision Attacks on Keccak. *IACR Trans. Symmetric Cryptol.* **2021**, *2021*, 239–268. [CrossRef]
16. Al-Odat, Z.; Ali, M.; Khan, S.U. Mitigation and Improving SHA-1 Standard Using Collision Detection Approach. In Proceedings of the 2018 International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 17–19 December 2018; pp. 333–338. [CrossRef]
17. Dobraunig, C. Security of the Suffix Keyed Sponge. *IACR Trans. Symmetric Cryptol.* **2019**, *2019*, 223–248. [CrossRef]
18. Stevens, M.; Karpman, P.; Peyrin, T. Freestart Collision for Full SHA-1. In *Advances in Cryptology—EUROCRYPT 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 459–483. [CrossRef]
19. Stevens, M. Speeding up detection of SHA-1 collision attacks using unavoidable attack conditions. In *Proceedings of the 26th USENIX Security Symposium*; Vancouver, BC, Canada, 16–18 August 2017, USENIX: Berkeley, CA, USA, 2017; 19p, ISBN 978-1-931971-40-9.
20. Hosoyamada, A.; Sasaki, Y. Finding Hash Collisions with Quantum Computers by Using Differential Trails with Smaller Probability than Birthday Bound. In *Advances in Cryptology—EUROCRYPT 2020*; Springer: Cham, Switzerland, 2020; Volume 12106, pp. 249–279. [CrossRef]
21. Hosoyamada, A.; Sasaki, Y.; Xagawa, K. Quantum Multicollision-Finding Algorithm. In *Advances in Cryptology—ASIACRYPT 2017*; Springer: Cham, Switzerland, 2017; pp. 179–210. [CrossRef]
22. Mittelbach, A. *The Theory of Hash Functions and Random Oracles*; Springer Nature Switzerland: Cham, Switzerland, 2021; 780p, ISBN 978-3-030-63287-8.
23. Chailloux, A.; Naya-Plasencia, M.; Schrottenloher, A. An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography. In *Advances in Cryptology—ASIACRYPT 2017*; Springer: Cham, Switzerland, 2017; pp. 211–240. [CrossRef]
24. Hosoyamada, A.; Sasaki, Y. Quantum Collision Attacks on Reduced SHA-256 and SHA-512. In *Advances in Cryptology—CRYPTO 2021*; Springer: Cham, Switzerland, 2021; pp. 616–646. [CrossRef]
25. Sakan, K.; Nyssanbayeva, S.; Kapalova, N.; Algazy, K.; Khompysh, A.; Dyusenbayev, D. Development and analysis of the new hashing algorithm based on block cipher. *Eastern-European J. Enterp. Technol.* **2022**, *2*, 60–73. [CrossRef]
26. Tutueva, A.V.; Karimov, A.I.; Moysis, L.; Volos, C.; Butusov, D.N. Construction of one-way hash functions with increased key space using adaptive chaotic maps. *Chaos Solitons Fractals* **2020**, *141*, 110344. [CrossRef]
27. Wang, Y.; Chen, L.; Wang, X.; Wu, G.; Yu, K.; Lu, T. The design of keyed hash function based on CNN-MD structure. *Chaos Solitons Fractals* **2021**, *152*, 111443. [CrossRef]
28. Da Silva, E.V. Viktoria: A New Architecture for Hash Functions. In Proceedings of the Future Technologies Conference (FTC), Vancouver, BC, Canada, 28–29 November; Springer: Cham, Switzerland, 2021; Volume 3, pp. 251–271. [CrossRef]
29. Thanalakshmi, P.; Anitha, R.; Anbazhagan, N.; Park, C.; Joshi, G.P.; Seo, C. A Hash-Based Quantum-Resistant Designated Verifier Signature Scheme. *Mathematics* **2022**, *10*, 1642. [CrossRef]