BIO Web of Conferences **97**, 00093 (2024) https://doi.org/10.1051/bioconf/20249700093
*ISCKU 2024*

# Comparing two cryptographic hash algorithms: sha-512 and whirlpool- a case study on file integrity monitoring

*Farah* Al-Shareefi[1]* *and Zahraa* Al-Barmani[2]
[1,2]Computer Science Department Babylon University, Hilla, Iraq

**Abstract.** SHA-512 and Whirlpool are two distinct hashing algorithms in the cryptography domain. Although they are different, these methods share some characteristics; such as both of them produce secure digests of the same size. In addition, being increasingly used for sustaining the integrity of confidential files. As a result, it is crucial to explore a method to contrast and compare their performance. This paper presents a comparative analysis of SHA-512 and Whirlpool, in terms of: time consumption, avalanche effect, and resistance to collision. It also investigates their suitability in the context of File Integrity Monitoring. Our results reveal that Whirlpool outperforms SHA-512 in the avalanche effect scenario, whereas a comparable resistance to the collision behavior is exhibited. Regarding time consumption, it is slower than SHA-512. However, this positively reflects its resistance to brute force attacks.

## 1 Introduction

Sustaining the integrity of critical and private files is a paramount necessity in light of intrusive access to their con- tents. The intrusive access refers to the malevolent actions performed by attackers, such as stealing confidential data, subverting files to conceal their traces, and so on. One of the security practices that ensures the integrity of these files is File Integrity Monitoring (FIM) [5]. In essence, FIM is a cybersecurity solution that deals with overseeing momentous files and instantly detecting any alterations [10]. The FIM process relies on using one of the crypto-primitives known as a hashing algorithm [11], that mathematically transforms a finite input to a unique output with a constant size, called hash value or sometimes digest. Based on the utilized hashing algorithm, FIM generates hashes of files at the same time that they are created. Subsequently, these hashes are cross-referenced with the present hashes of the files. If they are identical, it indicates that the integrity of the file is to be trusted; otherwise, it is not trustful.

Three essential features must be involved in any hashing algorithms, including: quick consumption time, avalanche effect, resistance to collision [9]. The first one denotes the amount of time required to calculate the hash value for a given input. The second feature means that the input and output are not correlated, i.e., a minor change to the input causes a significant change to the output. While the third indicates the computational impossibility of locating two inputs that collide. These features consequently play a vital role in enhancing the performance in FIM.

Although various types of cryptographic hashing algorithms have been developed for distinct security issues, most of the attention is paid towards verifying the integrity of files. Secure Hash Algorithm 512 (SHA-512) [4] and Whirlpool [3] are examples of these algorithms. Both of them are extensively utilized in many cryptography applications. Furthermore, they are similar in certain ways; for example, they both generate secure digests of 512 bits length. Consequently, it is advan- tageous to construct a method to juxtapose them. The main focus of this paper is to compare and assess SHA-512 with Whirlpool according to the perspective of using them in FIM. In fact, the comparative assessment is performed with regard to the time consumption, avalanche effect, and resistance to collision. We believe that this is the first piece of work that evaluates the suitability of SHA-512 and Whirlpool for FIM. Our comparative analysis not only provides guidance for current FIM but also offers insights for other security practices and technologies within the field of cryptography research.

The rest of the paper is structured as follows: Section II reviews the previous researches related to SHA-512 and Whirlpool algorithms. An overview of both algorithms is pro- vided in Section III. Section IV describes the main concept of FIM. Section V details the comparison metrics. Our systematic methodology is elucidated in Section VI, While Section VII discusses the obtained results. Finally, Section VIII concludes the paper.

*Corresponding author:wsci.farah.mohammed@uobabylon.edu.iq

## 2 Literature Survey

The authors in [12] conducted a comparison testing between two popular hashing algorithms: MD5 and SHA 512, as an attempt to ensure the security of passwords in the web-based applications. Penetration testing, that makes use of brute-force attacks, and User Acceptance Testing (UAT), which involves completing a questionnaire were used for the testing process.

The results of penetration have shown that the SHA 512 algorithm is more reliable as the brute force attack takes longer time to locate the original text of the hash value. The results of the UAT have generated a strongly agreed percentage at 86.00.

The limitations of the standard algorithms, including: MD5, SHA-1, SHA-2, SHA-3, and RIPEMD, against common at- tacks were discussed by the authors of [13]. According to their results, these algorithms are vulnerable to collision attacks.

The authors of [14] aimed at elucidating the rationale behind the discontinuation of MD5 and SHA1 hash functions in digital forensics. This decision stems from the fact that MD5 and SHA1 have been deemed compromised due to a collision attack in cryptography. They also mentioned some recommendations for enhancing the performance of MD5 and SHA-1 functions, along with suggestions to use alternative options, such as SHA-256, RIPEMD-160, and the Whirlpool hash algorithm, instead.

In [15] an examination of applying hashing functions to Blockchain systems. The research's focus was directed towards the outcomes of the comparative analysis pertaining to the primary characteristics of hash functions, as well as the suggestions for their implementation. It was advised to utilize 512-bit hash functions for implementation in contemporary blockchain systems, such as SHA-512 and Kupyna 512.

### 2.1 Cryptographic Hash Algorithms

A cryptographic hash algorithm indicates a mathematical algorithm employed in the field of cryptography. This al- gorithm processes an input of arbitrary length to produce a determined length output, known as hash value [2]. Hashing is alternatively referred to as Digest, Message Digest, Checksum, and other similar terms. A variety of hashing algorithms are available, but this paper focuses on the below two types which are technically compared in table I:

#### *2.1.1 Secure Hash Algorithm 512 (SHA-512)*

The SHA-512, which stands for Secure Hash Algorithm 512-bit, is a cryptographic algorithm that employs a one- way hash function developed by Ron Rivest [4]. It is one of the five members that constitute the SHA family. It was initially developed by the National Security Agency (NSA) and officially released by the National Institute of Standards and Technology (NIST) in 2001. The SHA-512 generates a predetermined-size output of 512 bits, which is equivalent to 64 bytes. In addition, an input message of no more than $2^{128}$ bits in length is accepted by SHA-512. It processes a block size of 1024 bits during the computation.

#### *2.1.2 Whirlpool*

Whirlpool is a hashing-based block cipher algorithm that was created by Vincent Rijmen and Paulo Barreto, in 2000 [3]. It has a similar structure to the Advanced Encryption Standard (AES) method. It has received the endorsement of the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) initiative. An input message with a maximum length of $2^{256}$ bits or less is digested by Whirlpool in 512 bits. A block size of 512 bits is processed throughout the computation.

**Table I**.Comparison of Technical Characteristics for SHA-512 and Whirlpool

| **Algorithm** | **Input Length (in bits)** | **Block Size (in bits)** | **Hash Value Length (in bits)** |
|---|---|---|---|
| SHA-512 | $< 2^{128}$ | 1024 | 512 |
| Whirlpool | $< 2^{256}$ | 512 | 512 |

### *2.1.3 File Integrity Monitoring: Case Study*

File Integrity Monitoring (FIM) is a security mechanism that entails observing and demonstrating the integrity of sensitive files in order to address unauthorised modifications [5]. Its main objective is identifying when, how, and sometimes who made the modification. It can also generate alerts that facilitate further analysis of the insecure incident.

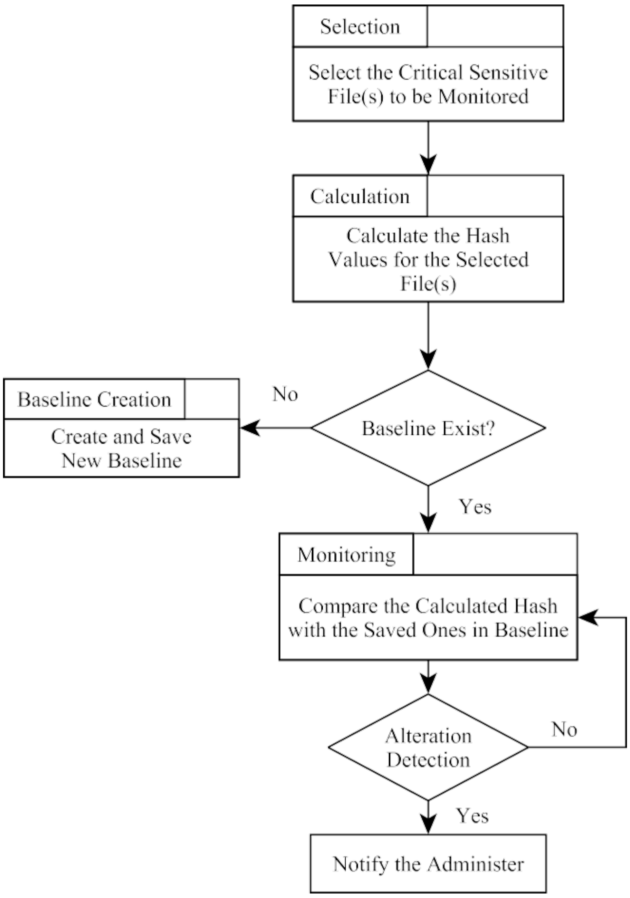


**Fig 1.** A Typical File Integrity Monitoring (FIM) Process

Cryptographic hash algorithms play an essential role in File Integrity Monitoring (FIM) since they provide a means for ensuring the integrity of files and detecting unauthorized alterations [7].

In general, file integrity monitoring strategies can be classi- fied into two categories : offline (periodic detection mode), and online (continuous detection mode) [6]. The primary premise behind both monitoring categories is to compare the current hash value(s) of the most recent file version(s) to the baseline, which is a database consisting of precomputed and trusted digest values, see Fig. 1 which illustrates the typical process of an FIM system. The first phase in this process is the Selection of the file(s) which will be monitored for the changes. Next the Calculation phase is initiated to calculate the distinctive hash value based on the content of the selected file(s). After that, the New Baseline Creation is processed if it has not been created before; otherwise, the Monitoring phase is undertaken. During the baseline creation phase, the hashing algorithm generates a unique identifier for each file in its original and unchanged condition. The created values can be used as references for further comparisons in the subsequent monitoring. In the monitoring, the FIM system re-computes the hash values for the current contents of each selected file. FIM compares the recent computed hash values with the saved ones in the baseline. If the hash values match, the file is considered unmodified. If they differ, it indicates a potential modification and an administer must be notified.

## 3 Comparison Metrics

To evaluate SHA-512 and Whirlpool functions for File Integrity Monitoring (FIM), we examine three key metrics, which are as follows:

**Time Consumption** metric which refers to the total time needed to complete the hashing process. The computed time is affected by the file size and the complexity of the applied hashing algorithm.

**Avalanche Effect (AE)** metric denotes the property of a minor alteration in the input file of the hash func- tion which causes a substantially distinct, unpredictable output (hash value), and unrelated to the original hash [8]. The choice of a cryptographic hash function with a strong avalanche effect is crucial in FIM to effectively thwart any attempt made by an attacker to alter a monitored file. An effective cryptographic hash function exhibits a high sensitivity to changes in the input. The AE can be calculated using the Hamming distance that computes the number of the bits that are different.

**Resistance to Collision** metric which means that a hash function is considered collision resistant if it is difficult to discover a pair of files with dissimilar contents that hold an identical hash value [1]. In other words, the collision occurs only if the following property is satisfied:

$$f_1 = f_2 \wedge h(f_1) = h(f_2) \qquad (1)$$

where $f_1$ and $f_2$ are two input files with distinct con- tents, $h(f_1)$ and $h(f_2)$ are the produced hash values by applying a hash function $h$ to the given files $f_1$ and $f_2$, respectively.

## 4 Methodology

In this part, we will elucidate the methodology utilised to systematically evaluate and contrast the two hashing algo- rithms, SHA-512 and Whirlpool.

**Dataset Establishment:** In this stage, the features of the dataset that provided the basis for this study is outlined. Typically, the dataset is composed of text files that are arbitrarily generated with different sizes using the Python programming language. In essence, each generated file contains a string of random digits and alphabets based on the random.choices() function, which is from the random built-in module.

**Time Consumption Evaluation:** This stage is dedicated to calculating the duration of the hashing algorithm, as rapid computation is advantageous for file integrity systems.

**Avalanche Effect (AE) Evaluation:** To evaluate the AE of each algorithm, we apply Algorithm 1 that takes the Binary representation of two input messages, and produces the number of bits that are changed. Essen- tially, the hashing algorithm is implemented to each given message and Hamming distance is then computed between the two hash values.

**Algorithm 1** Calculating the avalanche effect
**Input:** X and Y: A Binary representation of two input files,
$h_A$: the applied hashing algorithm **Output:** n: total number of bits that are distinct $H_1 \leftarrow h_A$(X)
$H_2 \leftarrow h_A$(Y)
$n \leftarrow 0$
**for** *i=1 to 512* **do**
$n \leftarrow n + (H_1(i) \oplus H_2(i))$

**Resistance to Collision Evaluation:** This evaluation is performed by conducting Algorithm 2.

**Algorithm 2** Determining the number of collisions

**Input:** files: A set of text files
**Output:** collision-no filehash-baseline ← {} collision-no ← 0
**for** *i=1 to size(files)* **do**
hashvalue ← hash(files{i})
**if** *hashvalue* ∈ *filehash-baseline* **then**
collision-no ← collision+1
**else**
filehash-baseline{i} ← hashvalue

This algorithm takes a set of random text files as an input, and returns the total amount of collisions. In general, it initially computes the hash value for each file in the given set. Next, it checks whether the computed hash already exists in the baseline or not to indicate a collision.

Once the evaluation phases are complete, it is possible to determine which algorithm is superior.

## 5 Results and Discussion

As our goal is to gain a comprehensive understanding of the computational efficiency and performance for both SHA-512 and Whirlpool in the FIM context, we have subjected these algorithms to three empirical tests: time consumption, avalanche effect, and collision resistance. The tests were carried out on a 64-bit Windows platform supplied with a 2.50 GHz Intel(R) Core(TM) i5-6300U CPU. The established tests together with the compared hashing algorithms are implemented in the Python programming language. The following subsections provide a detailed evaluation of both hash algorithms:

## 6 Time Consumption Evaluation

The hashing time for each algorithm is assessed according to a number of sensitive files with variable sizes, including: 5KB, 10KB, 50KB, 100KB, 500KB, and 1MB. In essence,

our experiments indicate that the hashing time for both SHA- 512 and Whirlpool grows as the file size expands, with SHA- 512 being faster than Whrilpool, see Table II which illustrates the required execution time for SHA-512 and Whrilpool. The results demonstrating that SHA-512 is quicker than Whirlpool have both positive and negative implications. In other words, SHA-512 reduces the processing time for the hash value, however, it enables an attacker to execute brute force attacks at an accelerated pace.

**Table 2.**Timing Comparison for Hashing Sensitive Files

| Test No. | File Size | Execution Time/ms | |
|---|---|---|---|
| | | SHA-512 | Whirlpool |
| 1 | 5KB | 0.983715 | 0.997066 |
| 2 | 10KB | 0.989437 | 0.997782 |
| 3 | 50KB | 0.998497 | 0.999689 |
| 4 | 100KB | 1.033068 | 1.982689 |
| 5 | 500KB | 3.007412 | 8.975267 |
| 6 | 1MB | 5.028248 | 16.953230 |

### 6.1 Avalanche Effect Evaluation

The AF is tested by repeating Algorithm 1 a million times to get a mean of AE for both SHA-512 and Whirlpool. In our tests, the number of flipped bits belonging to the input file contents varies between one and four bits. Table III illustrates the results of four tests for evaluating the AE values. From Table III, we can deduce that the AE values for both algorithms are nearly identical, although the Whirlpool algorithm exhibits a slightly higher AE value in comparison to the SHA-512 algorithm. This means that both algorithms possess the capability to identify and react to unauthorized alterations in files by guaranteeing that even the most minor update to the file contents leads to a substantially distinct hash value.

**Table 3**.Comparative Analysis of Avalanche Effect for SHA-512 and Whirlpool Over 1000000 Tests

| Test No. | No. of Modified Bits | AE | |
|---|---|---|---|
| | | SHA-512 | Whirlpool |
| 1 | 1 | 0.499991 | 0.500001 |
| 2 | 2 | 0.500029 | 0.500038 |
| 3 | 3 | 0.500018 | 0.500001 |
| 4 | 4 | 0.500051 | 0.500005 |

### 6.2 Resistance to Collision Evaluation

In order to evaluate the collision resistance of both SHA- 512 and Whirlpool, algorithm 2 is conducted over six tests. The number of generated file in each test is as follows: 50, 100, 1000, 10,000, 100,000, 1,000,000. The size of files in these tests is 1KB. It is found that SHA-512 and Whirlpool exhibit a high level of collision resistance, since there have been no collisions discovered in any of the conducted tests.

## 7 Conclusion

A systematic methodology for analytically comparing two cryptographic hashing algorithms, namely SHA-512 and Whirlpool, with respect to FIM, have been presented in this paper. The comparison has been conducted based on three metrics: time consumption, avalanche effect, and resistance to collision. According to our findings, Whirlpool performs better than SHA-512 in terms of the avalanche effect and it shows similar resistance to collision. Despite being slower than SHA-512, it is a good indicator of Whirlpool's withstanding brute-force attacks. As a result, the Whirlpool can increase the effectiveness of file integrity monitoring. As a future step, we will investigate how their performances can be affected by the cybersecurity attacks, like pre-image attack, in the same FIM context.

## References

1. X. Wang and H. Yu, "How to break MD5 and other hash functions," in Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–35.
2. J. Pieprzyk and B. Sadeghiyan, Eds., Design of hashing algorithms. New York, NY: Springer, 2014.
3. P. S. L. M. Barreto and V. Rijmen, "The Whirlpool hashing function". Primitive submitted to NESSIE, September 2000, revised on May 2003, http://planeta.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html
4. U S Department of Commerce and National Institute of St And Tech- nology, Secure hash standard (shs): Federal information processing stan- dards publication 180-4. North Charleston, SC: Createspace Independent Publishing Platform, 2012.
5. S. K. Peddoju, H. Upadhyay, and L. Lagos, "File integrity monitoring tools: Issues, challenges, and solutions," Concurr. Comput., vol. 32, no. 22, 2020.

6. Z. H. Abdullah, N. I. Udzir, R. Mahmod, and K. Samsudin, “Towards a dynamic file integrity monitor through a security classification,” Interna- tional Journal on New Computer Architectures and Their Applications, 2011.
7. S. K. Peddoju, H. Upadhyay, and L. Lagos, “File integrity monitoring tools: Issues, challenges, and solutions,” Concurr. Comput., vol. 32, no. 22, 2020.
8. K. Donnelly, Privacy, trust and elements of cryptography in computer security. North Charleston, SC: Createspace Independent Publishing Platform, 2017.
9. Y. Dodis, Theory of Cryptography. Heidelberg: Springer, 2015.
10. A. Pinheiro, E. D. Canedo, R. T. De Sousa, and R. De Oliveira Albuquerque, “Monitoring file integrity using blockchain and smart contracts,” IEEE Access, vol. 8, pp. 198548–198579, 2020.
11. J. Pieprzyk and B. Sadeghiyan, Eds., Design of hashing algorithms. New York, NY: Springer, 2014.
12. M. Sumagita, I. Riadi, J. P. D. S. Sh, and U. Warungboto, “Analysis of secure hash algorithm (SHA) 512 for encryption process on web based application,” International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 7, no. 4, pp. 373–381, 2018.
13. A. Maetouq, S. M. Daud, N. A. Ahmad, N. Maarop, N. N. A. Sjarif, and H. Abas, “Comparison of hash function algorithms against attacks: A review,” International Journal of Advanced Computer Science and Applications, no. 8, 2018.
14. S. D. Almotiri, Forensic Hash Value Guidelines: Why Md5 and SHA1 should no longer be used and a recommendation for their replacement, 2022.
15. P. V. Kravchuk, I. D. Gorbenko, and A. I. Pushkaryov, “Analysis of application of hash function in blockchain technology,” Applied radioelectronics, vol. 17, no. 3, 2018.