International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 13, No. 3, November 2024, pp. 502~511
ISSN: 2089-4864, DOI: 10.11591/ijres.v13.i3.pp502-511

Comparative study of password storing using hash function with MD5, SHA1, SHA2, and SHA3 algorithm

Parinya Natho, Suwit Somsuphaprungyos, Salinun Boonmee, Sangtong Boonying
Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi, Phra Nakhon Si Ayuttaya, Thailand

Article Info

Article history:

Received Oct 20, 2023
Revised Feb 21, 2024
Accepted Mar 21, 2024

Keywords:

Hash fuction
Message digest algorithm
Password attack
Password secure storing
Secure hash algorithm

ABSTRACT

The main purpose of passwords is to prevent unauthorized people from accessing the system. The rise in internet users has led to an increase in password hacking, which has resulted in a variety of problems. These issues include opponents stealing a company's or nation's private information and harming the economy or the organization's security. Password hacking is a common tool used by hackers for illegal purposes. Password security against hackers is essential. There are several ways to hack passwords, including traffic interception, social engineering, credential stuffing, and password spraying. In an attempt to prevent hacking, hashing algorithms are therefore mostly employed to hash passwords, making password cracking more difficult. In the suggested work, several hashing techniques, including message digest (MD5), secure hash algorithms (SHA1, SHA2, and SHA3) have been used. They have become vulnerable as a result of being used to store passwords. A rainbow table attack is conceivable. Passwords produced with different hash algorithms can have their hash values attacked with the help of the Hashcat program. It is proven that the SHA3 algorithm can help with more secure password storage when compared to other algorithms.

Corresponding Author:

Sangtong Boonying
Department of Information System and Business Computer
Faculty of Business Administration and Information Technology
Rajamangala University of Technology Suvarnabhumi
190 Moo 3 Tha Wa Su Kri, Phra Nakhon Si Ayuttaya, Thailand
Email: sangtong.b@rmutsb.ac.th

1. INTRODUCTION

At present, advances in communication technology, transmission, and storage in electronic form are now considered very important. When the data sent from the sender to the receiver must be secured, the main idea about data security is data confidentiality is the ability to maintain confidentiality so that unauthorized persons can read information. The mechanism that will help achieve the objectives in this area is the encryption of the data itself. data integrity is the ability to maintain the integrity and integrity of the data so that the data will not be lost or altered in any way. The mechanism that helps to achieve this objective is the hash function process and availability is the ability to identify the person who has access to information. The simplest mechanism to achieve this objective is the identity verification process authentication to access the system. Authentication is regarded as a crucial security measure for all IT systems. It serves as a safeguard that only authorized users are permitted to access and utilize the systems. Additionally, it may be utilized to thwart phishing [1] and information fabrication [2] assaults.

The most widely used authentication technique in use today is the password. It is the primary technique for confirming a user's authorization before allowing them access to an IT system [3], [4]. Many people have developed strategies to assist make passwords more safe because of how important they are. For instance, some organizations only permit the use of passwords for a certain period of time, while others encourage their staff to create passwords that are challenging to decipher. Sadly, a lot of individuals still prioritize convenience over security. In other words, a lot of people still use weak passwords, and a lot of people even use the same password across many accounts. The brute-force attack is one of the frequent attacks that risk user-sensitive data. Password guessing utilizing automated software that creates several passwords in order to access message content is known as the brute force attack approach [5]. Therefore, it is the responsibility of the systems or service providers to figure out how to protect such credentials. Without a safe method of saving or storing passwords, it is feasible for an adversary to obtain the credentials and use them for malicious purposes.

Cryptographic hash algorithms like message digest (MD5) algorithm [6] and secure hash algorithm (SHA) [7] are a common technique that individuals have used to keep passwords more secure. In other words, password hash values are saved in the database rather than plaintext passwords. The issue with these routines is that their primary purpose of message integrity checking requires them to be extremely quick. Because it enables a rainbow table assault [8] or a dictionary attack [9] and the brute force attack [10], MD5 and SHA1 speed is seen as the primary enemies of passwords. With current technology, a graphic processing unit (GPU) can compute more than 100 million MD5 hashes per second [4]. Furthermore, among of the most potent and dangerous assaults on the internet are distributed denial of service (DDoS) and denial of service (DoS), which bombard servers with massive volumes of traffic in an attempt to prevent users from accessing them [11]. This means that passwords with one to six characters may be broken quickly, but passwords with more characters would take longer to crack. A rainbow table has been made as a result. A rainbowtable is a large database that contains already-calculated hash values for potential plaintext patterns [12]. This implies that in order to determine the associated plaintext, an adversary merely has to compare the hash values of the passwords in a password database with those in the rainbow table. This suggests that employing only cryptographic hash methods to maintain passwords is insufficient. Attackers were able to gain access to a record of 15 million SHA1 hash values of compromised passwords stored in the social network's database. This is a well-known case [13]. The rainbow table was used to search for associated plaintext passwords once the hash values were entered. Later, the social network must inform users that security improvements are needed, such as updating passwords and finding better ways to store passwords. Choose how to add salt. This is a random integer that is added to the front of the password before the hash function is considered more secure. In this study We will show that this also does not withstand attacks.. Searching for algorithms that help provide secure, dynamic password storage. Instead, connecting them together is the main goal of this study. Each password's storage usage is different. As a result, password storage will be able to store passwords more securely.

From these safety concepts and objectives in terms of the accuracy of the data, there are techniques that can be used to verify the accuracy of the data. Process hash function which has the property to reduce the size of the data. When data is hashed via hash function The results are specific to each field. It can also be used for security purposes. This is used in conjunction with storing passwords to verify access to the system. Authentication methods are an easy way to verify access. By verifying this identity with a username and password, the information used to verify this identity should be kept confidential. Safe from malicious attacks because if an attacker or a malicious person can know the information used for this authentication. The stored information may no longer be confidential. In most password storage systems such information will be stored in the system database. By setting a password to access the system, there will be a mechanism to help set the password to be more secure, such as setting the length of the password. Setting up a password pattern or even taking time to change a new password. As a result, this study focuses on determining the modification detection codes hash function's processing speed performance as a benchmark for assessing and contrasting the effectiveness of saving passwords in the database system. include MD5, SHA1, SHA2, and SHA3 algorithms, all of which have an impact on the security of data validation and storage, the production of digital signatures, and even the saving of passwords as a hash function.

The rest of this paper is organized as follows. Section 2 literature review of background knowledge and existing password storage systems, as well as an analysis of each. Section 3 methodology and proposed a scheme for value into a password. Section 4 shows the results of algorithms performance. Section 5 gives a summary of the paper and a plan on the future work.

2. LITERATURE REVIEW

This section gives a summary of the fundamental understanding of password storage techniques as well as the state of the art in password research. The section starts with a quick introduction of cryptographic hash functions, which are currently popularly employed to store passwords. Then we present studies of the password-storage techniques now in use. The latest research on passwords and password-related subjects is also offered.

2.1. Hash function cryptography

Cryptographically, a hash function is any function that can be used to convert data of any size into fixed-sized data values. Some hash functions can also output variable length data [14]. The result of a hash function is known as hashes, hash values, hash codes, digests, or simply hashes. The values are commonly used to index a hash table, which is a fixed-size table. The process of hashing, also known as scatter storage addressing, involves using a hash function to index a hash table. To access data in a tiny and practically constant amount of time every retrieval, hash functions and the accompanying hash tables are employed in data storage and retrieval applications. They just need a little amount of storage space, somewhat more than what is needed for the actual data or records. Hashing is a method of accessing data that is both computationally and storage-efficient. It circumvents the arbitrarily long storage requirements of direct access to state spaces with large or variable-length keys, as well as the irregular access times of organized trees and sorted and unordered lists.

2.2. MD5 algorithm

Rivest [6] developed the message digest algorithm 5, or MD5. Among the MD family of hash algorithms, it is the most often utilized. A 128-bit hash value is produced by MD5 using any length of input. Over the years, MD5 has gained a lot of popularity, although flaws have been revealed where collisions might be located in a reasonable period of time. MD5 is the most often used algorithm for verifying file integrity. It is also used in other security protocols and applications, such as secure shell (SSH), secure socket leyer (SSL), and internet protocol security (IPSec). Some applications improve the MD5 technique by including a salt value into the plaintext or utilizing the hash function multiple times. Dobbertin revealed in 1996 that MD5 was the subject of collision attacks. Additionally, successful collision attacks against MD5 were documented in [15]. Prior research has also shown that MD5 collision attacks have improved [16], [17].

2.3. Modern hash algorithm

The family of cryptographic hash methods known as SHA, which consists of three standards such as SHA-1, SHA-2, and SHA3, is now the most widely used. The Federal Information Processing Standard (FIPS PUB) 180-1 [18] has a description of the SHA-1 algorithm, which was created in 1995. The frayed prefix collision attack was created and put into use in 2017 [19]. The computational complexity of the attack is 263 [10]. As SHA-1 is no longer believed to be resistant to collision attacks, using it is no longer advised [20]. By 2022, no upgrade will enable any collision attack to be faster than a general assault due to the collision-resistant nature of the SHA-2 algorithm [21]. Although collision attacks on SHA-2 with fewer rounds have been proposed, they are only beneficial in theory since they cannot be utilized against the complete SHA-2 algorithm [22]. Because SHA-2 is built on the Davis-Meyer structure, theorists were deeply concerned that attacks based on differential cryptanalysis may be created against it. This finding led to the creation of the new SHA-3 standard [23]. The secure hash algorithm with the SHA-3 algorithm was created by a group of four scientists led by Joan Dymen in the 2012 special competition of the National Institute of Standards and Technology. Keccak became the winner. This function is used to develop the SHA-3 algorithm encryption standard [24].

2.4. Password storing and related work

The primary issue examined in this work is comparing algorithms to safely store passwords, and this has to be emphasized once again. In addition to the basics covered in previous sections, we'll let take a look at what other researchers have done to improve password security and secure password storage. Before a suggested comparison technique for storing passwords is considered [25], it is required to first look at the currently used approaches. Now let's go over and evaluate each technique individually.

2.4.1. Plaintext password storing

A database may store a password in its unencrypted state, which is the easiest method to do so. This indicates that a human-readable format for users and passwords is used to store them in the database. The password database will also record a password with the value 1,234, for instance, if it is 1,234. When a user signs in, the system will ask for their username and password, which it will then compare to the database to verify whether they match. If so, access to the system will be granted to the user. In terms of security, this is the worst way to keep a password safe. It's because, in the event that a threat actor gained access to the password database, they would be able to read every user's password right away. Therefore, every password would be vulnerable.

2.4.2. Encrypted password storing
Encryption has become popular as a technique to lessen the chance that passwords may be made public. Using a secret key, the function of encryption converts plaintext into ciphertext. As a result, an attacker would not be able to examine the passwords in plaintext even if they had access to the password database. Cryptotext would be the sole place to view passwords.

This can seem safe at first glance. However, this strategy has a flaw. It is a challenge because secret keys, or the keys used to encrypt and decrypt passwords, are often kept in the same database as passwords. This implies that if the password database was compromised, the attacker would also be able to gain access to the stored keys. Therefore, he or she might use it to decode every password that was encrypted in ciphertext and get the passwords in plaintext format. As a result, this approach is likewise seen as unsafe.

2.4.3. Hash password storing
Password hashing is the process of entering a password into a cryptographic hash algorithm like MD5. Following processing and output scrambling, it looks to be some random value. For instance, while processing the 1234 password input using the MD5 technique, we get the corresponding hash value of 81dc9bdb52d04dc20036dbd8313ed055.

This strategy is commonly used in password management systems. This is because the database only stores the password's hash value. When a user registers with the system, the password hash value is produced and compared to the value kept in the database, rather than being stored in plain text format. If both numbers match, the system accepts the password and allows the user to log in. If the passwords do not match, the user must try to log in again.

The storage of passwords using a hash function seems to be a secure practice. This is due to the fact that a password will not be seen in plaintext even if an attacker has access to the password database. Large social networks' users' passwords have, however, been exposed to the public in a number of high-profile situations in recent years. This is made feasible by the rainbow table attack or technique, which is effectively a pre-calculated hash value of potential plaintext [26], [27]. As a result, if password hash values are revealed, an attacker can use the rainbow table to find the matching plaintext passwords.

Nevertheless, not every combination of plaintext or password has been discovered in the rainbow table. When rainbow table hash values are missing, this indicates that they are either unpre-computed plaintext or the hash values of lengthy, difficult passwords. However, as the rainbow table gets bigger over time, more passwords will inevitably be discovered.

2.4.4. Double hash password storing
This approach is comparable to the password hashing approach described above. The difference is that a different hash function or the same hash algorithm is used to hash the given hash value once more, as opposed to only hashing a password once. The amount of hash iterations must be known in order to create a database resembling the rainbow table, even if this approach makes it harder for the attacker to obtain the passwords.

2.4.5. Salted hash password storing
There has to be a solution to assuage the worry caused by the rainbow table. The introduction of a salt value was motivated by [25], [28]. Before hashing a password, a random string of characters or numbers appended to the start or end of the password is known as a salt. To put it another way, we compute h(saltabpassword) rather of just hashing a password, h(password). Each password is given a unique salt value in this manner, and each password is also saved in clear text in the same password database. This implies that an attacker could still use the rainbow table to discover the plaintext password if they knew the salt value and its location.

Many websites and organizations have recently started using this technique to safeguard user passwords. However, the salt value is frequently inserted either at the beginning or end of a password. We believe that the position of the salt value can influence how securely passwords are kept. Finding a technique to set the salt value so that saved passwords grow harder to break while maintaining a reasonable level of performance is therefore important. This research's primary goal is to achieve this.

3. METHOD
3.1. Approach for securely storing passwords
In the event that a user registers their password, it is crucial to verify the quality of the supplied password before keeping it. To guarantee that it is not too simple to be compromised, we advise doing this. According to the password quality index developed by Ma *et al.* [29], a strong password must include at least

eight characters, three of which must be special characters. The password must also contain some digits. Table 1 provides a summary of the criteria. In the following phase, we will make advantage of this.

As previously indicated, Ma *et al.* [29] research discovered that the length of time needed to decipher a password may be used to gauge its strength. In addition, they came to the conclusion that a password should include at least eight characters and at least three special characters in it. A stronger password should also include digits. In our design trials, these ideas and character combinations will be employed. Table 2 provides a list of the characters that will be employed.

The next stage is to build a rainbow table with an Intel Core i7 central processing unit (CPU) clocked at 2.50 GHz and 24 GB of random access memory (RAM), then install the Kali Linux operating system on virtual box software. The purpose of this is to ascertain the minimum password length necessary to withstand the rainbow table attack. The rainbow table was created using Rainbowcrack version 1.8. It was also used to calculate how long it would take to complete the rainbow table using various password lengths. Additionally, the time needed to build a rainbow table using a GPU with the same password sizes is contrasted to that of the CPU. This data was taken from Ferrara [30]. Table 3 shows the time it took to create a rainbow table with the CPU and GPU in this experiment.

Table 1. Criteria for secure password

Criterion	Property
The quantity of special characters	3 minimum characters
The quantity of numberical values	1 minimum number
The quantity of letters values	1 minimum letter
The quantity of pasword length	8 or more characters minimum

Table 2. Character sets for passwords

Type of character sets for passwords	Characters
The special characters	!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
The numberical values	0123456789
The lowercase letters	abcdefghijklmnopqrstuvwxyz
The capital letters	ABCDEFGHIJKLMNOPQRSTUVWXYZ

Table 3. The time required to create a rainbow table

Length of passwords	Number of posibilities	Computations time of CPU	Computations time of GPU
4 characters	35,153,041	3 minutes	0.0083 seconds
5 characters	2,706,784,157	3.75 hours	0.64 seconds
6 characters	208,422,380,089	12 days	49 seconds
7 characters	16,048,523,266,853	2.5 years	1.06 hours
8 characters	1,235,736,291,547,681	195 years	3.4 days
9 characters	2 ^ 53.6112	4 years 287 days	16 months 40 seconds
10 characters	2 ^ 59.5654	297 years	3 years 123 days

We may gain a basic understanding of the appropriate input sizes by looking at the results in Table 3. To put it another way, Table 3 shows that it is exceedingly hard to construct a full rainbow table, that is, a table with every possible combination of plaintext and hash value, if the input is nine characters or longer. The outcome of Table 3 also shows that approximately the same amount of time was required to hash inputs with lengths ranging from 9-10 characters.

3.2. Proposed scheme hash algorithm

In this step, the user's input pattern is used to create a hash for the input password using an algorithm such as MD5, SHA1, SHA2, and SHA3. In the next step, the resulting hash value is stored in the tables created for each algorithm in MySQL so that the strength of the hash values can be compared. The method used in this study to arrange the results, which involves storing each hash value for each hashing technique in a separate table to improve results organization, has not been discussed in any of the relevant publications. The following step is shown in Figure 1.

The researcher processed the original text in each format in the next steps. The hash value of the function is the result. After that, attack tests are performed using the hash value obtained from each function. To confirm the safety of each type of method. To compare the effectiveness of storing passwords with the most secure method. This is consistent with the research of the past [31], [32] which stores the results of each hash value function. The test results and a summary of the experimental results will be presented in the next section.

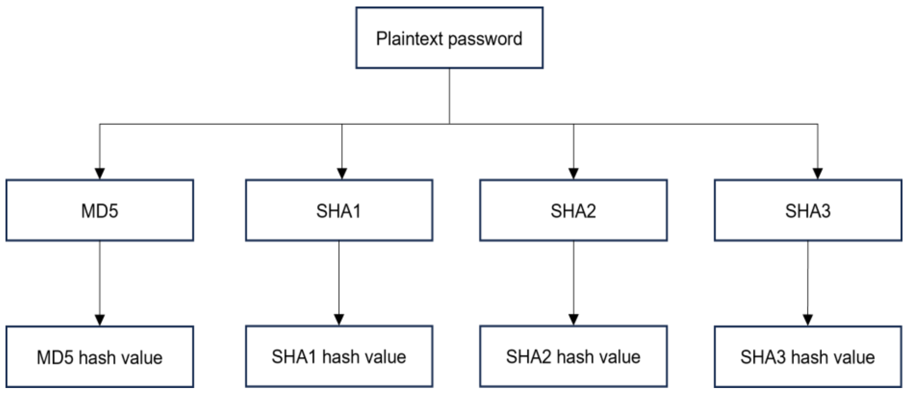

Figure 1. Process flow of create hash values

4. RESULTS AND DISCUSSION

In this section, we describe the two-fold analysis of the proposed method. The algorithm's speed in comparison to alternative password storage techniques is the first factor. Attack tolerance is the second. Before moving into the specifics of the studies and findings, it must be noted that we created our test program in personal home page (PHP hypertext processor) since MD5, SHA1, SHA2, and SHA3, the two most fundamental one-way hash algorithms, were readily available. In order to run the PHP script on the web server, Apache was utilized as the service. Hash values were stored in a MySQL database, which was controlled using phpMyAdmin.

4.1. Speed analysis

The experiment included different password storage techniques together with MD5, SHA1, SHA2, and SHA3. There was to take into account how long it took to calculate the resulting hash values of inputs passwords of various sizes. The 100 passwords from Cruz [33] a website that has ten thousand of the most popular passwords. A thousand times were used to hash each password. It was determined how long it takes to finish one hash on average. Table 4 displays the results.

Table 4. The time required to hash inputs of various sizes

Number of characters (bytes)	Computations time (ms) of MD5	Computations time (ms) of SHA1	Computations time (ms) of SHA2	Computations time (ms) of SHA3
4	0.00586	0.03198	0.031982	0.03711
8	0.00781	0.032715	0.033936	0.04102
16	0.00903	0.03906	0.04590	0.04883
32	0.02002	0.04004	0.05200	0.06104
64	0.02173	0.04834	0.05615	0.08398
128	0.02319	0.07300	0.08813	0.10425
256	0.03198	0.11011	0.11108	0.1250
512	0.04102	0.14307	0.16284	0.2290
1024	0.04614	0.15601	0.25903	0.27548
2048	0.06982	0.17407	0.42285	1.13208

It may be seen that while calculating the hash value of inputs of different sizes using MD5, SHA1, SHA2, and SHA3 algorithms. Table 4 shows the time it takes to process the hash function of each algorithm according to the size of the data. It was found that as the size of the data increases, the time required to process the data increases accordingly. As for the data processing speed performance measurement, the MD5 algorithm has the highest data processing speed, followed by the SHA1 algorithm, which has a similar speed to the algorithm. SHA2 and SHA3. Finally, the SHA3 algorithm takes the most time to process.

4.2. Security analysis

In this section, to emphasize once again, the main objective of this research is to compare algorithms for securely storing passwords using MD5, SHA1, SHA2, and SHA3. Therefore, it is necessary to test the safety of the proposed method. For assessing attack resistance, it is assumed that the attacker has access to a password database that contains users, salt values, and password hash values. Finding a plaintext password from the provided data is the attacker's responsibility. We employ the well-known Hashcat [34] program, which is used for password breach and recovery, in this study. Moreover, the list of passwords we analyzed Weak and strong passwords are the two types that they come under. The worst passwords for 2023 are on the listed of weak passwords, which was put up by Cruz [33]. The strong password list is designed to include special characters and/or numerals [35]. Table 5 displays both passwords.

Table 5. The character sets for passwords

Weak passwords	Strong passwords
123456	P@ssw0rd
password	p@ssw0rd1234
123456789	charles&jun!0r@
12345678	Qw3rty1234
1234567	Willi@md@ll@s
password1	J3llyFish
12345	p@ssw0rd1@dmin
1234567890	S3cur3Dev!ce
1234	Il0vey0u7777
qwerty123	A11B1ack$!
qwertyuiop	MaryHad_A_Sm@ll
1q2w3e4r	wroaps9ds
1qaz2wsx	Doct0rH0use.
superman	@damS@ndler
iloveyou	DisneyL@nd3
qwerty1	ILov3MyPi@no
qwerty	Jul1eLovesK3v1n
123456a	I34tcarr0ts
letmein	cookie%peanut@
football	m0nkEyil0vey0u

The security level of the password approach was compared and evaluated using the following techniques. The first technique used the MD5, SHA1, SHA2, and SHA3 algorithms to calculate a hash value from a plaintext password directly. Multiple iterations were the second technique. This method was used more than once to hash a password in plaintext. Stated otherwise, the hash function's result would be repeatedly generated. The passwords in this study were hashed twice before the results were entered into the database. Simulating assault scenarios was the following stage. The password database, which contained usernames and hash values, was thought to be accessible to an attacker. These were the only pieces of knowledge the attacker had. The purpose of the attack was to convert the relevant plaintext passwords from the hash values. This study employed the Hashcat program [34] as an attack tool. In order to conduct the experiment, Hashcat version 6.2.5 was used to test attacks on both the passwords listed in Table 5, such as weak and strong passwords. Additionally, we categorized our experimentation into two basic assault scenarios, which are best explained by the following.

4.2.1. Scenario 1

Scenario 1 included using the two techniques to get into stored weak passwords. The one-way hash functions were MD5, SHA1, SHA2, and SHA3. The outcomes of the first attack scenario are displayed in Table 6. It can be seen that when used to attack the hash value of weak password. Table 6 shows that all algorithms can be able to attack 19 out of 20 passwords, according to 95% of all passwords. Therefore, it can be concluded that storing passwords using weak passwords that have been processed using MD5, SHA1, SHA2, and SHA3 hash algorithms is not resistant to attacks.

Table 6. The results of scenario 1

Password hash algorithm	Attack password success (%)
MD5	95
SHA1	95
SHA2	95
SHA3	95

4.2.2. Scenario 2

Scenario 2 included using the two techniques to get into stored strong passwords. The one-way hash functions were MD5, SHA1, SHA2, and SHA3. The outcomes of the second attack scenario are shown in Table 7. It can be seen that when used to attack the hash value of strong password. Table 7 shows that MD5 and SHA1 algorithm can be able to attack 7 out of 20 passwords, according to 35% of all passwords. The SHA2 algorithm can be able to attack 6 out of 20 passwords, according to 30% of all passwords. The SHA3 algorithm can be able to attack 4 out of 20 passwords, according to 20% of all passwords.

Table 7. The results of scenario 2

Password hash algorithm	Attack password success (%)
MD5	35
SHA1	35
SHA2	30
SHA3	20

5. CONCLUSION

One of the most used forms of authentication is a password. Although using a password for authentication really works. But there are some things to consider. This paper's main topic is to compare hash functions and how to secure storing passwords. The researcher initially conducted a study and discovered that current password storage techniques were not as safe as they should be. The approaches of password storing with no salt, numerous iterations, fixed and dynamic salt were among those currently in use for password storage. They were all open to an assault.

The main aim of this article is to compare ways to store passwords more securely. Start by comparing the strength of strong and weak passwords. This is the article's first contribution. After conducting research, we discovered that the rainbow table needed to withstand the onslaught. There are two sections to the analysis of the suggested approach. The first is the outcome of the examination of speed. It was found that the MD5 algorithm's execution time was the fastest, followed by SHA1, which had a similar speed to the MD5 algorithm, followed by the SHA2 algorithm, and finally, the MD5 algorithm. SHA3 algorithm which takes the most time. This is because the MD5 algorithm is processed in 64 rounds to get a 128-bit hash function value. The SHA1 algorithm is processed. 80 rounds to get a 160-bit hash function. The SHA2 algorithm has 64 rounds and the SHA3 algorithm has 128 rounds.

The second analysis is the security analysis. Both strong and weak passwords are used for this, along with a one-way hash mechanism. Then, the Hashcat program was used to attack them. The results showed that Hashcat had the lowest success rate when it came to breaking passwords stored in SHA3 algorithm. Thus, the SHA3 algorithm can help with more secure password storage when compared to other hashing algorithms. However, Although the SHA3 algorithm is more secure at storing passwords than other algorithms tested, if you want to use it for storing passwords in a database system, can may need to add salt technique or the salt and slow hash algorithm in the feature work, which will make passwords more secure.

REFERENCES

[1] P. A. Wang, "Online phishing in the eyes of online shoppers," *IAENG International Journal of Computer Science*, vol. 38, no. 4, pp. 378–383, 2011.

[2] M. F. Hashmi, A. R. Hambarde, and A. G. Keskar, "Robust image authentication based on HMM and SVM classifiers," *Engineering Letters*, vol. 22, no. 4, pp. 183–193, 2014.

[3] H. Kumar *et al.*, "Rainbow table to crack password using MD5 hashing algorithm," *2013 IEEE Conference on Information and Communication Technologies, ICT 2013*, pp. 433–439, 2013, doi: 10.1109/CICT.2013.6558135.

[4] A. A. P. Ratna, P. D. Purnamasari, A. Shaugi, and M. Salman, "Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system," *2013 International Conference on Quality in Research, QiR 2013 - In Conjunction with ICCS 2013: The 2nd International Conference on Civic Space*, pp. 99–104, 2013, doi: 10.1109/QiR.2013.6632545.

[5] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, "Machine learning for detecting brute force attacks at the network level," *Proceedings - IEEE 14th International Conference on Bioinformatics and Bioengineering, BIBE 2014*, pp. 379–385, 2014, doi: 10.1109/BIBE.2014.73.

[6] R. L. Rivest, "RFC 1321: The MD5 message-digest algorithm," *Internet Activities Board*, pp. 1–21, 1992.

[7] H. Handschuh, "SHA family (secure hash algorithm)," *Encyclopedia of Cryptography and Security*, pp. 565–567, 2006, doi: 10.1007/0-387-23483-7_388.

[8] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2729, pp. 617–630, 2003, doi: 10.1007/978-3-540-45146-4_36.

[9] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 364–372, 2005, doi: 10.1145/1102120.1102168.

[10] L. Bosnjak, J. Sres, and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pp. 1161–1166, 2018, doi: 10.23919/MIPRO.2018.8400211.

[11] S. U. Rehman, S. Manickam, and N. F. Firdous, "Impact of DoS/DDoS attacks in IoT environment: a study," *AIP Conference Proceedings*, vol. 2760, no. 1, 2023, doi: 10.1063/5.0150000.

[12] K. Theoharoulis, I. Papaefstathiou, and C. Manifavas, “Implementing rainbow tables in high-end FPGAs for super-fast password cracking,” *Proceedings - 2010 International Conference on Field Programmable Logic and Applications, FPL 2010*, pp. 145–150, 2010, doi: 10.1109/FPL.2010.120.

[13] Cybernews Team, “Scraped data of 500 million LinkedIn users being sold online, 2 million records leaked as proof,” *Cybernews*, 2023, Accessed: Jul. 08, 2023, [Online]. Available: https://cybernews.com/news/stolen-data-of-500-million-linkedin-users-being-sold-online-2-million-leaked-as-proof-2/

[14] P. Li, Y. Sui, and H. Yang, “The parallel computation in one-way hash function designing,” *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering, CMCE 2010*, vol. 1, pp. 189–192, 2010, doi: 10.1109/CMCE.2010.5610467.

[15] W. Xiaoyun and Y. Hongbo, “How to break MD5 and other hash functions,” *Lecture Notes in Computer Science*, vol. 3494, pp. 19–35, 2005, doi: 10.1007/11426639_2.

[16] T. Xie, F. Liu, and D. Feng, “Fast collision attack on MD5,” *IACR ePrint archive Report*, vol. 104, p. 17, 2006.

[17] V. Chiriaco, A. Franzen, R. Thayil, and X. Zhang, “Finding partial hash collisions by brute force parallel programming,” *2017 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2017*, pp. 1–6, 2017, doi: 10.1109/LISAT.2017.8001964.

[18] K. Wu, Y. Li, L. Chen, and Z. Wang, “Research of integrity and authentication in OPC UA communication using Whirlpool hash function,” *Applied Sciences (Switzerland)*, vol. 5, no. 3, pp. 446–458, 2015, doi: 10.3390/app5030446.

[19] X. Wang, Y. L. Yin, and H. Yu, “Finding collisions in the full SHA-1,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3621 LNCS, pp. 17–36, 2006, doi: 10.1007/11535218_2.

[20] M. Stevens, P. Karpman, and T. Peyrin, “Freestart collision for full SHA-1,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9665, pp. 459–483, 2016, doi: 10.1007/978-3-662-49890-3_18.

[21] M. Sumagita and I. Riadi, “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.

[22] H. Tiwari, “Merkle-Damgård construction method and alternatives: a review,” *Journal of Information and Organizational Sciences*, vol. 41, no. 2, pp. 283–304, 2017, doi: 10.31341/jios.41.2.9.

[23] J.-P. Aumasson, S*erious cryptography a practical introduction to modern encryption*, No Starch Press, p. 305, 2017.

[24] J. Guo, G. Liao, G. Liu, M. Liu, K. Qiao, and L. Song, “Practical collision attacks against round-reduced SHA-3,” *Journal of Cryptology*, vol. 33, no. 1, pp. 228–270, 2020, doi: 10.1007/s00145-019-09313-3.

[25] R. Morris and K. Thompson, “Password security: a case history,” *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979, doi: 10.1145/359168.359172.

[26] D. V. Klein, “Foiling the cracker: a survey of, and improvements to, password security,” *Programming and computer software*, vol. 17, no. 3, pp. 158–166, 1992.

[27] L. Zhang, C. Tan, and F. Yu, “An improved rainbow table attack for long passwords,” *Procedia Computer Science*, vol. 107, pp. 47–52, 2017, doi: 10.1016/j.procs.2017.03.054.

[28] D. L. Jobusch and A. E. Oldehoeft, “A survey of password mechanisms: weaknesses and potential improvements. Part 2,” *Computers and Security*, vol. 8, no. 8, pp. 675–689, 1989, doi: 10.1016/0167-4048(89)90006-0.

[29] W. Ma, J. Campbell, D. Tran, and D. Kleeman, “Password entropy and password quality,” *Proceedings - 2010 4th International Conference on Network and System Security, NSS 2010*, pp. 583–587, 2010, doi: 10.1109/NSS.2010.18.

[30] A. Ferrara, “The rainbow table is dead,” blog.ircmaxell.com. Accessed: Jul. 08, 2023. [Online]. Available: https://blog.ircmaxell.com/2011/08/rainbow-table-is-dead.html

[31] Sutriman and B. Sugiantoro, “Analysis of password and salt combination scheme to improve hash algorithm security,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 11, pp. 420–425, 2019, doi: 10.14569/IJACSA.2019.0101158.

[32] S. Chethana, S. S. Charan, V. Srihitha, D. Radha, and C. R. Kavitha, “Comparative analysis of password storage security using double secure hash algorithm,” *2022 IEEE North Karnataka Subsection Flagship International Conference, NKCon 2022*, pp. 1–5, 2022, doi: 10.1109/NKCon56289.2022.10127057.

[33] A. Cruz, “100 worst passwords of 2023,” www.purevpn.com. Accessed: Jul. 08, 2023. [Online]. Available: https://www.purevpn.com/blog/worst-password-list/

[34] M. Shivanandhan, “How to crack hashes with Hashcat - a practical pentesting guide,” 2022, www.freecodecamp.org. Accessed: Jul. 08, 2023. [Online]. Available: https://www.freecodecamp.org/news/hacking-with-hashcat-a-practical-guide/

[35] M. M. Taha, T. A. Alhaj, A. E. Moktar, A. H. Salim, and S. M. Abdullah, “On password strength measurements: password entropy and password quality,” *Proceedings - 2013 International Conference on Computer, Electrical and Electronics Engineering: “Research Makes a Difference”, ICCEEE 2013*, pp. 497–501, 2013, doi: 10.1109/ICCEEE.2013.6633989.

BIOGRAPHIES OF AUTHORS

Parinya Natho is currently a lecturer of the Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi (RMUTSB). He received his Ph.D. in information technology at the Faculty of Information Technology, King Mongkut’s University of Technology North Bangkok (KMUTNB), Thailand. He received an M.Sc. in computer science and information systems from the National Institute of Development Administration, Thailand. His research interests include cyber security, information security, IoT, machine learning, image processing, data analytics and artificial intelligence (AI). He can be contacted at email: parinya.n@rmutsb.ac.th.

Suwit Somsuphaprungyos is currently a lecturer of the Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi (RMUTSB). He received his Ph.D. in management of information technology at the School of Informatics, Walailak University (WU), Thailand. He received an M.Sc. in internet and information technology from the Naresuan University (NU), Thailand. His research interests include internet technology, internet of things, smart technology, data analytics and ontology. He can be contacted at email: suwit.s@rmutsb.ac.th.

Salinun Boonmee is currently an assistant professor and a lecturer of the Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi (RMUTSB). She received her Ph.D. in computer education at King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand. Her research interests include IoT, machine learning, data analytics and artificial intelligence. She can be contacted at email: salinun.b@rmutsb.ac.th.

Sangtong Boonying is currently an assistant professor and a lecturer of the Department of Information System and Business Computer, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Suvarnabhumi (RMUTSB). He received the Ph.D. degree in education technology from Burapha University, Thailand. His research interests include IoT, machine learning, image processing, data analytics and artificial intelligence. He can be contacted at email: sangtong.b@rmutsb.ac.th.