ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 5, May 2023

One Way Hashing for Password Authentication

[1] Dr. Logesh Babu, [2] Gnaneswar, [3] Jai Sai Nath Reddy, [4] Indrasena Reddy, [5] Gokul Vardhan

[1] Assitant Professor, Department of Computer Science, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh, India

[2] [3] [4] [5] Student, Department of Computer Science, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh, India

Abstract— ***Password security has become a major worry in the current digital era. The fact that passwords are frequently the first line of defense against unauthorized access to sensitive data makes them a prime target for hackers looking to exploit security system flaws. An overview of various password attack methods and secure hashing algorithms that can be applied to counteract these assaults is given in this paper. The various password assaults, such as rainbow table attacks, dictionary attacks, and brute-force attacks. The discussion of hashing algorithms, which can be used to secure passwords by password authentication. Some of the most used algorithms, including MD5, SHA-1, SHA-2, and SHA-3, as well as the characteristics of secure hash functions. It also briefly explains more complex algorithms like bcrypt, scrypt, and argon2.***

Keywords: authentication, argon2, brute force, cryptography, hash function, SHA.

I. INTRODUCTION

Numerous online services have developed as a result of the growth of the Internet, and password authentication is the most used authentication method as a security measure. Passwords are continued to be cracked despite significant advancements in password security research due to users' reckless actions. For instance, many users frequently choose weak passwords, reuse the same passwords across many platforms, and typically establish their passwords using common words to make them easier to remember. To secure the password authentication process, the cryptographic hashing technique will give high-security hashed passwords, which are encrypted through the cryptographic method called Hashing. Hashing is the only one way process in which passwords are only encrypted and cannot be decrypted such passwords are called Hashed Passwords.

It is impossible to directly retrieve plain passwords from passwords that have been hashed using a cryptographic hash algorithm. Data of any size can be swiftly converted to a fixed-size sequence of bits using the cryptographic hash function. Only hashed passwords are kept in the database for the authentication system that uses this password hashing technique. Systems with hashed passwords are more resistant to attack if the database is hacked. We used MD5, SHA-1, SHA-2, SHA-3, and argon2 as our hashing algorithms.

II. LITERATURE SURVEY

The authors in [1] used the SHA-2 hash function, for the best security SHA-3 can be used for password authentication and we show how it can be secure between those functions. MD5 used by authors in [2] which is the older hash version where more prone to attacks and insecure for password authentication which is prone to get hacked. SHA-2, SHA-3 having 256-bit> hashing provides eminent security than SHA-1 used by authors [3] which is a 160-bit hash value. The CAPTCHA uses an MD5 hash function for encryption and it can be attacked by hackers and it is not a better process of authentication [4]. [5] Used AES-256 which encrypts and decrypts the text during the decryption process, whereas SHA performs one way encryption which decryption can make hackers easy to attack systems. The authors in [6] used the SHA-3 Hash function and AES-256 for fingerprint authentication for database access control. As AES follows both encryption and decryption. SHA-3 is secured but AES makes the data decryption.

III. METHODOLOGY

3.1 Hashing

Hashing is a fundamental concept in computer science and cybersecurity that involves converting data of arbitrary size into a fixed-size sequence of bits using a mathematical function known as a hash function. Hashing is commonly used for a wide range of applications, including data storage, password authentication, data compression, and data fingerprinting.

In simple terms, a hash function takes an input message or data and returns a fixed-length output string, called a hash value or message digest. Hashing is a one-way process in which it is easy to compute the hash value from the input data, but it is computationally infeasible to recover the original data from the hash value.

In order to store passwords securely, hashing is frequently used in password authentication systems. The database stores the password's hash value rather than the plaintext password. The system computes the hash value of the password entered by the user and compares it to the previously stored hash value. The user is given access if the two hash values match.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 5, May 2023

With this method, even if the database is compromised, the attacker won't be able to figure out the plaintext password from the hash value that was previously stored.

Some of the commonly used hash functions include MD5, SHA-1, SHA-2, and SHA-3. Each of these hash functions has unique features and properties that make them suitable for specific applications. For instance, SHA-1 and SHA-2 are widely used in password authentication systems, while SHA-3 is commonly used for data fingerprinting.

Overall, hashing is a vital concept in cybersecurity and computer science that plays a crucial role in securing data and systems. By converting data into a fixed-size sequence of bits using a hash function, hashing enables secure data storage, password authentication, data compression, and data fingerprinting.

3.2 Hashing Algorithms

Some Commonly Hashing algorithms that are used for password authentication are

- MD5
- SHA-1
- SHA-2
- SHA-3

3.2.1 MD5

This algorithm produces a message digest of 128 bits, or 16 bytes, from data of any length. The input message is divided into 512-bit blocks for this algorithm. In order to make the message length 64 bits smaller than a multiple of 512, the message is padded with a 1 followed by zeros. 64 bits are used to fill the remaining bits, which correspond to the original message's length. Although this hashing algorithm is widely used, it is prone to collisions. The collision attack, however, is too slow in practice to be effective. This is broken for collisions but not for first- or second-previous-images.

3.2.2 SHA-1

SHA-1 (Secure Hash Algorithm 1) is a widely used cryptographic hash function that takes an input message of any length and produces a fixed-size output, called a message digest, of 160 bits. It was developed by the National Security Agency (NSA) and is commonly used for digital signatures and data integrity checks. However, due to security vulnerabilities, it is no longer considered secure.

3.2.3 SHA-2

The National Security Agency (NSA) in the United States created the widely used cryptographic hash function family known as SHA-2 (Secure Hash Algorithm 2). A variety of hash functions, including SHA-224, SHA-256, SHA-384, and SHA-512, are included in SHA-2. SHA-2 is used often in applications like SSL/TLS certificates, password authentication, and digital signatures because it is more secure than its forerunner, SHA-1. The SHA-2 method creates a fixed-length message digest using a set of logical functions and modular arithmetic operations that are extremely resistant to collisions and pre-image attacks.

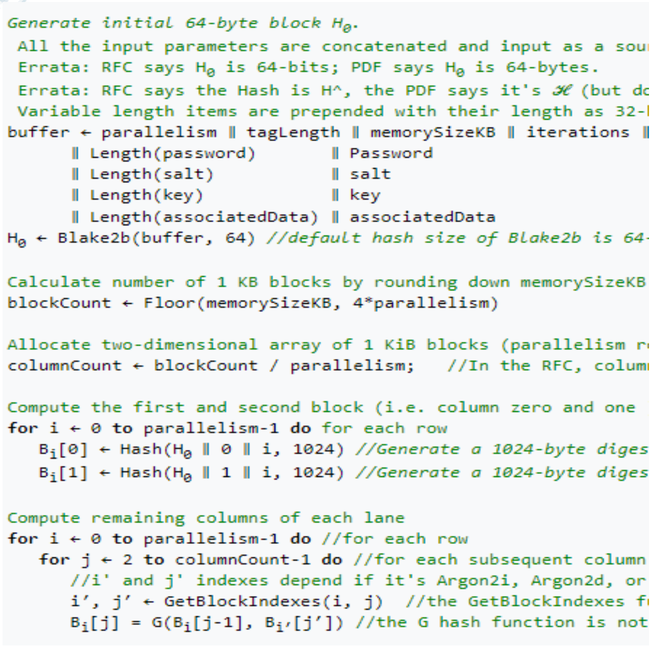
3.2.4 SHA-3

The National Institute of Standards and Technology (NIST), in response to a public competition, created the cryptographic hash function family known as SHA-3 (Secure Hash Algorithm 3). In order to counteract any potential flaws in the currently used SHA-1 and SHA-2 algorithms, SHA-3 was created. SHA3-224, SHA3-256, SHA3-384, and SHA3-512 are only a few of the hash algorithms in the SHA-3 family that have a range of output sizes. The SHA-3 algorithm employs a sponge construction, which involves absorbing the input message into a state and then compressing the state to produce the message digest. For a variety of uses, including password authentication and digital signatures, SHA-3 is well-suited since it is highly resistant to collisions and pre-image attacks.

3.2.5 Argon2

Argon2 is a password hashing algorithm that was created to withstand numerous attacks, such as side-channel attacks and brute-force attacks. A group of cryptography professionals created it as a part of the 2013 Password Hashing Competition, which was run by the Password Hashing Consortium. Since Argon2 is intended to be memory-hard, computing the hash function necessitates a sizable amount of memory. This makes it challenging for attackers to conduct simultaneous attacks on numerous credentials. Argon2 is suited for usage in contemporary computer systems due to its resistance against GPU-based assaults.

Algorithm

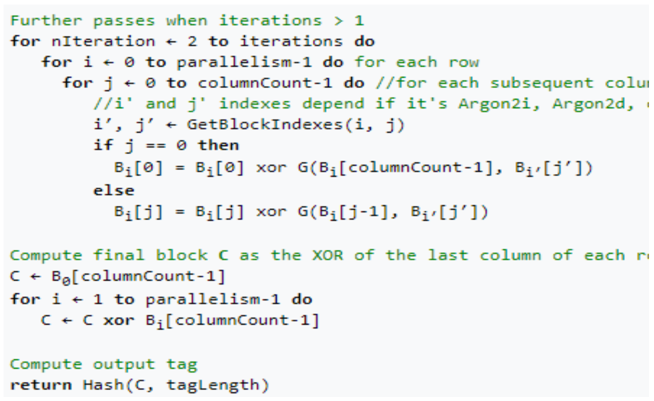


```
Generate initial 64-byte block H_0.
 All the input parameters are concatenated and input as a sou
 Errata: RFC says H_0 is 64-bits; PDF says H_0 is 64-bytes.
 Errata: RFC says the Hash is H^, the PDF says it's ℋ (but do
 Variable length items are prepended with their length as 32-
buffer ← parallelism ‖ tagLength ‖ memorySizeKB ‖ iterations ‖
      ‖ Length(password)        ‖ Password
      ‖ Length(salt)            ‖ salt
      ‖ Length(key)             ‖ key
      ‖ Length(associatedData)  ‖ associatedData
H_0 ← Blake2b(buffer, 64) //default hash size of Blake2b is 64

Calculate number of 1 KB blocks by rounding down memorySizeKB
blockCount ← Floor(memorySizeKB, 4*parallelism)

Allocate two-dimensional array of 1 KiB blocks (parallelism r
columnCount ← blockCount / parallelism;   //In the RFC, colum

Compute the first and second block (i.e. column zero and one
for i ← 0 to parallelism-1 do for each row
   B_i[0] ← Hash(H_0 ‖ 0 ‖ i, 1024) //Generate a 1024-byte diges
   B_i[1] ← Hash(H_0 ‖ 1 ‖ i, 1024) //Generate a 1024-byte diges

Compute remaining columns of each lane
for i ← 0 to parallelism-1 do //for each row
   for j ← 2 to columnCount-1 do //for each subsequent column
      //i' and j' indexes depend if it's Argon2i, Argon2d, or
      i′, j′ ← GetBlockIndexes(i, j)  //the GetBlockIndexes f
      B_i[j] = G(B_i[j-1], B_i′[j′]) //the G hash function is not
```

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 5, May 2023



```
Further passes when iterations > 1
for nIteration ← 2 to iterations do
   for i ← 0 to parallelism-1 do for each row
     for j ← 0 to columnCount-1 do //for each subsequent colu
        //i' and j' indexes depend if it's Argon2i, Argon2d,
        i′, j′ ← GetBlockIndexes(i, j)
        if j == 0 then
          Bi[0] = Bi[0] xor G(Bi[columnCount-1], Bi′[j′])
        else
          Bi[j] = Bi[j] xor G(Bi[j-1], Bi′[j′])

Compute final block C as the XOR of the last column of each r
C ← B0[columnCount-1]
for i ← 1 to parallelism-1 do
   C ← C xor Bi[columnCount-1]

Compute output tag
return Hash(C, tagLength)
```

Figure 1 pseudo code of Argon2

IV. MODELING AND ANALYSIS

While choosing hashing algorithm we will consider some security factors such as

- Bit Length
- Preimage
- Collision
- Time

Bit Length

Bit length, also known as the digest size or output size, is a measure of the size of the hash value produced by a hashing algorithm. It represents the number of bits in the resulting hash value, which determines the number of possible hash values that can be generated by the algorithm. For example, a hashing algorithm that produces a 128-bit hash value can generate 2^128 possible hash values. As the bit length increases, the number of possible hash values also increases exponentially, making it more difficult for attackers to find collisions (two different inputs that produce the same hash value) or to reverse-engineer the original input data from the hash value.

In general, longer bit lengths provide better security, as they make it more difficult for attackers to guess or brute-force the original input data or to create hash collisions. However, longer bit lengths can also increase the processing time and memory requirements for the hashing algorithm. Therefore, the choice of bit length depends on the level of security required by the application and the performance constraints of the system. In practice, commonly used bit lengths for hashing algorithms include 128 bits, 256 bits, 384 bits, and 512 bits.

Preimage

A preimage attack is a type of attack on a hashing algorithm where an attacker attempts to find a message or input that produces a specific hash output. In other words, the attacker wants to find a message that matches a given hash value. Preimage attacks can be categorized into two types:

First preimage attack: In this type of attack, the attacker tries to find any message that produces a specific hash output. This is similar to finding a password that matches a given hash value.

Second preimage attack: In this type of attack, the attacker tries to find a different message that produces the same hash output as a known message. This is similar to finding a different password that produces the same hash value as a known password.

Preimage attacks are a serious security concern for hashing algorithms because they allow an attacker to bypass the security of the algorithm and access sensitive data. A hashing algorithm that is vulnerable to preimage attacks is considered weak and should be replaced with a stronger algorithm that can resist these attacks. Cryptographic hash functions, such as SHA-2 and SHA-3, are designed to be resistant to preimage attacks, making them suitable for use in secure applications. However, it's important to note that even these algorithms are not completely immune to attacks, and researchers continue to study and improve upon these algorithms to ensure their security.

Collision

An attack of this kind against a hashing algorithm involves trying to identify two distinct messages that give the same hash output. In other words, the attacker is looking for a hash value collision between two distinct messages. Hashing algorithms should be wary of collision attacks because they might be used by an attacker to circumvent the algorithm's security and produce malicious data that could be mistaken for valid data. For example, an attacker could use a collision attack to create two different messages that have the same hash value, one of which is legitimate and the other is malicious. This could allow the attacker to substitute the malicious message for the legitimate one without detection. Cryptographic hash functions, such as SHA-2 and SHA-3, are designed to be resistant to collision attacks, making them suitable for use in secure applications. However, as computing power continues to increase, collision attacks become more feasible, and researchers continue to study and improve upon hashing algorithms to ensure their security.

Time

The time factor in hashing refers to the amount of time it takes to perform a hashing operation on a given input message. This is an important performance factor for hashing algorithms, especially in applications where large amounts of data need to be processed quickly. The time taken to hash a message depends on various factors such as the size of the input message, the complexity of the hash function, and the computing resources available for the hashing operation. Some hashing algorithms may be faster than others for certain input sizes or on certain hardware platforms. In applications where high performance is critical, such as password authentication systems, the time taken to hash a message can have a significant impact on the overall system performance. Slow hashing algorithms can result in slower

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 10, Issue 5, May 2023

response times for users, which can be a frustrating experience.

To address this issue, some hashing algorithms, such as Argon2, are designed to be memory-hard, which means they require a significant amount of memory to perform the hashing operation. This makes them more resistant to brute-force attacks but also means they are slower to compute, which can be an advantage in some applications. Overall, the time factor in hashing is an important consideration when selecting a hashing algorithm for a particular application. It's important to choose an algorithm that provides the necessary level of security while also meeting the performance requirements of the application.

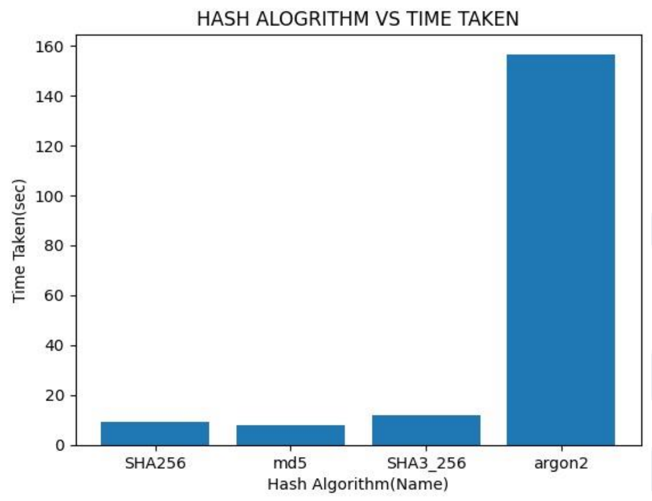


Figure 2 Comparison of Different Hashing Algorithms based on Time

Some commonly used for Password Attacks by used by hackers are

- Dictionary Attack
- Rainbow Table Attack
- Brute Force

Dictionary Attack

A dictionary attack involves the attacker attempting to use a word list, which may include commonly used terms, passwords, dates, names, etc. The attacker will use the hash algorithm for each phrase and check to verify if the produced hash matches the hash stored in the database. If this is the case, the attacker is aware that the word used to create the hash is also the password.

Rainbow Table Attack

Rainbow tables are a time-memory tradeoff approach. They are comparable to lookup tables, with the exception that to make lookup tables smaller, the speed of hash cracking is sacrificed. They are more efficient since more hashes' solutions can be stored in the same volume thanks to their reduced size. Any md5 hash of a password that is up to 8 characters long can be cracked using rainbow tables.

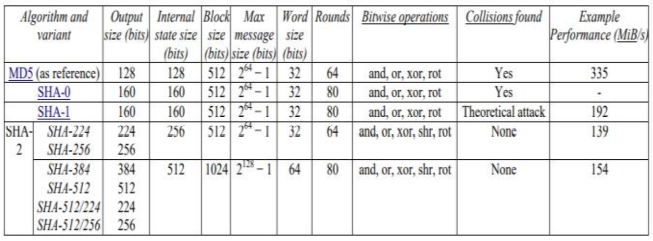
Brute Force Attack

Brute force is a type of attack in which an attacker tries every possible combination of characters to guess a password or encryption key. This attack method is based on trial and error, where the attacker uses automated software or tools to repeatedly attempt different passwords until the correct one is found. Brute force attacks can be effective if the password is weak or the encryption key is short, but they can also be time-consuming and resource-intensive. The best way to protect against brute force attacks is to use strong passwords or encryption key that are long, complex and difficult to guess.

V. RESULTS AND DISCUSSION

An overview of the various hashing methods used in computer security for password authentication is given in this paper. The paper gives a thorough analysis of the most widely used hashing algorithms and emphasizes the significance of hashing algorithms in password security. In the study, a number of hashing algorithms were examined, including MD5, SHA-1, SHA-2, SHA-3, and Argon2. Based on these algorithms' advantages and disadvantages, including security, performance, and memory usage, the authors compared them.

The investigation discovered that due to their vulnerability to collision attacks, the once-commonly used MD5 and SHA-1 are no longer advised. According to the study, SHA-2, SHA-3, and Argon2 are more secure and provide stronger defense against attempts to guess passwords. Sure, the paper briefly discusses Argon2 as a hashing algorithm for password authentication in the conclusion. The authors note that Argon2 is a relatively new algorithm and has gained popularity due to its resistance to side-channel attacks and its ability to adjust the amount of memory required for hashing. However, the authors also highlight that more research is needed to fully understand the security properties of Argon2 and its performance in different scenarios. They recommend that further studies should focus on evaluating the effectiveness of Argon2 in protecting against attacks such as collision attacks, preimage attacks, and dictionary attacks, and comparing its performance with other widely used hashing algorithms. Overall, while Argon2 shows promise in the field of password authentication, further research is needed to establish its suitability for different applications and environments.



Algorithm and variant		*Output size (bits)*	*Internal state size (bits)*	*Block size (bits)*	*Max message size (bits)*	*Word size (bits)*	*Rounds*	*Bitwise operations*	*Collisions found*	*Example Performance (MiB/s)*
MD5 (as reference)		128	128	512	$2^{64}-1$	32	64	and, or, xor, rot	Yes	335
SHA-0		160	160	512	$2^{64}-1$	32	80	and, or, xor, rot	Yes	-
SHA-1		160	160	512	$2^{64}-1$	32	80	and, or, xor, rot	Theoretical attack	192
SHA-2	*SHA-224*	224	256	512	$2^{64}-1$	32	64	and, or, xor, shr, rot	None	139
	SHA-256	256								
	SHA-384	384	512	1024	$2^{128}-1$	64	80	and, or, xor, shr, rot	None	154
	SHA-512	512								
	SHA-512/224	224								
	SHA-512/256	256								

Figure 3 Comparison of Different SHA Functions

VI. CONCLUSION

In all the different types of cryptographic hash functions like MD5, SHA-1, SHA-2, SHA-3, and Argon2. We used the Argon2 hash function for password authentication as it's a newer version and the most secure among other hash functions to protect passwords from attacks. The Argon2 takes more time than other algorithms but as a concern of security, we took it for password authentication.

REFERENCES

[1] W. Luo, Y. Hu, H. Jiang and J. Wang, "Authentication by Encrypted Negative Password," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 1, pp. 114-128, Jan. 2019.

[2] E. Sediyono, K. I. Santoso and Suhartono, "Secure login by using One-time Password authentication based on MD5 Hash encrypted SMS," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, India, 2013, pp. 1604-1608.

[3] H. Seta, T. Wati and I. C. Kusuma, "Implement Time Based One Time Password and Secure Hash Algorithm 1 for Security of Website Login Authentication," 2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, 2019, pp. 115-120.

[4] Alajmi, Masoud; Elashry, Ibrahim; El-Sayed, Hala S.; Faragallah, Osama S. (2020). A Password-based Authentication System based on The CAPTCHA AI Problem. IEEE Access, *(), 1–1*.

[5] AbouSteit, Mohamed. H.S.; Tammam, Ashraf Farouk; Wahdan, AbdelMoneim (2020). [IEEE 2020 Fourth World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4) - London, United Kingdom (2020.7.27-2020.7.28)] 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4) - A Novel Approach For Generating One-Time Password With Secure Distribution. , (), *461–466*.

[6] A. M. Tillah, D. Ogi, M. Febriyanto and D. A. Farhatin, "Access Control System based on Secret Sharing Scheme with Secure Web Database and SHA-3 Password Authentication," 2021 6th International Workshop on Big Data and Information Security (IWBIS), Depok, Indonesia,2021,pp. 145-152.