International Journal of Safety and Security Engineering
Vol. 10, No. 6, December, 2020, pp. 785-796
Journal homepage: http://iieta.org/journals/ijsse

H-Rotation: Secure Storage and Retrieval of Passphrases on the Authentication Process

Hamza Touil[1*], Nabil El Akkad[2], Khalid Satori[1]

[1] LISAC Faculty of Sciences Dhar-Mahraz (FSDM), Sidi Mohamed Ben Abdellah University, Fez 30050, Morocco
[2] Laboratory of Engineering Systems and Applications (LISA), National School of Applied Sciences (ENSA) Sidi Mohamed Ben Abdellah University, Fez 30000, Morocco

Corresponding Author Email: hamza.touil@usmba.ac.ma

https://doi.org/10.18280/ijsse.100609

Received: 27 September 2020
Accepted: 3 December 2020

Keywords:
passphrase, man in the middle, rainbow-table attack, hash function, SHA-3, rotation, authentication

ABSTRACT

Passwords/passphrases can be either system generated or user-selected. A combination of both approaches is also possible—encryption created by the system and assigned to the user by the information system meeting the policy requirements. Policy rules can be designed to increase security and usability factors, such as information storage and retrieval. This paper proposes an algorithm dedicated to the security of passphrases in an online authentication, so the passphrase entered will be stored in a remote database. Through an SHA-3 hash function, the system must hash the pass phase. Before storage, the system must apply random rotations on the already generated HASH while eliminating any traceability performed on the different transactions performed. To prevent the hacker from using them recurrently if he wants to attack our database. Then the system must recover the real HASH and then the passphrase based on the data provided by the user in the form of codes.

1. INTRODUCTION

Several authentication methods are available, especially on Web platforms, including password/password authentication, certificates, one-time passwords/passphrases, access keys, and tokens. It is essential to differentiate between identification that consists of a declaration of user identity. Depending on the situation, it can be a name, an e-mail address, a telephone number—furthermore, an authentication on which the user has to prove that he is the one who identified.

An authentication process [1-4] may involve one or a combination of two or more means to verify the credibility of the targeted system. If the authentication is done through a unique token, the applicant presents this token as a valid identifier to prove his identity to the verifier. For example, when a user wants to connect to a web platform protected by a password or passphrase, he must enter his login and then his password/passphrase. In this case, only the second party is considered as an initial access point, i.e., any mistake or level of password/password entry automatically results in a denial.

In multiple token authentications, the applicant presents authenticators generated by two or more tokens to prove their identity to the verifier. The combination of tokens is characterized by the combination of factors used by the tokens.

On the security side, some cryptographic mobility solutions allow full or partial cryptography [5, 6], including keys to be stored on an online server and downloaded locally by the applicant after successful authentication using a passphrase/password. Subsequently, the applicant can use the cryptographic token from the downloaded software to authenticate with a remote e-authentication verifier. This type of solution is considered secure as the requester's password to obtain the cryptographic token [7-9].

The absolute advantage of passphrases is that they are longer than traditional passwords. This change's main reason is to increase security, especially against man-in-the-middle and brute force attacks, which occur when an attack is carried out not by making informed guesses but by exhaustively trying all possible permutations in a password. In the rest of this document, we will see other work interrelated with our method and the types of authentication available today. We passed by taking a look at the hash functions and their historical development and ending by defining some types of attacks and our proposed solution in this context.

2. RELATED WORKS

A set of studies has been proposed to ensure the security of the authentications. They are beginning with the technique [10] capable of hiding user data and making it abstract. They can also support all forgery attacks, using a secure remote authentication scheme based on smart cards. Hariharan [11] use a new graphical approach in the form of a set of alternatives to the alphanumeric password, a highly repetitive procedure. At a time when any request is provided with an easy-to-understand confirmation. By giving detailed calculations that depend on the determination of the "Identifier" and "images" as a password. Based on the letters to rearrange the positions of the characters in the username. In the same context. Kumar et al. [12] presents a new advanced security architecture for user identification, which includes two authentication factors, encryption and decryption of data uploaded to the cloud using an AES system, verification and locking of users by the administrator, recovery of users' IP data, distributed database storage, i.e., data is stored in tiers, meaning that user login data is stored in one database and encrypted/decrypted data (downloaded files, key) is stored in

different databases.

On the other hand, using the hash technique [13], uses a message authentication protocol using a homomorphic hash function without full key security in a WANET environment with limited resources for each node. Bebe and Akila [14] invent a mechanism under the name "OSA-SHSDS" to store user data in the cloud with a higher confidentiality rate and less complicated space. To store the data in the server, the cloud user's data must be recorded. After registration, the cloud server generates the ID and password for each registered cloud user. Each time the user wants to store the data, he has to log in to the cloud server with his ID and password. The approach [15] is to solve keystrokes that can falsify data processing by offering a new validation of the passphrase. Arya et al. [16] implemented a new authentication system based on a passphrase, the system can insert a user-generated mnemonic image displayed during login to minimize typing errors. However, only a few research papers have been presented to strengthen password hashes since [17]. El focused on improving security outside the processing, storage, and transmission cycle compared to existing classical methods. This method can be used in several applications such as remote login, encrypted and authenticated communication, and payment. On the other hand, Shimizu et al. [18] offers a method suitable for communication in secure environments such as the Internet. In particular, it can be adapted to Internet devices or, for example, Java applets that have limited performance. The PERM method does not require a password setting and allows high-speed authentication processing with a small program.

3. TYPES OF AUTHENTICATION

In modern systems, there are more complex authentication and authorization schemes that are simple and understandable. Starting with:

Authentication through passwords/passphrases: this method is based on the fact that the user must provide a username and a password/passphrase for successful identification and authentication in the system. The username / password pair is defined by the user when registering in the system. There are several standard protocols for password/password authentication, which can be quoted exhaustively:

- **HTTP:** uses standardized processes that are well supported by all browsers and web servers. Several authentication schemes differ in the level of security:
  - Primary is the most straightforward scheme, in which the user's username and password/password are transmitted in the authorization header in unencrypted form. However, the use of HTTPS (HTTP over SSL) is relatively secure.
  - The digest is a challenge-response scheme in which the server sends a unique nuance value, and the browser sends the user's MD5 password hash, calculated using the specified nuance.
  - Other decided schemes can be found on the Windows system such as NTLM, Negotiate.

- **Forms:** There is no specific standard for this protocol. Therefore, all its implementations are system-specific, and more specifically, the authentication modules of the development frameworks. It works on the following principle: an HTML form is included in the web application. The user must enter his username/password and send them to the server via HTTP POST for authentication. If successful, the web application creates a session token, usually placed in the browser's cookies. On subsequent Web requests, the session token is automatically transferred to the server and allows the application to obtain information about the current user to authorize the request.

Certificate Authentication: A certificate is a set of attributes identifying the owner, signed by the Certificate Authority (CA). CA acts as an intermediary that guarantees the authenticity of certificates (similar to the FMS issuing passports). The certificate is also cryptographically associated with the private key, stored by the certificate holder, and allows you to confirm ownership of the certificate, as shown below in Figure 1.

Figure 1. Using a certificate for authentication

One-Time Password Authentication: One-Time Password Authentication is generally applied in addition to Password Authentication to implement Two-Factor Authentication (2FA). In this concept, the user must provide two types of data to enter the system: something They know (e.g., a password) and something they own (e.g., a device for generating one-time passwords). The presence of two factors can significantly increase the level of security required for certain types of web applications.

Access Key Authentication: This method is most often used to authenticate devices, services, or other applications when accessing web services. Here, access keys (access key, API key) are used as long-secret single lines containing an arbitrary set of characters, permanently replacing the username/password combination. In most cases, the server generates access keys at the user's request, storing these keys in client applications. When creating a key, it is also possible to limit the validity and level of access that the client application will receive when authenticating with that key, as shown below in Figure 2.

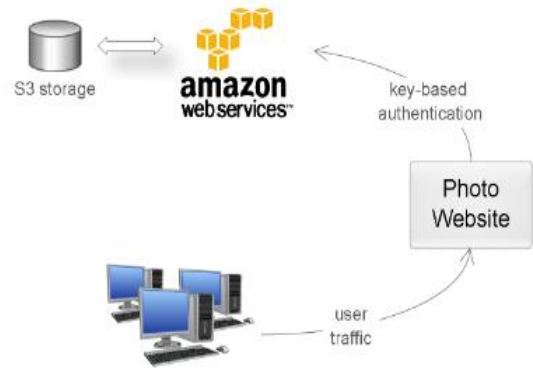


Figure 2. An example of the application of key authentication

Token authentication: This authentication method is most often used in the creation of distributed SSO (Single Sign-On) systems, where one application (service provider or trusted party) delegates the user authentication function to another application (identity provider or authentication service) in Figure 3. A typical example of this method is logging on to the application via an account on social networks. Here, social networks are authentication services, and the application trusts the user authentication function in social networks [19-22].

Figure 3. An example of authenticating an "active" client using a token transmitted via a support scheme

4. THE HASH FUNCTIONS

A cryptographic hash function maps a string of bits of random length to a string of bits of fixed and short length, typically between 128 and 512 bits. Therefore, it can be visualized as the opposite of a pseudo-random generator, which develops a short to an arbitrarily long string. Like a pseudo-random cryptographic generator, a cryptographic hash function is expected to provide various security properties.

For cryptographic hash functions, it is also essential that the value of the function changes significantly at the slightest change of argument. This is called an avalanche effect.

The following requirements are imposed on key hash functions:

- impossible to manufacture, means the high complexity of selecting a message with the correct HASH value.
- impossible to modify, meaning the high complexity of selecting a given message with a known HASH value from another message with the correct HASH value.

Keyless functions have the following requirements:

- unidirectionality, the high complexity of finding a message with a given HASH value. It should be noted that currently, there are no hash functions used with proven unidirectionality.
- collision resistance is understood as the difficulty of finding a pair of messages with the same convolution values. Usually, it is discovering a collision construction method by cryptanalysts that serves as a first signal of the algorithm's obsolescence and the need for its early replacement.
- resistance to the search for the second prototype, one understands the difficulty of finding a second message with the same convolution value for a given message with a known convolution value

Several chopping solutions available SHA-0, SHA-1, SHA-2, SHA-3, MD2,3,4,5.

The MD family, which was widely used to check file integrity (checksums) and store hashed passwords in web application databases. On the other hand, its low output length and ease of use made the MD5 very easy to crack and sensitive to a potential attack.

On the SHA family, we can identify SHA-0 as the weakest algorithm on the SHA family than what had been announced and showed in 2005 when SHA-1 [23, 24] also had flaws in the design of SHA-2 is not very different from the design of SHA-1. No one will be surprised to learn that SHA-2 is also weak, but on the other hand, no attacks are recorded on this algorithm so far.

It can be said that the fourth generation of the SHA protocol, also known as SHA-3, is not intended to replace SHA-2. Due to the successful attacks on MD5, SHA-0 and SHA-1, [25, 26] the responsible entity saw the need for an alternative, a unique cryptographic hash, which became SHA-3.

5. POSSIBILITIES OF ATTACK

MITM (Man In The Middle): A classic attack method consists in creating its access point [27], and forging the gateway to access a requesting server. When an attacker secretly relays and, if necessary, modifies the connection between two objects that believe they interact directly with each other, it is a method to compromise a communication channel by connecting to the channel between contractors, followed by an intervention the transmission protocol (Figure 4).

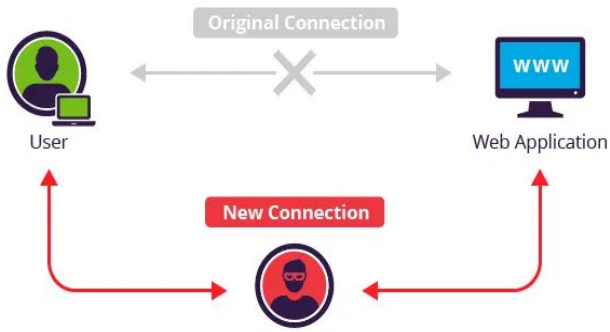


Figure 4. Man in the middle attack

Rainbow-table attack: based on the creation strings of possible passwords. Each string starts with a random possible password, and then it is exposed to a hash function and a collapse function. This function converts the result of the hash function into a possible password. The string's common passwords are removed, and only the first and last elements of the string are written to the array. Creating tables requires time and memory (up to hundreds of gigabytes), but they allow you to quickly restore the original password (compared to conventional methods).

This hash value is subject to the reduction function to recover the password and is searched in the table. If no match is found, the hash function and the reduction function are applied again. This operation continues until a match is found. After a match is found, the string containing it is restored to find the deleted value, the desired password. The result is a table that can, with a high probability, restore the password in a short time.

6. H-ROTATION

The H-ROTATION authentication method consists of two different phases (Figure 6). In the first phase, where the registration page is invoked, the user fills out and sends save the requested form. Then the server extracts the submitted information, username, passphrase, and other additional information. The SHA3 hash algorithm then processes the chosen passphrase. Then, a HASH rotation will be performed correctly to generate a new hash with the same size but a different shape. When storing the hash in the database, the user receives three integers to store them. The meaning of each number is detailed (Figure 5). Later in the next authentication request, the user must enter his user name and passphrase, and the numbers already communicated (Table 1).

Table 1. Notations

Notation	Significance
T	Size of HASH
M	Size of the block used in the rotation; M<T/2
N	The number of iterations performed in the rotation

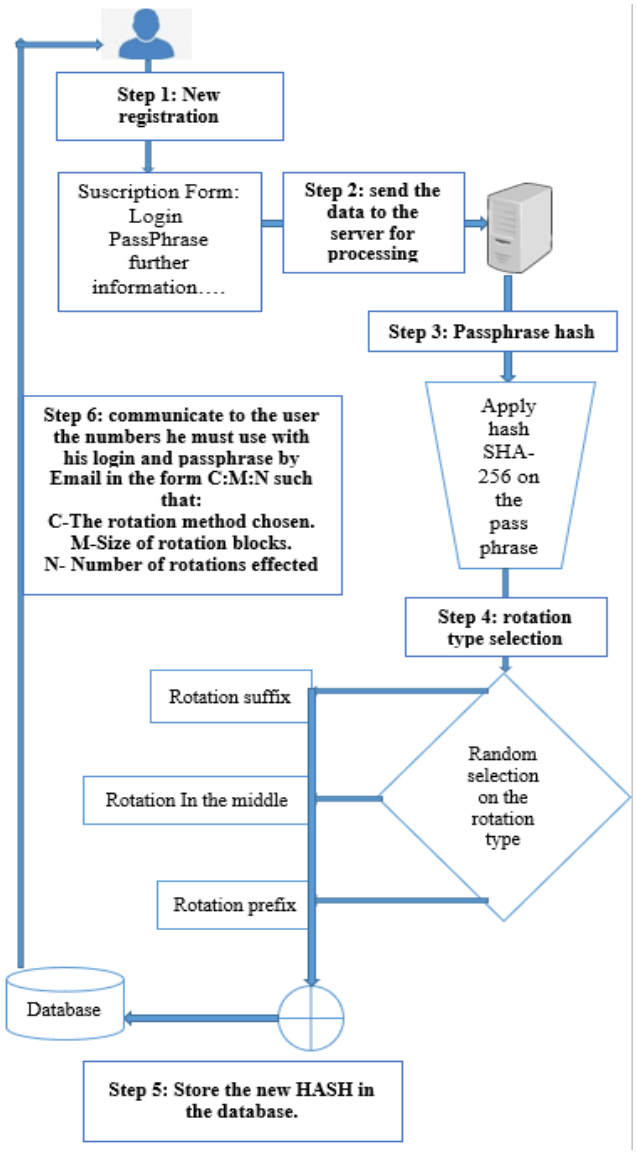


Figure 5. Operating principle H-ROTATION- "underwriting phase."

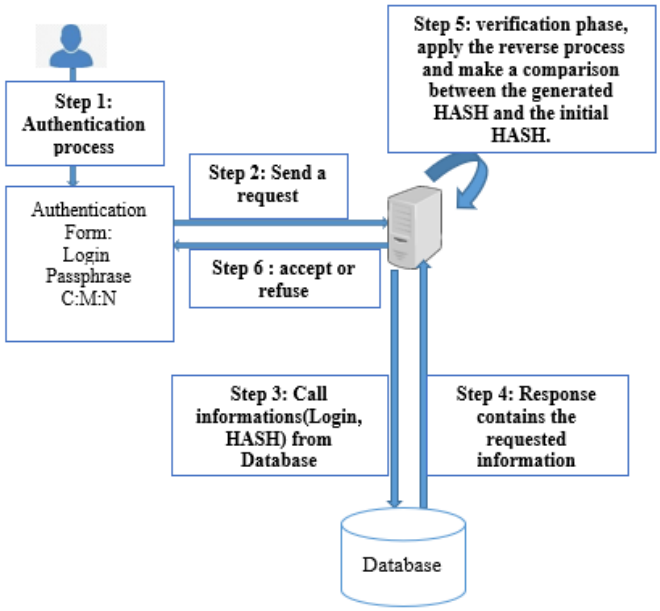


Figure 6. Operating principle H-Rotation- "Authentication phase."

H-ROTATION hides the hash stored in the database by hiding the real hash; it will be almost impossible for a malicious person to detect the real value and, therefore, the correct passphrase. For a hacker to start thinking about making an attack, it requires giant hardware at a very high level. CPU/RAM and electricity cost and will need powerful specialized devices to make the attack happen, but to start the attack not necessarily that it will be successful. Below are the different transactions between the three entities (client, server, database) described in the sequence diagram (Figure 7).

6.1 Explanation of rotation methods

This algorithm is applicable just after the hash calculation [28-30], below we will dissect the three possible types of rotations so that the HASH is not reversible to the right passphrase, i.e., the hash will have another form that will be useless for a hacker if he wants to attack the database.

Through this method, the user can use a simple passphrase to remember without passing a complex one to increase the security level, because in any case, a rotation of the generated hash will provide an additional layer of security [31, 32].

6.1.1 Rotation in prefixes

This method consists of going through a block rotation starting from right to left, as explained in Figure 8:

The system will generate two random numbers (as mentioned on the annotation table). Such as.

M: is a random number representing the characters of the HASH on which we will make the block rotations, the only condition is that M must be strictly greater than 0 and less than T/2 (size of the HASH), the reason is that we must make at least one rotation.

N: will be randomly generated from M such that N>0.

The reverse process of each case will be used to retrieve the original HASH. The server can only perform this operation.

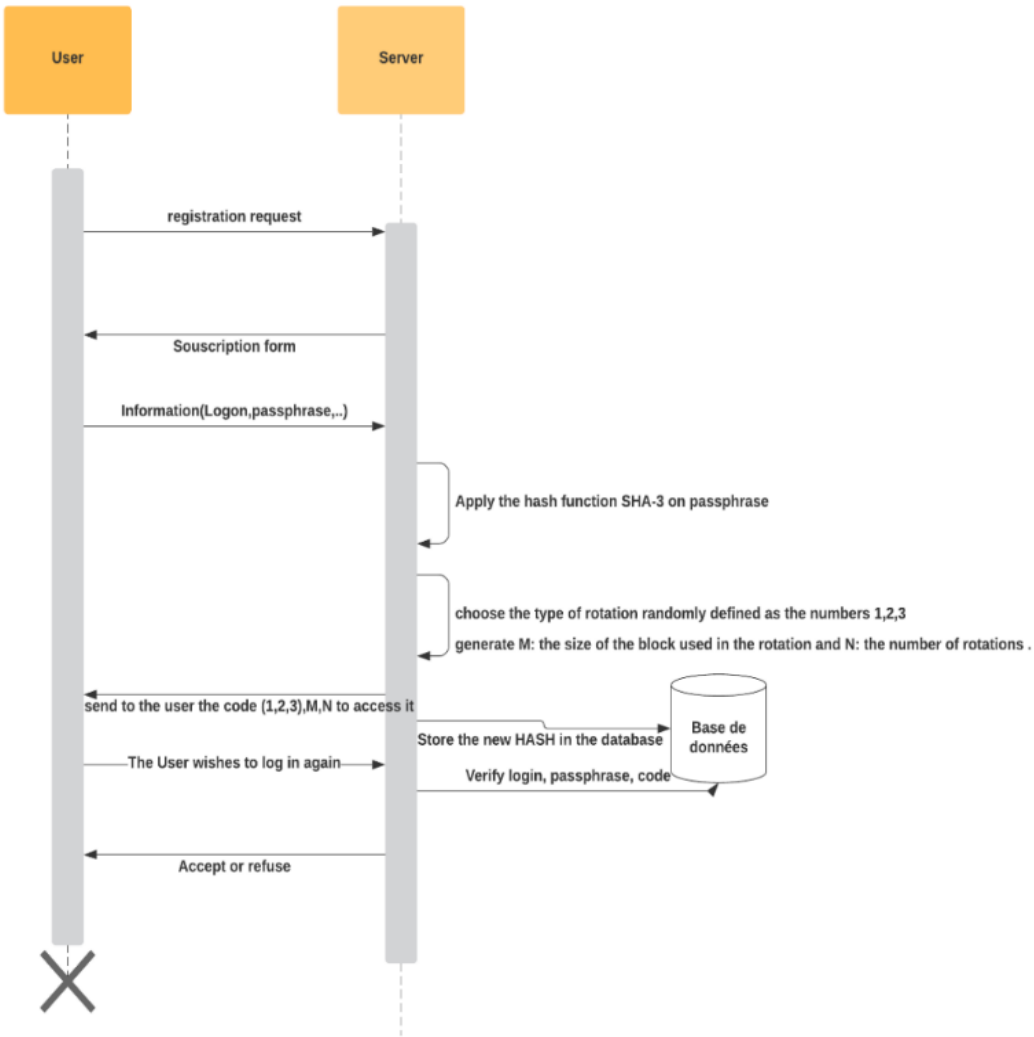


Figure 7. Sequence diagram describes the authentication process used H-Rotation

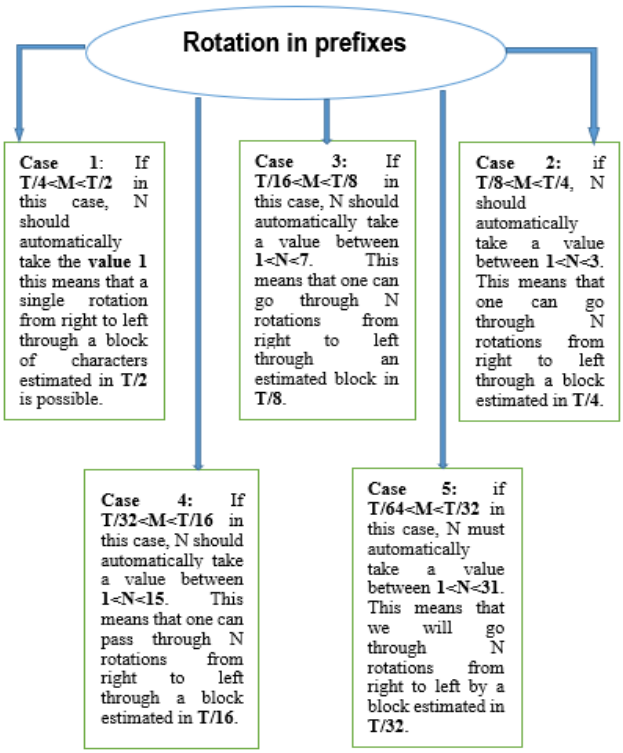


Figure 8. Different possibilities of the "Rotation in prefixes" method

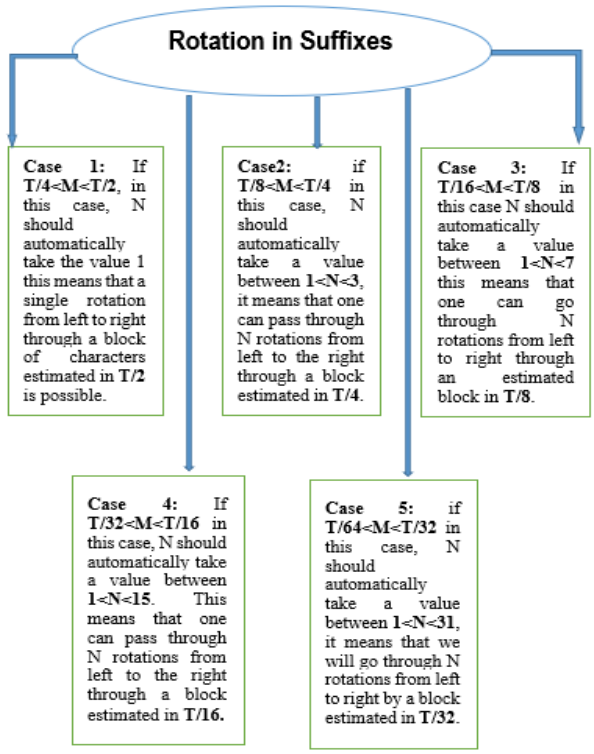


Figure 9. Different possibilities of the "Rotation in Suffixes" method

6.1.2 Rotation in suffixes

This method consists of going through a block rotation starting from the left to the right of the hash, as explained in Figure 9:

M: it is a random number presents the hash bytes on which we are going to make the rotation by block, the only condition is that M must be strictly superior to 0 and inferior to T/2 (size of the hash), the reason is that we must make at least one rotation.

N: will be randomly generated from M such that N>0.

The reverse process on each case will be used to retrieve the original HASH, this operation can only be performed by the server.

6.1.3 Rotation in the middle

This method is a bit different: the idea this time is to go through a block rotation starting in both directions towards the middle of the HASH, exactly on the T/2 position as explained in Figure 10.

The system will generate two random numbers (as mentioned on the annotation table). Such as,

M: this is a random number presents the bytes of the HASH on which we are going to rotate by block, the only condition is that M must be strictly greater than 0 and less than T/2 (size of the HASH), the reason is that we must make at least one rotation.

N: will be randomly generated from M and N>0.

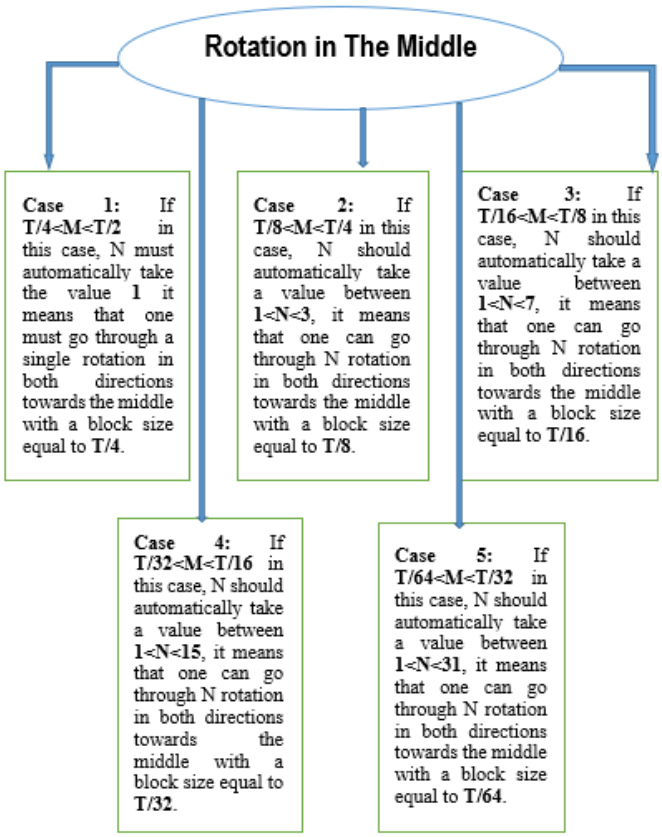


Figure 10. Different possibilities of the "Rotation in the Middle" method

6.2 Experiments

The SHA-3 256 size that will be used during the experiment is estimated to be 64 characters.

6.2.1 Rotation in prefixes

Taking the passphrase detailed in the Table 2 below:

Table 2. Hash the passphrase

Passphrase	HASH FUNCTION	HASH generated
Hamza TOUIL	SHA-3 256	1d97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6e67b7df23844

The system will generate two random numbers as detailed below and already mentioned on the annotation Table 3:

Table 3. The keys to rotation

M	N
A random integer represents how many characters make up the block with which we are going to rotate. The only condition is that M must be strictly greater than 0 and less than T/2 (size of the HASH), the reason is that we must rotate at least one block.	Generate randomly according to the M such that N>0

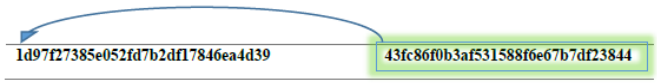
Case 1: As already mentioned, T/4<M<T/2 i.e., the number must be generated automatically in the range between 16<M<32.

Assuming that the system randomly generates M under the value 18, one can only go through one rotation, which implies that the only possible value for N is 1.

- Starting by broadcasting HASH in two parts.

1d97f27385e052fd7b2df17846ea4d39	**43fc86f0b3af531588f6e67b7df23844**

- The green block must be rotated from left to right:



- The newly generated HASH becomes:

43fc86f0b3af531588f6e67b7df238441d97f27385e052fd7b2df17846ea4d3

Case 2: T/8<M<T/4 i.e., the number must be generated automatically in the range between 8<M<16.

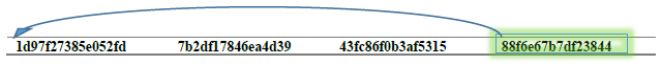
Assuming that M is randomly generated below the value 13, this implies that 1<N<3.

Assuming that N is randomly generated under the value 3, one must rotate a block of size T/4 from right to left 3 times.

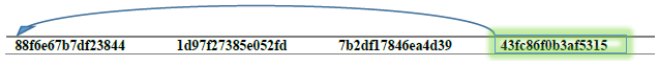
- Starting by spreading the HASH over four parts

1d97f27385e052fd	**7b2df17846ea4d39**	**43fc86f0b3af5315**	**88f6e67b7df23844**

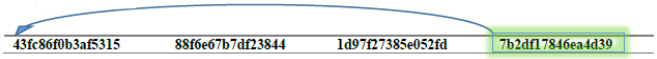
- You have to go through the first rotation of the green block from left to right:



- First rotation:



- Second rotation:



- Third rotation and the new HASH generated becomes:

7b2df17846ea4d3943fc86f0b3af531588f6e67b7df238441d97f27385e052fd

Case 3: T/16<M<T/8 i.e., the number must be generated automatically in the range between 4<M<8.

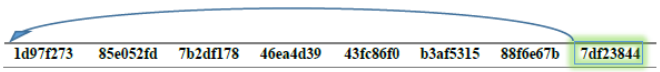
Assuming that M is randomly generated below the value 6, this implies that 1<N<7.

Assuming that N is randomly generated under the value 4, then one must rotate a block of size T/8 from right to left 4 times.

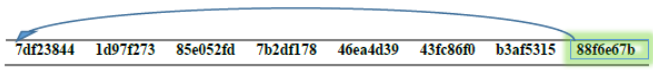
- Beginning by broadcasting the HASH on eight parts.

1d97	**85e0**	**7b2d**	**46ea**	**43fc**	**b3af**	**88f6**	**7df2**
f273	**52fd**	**f178**	**4d39**	**86f0**	**5315**	**e67b**	**3844**

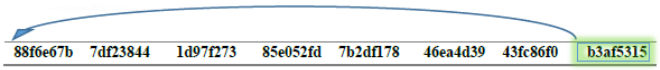
- You have to go through the first rotation of the green block from left to right:



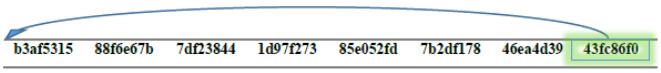
- First rotation:



- Second rotation:



- Third rotation:



- Fourth rotation and the new HASH generated becomes:

43fc86f0 b3af5315 88f6e67b 7df23844 1d97f273 85e052fd 7b2df178 46ea4d39

Case 4: T/32<M<T/16 i.e., the number must be generated automatically in the range between 2<M<4.

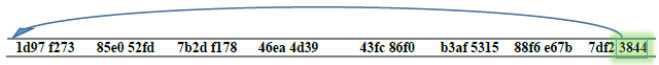
Assuming that M is randomly generated under value 3, this implies that 1<N<15.

Assuming that N is randomly generated under the value 6, then one must rotate a block of size T/16 from right to left 6 times.

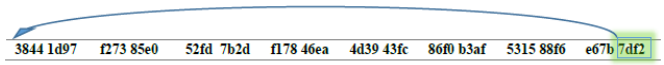
- Starting by spreading the HASH over eight parts.

1d97	**85e0**	**7b2d**	**46ea**	**43fc**	**b3af**	**88f6**	**7df23**
f273	**52fd**	**f178**	**4d39**	**86f0**	**5315**	**e67b**	**844**

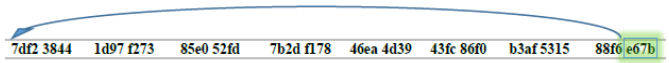
- You have to go through the first rotation of the green block from left to right:



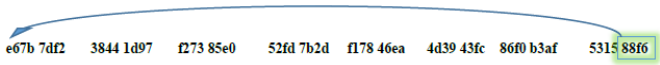
- First rotation:



- Second rotation:

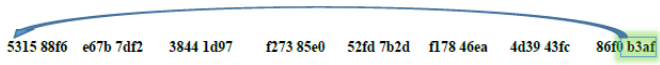


- Third rotation:

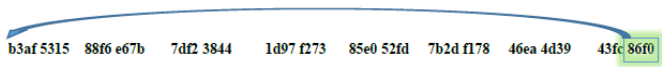


- Fourth rotation:

- fifth rotation:



- Sixth rotation:



- Seventh rotation the new HASH becomes:

86f0 b3af 5315 88f6 e67b 7df2 3844 1d97 f273 85e0 52fd 7b2d f178 46ea 4d39 43fc

Case 5: T/64<M<T/32 i.e. the number must be generated automatically in the range 1<M<2.

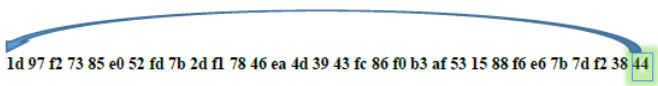
Assuming that M is randomly generated under value 2, this implies that 1<N<31.

Assuming that N is randomly generated under the value 31, then one must rotate a block of size T/16 from right to left 31 times.

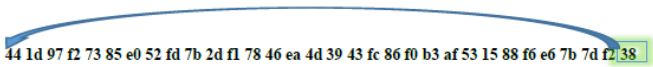
- Starting by spreading the HASH over 32 parts

1d 97 f2 73 85 e0 52 fd 7b 2d f1 78 46 ea 4d 39 43 fc 86 f0 b3 af 53 15 88 f6 e6 7b 7d f2 38 44

- First rotation:



- Second rotation:



- 31st rotation the new HASH becomes:

97 f2 73 85 e0 52 fd 7b 2d f1 78 46 ea 4d 39 43 fc 86 f0 b3 af 53 15 88 f6 e6 7b 7d f2 38 44 1d

The recursive operation on each case will be used to recover the original HASH.

6.2.2 Rotation in suffixes

Taking the passphrase detailed in the Table 4 below:

Table 4. Hach the passphrase

Passphrase	HASH FUNCTION	HASH generated
Nabil EL AKKAD	SHA-3 256	9e0f5121782399b5e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d11

The system will generate two random numbers as detailed below and already mentioned on the annotation Table 5:

Table 5. The keys to rotation

M	N
A random integer represents how many characters make up the block with which we are going to rotate, the only condition is that M must be strictly greater than 0 and less than T/2 (size of the HASH), the reason is that we must rotate at least one block.	Generate randomly according to the M such that N>0

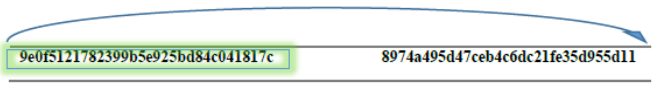
Case 1: as already mentioned T/4<M<T/2 i.e. the number must be generated automatically in the range between 16<M<32.

Assuming that the system randomly generates M under the value 20, this means that one can only go through one rotation which implies that the only possible value for N is 1.

- Starting by spreading the HASH over two parts.

9e0f5121782399b5e925bd84c041817c	**8974a495d47ceb4c6dc21fe35d955d11**

- The green block must be rotated from right to left:



- The newly generated HASH becomes:

8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5 e925bd84c041817c

Case 2: T/8<M<T/4 i.e. the number must be generated automatically in the range between 8<M<16.

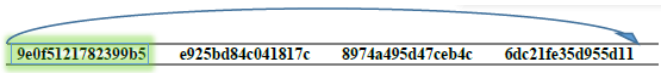
Assuming that M is randomly generated below the value 15, this implies that 1<N<3.

Assuming that N is randomly generated under the value 2, then one must rotate a block of size T/4 from left to right 2 times.

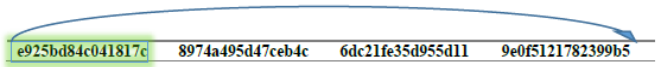
- Starting by spreading the HASH over four parts

9e0f5121782399b5	**e925bd84c041817c**	**8974a495d47ceb4c**	**6dc21fe35d955d11**

- You have to go through the first rotation of the green block from right to left:



- First rotation:



- Second rotation the new HASH becomes:

8974a495d47ceb4c 6dc21fe35d955d11 9e0f5121782399b5 e925bd84c041817c

Case 3: T/16<M<T/8 i.e. the number must be generated automatically in the range between 4<M<8.

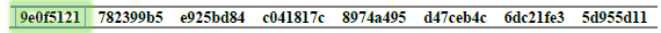
Assuming that M is randomly generated under the value 4, this implies that 1<N<7.

Assuming that N is randomly generated under the value 4, then one must rotate a block of size T/8 from left to right 4 times.

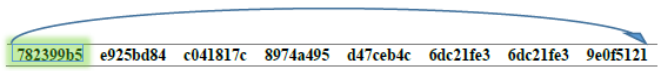
- Starting by spreading the HASH over eight parts

9e0f 5121	**7823 99b5**	**e925 bd84**	**c041 817c**	**8974 a495**	**d47c eb4c**	**6dc2 1fe3**	**5d95 5d11**

- You have to go through the first rotation of the green block from right to left:



- First rotation:



- Continuing in the same way the other three iterations and the new HASH becomes:

8974a495 d47ceb4c 6dc21fe3 5d955d11 9e0f5121 782399b5 e925bd84 c041817c

Case 4: T/32<M<T/16 i.e. the number must be generated automatically in the range between 2<M<4.

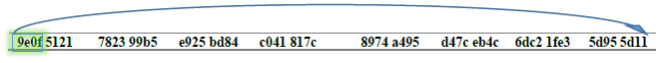
Assuming that M is randomly generated under value 3, this implies that 1<N<15.

Assuming that N is randomly generated under the value 3, then one must rotate a block of size T/16 from left to right to left 6 times.

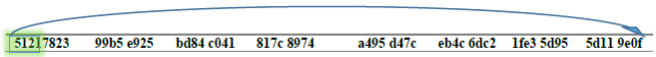
- Beginning by spreading the HASH over eight parts

9e0f 5121	**7823 99b5**	**e925 bd84**	**c041 817c**	**8974 a495**	**d47c eb4c**	**6dc2 1fe3**	**5d95 5d11**

- You have to go through the first rotation of the green block from the Right to the Left:



- First rotation:



- Continuing in the same way the other two iterations and the new HASH becomes:

99b5e925bd84c041817c8974a495d47ceb4c6dc21fe35d95 5d119e0f 51217823

Case 5: T/64<M<T/32 i.e. the number must be generated automatically in the range 1<M<2.

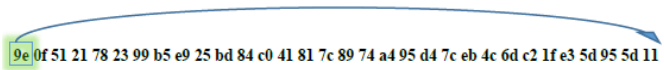
Assuming that M is randomly generated under the value 2, this implies that 1<N<31.

Assuming that N is randomly generated under the value 2, then one must rotate a block of size T/16 from right to left 31 times.

- Beginning by spreading the HASH over 1 part.

9e 0f 51 21 78 23 99 b5 e9 25 bd 84 c0 41 81 7c 89 74 a4 95 d4 7c eb 4c 6d c2 1f e3 5d 95 5d 11

- First rotation:



- The new HASH becomes:

0f5121782399b5e925bd84c041817c8974a495d47ceb4c6dc 21fe35d955d119e

The recursive operation on each case will be used to recover the original HASH.

6.2.3 Rotation in the middle

Taking the passphrase detailed in the Table 6 below:

Table 6. Hach the passphrase

Passphrase	HASH FUNCTION	HASH generated
H-Rotation: Secure storage and retrieval of passphrases on the authentication process	SHA-3 256	1be0009ba9981a905f edb24d5ffff9e76d 97d0b0f62327de451d a850e2a0e4cb

The system will generate two random numbers as detailed below and already mentioned on the annotation Table 7:

Table 7. The keys to rotation

M	N
A random integer represents how many characters make up the block with which we are going to rotate, the only condition is that M must be strictly greater than 0 and less than T/2 (size of the HASH), the reason is that we must rotate at least one block.	Generate randomly according to the M such that N>0

This method is a bit different because the idea is to go through a block rotation starting in both directions towards the middle of the T/2 HASH:

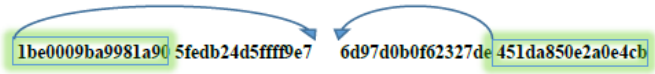
First, we have to identify the middle of the HASH, it's simply the T/2 position.

1be0009ba9981a905fedb24d5ffff9e7 6d97d0b0f62327de451da850e2a0e4cb

Case 1: T/4<M<T/2 i.e. the number must be generated automatically in the range between 16<M<32.

Assuming that the system randomly generates M under the value 20, this means that one can only go through one rotation which implies that the only possible value for N is 1.

Then one must rotate T/4 in both directions towards the middle only once:



The new HASH becomes:

5fedb24d5ffff9e71be0009ba9981a90451da850e2a0e4cb6d 97d0b0f62327de

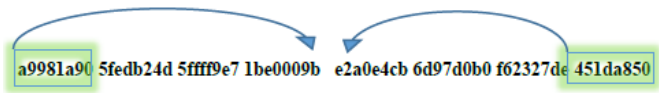
Case 2: T/8<M<T/4 i.e. the number must be generated automatically in the range between 8<M<16.

Assuming that the system generates M randomly under the value 8 this implies that 1<N<3.

Assuming that N is randomly generated under the value 2, then one has to rotate T/8 in both directions towards the middle twice:

- First rotation:

- Second rotation:

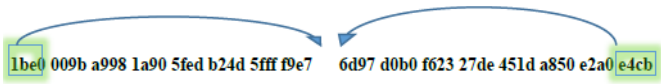


The new HASH becomes:

5fedb24d5ffff9e71be0009ba9981a90451da850e2a0e4cb6d 97d0b0f62327de

Case 3: T/16<M<T/8 i.e. the number must be generated automatically in the range between 16<M<32.

Automatically, 1<N<7 this means that we can go through N rotation in both directions towards the middle with a block size equal to T/16, in the same way explained previously.



Case 4: T/32<M<T/16 i.e. the number must be generated automatically in the range between 32<M<64.

Automatically, 1<N<15 means that N rotation in both directions towards the middle with a block size equal to T/32, in the same way explained previously.

Case 5: T/64<M<T/32 i.e. the number must be generated automatically under the value 64.

Automatically, 1<N<31 means that N rotation in both directions towards the middle with a block size equal to T/32 is possible, in the same way as explained above.

The recursive operation on each case will be used to recover the original HASH.

6.3 Comparison between our algorithm and existing algorithms in terms of security

Contrary to existing methods (Table 8 below), our algorithm can withstand the most potent passphrase attacks due to the complexity of the Hash used with the same components but with different formats and the impossibility to find the exact passphrase in clear text from the passphrase generated after the hacking which is among a huge number of false propositions. Moreover, our system is protected against theft the attack of the verifiers, so even possessing the passphrase, the hacker still needs to secretly use represent on the rotations performed.

Table 8. Comparison between the different types of authentication, i.e., those based on the Hashage function

Methods/ Attack types	Man In The Middle	Statistical attack	Password Cracking	DOS
H-Rotation	Resists	Resists	Resists	Resists
OSPA [33]	No Resists	No Resists	No Resists	No Resists
PERM [18]	No Resists	No Resists	No Resists	No Resists
Authentication Digital [16]	Resists	Resists	Resists	No Resists

6.4 Typical attacks failures

- **Dictionary attack:** This kind of attack based on iterating over all strings in a pre-compiled list. These attacks initially used words that can be found in a dictionary (hence the dictionary attackphrases); however, much larger lists are now available on the open Internet, containing hundreds of millions of passwords recovered from past data breaches. There is also a jailbreak program that can use such lists and create common variations, such as replacing similar letters with numbers. Dictionary attack checks only those possibilities that are considered most likely. Based on our approach, this attack has no more risk on user passphrases. Even if the cracker retrieves the passphrase, he will need the CMN parameters generated beforehand during the registration. if not, the hacker must try three combinations (C:M:N) + phrase pass, thing that appears almost impossible.

- **Brute-force attack:** as explained an attacker presenting many passwords or passphrases in the hope of ultimately guessing the combination correctly. The attacker systematically checks all possible passwords and passphrases until the correct one is found. Alternatively, an attacker can try to guess the key, which is usually generated from a password, using the get key function. This exhaustive key search will be of no use to the hacker as the hash is transformed before storage in the database, and the match will lead to another password or passphrase, in addition to our approach capable of disrupting malevolence.

- **Rainbow Table Attack:** Based on our method, this attack becomes falsified because the hacker will never find the Passphrase/hash match stored in the database. Because, after the rotation, the stored hash represents another passphrase or password different from the original one.

6.5 Setting up

If the customer wishes to authenticate, he must go through the form (Figure 11) by typing the Login, passphrase, and the three numbers received already after the subscription to access his account.

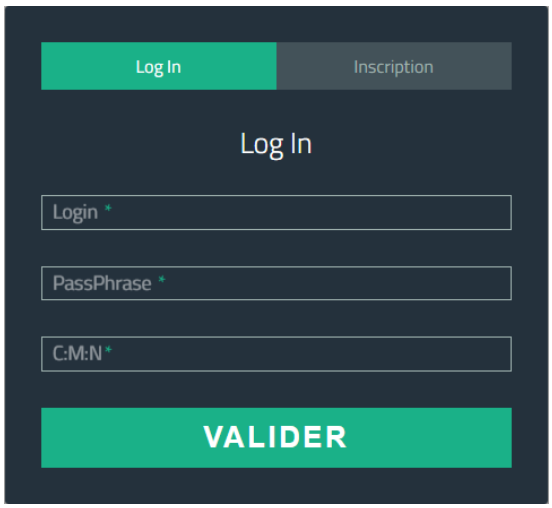


Figure 11. Authentication form

7. CONCLUSION

We have deconstructed a new algorithm specializing in authentication security that will anticipate the detection of passphrase cracking attacks that aim to violate personal data for any user through the access to his account.

Our method offers a relatively high level of security. Even if a hacker manages to access our database and decrypt the HASH, he will inevitably come across a value that has nothing to do with the passphrase stored with good user experience and very low delays.

The security of passwords/passwords has always been a significant concern for users and organizations because it is the access point that can provide reliability to the system being used.

REFERENCES

[1] Thorawade, M.B., Patil, S.M. (2012). Authentication scheme resistant to shoulder surfing attack using image retrieval. International Journal of Knowledge Engineering, 3(2): 197-201.

[2] Porter, S.N. (1982). Porter: A password extension for

improved human factors. Computers and Security, 1(1): 54-56. https://doi.org/10.1016/0167-4048(82)90025-6

[3] Still, J.D., Cain, A., Schuster, D. (2017). Human-centered authentication guidelines. Information and Computer Security, 25(4): 437-453. https://doi.org/10.1108/ICS-04-2016-0034

[4] Burr, W.E., Dodson, D.F. Polk, W.T. (2017). Electronic Authentication Guideline. NIST Special Publication. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf.

[5] Es-Sabry, M., El Akkad, N., Merras, M., Saaidi, A., Satori, K. (2020). A new image encryption algorithm using random numbers generation of two matrices and bit-shift operators. Soft Computing, 24: 3829-3848. https://doi.org/10.1007/s00500-019-04151-8

[6] Es-sabry, M., El Akkad, N., Merras, M., Saaidi, A., Satori, K. (2018). Grayscale image encryption using shift bits operations. International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, pp. 1-7. https://doi.org/10.1109/ISACV.2018.8354028

[7] Es-sabry, M., El Akkad, N., Merras, M. Saaidi, A., Satori, K. (2018). A novel text encryption algorithm based on the two-square cipher and Caesar cipher. In: Tabii Y., Lazaar M., Al Achhab M., Enneya N. (eds) Big Data, Cloud and Applications. BDCA 2018. Communications in Computer and Information Science, vol 872. Springer, Cham. https://doi.org/10.1007/978-3-319-96292-4_7

[8] Elazzaby, F., El Akkad, N., Kabbaj, S. (2020). A new encryption approach based on four-square and zigzag encryption (C4CZ). In: Bhateja V., Satapathy S., Satori H. (eds) Embedded Systems and Artificial Intelligence. Advances in Intelligent Systems and Computing, vol 1076. Springer, Singapore. https://doi.org/10.1007/978-981-15-0947-6_56

[9] Es-sabry, M., El Akkad, N., Merras, M., Saaidi, A., Satori, K. (2020). A new color image encryption using random numbers generation and linear functions. Advances in Intelligent Systems and Computing, 1076: 581-588.

[10] Soni, M., Patel, T., Jain, A. (2020). Security analysis on remote user authentication methods. In: Pandian A., Senjyu T., Islam S., Wang H. (eds) Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018). ICCBI 2018. Lecture Notes on Data Engineering and Communications Technologies, vol 31. Springer, Cham. https://doi.org/10.1007/978-3-030-24643-3_60

[11] Hariharan, R.D. (2020). Enhancing the security authentication on system passwords using graphical methods. Test Engineering and Management, 83: 11314-11318.

[12] Kumar, S., Jafri, S.A.A., Nigam, N., Gupta, N., Gupta, G., Singh, S.K. (2019). A new user identity based authentication, using security and distributed for cloud computing. IOP Conf. Ser.: Mater. Sci. Eng., 748: 012026. https://doi.org/10.1088/1757-899X/748/1/012026

[13] Matsunaga, S., Adachi, N. (2019). Message authentication scheme for ad hoc networks with homomorphic hash function. 2019 2nd World Symposium on Communication Engineering (WSCE), Nagoya, Japan, pp. 138-141. https://doi.org/10.1109/WSCE49000.2019.9040951

[14] Bebe, P.C., Akila, D (2019). Orchini similarity user authentication based Streebog hash function for secured data storage in cloud. 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, pp. 461-466. https://doi.org/10.1109/ICCIKE47802.2019.9004393

[15] Nielsen, G., Vedel, M., Jensen, C.D. (2014). Improving usability of passphrase authentication. 2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, pp. 189-198. https://doi.org/10.1109/PST.2014.6890939

[16] Arya, M.S., Rani, M., Bedi, C.S. (2016). Improved capacity image steganography algorithm using 16-pixel differencing with n-bit LSB substitution for RGB images International Journal of Electrical and Computer Engineering, 6(6): 2735-2741. https://doi.org/10.11591/ijece.v6i6.10792

[17] Aumasson, J.P., Meier, W., Phan, R.C.W. (1998). Information Security and Cryptography. Springer-Verlag Berlin Heidelberg.

[18] Shimizu, A., Horioka, T., Inagaki, H. (1998). A password authentication methods for contents communication on the Internet. IEICE Transactions on Communications, pp 1666-1673.

[19] Gajula, R., Qyser, A.A.M., Rajender, N. (2017). Combining two factor authentication and public key encryption to ensure the authentication in cloud computing. International Journal of Recent Trends in Engineering and Research, pp. 118-121.

[20] Burr, W.E., Dodson, D.F., Polk, T.W. (2011) Electronic authentication guideline: information security. NIST Special Publication 800-63-1.

[21] van Thanh, D., Jorstad, I., Jonvik, T. (2009). Strong authentication with mobile phone as security token. 2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, Macau, pp. 777-782. https://doi.org/10.1109/MOBHOC.2009.5336918

[22] Gupta, B.B. (2019) Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives. CRC Press. 666-695.

[23] Bhargavan, K., Leurent, G. (2016). Transcript collision attacks: Breaking authentication in TLS, IKE and SSH NDSS 2016. Network and Distributed System Security Symposium – NDSS 2016, Feb 2016, San Diego, United States. https://doi.org/10.14722/ndss.2016.23418

[24] Leurent, G., Peyrin, T. (2020). SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. Proceedings of the 29th USENIX Security Symposium2020, pp. 1839-1856

[25] Ramya, P., Sairamvamsi, T. (2018). Impact analysis of blackhole, flooding, and grayhole attacks and security enhancements in mobile ad hoc networks using SHA_3 algorithm In: Anguera J., Satapathy S., Bhateja V., Sunitha K. (eds) Microelectronics, Electromagnetics and Telecommunications. Lecture Notes in Electrical Engineering, vol 471. Springer, Singapore. https://doi.org/10.1007/978-981-10-7329-8_65

[26] Zhang, L., Tan, C., Yu, F. (2017). An improved rainbow table attack for long passwords. Procedia Computer Science, 107: 47-52. https://doi.org/10.1016/j.procs.2017.03.054

[27] De La Hoz, E., Cochrane, G., Moreira-Lemus, J.M., Paez-Reyes, R., Marsa-Maestre, I., Alarcos, B. (2014). Detecting and defeating advanced man-in-the-middle

attacks against TLS. 2014 6th International Conference on Cyber Conflict (CyCon 2014), Tallinn, pp. 209-221. https://doi.org/10.1109/CYCON.2014.6916404
[28] Kundi, D., Aziz, A. (2016). A low-power SHA-3 designs using embedded digital signal processing slice on FPGA. Computers and Electrical Engineering, 55: 138-152. https://doi.org/10.1016/j.compeleceng.2016.04.004
[29] Wu, L., Weaver, C., Austin, T. (2001). CryptoManiac: A fast flexible architecture for secure communication. Proceedings 28th Annual International Symposium on Computer Architecture, Goteborg, Sweden, pp. 110-119. https://doi.org/10.1109/ISCA.2001.937439
[30] Sayilar, G., Chiou, D. (2014). Cryptoraptor: High throughput reconfigurable cryptographic processor. 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, pp. 155-161. https://doi.org/10.1109/ICCAD.2014.7001346
[31] Touil, H., El Akkad, N., Satori, K. (2020). Text Encryption: Hybrid cryptographic method using Vigenere and Hill Ciphers. 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, pp. 1-6. https://doi.org/10.1109/ISCV49265.2020.9204095
[32] Elazzaby, F., El Akkad, N., Kabbaj, S. (2020). Advanced encryption of image based on S-box and chaos 2D (LSMCL). 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Meknes, Morocco, pp. 1-7. https://doi.org/10.1109/IRASET48871.2020.9092254
[33] Lin, C., Sun, H., Hwang, T. (2001). Attacks and solutions on strong-password authentication, IEICE Transactions on Communications, 84(9): 2622-2627.