JSH Algorithm: A Password Encryption Technique using Jumbling-Salting-Hashing

Prathamesh Churi
Shah and Anchor Kutchhi Engineering College, University of Mumbai, Mumbai, India

Medha Kalelkar
Vidyalankar Institute of Technology, University of Mumbai, Mumbai, India

Bhavin Save
Mahatma Gandhi Mission's College of Engineering, University of Mumbai, Mumbai, India

ABSTRACT
This paper presents a new algorithm for improvising password encryption using Jumbling-Salting-Hashing technique. One of the most important password protection issue is to secure encrypted passwords on server's database. In cryptanalysis, a dictionary attack or brute force attack are the most common ways of guessing passwords. In order to augment the security aspect regarding passwords, we are devising JSH algorithm which will be responsible for preventing dictionary or brute force attacks on the passwords. In this algorithm, the jumbling process consists of randomly selecting characters from pre-defined character set and adding them into the plain password; salting comprises of prepending a random string and hashing process is implemented using cryptographic hash function to obtain a fixed length "fingerprint" which is stored in the server's database. As JSH algorithm deals with randomization, the password encryption technique forms a highly secured form of encrypted password which makes it difficult to decrypt reducing the probability of guessing password.

General Terms
Security, Authentication, Encryption, Algorithms.

Keywords
JSH, jumbling, salting, hashing, process array.

1. INTRODUCTION
According to Bruce Schneider "Security is a process, not a product."[1][7] This famous quote is well echoed by the phenomenon that although there exist umpteen number of security techniques today, none of these tools can single-handedly address all the security goals of an organization.

As password is an authentication technique which provides the claimant access to system resources, it is an important aspect of security. Being the simplest form of authentication technique used, the probability of attacking the password is considerably high. The most common attack to obtain a password is by attacking the server's database consisting of a list of passwords. Although password encryption provides solution to prevent such attacks, "brute force attack" or "dictionary attack" have proven this password encryption technique to be futile. To overcome the problem of securing encrypted password, we are developing JSH technique which will provide additional security to the stored passwords.

JSH algorithm consists of three techniques namely jumbling, salting and hashing. In the jumbling part, the password undergoes "addition", "selection" and "reverse" processes. Addition process is responsible for generating a value required for determining the number of characters to be added to the password. Selection deals with selecting characters to be added to the password from predefined character set. Reverse process is responsible for reversing the output of selection process on some predefined condition. In salting part, random salt is added to the jumbled password. Selection of salt is based on timestamp value which is determined when the user creates an account. Finally, jumbled and salted password is given to the hashing procedure where predefined hashing algorithm such as SHA algorithm is implemented.

Randomized algorithms are particularly useful when attacker who deliberately tries to perform dictionary or brute-force attack. It is for this reason that randomness is ubiquitous in cryptography. At the heart of all cryptographic systems is the generation of secret, pattern. Each and every stage of JSH algorithm is randomized, hence we can achieve "Randomness in Security".

2. RELATED WORK
There are some existing algorithms for encrypting the password which do not fulfill all the aspects of security. However dictionary or brute force attacks on server side are done by performing some permutations within the known password parameters (such as max length of password) and are hashed to try and generate the same hash value. [6]

There are some predefined hashing algorithms available today that do not offer complete security to the passwords. Here are some hashing algorithms mentioned below:

a. SHA: There are four Secured Hash Algorithms namely SHA-0, SHA-1, SHA-2, and SHA-3 [5]. Specifically, SHA-1 produces a hash value of 160 bits. A SHA-1 hash value typically forms a hexadecimal number, 40 digits long. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses.

b. MD5: The MD5 (Message-Digest algorithm) is a widely used cryptographic hash function. It forms a 128-bit hash value which is expressed in text format as a 32 digit hexadecimal number. Due to its effectiveness, MD5 has been utilized in many cryptographic applications. Also, it is commonly used to verify data integrity. MD5 is not apt for applications like SSL certificates or digital signatures. [4]

It can be easily inferred that the simplest way to crack a hash is to try to guess the password, hash each guess, and checking if the guess's hash equals the hash being cracked. If the hashes are found to be equal, the guess is the password which is the basic principle of dictionary attack. The effectiveness of dictionary attacks or brute force attacks can be reduced using hashing and salting techniques like SHA or MD5.

However, along with hashing and salting, JSH algorithm additionally implements "Jumbling" technique which will further reduces the probability of guessing password drastically. Thus, this increase in probability actually refers to "Randomness in Security".

The central question regarding to security provided by JSH algorithm is: "Whether JSH algorithm is more difficult to crack or not?" The answer to this question can be tabulated below:

Table 1. Comparison between JSH and other existing hashing algorithms.

Functions	JSH Algorithm	Existing algorithms (SHA, MD5)
Decryption Level	Difficult , due to addition of jumbling block	Can be decrypted using Dictionary attack
Randomness	More randomized	Less randomized
Implementation level	More difficult due to randomization in jumbling technique, production of salting and hashing	Difficult , due to the production of hash value
Cryptographic processes involved	Jumbling + Salting + Hashing	Salting (Optional) + hashing
Selection of salt	User's sign-up timestamp value	Any random string

3. OVERVIEW

JSH algorithm consist of three major processes; Jumbling, Salting, Hashing. Jumbling process includes three sub-processes viz. addition, selection, and reverse process. The block diagram of JSH algorithm is given in Fig. 1. The input to JSH algorithm is plain-text password which is stored in Process array. The description of three blocks is given below:

3.1 Jumbling Block:

Process array is given to Jumbling block. Jumbling block is responsible for prepending some characters from character set and jumbling them with the help mod function. Jumbling block itself is a combination of three sub-blocks:

Addition sub-block: This block deals with is nothing but generating principle random value 'l' and updating the size of a Process array.

Selection sub-block: This block is about selecting characters from predefined character set A. The size of character array is large and the character set for a particular password entry is different. Selection of characters is based on the random values which are generated 'l' times.

Reverse sub-block: This block reverses the entire process array based on some predefined condition. The predefined condition is to check the value of 'l' is even or odd. If the value of 'l' is even then we reverse the process array else we keep it as it is.

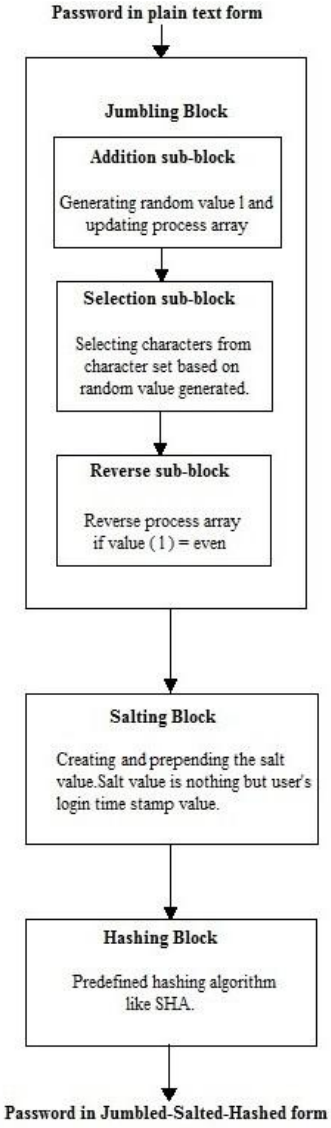

Fig. 1: Password encryption using JSH algorithm

3.2 Salting Block:

The objective of salting block is to add random string along with jumbled version of password. The criterion of selection of salt is user's sign-up timestamp value. The salt is added in order to make the password more complicated thereby making it difficult for the attacker to obtain it.

3.3 Hashing Block:

In hashing, we use predefined hashing algorithms such as SHA. The alternative to SHA algorithm can also be used (such as MD5).

4. ALGORITHM

The pseudo code implementation of JSH is given below:

// **Random ( )**: It is a predefined method which is responsible for calling random value from predefined set of objects.

// **Process array P [ ]**: This array stores the actual plain-text password along with randomly generated characters. We are using this array for actual encryption process.

// **Salt array S[ ]:** This is use for storing timestamp value from user. In this case, timestamp value will be nothing but user's sign-up time value.

// **input:** password in plain-text form.

// **output:** password in Jumbled, salted and hashed form.

INITIALIZE 'x' to 0;

STORE the length of plain text input password in variable 'x';

CREATE an Process array P[ ] such that P[length =x];

STORE each character in an array block;

// P {0, 1, 2... x-1) = {characters in password}

/* implementation of Jumbling Technique */

function jumbling (P[ ])

{

// implementation of jumbling technique: Addition Process

A.

Label 1 : CALL Random() function;

// Random function returns any random value from predefined set of integers.

SET 'l' as principle random value;

If (l >= x)

STORE random number value generated from random() function;

Else

goto **Label 1** ;

break;

End If

B.

UPDATE an array P[ ] of size (x + l) as shown :

// this array is referred as "Process array"

Process array of size (x + l)

C.

DEFINE the set of characters A.

Size (A) = M;

M = any large value;

A = {A....Z, a....z, 0....9, special characters, operators};

//Character set for a particular password entry should be different;

//implementation of jumbling technique: Selection Process

/* This process is responsible for selecting characters from given character set. All these symbols later on added with plain-text password. These process is also randomized */

A.

CALL random ( ) function 'l' times;

// At each iteration, random value is generated which acts as an index of the character in character set.

/* for Example : character set A ={ %,$,C,7,*,y,W,8,+, |, @ }

CALL Random( )

number generated: 2

hence character selected : C */

B.

FILL the process array with characters as shown:

---

x(password characters) | l (selected characters)

C.

STORE the original length of an array (x + l) in variable 'FIX'

For i= 0 to (x+l-1)

While (l !=0)

SET j to 0;

j= (FIX mod l);

Create 'temp' variable;

temp = P [j] ;

P [j] = P [i];

P [i] = temp;

//output of above mod function is nothing but index within the range 0 to (x + l - 1), Hence we must swap output index 0th position.

l= l-1;

End While

End For

// implementation of jumbling technique: Reverse process

A.

If (l mod 2 == 0)

Reverse the process array; // l is EVEN number

Else

Do not reverse the process array; // l is ODD number

return (P); // pass the process array to salting function

End If

} // end of jumbling function

/* implementation of Salting Technique */

function salting(P[ ])

{

A.

STORE Timestamp value of Sign-up process for each user;

OBTAIN the length of timestamp as 't';

CREATE Salt [size=t] array which stores random salt characters;

Salt [ ] = {characters obtained from Timestamp value};

UPDATE an array P of size (x +l + t) as shown:

Process array will become

Process array of size (x + l + t)

B.

FILL the process array with

Jumbled password of size (x + l) | salt characters of size t

---

return (P); // pass the process array to hashing function

} **// end of salting function**

/* implementation of Hashing Technique */

function hashing(P [ ])

{

A.

Use of predefined SHA algorithm (SHA0, SHA1, SHA2, and SHA3) can be implemented in Hashing function; [2]

}**// end of hashing function**

5. LIMITATIONS

JSH algorithm does not completely fulfill the requirements of all the authentication aspects of security. However, it tries to improve the difficulty level of decryption of password for any unauthorized person (or attacker) using the process of randomization.

Another important issue is to secure random number 'l' on server side. 'l' is the random number generated during jumbling process which informs about the number of characters to be added. However, the value of 'l' must be stored for future processing of the algorithm (say decryption). Storing the value of 'l' might be an issue as the attacker can try to obtain the value of 'l'.

6. FUTURE SCOPE

The future scope of algorithm includes some modifications that can be implemented instead of MOD function which is used in Jumbling step of JSH algorithm.

Alternative hashing technique other than SHA like MD5 or a self designed hashing technique can also be used. [3], [4]

7. CONCLUSION

The two most common ways of guessing passwords are dictionary attacks or brute-force attacks. There is no way to prevent dictionary attacks or brute force attacks. They can be made less effective, but there isn't a way to prevent them altogether. JSH technique however reduce the probability of cracking passwords.

Due to involvement of different randomization processes, JSH algorithm builds an encrypted version of password which is almost difficult to decrypt. JSH technique however reduces the probability of cracking the passwords.

8. ACKNOWLEDGEMENT

The authors wish to thank Prof. Vaishali Ghate, Assistant Professor of Shah and Anchor Kutchhi Engineering College, India for her valuable guidance.

9. AUTHORS

First Author – Prathamesh P. Churi is currently pursuing Master's program in Information Technology from Shah and Anchor Kutchhi Engineering College affiliated to University of Mumbai, He has completed his Graduate program in Computer Engineering from Vidyalankar Institute of Technology affiliated to University of Mumbai, India. E-mail: prathamesh.churi@gmail.com.

Second Author – Medha D. Kalelkar has completed Graduate program in Computer Engineering from Vidyalankar Institute of Technology affiliated to University of Mumbai, India. E-mail: kalelkar.medha@gmail.com

Third Author – Bhavin D. Save has completed Graduate program in Computer Engineering from Mahatma Gandhi Mission's College of Engineering affiliated to University of Mumbai, India
E-mail: bhavinsave@gmail.com

10. REFERENCES

[1] Secure password methods(salting and hashing methods) , https://crackstation.net/hashing-security.htm#salt

[2] SHA-1 algorithm implementation details http://tools.ietf.org/html/rfc3174

[3] MD5 algorithm concept and implemenation, http://www.ietf.org/rfc/rfc1321.txt

[4] MD5 Algorithm,http://en.wikipedia.org/wiki/MD5

[5] http://en.wikipedia.org/wiki/SHA-1

[6] http://programmers.stackexchange.com/questions/90211/how-would-i-go-about-changing-encryption-methods-on-existing-passwords

[7] Applied Cryptography-Protocols, Algorithms and Source Code in C , John Wiley Publications.