IOSR Journal of Engineering (IOSR JEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 15-17

Image Encryption Using Jumbling Salting

Akash Tiwari[1], Venkatachalam Muthukrishnan Iyer[2] , Mayuresh Vartak[3], Puneet Verma[4], Renuka Nagpure[5], Komal Gothwal[6]

[1,2,3,4]*(Department of Information Technology, Atharva College of Engineering, Mumbai-95, India)*
[5,6]*(Assistant Professor, Department of Information Technology, Atharva College of Engineering, Mumbai-95)*

Abstract : *Today almost all digital services like internet communication, medical and military imaging systems, the multimedia system requires reliable security in storage as well as in the transmission of digital images. There becomes a necessity for security in digital images because of the faster growth in the fields of multimedia technology, internet and cellphones. Hence, it becomes a need for image encryption techniques in order to hide images from such attacks. In this system, we use JS (Jumbling Salting) algorithm in order to hide the image. JS algorithm is being executed in .NET framework. It provides a brand new access to assure high-level security of information as required in the fields of aerospace, military, confidential, financial and economic, national security and so on. To know the security aspect regarding images, we are devising JS algorithm. The image encryption technique forms a highly secured form of the encrypted image which makes it difficult to decrypt reducing the probability of guessing the key, since JS algorithm deals with randomization.*

Keywords : *JS algorithm, Encryption, Jumbling, Key, Salting, Security*

I. Introduction

"Security is a process of implementing measures and systems designed to securely protect and safeguard information" thereby preserving the value, confidentiality, integrity, availability, intended use and its ability to perform their allowed critical functions. We are developing JS technique to overcome the problem of securing an encrypted image providing additional security to stored image.

Jumbling and Salting are the two processes of Jumbling-Salting algorithm. In the jumbling process, the image undergoes "addition", "selection" and "reverse" sub-processes. Addition process decides number of character to be appended to the image pixel. Selection select random character from the character set based on the length provided in Addition block. In general, there are many numbers of character set in the server. Selection of characters from different character set is made random. The reverse process reverses the output of the selection process based on some pre-defined condition. By using any mathematical technique such as even, odd, divisibility etc. the condition can be implemented. In the salting part, random salt is added to the jumbled encrypted image. Timestamp value determines selection of salt and the value is determined when the user creates the account.

Randomized algorithms are particularly effective when the attacker who deliberately tries to perform a dictionary or brute-force attack. It is said that randomness is ubiquitous in cryptography. The processes involved in JS algorithm are randomized; hence we can achieve "Randomness in Security"

This paper is an effort to compare numerous classical to modern cryptography algorithms based on various performance parameters like time complexity and key sensitivity. Moreover, the performance of all these algorithms under statistical, differential and qualitative attacks is also analyzed.

II. Proposed System

2.1 JS Block Diagram

The block diagram of the process arrangement in Jumbling Salting algorithm is as shown below in Fig. 2.1 which highlights the transition of the plain image to a jumbled and salted form.

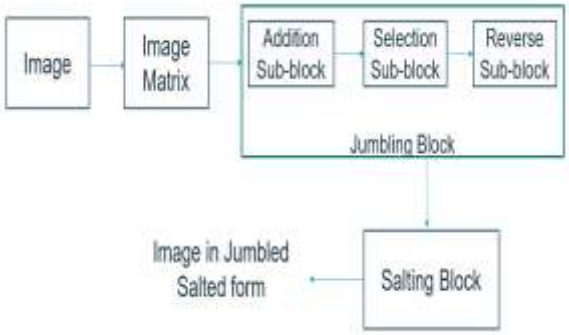

Fig. 2.1 Process arrangement in JS algorithm

2.1.1 Jumbling block :

Addition, selection, and reverse are the three sub-processes in the Jumbling process. Jumbling block is given the process array. A mathematical modulus function jumbles the characters that are prepended from some character set. Modulus is the mathematical function which returns reminder of a division operation. Jumbling block itself is a combination of three sub-blocks:

2.1.1.1 Addition sub-block:

This block generates principle random value "l". The size of a Process array is updated. (x + l) is the new update of P[ ] = x which is the original size of the process array.

The position of Process array P[] is illustrated as below in Fig. 2.2:

Fig 3.1. Process array in JS algorithm

2.1.1.2 Selection sub-block:

Selection sub-block selects characters from a predefined character set A. The size of character array is large and the character set for a particular password entry is different. Random values generated "l" times are the basis of selection of characters. The size of Process array is practically large enough to select the different characters.

The character set of general Process array is shown in Fig. 2.3 below:

Alphabets	A, B, ..., Y, Z a, b, ..., y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ' ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /

Fig 2.3. Character set of JS algorithm

2.1.1.3 Reverse sub-block:

Some predefined condition is the basis of Reverse sub-block to obtain the entire process array. Whether "l" is even or odd is the predefined condition to be checked. We reverse the process array if "l" has even values otherwise we keep it as it is.

For reversing the process array, we can use any mathematical condition, depends on the application. For example we can reverse process array if the value of "l" is prime, or "l" is an Armstrong number and so on

Reverse condition of JS algorithm is to create more confusion when the password file is accessed. Depending upon the application, the reverse condition can be altered. For example, if "l" is a prime number the entire reverse process is changed.

2.2 Salting block

Along with jumbled version of password, random string may be added as an objective of the salting block. The user's sign-up timestamp value is the criteria for salt selection. The password becomes more complicated when the salt is added. The attacker finds it difficult to obtain it. The timestamp value is chosen for adding salt in the jumbled version of password, as it is unique value. For every user, the timestamp value is different. We have chosen servers value as timestamp value.

The general form of salt array is shown in Fig. 2.4 below:

Y	Y	Y	Y	D	D	M	M	H	H	m	m	s	s

Where:
Y = Year, M= Month
D = Date
H = Hours in 24 hours format
m = minutes
s= seconds

III. Conclusion

JS algorithm reduces the probability of decrypting the image. Due to the various randomization technique, it builds a encrypted image which is almost difficult to decrypt. To decipher this encrypted image is a difficult task.

JS algorithm however takes a lot of space and time for both the encryption and decryption which is due to the various randomization technique but on the other hand this makes the algorithm fully secured.

Jumbling Salting algorithm's Encryption as well as the Decryption time is large. The additional overhead of jumbling and salting process increases due to the value of processing time. The encryption and decryption time rises due to the process of randomization.

Acknowledgements

We are using this golden opportunity to express our heartfelt gratitude to one and all who have supported us throughout this research. We are very much thankful for the aspiring guidance, invaluably constructive criticisms and friendly advice during the process of conducting research. We are also thankful for the support and encouragement of Principal Dr. Shrikant Kallurkar and the members of the management of Atharva College of Engineering, Malad West. We are highly indebted to Head of Department of Information Technology, Prof. Nileema Pathak and our guide Prof. Renuka Nagpure and co-guide Prof. Komal Gothwal for their guidance and constant supervision as well as for providing necessary information regarding the research & also for their support and co-operation in completing the project on research.

We are highly obliged to the laboratory technicians who have given their time, energy and attention for their invaluable support. Last but not least, our thanks are due to both the teaching and non-teaching staff of ACE, Malad (West).

References

[1]. Ch. Santhosh Reddy, Ch. Sowjanya, Praveena, Prof. Shalini L, "Poly-alphabetic Symmetric Key Algorithm Using Randomized Prime Numbers", IJSR, Volume 2, Issue 9, September 2012, ISSN 22503153.
[2]. C. Pfleeger, S. Pfleeger, Security in Computing, Third Edition, Prentice Hall PTR, ISBN: 0-13-035548-8.
[3]. Kahate Atul, 2003, "Cryptography and Network Security", Third Edition, Tata McGraw Hill India.
[4]. M. Stamp, Information Security : Principles an practice, Wiley publications , ISBN-13 978-0-471-73848-0
[5]. Subramania Sudharsanan, "Shared Key Encryption of JPEG Color Images", IEEE[J], 2005, 51(4), 1204-1211.
[6]. Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard, "Dr. Dobb's Journal, March 2001, pp. 137-139.
[7]. Priya Deshmukh, An image encryption and decryption using AES algorithm, *International Journal of Scientific and Engineering Research, 7(2)*, 2016, 2229-5518.
[8]. O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions", J. ACM, 33(4), 1986, 210-217.
[9]. "A Performance Comparison of Data Encryption Algorithms," IEEE Information and Communication Technologies, 2006-02-27, 84- 89.
[10]. Tingyuan Nie, Chuanwang Songa, Xulong Zhi, "Performance Evaluation of DES and Blowfish Algorithms", *Proc. IEEE Conf. on Biomedical Engineering and Computer Science*, Wuhan, China, 2010, 978-1-4244-5315-3.