INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

Enhancing the Security of Image Encryption Algorithms by Adding Timestamp

Lini Abraham[1], Neenu Daniel[2]

[1]M.Tech Student (CSE), Mahatma Gandhi University
Viswajyothi College of Engineering and Technology, Muvattupuzha, Kerala, India.
linirt33@gmail.com,

[2]Assistant Professor (CSE), Mahatma Gandhi University
Viswajyothi College of Engineering and Technology, Muvattupuzha, Kerala, India.
neenudaniel@gmail.com

Abstract *– We can ensure information security with various encryption techniques. Image encryption plays an important role in the field of protection of multimedia data like images. Most of the image encryption techniques have some security and performance issues. Chaos based encryption algorithms are employed nowadays because of their better security and performance aspects. This work is proposes a method for enhancing the security of image encryption algorithms by adding the effect of timestamp. The overview of this technique is given. It can be applicable on any image encryption algorithms with suitable modification based on that particular method. The main attraction of the proposed system is protection from replay attack and different cipher texts are produced even though with the same input image and key.*

Key Terms *– Cryptography, encryption, decryption, timestamp, replay attack, plaintext, cipher-text, chaos.*

1. INTRODUCTION

Multimedia information like digital image can be defined as a two dimensional rectangle array of pixels. Each pixel has some intensity value. So image has some special characteristics like high capacity, redundancy, and stronger correlation among pixels etc., compared to the normal data. For protecting these types of information from unauthorized access, various cryptographic techniques can be adopted. By using these techniques we can protect the information from different kinds of attacks by converting them into an unreadable form. The security of image can be achieved through various types of encryption techniques. One among the classification of image encryption technique is chaos based and non-chaos based algorithms. Due to the unpredictable behavior of chaotic system, it is considered to be more promising.

Images are widely used in different-different processes. Therefore, the security of image data from unauthorized uses is important. Image encryption plays an important role in the field of information hiding. Image hiding or encrypting method and algorithm can be very from simple methods to more complicated and reliable frequency method. Most of the encryption algorithms which are available mainly used for text data and cannot be suitable for multimedia data such as images. There are many image encryption schemes have been proposed, each one of them has its strength and weakness. Differential and cryptographic attacks are major concerns in data transmission. Even thorough some of the chaos based image encryption techniques resist these types of attacks to some extent; more protecting solutions are needed for further improvement. Replay attack is one of the major security outbreaks. It can be defined as a network attack in which a genuine data transmission is maliciously repeated or delayed. It can be done by the originator or by an advisory. The common measures for this type of attacks are session tokens, one time passwords, combination of nonce and MAC and timestamps. The content in [3] describes the replay attack. [4] and [5] proposes some methods to prevent such attacks. In [6], The proposed system is all about to prevent the replay attack on digital image transmission with the help of timestamps. The explanation of the technique is based on the grey scale images.

The method in [7] is chaos based using bit level permutation. Permutation at the bit level not only changes the position of the pixel but also alters its value. In [8] a novel image encryption method based on total shuffling scheme is illustrated. In [9] combinations of two logistic maps are used for improving the security of encryption. Encryption in [10] uses multiple chaotic systems. But each of these methods has some security issues. As the key space increases the security of the algorithm also get improved. Here this proposed scheme is applied on a chaos based secure image encryption algorithm based on Rubik's cube principle in [1]. It is an image encryption algorithm based on the principle of Rubik's cube. From [2] it is evident that this algorithm performs well with smaller time period. Here chaotic logistic map is used as the chaotic system for generating the sequence for encryption. The timestamp is also appended with the initial condition which is the shared secret key. So that the encrypted content will change even thorough we are using the same image and key on each time. It will make the brute force attack practically difficult.

The remaining of this paper is organized as follows. Chapter 2 describes the proposed system. In this the overview of the method is presented. Chapter 3 is description of the conclusions.

2. PROPOSED SYSTEM

Rubik's Cube's complexity is gives by the large number of permutations. It will use two keys so that the security can be improved. In addition to this its simplest implementation will make it more attractive. Here the time-stamp is appended with the original key. So the time-stamp is specially added to produce different cipher texts by applying same key on same plain text. Also it can be used to check the replay attack. The steps involved in the proposed scheme are explained below.

2.1 ASSUMPTIONS

Denning proposed a method for preventing replay attack with the help of timestamps. These methods are usually used for preventing replay attack in normal data transmission.

The main requirement for the application of this approach is the system synchronization. It is under the assumption that the two communication parties are already synchronized their time by some proper mechanisms. Also the encryption and decryption are performed just before sending and immediately receiving the image.

2.2 OVERVIEW OF THE PROPOSED SCHEME

In this new approach the time-stamp is appended with the original key. Then that modified key is given to the chaotic map. Then two random keys are generated and the Rubik's cube encryption is performed. Then KBRP (Key Based Random Permutation) is used to hide the time-stamp in the encrypted image. That will form the cipher image.

During decryption, first the time-stamp is extracted from the encrypted image by using the shared secret key. The difference between the extracted time and the current time is taken. If that difference is within the threshold then the decryption is performed otherwise it is rejected. For performing the decryption, the time- stamp is appended with the key and the two random numbers are generated. The decryption is then performed to get the original image. Each module in the proposed scheme is given below in figure 1.

2.3 ENCRYPTION PROCEDURE

Each step included in this process is explained below. From step 2 to step 4 the Rubik's cube based encryption in [1] is explained with a little modification.

Step 1: Adding the Time-stamp

The advantage of using time-stamp is that it can identify the replay attack. Also this is specially designed to produce different cipher texts by applying same key on same plain text. But a requirement is that the systems must be synchronized. Here for taking the current system time the command "clock" can be used in matlab. This time-stamp is composed of the year, month, day, and time. This is appended with the original key. So the size of the key is increased. Then each part in the time-stamp is represented in binary format. The time-stamp appended with the shared secret key forms the modified key that is used for further encryption.

Step 2: Generation of secret keys

The modified key is used to generate the actual keys for encryption. On each time the time-stamp is different, so the modified key will also be different. Here two random keys are generated by using logistic chaotic system. The size of the key is depends on the size of the image to be encrypted. For an $M \times N$ image, where M and N represents the number of rows and columns respectively, keys with size $1 \times M$ and $1 \times N$ are generated. The chaotic logistic map is given by the following equation (1): -

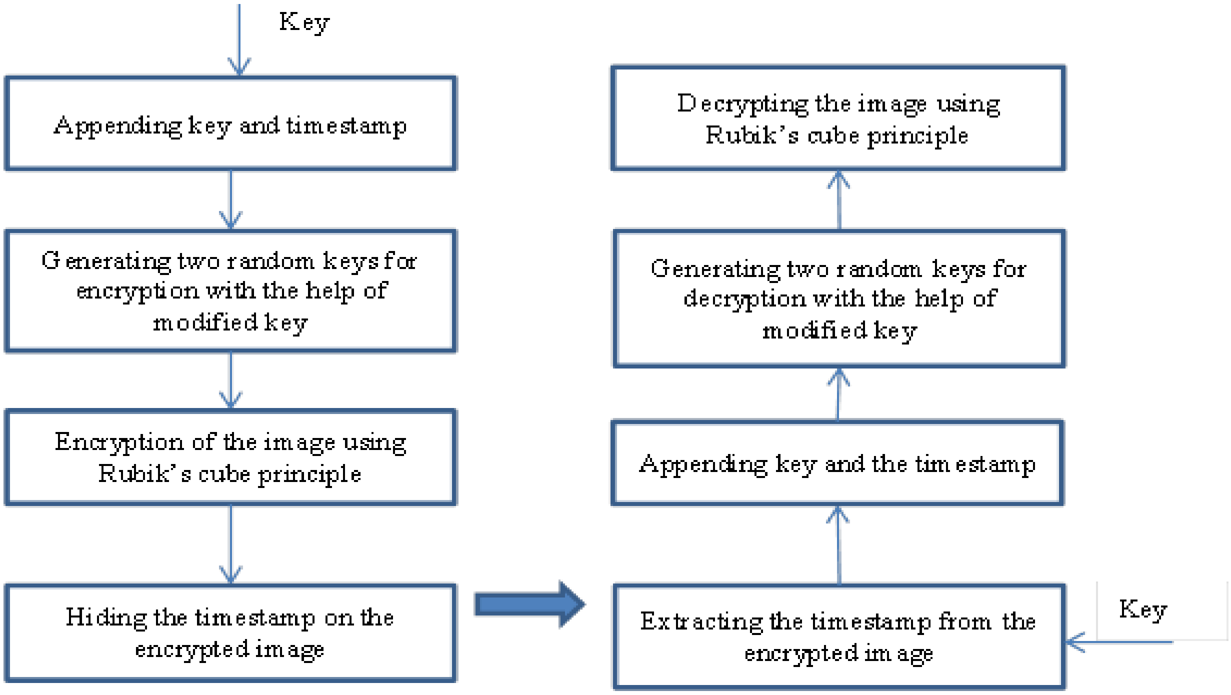
Key
Appending key and timestamp
Generating two random keys for encryption with the help of modified key
Encryption of the image using Rubik's cube principle
Hiding the timestamp on the encrypted image
Extracting the timestamp from the encrypted image
Key
Appending key and the timestamp
Generating two random keys for decryption with the help of modified key
Decrypting the image using Rubik's cube principle

Figure 1: Overview of the proposed system

$$x_{n+1} = rx_n(1 - x_n) \tag{1}$$

where x_n is the time index with x_0 as its initial value and r is a constant value. The logistic map is iterated repeatedly to get the desired key vector for encryption.

Step 3: Pixel scrambling process

Let I be the α-bit grey scale image to be encrypted with size $M \times N$ and K_r and K_c be two randomly generated vectors for encrypting the pixels of the original image I. Then the steps involved in pixel scrambling are given by: -

1. Determine the number of iterations, $ITER_{max}$ and initialize the counter ITER at 0.

2. Increment the counter by one.

3. For each row *i* of image I,

Compute the sum of all elements in the row *i*, one difference is that here only the first seven bits are used for calculating the sum in each pixel, which can be denoted as $\alpha(i)$. Compute $M_{\alpha(i)}$ by taking modulo 2 of $\alpha(i)$. If $M_{\alpha(i)}$ is 0 then the row *i* is right circular shifted by $K_r(i)$ positions, if it is 1 then the row *i* is left circular shifted by $K_r(i)$ positions.

4. For each column *j* of image I,

Compute the sum of seven bits in each element in the column j, which can be denoted as β(j). Compute Mβ(j) by taking modulo 2 of β(j). If Mβ(j) is 0 then the column j is up circular shifted by Kc(i) positions, if it is 1 then the column j is down circular shifted by Kc(i) positions.

The result of the above four steps will be the scrambled image; it can be denoted as ISCR.

Step 4: XOR operation

Now the bitwise XOR operator is applied to each row of the scrambled image using the key vector Kc by using the equation (5) in [1]. Then bitwise XOR operator is applied to each column of the image using the key vector Kr by the equation (6) in [1].

As the last step of encryption process, check whether the parameter ITER = ITERmax, the encrypted image IENC is obtained. Otherwise the above steps are repeated. For obtaining more security the technique can be applied to more than one round.

Step 5: Hiding the time-stamp in the encrypted image

After step 4 the cipher image is obtained. But for detecting the replay attack the receiver require the time-stamp used for encryption. Also for the purpose of decryption he wants the same. So the time-stamp needs to be embedded within the Red component of the encrypted image. For that LSB (Least Significant Bit) hiding is used. The time-stamp is composed of 72 bits. If they are hidden in consecutive locations then an attacker can easily retrieve the time and he can modify that. But here the hiding process is performed on random positions. For generating the random pixels for hiding a method called KBRP (Key Based Random Permutation) in [11] is used.

KBRP depends on using a specific key and size in order to cover the randomness and secrecy properties for permutation. This method introduces a method for generating a particular permutation P of a given size N out of N! variation from a given key. This method computes a unique permutation for a specific size since it takes the same key; therefore, the same permutation can be computed each time the same key and size are applied. The name of random permutation comes from the fact that the probability of getting this permutation is 1 out of N! possible permutations. The permutation cannot be guessed because of its generating method that is depending completely on a given key and size. Here the KBRP is used to generate 72 random numbers. And to those positions the LSB hiding is applied. The input key to this method will be the shared secret key.

There are three important functions in KBRP. They are given as: -

- Initialization (init)
- Eliminate
- Fill

The permutation is stored in one-dimensional array of size equal to the permutation size N. Each of the above functions is explained below.

First step, init(), is used to create an array of size n elements. The elements are generated from the key, by taking the ASCII code of each element in the key and storing them in the array consecutively. After adding the key value, the remaining values are computed by adding two consecutive values until the required number of elements in the array. As the final step, all values are set to the range 1 to N by applying the mode operation.

The second step, eliminate(), is to get rid of repeated values by changing them with a value zero and keep only one value out of these repeated values. Here the created array is scanned from the left and then from the right respectively for getting high randomness. (). Here array P contains N values. And there may be some repetitions, so the repeated values are observed and replaced with a value zero. Only a single value out of the repeated values is kept in the array P. Now P has only dissimilar elements in the range 1 to N and some zero values are also be seen in P. Missing elements are that are not occur in array P will also be substituted by the zero elements.

The last step, fill(), is to substitute all zero values with nonzero values in the range 1 to N which are not exist in the array. The array after the function fill will represents the random values into which positions the LSB hiding is applied. It is used to replace any zero value in P by a value in the range 1 to N which is not present in the array P. All zero values will be replaced through a sequence of one value from the left side of P and one value from the right side of P and repeating this sequence until all zero values are eliminated. Now the resulted array consists of all distinct values in the range 1 to N. This forms the random sequence. The first bit of the time-stamp is hiding at the position represented by the first random number in the generated series.

Step 6: LSB hiding

LSB hiding is one of the simplest hiding methods that embeds one bit in the least significant position of each pixel. If the bit to be hidden is 1 then the LSB of the pixel is set to 1. If the bit to be hidden is 0 then the LSB bit is changed to 0. Here each bit is hidden in randomly positioned pixels determined by the random numbers generated in the previous step. Here hiding is performed on the red component of the encrypted image. With this step the encryption process is completed.

2.4 DECRYPTION PROCEDURE

Decryption process is the reverse of the encryption steps with some differences. The decryption steps are briefly described below. It includes extraction of time-stamp, security checking, key generation, XOR operation and inverse scrambling process.

Step 1: Extraction of time-stamp

First the encrypted image is separated into red, green and blue components. The shared secret key and the same size used in the encryption can be used to get the random numbers using the KBRP method that have already described. And from those 72 positions the time-stamp can be retrieved.

Step 2: Checking the replay attack

The difference between the current time and the extracted time-stamp is calculated. If it is within some threshold then the decryption is performed, otherwise the image is rejected. By this way the replay attack can be prevented. But the requirement is that the systems should be synchronized.

Step 3: Key generation

The two keys based on the size of the image is generated as same as that of the key generation in encryption step after appending the time-stamp with the shared secret key. The two keys are represented as Kr and Kc. Initialize the value of ITERmax and set ITER = 0. Then increment the value of ITER by one.

Step 4: XOR operation

Bitwise XOR operator is applied to each column of the encrypted image IENC image using the key vector Kr. Then the bitwise XOR operator is applied to each row of the image using the key vector Kc. The ISCR represents the scrambled image. Now the reverse of the scrambling process is to be performed to get the original image.

Step 6: Inverse scrambling process

During encryption, the row scrambling was performed, followed by column scrambling. So during decryption the column scrambling should be performed first. The steps in this process are given below: -

1. For each column j of the scrambled image ISCR,

(a) Compute the sum of the first seven bits in each element in the column j, which can be denoted as βSCR(j).
(b) Compute Mβscr(j) by taking modulo 2 of βSCR(j).
(c) If Mβscr(j) is 0 then the column j is up circular shifted by Kc(i) positions, if it is 1 then the column j is down circular shifted by Kc(i) positions.

2. For each row i of image ISCR,

(a) Compute the sum of seven bits in each element in the row i, which can be denoted as αSCR(i).
(b) Compute Mαscr(i) by taking modulo 2 of αSCR(i).
(c) If Mαscr(i) is 0 then the row i is right circular shifted by Kr(i) positions, if it is 1 then the row i is left circular shifted by Kr(i) positions.

As the last step of decryption procedure check whether ITER = ITERmax. If so the decryption process is over and the decrypted image is obtained. Otherwise the above steps are to be repeated until the number of iterations is reached.

3. CONCLUSIONS

Chaos based encryption algorithms are employed nowadays because of their better security and performance aspects. Chaotic behavior of a system is the sophisticated nature of a nonlinear system that looks random. The time-stamp is appended with the original key. So the time-stamp is specially added to produce different cipher texts by applying same key on same plain text. Also it can be used to check the replay attack. But it is under the assumption that the system time is synchronized. So it is evident that the security is improved. Also the performance of the proposed scheme is comparable with the original scheme. The proposed system for identifying replay attack can be applicable on any image encryption algorithms with appropriate modification. New techniques and modifications can be added on to the proposed system for making excellent multimedia applications.

REFERENCES

[1] KhaledLoukhaoukha, Jean-Yves Chouinard, and AbdellahBerdai, “A Secure Image Encryption Algorithm Based on Rubik’s Cube Principle”, Department of Electrical and Computer Engineering, 2011.

[2] Lini Abraham, Neenu Daniel, “Secure image encryption algorithms: A review”, IJSTR, 2013.

[3] Li Gong and Paul Syverson. Fail-stop protocols: An approach to design- ing secure protocols. In 5th International Working Conference on De- pendable Computing for Critical Applicaitons, pages 44–55, September 1995.

[4] Sreekanth Malladi, Jim Alves-Foss, Robert B. Heckendorn, “On Preventing Replay Attacks on Security Protocols”.

[5] T. Aura. Strategies against replay attacks. In Proceedings of the 10th IEEE Computer Society Foundations Workshop, pages 59 – 68, Rockport, MA, June 1997. IEEE Computer Society Press.

[6] D. Denning and G. Sacco. Timestamps in key distribution protocols. Communications of the ACM, 24(8):553–536, August 1981.

[7] Zhi-liang Zhu, Wei Zhang, Kwok-wo Wong, Hai Yu, “A chaos-based symmetric image encryption scheme using a bit-level permutation”, Information Sciences 181 1171–1186 Elsevier, 2010.

[8] G. Zhang and Q. Liu, “A novel image encryption method based on total shuffling scheme,” Optics Communications, vol. 284, no. 12, pp. 2775–2780, 2011.

[9] Ismail1, Mohammed Amin, Hossam Diab, “ A Digital Image Encryption Algorithm Based A Composition Of Two Chaotic Logistic Maps”, Proc. 27th IEEE Int’l Conf. Signal Processing., pp. 733-739,2011.

[10] H.Alsafasfeh, and, A.A.Arfoa, Image encryption based on the general approach for multiple chaotic system, Journal of Signal and Information Processing 2, 238-244, 2011.

[11] Shakir M. Hussain1 and Naim M. Ajlouni, “Key Based Random Permutation”, Journal of Computer Science 2 (5): 419-421, 2006.