

A Faster and Efficient Image Encryption Algorithm Using Color Object, Byte Confusion and Diffusion

Parameswaran Bose*

* Software Engineer and Researcher

Article Info

Article history:

Received Jun 12th, 2014

Revised Aug 20th, 2014

Accepted Aug 26th, 2014

Keyword:

Chaotic System

Chaotic Map

Image Encryption

Permutation Sequence

Pixel Permutation

ABSTRACT

There are lots of efficient algorithms proposed for image encryption by using chaotic maps in the literature. But time is the main constraint for encrypting and decrypting the images. Even a small image of 256*256 resolution is taking more time for encryption and decryption. Because the computation takes more time for generating permutation sequence from the chaotic sequence elements by using linear search. In this paper we propose a new, faster and efficient algorithm for color images to generate the permutation sequence from the chaotic sequence elements by using the binary search method. The proposed algorithm produces a cipher of the test images that has good confusion and diffusion properties and was verified to provide a high security level. The results of several experiments show that the proposed algorithm for image cryptosystems provides an efficient and secure approach to real-time image encryption and transmission.

Copyright © 2014 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Parameswaran Bose,
#15, Anna Poorna House,
12th Cross, Indira Gandhi Street,
Udaya Nagar, Bangalore – 500 016, Karnataka, India.
Email: parameswaran.gri@gmail.com

1. INTRODUCTION

The rapid growth of internet allows us to transmit large files such as image, video and audio files. Since the internet is open to everyone, security has been the major issue to protect the data from the unauthorized access. Discrete chaotic dynamical systems are nonlinear dynamic behaviour, they are pseudorandom, sensitivity to the initial conditions and generate highly complicated signals by a simple recursive procedure. Chaotic systems are widely used in communications, optimization, control and image processing. To predict the behaviour of such systems, one should have sufficient knowledge about the initial conditions. Some of the important properties of the chaotic systems are as follows. Chaotic maps are deterministic, so they are characterised by mathematical equations. They are unpredictable, non-linear and highly sensitive to initial conditions. Even a slight change in the starting point can lead to significant different outcomes. Chaos based systems are useful to encrypt and decrypt all the image types since its approach is carried out by the confusion(shuffling) and diffusion with respect to the pixel position.

Chaos theory is not new to computer science and has been used for many years in cryptography. One type of encryption, secret key or symmetric key, relies on diffusion and confusion, which is modelled well by chaos theory [1]. Another type of computing, DNA computing, when paired with chaos theory, offers a more efficient way to encrypt images and other information [2]. Robotics is another area that has recently benefited from chaos theory. Instead of robots acting in a trial-and-error type of refinement to interact with their environment, chaos theory has been used to build a predictive model [3].

Chaos is a definitive and similar random procedure which appears in a nonlinear system [4, 5]. Chaos has been widely used in physics [6], biology [7], medical technology [8], synchronous control [9],

complex network [10], electrical engineering [11], and so on. Since the chaos-based encryption algorithm was first proposed in 1989 [12], many cryptographic protocols have emerged in the scientific literature [13, 14]. Chaos has many important properties, such as aperiodicity, topological transitivity, sensitive dependence on initial conditions, random-like behaviours, etc. [15, 16]; so, chaos based encryption algorithms have proved to be superior for encrypting images [17]. Traditional image encryption algorithms have 2 phases: confusion and diffusion. For example Gao and Chen [18] use matrix transformation and chaotic maps for shuffling and diffusion, respectively. Some only have one phase. For example, Wang et al.[19] proposed an image encryption algorithm which combines the shuffling and diffusion stage into one stage. But the algorithm in Ref.[20] states that the processing should be more than 3 rounds.

In this paper we proposed a new permutation sequence based color object transformation algorithm which is faster and efficient in encrypting and decrypting images by confusion and diffusion. We are experimenting the time complexity for generating the permutation sequence by our binary search method with several pictures of various resolutions. Finally we encrypt and decrypt various images with different resolutions by using the permutation sequence. And also the security measures are calculated with the original, encrypted and the decrypted images.

2. CHAOTIC SYSTEM

A chaotic dynamical system is an unpredictable deterministic and uncorrelated system the exhibits noise like behaviour through its sensitive dependence on its initial conditions, which generates sequences similar to pseudorandom sequence. The chaotic dynamics have been successfully employed to various engineering applications such as automatic control, signals processing and watermarking. Since the signals generated from chaotic dynamic systems are noise like super sensitive to initial conditions and have spread flat spectrum in the frequency domain, it is advantageous to carry messages with this kind of signal that is wide band and has high communication security. For this reason numerous engineering applications of secure communication with chaos have been developed.

2.1. Chaotic Sequences

A chaotic sequence is non converging and non periodic sequence that exhibits noise like behaviour through its sensitive dependence on its initial condition. A large number of uncorrelated, random like, yet deterministic and reproducible signals can be generated by changing initial value. These sequences so generated by chaotic systems are called chaotic sequences. Chaotic sequences are real valued sequences. This real valued sequence can be converted into integer valued sequence. This generated sequence makes it effective for pixel permutation and diffusion that can be used for image encryption and decryption.

2.2. Generation of Chaotic Sequences

The simplest and the most widely studied nonlinear dynamic systems capable of exhibiting chaos is the logistic map. The logistic map is defined as follows:

$$x_{n+1} = \mu x_n (1 - x_n), x_n \in (0, 1), \mu \in [0, 4] \quad (1)$$

Where x_n is an independent variable; μ is the control parameter of logistic map; $n = 1, 2, 3, \dots$. When $3.5699456 < \mu \leq 4$ and $x_n \in (0, 1)$, this logistic map is chaotic and can be used to generate a sequence up to length n .

2.3. Generation of Permutation Sequence

Chaotic dynamical systems are nonlinear dynamic behaviour, they are pseudorandom, sensitivity to the initial conditions. So they have used widely to generate the sequence of real number, By using those real sequence number and their sorted value a pseudorandom sequence can be generated. This pseudorandom sequence can be used for image pixel permutation. In this section we are generating the pseudorandom sequence in two ways; one is with the Linear Search method which is followed by many of the researchers in the literature and another method with the Binary Search which is proposed by us. Finally we are comparing the time complexity to generate the pseudorandom sequence by both the methods.

2.3.1. Generation of Permutation Sequence by Linear Search

Step 1. Generate the real chaotic sequence of length n by using the Equation 1, and store it in an one dimensional array as $\{a_1, a_2, a_3, \dots, a_n\}$. Since this is not sorted in nature, this array will be called as unsorted real chaotic sequence array.

Step 2. Take a clone of this unsorted real chaotic sequence array and sort it in ascending order. Store the sorted real sequence values in an one dimensional array as $\{b_1, b_2, b_3, \dots, b_n\}$. Since this is sorted in nature, this array will be called as sorted real chaotic sequence array.

Step 3. Iterate through all the sequence elements one by one in the sorted real chaotic sequence array and compare each element with the unsorted real chaotic sequence array. If the sequence element value is equal find the index position of the sequence element in the unsorted real chaotic sequence array and store it in an one dimensional array as $\{c_1, c_2, c_3, \dots, c_n\}$.

In detail the code which is used to get the permutation sequence by using linear search is as follows,

```
public int[] getChaosSequenceArrayIndex(Double[] sequenceArray, Double[] sortedSequenceArray){
    int indexArray[] = new int[sortedSequenceArray.length];
    for(int i=0; i<sortedSequenceArray.length; i++){
        Double number = sortedSequenceArray[i];
        for(int j=0; j<sequenceArray.length; j++){
            if(number == sequenceArray[j]){
                indexArray[i] = j;
                break;
            }
        }
    }
    return indexArray;
}
```

The resultant integer array is called as permutation sequence and mainly used for pixel permutation. Since the linear search is iterating through two for loops, the time taken to generate the permutation sequence is too high. And obviously it affects the time of encryption and decryption of the image. If the size of the image is too large automatically the iteration size of the two for loops will be increased. And automatically the time taken to generate the permutation sequence will be increased. So finally the time taken for encryption and decryption of the image will be too high.

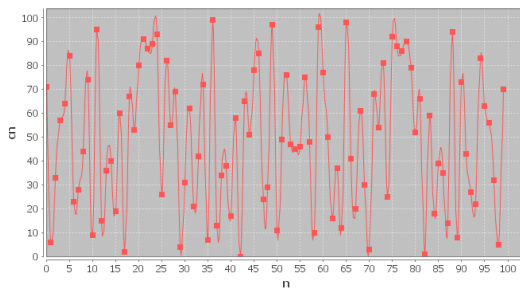


Figure 1. c_n for $\mu = 3.8000001$

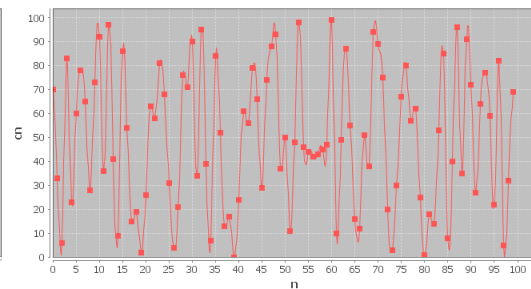


Figure 2. c_n for $\mu = 3.8000002$

The generation of the permutation sequence is the vital part of our proposed algorithm. Figure 1 shows the permutation sequence for the length $n = 100$ and the initial parameters $\mu = 3.8000001$ and $x_0 = 0.8$. The integer sequence is varied a lot in its value by changing the initial parameters value. The major advantage of this permutation sequence is, it is random in nature and defined in a specific range. If the initial seed value is changed then it drastically changes the sequence. Figure 2 shows the permutation sequence for the length $n = 100$ and the initial parameters $\mu = 3.8000002$ and $x_0 = 0.8$.

2.3.2. Generation of Permutation Sequence by Binary Search

Step 1. Generate the real chaotic sequence of length n by using the Equation 1, and store it in an one dimensional array as $\{a_1, a_2, a_3, \dots, a_n\}$. Since this is not sorted in nature, this array will be called as unsorted real chaotic sequence array.

Step 2. Take a clone of this unsorted real chaotic sequence array and sort it in ascending order. Store the sorted real sequence values in an one dimensional array as $\{b_1, b_2, b_3, \dots, b_n\}$. Since this is sorted in nature, this array will be called as sorted real chaotic sequence array.

Step 3. Iterate through all the sequence elements one by one in the unsorted real chaotic sequence array and find the index position of each element with the sorted real chaotic sequence array by binary search and store it in an one dimensional array as $\{c_1, c_2, c_3, \dots, c_n\}$. In detail the code which is used to get the permutation sequence by using binary search is as follows,

```
public int[] getChaosSequenceArrayIndex(Double[] sequenceArray, Double[] sortedSequenceArray){
```

```

int indexArray[] = new int[sortedSequenceArray.length];
for(int i=0; i<sequenceArray.length; i++){
    Double number = sequenceArray[i];
    indexArray[i] = Arrays.binarySearch(sortedSequenceArray, number);
}
return indexArray;
}

```

Since we are iterating through only one for loop and using the powerful mechanism binary search to find the index position of each element, the time taken to generate the permutation sequence is very less. And obviously it takes very less time to encrypt and decrypt the image. Even though the size of the image is too large, it won't take more time to generate the permutation sequence. So finally the time taken for the encryption and decryption of the image by this method will be very less when we are comparing the time with linear search.

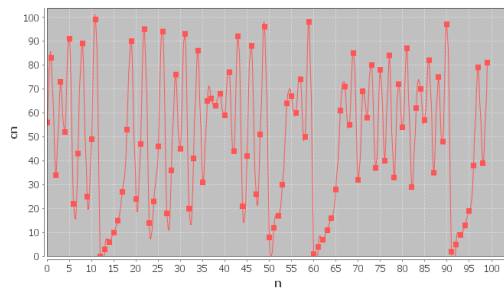


Figure 3. c_n for $\mu = 3.9999$

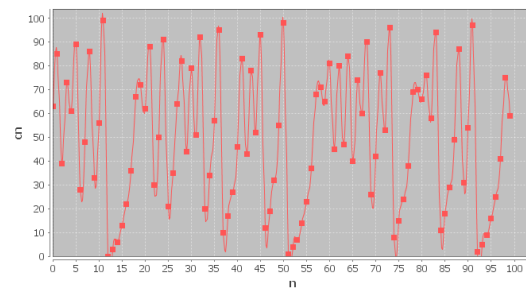


Figure 4. c_n for $\mu = 4.0$

Figure (3) shows the permutation sequence for the length $n = 100$ and the initial parameters $\mu = 3.9999$ and $x_0 = 0.8$. The integer sequence is varied a lot in its value by changing the initial parameters value. The major advantage of this permutation sequence is, it is random in nature and defined in a specific range. If the initial seed value is changed then it drastically changes the sequence. Figure (4) shows the permutation sequence for the length $n = 100$ and the initial parameters $\mu = 4.0$ and $x_0 = 0.8$.

2.3.3. Comparison of time to generate the permutation sequence

In this section we are investigating the time complexity to generate the permutation sequence by linear search and as well as by the binary search method with various images. The results of the time taken to generate the permutation sequence are presented in the Table 1. So from the investigation it clearly shows that the time taken to generate permutation sequence by binary search method is very less when we are comparing with linear search method.

Table 1. Time taken to generate the permutation sequence for various images

Image Name	Width*Height	No of Pixels	Linear Search(sec)	Binary Search(sec)
Meeting.png	256*256	65536	7.4	0.132
Peppers.png	512*512	262144	117.29	0.596
Lion.png	1024*1024	1048576	1894.009	3.246
Car.png	1920*1080	2073600	7272.65	6.855

3. PROPOSED ALGORITHM FOR ENCRYPTION AND DECRYPTION

Image encryption techniques try to convert an image to a pepper and salt image. Where image decryption retrieves the original image from the encrypted one. The algorithm starts with the generation of the chaotic sequence using the logistic map then the permutation sequence by using the chaotic sequence is calculated. This permutation sequence is mainly used to change the pixel position of the image to get the encrypted image. Also we can create the diffusion sequence by using the chaotic sequence. The diffusion sequence which is called by the name masking sequence is used to change the pixel's Red, Green, Blue color byte values. So permuting the pixel position and changing the color byte values of the pixel for many rounds will finally produce a salt and pepper image. This image is called as Encrypted Image. It is very safe to transmit the encrypted image through the public channel. And the parameters to decrypt the encrypted image can be transmit through various channels.

3.1. Encryption Algorithm

- Step 1. Load the image and get the width and height as M , N and calculate $n = M*N$.
- Step 2. Generate a permutation sequence co using μ_{co} and X_{co} .
- Step 3. Generate a permutation sequence pr using μ_{pr} and X_{pr} .
- Step 4. Generate a permutation sequence pg using μ_{pg} and X_{pg} .
- Step 5. Generate a permutation sequence pb using μ_{pb} and X_{pb} .
- Step 6. Generate a masking sequence mr using μ_{mr} and X_{mr} , where $mr_i = (|mr_i| / \max(|mr_i|)) * 255$.
- Step 7. Generate a masking sequence mg using μ_{mg} and X_{mg} , where $mg_i = (|mg_i| / \max(|mg_i|)) * 255$.
- Step 8. Generate a masking sequence mb using μ_{mb} and X_{mb} , where $mb_i = (|mb_i| / \max(|mb_i|)) * 255$.
- Step 9. Read the color objects (pixels) from image and put it in to a one dimensional array p .
- Step 10. Rearranging the elements in the color objects array p by using the permutation sequence co .
- Step 11. Split the Red, Green, Blue color bytes from the resultant color objects array from step 10, and store them in to 3 one dimensional arrays r , g , b .
- Step 12. Execute a Bitwise XOR operation between the elements of r with mr .
- Step 13. Execute a Bitwise XOR operation between the elements of g with mg .
- Step 14. Execute a Bitwise XOR operation between the elements of b with mb .
- Step 15. Rearranging the elements in the array r from step 12 by using the permutation sequence pr .
- Step 16. Rearranging the elements in the array g from step 13 by using the permutation sequence pg .
- Step 17. Rearranging the elements in the array b from step 14 by using the permutation sequence pb .
- Step 18. Create the color objects by r , g , b from the step 15, 16, 17 and assign it to p .
- Step 19. Repeat the steps 10 to 18 for R rounds.
- Step 20. Create the encrypted image with the resultant color bytes array.

3.2. Decryption Algorithm

- Step 1. Load the Encrypted Image and get the width and height as M , N and calculate $n = M*N$.
- Step 2. Generate a permutation sequence co using μ_{co} and X_{co} .
- Step 3. Generate a permutation sequence pr using μ_{pr} and X_{pr} .
- Step 4. Generate a permutation sequence pg using μ_{pg} and X_{pg} .
- Step 5. Generate a permutation sequence pb using μ_{pb} and X_{pb} .
- Step 6. Generate a masking sequence mr using μ_{mr} and X_{mr} , where $mr_i = (|mr_i| / \max(|mr_i|)) * 255$.
- Step 7. Generate a masking sequence mg using μ_{mg} and X_{mg} , where $mg_i = (|mg_i| / \max(|mg_i|)) * 255$.
- Step 8. Generate a masking sequence mb using μ_{mb} and X_{mb} , where $mb_i = (|mb_i| / \max(|mb_i|)) * 255$.
- Step 9. Read the color objects (pixels) from Encrypted Image and put it in to a one dimensional array p .
- Step 10. Split the Red, Green, Blue color bytes from the color objects array p , and store them in to 3 one dimensional arrays r , g , b .
- Step 11. Rearranging the elements to its proper place in r by using the permutation sequence pr .
- Step 12. Rearranging the elements to its proper place in g by using the permutation sequence pg .
- Step 13. Rearranging the elements to its proper place in b by using the permutation sequence pb .
- Step 14. Execute a Bitwise XOR operation between the elements of r from step 11 with mr .
- Step 15. Execute a Bitwise XOR operation between the elements of g from step 12 with mg .
- Step 16. Execute a Bitwise XOR operation between the elements of b from step 13 with mb .
- Step 17. Create the color objects array dp by the arrays r , g , b from the step 14, 15, 16.
- Step 18. Rearranging the elements in the array dp to its proper place by using the permutation sequence co , and assign it to the color objects array p .
- Step 19. Repeat the steps 10 to 18 for R rounds.
- Step 20. Create the decrypted image with the resultant color bytes array.

4. TOOLS USED FOR EXPERIMENT

The tools what we are using for the experiment will satisfy the modern world's requirement. In this section we are listing the hardware and software tools that we are using for the experiment. The hardware and the software tools are the latest and the software tools are open source that are freely available in the software industry.

4.1. Hardware Used

A Hewllet-Packard Laptop with the 3rd generation Intel Core i5-3230M processor with a speed of 2.60 GHz, and the RAM capacity is 4GB. The machine is installed with the operating system 64 bit Microsoft Windows 8 Single Language Edition.

4.2. Java Language

Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. The version of the Java Development Kit used for the implementation of the proposed algorithm is jdk1.7.0_45.

4.3. JavaFX Image Ops API

JavaFX is a software platform for creating and delivering rich internet applications (RIAs) that can run across a wide variety of devices. JavaFX applications could run on any desktop that could run Java SE, on any browser that could run Java EE, or on any mobile phone that could run Java ME. Image Ops, an API that enables us to read and write raw pixels within our JavaFX applications. In our implementation we used JavaFX Image Ops API, Release 2.2.4 to read pixel from images, write pixels to images, and create snapshots.

4.4. IntelliJIDEA

IntelliJ IDEA is a Java Integrated Development Environment (IDE) by JetBrains, available as an Apache 2 Licensed community edition and commercial edition. It is often simply referred to as "IDEA" or "IntelliJ". In a report by Infoworld organization in 2010, IntelliJ got the highest test center score out of the 4 Top Java Programming Tools : Eclipse, IntelliJ IDEA, NetBeans, and Oracle JDeveloper. The first version of IntelliJ IDEA was released in January 2001, and at the time was one of the first available Java IDE with advanced code navigation and code refactoring capabilities integrated. Google is now developing Android Studio, a new open source Android Development IDE, based on the open source community edition of IntelliJ IDEA.

4.5. JFreeChart

JFreeChart is an open-source framework for the programming language Java, which allows the creation of a wide variety of both interactive and non-interactive charts. JFreeChart supports a number of various charts, including combined charts: X-Y charts (line, spline and scatter). Pie charts, Gantt charts, Bar charts are also possible. Various specific charts (wind chart, polar chart, bubbles of varying size, etc.). It is possible to place various markers and annotations on the plot. JFreeChart also works with GNU Classpath, a free software implementation of the standard class library for the Java programming language.

5. EXPERIMENTAL RESULT

Now we provide some experimental results to illustrate the performance of the proposed chaotic cryptosystem with various images with different resolutions. We select the initial and control parameters as, $\mu_{co} = 3.9867473654678$, $\mu_{mr} = 3.9867473667854$, $\mu_{mg} = 3.9877473654678$, $\mu_{mb} = 3.9867473654876$, $\mu_{pr} = 3.8867473667854$, $\mu_{pg} = 3.9867473544678$, $\mu_{pb} = 3.9867437654876$, $R = 100$, $x_{co} = 0.834567234$, $x_{mr} = 0.835567234$, $x_{mg} = 0.836567234$, $x_{mb} = 0.834567432$, $x_{pr} = 0.845567234$, $x_{pg} = 0.836657234$, $x_{pb} = 0.854367432$. According to the proposed algorithm the encrypted images are shown in the following figures.



Figure 5. Meeting.png

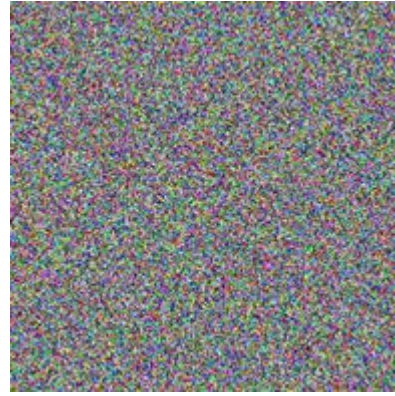


Figure 6. EncryptedMeeting.png



Figure 7. Peppers.png



Figure 8. EncryptedPeppers.png



Figure 9. Lion.png



Figure 10. EncryptedLion.png



Figure 11. Car.png



Figure 12. EncryptedCar.png

6. SECURITY ANALYSIS

In order to demonstrate that the proposed cryptosystem is secure against most common attacks, detailed security analyses of the proposed image encryption scheme are carried out, including key space analysis, statistical analysis and sensitivity analysis with respect to the key and plaintext.

6.1. Key Space Analysis

It was experimentally verified that for the sensitivity of the cryptosystem with respect to the control parameters is around 10^{-16} . On the other hand, the sensitivity with respect to the initial conditions was also experimentally measured as 10^{-15} . Having in mind that the encryption of every color component depends on each other and the selection of the keys of every color component is independent from the others. The key space is calculated by the control parameters $(\mu_{co}, (\mu_{mr}, \mu_{mg}, \mu_{mb}), (\mu_{pr}, \mu_{pg}, \mu_{pb}))$ and the initial parameters $(X_{co}, (X_{mr}, X_{mg}, X_{mb}), (X_{pr}, X_{pg}, X_{pb}))$ that we used in our experiment for encrypting and decrypting the image.

$K = (10^{16} * 10^{16} * 10^{16}) * (10^{15} * 10^{15} * 10^{15}) = 10^{93}$, this satisfies the security requirement related to the resistance against brute-force attacks. Once a cryptosystem has been designed, the next step is to test it in order to elucidate if it is really secure and efficient. This aim is very difficult to fulfil, since there is no standard framework to examine the quality of a cryptosystem. In other words, it is not possible to be totally sure about the invulnerability of our cryptosystem. Nevertheless, we have to be sure that the cryptosystem is secure against the best-known attacks. This is the goal of this section, along with the evaluation of the cryptosystem's performance.

6.2. The Number of Encryption Rounds

The number of encryption rounds R not only conditions the key sensitivity of the cryptosystem, but also its performance. In this sense, as R increases, the diffusion process is amplified and, in a similar manner, the encryption/decryption speed is decreased. In order to elude a timing-attack based on the relationship between the encryption/decryption speed and R , this parameter has been considered as a design parameter and not being part of the secret key. Its value has to be chosen carefully to fulfill a great level of security and a moderate value of the encryption/decryption time. To do so, a random image and the image resulting from the random modification of one of its pixels are encrypted using a random value for the key and different values for R . In this sense, the values of the Number of Pixels Change Rate or NPCR and the Unified Average Changing Intensity or UACI (see 6.3 section for the definition of both concepts) between the resulting cipher-images with respect to r has been measured, and confirming that a good level of NPCR and UACI is reached for round R .

6.3. Differential Attack Analysis

Generally speaking, an opponent may make a slight change (e.g., modify only one pixel) of the encrypted image to observe the change in the result. In this way, we may be able to find out a meaningful relationship between the plain image and the cipher image. This is known as the differential attack. However, if one minor change in the plain image can cause a significant change in the cipher image, with respect to diffusion and confusion, then the differential attack would become very inefficient and useless. The proposed cryptosystem can ensure two ciphered images different completely, even if there is only one bit difference between plain images. We have done differential analysis by calculating the NPCR (Net Pixel Change Rate) and UACI (Unified Average Changing Intensity) for several images. Here are the formulas:

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M * N} * 100\% \quad , \quad (2)$$

$$UACI = \frac{1}{M * N} \sum_{i,j} \frac{|c_1(i,j) - c_2(i,j)|}{255} * 100\% \quad (3)$$

Where $D(i, j)$ represents the difference between $c_1(i, j)$ and $c_2(i, j)$. If $c_1(i, j) = c_2(i, j)$ then $D(i, j) = 0$, otherwise $D(i, j) = 1$. For an 8-bit grey image, the expected estimates are $NPCR_E = 99.6094\%$ and $UACI_E = 33.4635\%$. We have done plaintext sensitivity analysis (differential analysis) by calculating the NPCR and UACI for plain-image Meeting, Peppers, Lion, and Car. The results of NPCR and UACI for the tested images are shown in the following figures respectively. It is clear that the NPCR and UACI values remain in the vicinity of the expected values. So, the proposed image encryption technique shows extreme sensitivity on the plaintext. Also, Table 2 shows the average values of NPCR and UACI for the plain-image Meeting, Peppers, Lion, and Car. We can find that the mean NPCR is over 99% and the mean UACI is over 33%. The results show that the proposed algorithm is very sensitive to tiny changes in the plain image; even

if there is only one bit difference between two plain images, the encrypted images will be different completely. Thus, the algorithm is robust against differential attack.

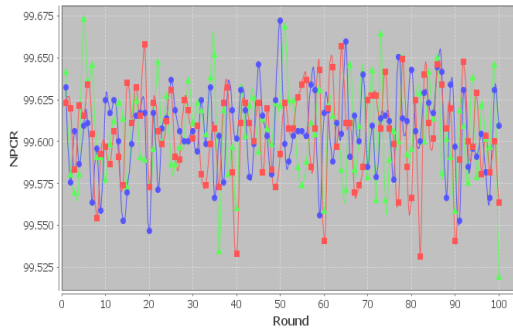


Figure 13. NPCR for 100 Rounds for Meeting.png

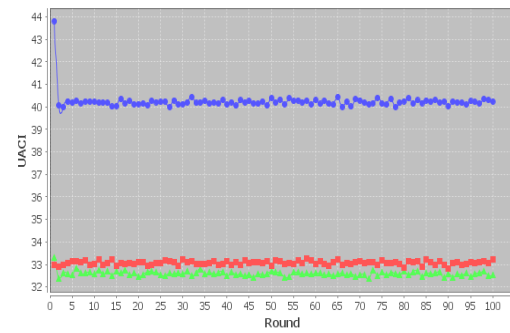


Figure 14. UACI for 100 Rounds for Meeting.png

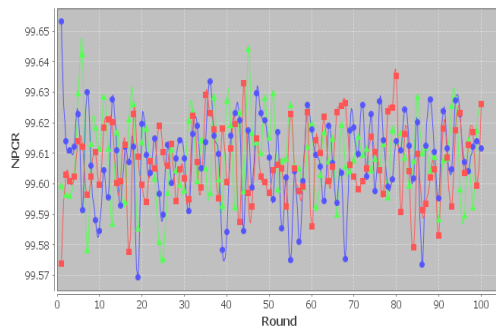


Figure 15. NPCR for 100 Rounds for Peppers.png

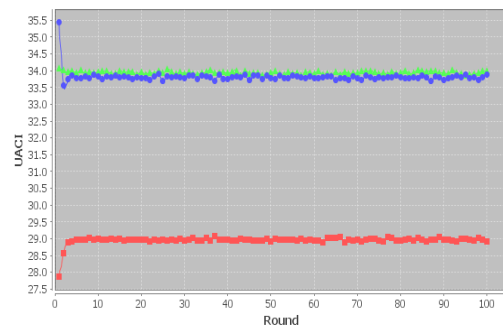


Figure 16. UACI for 100 Rounds for Peppers.png

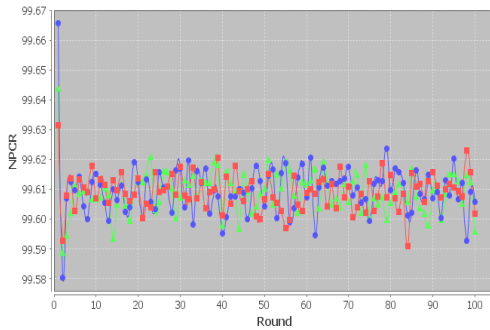


Figure 17. NPCR for 100 Rounds for Lion.png

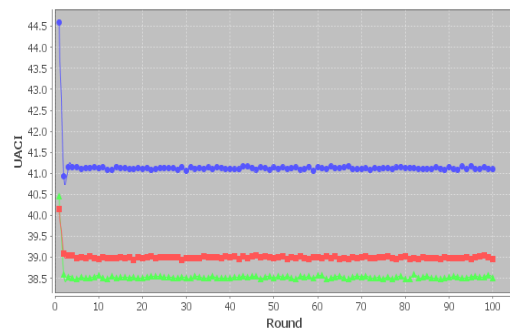


Figure 18. UACI for 100 Rounds for Lion.png

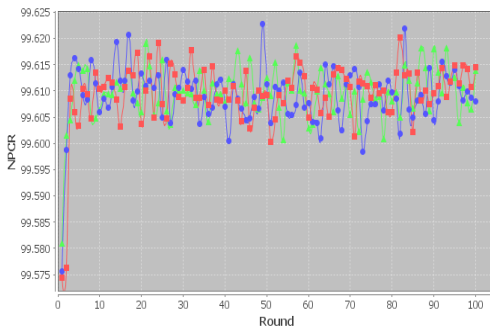


Figure 19. NPCR for 100 Rounds for Car.png

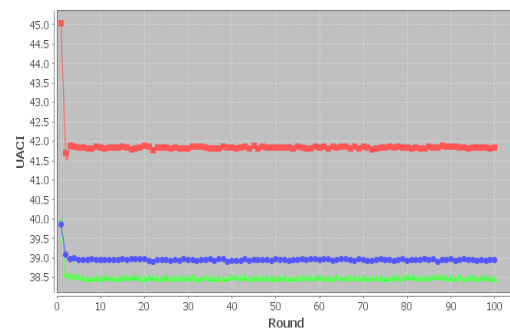


Figure 20. UACI for 100 Rounds for Car.png

Table 2. The NPCR and UACI of Encrypted images

Image Name	NPCR%			UACI%		
	Red	Green	Blue	Red	Green	Blue
Meeting.png	99.5636	99.51935	99.609375	33.242992	32.5194	40.246376
Peppers.png	99.62616	99.62616	99.611664	28.919258	33.993042	33.8869
Lion.png	99.60194	99.59593	99.60585	38.9613	38.511105	41.09514
Car.png	99.61454	99.61381	99.608025	41.842316	38.461365	38.937805

6.4. Correlation coefficient of two adjacent pixels

To test the correlation between two adjacent pixels in plain image and the cipher image, all pairs of two-adjacent pixels (in vertical, horizontal, and diagonal direction) from plain image and cipher image were selected and the correlation coefficients were calculated by using the following formulas:

$$E(x) = \frac{1}{L} \sum_{i=1}^L x_i, \quad D(x) = \frac{1}{L} \sum_{i=1}^L [x_i - E(x)]^2, \quad Conv(x, y) = \frac{1}{L} \sum_{i=1}^L [x_i - E(x)][y_i - E(y)]$$

$$\gamma_{xy} = \frac{Conv(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad (4)$$

Where x and y are the Red, Green, Blue values of two-adjacent pixels in the image and γ_{xy} is the correlation coefficient of two adjacent pixels. The test results are shown in Table 3, 4, 5, 6, 7, and 8. From the following tables it can be seen that the encryption scheme satisfies zero co correlation, which is of high-level security. Compared with the algorithms proposed in the literature, it shows superior performance.

Table 3. The Correlation Coefficient of two adjacent pixels (Vertical)

Image Name	Vertical		
	Red	Green	Blue
Meeting.png	0.95476997	0.93699350	0.9320784
Peppers.png	0.96450365	0.97740924	0.9609231
Lion.png	0.98069614	0.96738845	0.9464552
Car.png	0.97309950	0.97803110	0.9810139

Table 4. The Correlation Coefficient of two adjacent pixels (Horizontal)

Image Name	Horizontal		
	Red	Green	Blue
Meeting.png	0.9577724	0.93973200	0.93301260
Peppers.png	0.9584898	0.97797270	0.96468280
Lion.png	0.9735432	0.95384425	0.92447490
Car.png	0.9871292	0.99145055	0.99249136

Table 5. The Correlation Coefficient of two adjacent pixels (Diagonal)

Image Name	Diagonal		
	Red	Green	Blue
Meeting.png	0.92875385	0.9007650	0.89332783
Peppers.png	0.95073680	0.9623896	0.94063574
Lion.png	0.96461070	0.9350809	0.89814460
Car.png	0.96743520	0.9739547	0.97816220

Table 6. The Correlation Coefficient of two adjacent pixels (Vertical)

Image Name	Vertical		
	Red	Green	Blue
EncryptedMeeting.png	0.005568304	0.0030425778	0.0007291584
EncryptedPeppers.png	-0.0023794845	-0.0025698487	0.0034875246
EncryptedLion.png	-0.0017586248	0.0017701627	0.0012972517
EncryptedCar.png	-0.0012235832	-0.0003417274	0.00018416268

Table 7. The Correlation Coefficient of two adjacent pixels (Horizontal)

Image Name	Horizontal		
	Red	Green	Blue
EncryptedMeeting.png	0.00085462653	0.005135012	0.0059428816
EncryptedPeppers.png	0.007242696	0.004915961	0.003594824
EncryptedLion.png	0.0005232745	0.000529739	-0.000009191621
EncryptedCar.png	-0.0029038102	0.00012893182	0.00023931605

Table 8. The Correlation Coefficient of two adjacent pixels (Diagonal)

Image Name	Diagonal		
	Red	Green	Blue
EncryptedMeeting.png	-0.002212585	0.00694622	0.0027108332
EncryptedPeppers.png	-0.00270758	0.0003726096	-0.002960476
EncryptedLion.png	0.0007904043	0.0010226489	-0.0012095304
EncryptedCar.png	-0.0009900243	0.0011542992	0.0013351511

6.5. Key Sensitivity Analysis

Key sensitivity is an essential property for any good cryptosystem, which ensures the security of the cryptosystem against the brute-force attacks. The encrypted image produced by the cryptosystem should be sensitive to the secret key. That is to say, if the attacker uses two slightly different keys to decrypt the same plain image, the two encrypted images should be completely independent of each other. Recall that all the images are encrypted and decrypted with the initial parameters as given in the experiment result. Now we attempt to decrypt the encrypted images with different keys. Given that there is a change in the value of μ_{co} (eg., $\mu_{co} = 3.9867473654679$) which is slightly different from the encryption key, the resultant decrypted images are as follows. Obviously the decrypted images produced by using a slightly different key are completely different from the original one. It should be noted that other experimental results including individual changes in the value of $(\mu_{mr}, \mu_{mg}, \mu_{mb}), (\mu_{pr}, \mu_{pg}, \mu_{pb})$ and the initial parameters $X_{co}, (X_{mr}, X_{mg}, X_{mb}), (X_{pr}, X_{pg}, X_{pb})$ can be implemented in the same way.



Figure 21. DecryptedMeeting.png



Figure 22. DecryptedPeppers.png



Figure 23. DecryptedLion.png



Figure 24. DecryptedCar.png

7. CONCLUSION

This paper proposes a faster and efficient symmetric cryptographic system using logistic map to encrypt 24-bit color image. We can see that the proposed cryptosystem can process any size of image. Security analysis and experimental results demonstrated the effectiveness of the proposed scheme. The key space is large enough to resist brute-force attacks. Statistical analysis shows that the scheme can well protect the image from the statistical attack. The scheme possesses high sensitivity to plain image and key, so it has a good ability to resist differential attack. With high-level security, it can be used in secure image communications.


ACKNOWLEDGEMENTS

The Author would like to thank everyone who helped to make this research as a successful one.

REFERENCES

- [1] Wang, Xingyuan, Zhao and Jianfeng, "An improved key agreement protocol based on chaos", Communication Nonlinear Science Numerical Simulation, vol. 15, pp. 4052-4057, 2012.
- [2] Babaei, Majid, "A novel text and image encryption method based on chaos theory and DNA computing", Natural Computing, vol. 12, pp. 101-107, 2013.
- [3] Nehmzow, Ulrich and Keith Walker, "Quantitative description of robot-environment interaction using chaos theory", Robotics and Autonomous Systems, vol. 53, pp. 177-193, 2005.
- [4] Liu Y.J., Zheng Y.Q., "Adaptive robust fuzzy control for a class of uncertain chaotic systems", Nonlinear Dynamics, vol. 57, pp. 431-439, 2009.
- [5] Han Q., Liu C.X., Sun L. and Zhu D.R., "A fractional order hyperchaotic system derived from a Liu system and its circuit realization", Chinese Physics, vol. 22(2), 2013.
- [6] Weidenmuller H.A., Mitchell G.E., "Random matrices and chaos in nuclear physics", nuclear structure. Rev. Mod. Phys, vol.81(2), pp. 539-589, 2009.
- [7] Lesne. A, "Chaos in biology", Riv. Di Biol, vol. 99(3), pp. 467-481, 2006.
- [8] Kawashima. M., "Terminal care (1): chaos brought about by the progress of modern medical technology", Kangogaku Zasshi, vol. 49(10), pp. 1092-1095, 1985.
- [9] Zhu. D.R., Liu, C.X., Yan, B.N., "Control and synchronization of a hyperchaotic system based on passive control", Chinese Physics, vol. 21(9), 090509, 2012.
- [10] Lu, L., Luan, L., Meng, L., Li, C.R. "Study on spatiotemporal chaos tracking synchronization of a class of complex network", Nonlinear Dynamics, vol. 70(1), pp. 89-95, 2012.
- [11] Liu, C.X., Lu, J.J. "A novel fractional-order hyperchaotic system and its circuit realization", International Journal of Modern Physics, vol. 24(10), pp. 1299-1307, 2010.
- [12] Matthes, R., "On the derivation of a chaotic encryption algorithm", Cryptologia, vol. 13(1), pp. 29-42, 1989.
- [13] Huang, X.L., "Image encryption algorithm using chaotic chebyshev generator", Nonlinear Dynamics, vol. 67(4), pp. 2411-2417, 2012.
- [14] Zhang, L.Y., Li, C.Q., "Cryptanalyzing a chaos-based image encryption algorithm using alternate structure", Journal of System Software, vol. 85(9), pp. 2077-2085, 2012.
- [15] Wang, X.Y., Zhao, J.F., "A new image encryption algorithm based on chaos", Optics Commun, vol. 285(5), pp. 562-566, 2012.
- [16] Lian, S.G., "A block cipher based on chaotic neural networks", Neurocomputing, vol. 72(4-6), pp. 1296-1301, 2009.
- [17] Hussain, I., Shah, T., "Application of S-box and chaotic map for image encryption", Math. Comput. Model, vol. 57(9-10), pp. 2576-2579, 2013.
- [18] Gao, T.G., Chen, Z.Q., "Image encryption based on a new total shuffling algorithm", Chaos Solitons Fractals, vol. 38(1), pp. 213-220, 2008.
- [19] Wang, Y., Wong, K.W., Liao, X.F., Chen, G.R., "A new chaos-based fast image encryption algorithm", Applied Soft Computing, vol. 11(1), pp. 514-522, 2011
- [20] Gonzalo Alvarez and Shujun Li, "Some basic cryptographic requirements for chaos-based cryptosystems", International Journal of Bifurcation and Chaos, vol. 16, no. 8, pp. 2129-2151, 2006.

BIBLIOGRAPHY OF AUTHOR

	<p>Parameswaran Bose has received his M.Sc (Information Technology) degree from Gandhigram Deemed University, Gandhigram, India in 2002. Presently he is working as a software Engineer. He has ten years of working experience in Java, J2EE and has hands on experience in many frameworks. He worked with many Enterprise Applications and Web Applications which is used by the end users in the Healthcare (Drug Development) industry and by various Internet Service Providers. His research interests include Information Security, Digital File Encryption, Image Encryption, Chatoic Cryptology, Image Processing, Steganography, Data Compression, Elliptic Curve Cryptography, Mobile Security and Information Retrieval Systems. Currently he is working with the light weight cryptography which can be applied to android devices.</p>
---	---