Tribhuvan University

Institute of Science and Technology

Security Analysis of Rubik's Cube Algorithm for Image Encryption

Thesis

Submitted to

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

By

Apin Maharjan

T.U. Registration No.: 5-2-22-1652-2007

T.U. Examination Roll No.: 63/069

Date (April, 2019)

Supervisor

Mr. Bikash Balami

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than listed here have been used in this work.

____

Apin Maharjan

Date: April, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that this thesis prepared under my supervision by **Mr. Apin Maharjan** titled "**Security Analysis of Rubik's Cube Algorithm for Image Encryption**" in partial fulfillment of the requirements for the degree of M. Sc. In Computer Science and Information Technology be processed for the evaluation.

____

Asst. Prof. Bikash Balami

Central Department of Computer Science and Information Technology,

Kirtipur, Kathmandu, Nepal

Date: April, 2019

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that, we have read this thesis and in our opinion it is satisfactory in the scope and quality as a thesis in partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

____

Asst. Prof. Nawaraj Paudel

Head of Department
Central Department of Computer Science
& Information Technology
Tribhuvan University
Kirtipur

____

Asst. Prof. Bikash Balami

Central Department of Computer Science
and Information Technology (T.U)
(Supervisor)

____

(External Examiner)

____

(Internal Examiner)

ACKNOWLEDGEMENTS

First of all, I want to express my sincere gratitude to my respected teacher as well as my dissertation supervisor, Mr. Bikash Balami, Assistant Professor, Central Department of Computer Science and Information Technology (CDCSIT), Tribhuvan University for his wholehearted cooperation, encouragement and strong guidelines throughout the preparation of this work. With his enthusiastic presentation of new problems and ideas of possible solutions, he always managed to provide me with the necessary motivation. With this regard I wish to extend my genuine appreciation to respected Head of Central Department of Computer Science and Information Technology, Asst. Prof. Nawaraj Poudel for his kind help, encouragement and constructive suggestions.

I consider my pleasant duty to express my sincere gratitude to all the people who supported and encouraged me involving directly or indirectly to complete this thesis. I am also obliged to all respected teachers and staffs of CDCSIT for their willing co-operation to bring this thesis work in tangible form.

I have given my best effort to make this thesis work complete and error free. However, I am always looking forward to the suggestions from the readers which will improve this work.

Apin Maharjan

(CDCSIT, TU)

ABSTRACT

Among various approaches for image encryption, chaos based image encryption approaches are gaining popularity due to its resistance to reconstruction, reduction in degradation of quality digital images, simplicity of implementation and lower resource consumption.

In this study a chaos based approach for image encryption. "Rubik's cube algorithm" is analyzed before and after introducing a value transformation function. NPCR, UACI, Entropy and histogram analysis has been used for security analysis. In average the NPCR and Entropy value is found to be greater after applying the value transformation and also the UACI values tend to be closer to the ideal value of 33% upon applying the value transformation.

Keywords: *Image encryption, Rubik's cube algorithm, chaos based image encryption*

TABLE OF CONTENTS

ABBREVIATIONS

AES	Advanced Encryption Standard
DRFT	Discrete Random Fractional Transform
ECC	Elliptic Curve Cryptography
IDE	Integrated Development Environment
MSE	Mean Squared Error
NPCR	Number of Pixel Change Rate
UACI	Unified Average Change Intensity

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Information security is becoming more important in data storage and transmission as an increasing amount of information is being transmitted over the Internet, including not only text but also audio, image, and other multimedia files. Internet, being a public network, is not so secure for the transmission of confidential images [1]. Besides, due to high information capacity and high correlation among pixels also image encryption is the must. Hence, different techniques have been introduced such as encryption and digital watermarking. The first one consists of transforming multimedia documents using an algorithm to make it unreadable to anyone except for the legitimate users. The second one consists of embedding digital watermarks into multimedia documents to guarantee the ownership and integrity of the digital multimedia contents. Encryption is defined as the conversion of plain message into a form called a cipher text that cannot be read by any people without decrypting the encrypted text. Decryption is the reverse process of encryption which is the process of converting the encrypted text into its original plain text, so that it can be read. Various image-encryption algorithms like chaotic map, logistic map, advance encryption standard, Arnold map, affine transformation, Fourier transform and fractional Fourier transform are used to provide this security. These algorithms can be divided into two groups with respect to the approach used to construct the encryption scheme: chaos-based methods and non-chaos-based methods.

Although we can use the traditional encryption algorithms to encrypt images directly, this may not be a good idea for two reasons. First, the image size is often larger than text. Consequently, the traditional encryption algorithms need a longer time to directly encrypt the image data. Second, the decrypted text must be equal to the original text but this requirement is not necessary for image data. Due to the characteristic of human perception, a decrypted image containing small distortion is usually acceptable. [2]

Beside these, more sophisticated steganography techniques are also being used to hide large amounts of information within an image. Thus, it is often used in conjunction with cryptography so that the information is doubly protected, that is, first it is encrypted, and then it is hidden so that an adversary has to find the hidden information before the decryption takes place. [3]

Image encryption can also be divided into full encryption and partial or selective encryption schemes according to the percentage of the data that is encrypted. Encryption schemes can also be classified as either combined-compression methods or no compression methods. In traditional image and video content protection schemes, called fully layered, the whole content is first compressed then, the compressed bit-stream is entirely encrypted using a standard cipher (DES, AES, IDEA, etc.). Limitation of fully layered systems consists of altering the whole bit-stream syntax which may disable some codec functionalities. Selective encryption It consists of encrypting only a subset of the data. The aim of selective encryption is to reduce the amount of data to encrypt while preserving a sufficient level of security. This computation saving is very desirable especially in constrained communications (real-time networking, high-definition delivery, and mobile communications with limited computational power devices). [4]

1.2 Problem Statement

Many digital services like multimedia systems, medical and military imaging systems, and public internet communication require reliable security in storage and transmission of digital images. Due to growth of internet and multimedia technology in our society, the digital image security has become the most critical problem. It demands serious protection of users' privacy for all applications. Together with higher security level, maintaining quality of image data without losing the parametric properties of original image is equally important. In this context, building a secure image encryption framework with better efficiency, confidentiality and quality preference is of utmost research concentration. The traditional cryptosystems have been found not effective for multimedia data because of low scale performances and distorting quality of the data.

1.3 Objectives

The primary objective of this work are to improve the security of the existing Rubik's cube algorithm by introducing another layer of value transformation function on some randomly selected pixel values.

The outcome before and after value transformation on Rubik's cube algorithm is analyzed based on NPCR, UACI, Entropy and histogram analysis.

CHAPTER 2

LITERATURE REVIEW

2.1 Classic Image Encryption

Symmetric encryption is the oldest and best-known technique. In this system, both the sender and receiver share a single key. This method is also called Secret Key Cryptography because a single key is used for both encryption and decryption. A secret key, which can be a number, a word, or just a string of random letters, is applied to the original data to change the content in a particular way. As long as both sender and recipient know the secret key, they can encrypt and decrypt all data using this key.

Advanced Encryption Standard (AES) is a symmetric cryptosystem proposed for content encryption by Rijmen and Daemen in 1999, however it has been used for image encryption with a few changes in key generation and other requirements.

Zeghid et al. [5] proposed an improved AES based algorithm by including a key stream generator to AES to guarantee enhancement over the encryption execution for image encryption process.

An alternate algorithm proposed by Subramanyan et al. [6] focused around AES Key Expansion in which the encryption process is a bit wise exclusive or operation of a set of image pixels along with the 128-bit key which changes for every set of pixels. The keys to be used are generated independently at the sender and receiver side based on AES Key Expansion process hence the initial key is shared alone rather than sharing the whole set of keys.

Secret key cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same cipher text when using the same key in a block cipher whereas the same plaintext will encrypt to different cipher text in a stream cipher.

2.2 Public Key Image Encryption

One approach for public key image encryption is proposed by Shuihua et al. [7] in which the key pair is generated by some matrix transformation. And the image is encrypted by using private key in its transformation domain. And the receiver uses the public key for decryption.

An image encryption strategy utilizing ECC discrete logarithm problem is proposed by L. chen et al. [8] which is computationally less complex and suitable for large image encryption.

2.3 Chaos based Image Encryption

Chaos refers to a state that does not have deterministic behavior and Chaotic structures depend totally on initial condition. The basic principle of encryption with chaos is based on the ability of some dynamic systems to produce sequence of numbers that are random in nature. This sequence is used to encrypt messages. For decryption, the sequence of random numbers is highly dependent on the initial condition used for generating this sequence. A very minute deviation in the initial condition will result in a totally different sequence. This sensitivity to initial condition makes chaotic systems ideal for encryption. [1]

There are two general ways to apply a chaos map in a cipher system: Using chaotic systems to generate pseudo-random key streams and using the plaintext or secret keys as the preliminary conditions and control parameters then apply some iterations on chaotic systems to obtain cipher-text corresponding to the block ciphers. [9]

Min Li Ting Liang Yu-jie He [10] have presented a scrambling system based on Arnold transform that could work on non-square image by splitting the non-square image to multiple square regions, and scrambling each region.

Q. Guo, Z. Liu, and S. Liu [11] presented a system accommodating Arnold transform with discrete random fractional transform in intensity-hue-saturating space from RGB space. Where some component is encrypted using AT and some component is transformed using DRFT.

Changjiang Zhang et al. [12] proposed an algorithm for watermark inserting and detecting algorithm based on stationary wavelet transform. In this method firstly the digital watermarking was transformed randomly (Arnold transformation), then encrypted by the use of logistic map.

CHAPTER 3

METHODOLOGY

3. Methodology

3.1 Tools Used

The algorithms are implemented using Matlab R2016a IDE.

3.2 Data Collection

The data for this study are the various benchmark images for image encryption like Lena, Checkboard, black in different dimensions and file formats viz .jpeg, .jpg, .png, .bmp. Besides non-benchmark images are also used.

3.3 Rubik's Cube Image Encryption

3.3.1 Rubik's Cube Based Encryption Algorithm.

Let I_o represent an α-bit gray scale image of the size M*N. Here, I_o represent the pixels values matrix of image I_o. The steps of encryption algorithm are as follows:

(1) Generate randomly two vectors K_R and K_C of length M and N, respectively. Element $K_R(i)$ and $K_C(j)$ Each take a random value of the set A = {0, 1, 2, . . . , $2^{\alpha} - 1$}. Note that both K_R and K_C must not have constant values.

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

(3) Increment the counter by one: ITER = ITER + 1.

(4) For each row i of image I_o,

(a) compute the sum of all elements in the row i, this sum is denoted by α(i)

$$\alpha(\mathrm{i}) = \sum_{j-1}^{N} I_o(i,j), \quad i = 1,2, \dots \dots \dots ., M$$

(b) compute modulo 2 of α(i), denoted by $M_{\alpha(i)}$,

(c) row i is left, or right, circular-shifted by $K_R(i)$ positions (image pixels are moved $K_R(i)$ positions to the left or right direction, and the first pixel moves in last pixel.), according to the following:

if $M_{\alpha(i)} = 0 \rightarrow$ right circular shift

else $\rightarrow$ left circular shift.

(5) For each column j of image I_o,

(a) compute the sum of all elements in the column j, this sum is denoted by β(j),

$$\beta(j) = \sum_{i-1}^{M} \mathrm{Io}(i, j), i = 1,2, \ldots., M$$

(b) compute modulo 2 of β(j), denoted by $M_{\beta(j)}$.

(c) column j is down, or up, circular-shifted by $K_C(i)$ positions, according to the following:

if $M_{\beta(j)} = 0 \rightarrow$ up circular shift

else $\rightarrow$ down circular shift.

Steps 4 and 5 above will create a scrambled image, denoted by I_{SCR}.

(6) Using vector K_C, the bitwise XOR operator is applied to each row of scrambled image I_{SCR} using the following expressions:

$$I_1(2i-1, j) = I_{SCR}(2i-1, j) \oplus k_c(j),$$
$$I_1(2i, j) = I_{SCR}(2i, j) \oplus rot\ 180(k_c(j)),$$

where ⊕ and rot 180(K_C) represent the bitwise XOR operator and the flipping of vector K_C from left to right, respectively.

(7) Using vector K_R, the bitwise XOR operator is applied to each column of image I_1 using the following formulas:

$$I_{ENC}\,(i, 2j-1) = I_1(i, 2j-1) \oplus K_R(j),$$
$$I_{ENC}\,(i, 2j) = I_1(i, 2j) \oplus rot180\ (K_R(j)),$$

with rot 180(K_R) indicating the left to right flip of vector K_R.

(8) If ITER = $ITER_{max}$, then encrypted image I_{ENC} is created and encryption process is done;

otherwise, the algorithm branches to step 3.

Vectors K_R, K_C and the $ITER_{max}$ are considered as secret keys.

Following transformation has been added to the prevailing algorithm

- randomly select n pixels, $n \leq M*N$ and n pixel index values (say m), such that for each m, $1 \leq m \geq 8$.
- for each selected pixel and pixel index value, convert pixel value to binary and set all bit to 0 except pixel index value, which is set to 1.
- perform XOR of pixel value with random binary number generated in previous stage.

3.3.2 Rubik's Cube Decryption Algorithm.

The decrypted image, Io, is recovered from the encrypted image, I_{ENC}, and the secret keys, K_R, K_C, and $ITER_{max}$ as follows in the following.

(1) Initialize ITER = 0.

(2) Increment the counter by one: ITER = ITER + 1.

(3) The bitwise XOR operation is applied on vector K_R and each column of the encrypted image I_{ENC} as follows:

$$I_1\,(i, 2j-1) = I_{ENC}(i, 2j-1) \oplus K_R(j),$$
$$I_1\,(i, 2j) = I_{ENC}(i, 2j) \oplus rot180\,(K_R(j)),$$

(4) Then, using the KC vector, the bitwise XOR operator is applied to each row of image I_1

$$I_{SCR}(2i-1, j) = I_1(2i-1, j) \oplus k_c(j),$$
$$I_{SCR}(2i, j) = I_1(2i, j) \oplus rot\ 180(k_c(j)),$$

(5) For each column j of the scrambled image I_{SCR},

(a) compute the sum of all elements in that column j, denoted as $\beta_{SCR}(j)$:

$$\beta_{SCR}(j) = \sum_{i=1}^{M} I_{SCR}(i, j), \qquad j = 1,2, \ldots\ldots, N$$

compute modulo 2 of $\beta_{SCR}(j)$, denoted by $M_{\beta_{SCR}}(j)$.

(b) column j is down, or up, circular-shifted by $K_C(i)$ positions according to the following:

if $M_{\beta_{SCR}}(j) = 0 \rightarrow$ up circular shift

else $\rightarrow$ down circular shift.

(6) For each row i of scrambled image I_{SCR},

(a) compute the sum of all elements in row i, this sum is denoted by $\alpha_{SCR}(i)$:

$$\alpha_{SCR}(i) = \sum_{j=1}^{N} I_{SCR}(i,j), \qquad j = 1,2, \ldots\ldots, N$$

(b) compute modulo 2 of $\alpha_{SCR}(j)$, denoted by $M_{\alpha_{SCR}}(j)$

(c) row i is then left, or right, circular-shifted by $K_R(i)$ according to the following:

if $M_{\alpha_{SCR}}(j) = 0 \rightarrow$ right circular shift

else $\rightarrow$ left circular shift.

(7) If ITER = $ITER_{max}$, then image I_{ENC} is decrypted and the decryption process is done; otherwise, the algorithm branches back to step 2.

CHAPTER 4

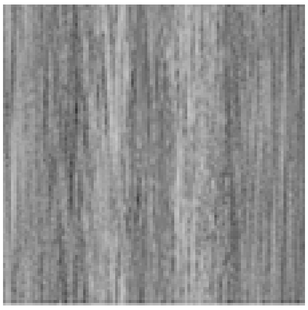
ANALYSIS

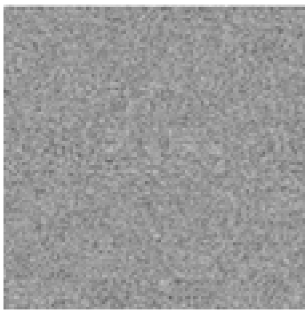
4.1 Analysis

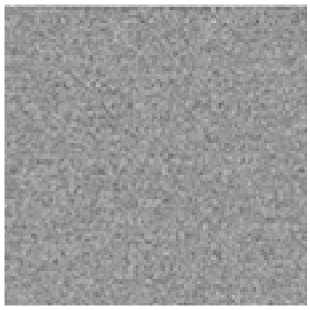
The encrypted images are analyzed based on the standard measures like entropy, histogram, NPCR and UACI. Study is also done on the key sensibility analysis.

Original Image	Scrambled Image	Encrypted Image Using Rubiks Cube Algorithm	Encrypted Image after value transformation on Rubiks cube algorithm

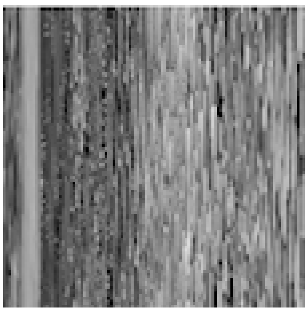
Baboon.jpg
512 x 512

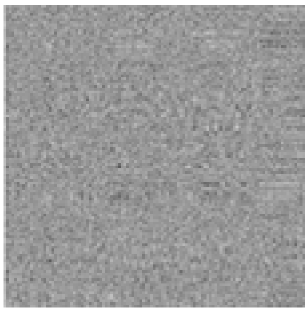


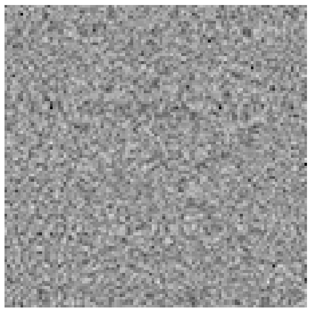




Lena.png
512 x 512







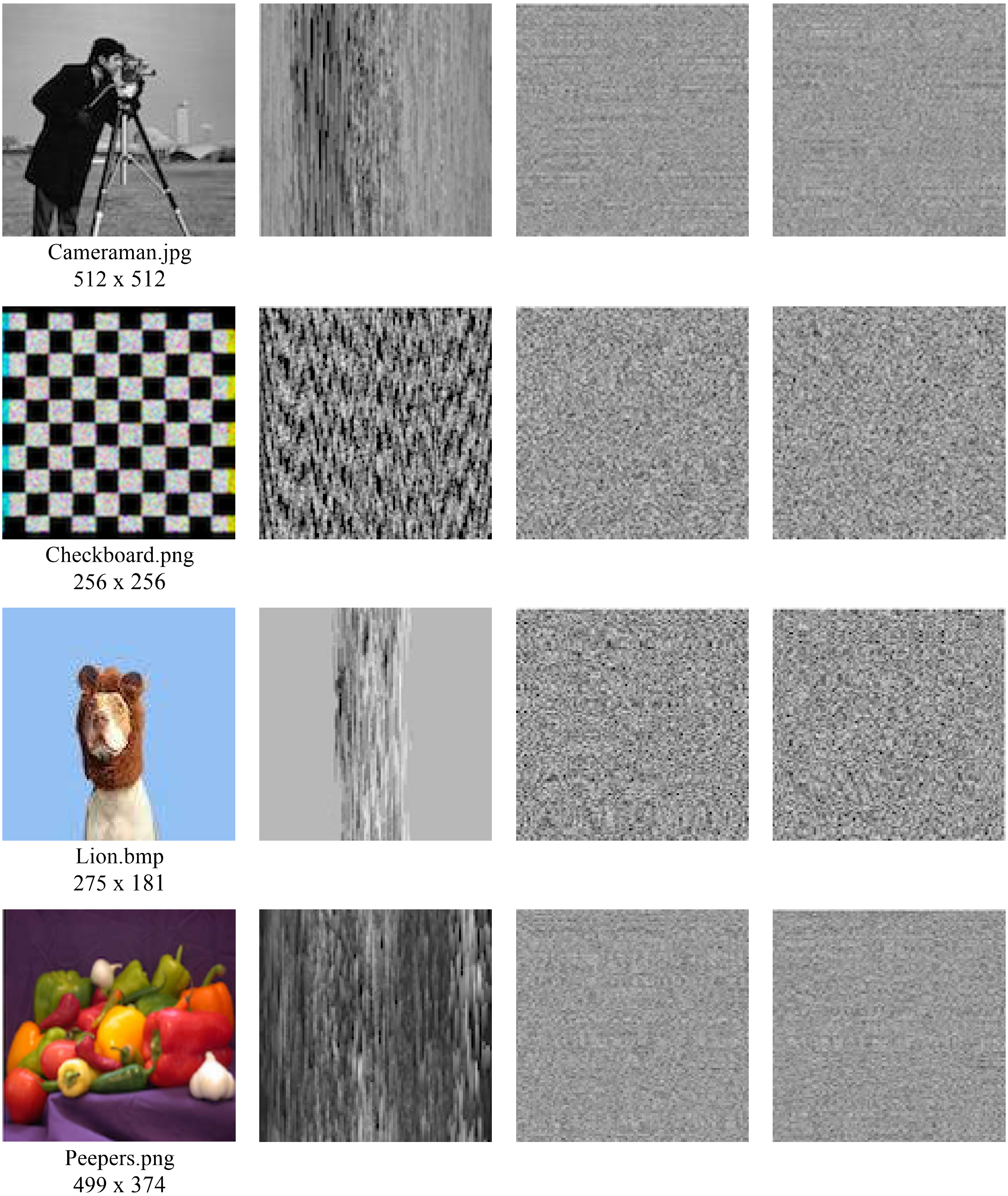


Fig 4.1 : original image, scrambled image and Comparison of encrypted images

4.1.1 Differential Analysis

The encrypted image should greatly differ from its original form. In general, two difference measures are used to quantify this requirement. The first measure is the number of pixels change rate (NPCR), which indicate the percentage of different pixels between two images. The second one is the unified average changing intensity (UACI), which measures the average intensity of differences in pixels between two images [13]. In image encryption, the cipher resistance to differential attacks is commonly analyzed via the NPCR and UACI tests. [14,15,16] To build a near ideal image encryption algorithm, NPCR values must be greater than 99% and UACI values must be around 33%. [16,17]

Let I_o (i , j) and I_{ENC} (i , j) be the pixels values of original and encrypted images, I_o and I_{ENC} , at the i^th^ pixel row and j^th^ pixel column, respectively. Then following Equations give the mathematical expressions of the NPCR and UACI measures:

$$NPCR = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} D(i,j)}{M * N} * 100\ \%$$

$$\text{with } D(i,j) = \begin{cases} 0 & if\ I_0(i,j) = I_{ENC}(i,j), \\ 1 & otherwise. \end{cases}$$

$$UACI = \left[\sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|I_0(i,j) - I_{ENC}(i,j)|}{255} \right] * \frac{100\%}{M * N}$$

Similarly, the entropy of a digital image is a statistical measure that expresses the randomness of gray levels and is defined as:

$$E = -\sum_{i=1}^{n} p_i log_2\, p_i$$

where n refers to number of possible gray scale levels(i) and p_i denotes probability of occurrence of gray level i.

Difference Measures between Encrypted image before and after applying value transformation function: values of NPCR, UACI and Entropy

Image	*Encrypted Image Using Rubik's Cube Algorithm*			*Encrypted Image after value transformation on Rubik's cube algorithm*		
	NPCR (%)	UACI (%)	Entropy	NPCR (%)	UACI (%)	Entropy
Baboon.jpg	99.593735	27.889401	7.999	99.628067	27.789742	7.999
Lena.png	99.711227	50.109717	7.998	99.720001	50.060263	7.999
Cameraman.jpg	99.620819	31.114451	7.999	99.610519	31.175560	7.999
Checkboard.png	99.609375	42.508341	7.997	99.665833	42.684338	7.997
Lion.bmp	99.757576	30.780936	7.996	99.777447	30.798804	7.996
Peepers.png	99.587410	31.747902	7.998	99.616345	31.927542	7.998

Table 4.1 : NPCR, UACI and Entropy values

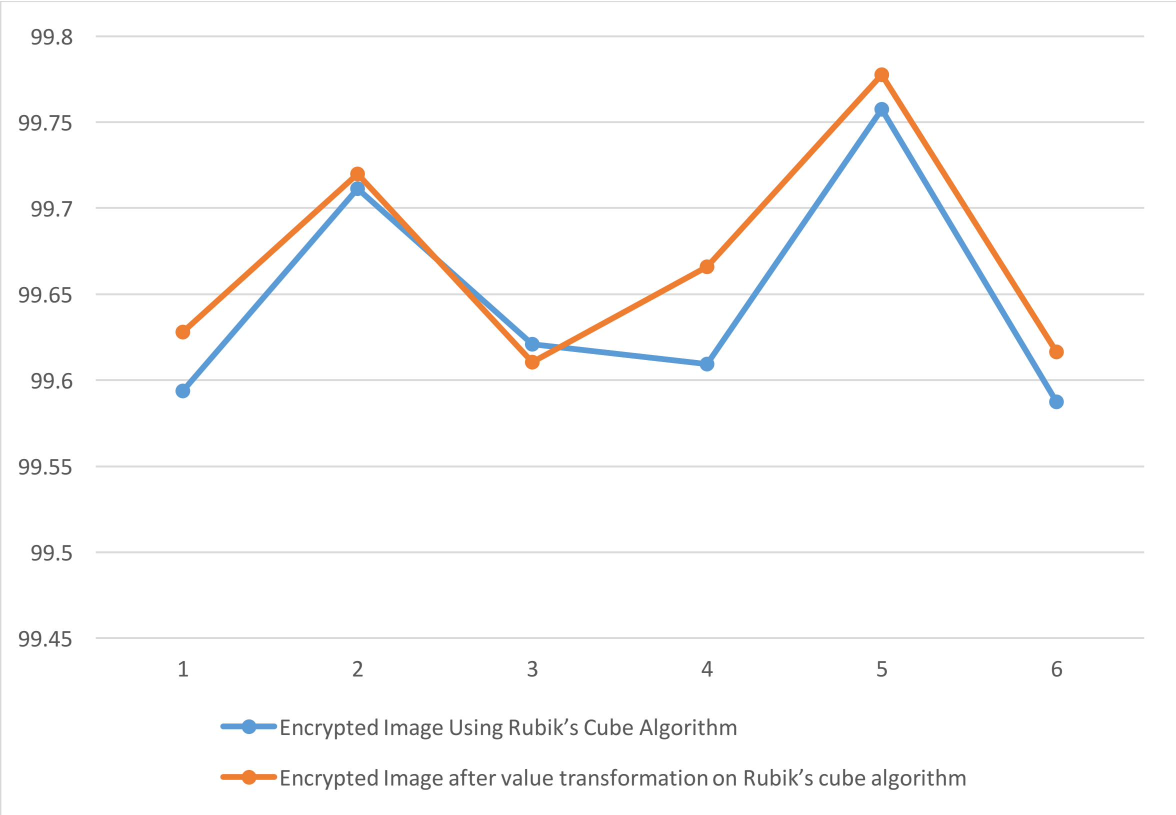


Fig 4.2 : Comparison of NPCR values

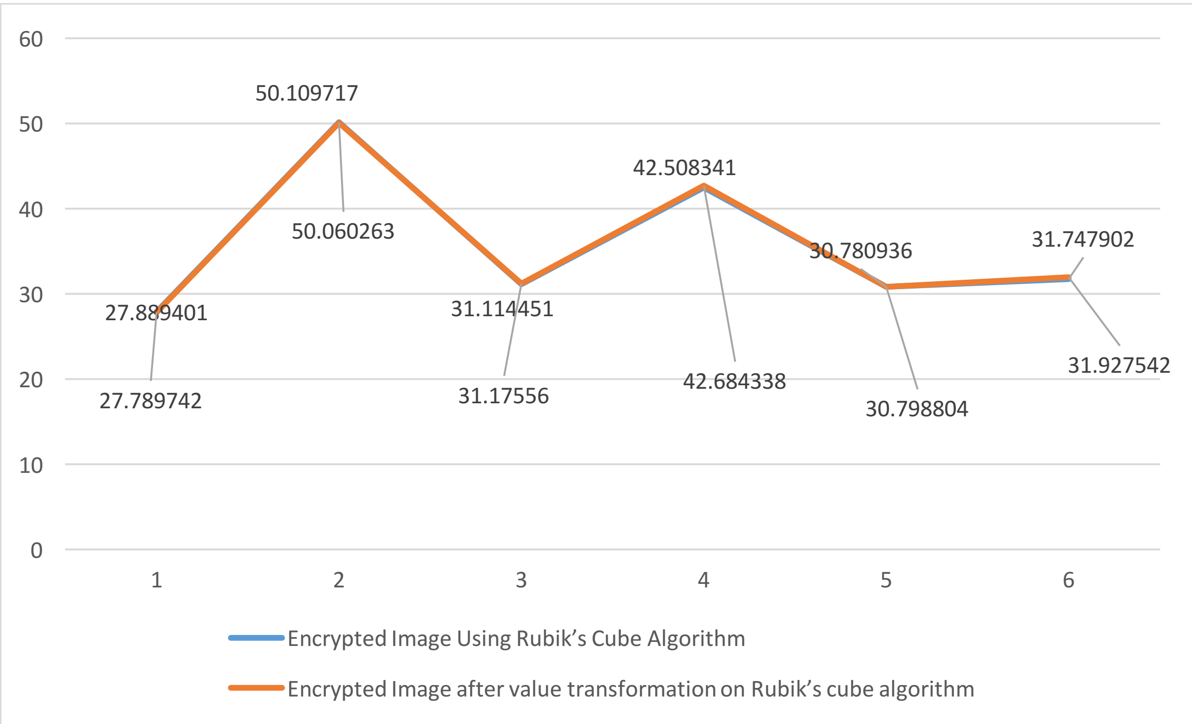


Fig 4.3 : Comparison of UACI values

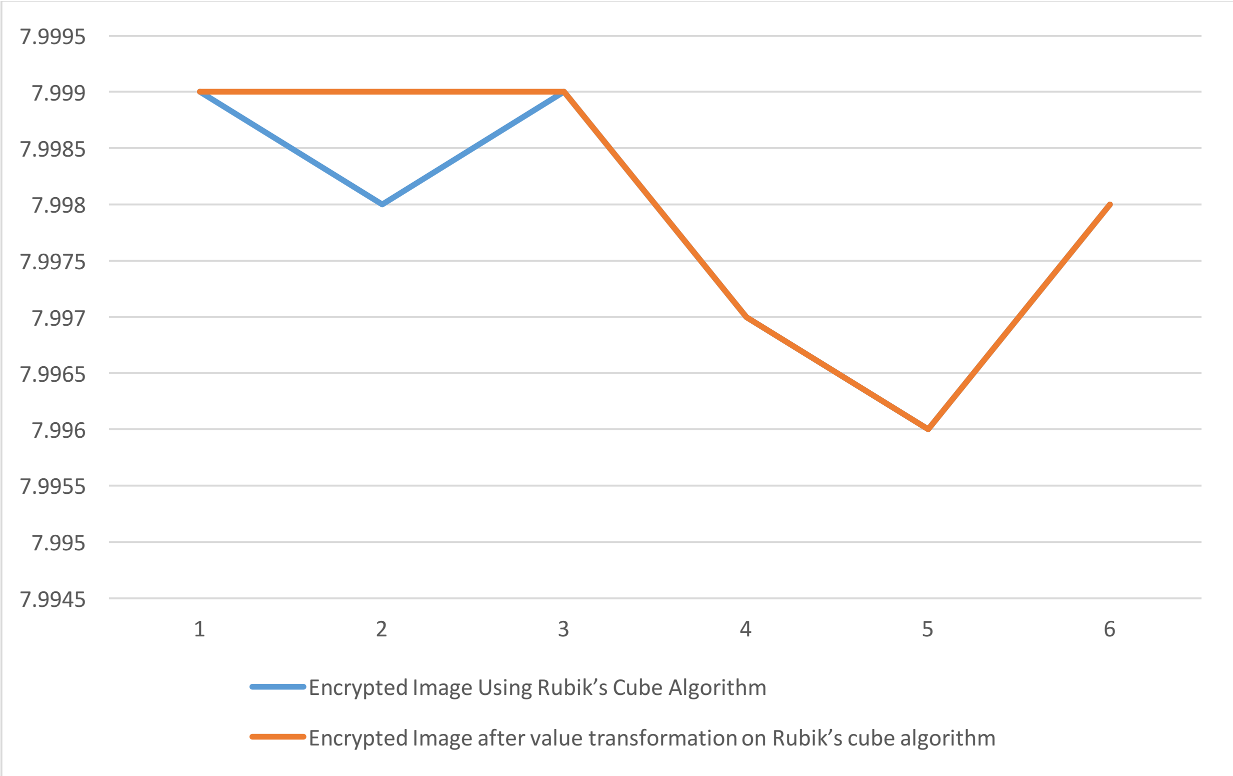


Fig 4.4 : Comparison of Entropy values

4.1.2 Statistical Analysis

Statistical analysis is carried out by analyzing the histogram, that reflects the pixel intensity values of an image. statistical analysis has been performed to demonstrate the superior confusion and diffusion properties of the algorithm against statistical attacks. “It is possible to solve many kinds of ciphers by statistical analysis.” [18]

Fig 4.6: represents the histogram of the original and the encrypted images; before and after applying value transformation for image checkboard.png. It is found that the histograms of the encrypted images are almost uniform and are significantly different from that of the original images. For instance, the histogram of original image Checkerboard shows as expected mostly values: 0 and 255; however, the histogram of the encrypted Checkerboard image is fairly uniform. Therefore, the image encryption algorithm responds well to the diffusion properties: it does not provide information that can be exploited for attacks based on statistical analysis of the encrypted image.

Baboon.jpg

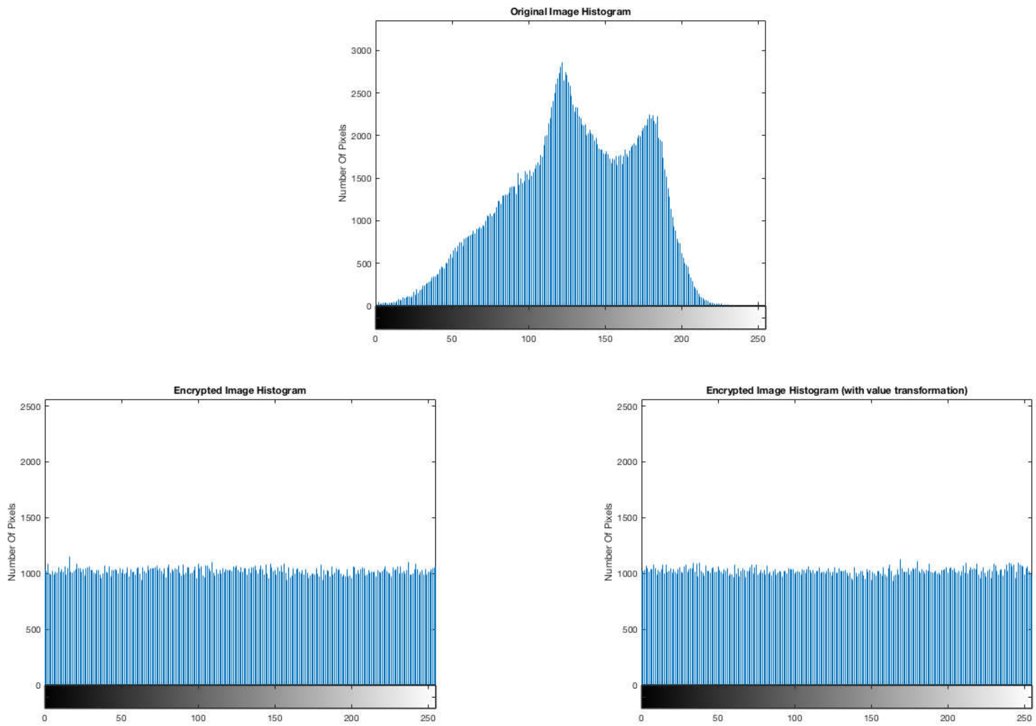


Fig 4.5 : Original Histogram vs histogram after encryption; before and after applying value transformation for Baboon.jpg

Checkboard.png

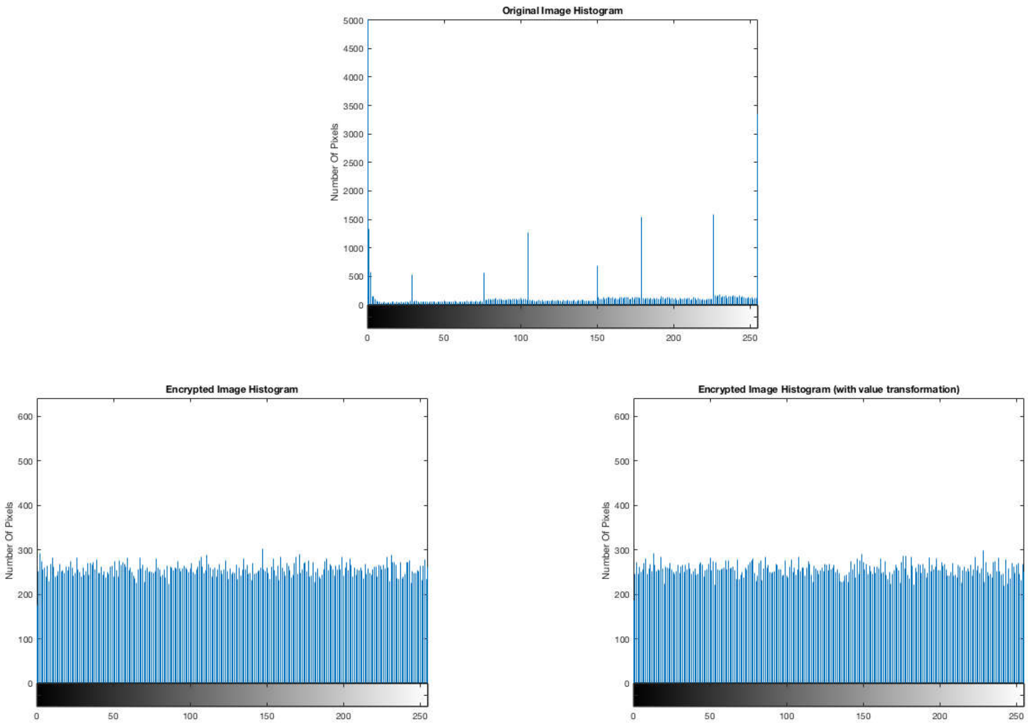


Fig 4.6 : Original Histogram vs histogram after encryption; before and after applying value transformation for checkboard.png

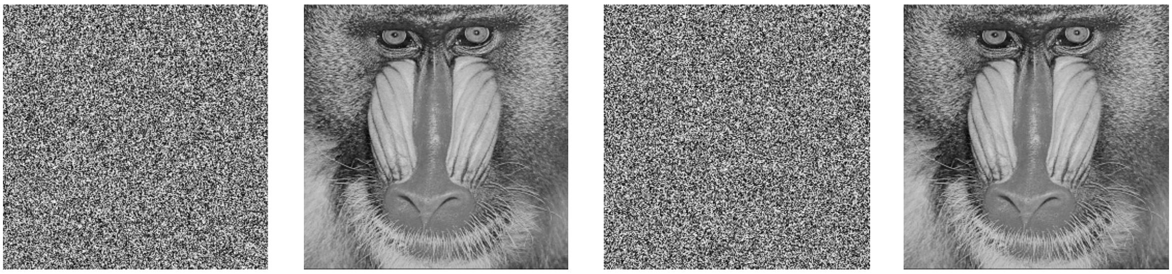
Also upon the computation of MSE of original images and decrypted images, it was found to zero for all image pairs.

4.1.3 Key sensibility Analysis

A good encryption algorithm should be sensitive to the changes in plaintext or key, this means that any small change in the user key should lead to a significant change in the encrypted, or decrypted image. [19]

User key refers to some values given by a user as a set of parameters for an encryption scheme.

(Rubik's Cube algorithm) (Rubik's Cube algorithm with value transformation)



Decrypted using different key | Decrypted using correct key | Decrypted using different key | Decrypted using correct key

Fig 4.7: Key sensibility for decryption

4.2 Result

It was found that Rubik's cube algorithm can efficiently encrypt images. Out of the tested six images, the NPCR values for the Rubik's cube algorithm were found to be between the range of 99.58 %and 99.75% and with the addition of value transformation function, the NPCR values for the same images were found to be in the range of 99.61% to 99.77 % .

Similarly, for UACI values for the Rubik's cube algorithm were found to be between the range of 27.88%and 50.10% and with the addition of value transformation function, the NPCR values for the same images were found to be in the range of 27.78% to 50.06 %.

For the Entropy values, both the Rubik's cube algorithm before and after applying the value transformation function, gave the values between 7.996 and 7.999.

The histograms obtained before and after the value transformation on Rubik's cube algorithm were significantly different from that of the original image. Also when the encrypted image was decrypted using incorrect key, decryption didn't succeed.

CHAPTER 5

CONCLUSION AND LIMITATION

5.1 Conclusion

Images contains higher amount of information and since the pixels in an image can be highly correlated, a secure encryption algorithm that creates significantly dispersed cipher image is a must. Though Rubik's cube algorithm can produce cipher images with higher NPCR, entropy and ideal UACI values, Addition of transformation function showed some improvement on NPCR for 5 images out of 6 test images and improvement on UACI values for 4 images out of 6 test images. Similarly, there was some improvement on entropy value for an image while for other images the values remained constant.

Thus it is concluded that value transformation on Rubik's cube algorithm for image encryption makes it more secure and efficient.

5.2 Limitations and Future Work

The study is done only on gray-scale images and could be done on color images too. Parameters like NPCR, UACI have been used for the analysis purpose and other parameters could also be incorporated.

REFERENCES

[1] Somaya Al- Maadeed, Afnan Al-Ali & Turki Abdalla, "A New Chaos-Based Image-Encryption and Compression Algorithm," *Journal of Electric and Computer Engineering, vol 2012, Article ID 179693*, 2012.

[2] D. Salomon, Data Compression (Springer, 2nd edition, 2000, New York)

[3] Chang Kisik, Deng Robert, Feng Lee, Sangjin Kim, Hyungjun, and Lim Jongin. "On Security Notions for Steganalysis", Information Security and Cryptology – ICISC 2004: 7th International Conference, Seoul, Korea

[4] A. Massoud, F. Lefebvre, C. De Vleeschouwer, B. Macq and J. Quisquater, "Overview on Selective Encryption of Image and Video: Challenges and Perspectives" *EURASIP Journal on Information Security* 2008

[5] Zeghid, Medien & Machhout, Mohsen & Khriji, Lazhar & Baganne, Adel & Tourki, Rached, "A Modified AES Based Algorithm for Image Encryption," *World Academy of Science, Engineering and Technology,21*,2008.

[6] B.Subramanyan, Vivek.M.Chhabria, T.G.Sankar bab, "Image Encryption Based On AES Key Expansion," *International Conference on Emerging Applications of Information Technology*, 2011.

[7] Shuangyuan, Yang & Zhengding, Lu & Han, Shuihua, "An asymmetric image encryption based on matrix transformation," *IEEE International Symposium on Communications and Information Technology,* pp. 66-69 vol.1, 2004.

[8] L. Chen, X. Chen and Z. Peng, "A Novel Public Key Encryption Scheme for Large Image," *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 955-960, 2014.

[9] L. Kocarev, "Chaos-based cryptography: a brief overview," *IEEE Circuits and Systems Magazine*,Vol.1(3), pp. 6-21, 2001.

[10] Min Li Ting Liang Yu-jie He "Arnold Transform Based Image Scrambling Method*", 3rd International Conference on Multimedia Technology,* 2013

[11] Q. Guo, Z. Liu, and S. Liu, "Color image encryption by using Arnold and discrete fractional random transforms in HIS space*," Optics and Lasers in Engineering*, vol. 48, no. 12, pp. 1174–1181, 2010.

[12] Changjiang Zhang et al. , "Digital Image Watermarking Algorithm with Double Encryption by Arnold Transform and Logistic", *Fourth International Conference on Networked Computing and Advanced Information Management*, 2008, pp. 329-334

[13] Khaled Loukhaoukha, jean- yeaves chouinard, and Abdellah Berdai, "A Secure Image Encryption Algorithm Based on Rubik's Cube Principle" *Journal of Electrical and Computer Engineering* vol. 2012, Article ID 173931

[14] M. Yang, N. Bourbakis, and L. Shujun, "Data-image-video encryption," Potentials, IEEE, vol. 23, pp. 28-34, 2004.

[15] C. X. Zhu, Z. G. Chen, and W. W. Ouyang, "A new image encryption algorithm based on general Chen's chaotic system," Journal of Central South University (Science and Technology), 2006.

[16] Yue Wu, Joseph P. Noonan, Sos Agaian, "NPCR and UACI Randomness Tests for Image Encryption" *Journal of Selected Areas in Telecommunications (JSAT)*, 2011

[17] Loukhaoukha, Khaled, Makram, Nabti, Zebbiche, K., “An efficient image encryption algorithm based on blocks permutation and Rubik's cube principle for iris images” *8th International Workshop on Systems, Signal Processing and Their Applications*, 2013

[18] C. E. Shannon, “Communication theory of secrecy systems,” *Bell System Technical Journal*, vol. 28, pp. 656–715, 1949.

[19] Cao Guanghui, Hu Kai, Zhang Yizhi, Zhou Jun, and Zhang Xing, “Chaotic Image Encryption Based on Running-Key Related to Plaintext,” *The Scientific World Journal*, vol. 2014, Article ID 490179, 9 pages, 2014.

.

APPENDIX

Source code

Rubik’s cube encryption

```
warning off;
clc;
clear all;
close all;
mkdir('Encrypted Images');
mkdir('Encryption Keys');
ITER_MAX=1;
disp('Select Image To be Encrypted...');
[file_name path_name] = uigetfile('*.*','Select Input Image');
disp('Encrypting...');
complete_name= strcat(path_name,file_name);
Img=imread(complete_name);
imshow(Img);
Io=rgb2gray(Img);title('Original Image');
figure;
imshow(Io);title('Grayscale Image');
[M,N]=size(Io);
KR=zeros(1,M);
KC=zeros(1,N);
A=randperm(256,256);
A=A-1;
ind=1;
for i=1:M;
    KR(i)=A(ind);
    if(ind==256)
        ind=1;
    else
```

```
        ind=ind+1;
    end;
end;
A1=randperm(256,256);
A1=A1-1;
ind=256;
for j=1:N;
    KC(j)=A(ind);
    if(ind==1)
        ind=256;
    else
        ind=ind-1;
    end;
end;
for i=1:M;
    a(i)=sum(Io(i,:));
    M_a(i)=mod(a(i),2);
end;

I_SCR=zeros(size(Io));
for i=1:size(Io,1);
    if(M_a(i)==0)
        I_SCR(i,:)=circshift(Io(i,:),KR(i));
    else
        I_SCR(i,:)=circshift(Io(i,:),-KR(i));
    end;
end;
for j=1:N;
    B(j)=sum(I_SCR(:,j));
    M_B(j)=mod(B(j),2);
end;
```

```
for j=1:size(Io,2);
    if(M_B(j)==0)
        I_SCR(:,j)=circshift(I_SCR(:,j),-KC(j));
    else
        I_SCR(:,j)=circshift(I_SCR(:,j),KC(j));
    end;
end;
I1=zeros(size(I_SCR));
figure;
imshow(I_SCR,[]);title('Scrambled Image');
for i=1:M;
    if(mod(i,2)==0)
        for j=1:N;

I1(i,j)=bi2de(bitxor(de2bi(I_SCR(i,j),8),fliplr(de2bi(KC(j),8)))
);
        end;
    else
        for j=1:N;

I1(i,j)=bi2de(bitxor(de2bi(I_SCR(i,j),8),de2bi(KC(j),8)));
        end;
    end
end;
I_ENC=zeros(size(I1));
for j=1:N;
    if(mod(j,2)==0)
        for i=1:M;

I_ENC(i,j)=bi2de(bitxor(de2bi(I1(i,j),8),fliplr(de2bi(KR(i),8)))
);
```

```
        end;
    else
        for i=1:M;

I_ENC(i,j)=bi2de(bitxor(de2bi(I1(i,j),8),de2bi(KR(i),8)));
        end;
    end
end;
for i=1:size(Io,1);
    for j=1:size(Io,2);
        if(Io(i,j)==I_ENC(i,j))
            D(i,j)=0;
        else
            D(i,j)=1;
        end;
    end;
end;
MSE=mean((double(Io(:))- double(I_ENC(:))).^2);
fprintf('MSE Between Original Image and Decrypted
Image:%d\n',MSE);
NPCR=(((sum(D(:)))/(size(Io,1)*size(Io,2)))*100);
UACI=((sum(sum((abs(double(Io)-
I_ENC))./(255))))*((100)/(size(Io,1)*size(Io,2))));
disp('Encryption Completed Successfully!!!');
fprintf('Number Of Pixels Change Rate (NPCR) = %f%%\n',NPCR);
fprintf('Unified Average Changing Intensity (UACI) =
%f%%\n',UACI);
fprintf('Entropy of Original Image = %f Sh\n',entropy(Io));
fprintf('Entropy of Encrypted Image = %f
Sh\n',entropy(uint8(I_ENC)));
figure;imshow(I_ENC,[]);title('Encrypted Image');
```

```
figure;imhist(Io);title('Original Image
Histogram');ylabel('Number Of Pixels');
figure;imhist(uint8(I_ENC));title('Encrypted Image
Histogram');ylabel('Number Of Pixels');
imwrite(uint8(I_ENC),strcat(cd,'/Encrypted
Images/',file_name(1:end-4),'_Rubiks_Cube_Encrypted_Img.png'));
save(strcat(cd,'/Encryption Keys/',file_name(1:end-
4),'_Key.mat'),'KC','KR','ITER_MAX');
```

Rubik's Cube decryption

```
warning off;
clc;
clear all;
close all;
mkdir('Decrypted Images');
ITER_MAX=1;
disp('Select Image To be Decrypted...');
[file_name path_name] = uigetfile('*.*','Select Encrypted
Image');
complete_name= strcat(path_name,file_name);
I_ENC=imread(complete_name);
disp('Select Encryption Key of Image...');
[file_name path_name] = uigetfile('*.*','Select Encryption Image
Key');
complete_name= strcat(path_name,file_name);
load(complete_name);
disp('Decrypting...');
figure;
imshow(iI_ENC,[]);title('Encrypted Image');
I_ENC=double(I_ENC);
for j=1:size(I_ENC,2);
```

```
    if(mod(j,2)==0)
        for i=1:size(I_ENC,1);

I1(i,j)=bi2de(bitxor(de2bi(I_ENC(i,j),8),fliplr(de2bi(KR(i),8)))
);
        end;
    else
        for i=1:size(I_ENC,1);

I1(i,j)=bi2de(bitxor(de2bi(I_ENC(i,j),8),de2bi(KR(i),8)));
        end;
    end
end;
I_SCR=zeros(size(I_ENC));
for i=1:size(I_ENC,1);
    if(mod(i,2)==0)
        for j=1:size(I_ENC,2);

I_SCR(i,j)=bi2de(bitxor(de2bi(I1(i,j),8),fliplr(de2bi(KC(j),8)))
);
        end;
    else
        for j=1:size(I_ENC,2);

I_SCR(i,j)=bi2de(bitxor(de2bi(I1(i,j),8),de2bi(KC(j),8)));
        end;
    end
end;
for j=1:size(I_SCR,2);
    B_SCR(j)=sum(I_SCR(:,j));
    M_B_SCR(j)=mod(B_SCR(j),2);
```

```
end;
for j=1:size(I_SCR,2);
    if(M_B_SCR(j)==0)
        I_SCR(:,j)=circshift(I_SCR(:,j),KC(j));
    else
        I_SCR(:,j)=circshift(I_SCR(:,j),-KC(j));
    end;
end;
for i=1:size(I_SCR,1);
    a_SCR(i)=sum(I_SCR(i,:));
    M_a_SCR(i)=mod(a_SCR(i),2);
end;
for i=1:size(I_SCR,1);
    if(M_a_SCR(i)==0)
        I_SCR(i,:)=circshift(I_SCR(i,:),-KR(i));
    else
        I_SCR(i,:)=circshift(I_SCR(i,:),KR(i));
    end;
end;
disp('Decryption Completed Successfully!!!');
Org_Img=imread(strcat(cd,'/Data Images/',file_name(1:end-
8),'.png'));
Org_Img=double(rgb2gray(Org_Img));
Dec_Img=double(I_SCR);
MSE=mean((Org_Img(:)-Dec_Img(:)).^2);
fprintf('MSE Between Original Image and Decrypted
Image:%d\n',MSE);
figure;
imshow(I_SCR,[]);title('Decrypted Image');
imwrite(uint8(I_SCR),strcat(cd,'/Decrypted
Images/',file_name(1:end-4),'_Rubiks_Cube_Decrypted_Img.png'));
```

Code for added transformation encryption

```
Image_Pixel_Count=size(I_ENC,1)*size(I_ENC,2);
Selected_Pixels=randperm(Image_Pixel_Count,500);
Bit_Location=randi([1,8],1,500);
for pix_ind=1:length(Selected_Pixels);
    Binary_Num=zeros(1,8);
    Binary_Num(Bit_Location(pix_ind))=1;

I_ENC(Selected_Pixels(pix_ind))=(bi2de(bitxor(de2bi(I_ENC(Select
ed_Pixels(pix_ind)),8),Binary_Num)));
end;
```

Code for added transformation decryption

```
for pix_ind=1:length(Selected_Pixels);
    Binary_Num=zeros(1,8);
    Binary_Num(Bit_Location(pix_ind))=1;

I_ENC(Selected_Pixels(pix_ind))=(bi2de(bitxor(de2bi(I_ENC(Select
ed_Pixels(pix_ind)),8),Binary_Num)));
end
```