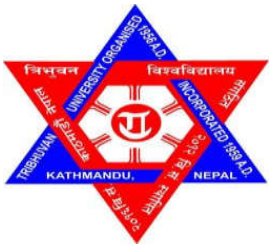


Tribhuvan University

Institute of Science and Technology

"Comparative Analysis of Double Image Encryption Schemes: Arnold Cat Map-Blowfish and Arnold Cat Map-TEA"

Dissertation

Submitted To

Central Department of Computer Science & Information Technology
Institute of Science and Technology
Tribhuvan University, Kathmandu, Nepal

In Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science & Information Technology

By

Sampurna Shakya

T.U. Registration No.: 5-1-22-63-2004

T.U. Examination Roll No.:76/069

Date (April, 2019)

Supervisor

Mr. Dhiraj Kedar Pandey

Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Date: …….

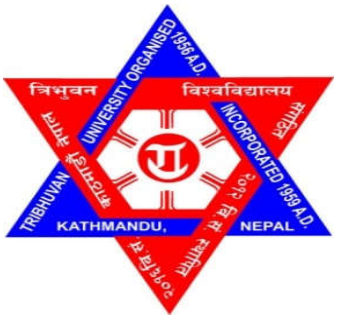
Student's Declaration

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

… … … … … … …

Sampurna Shakya

Date: April, 2019



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

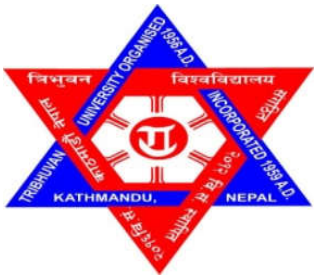
Date: …….

Supervisor's Recommendation

I hereby recommend that the dissertation prepared under my supervision by **Mr. Sampurna Shakya** entitled **"Comparative Analysis of Double Image Encryption Schemes: Arnold Cat Map-Blowfish and Arnold Cat Map-TEA"** be accepted as in fulfilling partial requirement for the completion of Master's Degree of Science in Computer Science & Information Technology. In my best knowledge this is an original work in computer science.

Asst. Prof. Dhiraj Kedar Pandey

Central Department of Computer Science and Information Technology,
Institute of Science and Technology,
Kirtipur, Kathmandu, Nepal



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Date: ……..……..

LETTER OF APPROVAL

We certify that we have read this dissertation work and in our opinion it is appreciable for the scope and quality as a dissertation in the partial fulfillment of the requirements of Master's Degree of Science in Computer Science & Information Technology.

Evaluation Committee

Asst. Prof. Nawaraj Paudel
Head of Department
Central Department of Computer Science
& Information Technology
Tribhuvan University
Kirtipur

Asst. Prof. Dhiraj Kedar Pandey
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. Dhiraj Kedar Pandey, 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.

Sampurna Shakya
CDCSIT, TU

Abstract

Security of digital images becomes an important issue with the rapid growth of digital communication and multimedia applications due to the development of the communication of digital products through networks which occurs frequently. Recent cryptography algorithms are providing essential techniques to protect information and multimedia data from unauthorized access. In this study double image encryption based on chaos image encryption and block cipher encryption are implemented and analyzed with different parameters like NPCR, UACI, correlation-coefficient, histogram and encryption speed to measure the strength of algorithms. The average value of NPCR for Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are 99.87008837 and 99.87275865 respectively. The average correlation value of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are -7.97E-07 and -1.89E-06 respectively. The average encryption time of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA is 69.56 ms and 73 ms respectively. Similarly, the average decryption time of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA is 69.44 ms and 72.89 ms respectively. The average value of UACI for Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are 28.1520017 and 27.1723318 respectively. Hence, on the basis on NPCR and correlation coefficient Arnold Cat Map-TEA has better performance while on the basis of computational time and UACI Arnold Cat Map-Blowfish showed better performance. Arnold Cat Map-Blowfish and Arnold Cat Map-TEA both have greater difference in histogram of cipher and plain image so both strategies have strong impact on histogram analysis.

Keywords: *Image encryption, Chaos encryption, Block cipher, Arnold Cat Map, Blowfish, Tiny Encryption Algorithm.*

Table of Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

TEA	Tiny Encryption Algorithm
ACM	Arnold Cat Map
ECB	Electronic Code Book
CBC	Cipher Block Chaining
OFB	Output Feedback
CFB	Cipher Feedback
IDE	Integrated Development Environment
NPCR	Number of Pixel Change Rate
UACI	Unified Average Change Intensity

Chapter 1

Introduction

1.1 Introduction

With the growing development of digital communication, the communication through multimedia components is on demand. The data like text, images, video and audio is communicated through network. For security issues cryptographic techniques are used which gives security services like confidentiality, data integrity, and authentication to protect against the attacks [1].

Cryptography is an art and science of providing information by transferring it into an unreadable format which can be only read by the expected receiver having the secret key. It provides a method for securing and authenticating the transmission of information across insecure communication channels. It enables us to store sensitive information or transmit it over insecure communication networks so that unauthorized persons cannot read it [2].

Cryptography is applied to protect image at the receiving and sending ends. Image encryption is one of the important fields of cryptography. Image encryption is the conversion of an image to unknown format using a cryptographic algorithm and a key and image decryption is the conversion of the unknown format of an image to the original image using a decryption algorithm. A digital image can be considered as a two-dimensional matrix of numbers. The elements of matrix are called pixels. The value of these pixels are digital numbers and each pixel can be denoted by a position as (row, column). By encrypting an image, it is meant to apply a symmetric or asymmetric encryption algorithm on an input image to be converted into a cipher image. Symmetric ciphers only use one key for encryption and decryption processes while asymmetric ciphers use two different key pairs (i.e. public and private keys) [3].

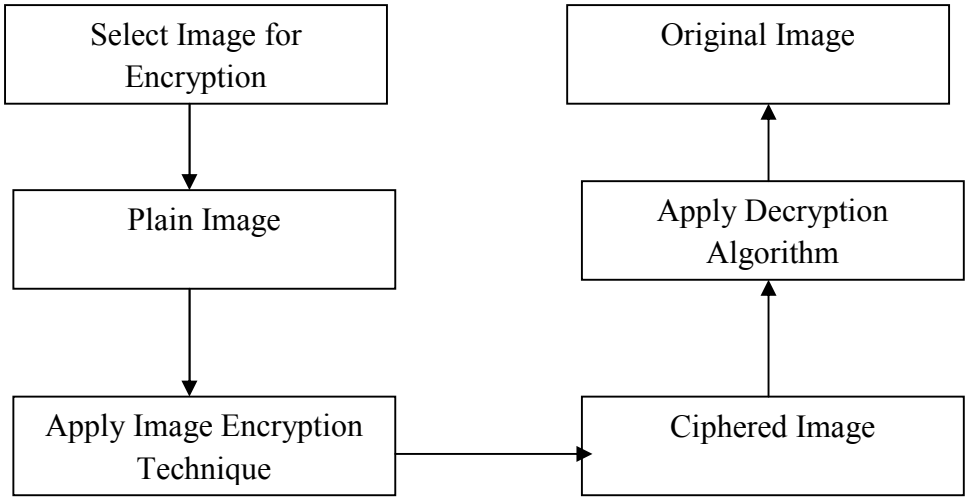


Fig: 1 Image Encryption and Decryption

Symmetric key algorithms are broken down into Block Cipher and Stream Cipher algorithms. Block Cipher algorithms work by breaking up the message into smaller blocks and encrypting each block individually. Stream Ciphers, on the other hand, encrypt data one bit at a time. Blowfish and Tiny Encryption Algorithms are symmetric block cipher algorithms. This study aims to encrypt an image with Arnold Cat Map and symmetric block cipher algorithms.

Blowfish is also a symmetric key Feistel structured algorithm consisting of 2 parts: key expansion part and data encryption part. It is a fast and simple block encryption algorithm. Blowfish is based on Feistel network supports 64-bit block and a key size of 32-448 bit. It contains 4 s-boxes and performs 16 processing rounds [4]. Blowfish is considered a highly rated encryption algorithm in terms of security, with different structure and functionality than the other encryption algorithms. The advantage of this algorithm is that it is highly secure and has not been cracked yet [5, 6].

Tiny Encryption Algorithm (TEA) is also a Feistel structured symmetric key algorithm. It also consists of a 128-bit key size and is known for its simple structure and easy implementation [7]. TEA is a block cipher that uses a 64-bit plain text with 64 rounds and a key Length of 128-bit.

Arnold's Cat Map is chaotic two dimensions that can be used to change the position of the pixel of the image without removing any information from the image. An iteration of

Arnold's cat map is the effect of a matrix multiplication and then the modulo operation on the pixel coordinate values.

1.2 Problem Definition

The security of digital images has become more important due to the rapid evolution of the internet in the digital world today. The security of digital images has attracted more attention recently, and many different image encryption methods have to enhance the security of these images [8]. Each of the encryption technique has its own strong and weak points. In order to apply an appropriate technique in a particular application, it is required to know their strengths and weakness. Therefore, the analysis of these techniques is critically necessary. An algorithm is considered to better if it provides strong security. A desirable property of any encryption algorithm is that a small change in either the plaintext image or the key should produce a significant change in the cipher image. Using only chaotic has disadvantage of having less key space. Chaos is used with block cipher to expand diffusion and confusion in the image. There are many encryption algorithms for creating a cipher, so performance and security analysis is needed for selecting the best encryption algorithm with the strong security. In images, the values of the two neighbor pixels are strongly related to each other. i.e. if we have the value of any one of the pixel than we can easily predict the value of other one pixel (called correlation among pixels). With the aim to reduce this high correlation between pixels and to increase the number of pixel change rate, double image encryption is required with strong security.

1.3 Objective:

The objectives of the research are:-

- To implement two Double Image Encryption strategies :- Arnold Cat Map-Blowfish Encryption and Arnold Cat Map-TEA Encryption.
- To compare implemented algorithms on the basis of different parameters like Number of Pixel Change Rate(NPCR), Unified Average Change Intensity(UACI), Computational Time Analysis, Correlation Coefficient of image and Histogram Analysis.

1.4 Thesis Organization

The organization of this thesis is as follows:

Chapter 1 describes the introduction, problem statement and objectives.

Chapter 2 describes about the background study for the research and literature review of the related work by different authors.

Chapter 3 describes the overview of the candidate algorithms and methodology of Double Image Encryption Schemes: Arnold Cat Map-Blowfish and Arnold Cat Map-TEA.

Chapter 4 contains the implementation overview of the Double Image Encryption Schemes: Arnold Cat Map-Blowfish and Arnold Cat Map-TEA in C# platform along with the analysis of different performance parameters of methodology.

Finally chapter 5 concludes with the main theme of the work.

Chapter 2

Background Study and Literature Review

2.1 Background Study

2.1.1 Chaotic Maps

Chaos theory was discovered by U.S meteorologist Edward N. Lorenz in 1963. Chaos theory is based on the observation that simple rules when iterated can give rise to apparently complex behavior [9]. Chaotic models are quite sensitive to the initial point, which means even a very slight change in the value of initial point would result in a dramatic change of the sequence produced by the chaotic map. With the properties of sensitivity to initial conditions and control parameters, pseudo-randomness and ergodicity, chaotic maps have been widely used in data encryption recently. Compared with traditional cryptosystems, the ones based on chaos are easier to be realized, which makes it more suitable for large-scale data encryption such as images, videos or audio data [10]. In mathematics, a chaotic map is a map that exhibits some sort of chaotic behavior. Maps may be parameterized by a discrete-time or a continuous-time parameter. Discrete maps usually take the form of iterated functions. Chaotic maps often occur in the study of dynamical systems.

2.1.2 Chaotic Encryption

Chaotic encryption uses a mathematical function of the chaotic map to shuffle and scramble the image. The encryption scheme is composed of two steps: chaotic confusion and pixel diffusion, where the former process permutes a plain-image with a 2D chaotic map, and the latter process changes the value of each pixel one by one. In the confusion process, the parameters of the chaotic map serve as the confusion key and in the diffusion process, such parameters as the initial value or control parameter of the diffusion function serve as the diffusion key which ensures high randomization of the image data during encryption.

2.1.3 Block Cipher

There are number of methods in cryptography to handle the encryption/decryption process where some handle the data in bit level and some works with block of data. Those which works with bit level is stream ciphers whereas those which works with block of data is block cipher. Mathematically block cipher can be defined as: let E be an encipherment algorithm and let $E_k(b)$ be the encipherment of message b with key k. Let a message $m = b_1b_2\ldots\ldots b_n$. where each b_i is of fixed length (usually 64 bit or 128 bit etc). Then block cipher is a cipher for which $E_k(m) = E_k(b_1)E_k(b_2)\ldots E_k(b_n)$. Block cipher encrypts a block of text, rather than encrypting one bit at a time as in stream ciphers. For example, a common block cipher, Blowfish, encrypts 64 bit blocks with a key of length 32-448 bits. Similarly, a block cipher, Tiny Encryption Algorithm, encrypts 64 bit blocks with a key of length of 128 bits. The block cipher approach uses a block of image pixel value with a secret key to encrypt the pixel and change it to unrecognized form.

Different ways of using a block cipher for encryption, combining some simple operations, are called block cipher modes of operation. There are several modes of operation. They are divided into two groups: the ones which result in deterministic encryption, and the ones in probabilistic encryption. In deterministic encryption schemes, if the key does not change, a particular plaintext is mapped to a fixed cipher text. Example of deterministic mode of operation is Electronic Code Book(ECB). On the other hand, probabilistic encryption schemes use randomness to achieve a nondeterministic generation of cipher text. Example of probabilistic mode of operation are Cipher Block Chaining(CBC), Cipher Feedback mode(CFB), Output Feedback (OFB) mode[11].

2.1.4 Image Encryption

Image encryption is the strategy of encoding image in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can. In an encryption scheme, the image is encrypted using an encryption algorithm, turning it into an unreadable image. This is usually done with the use of encryption keys, which specifies how the image is to be encoded. Any adversary that can see the encrypted image should not be able to determine anything about the original image. An authorized party, however, is able to decode the encrypted image

using a decryption algorithm. That usually requires a secret decryption key, so adversaries do not have access.

2.2 Literature Review

A wide variety of cryptographic algorithms have been proposed for image security.

Jawad Ahmad and Fawad Ahmed [12] proposed Advanced Encryption Standard(AES) and Compression Friendly Encryption Scheme(CFES) for image encryption and were compared and analyzed using different parameters like correlation coefficient, information entropy, Number of Pixel Change Rate and Unified Average Change Intensity etc. In correlation coefficient analysis, AES encrypted image had less correlation in all directions. During the avalanche effect test, AES proved good diffusion characteristics while CFES has less diffusion. The value of NPCR and UACI was very high for AES.

Sudeshna Bora, Pritam Sen and Chittaranjan Pradhan [13] proposed Double Encryption Technique using Blowfish and Cross Chaos Map for image encryption and Double Encryption Technique was compared with Blowfish and Cross Chaos Map separately with different statistical and security parameters. They first used Blowfish encryption algorithm for image encryption and then resulted encrypted image is again encrypted by Cross Chaos Map. It was found that purposed Blowfish- Cross Chaos is able to protect different types of images with a high level of security.

Min Li, Ting Liang, Yu-jieHe [14] proposed Arnold Transform Based Image Scrambling Method, in which they introduced a multi-area scrambling method which not only used for a square image but also any non-square images. They found that by using multi-region scrambling, it could more effectively improve the security of image, lead decipher more difficult.

Zhang Yun-peng, Zheng-jun, Liu Wei [15] proposed Digital Image Encryption Algorithm Based on Chaos and Improved DES. Combination of Chaos and Improved DES makes the final algorithm more secure and more suitable for digital image encryption on the basis of statistical, security analysis, correlation-coefficient analysis etc.

Akshit Shah, Aagam Shah, Tanaji Biradar [16] proposed Image Encryption and Decryption using Blowfish algorithm using Matlab in which he found histogram of the encrypted image is less dynamic and significantly different from the histograms of the original image.

Alireza Jolfaei, Abdolrasoul Mirghadri [17] proposed Image Encryption using Chaos and Block Cipher in which they encrypted image based on a combination of Chaotic Baker's Map and a modified version of S-AES. They used Baker's map to generate a permutation matrix, which is in turn used to generate S-box in the S-AES algorithm.

Image Encryption Based On Arnold Cat Map and S-Box was proposed by Priyanka Gupta, Sonia Singh, Isha Mangal [18]. In this paper, firstly Arnold Cat Map was used to shuffle the positions of the image pixels and then that shuffled image was encrypted by byte-substitution using S-box of AES encryption algorithm. Experimental results showed that the presented algorithm was not vulnerable to statistical attack but it is weak against differential analysis.

K.Brindha, Ritika Sharma, Sapanna Saini [19] proposed image encryption using the symmetric algorithm, that is, DES and compared with AES algorithm. For image encryption, he first converted the image into a byte array and that byte array was changed into a string object. Then that string was passed for encryption in DES. He showed the image processing using MATLAB and encryption, decryption part in the Java language. He concluded as DES algorithm provides high security during transmission.

Chapter 3

Methodology

3. 1 Methodology

In this present work, an image cryptography approach is presented using Arnold Cat Map and Block cipher approaches like Blowfish and TEA algorithm. That is, two schemes for image encryption are described.

In first image encryption technique, an image is scrambled first with Arnold Cat Map, then that resultant scrambled image is encrypted by Blowfish algorithm. In second image encryption technique, an image is scrambled first with Arnold Cat Map, then the resultant scrambled image is encrypted by TEA algorithm. In this way, in this thesis, two encryption techniques are compared on the basis of Number of Pixel Change Rate (NPCR), Unified Average Change Intensity (UACI), Computational Time Analysis, Correlation Coefficient and Histogram Analysis.

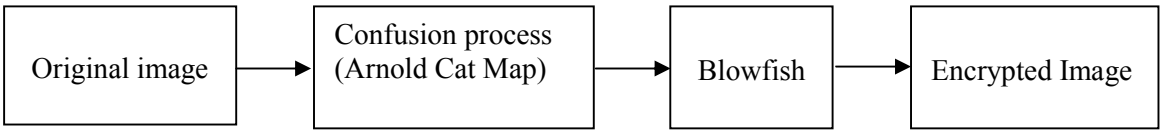


Fig: 2 Double Image Encryption by Arnold Cat Map and Blowfish

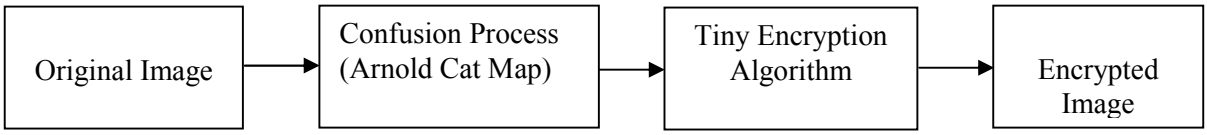


Fig: 3 Double Image Encryption by Arnold Cat Map and Tiny Encryption Algorithm

3.2 Candidate Algorithms:

3.2.1 Arnold Cat Map (ACM)

Arnold's Cat Map was discovered by Russian mathematician Vladimir Arnold in 1960. It is one of the cryptographic algorithm used to encrypt the image. The concept of the algorithm continuously rotates the image so that it becomes a form that is not visible and random and the image cannot be recognized by the naked eye. ACM reduces the relationship among neighboring pixels [20]. Arnold cat map is well known for its complex randomization of the image pixels. After several iterations the actual image reappears. The number of iterations taken to change the pixel positions is defined as Arnold's period. The periods change in correspondence to the size of images i.e. when the size of the image increases, Arnold's period also increases [21, 22].

Consider an N × N image and x and y be the row and column number of the pixels in the image. Thus x and y both ranges from 1 to N. Arnold's Cat Map transformation of the image is obtained by implementing the formula given below:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} (\text{mod } n) \quad \text{.....Eq(1)}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} (\text{mod } n) \quad \text{.....Eq(2)}$$

Where, p and q are positive integers, and determinant of matrix A is 1. When Arnold cat map algorithm is executed once, the original pixel positions co-ordinate will be transferred from the (x, y) to a new pixel position (x', y') then the process is repeated with the A matrix multiplied and the number of moves is T. After being multiplied few times, the correlation between the pixels will be completely chaotic [23].

The reverse procedure to re-shuffle the process is given by the following equation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} (\text{mod } n) \quad \text{.....Eq(3)}$$

Arnold's Cat Map encryption algorithm

let sx, sy be source pixel coordinates of a square image.

let dx, dy be destination pixel coordinates in the same image.

let n be the width/height of image.

```
for i = 1 to n Iterations
        for sx = 0 to n - 1
                for sy = 0 to n - 1
                        dx = (sx + sy) % n
                        dy = (sx + 2 * sy) % n
                        copy pixel from (sx, sy) to (dx, dy)
                end
        end
end
```

Arnold's Cat Map Decryption algorithm

let sx, sy be source pixel coordinates of a square image.

let dx, dy be destination pixel coordinates in the same image.

let n be the width/height of image

```
for i = 1 to n Iterations
        for sx = 0 to n - 1
                for sy = 0 to n - 1
                        dx = (2 * sx - sy) % n
                        if dx < 0 then dx = dx + n
                        dy = (-sx + sy) % n
                        if dy < 0 then dy = dy + n
                        copy pixel from (sx, sy) to (dx, dy)
                end
```

end

end

3.2.2 Blowfish

Blowfish is designed by Bruce Schneier in 1993. It is a 64-bit block cipher that processes image which is of blocks size 64 bits using variable length key from 32 bits (4 bytes) to 448 bits (56 bytes). Blowfish image encryption is the conversion of original image i.e plain image into encrypted image i.e. cipher image. In this encryption, a P array of size 18 and four S boxes whose size is 256 each of which are initialized to hexadecimal digits of π. The algorithm is applied for both image encryption and decryption.

Blowfish Encryption

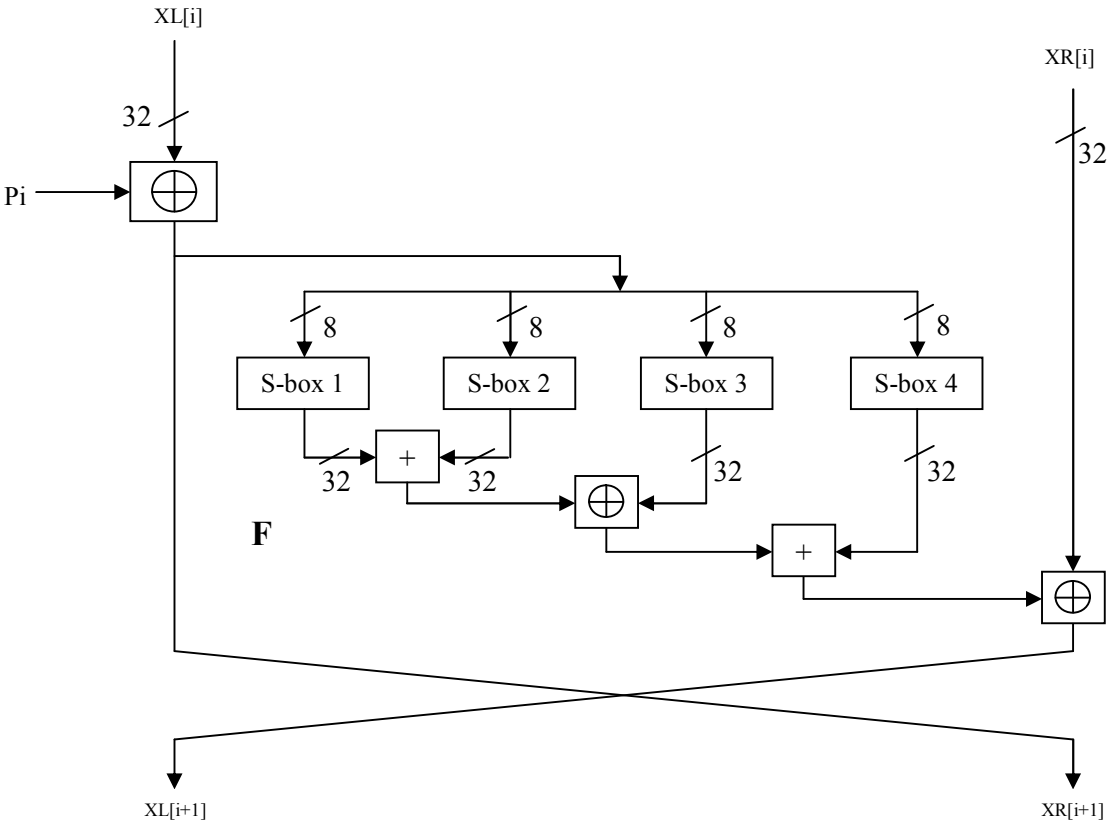


Fig 4: One round of Blowfish Encryption

Data image and the encryption key are two inputs of Blowfish image encryption process. In this case, original image data bit stream is divided into blocks, length of Blowfish algorithm.

For encryption, the 64-bits block is separated into a left and right half each consisting of 32-bits. The encryption routine consists of a 16 round Feistel network.

In the first round, an exclusive-or operation is performed between the left 32-bits (XL-0) and the 32-bit P_1 of the P-array. This value becomes the next 32-bit right value (XR-1) and this value is also inserted into the F function. The F function takes the 32-bit input and separates it into 4 bytes (8-bits each). These four values are then used for table lookup in their respective S-Boxes. The 32-bit values of the S-boxes are then manipulated according to the following formula:

32-bit Output = ((((S[1][a] +S[2][b]) mod 2^32) ⊕S[3][c])+S[4][d])mod 2^32

The result of this operation becomes the left half 32-bit input for the next round (XL-1). The results of round 1 can be explained by the following equations:

XL-1 = F(XL-0 ⊕P_1) ⊕XR-0
XR-1 = XL-0 ⊕P_1

Round 2 is then performed with inputs XL-1 and XR-1. This process is repeated for a total of 16 rounds. After completing the 16 rounds, XL-16 and XR-16 values are swapped. An exclusive-or operation is then performed between the swapped XL-16 and P_{18} and also with the swapped XR-16 and P_{17} to obtain XL-17 and XR-17, respectively. The 32-bit values of XL-17 and XR-17 are combined to obtain the 64-bit cipher block.

Algorithm

1. Divide X into two 32-bit halves: XL, XR
2. For i = 1 to 16
   XL = XL⊕ Pi
   XR=F(XL) ⊕ XR
   Swap XL and XR
3. Swap XL and XR (Undo the last swap)
4. XR=XR ⊕ P17
5. XL = XL⊕ P18
6. Concatenate XL and XR

Where, F(XL)={(S1[a] + S2[b]) ⊕ S3[c]} + S4[d])}

The F function takes 32 bit input and separated it into 4 bytes as a, b, c and d and here + means addition modulo 2^32, and ⊕ means exclusive OR and S1, S2, S3, S4 are four substitution boxes.

Blowfish Decryption

The decryption process for Blowfish is almost identical to the encryption process except the P-array(sub keys) values are reversed. The encrypted image is divided into the same block length of Blowfish algorithm from top to bottom.

Algorithm

1. Divide X into two 32-bit halves: XL, XR
2. For i =18 down to 3:
   XL = XL XOR Pi
   XR = F (XL) XOR XR
   Swap XL and XR
3. Swap XL and XR (Undo the last swap)
4. XR = XR XOR P2
5. XL = XL XOR P1
6. Recombine XL and XR.

3.2.3 Tiny Encryption Algorithm (TEA)

TEA was designed by David Wheeler and Roger Needham in 1994, first presented and published in the proceedings at the Fast Software Encryption workshop [7]. The single TEA round function performs the simple mixed orthogonal algebraic functions such as Right/Left shifts, Integer addition and exclusive-or operations. TEA is a 64-bit block cipher that processes image which is of blocks size 64 bits using key size 128 bits.

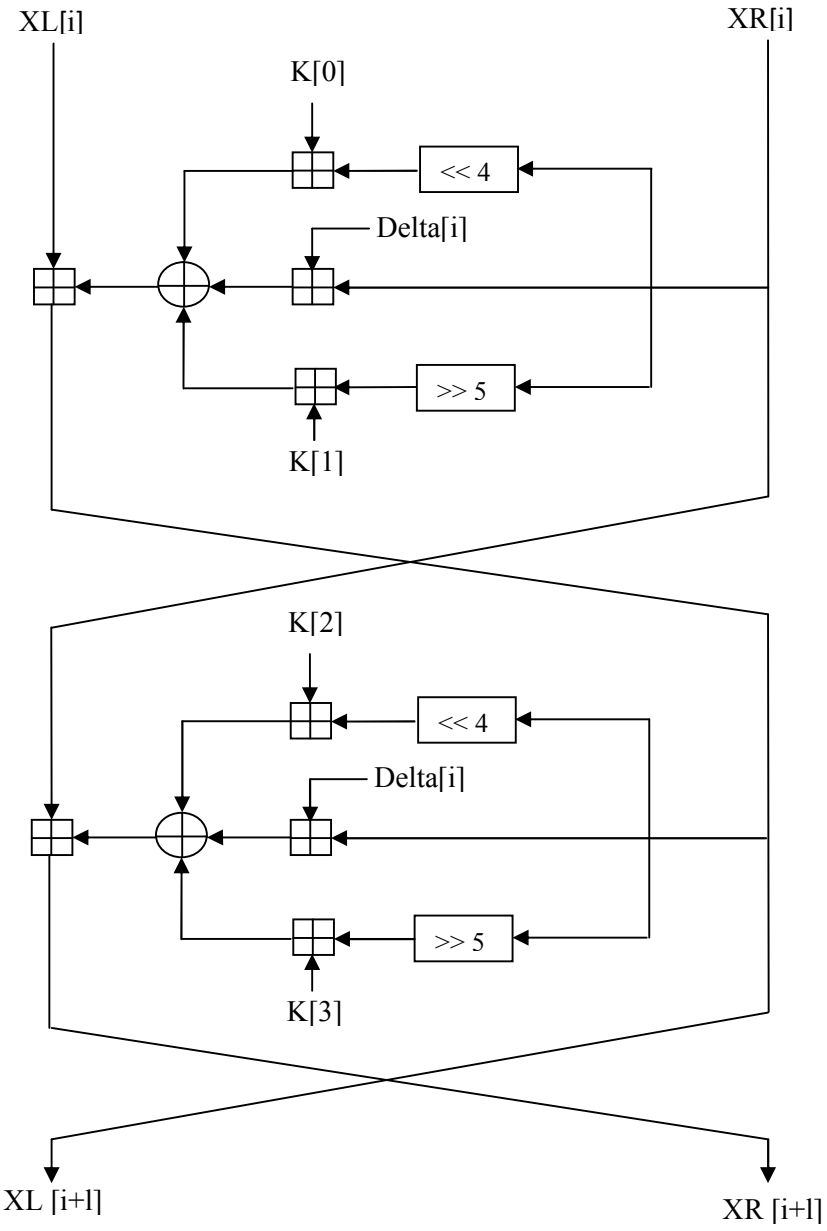


Fig: 5 One Cycle of TEA

A 128-bit key is split into four 32 bit sub-keys. Delta is defined as a constant, 2^32/(golden ratio), which is 2654435769 as an integer (9E3779B9 in hex). Multiples of the delta are used in each cycle. The block size, 64 bit X, is split up into two 32 bit blocks where XL is the left side of the block and XR is the right side. In the first Feistel round, XR is used as an input. All addition operations are (mod 2^32).

1. XR goes through a left shift of 4 and then is added to K[0].

2. XR is added to Delta.

3. XR goes through a right shift of 5 and then is added to K[1].

An XOR operation is then applied to the result of those three operations and finally, the result of the XOR operation is added to XL. This result then again sent for the next Feistel round[24]. A complete cycle of TEA encryption is repeated for thirty two times to reach the requirement of a full TEA encryption.

TEA Algorithm

1. Divide X into two 32-bit halves: XL, XR
2. Initialize delta=0×9E3779B9, y=XL, z= XR
3. For i = 1 to 32

    sum+=delta

    y+=((z<<4)+k[0] ⊕(z+sum) ⊕(z>>5)+k[1])

    z+=((y<<4)+k[2] ⊕(y+sum) ⊕(y>>5)+k[3])

4. XL=y, XR=z

TEA decryption

The decryption process for TEA encryption is identical to the encryption process except the sub keys are used in the reverse order. The encrypted image is divided into the same block length of TEA algorithm.

Algorithm

1. Divide X into two 32-bit halves: XL, XR
2. Initialize delta=0×9E3779B9, sum=delta<<5, y=XL, z= XR
3. For i = 1 to 32

    z-=((y<<4)+k[2] ⊕(y+sum) ⊕(y>>5)+k[3])

    y-=((z<<4)+k[0] ⊕(y+sum) ⊕(z>>5)+k[1])

    sum- = delta

4. XL=y, XR=z

Chapter 4

Implementation and Analysis

4.1 Implementation Tools

All the implementation is done in C#.Net programming language using visual studio 2017. For encryption and decryption by Blowfish and TEA, Bouncy Castle library is used.

4.1.1 Visual Studio 2017

Visual studio 2017 is the integrated development environment developed by Microsoft. This IDE can be used to develop windows applications, websites, web services and windows mobile applications. Visual studio includes a code editor with intellisense as well as code refactoring. It also has the integrated debugger for debugging the .Net codes.

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS .

4.1.2 C#.NET Programming Language

C#.NET is the Microsoft implementation of the C# language integrated in .NET framework. C# was designed by Anders Hejlsberg. It is an object oriented programming language. C# language is powerful language that is used to build small to enterprise level application. It is one of the programming languages designed for the Common Language Infrastructure.

4.2 Testing Environment

The implemented algorithms are executed in the machine with following configuration.

Machine type	: Laptop
Model	: HP Probook 4431s
Processor	: Intel® Core™ i5-2450M CPU @ 2.5GHZ
Installed memory (RAM)	: 4GB
Operating System	: Windows 7 Enterprise, 32bit OS
Hard disk	: 650GB

4.3 Test Data Description

Test data is taken from different repository like photo database provided by Fabien a. p. petitcolas, SIPI image database etc. The input images types are of .gif, .tif, .png types. The image data are secondary. The secondary type of image datasets are the standard images like Lena, Peppers, Baboon, Cameraman, Barbara gold hill, Cat, Jet plane and House.

4.4 Analysis

In this part, comparative analysis of implemented algorithms is performed on the basis of different parameters like Number of Pixel Change Rate (NPCR), Unified Average Change Intensity (UACI), Correlation Coefficients, Computational time analysis, and Histogram Analysis of input image and cipher image. Here, five iterations of Arnold cat map is used and in five iterations the test images get completely randomized.

4.4.1 Number of Pixel Change Rate (NPCR)

NPCR gives the percentage of the number of different pixels to the total number of pixels. That is, NPCR means the number of pixels change rate of the ciphered image while one pixel of plaintext image is changed. Hence, it can be defined as the variance rate of pixels in the encrypted image caused by the change of a single pixel in the original image. For the better algorithm, NPCR should be high [25].

Let C1 and C2 be the two cipher images before and after a one-pixel change in the plaintext image respectively. The pixel value at grid (i, j) in C1 and C2 are represented as C1(i, j), C2(i, j) and a bipolar array D is defined as [12]:

$$D(i, j) = \begin{cases} 0, \text{if } C_1(i,j) = C_2(i,j) \\ 1, \text{if } C_1(i,j) \neq C_2(i,j) \end{cases}$$

Mathematically, NPCR is defined as:

$$NPCR(C_1, C_2) = \frac{\sum_{i,j} D(i,j)}{T} \times 100\% \quad \text{.....Eq(4)}$$

Where, T is total number of pixels.

NPCR of the sample datasets are

Table 1: NPCR Measures

S.No.	Image		NPCR(%)	
			Arnold Cat Map-Blowfish	**Arnold Cat Map-TEA**
1	baboon .gif	512 x 512	99.60365295	99.60403442
2	barbara.png	512 x 512	100	100
3	cameraman.tif	512 x 512	99.61051941	99.60517883
4	cat.png	354x354	100	100
5	goldhill.png	512 x 512	100	100
6	house.tif	512 x 512	100	100
7	jetplane.tif	512 x 512	100	100
8	lena.png	512 x 512	100	100
9	peppers.tif	512 x 512	99.61662292	99.64561462
	Average		99.87008837	99.87275865

The average value of NPCR for Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are 99.87008837 and 99.87275865 respectively. So both techniques have good results, although Arnold Cat Map-TEA has better results than that of Arnold Cat Map-Blowfish because the value it generates is high for NPCR. A high NPCR value is interpreted as high resistance to differential attacks.

4.4.2 Unified Average Change Intensity (UACI)

Unified Average Change Intensity (UACI) determines the average intensity of differences between the two images. For good quality, UACI value should be high [25].

Mathematically, UACI is defined as:

$$UACI(C_1, C_2) = \frac{\sum_{i,j}|C_1(i,j) - C_2(i,j)|}{F \times T} \times 100\% \quad \text{.....Eq(5)}$$

where, T is total number of pixels and F is largest supported pixel value.

Table 2: UACI Measures

S.No.	Image		UACI(%)	
			Arnold Cat Map-Blowfish	**Arnold Cat Map-TEA**
1	baboon .gif	512 x 512	0.2327828	0.18486287
2	barbara.png	512 x 512	50.0830789	49.8560015
3	cameraman.tif	512 x 512	0.26602546	0.21563463
4	cat.png	354x354	49.9809434	50.1060969
5	goldhill.png	512 x 512	50.0472944	50.0353278
6	house.tif	512 x 512	52.2523815	44.4810872
7	jetplane.tif	512 x 512	50.0752139	49.3144739
8	lena.png	512 x 512	0.26490972	0.21505551
9	peppers.tif	512 x 512	0.16538544	0.14244591
	Average		28.1520017	27.1723318

The average value of UACI for Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are 28.1520017 and 27.1723318 respectively. Arnold Cat Map-Blowfish has good results than that of Arnold Cat Map-TEA because the value it generates is high for UACI. A high UACI score is interpreted as high resistance to differential attacks.

4.4.3 Correlation Coefficients

Correlation describes the relationship between pixels in an image. If the encrypted image and plain image are completely different then their corresponding correlation coefficient is low or very close to zero [12]. An efficient image cryptosystem should produce the cipher image with sufficiently low correlation in the adjacent pixels. In the case of an encrypted image, the adjacent pixel correlation will be less if the encryption process is capable of hiding the details of the original image [26].

Correlation Coefficient is calculated as:

$$cc = \frac{cov\ (x,y)}{\sigma x \times \sigma y} \quad \text{.....Eq(6)}$$

where, $\sigma x = \sqrt{var\ (x)}$

$\sigma y = \sqrt{\text{var (y)}}$

$$\text{var(x)} = \frac{1}{N}\sum_{i=1}^{N}(xi - E(x))^2 \quad \text{.....Eq(7)}$$

$$\text{cov(x, y)} = \frac{1}{N}\sum_{i=1}^{N}\big(xi - E(x)\big)(yi - E(y)) \quad \text{.....Eq(8)}$$

where, x and y be the pixel values in the same place in the plaintext and cipher text images. N is the total number of pixels selected to find the correlation in the image.

Table 3: Correlation Coefficient Measures

S.No.	Image		Correlation-Coefficient	
			Arnold Cat Map-Blowfish	**Arnold Cat Map-TEA**
1	baboon .gif	512 x 512	8.32E-07	1.25E-06
2	barbara.png	512 x 512	6.10E-09	-2.92E-08
3	cameraman.tif	512 x 512	-3.24E-06	2.42E-06
4	cat.png	354x354	9.49E-09	3.26E-08
5	goldhill.png	512 x 512	-1.19E-08	1.98E-08
6	house.tif	512 x 512	-7.83E-09	1.57E-08
7	jetplane.tif	512 x 512	3.14E-08	2.50E-08

8	lena.png	512 x 512	-4.39E-06	-5.66E-06
9	peppers.tif	512 x 512	-3.91E-07	-1.50E-05
	Average		-7.97E-07	-1.89E-06

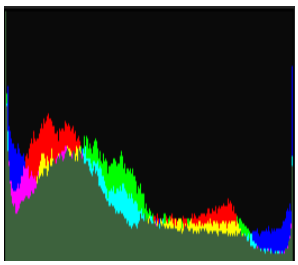
The average correlation value of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are -7.97E-07 and -1.89E-06 respectively and it implies that Arnold Cat Map-TEA gives less correlation value than Arnold Cat Map-Blowfish. So it can be said that the encrypted image given by Arnold Cat Map-TEA is less correlated than the encrypted image given by Arnold Cat Map-Blowfish with their corresponding original images. Hence, Arnold Cat Map-TEA is better than Arnold Cat Map-Blowfish on the basis of Correlation-Coefficient analysis.

4.4.4 Histogram Analysis

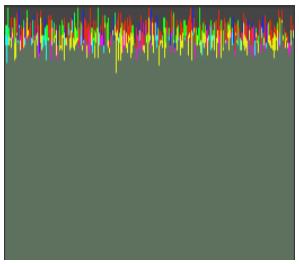
Histogram of an image refers to a graph of the pixel intensity values. Histogram of images provides a global description of the appearance of an image. Information obtained from the histogram is enormous. The encrypted image must have a uniform histogram distribution. The encrypted image does not provide any information about the actual image. This gives encryption in security against attacks like statistical attack [26]. Histogram of input image and cipher image is analyzed graphically. Histograms of the sample dataset are

1. Cat.png

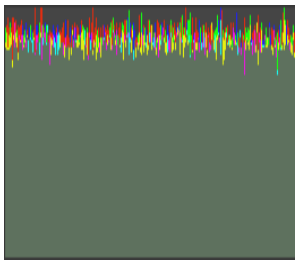
Plain Image



Plain-Image histogram



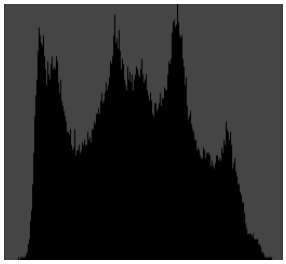
Cipher-image histogram of Arnold Cat Map-Blowfish



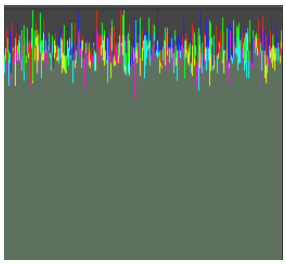
Cipher-image histogram of Arnold Cat Map-TEA

2. Barbara.png

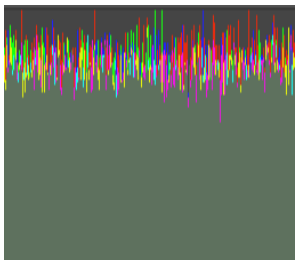
Plain Image



Plain-image histogram



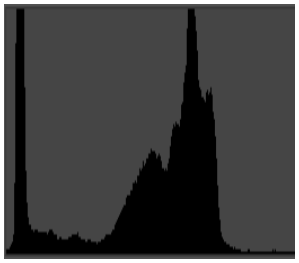
Cipher-image histogram of Arnold Cat Map-Blowfish



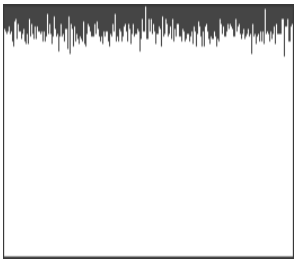
Cipher-image histogram of Arnold Cat Map-TEA

3. Cameraman.gif

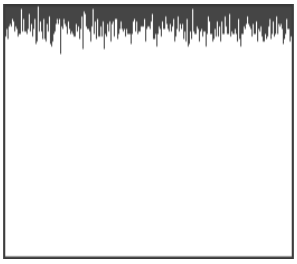
Plain Image



Plain-Image histogram

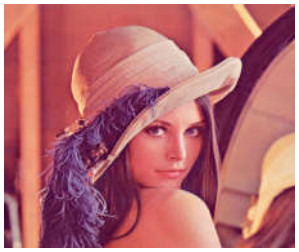


Cipher-image histogram of
Arnold Cat Map-Blowfish

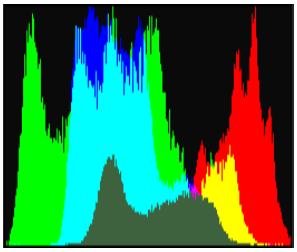


Cipher-image histogram of
Arnold Cat Map-TEA

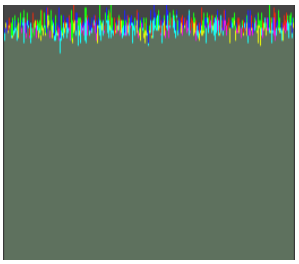
4. Lena.png



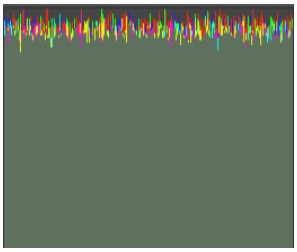
Plain Image



Plain-Image histogram



Cipher-image histogram of
Arnold Cat Map-Blowfish



Cipher-image histogram of
Arnold Cat Map-TEA

Histogram of cipher image of both Arnold Cat Map-Blowfish and Arnold Cat Map-TEA with original image has a significant difference, since greater the difference in histogram more robust the security mechanism is imposed. As the histogram is significantly changed, the frequency of pixel values also varies significantly which prevents statistical attack. Hence, encrypted image histogram does not provide any useful information for the attackers.

4.4.5 Computational Time Analysis

Computation analysis is the time taken by the algorithms to encrypt and decrypt data and it can be also said as execution time. Here, encryption and decryption time is measured for computational performance for different set of images in milliseconds to find which techniques has good results and it is found that the Arnold Cat Map-Blowfish has relatively less encryption/decryption time.

Table 4: Computational Time Analysis

S.No.	Image		Encryption Time		Decryption Time	
			Arnold Cat Map-Blowfish	**Arnold Cat Map-TEA**	**Arnold Cat Map-Blowfish**	**Arnold Cat Map-TEA**
1	baboon .gif	512 x 512	56	56	55	56
2	barbara.png	512 x 512	84	89	84	88
3	cameraman.tif	512 x 512	56	56	54	57
4	cat.png	354x354	41	45	42	43
5	goldhill.png	512 x 512	87	92	86	90
6	house.tif	512 x 512	84	93	84	94
7	jetplane.tif	512 x 512	87	89	86	91
8	lena.png	512 x 512	75	81	78	80
9	peppers.tif	512 x 512	56	56	56	57
	Average		69.55556	73	69.44444	72.88889

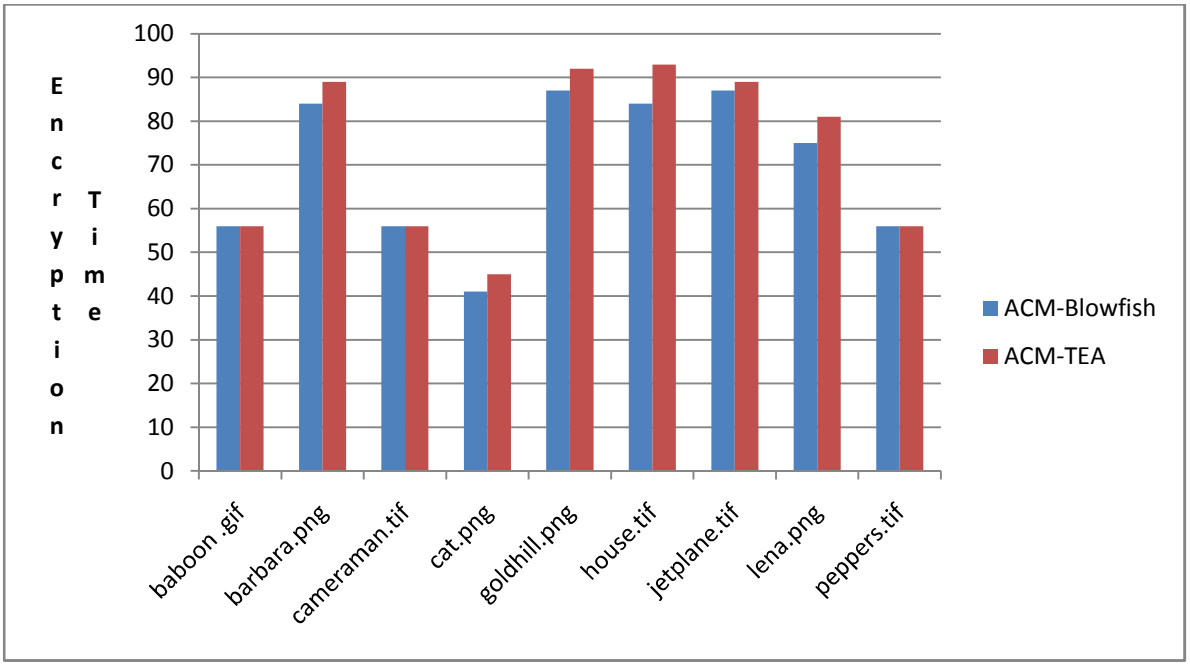


Fig: 4 Encryption Time Measurement(millisecond)

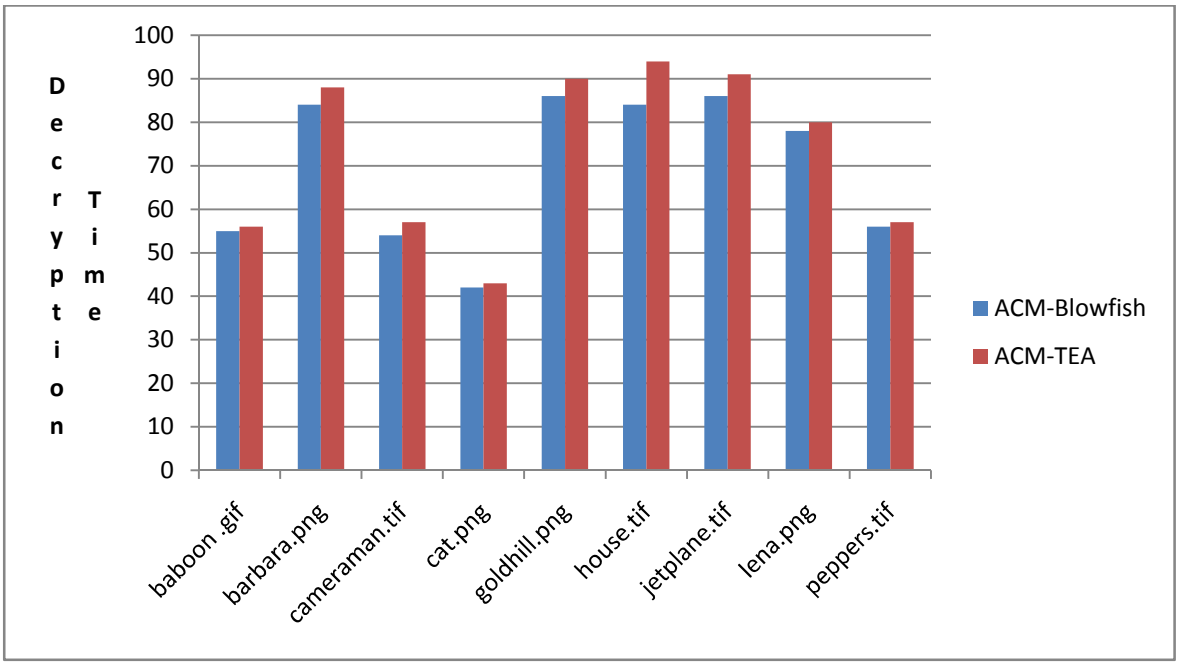


Fig: 5 Decryption Time Measurement (millisecond)

The average encryption time of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA is 69.56 ms and 73 ms respectively. Similarly, the average decryption time of Arnold Cat Map-

Blowfish and Arnold Cat Map-TEA is 69.44 ms and 72.89 ms respectively. Arnold Cat Map-Blowfish has less computational time therefore in this category, Arnold Cat Map-Blowfish works well.

4.5 Result

The range of NPCR is 99-100% from table 1 for both algorithms. It implies that when one pixel of plain image is changed then almost all pixels are changed which means both encryption strategies are very strong for image encryption. Although for some experiment, Arnold Cat Map-TEA gives higher value than Arnold Cat Map-Blowfish. It means that Arnold Cat Map-TEA has good differential analysis value and this value is able to protect the image from attackers as it able to hide the significant information of image data.

From table 2, it is found that average UACI value for Arnold Cat Map-Blowfish is 28.1520017 and for Arnold Cat Map-TEA is 27.1723318. Since the UACI value for Arnold Cat Map-Blowfish is higher than Arnold Cat Map-TEA, it can be said that Arnold Cat Map-Blowfish is better than Arnold Cat Map-TEA. It means there is more differences in the average intensity between images in Arnold Cat Map-Blowfish than Arnold Cat Map-TEA.

From Correlation-Coefficient analysis of table 3, Arnold Cat Map-TEA is better than Arnold Cat Map-Blowfish. Arnold Cat Map-TEA gives less correlation value than Arnold Cat Map-Blowfish. The average correlation value of Arnold Cat Map-Blowfish and Arnold Cat Map-TEA are -7.97E-07 and -1.89E-06 respectively. So it can be said that the encrypted image given by Arnold Cat Map-TEA is more different than the encrypted image given by Arnold Cat Map-Blowfish.

From computational time analysis, Arnold Cat Map-Blowfish has less computational time than Arnold Cat Map- TEA. On average, Arnold Cat Map-Blowfish is faster by 3.44 milliseconds on encryption process and by 3.45 milliseconds on decryption process than Arnold Cat Map-TEA.

The histogram of cipher and original image has significant variation for both algorithms. The greater the difference reduces the chances of attacks by the intruders since it doesn't give any hint of original image data. In histogram analysis both techniques are equally strong to hide the hint of plain image.

Chapter 5

Conclusion and Future Recommendation

5.1 Conclusion

In this study double encryption techniques for image, that is, Arnold Cat Map-Blowfish and Arnold Cat Map-TEA have been implemented. The image data sets of different types were tested with implemented algorithms. From the results obtained, it can be concluded that Arnold Cat Map-Blowfish and Arnold Cat Map-TEA both are very strong techniques for image encryption. The algorithms are implemented and analyzed with different parameters to test the strength of the algorithm and found that Arnold Cat Map-TEA has high NPCR which proves Arnold Cat Map-TEA has strong diffusion mechanism. But in case of UACI analysis, Arnold Cat Map-Blowfish has higher value than Arnold Cat Map-TEA. A high NPCR/UACI score is interpreted as a high resistance to differential attacks. The histogram of encrypted image is completely different from histogram of plain image in both techniques tells that it does not give any hint about original image data and prevents from statistical attacks. The histogram analysis illustrates the confusion and diffusion properties in the encrypted data. Since, Arnold Cat Map-Blowfish takes less time for encryption and decryption, Arnold Cat Map-Blowfish is said to be computationally efficient one compared to Arnold Cat Map-TEA. Both techniques give very small correlation value. But Arnold Cat Map-TEA has less correlation value than Arnold Cat Map-Blowfish. So Arnold Cat Map-TEA is better according to this parameter. It means cipher image produced from Arnold Cat Map-TEA is less correlated with original image than cipher image produced from Arnold Cat Map-Blowfish with its original image.

5.2 Future Recommendation

In this work, only 2D Arnold cat map is used, so in future 3D Arnold cat map can be embedded with block ciphers. Additionally, the cryptanalysis of the analyzed algorithms can also be performed to determine security threat and attack strengths.

References

[1] R. Choudhary and A. JB "Secure Image Transmission and Evaluation of Image Encryption*" IJISET - International Journal of Innovative Science, Engineering & Technology*, Vol. 1 Issue 2, April 2014

[2] M. Panda "Performance Evaluation of Symmetric Encryption Algorithms for Information Security", *International Journal of Advanced Research Trends in Engineering and Technology (IJARTET*) ,Vol. 4, Issue 11, November 2017

[3] P. Wu, "Research on Digital Image Watermark Encryption based on Hyperchaos" Doctor of Philosophy, May 2013

[4] M. Asif *Mushtaque "*Comparative Analysis on Different parameters of Encryption Algorithms for Information Security*"*,School of Computer Science and Engineering, Galgotias University, India *International Journal of Computer Sciences and Engineering*

[5] S. Kumari,J. Chawla "Comparative Analysis on Different Parameters of Encryption Algorithms for Information Security", *International Journal of Innovations & Advancement in Computer Science* IJIACS ISSN 2347 –8616Volume 4, Special Issue May 2015

[6] M. Agrawal, P. Mishra "A Comparative Survey on Symmetric Key Encryption Techniques", *International Journal on Computer Science and Engineering* , Vol. 4 No. 05 May 2012

[7]M. Ebrahim, S. Khan, U. Bin Khalid, "Symmetric Algorithm Survey:A Comarative Analysis", *International Journal of Computer Applications(0975-8887)* volume 61-No. 20, January 2013

[8]M. Ali, B. Younes and A. Jantan, "Image Encryption Using Block-Based Transformation Algorithm" IAENG *International Journal of Computer Science*, 35:1, IJCS_35_1_03

[9] S. Rani M.H, K.L. Sudha , "Design and Implementation of Image Encryption Algorithm Using Chaos*" International Journal of Advanced Computer Research* Volume-4 Number-2 Issue-15 June-2014

[10] S. Lian, J. Sun, Z. Wang ,Security Analysis of A Chaos-based Image Encryption Algorithm, Department of Automation, Nanjing University of Science and Technology Nanjing, Jiangsu 210094, P.R China

[11] D. Bujari and E. Aribas,"Comparative Analysis of Block Cipher Modes of Operation",*International Advanced Researches & Engineering Congress-2017* ,November 2017

[12]J. Ahmad and F. Ahmed ,"Efficiency Analysis and Security Evaluation of Image Encryption Schemes" *International Journal of Video & Image Processing and Network Security* Vol:12 No:04

[13]S. Bora, P. Sen and C. Pradhan Novel Color Image Encryption Technique using Blowfsh and Cross Chaos Map 2015 IEEE

[14] M. Li Ting Liang, Y. jie He "Arnold Transform Based Image Scrambling Method", 3rd International Conference on Multimedia Technology（ICMT 2013)

[15] Z. Yun-peng, Z. Zheng-jun, L. Wei, Nie Xuan, C. Shui-ping, "Digital Image Encryption Algorithm Based on Chaos and Improved DES", Proceedings of the 2009 *IEEE International Conference on Systems, Man, and Cybernetics* October 2009

[16] A. Shah, A. Shah, T. Biradar, "Image Encryption and Decryption using Blowfish Algorithm in MATLAB", *International Journal of Electronics, Electrical and Computational System*, IJEECS,Volume 4, November 2015

[17]A. Jolfaei , A. Mirghadri "Image Encryption Using Chaos and Block Cipher" Vol. 4, No. 1; January 2011

[18] P. Gupta, S. Singh, I. Mangal "Image Encryption Based On Arnold Cat Map and S-Box" Volume 4, Issue 8, August 2014 ISSN: 2277 128X ,*International Journal of Advanced Research in Computer Science and Software Engineering*

[19]K. Brindha, R. Sharma, S. Saini "Use of Symmetric Algorithm for Image Encryption" *International Journal of Innovative Research in Computer and Communication Engineering* Vol. 2, Issue 5, May 2014

[20]E. Hariyanto, R. Rahim "Arnold's Cat Map Algorithm in Digital Image" *International Journal of Science and Research Impact Factor* (2015): 6.391 Volume 5 Issue 10, October 2016

[21]S. Koppu, M. Viswanatham "A novel chaotic image encryption system for color images based Arnold cat map and efficient pixel shuffling" *International Journal Of Pharmacy & Technology* IJPT| June-2016 | Vol. 8 | Issue No.2 | 13353-13361

[22] B. Jyoti Saha , C. Pradhan Kunal Kumar, K. Ajay Kumar Bisoi "Robust Watermarking Technique using Arnold's Transformation and RSA in Discrete Wavelets" 2014 *International Conference on Information Systems and Computer Networks*

[23]A. Bhagat, A. Surve, S. Kalgutkar, A. Waghmare "Chaos Based Image Encryption and Decryption" *International Research Journal of Engineering and Technology* (IRJET) Volume: 03 Issue: 04 | Apr-2016

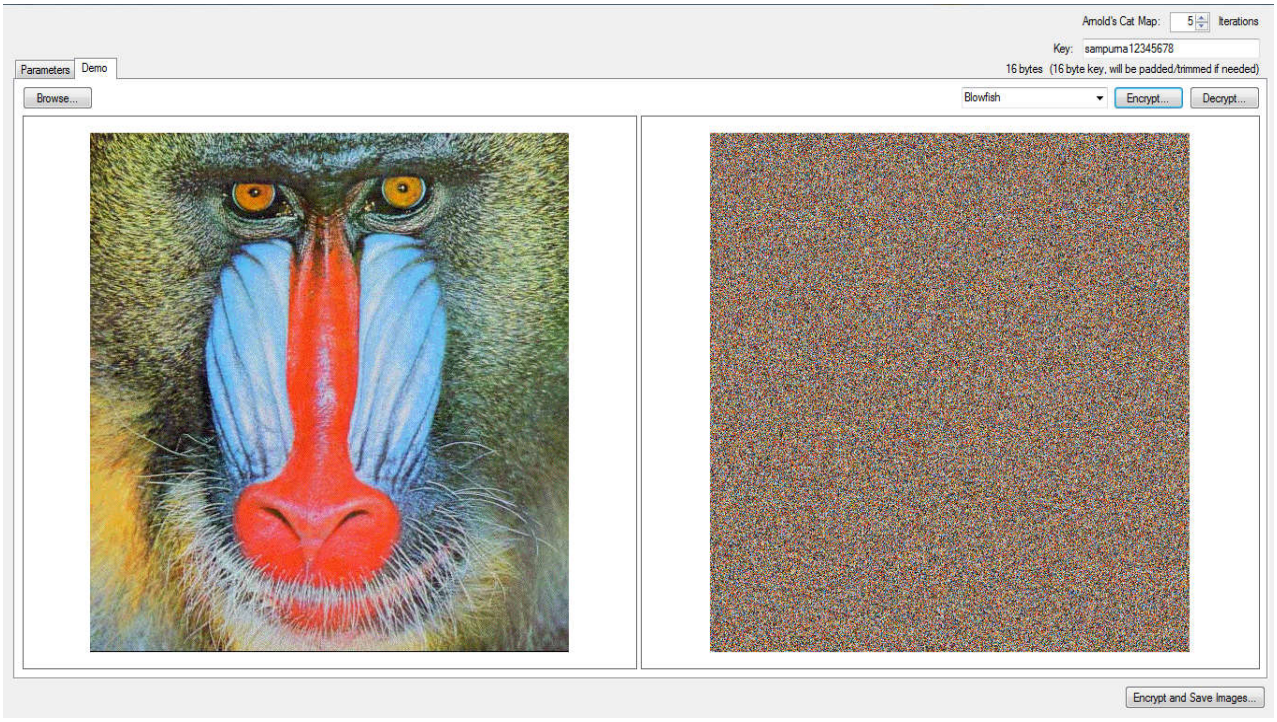
[24] B. Andrews, S. Chapman, S. Dearstyne "Tiny Encryption Algorithm (TEA) Cryptography 4005.705.01 Graduate Team ACD - Final Report"

[25]M. Govind Avasare, V. Vivek Kelkar, "Image Encryption using Chaos Theory", *International Conference on communication, Information & Computing Technology (ICCICT)*, Jan 2015

[26]S. Somaraj and M. Ali Hussain, “Performance and Security Analysis for Image Encryption using Key Image”, *Indian Journal of Science and Technology*, December 2015

Appendix A
Screenshots

1. Demo view



2. Parameters

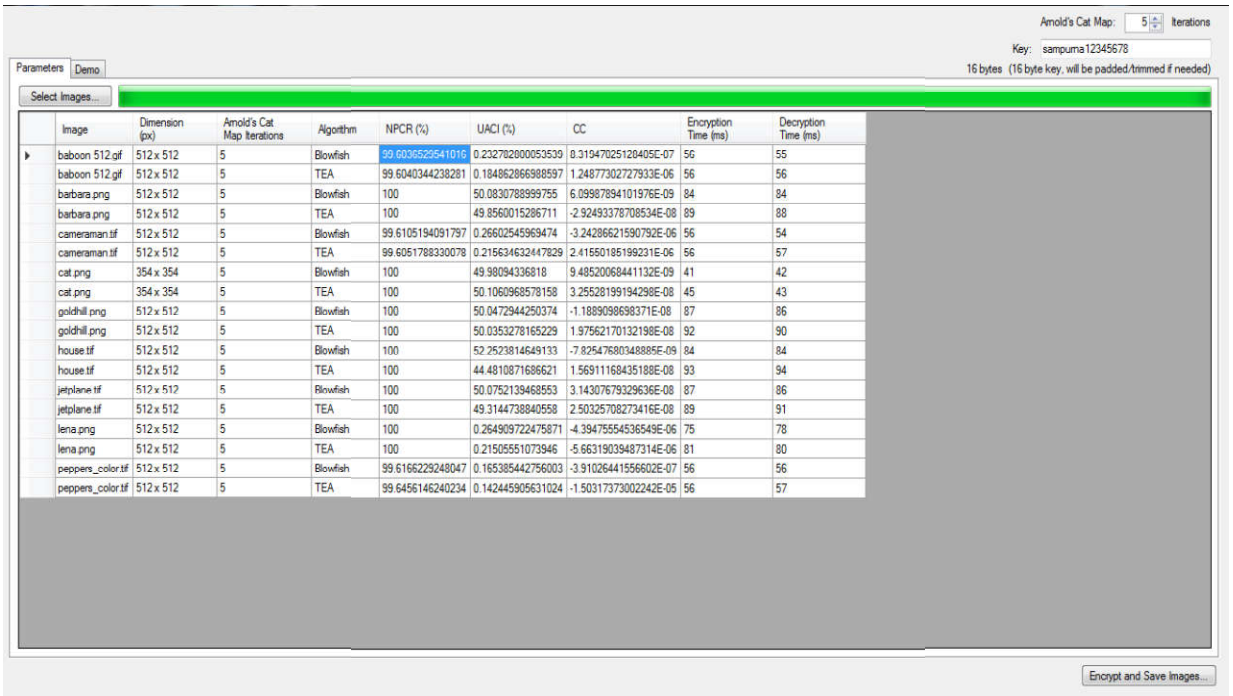
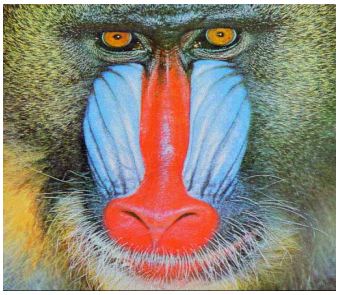


Image	Dimension (px)	Arnold's Cat Map Iterations	Algorithm	NPCR (%)	UACI (%)	CC	Encryption Time (ms)	Decryption Time (ms)
baboon 512.gif	512 x 512	5	Blowfish	99.6036529541016	0.232782000053539	8.31947025128405E-07	56	55
baboon 512.gif	512 x 512	5	TEA	99.6040344238281	0.184862866988597	1.24877302727933E-06	56	56
barbara.png	512 x 512	5	Blowfish	100	50.0830788999755	6.09987894101976E-09	84	84
barbara.png	512 x 512	5	TEA	100	49.8560015286711	-2.92493378708534E-08	89	88
cameraman.tif	512 x 512	5	Blowfish	99.6105194091797	0.26602545969474	-3.24286621590792E-06	56	54
cameraman.tif	512 x 512	5	TEA	99.6051788330078	0.215634632447829	2.41550185199231E-06	56	57
cat.png	354 x 354	5	Blowfish	100	49.98094336818	9.48520068441132E-09	41	42
cat.png	354 x 354	5	TEA	100	50.1060968578158	3.25528199194298E-08	45	43
goldhill.png	512 x 512	5	Blowfish	100	50.0472944250374	-1.18890986983771E-08	87	86
goldhill.png	512 x 512	5	TEA	100	50.0353278165229	1.97562170132198E-08	92	90
house.tif	512 x 512	5	Blowfish	100	52.2523814649133	-7.82547680348885E-09	84	84
house.tif	512 x 512	5	TEA	100	44.4810871686621	1.56911168435188E-08	93	94
jetplane.tif	512 x 512	5	Blowfish	100	50.0752139468553	3.14307679329636E-08	87	86
jetplane.tif	512 x 512	5	TEA	100	49.3144738840558	2.50325708273416E-08	89	91
lena.png	512 x 512	5	Blowfish	100	0.264909722475871	-4.39475554536549E-06	75	78
lena.png	512 x 512	5	TEA	100	0.21505551073946	-5.66319039487314E-06	81	80
peppers_color.tif	512 x 512	5	Blowfish	99.6166229248047	0.165385442756003	-3.91026441556602E-07	56	56
peppers_color.tif	512 x 512	5	TEA	99.6456146240234	0.142445905631024	-1.50317373002242E-05	56	57

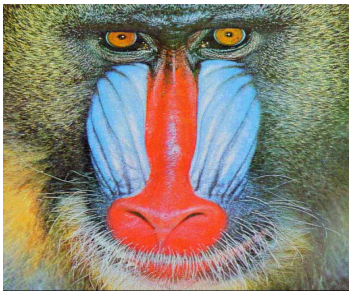
i. Baboon.gif

a. Arnold Cat Map-Blowfish Analysis



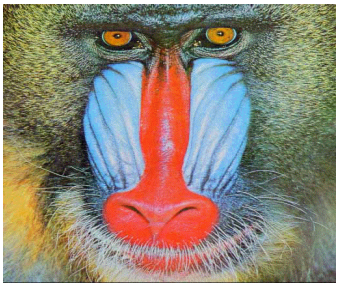
Original Image

Encrypted Image



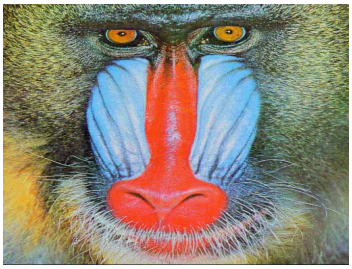
Decrypted Image

b. Arnold Cat Map-TEA Analysis



Original Image

Encrypted Image



Decrypted Image

ii. Barbara.png

a. Arnold Cat Map-Blowfish Analysis

Original Image

Encrypted Image

Decrypted Image

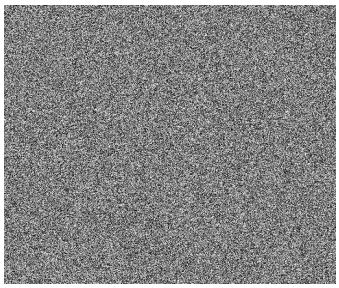
b. Arnold Cat Map-TEA Analysis



Original Image Encrypted Image Decrypted Image

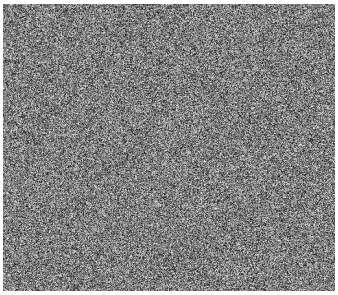
iii. Cameraman.tif

a. Arnold Cat Map-Blowfish Analysis


Original Image Encrypted Image Decrypted Image

b. Arnold Cat Map-TEA Analysis


Original Image Encrypted Image Decrypted Image

iv. Cat.png

a. Arnold Cat Map-Blowfish Analysis



Original Image Encrypted Image Decrypted Image

b. Arnold Cat Map-TEA Analysis



Original Image Encrypted Image Decrypted Image

v. Goldhill.png

a. Arnold Cat Map-Blowfish Analysis



Original Image Encrypted Image Decrypted Image

b. Arnold Cat Map-TEA Analysis


Original Image


Encrypted Image


Decrypted Image

v. House.tif

a. Arnold Cat Map-Blowfish Analysis


Original Image


Encrypted Image


Decrypted Image

b. Arnold Cat Map-TEA Analysis


Original Image


Encrypted Image


Decrypted Image

vi. Jetplane.tif

a. Arnold Cat Map-Blowfish Analysis



Original Image Encrypted Image Decrypted Image

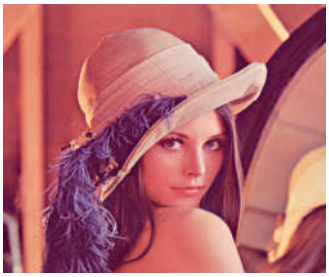
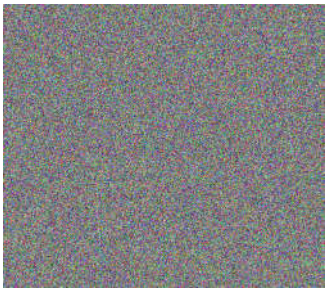
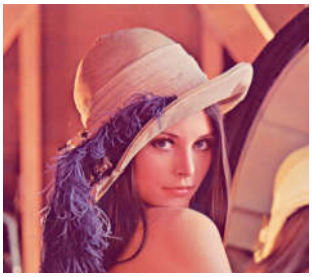
b. Arnold Cat Map-TEA Analysis



Original Image Encrypted Image Decrypted Image

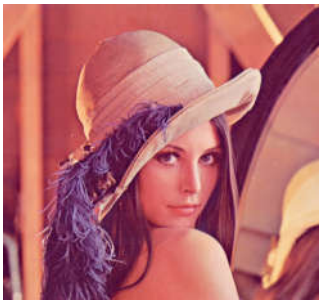
vii. Lena.png

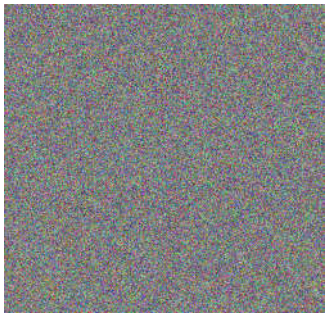
a. Arnold Cat Map-Blowfish Analysis

Original Image Encrypted Image Decrypted Image

b. Arnold Cat Map-TEA Analysis


Original Image

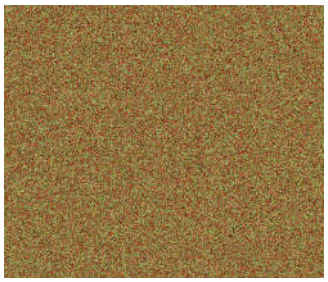

Encrypted Image


Decrypted Image

viii. Peppers.tif

a. Arnold Cat Map-Blowfish Analysis

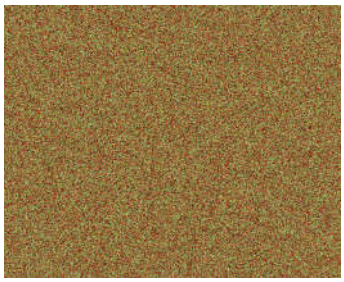

Original Image


Encrypted Image


Decrypted Image

b. Arnold Cat Map-TEA Analysis


Original Image


Encrypted Image


Decrypted Image

Appendix B

Source Code

ArnoldsCatMap.cs

```
using System;

namespace ThesisCode
{
    public static class ArnoldsCatMap
    {
        // n is the size of square bitmap

        // sx and sy are source pixel coordinates
        // dx and dy are destination pixel coordinates

        private static void CopyPixel(byte[] input, byte[] output,
            int bytesPerPixel, int n,
            int sx, int sy, int dx, int dy)
        {
            var sourceIndexStart = bytesPerPixel * (sx * n + sy);
            var destinationIndexStart = bytesPerPixel * (dx * n + dy);
            for (var offset = 0; offset < bytesPerPixel; offset++)
            {
                output[destinationIndexStart + offset] =
                    input[sourceIndexStart + offset];
            }
        }

        public static byte[] Iterate(byte[] input, int bytesPerPixel, int n)
        {
            var output = new byte[input.Length];
            Array.Copy(input, output, input.Length);

            for (var sx = 0; sx < n; sx++)
            {
                for (var sy = 0; sy < n; sy++)
                {
                    var dx = (sx + sy) % n;
                    var dy = (sx + 2 * sy) % n;
                    CopyPixel(input, output, bytesPerPixel, n, sx, sy, dx, dy);
                }
            }

            return output;
        }

        public static byte[] ReverseIterate(byte[] input, int bytesPerPixel, int n)
        {
            var output = new byte[input.Length];
            Array.Copy(input, output, input.Length);

            for (var sx = 0; sx < n; sx++)
            {
                for (var sy = 0; sy < n; sy++)
                {
                    var dx = (2 * sx - sy) % n;
                    dx = (dx < 0) ? dx + n : dx;
```

```
                    var dy = (-sx + sy) % n;
                    dy = (dy < 0) ? dy + n : dy;
                    CopyPixel(input, output, bytesPerPixel, n, sx, sy, dx, dy);
                }
            }

            return output;
        }
    }
}
```

BlockCipher.cs

```
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.Crypto.Parameters;
using System;
using System.Text;

namespace ThesisCode
{
    public static class BlockCipher
    {

        public static byte[] Process(
            IBlockCipher engine,
            string key,
            byte[] input,
            bool isEncryptOperation)
        {
            var cipher = new BufferedBlockCipher(engine);
            cipher.Init(isEncryptOperation, ToKeyParameter16(key));
            var output = new byte[input.Length];
            var length = cipher.ProcessBytes(input, 0, input.Length, output, 0);
            cipher.DoFinal(output, length);
            return output;
        }

        private static ICipherParameters ToKeyParameter16(string key)
        {
            var keyByte = Encoding.UTF8.GetBytes(key);

            const int MAX_LENGTH = 16;
            var keyByte16 = new byte[MAX_LENGTH];
            Array.Copy(keyByte, keyByte16,
                (MAX_LENGTH < key.Length) ? MAX_LENGTH : key.Length);

            return new KeyParameter(keyByte16);
        }
    }
}
```

CombinedCipher.cs

```
using Org.BouncyCastle.Crypto;
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;

namespace ThesisCode
{
```

```
    public class CombinedCipher
    {

        public static byte[] Encrypt(int iterations,
            IBlockCipher blockCipher, int bytesPerPixel,
            byte[] input, string key, int bitmapSize)
        {
            for (int i = 1; i <= iterations; i++)
            {
                input = ArnoldsCatMap.Iterate(input, bytesPerPixel, bitmapSize);
            }

            return BlockCipher.Process(
                blockCipher,
                key,
                input,
                true
            );
        }

        public static byte[] Decrypt(int iterations,
            IBlockCipher blockCipher, int bytesPerPixel,
            byte[] input, string key, int bitmapSize)
        {
            input = BlockCipher.Process(
                blockCipher,
                key,
                input,
                false
            );

            for (int i = 1; i <= iterations; i++)
            {
                input = ArnoldsCatMap.ReverseIterate(
                    input, bytesPerPixel, bitmapSize);
            }
            return input;
        }

    }
}
```

Parameter.cs

```
using Org.BouncyCastle.Crypto;
using System;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;

namespace ThesisCode
{
    public static class Parameter
    {
```

```
public static double NumberOfPixelChangeRate(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap bitmap,
    string key)
{
    var source1 = bitmap;
    var c1 = CombinedCipher.ProcessImage(
        iterations, blockCipher, source1, key, true);

    var source2 = bitmap.ChangeOnePixel();
    var c2 = CombinedCipher.ProcessImage(
        iterations, blockCipher, source2, key, true);

    var sum = 0L;
    for (var i = 0; i < c1.Width; i++)
    {
        for (var j = 0; j < c1.Height; j++)
        {
            sum += (c1.GetPixel(i, j).ToArgb()
                == c2.GetPixel(i, j).ToArgb()) ? 0 : 1;
        }
    }
    var t = c1.Width * c1.Height;
    return (double)sum / t * 100;
}

public static double UnifiedAverageChangeIntensity(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap bitmap,
    string key)
{
    var source1 = bitmap;
    var c1 = CombinedCipher.ProcessImage(
        iterations, blockCipher, source1, key, true);

    var source2 = bitmap.ChangeOnePixel();
    var c2 = CombinedCipher.ProcessImage(
        iterations, blockCipher, source2, key, true);

    var sum = 0L;
    for (var i = 0; i < c1.Width; i++)
    {
        for (var j = 0; j < c1.Height; j++)
        {
            sum += Math.Abs(c1.GetPixel(i, j).ToArgb()
                - c2.GetPixel(i, j).ToArgb());
        }
    }
    var t = c1.Width * c1.Height;
    var f = int.MaxValue;
    return (double)sum / t * 100 / f;
}

public static double CorrelationCoefficient(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap bitmap,
    string key)
{
    var source = bitmap;
```

```
    Bitmap c = CombinedCipher.ProcessImage(
        iterations, blockCipher, source, key, true);
    var x = source.ToArgbArray();
    var y = c.ToArgbArray();
    var cv = Statistics.Covariance(x, y);
    var sdX = Statistics.StandardDeviation(x);
    var sdY = Statistics.StandardDeviation(y);
    return cv / (sdX * sdY);
}

public static long EncryptionTime(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap bitmap,
    string key,
    out Bitmap outputBitmap)
{
    return ProcessImage(iterations, blockCipher,
        bitmap, key, false, out outputBitmap);
}

public static long DecryptionTime(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap bitmap,
    string key)
{
    Bitmap outputBitmap = null;
    return ProcessImage(iterations, blockCipher,
        bitmap, key, false, out outputBitmap);
}

private static long ProcessImage(
    int iterations,
    IBlockCipher blockCipher,
    Bitmap source,
    String key,
    bool isEncryptOperation,
    out Bitmap outputBitmap)
{
    var bitmap = (Bitmap)source.Clone();
    var rectangle = new Rectangle(0, 0, bitmap.Width, bitmap.Height);
    var bmpData = bitmap.LockBits(rectangle,
        ImageLockMode.ReadWrite, bitmap.PixelFormat);

    // Get the address of the first line.
    var ptr = bmpData.Scan0;

    var bytes = Math.Abs(bmpData.Stride) * bitmap.Height;
    var input = new byte[bytes];

    // Copy the RGB values into the array.
    Marshal.Copy(ptr, input, 0, bytes);

    byte[] output = null;
    int bytesPerPixel = bitmap.bytesPerPixel();

    var watch = new Stopwatch();
    watch.Start();
    if (isEncryptOperation)
        output = CombinedCipher.Encrypt(iterations,
            blockCipher, bytesPerPixel, input, key, bitmap.Width);
```

```
            else
                output = CombinedCipher.Decrypt(iterations,
                    blockCipher, bytesPerPixel, input, key, bitmap.Width);
            watch.Stop();

            // Copy the RGB values back to the bitmap
            Marshal.Copy(output, 0, ptr, bytes);

            bitmap.UnlockBits(bmpData);

            outputBitmap = bitmap;

            return watch.ElapsedMilliseconds;
        }
    }
}
```

Statistics.cs

```
using System;

namespace ThesisCode
{
    public static class Statistics
    {

        public static double Mean(int[] x)
        {
            var sum = 0L;
            for (var i = 0; i < x.Length; i++)
            {
                sum += x[i];
            }
            return (double)sum / x.Length;
        }

        public static double Variance(int[] x)
        {
            var mean = Mean(x);
            var sum = 0.0;
            for (var i = 0; i < x.Length; i++)
            {
                var difference = x[i] - mean;
                sum += difference * difference;
            }
            return sum / (x.Length - 1);
        }

        public static double StandardDeviation(int[] x)
        {
            return Math.Sqrt(Variance(x));
        }

        public static double Covariance(int[] x, int[] y)
        {
            var sumX = 0L;
            var sumY = 0L;
            var sumXY = 0L;
            for(int i = 0; i < x.Length; i++)
```

```
            {
                sumX += x[i];
                sumY += y[i];
                sumXY += x[i] * y[i];
            }
            return (sumXY - sumX * sumY / (double)x.Length) / (x.Length - 1);
        }
    }
}
```