Image Encryption Using Matrix Transmutation

Mini Project Report

Submitted in the partial fulfillment for the award of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:-

Mahendra Kumar (2012ECS53)
Rahul Mishra (2012ECS01)
Sumit Jha (2012ECS11)

TO

SHRI MATA VAISHNO DEVI UNIVERSITY
J&K, INDIA
DEC 2015

ACKNOWLEDGEMENT

It is a moment of great pleasure and immense satisfaction for us to express our deepest sense of gratitude and indebtedness to all the people who have contributed in making of our minor project a rich experience.

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to **Mr. Sanjay Sharma** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to express our gratitude towards members of **Shri Mata Vaishno Devi University** for their kind co-operation and encouragement which help us in completion of this project.

Finally, we would like to thank all our Teachers, Friends and Family Members who have supported us throughout and enlightened us to take the right path and reach there. The days we spent in this institute will be cherished forever and also be reckoned as a guiding factor in our career.

Mahendra Kumar
Rahul Mishra
Sumit Jha
B. Tech. 7th Semester (SCSE)

ABSTRACT

With the ever-increasing growth of multimedia applications, security is an important issue in the communication and storage of images, and encryption is one the ways to ensure security. Image encryption techniques try to convert original image to another image that is hard to understand; to keep the image confidential between users, in other word, it is essential that nobody could get to know the content without a key for decryption. Furthermore, special and reliable security in Storage and transmission of digital images is needed in many applications, such as cable-TV, online personal photograph album, medical imaging systems, military image communications and confidential video conferences, etc. In order to fulfill such a task, many image encryption methods have been proposed.

The two main characteristics that identify and differentiate one encryption algorithm from another are its ability to secure the protected data against attacks and its speed and efficiency in doing so. The effort has been put on to perform a comparison between some most commonly used encryption algorithms and to use one of them to do encryption of an image.

Matrix Transmutation is one of the most widely used technique for image encryption, it's because it offers a great amount of confusion and diffusion during encryption process hence, generates a highly encrypted image, thereby increasing data security which is our main concern. It will be used as an addition to the encryption algorithm and to study the effects.

TABLE OF CONTENTS

Page intentionally left Blank

INTRODUCTION

PURPOSE

The main purpose of the project is to compare various popular encryption techniques and compare them for encryption and decryption efficiency and the using the best available encryption scheme we encrypt the desired image and study it's pixel histogram for confusion and diffusion properties, furthermore we use this encrypted image and apply the matrix transmutation over and see if the strength of the encryption is strong enough or not.

The proposed method of Matrix Transmutation used confusion schemes for scrambling the positions of pixels of the colored images in two stages firstly, by dividing the original image to 64 blocks and rotate each one in clockwise with 90° . Secondly, using 2D Arnold Cat mapping to apply more scrambling of blocked image pixels. Shuffling mechanism combined with diffusion mechanism for encrypting the scrambling image by changing the gray values of the image pixels.

SYSTEM OVERVIEW

Our Encryption application will have the following functionalities:

1. User will implement different encryption schemes in MATLAB environment.
2. Then a comparison is drawn out between different schemes based on various criterion and the best one is chosen among them.
3. After that chosen scheme used to implement image encryption and histogram specification is carried over the image to study the confusion and diffusion in the encrypted image.

4. Apart from this matrix transmutation methodology is used to encrypted the previously encrypted image and again a comparison and conclusion is drawn out
   Whether this methodology is more efficient and effective.

The only requirement for implementation of the encryption schemes is MATLAB or Octave (equivalent Open Source Alternative) software. There is requirement for Internet connection, it's complete offline application environment although it can be run on any appropriate online Environment.

PROBLEM STATEMENT

Cryptography is usually referred to as "the study of secret", while nowadays is most attached to the definition of encryption. Encryption is the process of converting plain text "unhidden" to a cryptic text "hidden" to secure it against data thieves. This process has another part where cryptic text needs to be decrypted on the other end to be understood. Figure given below shows the simple flow of commonly used encryption algorithms.

In our project we are drawing out comparison of various encryption schemes and choosing one of the best possible scheme for image encryption. Every security system must provide a bundle of security functions that can assure the secrecy of the system. These functions are usually referred to as the goals of the security system. These goals can be listed under the following five main categories:

Authentication: This means that before sending and receiving data using the system, the receiver and sender identity should be verified.

Secrecy or Confidentiality: Usually this function (feature) is how most people identify a secure system. It means that only the authenticated people are able to interpret the message (date) content and no one else.

Integrity: Integrity means that the content of the communicated data is assured to be free from any type of modification between the end points (sender and receiver). The basic form of integrity is packet check sum in IPv4 packets.

Non-Repudiation: This function implies that neither the sender nor the receiver can falsely deny that they have sent a certain message.

Service Reliability and Availability: Since secure systems usually get attacked by intruders, which may affect their availability and type of service to their users. Such systems should provide a way to grant their users the quality of service they expect.

GOAL OF THIS PROJECT

Our basic goal is to study different encryption schemes and carry out comparisons between most commonly used encryption algorithm and come out with the best encryption methodology that will be further used to study encryption procedure alone and then with Matrix Transmutation as an additional encryption technique.

The project also include to study the effect of applying any one of the Matrix Transmutation technique in various ways.

REQUIRED SPECIFICATIONS

USER CHARACTERISTICS

There is no as such User Characteristics required to understand and implement the schemes .The only prerequisites are basic understanding of cryptography and familiarity with MATLAB/Octave Programming Environment.

FUNCTIONAL REQUIREMENTS

The Observations were conducted using 2.30GHz Intel i3 2nd Generation 64bit processor with 4GB RAM. The simulation program is compiled using the MATLAB in Windows 10 System Environment. The experiments will be performed couple of times to assure that the results are consistent and are valid to compare the different algorithms.

CONSTRAINTS AND ASSUMPTIONS

In order to evaluate the performance of the compared algorithms, the parameters that the algorithms must be tested for must be determined.

Since the security features of each algorithm as their strength against cryptographic attacks is already known. The chosen factor here to determine the performance is the algorithm's speed to encrypt/decrypt data blocks of various sizes.

CRYPTOGRAPHY: OVERVIEW

Cryptography is usually referred to as "the study of secret", while nowadays is most attached to the definition of encryption. Encryption is the process of converting plain text "unhidden" to a cryptic text "hidden" to secure it against data thieves. This process has another part where cryptic text needs to be decrypted on the other end to be understood.

As defined in RFC 1, cryptographic system is "a set of cryptographic algorithms together with the key management processes that support use of the algorithms in some application context." This definition defines the whole mechanism that provides the necessary level of security comprised of network protocols and data encryption algorithms.

BLOCK CIPHERS & STREAM CIPHERS

One of the main categorization methods for encryption techniques commonly used is based on the form of the input data they operate on. The two types are Block Cipher and Stream Cipher.

Block Cipher

In Block Cipher method data is encrypted and decrypted if data is in from of blocks. In its simplest mode, you divide the plain text into blocks which are then fed into the cipher system to produce blocks of cipher text.

ECB (Electronic Codebook Mode) is the basic form of block cipher where data blocks are encrypted directly to generate its correspondent ciphered blocks.

Stream Cipher

Stream cipher functions on a stream of data by operating on it bit by bit. Stream cipher consists of two major components: a key stream generator, and a mixing

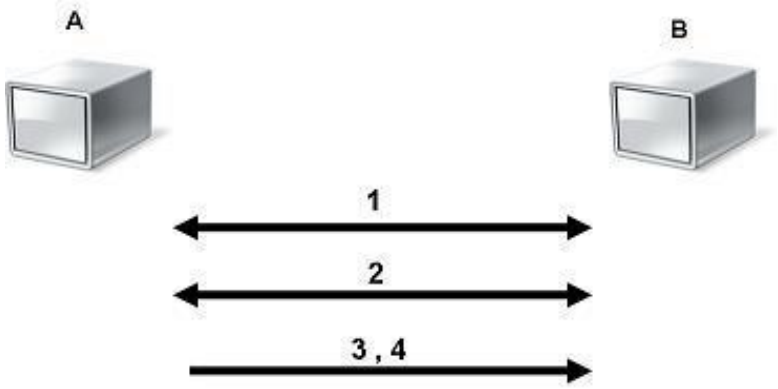
function. Mixing function is usually just an XOR function, while key stream generator is the main unit in stream cipher encryption technique.

SYMMETRIC AND ASYMMETRIC ENCRYPTION

Data encryption procedures are mainly categorized into two categories depending on the type of security keys used to encrypt/decrypt the secured data. These two categories are: Asymmetric and Symmetric encryption techniques.

Symmetric Encryption

In this type of encryption, the sender and the receiver agree on a secret (shared) key. Then they use this secret key to encrypt and decrypt their sent messages.



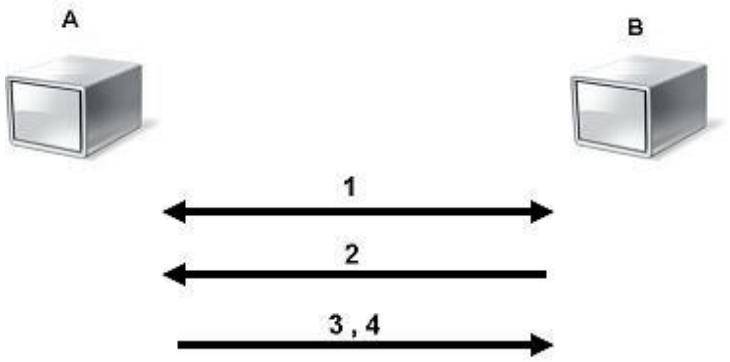
1- A and B agree on a cryptosystem.
2- A and B agree on the key to be used.
3- A encrypts messages using the shared key
4- B decrypts the ciphered messages using the shared key.

The main concern behind symmetric encryption is how to share the secret key securely between the two peers. If the key gets known for any reason, the whole system collapses.

Asymmetric Encryption

Asymmetric encryption is the other type of encryption where two keys are used. One being public key i.e. for the encryption purpose and the other being the decryption key pair for the public key, also termed private key.

This unique feature of public key encryption makes it mathematically more prone to attacks. Moreover, asymmetric encryption techniques are almost 1000 times slower than symmetric techniques, because they require more computational processing power.



1- A and B agree on a cryptosystem.
2- B sends its public key to A.
3- A encrypts messages using the negotiated cipher and B's public key.
4- B decrypts the ciphered messages using its private key and the negotiated cipher.

COMPARED ALGORITHMS

DES: (Data Encryption Standard), was the first encryption standard to be recommended by NIST (National Institute of Standards and Technology). It is based on the IBM proposed algorithm called Lucifer. DES became a standard in 1974. Since that time, many attacks and methods recorded that exploit the weaknesses of DES, which made it an insecure block cipher.

3DES: As an enhancement of DES, the3DES (Triple DES) encryption standard was proposed. In this standard the encryption method is similar to the one in original DES but applied 3 times to increase the encryption level. But it is a known fact that 3DES is slower than other block cipher methods.

AES: (Advanced Encryption Standard), is the new encryption standard recommended by NIST to replace DES. Rijndael (pronounced Rain Doll) algorithm was selected in 1997 after a competition to select the best encryption standard. Brute force attack is the only effective attack known against it, in which the attacker

tries to test all the characters combinations to unlock the encryption. Both AES and DES are block ciphers.

Blowfish: It is one of the most common public domain encryption algorithms provided by Bruce Schneier - one of the world's leading cryptologists, and the president of Counterpane Systems, a consulting firm specializing in cryptography and computer security.

SIMULATION PROCEDURE

By considering different sizes of data blocks (0.5MB to 20MB) the algorithms are evaluated in terms of the time required to encrypt and decrypt the data block. All the implementations were exact to make sure that the results will be relatively fair and accurate.

Algorithm	Key Size (Bits)	Block Size (Bits)
DES	64	64
3DES	192	64
Rijndael	256	128
Blowfish	448	64

These were the predefined Key Sizes and respective Block sizes for the various algorithms chosen for comparison.

After carrying out the simulation procedure for the above mentioned encryption schemes, following data was observed as in the following table.

Input Size (bytes)	DES	3DES	AES	BF
20,527	2	7	4	2
36,002	4	13	6	3
45,911	5	17	8	4
59,852	7	23	11	6
69,545	9	26	13	7
137,325	17	51	26	14
158,959	20	60	30	16
166,364	21	62	31	17
191,383	24	72	36	19
232,398	30	87	44	24
Average Time	14	42	21	11
Bytes/sec	7,988	2,663	5,320	10,167

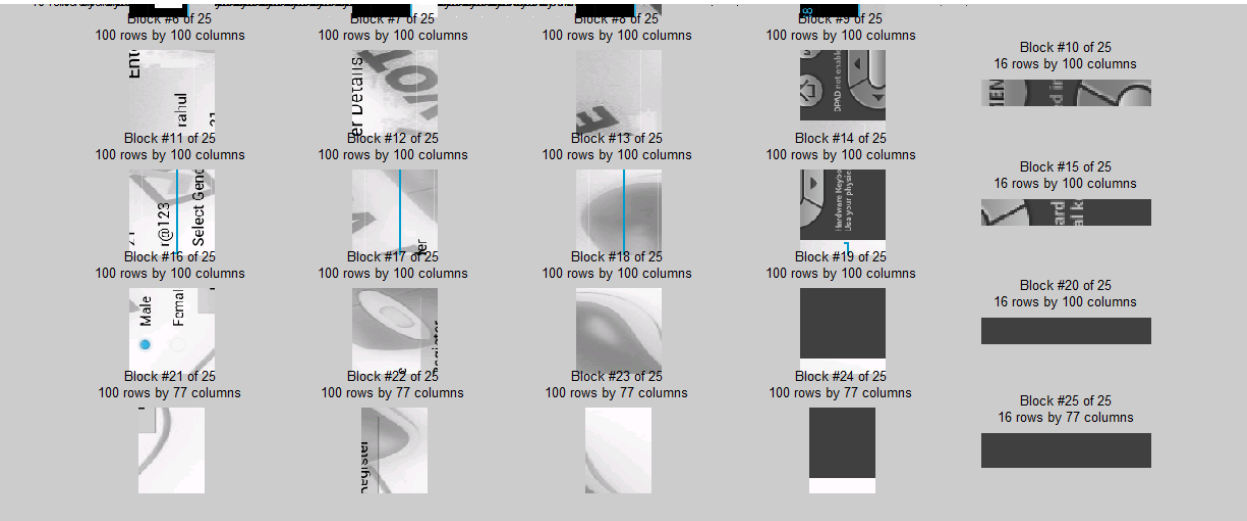
Matrix Transmutation: Applied Approach

The main aim of image encryption is to provide an easy and inexpensive scheme of encryption and decryption of digital data to all authorized users.

Image Shuffling

RGB image scrambling is useful to disturb the correlation between the neighboring pixels by changing the position of it and not changes of pixel value so the histogram of image is stable at the beginning of the encryption and the decryption process. Shuffling of image pixels is done in three steps:

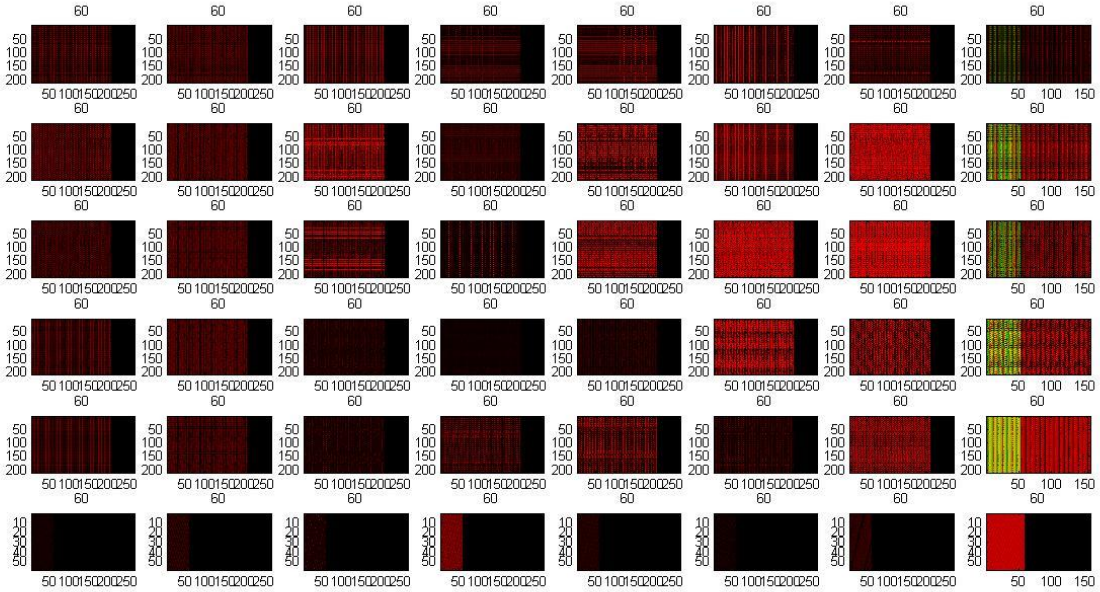
First step: Divide image in quadrant and rotate each one with 90 in clockwise direction.
Second step: Divide each quadrant into four sub-quadrants and rotate them with 90 and the result is 16 block of image.
Third step: Divide each block again to four blocks and rotate each one with 90 and the result is 64 blocks of image.



Arnold Cat Map System

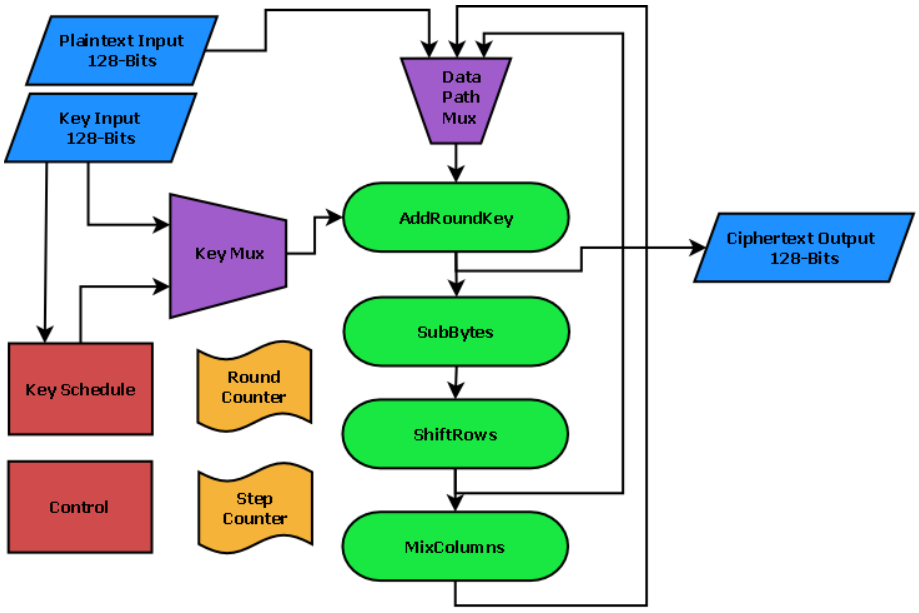
Arnold's Cat Map transformation use for scrambling the pixels of color image and to perform extra security of cipher system. The 2D Arnolds cat Map does not change the intensity value of the image; it only shuffles the data of image. After applying 2D Arnold cat transformation for several iterations, the relationship

between the neighboring pixels is entirely destroy and the original image seems deformation and meaningless. Actually, for iterating it to many times it will return to original look. This mean that Arnold cat map is a periodic transform. After image shuffling the statistical features are same for encrypt image and original image to increase the security of encryption system.

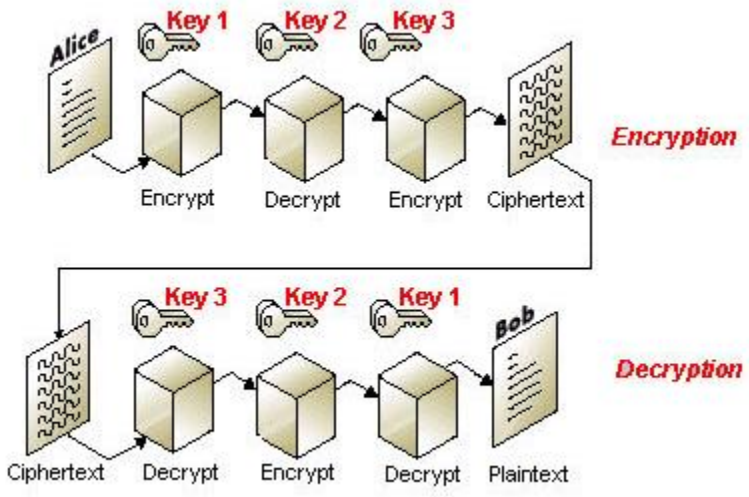


DESIGN

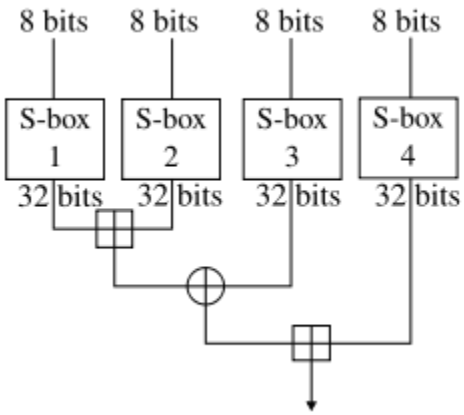
DATA FLOW DIAGRAMS



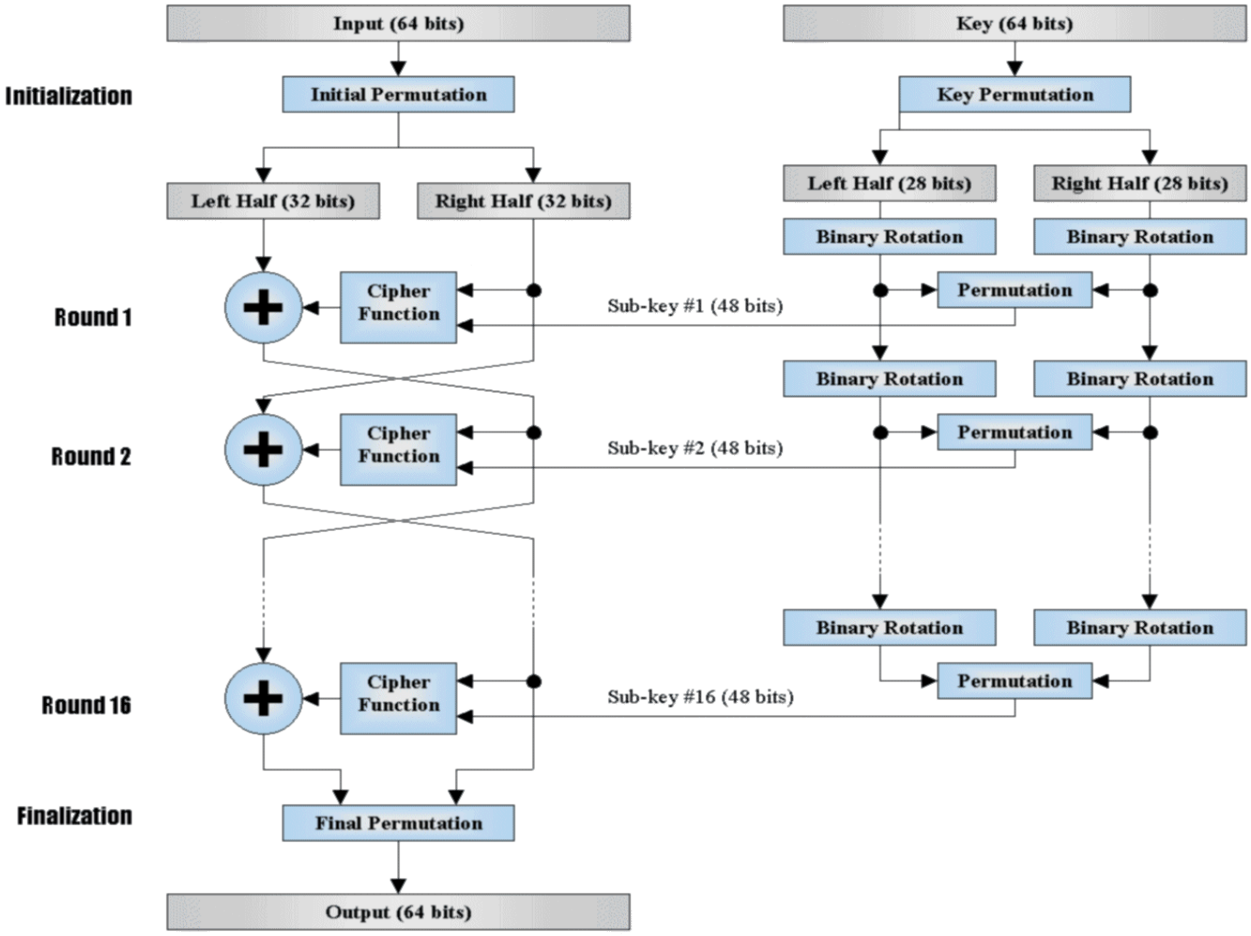
AES Algorithm



3DES Algorithm



Blowfish Algorithm



DES Algorithm

CODING

ImageEncryptionGui.m :

```
function varargout = ImageEncryptionGui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ImageEncryptionGui_OpeningFcn, ...
                   'gui_OutputFcn',  @ImageEncryptionGui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function ImageEncryptionGui_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
axes(handles.axes4)
BackGr = imread('leaf.jpg');
 imshow(BackGr);
guidata(hObject, handles);
clear all;
clc;
global Img;
global EncImg;
```

```
global DecImg;
function varargout = ImageEncryptionGui_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function pushbutton1_Callback(hObject, eventdata, handles)

global Img;
global key;
X = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png','pick a jpge file');
Img = imread(X);
  axes(handles.axes1)
  imshow(Img);

[n m k] = size(Img);
  key = keyGen(n*m);
guidata(hObject, handles);
function pushbutton2_Callback(hObject, eventdata, handles)
global Img ;
global EncImg;
global key;
EncImg = imageProcess(Img,key);
axes(handles.axes2)
imshow(EncImg);
imwrite(EncImg,'Encoded.jpg','jpg');
guidata(hObject, handles);

function pushbutton3_Callback(hObject, eventdata, handles)
global DecImg;
global EncImg;
global key;
DecImg = imageProcess(EncImg,key);
axes(handles.axes3);
imshow(DecImg);
imwrite(DecImg,'Decoded.jpg','jpg');
guidata(hObject, handles);
```

imageProcess.m :

```
1.  function [proImageOut] = imageProcess(ImgInp,key)
2.
3.
4.  [n m k] = size(ImgInp);
5.  % key =cell2mat(struct2cell( load('key5.mat')));
6.  % key = keyGen(n*m);
7.  for ind = 1 : m
8.      Fkey(:,ind) = key((1+(ind-1)*n) : (((ind-1)*n)+n));
9.  end
10.len = n;
11.bre = m;
12.for ind = 1 : k
13.    Img = ImgInp(:,:,ind);
14.for ind1 = 1 : len
15.    for ind2 = 1 : bre
16.        proImage(ind1,ind2) = bitxor(Img(ind1,ind2),Fkey(ind1,ind2));
17.    end
18.end
19.proImageOut(:,:,ind) = proImage(:,:,1);
20.end
21.% figure,imshow(proImageOut);
22.return;
```

keyGen.m :

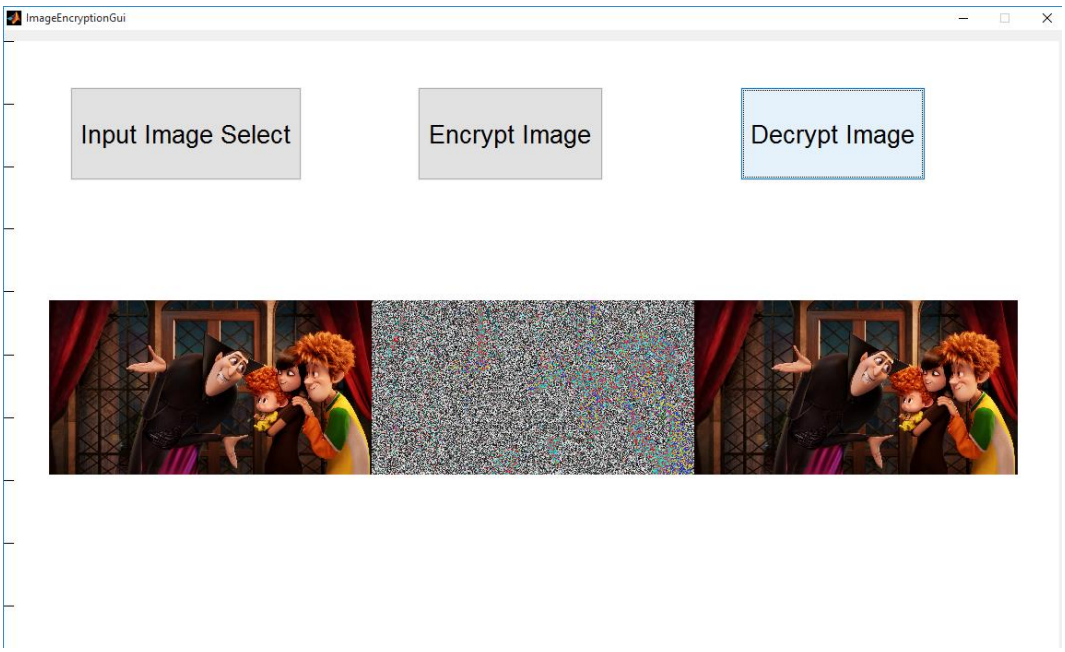
```
function [key] = keyGen(n)
n = n*8;
% n = 2048*2048*16;
% n = 24 * 24 * 8;
bin_x = zeros(n,1,'uint8');
bin_x_N_Minus_1 =  0.300001;
x_N = 0;
tic
for ind = 2 : n
   x_N = 1 - 2* bin_x_N_Minus_1 * bin_x_N_Minus_1;
    if (x_N > 0.0)
       bin_x(ind-1) = 1;
   end
    bin_x_N_Minus_1 =  x_N;

end
toc
% save bin_sec bin_x;
key = zeros(n/8,1,'uint8');
for ind1 = 1 : n/8

   for ind2 = 1 : 8
   key(ind1) = key(ind1) + bin_x(ind2*ind1)* 2 ^ (ind2-1);
   end

end
```

Fig. Image Encryption using Blowfish Algorithm



Build Histogram

```
1. input = imread('C:\Users\MAHE_MSK\Desktop\ImageEncryptionGUI\encoded.jpg')
   ;
2. Red = input(:,:,1);
3. Green = input(:,:,2);
4. Blue = input(:,:,3);
5. [yRed, x] = imhist(Red);
6. [yGreen, x] = imhist(Green);
7. [yBlue, x] = imhist(Blue);
8. figure(1);plot(x, yRed, 'Red', x, yGreen, 'Green', x, yBlue, 'Blue');
9. input=rgb2gray(input);
10.figure(2);imhist(input);
```

Fig. Histogram of Original Image

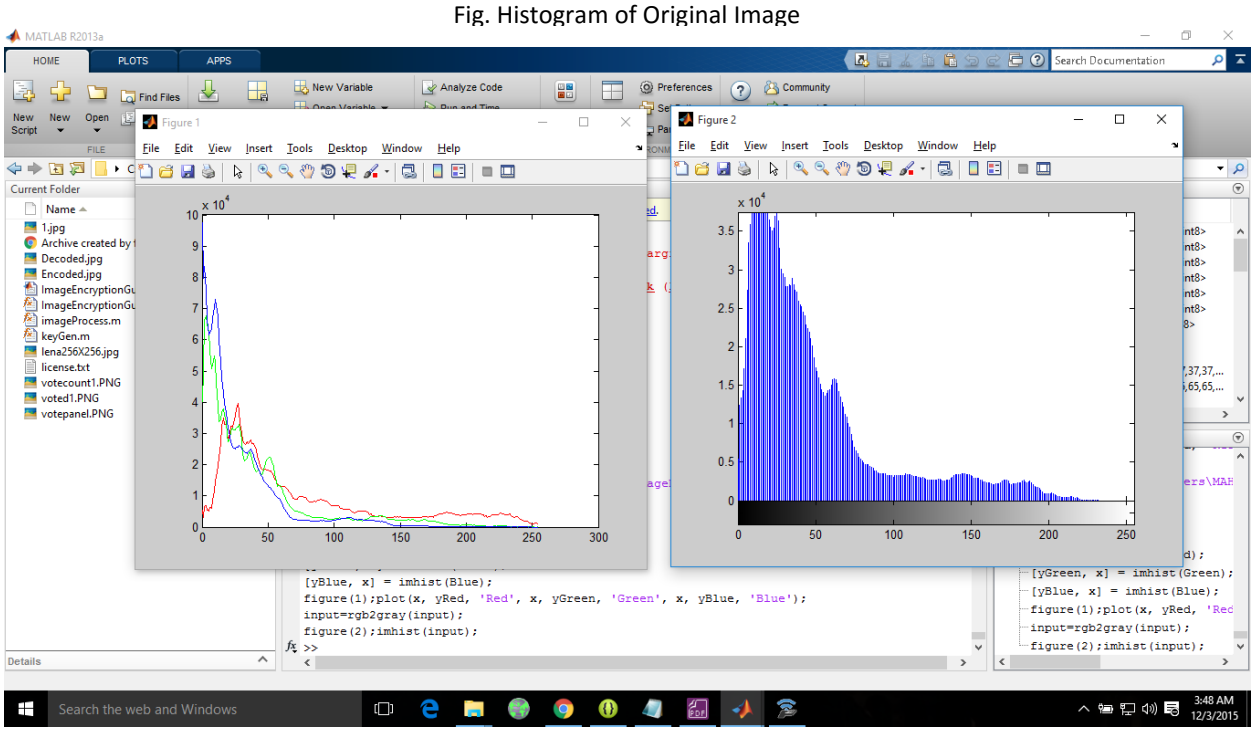
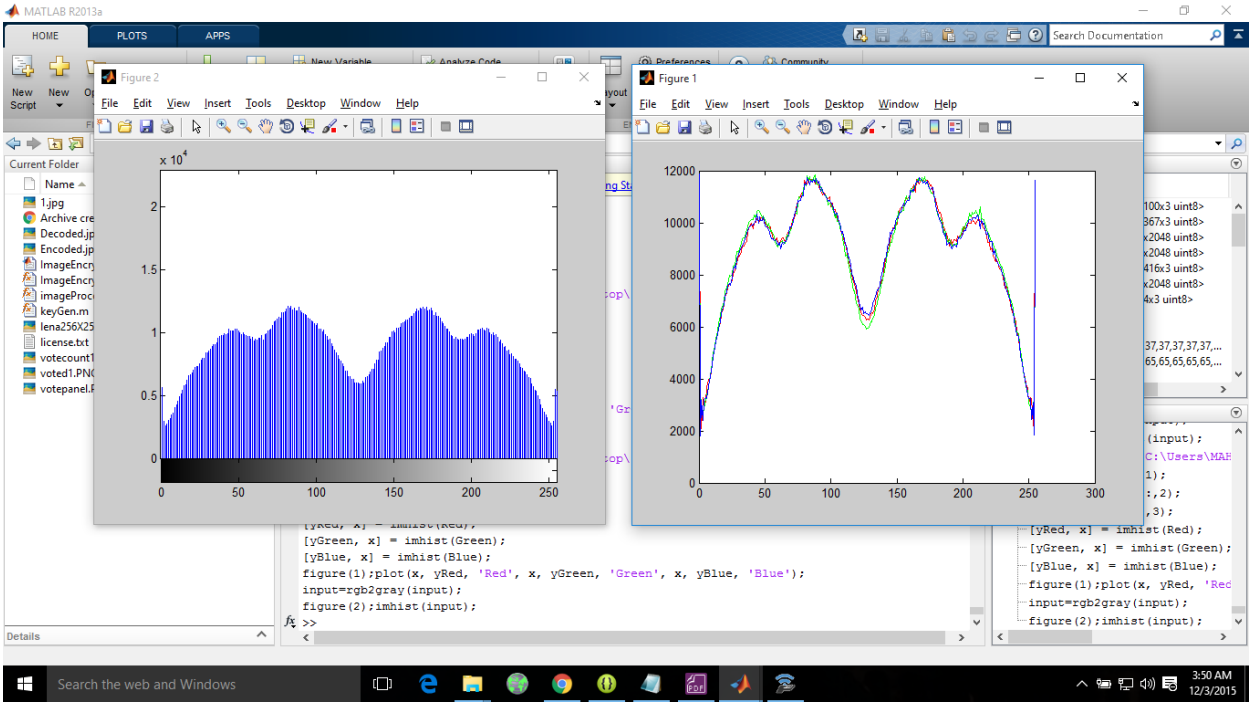


Fig. Histogram of Encoded Image(Applying Blowfish Algorithm)



Arnold's Cat Map Algorithm

```
1.  function X = catmap(Y)
2.  % applies Arnold's Cat Map to the image Y to produce image X
3.
4.  p = size(Y,1);      % get the number of pixels on each side
5.  X = zeros(size(Y)); % make space for X (all zeros to start)
6.  for i = 1:p         % loop through all the pixels
7.     for j = 1:p
8.        newi = mod(((i-1) + (j-1)),p) + 1;     % get new i coord (m+n) mod p
9.        newj = mod(((i-1) + 2*(j-1)),p) + 1;   % get new j coord (m+2n) mod p
10.       X(newi,newj) = Y(i,j);
11.   end
12.end
13.X = uint8(X);
14.function X = catmap(Y)
15.% applies Arnold's Cat Map to the image Y to produce image X
16.
17.p = size(Y,1);      % get the number of pixels on each side
18.X = zeros(size(Y)); % make space for X (all zeros to start)
19.for i = 1:p         % loop through all the pixels
20.   for j = 1:p
21.      newi = mod(((i-1) + (j-1)),p) + 1;     % get new i coord (m+n) mod p
22.      newj = mod(((i-1) + 2*(j-1)),p) + 1;   % get new j coord (m+2n) mod p
23.      X(newi,newj) = Y(i,j);
24.   end
25.end
26.X = uint8(X);
```

Applying Matrix Transmutation

```
1. rgbImage = imread('C:\Users\MAHE_MSK\Desktop\ImageEncryptionGUI\1.jpg');
2. imshow(rgbImage);
3. set(gcf, 'units','normalized','outerposition',[0 0 1 1]);
4. drawnow;
5. [rows columns numberOfColorBands] = size(rgbImage)
6.
7. blockSizeR = 210; % Rows in block.
8. blockSizeC = 270; % Columns in block.
9. wholeBlockRows = floor(rows / blockSizeR);
10.blockVectorR = [blockSizeR * ones(1, wholeBlockRows), rem(rows, blockSizeR)];
11.wholeBlockCols = floor(columns / blockSizeC);
12.blockVectorC = [blockSizeC * ones(1, wholeBlockCols), rem(columns, blockSizeC)];
13.
14.%block division
15.if numberOfColorBands > 1
16.     ca = mat2cell(rgbImage, blockVectorR, blockVectorC, numberOfColorBands);
17.else
18.     ca = mat2cell(rgbImage, blockVectorR, blockVectorC);
19.end
20.
21.%display each block
22.plotIndex = 1;
23.numPlotsR = size(ca, 1);
24.numPlotsC = size(ca, 2);
25.for r = 1 : numPlotsR
26.     for c = 1 : numPlotsC
27.          fprintf('plotindex = %d,   c=%d, r=%d\n', plotIndex, c, r);
28.          subplot(numPlotsR, numPlotsC, plotIndex);
29.          rgbBlock = ca{r,c};
30.          imshow(rgbBlock); % Could call imshow(ca{r,c}) if you wanted to.
31.          X=rgbBlock;
32.
33.          %for Arnold catmap function call
```

```
34.             for i = 1:60
35.              X = catmap(X);
36.              figure(1);image(X);title(i);
37.              refresh;
38.              pause(.000000001);
39.             end
40.       ca{r,c}=X;
41.       [rowsB columnsB numberOfColorBandsB] = size(rgbBlock);
42.             %caption = sprintf('Block #%d of %d\n%d rows by %d columns', ...
43.                   %plotIndex, numPlotsR*numPlotsC, rowsB, columnsB);
44.             %title(caption);
45.             drawnow;
46.             plotIndex = plotIndex + 1;
47.       end
48.end
49.
50.
51.%reconsolidate the blocks
52.for r = 1 : numPlotsR
53.       for c = 1 : numPlotsC
54.B=cell2mat(ca);
55.end
56.end
57.figure(2);
58.image(B);
```

Fig. Block formation for Encoded Image (Applying Matrix Transmutation Algorithm over Encoded Image Obtained by Blowfish Algorithm)

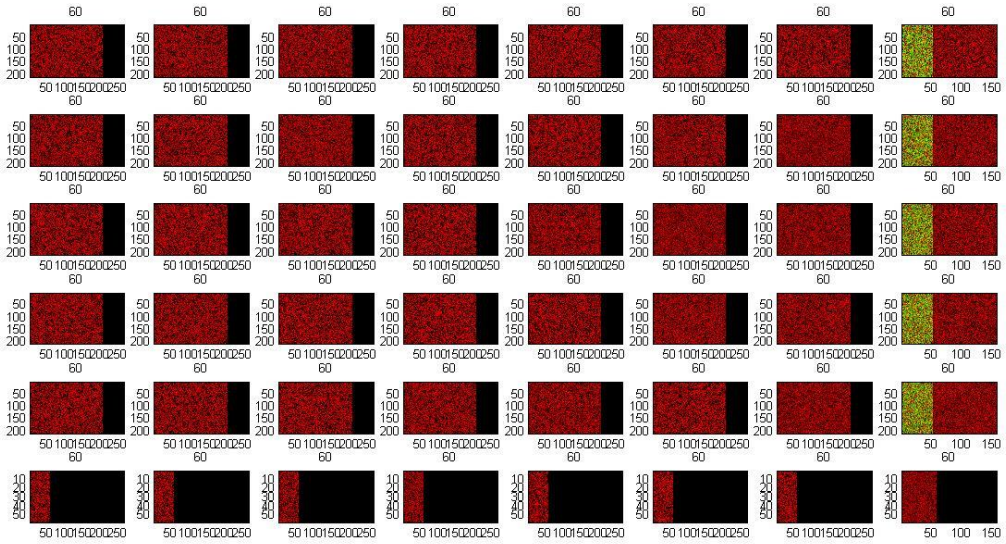


Fig. Encoded Image Applying Matrix Transmutation Algorithm over Encoded Image Obtained by Blowfish Algorithm

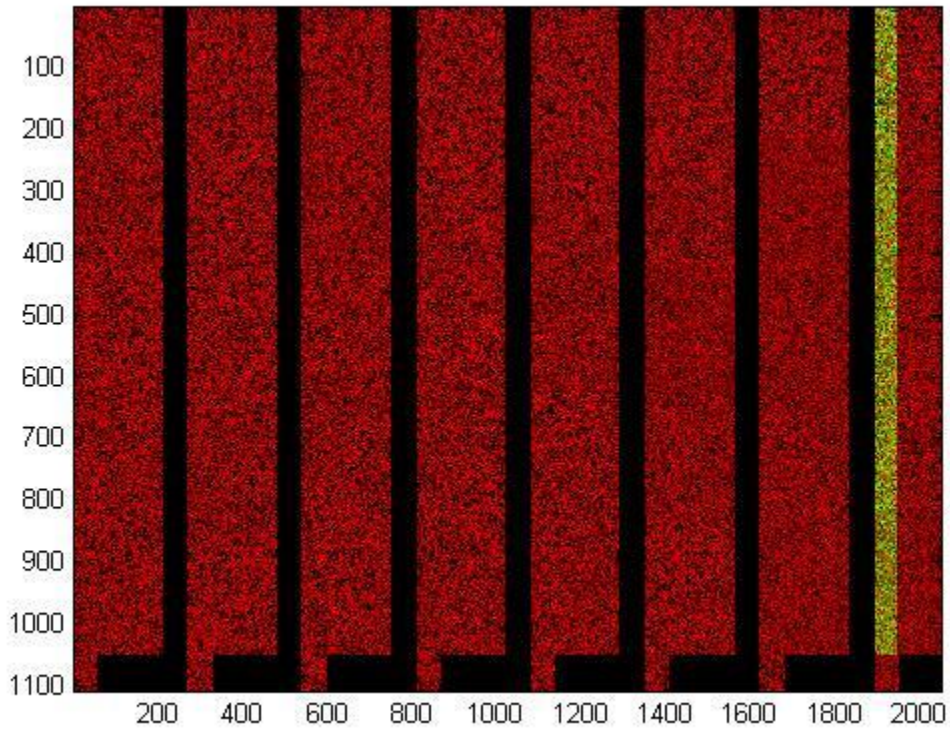


Fig. Histogram of Encoded Image (Applying Matrix Transmutation Algorithm over Encoded Image Obtained by Blowfish Algorithm)

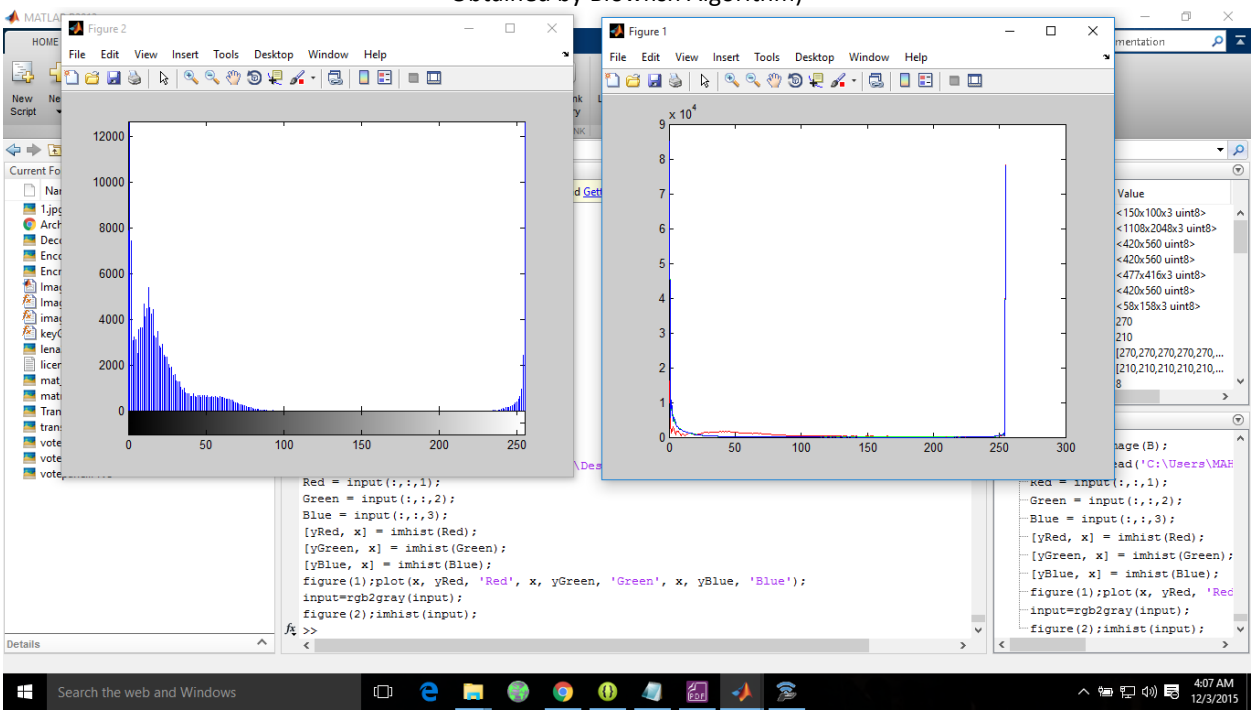
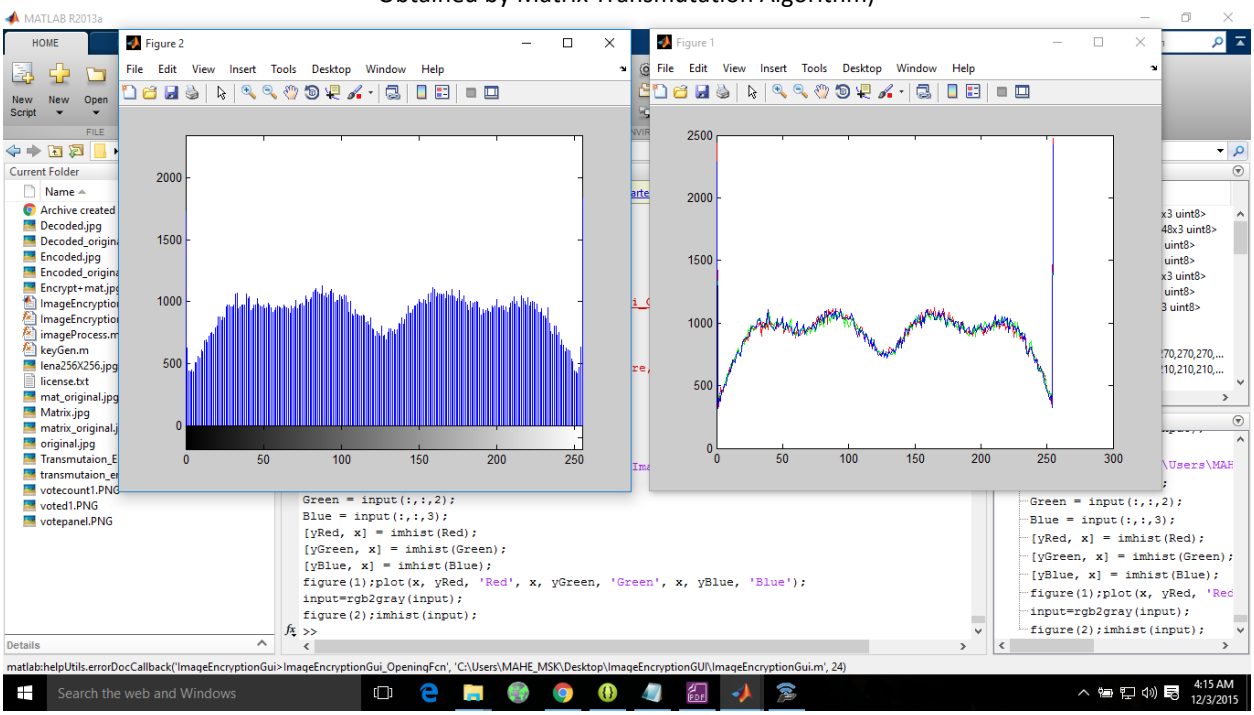
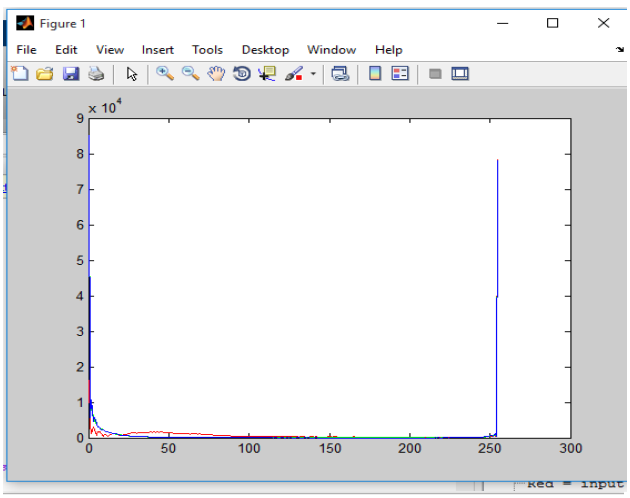
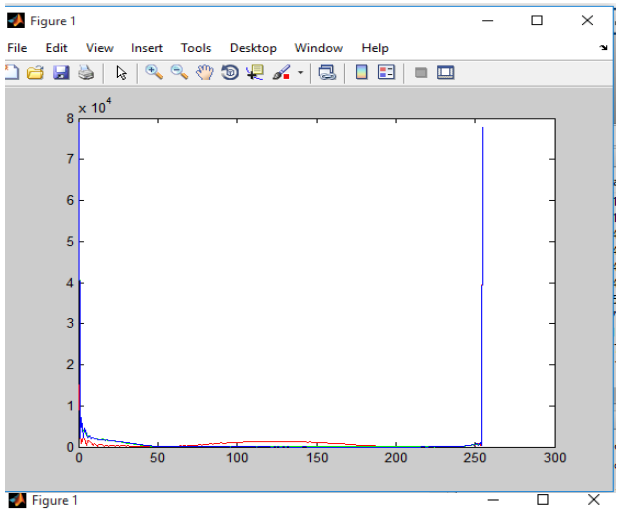


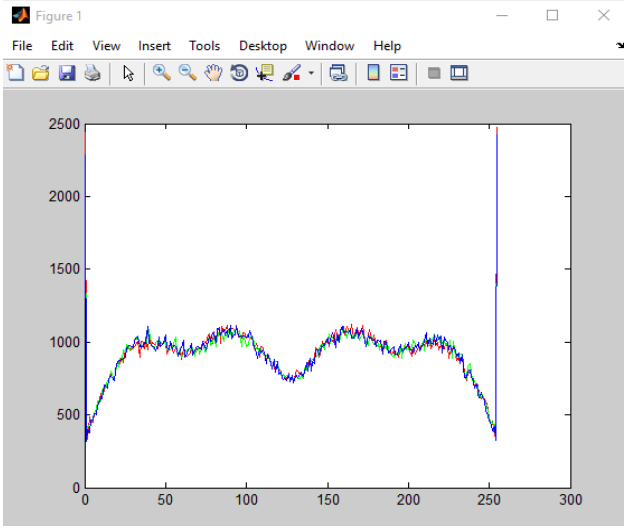
Fig. Histogram of Encoded Image (Applying Blowfish Algorithm over Encoded Image Obtained by Matrix Transmutation Algorithm)

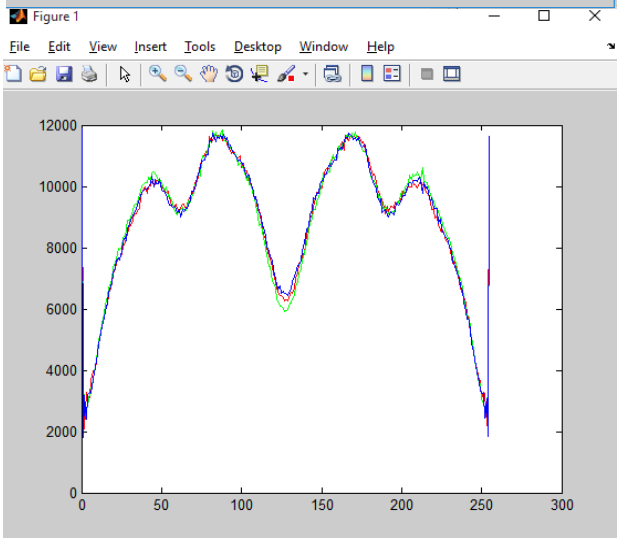


TESTING









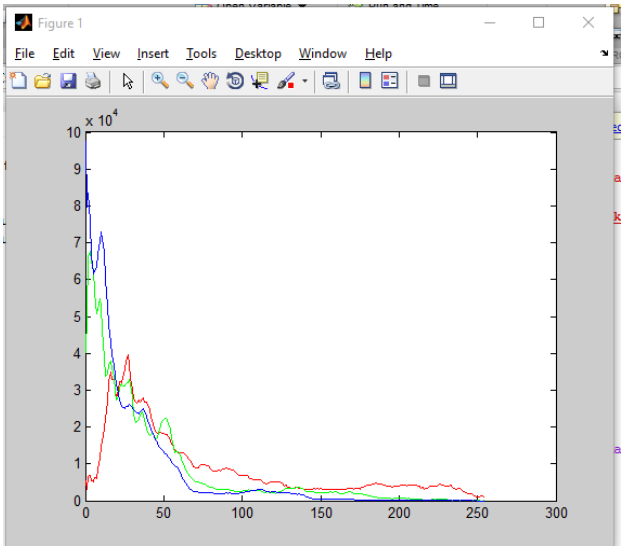


Fig: a d
b e
c

a) Histogram of Encoded Image Applying Matrix Transmutation Algorithm over Encoded Image Obtained by Blowfish Algorithm

b) Histogram of Encoded Image Applying Blowfish Algorithm over Encoded Image Obtained by Matrix Transmutation Algorithm

c) Histogram of Original image

d) Histogram of Encoded Image Applying Matrix Transmutation Algorithm

e) Histogram of Encoded Image Applying Blowfish Algorithm.

Test Report

Algorithm Used	Pixel Variation for max and min value(Approx.)	Average number of pixel Distribution
Original Image	9000	> 10000
Blowfish Only	7500	1000 - 5000
Matrix Transmutation Only	7000	< 1000
Matrix Transmutation over Blowfish	6000	< 500
Blowfish Over Matrix Transmutation	2100	500 - 1000

Fig: Pixel Distribution over Grayscale (0-250) for given Image

CONCLUSION

1. From the above results we can conclude that applying blowfish over Matrix Transmutation or vice-versa results in good encryption with large confusion and diffusion value.

2. Applying Matrix Transmutation or Blowfish algorithm one at a time on the original image is not sufficient as both are vulnerable to statistical attack.

3. When applying Matrix Transmutation most of the pixels lie on the extremes values of Grayscale but on an average, number of pixels remains constant so more resistant to statistical Attacks than Blowfish.
4. Whether we apply Matrix Transmutation over Blowfish or Blowfish over Matrix Transmutation both give us approximately the same result.

5. Out of the various encryption schemes DES, AES, 3DES and Blowfish discussed Blowfish Algorithm has the highest efficiency.

FUTURE WORK

1. Applying Image encryption by using Henon mapping system to diffuse the correlation between crypto-image and plain-image in the Matrix Transmutation Algorithm along with the applied one.

2. To Apply Matrix Transmutation in between the various rounds of blowfish Algorithm to check the efficiency.

3. Apply Matrix Transmutation along with DES, 3DES, AES and then doing comparison.

REFERENCES

1. REF 1, "Internet Security Glossary", http://www.faqs.org/rfcs/rfc2828.html
2. Performance Analysis of Data Encryption Algorithm, Abdel-Karim Al Tamimi, aa7@wustl.edu
3. The Scientific World Journal Volume 2014 (2014), Article ID 536930
4. Wikipedia
5. International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181
6. MATLAB Central – MathWorks, www.**mathworks**.in/matlabcentral/
7. Stack Overflow, stackoverflow.com/