VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

Image Security using Pixel Scrambling and Blowfish Algorithm

Team Members:

19BCI0171 (Prajjwal Gupta)

19BCI0183 (Anshul Ola)

19BCI0184 (Saurabh Singh)

19BCI0191 (ArpitUpadhyay)

Report submitted for the

Final Project Review of

Course Code: CSE1011

Slot: C1

Professor: Dr.IlanthenralKandasamy

1. Introduction:

Encryption is a method which uses a bounded set of instruction called an algorithm to convert original message, referred to as plaintext, into cipher text, i.e. its encrypted form. Cryptography algorithms normally require a set of characters called a key to encrypt or decrypt data.

So in this project we will be implementing scrambling and unscrambling techniques on the images with a secret key block cipher i.e. BLOWFISH ALGORITHM to encrypt and decrypt images and study hoe the images vary.

Blowfish is developed as a successor of DES as it is more secure and fast because of varying key size. Has very Flexible and simple design.

Image encryption plays a important role within the field of data hiding. Image encryption method prepared information unreadable, this algorithm will be used as a variable key size from 32bits up to 448 bits. It uses the Feistel structure network which iterates the simple function 16 times overall. The Blowfish algorithm is safer against the Unauthorized attacks and runs faster than most of the popular existing algorithms.

2. Literature Review:

Paper1:

"The Blowfish Encryption Algorithm"

- **Authors and Year(Reference)-**

Bruce Schneier ,Dr.Dobb's Journal,Vol.19,No.4,April 1994, pp. 38-40

- **Title(Study)-**

Description of a New Variable-Length Key, 64-BitBlock Cipher (Blowfish)

- **Concept / Theoretical model/Framework-**

The algorithm should manipulate data in large blocks 32 bits in size , 64 bits block size. Using of simple operations that are efficient on microprocessors i.e exclusive-or, addition, table lookup, modular-multiplication.

- **Methodology used/Implementation-T-**

Encryption occurs in a 16-round Feistel structure. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additionson 32-bit words. The only additional operations are four indexed array data lookups perround.

- **Datasetdetails/analysis**

Blowfish, -secret-key block cipher- Feistel network, iterating - 16

times. The block size is 64 bits, and the key length varies from 32 to 448 bits, 4 S- boxes, 18 subkeys needed. Complexinitialization phase needed before any encryption, the actual encryption of data is very efficient on largemicroprocessors.

- **RelevantFinding-**

Used mainly where key doesn't change often likecommunications link or an automatic file encryptor. It is significantly fast, compact, simple and secure thanDES.

- **Limitations/ Future Research/ Gapsidentified-**

It may be possible to reduce the number of S-boxes from four to one. It is probably safe to reduce the number of iterations from16 to 8 without compromising security. An alternate method of subkey calculation where every subkey can be calculated independently of any other can betried.

Paper 2:

- **Authors and Year(Reference)-**

B. Shamina Ross, V. Josephraj, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp.9236-9244

- **Title(Study)-**

"Performance Enhancement of Blowfish EncryptionUsing RK-BlowfishTechnique"

- **Concept / Theoretical model/Framework-**

An idea to enhance the performance of the Blowfish algorithm by introducing parallel processing technique and making changes to the Fiestel (F) structure of Blowfish by combining the Blowfishand the Runge-kutta (RK)Method.

- **Methodology used/Implementation-T-**

In this they used Range Kutta Method parallely with Blowfish, The parallel evaluation reduces the time consumed for two additions to one and the shift operation also executessimultaneously.

- **Datasetdetails/analysis-**

In this the algorithm uses 16 iterations, the time is reduced 16 times for every encryption and decryption. This is asignificant enhancement.

- **Relevant Finding-**

Less time consuming, hidher throughput and high security. RK-Blowfish can be used in consumer electronic devices such as PDAs and smart phones which require less memory andpower consumption.

- **Limitations/ Future Research/ Gapsidentified-**

This study can be further extended with optimizationtechniques which have highpotentials.

Paper 3:

- **Authors and Year(Reference)-**

PiaSinghandProf.KaramjeetSingh,InternationalJournal ofScientific&EngineeringResearch,Volume4,Issue7,July-2013ISSN2229-5518,

- **Title(Study)-**

"ImageEncryptionAndDecryptionUsingBlowfish Algorithm InMatlab"

- **Concept / Theoretical model/Framework-**

The main aim behind the design of this proposal is to get the best security/performance tradeoff over existing Web Images. In this paper we are implementing Blowfish algorithm which is strongest and fastest in data processing/storing compare to otheralgorithms.

- **Methodology used/Implementation-T**

Blowfish algorithm is highly secured because it has longer key length (more no of key size). Blowfish incorporates a 16 round Feistel network for encryption anddecryption.

- **Datasetdetails/analysis**

First obtaining the matrix and pixels of the chosen image & then we

would be encrypting the image matrix usingblowfish algorithm. The result shows the original image, encrypted image and the decryptedimage.

- **Relevant Finding-**

The text in the image will be hidden using a specific key and image hidden with a data is encrypted and decrypted by a32 bit iteration loop and display in MATLAB. We will clearly see that the decrypted image is same as the originalimage.

- **Limitations/ Future Research/ Gaps identified-**

Blowfish cannot be broken until an attacker tries 28r+1 combinations where r is the number of rounds. Hence ifthe number of rounds are been increased then the blowfish algorithm becomesstronger.

Paper 4:

- **Authors and Year(Reference)-**

International Journal of Information and Education Technology,Vol.1,No.2,June2011,publishedby**K. Sakthidasan and B. V. SanthoshKrishna**

- **Title(Study)-**

"ANewChaoticAlgorithmforImageEncryptionandDecryption of Digital ColorImages"

- **Concept / Theoretical model/Framework-**

The chaos-based encryption[5,6] has suggested a new andefficient way to deal with the intractable problem of fast and highly secure image encryption. After Matthews proposed the chaotic encryption algorithm in 1989, increasing researches of image encryption technology are based on chaotic systems.

- **Methodology used/Implementation-T-**

The chaos-based image cryptosystem mainly consists oftwo stages. The plain image is given at itsinput.
Secret key passes through key generator and then pixel permutation and sequential pixel value modification occurs.

- **Dataset details/analysis**

Key space analysis, statistical analysis and sensitivity analysis were carried out to demonstrate the satisfactory security of the new scheme. The image-histogram illustrates how pixels in an imageare distributed by graphing the number of pixels at each color intensity level.

- **Relevant Finding-**

By incorporating all the features, the proposed cryptosystemavoids all the crypto graphical weaknesses of earlier chaos-based encryptionsystems.

- **Limitations/ Future Research/ Gapsidentified-**

The chaos-based image encryption scheme can be applied to

moving images as well. The prevalence of multimedia technologyin the society has promoted digital images and videos to play a more significant role than the traditional texts, which demands a serious protection of users'privacy.

Paper 5:

- **Authors and Year(Reference)-**

IOSR Journal of Computer Engineering (IOSR-JCE)e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. X (Mar-Apr. 2014), PP 80-83, published by MsNehaKhatri Valmik1 and Prof. V. KKshirsagar

- **Title(Study)-**

"Blowfish Algorithm"

- **Concept / Theoretical model/Framework-**

This system basically uses the Blowfish encryption algorithm to encrypt the data file. This algorithm is a 64-bit block cipher with a variable length key. This algorithm has been used because it requires less memory. It uses only simple operations, therefore it is easy to implement.

- **Methodology used/Implementation-T-**

It is having a function to iterate 16 times of network. Each round

consists of key dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bitwords.

- **Dataset details/analysis**

Blowfish is a symmetric encryption algorithm, meaning that it uses the same secret key to both Encrypt and decrypt messages. Blowfish is also a block cipher [5], meaning that it divides a message up into fixed length blocks during encryption and decryption.

- **Relevant Finding-**

BLOWFISH is used frequently because:
It has been repeatedly tested & found to be secure.
It is fast due to its taking advantage of built-in instructions on the current microprocessors for basic bit shuffling operations.
It was placed in the public domains.

- **Limitations/ Future Research/ Gaps identified-**

By modifying the I/O from 64 bits to 16 bits the encryption becomes a bit vulnerable.

3. Objective of the project:

- The main objective for this project is to provide a new secure technique for image security over existing methods.

- Blowfish is developed as a successor of DES as it is more secure and fast because of varying key size. Has very Flexible and simple design.

- Blowfish algorithm will be used as a variable key size from 32bits up to 448 bits. It uses the Feistel structure network which iterates the simple function 16 times overall. The Blowfish algorithm is safer against the unauthorized attacks and runs faster than most of the popular existing algorithms.

- The basic idea of scrambling is to change the image pixel positions through matrix transform to achieve the visual effect of disorder.

- With the introduction of pixel scrambling and Blowfish algorithm image security increases twofold.

- And there are no viable techniques which can effectively break blowfish encryption as of now so we are usimg blowfish algorithm for encryption and decryption.

4. Innovation component in the project:

- We are providing a higher level of security with compared to existing methods.
- We are introducing scrambling in our project. The websites and companies which mostly depend on image transmission need a viable method encrypting the images and sending them to the user.
- Scrambling is one of the most important parts of confusion diffusion based image encryption. A huge number of scrambling techniques are available in literature. Therefore choosing the correct scrambling method for an encryption scheme becomes most crucial.
- We are using row and column shifting with circular rotation i.e block shuffling algorithm method for image scrambling and unscrambling.
- We are also implementing histogram comparison technique to find any change in pixel data representation.

If our methodology is used in a program then the image sent to other user's device will be encrypted in gibberish and the one which has key and know what we used can decrypt it.

5. Work done and implementation:

Methodology Adapted:

1. Image Scrambling:

- Image Scrambling makes images unreadable thus making it difficult for unauthorized users to view the image.

- The input image is scrambled using the block shuffling algorithm which starts with a 2x2 block of image and moves on to a specified block size (over here width of image/4) and rotates the block 180 degrees clockwise.

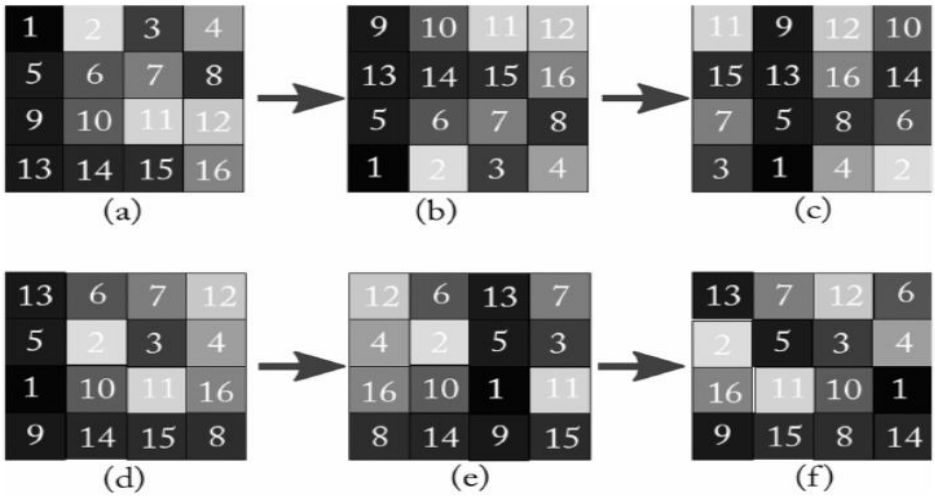


Figure 2.1 Step by step process of scrambling using row and column shifting with circular rotation: (a) plain sub-image with 16 pixels is presented, (b) the rows of the image are scrambled based on a random number sequence {3,4,2,1}, (c) the columns of the image are scrambled based on a random number sequence {3,1,4,2}, (d) shows that the pixels are only scrambled within the row or column, (e) column shifting using random sequence {4,2,1,3}, and (f) column wise circular rotation using random sequence {2,3,1,2}.

ALGORITHM 2.1 Row and Column Shifting with Circular Rotation

Input: The original image I.
for $i = 0$ to M **do**
    Cyclic shift the i^{th} row of I to right with step size $R1_i$;
end for
The row scrambled image is denoted as I_1.
for $i = 0$ to N **do**
    Cyclic shift the i^{th} column of I_1 to right with step size $R2_i$;
end for
The column scrambled image is denoted as I'.
Output: The scrambled image I'.

Algorithm for Block Shuffling

- def rot(A, n, x1, y1): #this is the function which rotates a given block
- temple = []
- for i in range(n):
- temple.append([])
- for j in range(n):
- temple[i].append(arr[x1+i, y1+j])
- for i in range(n):
- for j in range(n):
- arr[x1+i,y1+j] = temple[n-1-i][n-1-j]
- for i in range(2, BLKSZ+1):
- for j in range(int(math.floor(float(xres)/float(i)))):

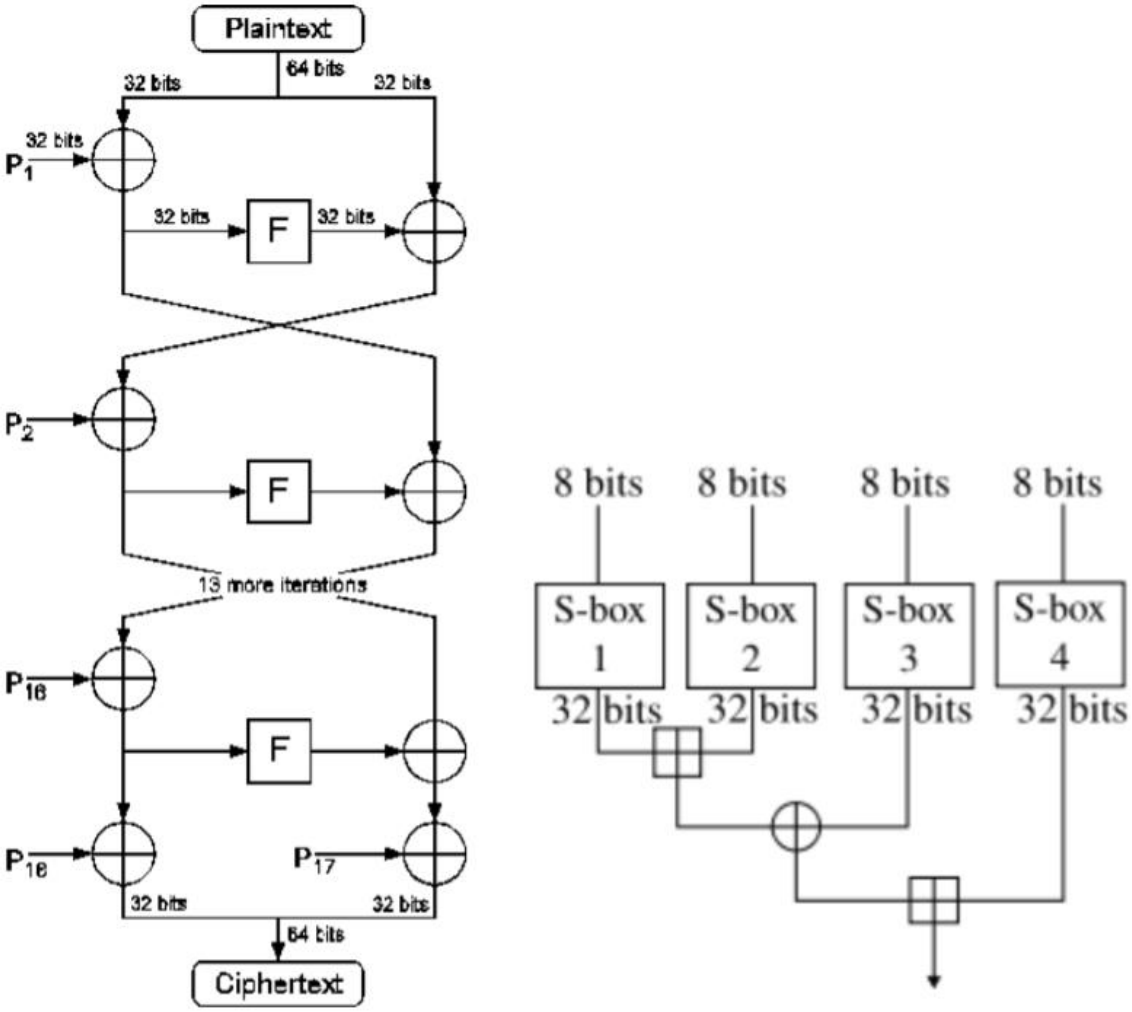
- for k in range(int(math.floor(float(yres)/float(i)))):
- rot(arr, i, j*i, k*i)
- for i in range(3, BLKSZ+1):
- for j in range(int(math.floor(float(xres)/float(BLKSZ+2-i)))): 16
  for k in range(int(math.floor(float(yres)/float(BLKSZ+2-i)))): 17
  rot(arr, BLKSZ+2-i, j*(BLKSZ+2-i), k*(BLKSZ+2-i))

2. The Blowfish Algorithm

- Bruce Schneier designed blowfish in 1993 as a fast, license free alternative to existing encryption algorithms. Since then it's been analysed considerably, and it's slowly gaining the acceptance as a strong Encryption Algorithm.

- Blowfish is a symmetric 64-bit block, fiestel network cipher. It considers key length ranges from 32 to 448 bits. It is a 16 round Feistel cipher and uses large key-dependent S-boxes.

- An S-box is used in block ciphers to distance the relation between the key and the cipher text by using methods of substitution. An S box takes any number of m bits as input and changes into an 'n' number of outputs.



3(a) 3(b)

Figure 3(a) shows the how blowfish is implemented as a feistel structure and Figure3(b) shows the function F employed in the feistel block.

Why Blowfish?

With problems arising in existing methods such as Digital Encryption Standard (DES) and Advanced Encryption Standard (AES) such as small block cipher size ranging in few megabytes, and inefficient generation of the block ciphers, Blowfish arose as a new alternative around in the early 90's.

Blowfish is one of the fastest block ciphers in general use, except when changing keys. Each new key needs a pre-processing equivalent to encrypting about 4 kilobytes of text data, which is very slow if compared to the other block ciphers. This stops its use in certain applications, but it's not a problem in others.

The advantages of Blowfish over the other algorithms discussed are as follows:

- Flexible and simple design. Flexible in the sense that it supports large variety of range in key lengths up to 448 bits which can be implemented on various platforms and applications.
- It takes at max 500 clock cycles on Intel's Pentium architecture per block to encrypt data.
- It supports only efficacy operations on any 8,16,32- bits micro processors and some 64-bit microprocessor.
- Generally on a 8-bit microprocessor it takes at most 10 milliseconds to encrypt the data.
- It requires only 64 bytes of RAM to run on a 8-bit micro processor.

HISTOGRAM ANALYSIS

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. It plots the number of pixels for each tonal value.

We tried to plot the graph of original image and unscrambled image on graph having:

X- axis- Pixel Values

Y- axis- frequency of pixels

Histogram analysis will show the intensity distribution of images.

ALGORITHM-

```
# importing required libraries of opencv
import cv2
# importing library for plotting
from matplotlib import pyplot as plt
img = cv2.imread('123.png',0)
```

Hardware and Software Requirements:

- Coding is done in PYTHON language as it has wide libraries which is very helpful for implementing the complex functions.
- The software which we will be using for the implementation of this project is

    - Spyder : Spyder is an open-source cross-platform IDE. The Python Spyder IDE is written completely in Python. It is used for running and editing codes and part of codes.

    - Anaconda : Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.

    - Notepad : It is used for simplify and editing of codes which then can be run on terminal.

    - Github : It provides access control and several collaboration features. Code is shared using this platform.

    - MS Teams/Meet : This platform is used for discussing and coding simultaneously.

Dataset used / Tools Used:

Blowfish, -secret-key block cipher- Feistel network, iterating - 16 times. The block size is 64 bits, and the key length varies from 32 to 448 bits, 4 S- boxes, 18 subkeys needed. Complex initialization phase needed before any encryption, the actual encryption of data is very efficient on large microprocessors.

Paper: *"The Blowfish EncryptionAlgorithm"*

- ***Title(Study)-***

  Description of a New Variable-Length Key, 64-BitBlock Cipher(Blowfish)

  - ***BruceSchneier****,Dr.Dobb'sJournal,Vol.19,No.4, April 1994, pp. 38-40*

- The Blowfish algorithm works pretty fast on images (faster than AES & DES) and is a good choice for going about image encryption.

- Next we used the Row and Column Shifting with Circular Rotation as discussed here: https://www.researchgate.net/publication/331064749_Cryptographic_image_Scrambling_techniques

- The whole codebase is implemented in Python programming language as it has support of vast number of libraries which have made our lives easier.

- Our work primarily involved the use of “Pillow” (imported as PIL) for image handling (loading & saving), “CryptoCipherDome” (imported as Crypto) which includes the Blowfish method used for the encryption and decryption, “math” for some mathematical computations and finally “matplotlib” for histogram analysis of the images obtained.

How does our project differ?

- We are provding a double level of security by scrambling the images and then encrypting it using Blowfish CBC mode which is then saved in a text file in gibberish form which is very hard or impossible to decode and a user Oscar will not even able to say what type of data it is.
- This Project also analyze the images using histogram implantation of images which will show us how the image differs when we scramble or unscramble the images and how it changes the pixel data.

Screenshot and Demo:

1. **SCRAMBLING OF IMAGE**
    a. Importing library files for image processing and getting pixel data of image using im.load().
    b. A Function rot() which is used for circular rotation of pixel data.(Block shuffling algorithm)
    c. Selecting a block of original image which will be sent for rotation for scrambling and unscrambling

```
from PIL import Image
from Crypto.Cipher import Blowfish #importing Image Processing Library from pillow
import math
from Crypto import Random
from struct import pack

im = Image.open("new.png", "r")
arr = im.load() #pixel data stored in this 2D array
w,h=im.size #saving value of image dimension in 2 variables w,h

def rot(A, n, x1, y1): #this is the function which rotates a given block
    temple = []
    for i in range(n):
        temple.append([])
        for j in range(n):
            temple[i].append(arr[x1+i, y1+j])
    for i in range(n):
        for j in range(n):
            arr[x1+i,y1+j] = temple[n-1-i][n-1-j]
xres = w
yres = h
BLKSZ = int(w /4)#blocksize

#selecting a block of original image which will be sent for rotation
for i in range(2, BLKSZ+1):
    for j in range(int(math.floor(float(xres)/float(i)))):
        for k in range(int(math.floor(float(yres)/float(i)))):
            rot(arr, i, j*i, k*i)
for i in range(3, BLKSZ+1):
    for j in range(int(math.floor(float(xres)/float(BLKSZ+2-i)))):
        for k in range(int(math.floor(float(yres)/float(BLKSZ+2-i)))):
            rot(arr, BLKSZ+2-i, j*(BLKSZ+2-i), k*(BLKSZ+2-i))

```

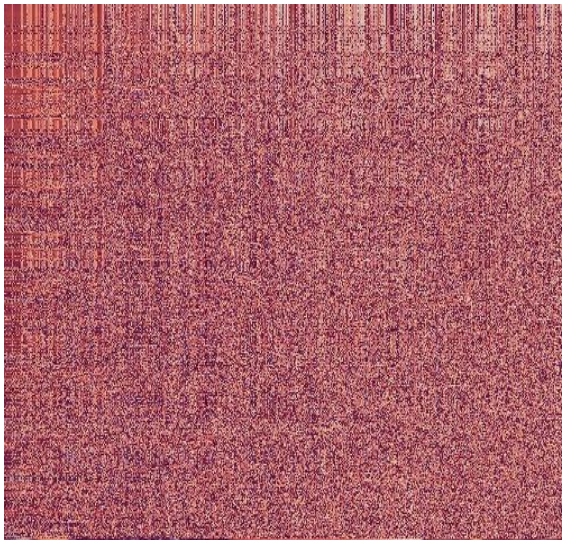
2- SAVING AND OPENING OF SCRAMBLED IMAGE

- Saving of Image as given in below example
- Opening of Image in read mode for getting the image data which we are encrypting.

```
#saving scrambled image
im.save("newscrambled.png")

key = b'11010011110111011101110110101110111011100001101'

im1 = Image.open("newscrambled.png", "r")
arr = im1.load() #pixel data stored in this 2D array
w,h=im1.size #saving value of image dimension in 2 variables w,h
#print(arr)

x=list(im1.getdata())#Returns the contents of this image as a sequence object containing pixel

#print(test)
x1=" ".join(str(a) for a in x)#replaces all commas with space


plaintext = bytes(x1, 'utf-8')#converting x1 into bytes from str
```

SAVING SCRAMBLED IMAGE

1- ORIGINAL IMAGE 2- SCRAMBLED IMAGE



3- BLOWFISH ALGORITHM IMPLEMENTATION-

- Padding: padding makes the data of equal length for encryption.
- Encryption: Encrypts the image data in CBC mode.This will provide us with a gibberish text.

(Ciphertext Block Chaining, defined in NIST SP 800-38A, section 6.2. It is a mode of operation where each plaintext block gets XOR-ed with the previous ciphertext block prior to encryption.)

- Decryption: Encrypted text is then decrypted by using Blowfish decrypt function an

```
57  bs = Blowfish.block_size
58  iv = Random.new().read(bs)
59
60
61  cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
62  plen = bs - divmod(len(plaintext),bs)[1]
63  padding = [plen]*plen
64  padding = pack('b'*plen, *padding)
65  msg1 = iv + cipher.encrypt(plaintext + padding)
66
67  #prints encrypted data
68  print("Encrypted Data")
69  print(msg1)
70
71
72  ciphertext = msg1
73  iv = ciphertext[:bs]
74  ciphertext = ciphertext[bs:]
75
76  cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
77  msg = cipher.decrypt(ciphertext)
```

4-Saves the encrypted data to a text file

```
last_byte = msg[-1]
msg = msg[:- (last_byte if type(last_byte) is int else ord(last_byte))]

#prints decrypted data
print("Decryption")

msg=str(msg, 'utf-8')
#msg1=str(msg1, 'utf-8')

text_file=open("sample.txt","wb")
text_file.write(msg1)
text_file.close
```

5-Saves decrypted data back to array (decrypted data is shown below)

```
a=msg.split()#cipher text to array data
arrr=[]
for i in range(0,h):#saving data to arrr[]
    arrr.append([])
    for j in range(0,w):
        arrr[i].append(a[j])


#saving and displaying encypted image


#unscrambling of scrambled image
print ("Retriving Original:")

x2=x1.split()#saving pixel data of scrambled image into x2

#x2 to arr2
arr2=[]
for i in range(0,h):
    arr2.append([])
    for j in range(0,w):
        arr2[i].append(x2[j])
```

6-Unscrambles the data saved in arr2[]

- Unscrambling the data using same block shuffling algorithm
- Saves unscrambled image and is shown

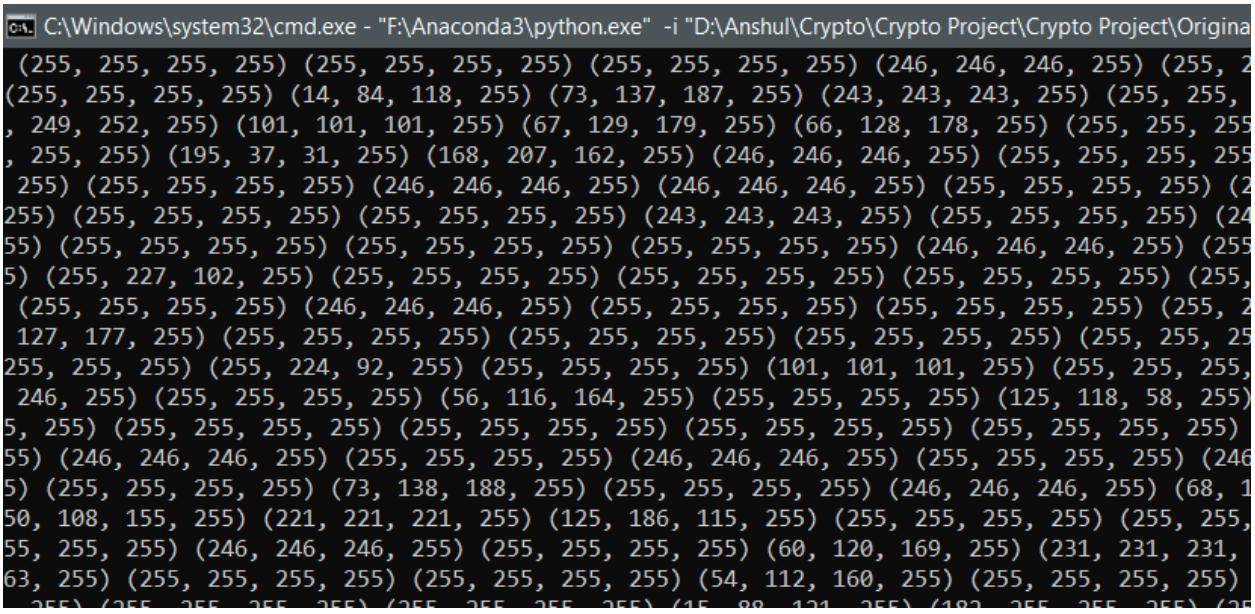
```
xres = w
yres = h
BLKSZ =int( w/4)#blocksize

#selecting a block of scrambled image which will be sent for rotation
for i in range(2, BLKSZ+1):
    for j in range(int(math.floor(float(xres)/float(i)))):
        for k in range(int(math.floor(float(yres)/float(i)))):
            rot(arr2, i, j*i, k*i)
for i in range(3, BLKSZ+1):
    for j in range(int(math.floor(float(xres)/float(BLKSZ+2-i)))):
        for k in range(int(math.floor(float(yres)/float(BLKSZ+2-i)))):
            rot(arr2, BLKSZ+2-i, j*(BLKSZ+2-i), k*(BLKSZ+2-i))

im1.save("newunscrambled.png")
im1.show("unscrambled.png")

```

OUTPUTS SCREENSHOTS-

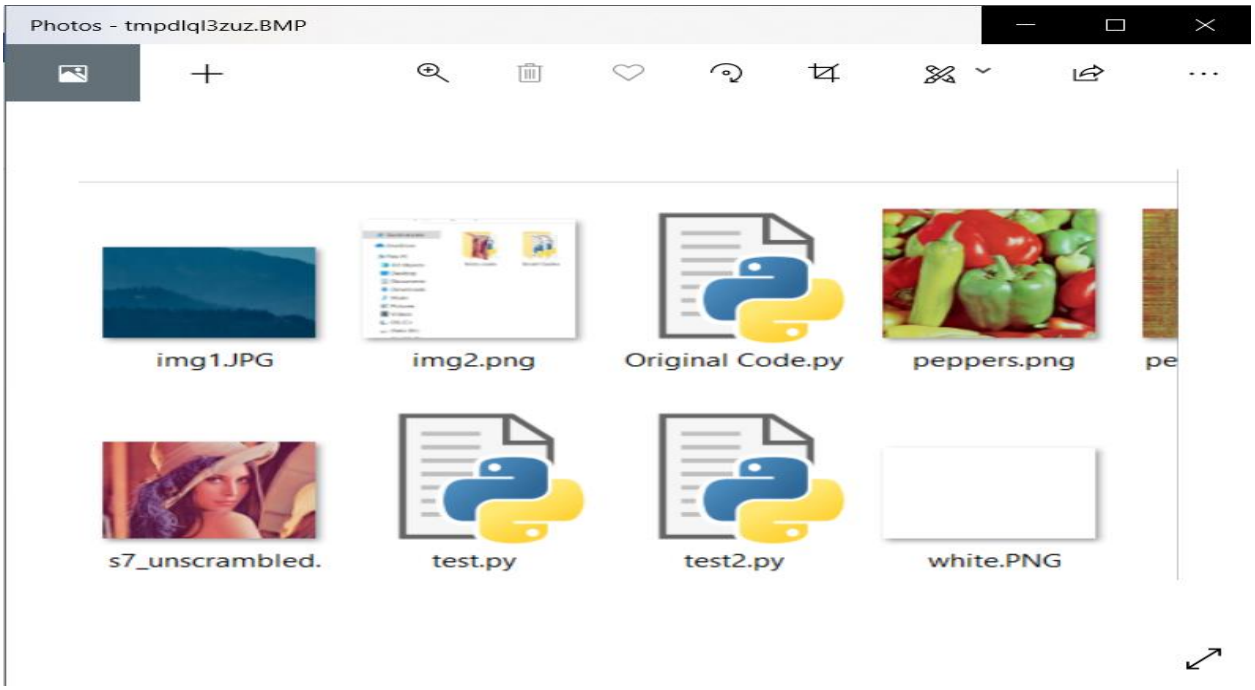
Encrypted Text:

```
C:\Windows\system32\cmd.exe - "F:\Anaconda3\python.exe"  -i "D:\Anshul\Crypto\Crypto
x95\x01\x8e\x9a\x15\x0b}\xafd\xe5\xdd\x91\xc1\xb9X\xadP_K\x8a\x94\xbb
\xcf\xab\xfc42K\x90\xe3\xb9\x80\xc6k\x9f`\x0c;F^q\xef\x94_G\xf6\xf3\'
W\xe25\xce\xf8\x08\xce\xc9Y\xe5\x89\xbdI4\x90F\x95Kj"\x13yR\xa9!\xa1\
4\tM\xb2\xd5\x039g\\I\x80\x02\xdcD\x19\xc4\xdaq\xa5\x88\xf3\x1fu\xf9%
\xb7\x93@\x1e5\x19=\xfbb\xe6E\xd1\x11\x07\xb0\x06\xbb\xe4\x1b\x87\xde
eaiy\x14\xe8\x9a\x02\x99\xea#m!6\xb3\x19c\x87\x84\x93\x88\xca\xaaZ\xf
x9e\x16\t\xe4\xc0\xa6\x99:\x81\x03\x13\x9aT\x14\x97\xd71q\x9d\x01\x0e
\xda\xe1a8L!\xf2+\xd0*>b\xc6\x08\x96N\xfd\x05..\x9b\x0f\x0e\xfc\x80\x
fa\x9f\x10\xce\x9c\xe2+$6\x05>\'\x82\x04\x98\x99\xf3\xa5\xc8\xf5\x0cE
xb3\x03\xe9\xe5/\xd6\xd8\x01\x0e\xda\xacF\x0b}\xf8L\xbb%\xd3,\xd7\xa6
E\xde\x15\xdc\xdb\xd7\xf8\x1b[\xdc\xc6\xf7\x14gSO\xb53?\xae\xccu\x01c
\xe1\xae<\xea_w4\xfbO\xb2\x12P\\\xd8\x1f\xbc\xe6\xee\xf8Ny\xf6{\x00O6
\xf4\x9bd\x01&\x9c\xc1\xbe\x02[\x1d\xa4\xca\xc2\xa5\xa1q\xd1<\x07R\x0
9\xd3C\xdc\x1dPs=\xa3\xd4\xe8\xe0\x7f#\xd9\t\xc7\xeap\xd3y\x0fUo\xf3\
f\xdd\x88\t\x15\xbc|\n\x129\xe5G|\xe5n\xdb\x89\xdd\x90\xbc\x06\xd2!e\
x05\x9c\x92\xc1\x9b,\xe3.\xcd\xd5x\xe5D\xd6"\xf5^\xea\xb5Z\x14wZ4\xe6
\x9e*\x85\xb0\x08\x1f\xb3\xf8\x1a\xce\xb7d-;\xc2\x9c\x0c\xf8P\xe2\xab
10\xef\x15\x07b\x1c\xdd\xfd\xdd\xb6\xaf\xac|sx7PRF]\xffd\x1bHs\xaaKo\
}Q\x18\x93\xcbx\xcf\xa6m\xff1\xee\xde\x95\xed~\x94V\x07f\xb7\x00]\xc3
xd4a\xc0\xe8*\xb2fC\x1al\xbe{\x9d,\x84g\xe2je$\x08\xdd6\x94\xdd\x15fk
```

Decrypted Text:

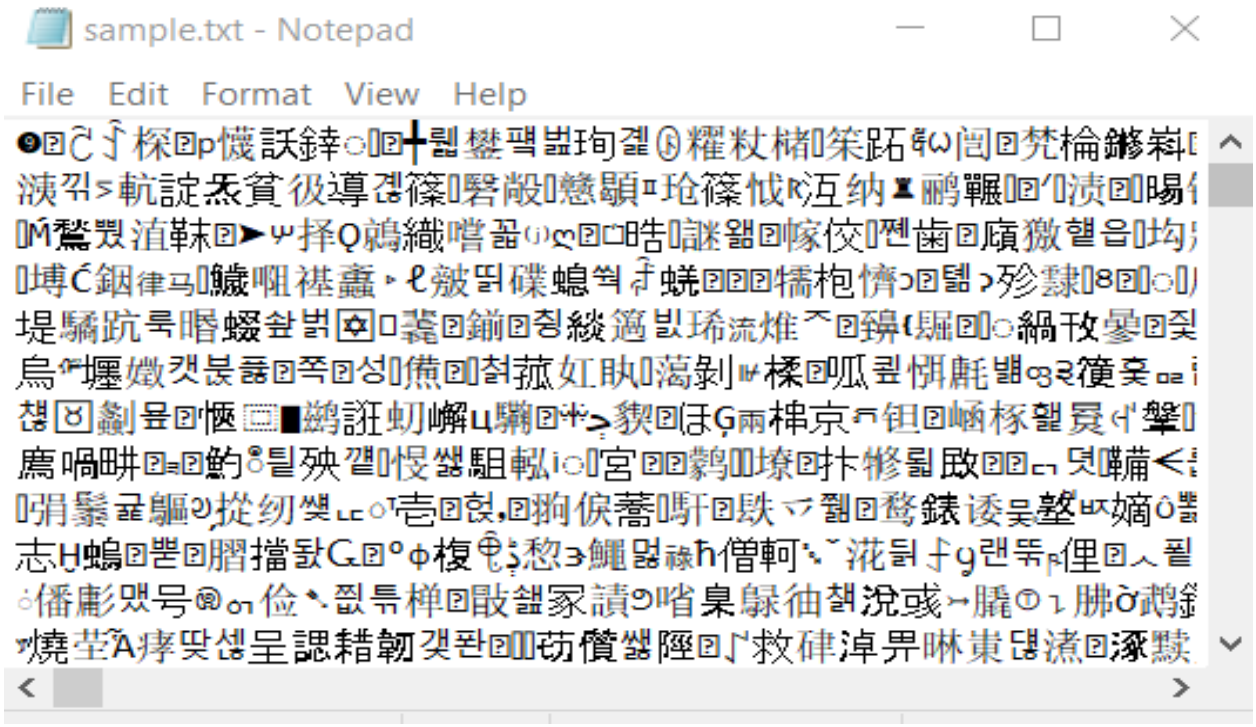


Unscrambled Image:



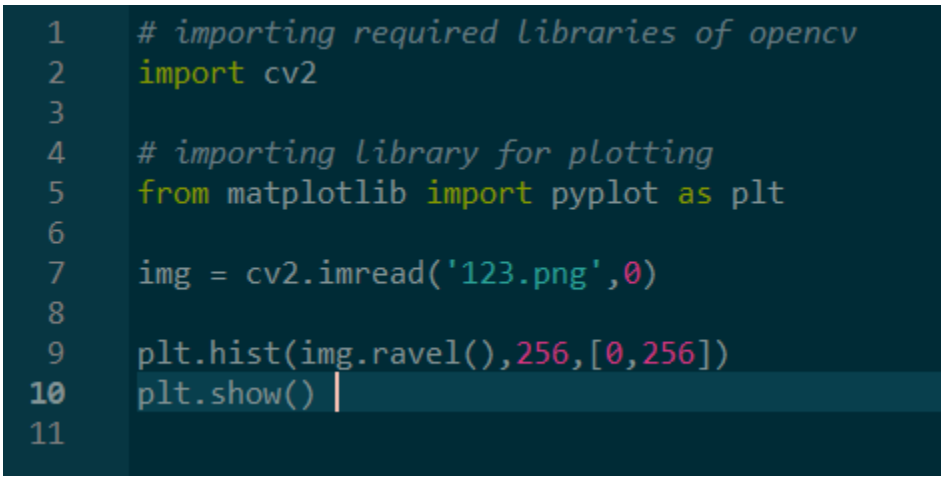
Saved Text File Which Contains Encrypted text:

The byte data stored in a text file is shown as-



HISTOGRAM ANALYSIS OF IMAGES

CODE SCREENSHOT-



```
# importing required libraries of opencv
import cv2

# importing library for plotting
from matplotlib import pyplot as plt

img = cv2.imread('123.png',0)

plt.hist(img.ravel(),256,[0,256])
plt.show()
```

Import cv2 library

Import matplotlib

Read image in directory

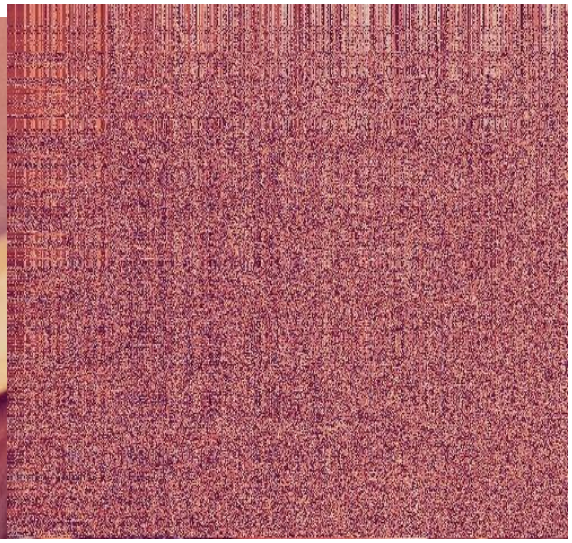
Plot the histogram

OUTPUT COMAPRISON-

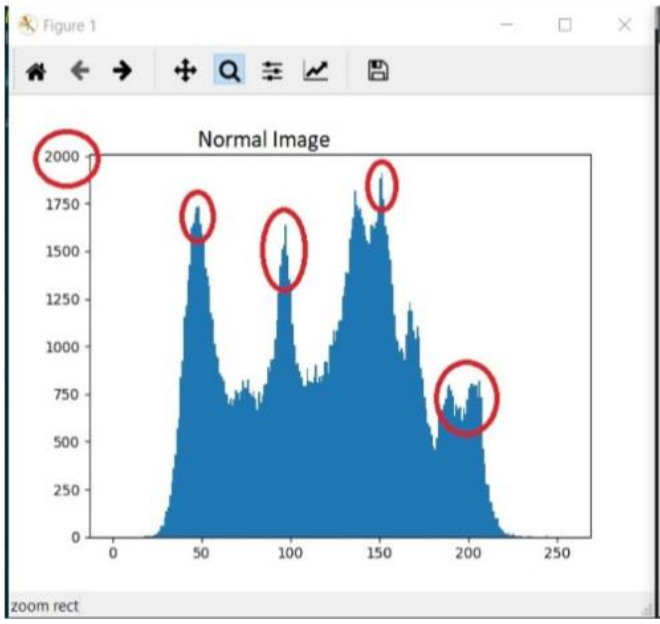
EXAMPLE-1

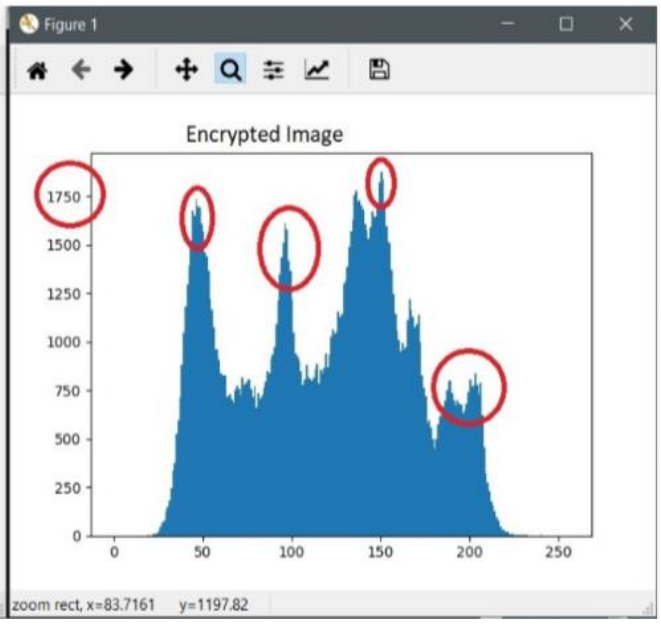
1- ORIGINAL IMAGE

2- SCRAMBLED IMAGE

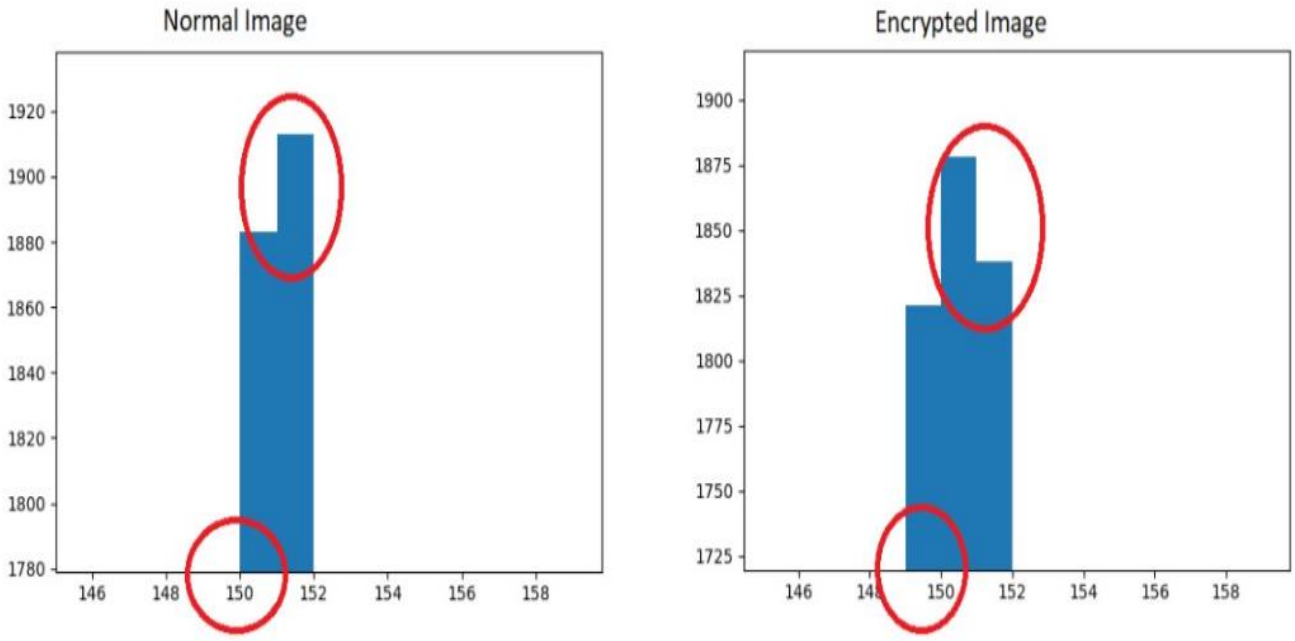


HISTOGRAM-



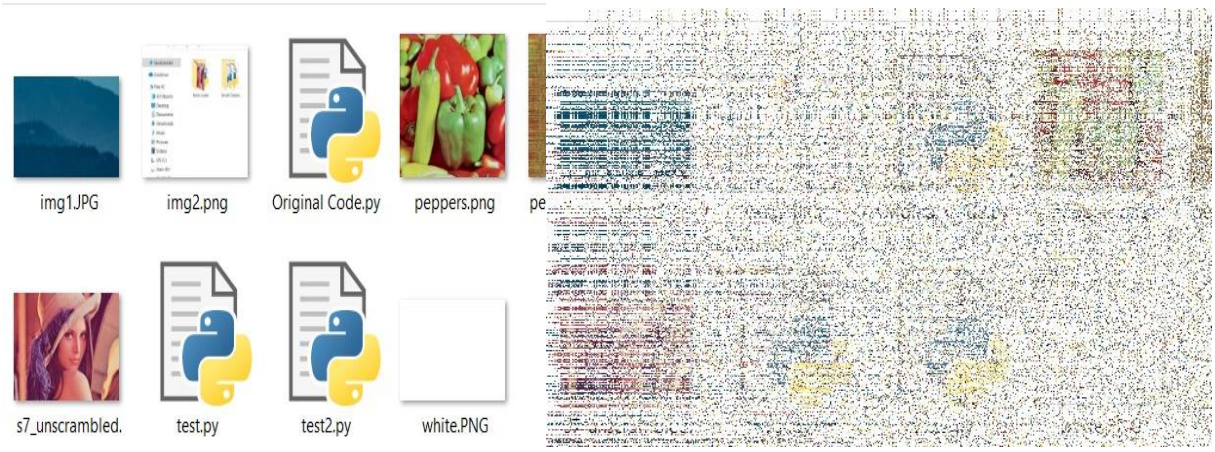


ZOOMED IMAGE AT A FIXED POINT-

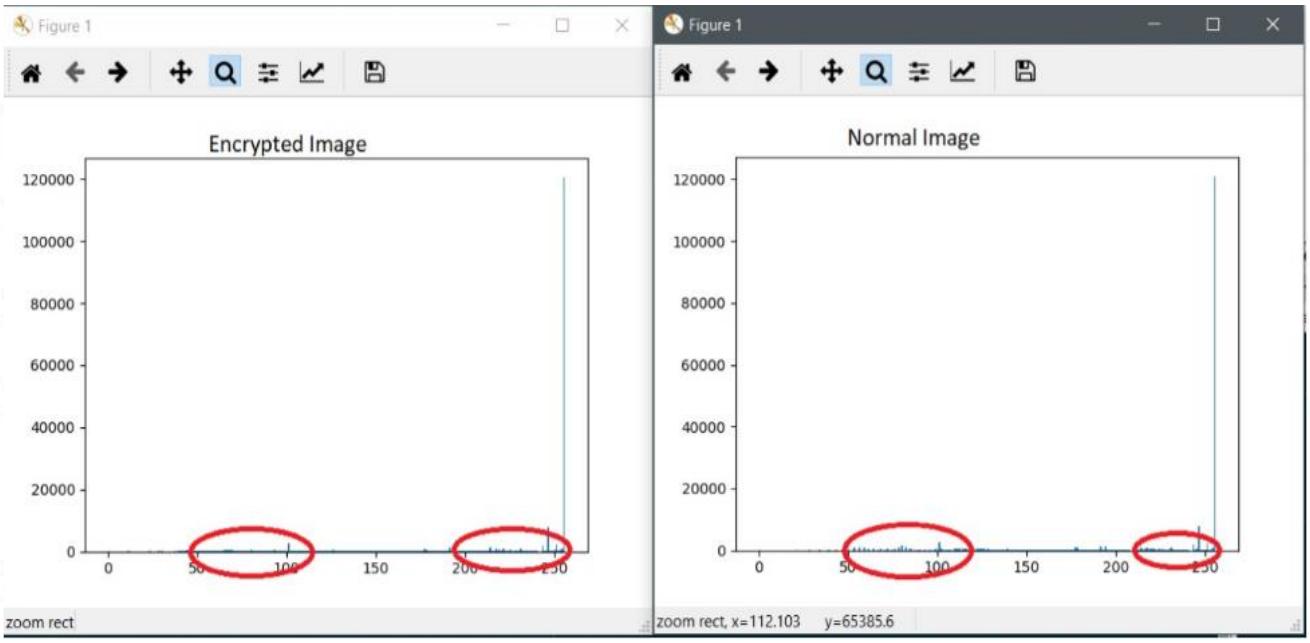


EXAMPLE 2-

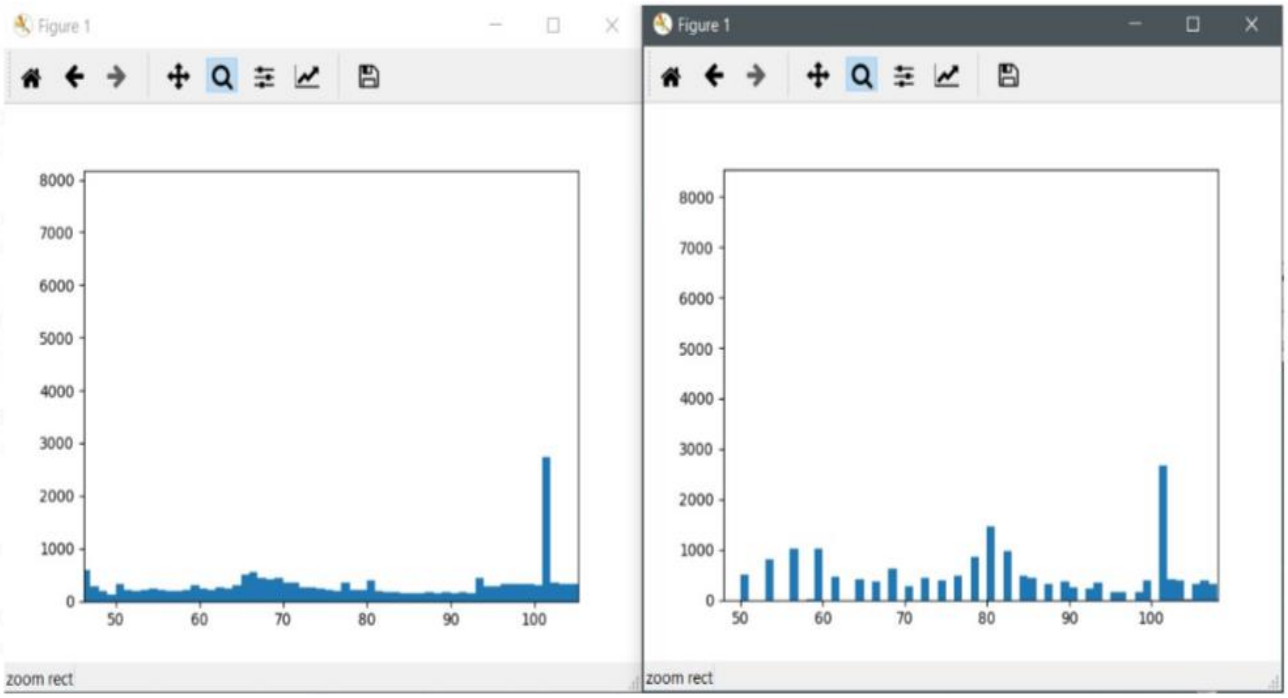
1- ORIGINAL IMAGE 2- SCRAMBLED IMAGE



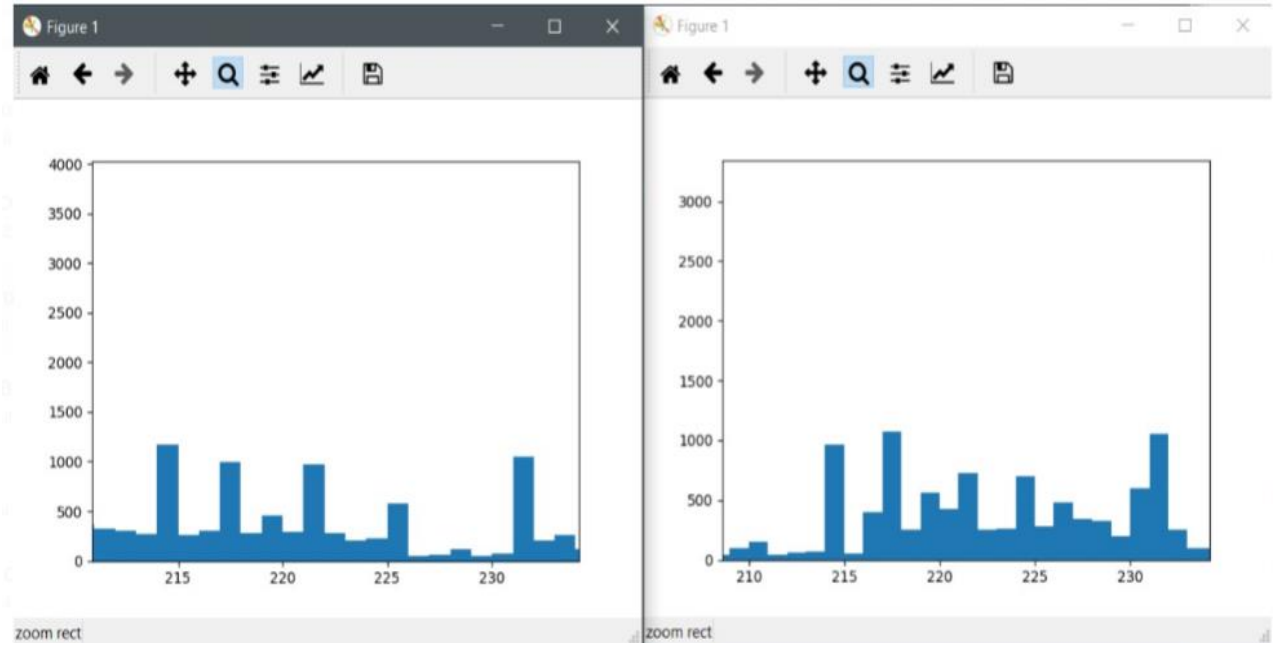
HISTOGRAM ANALYSIS-



ZOOMED

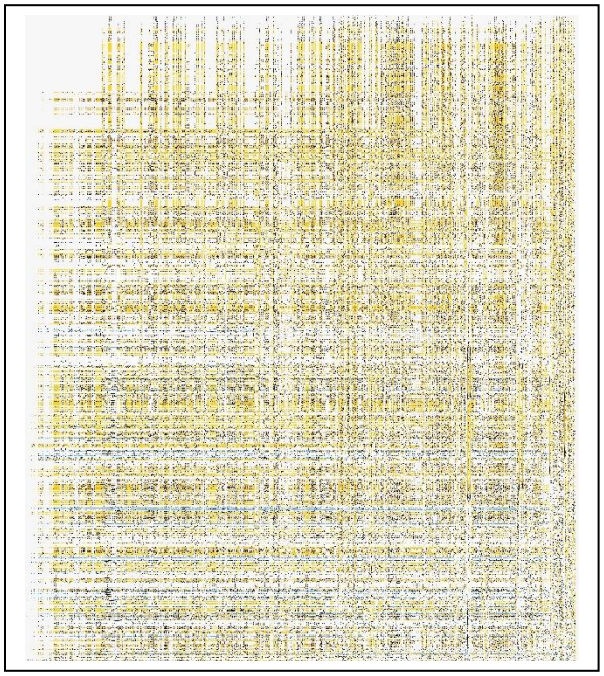


ZOOMED

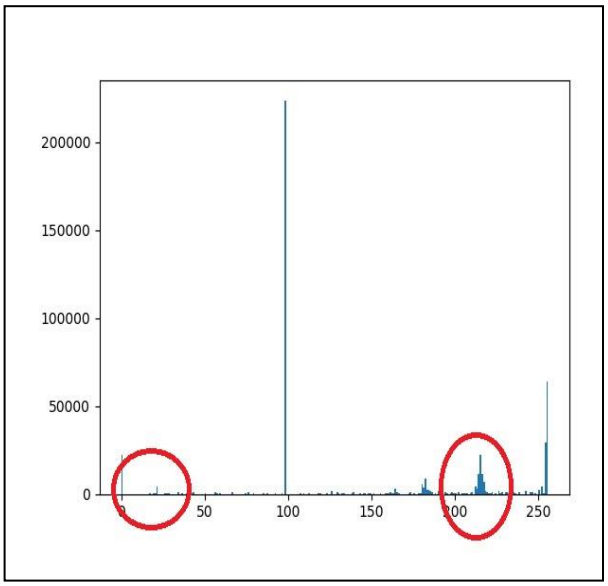


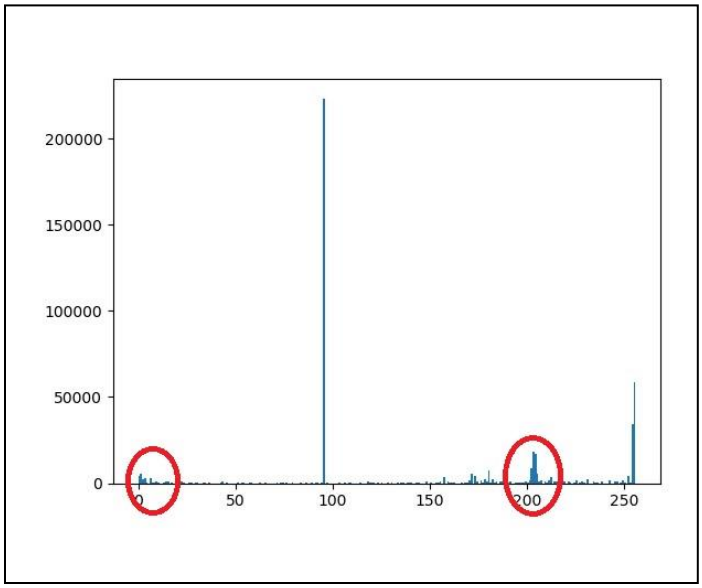
1- ORIGINAL IMAGE

2- SCRAMBLED IMAGE



HISTOGRAM ANALYSIS-





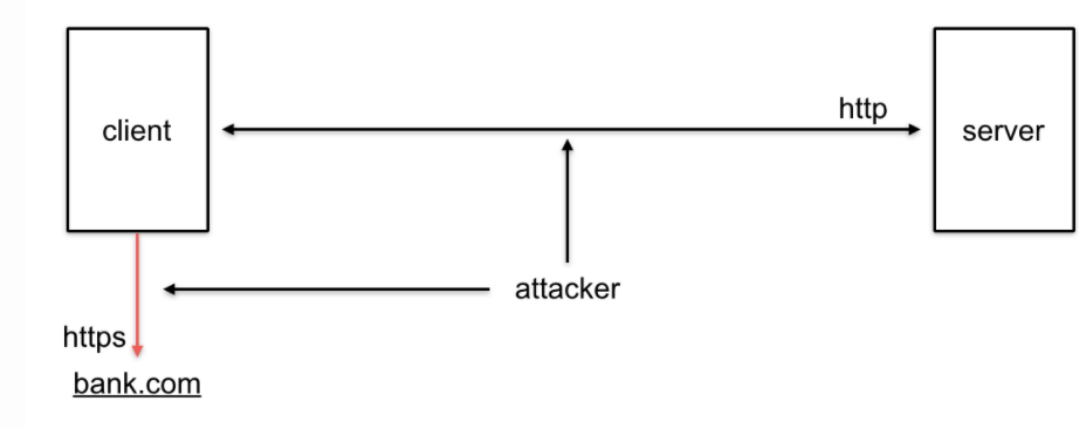
6- RESULTS AND DISCUSSIONS-

- As seen from the demonstration above we have successfully scrambled the image, encrypted it using blowfish algorithm, decrypted it and unscrambled the image with no loss! Thus image encryption with an extra layer of security has been achieved.
- The encrypted image is in the form of gibberish text which would be indistinguishable from other encrypted media (text messages, emails etc) in the channel , hence giving oscar (the channel snooper) a very hard time to figure out what exactly is being sent!
- In many cases it would be worthless for oscar to invest the huge time and effort to figure out what exactly is being sent over the

channel as the information(image being sent) is not that important, value may expire in the given time, futher decoding might be impractical.

NOTE-

- The known attacks against Blowfish includes SWEET32 (the most practical and feasible one) which is basically a combination of multiple man-in-the-middle attacks combined with phishing (not compulsory but quite effective).
- These attacks can be made harder to pull off by employing Twofish algorithm and the image scrambling techniques discussed above making the transmission highly secure.



- Another inference from the project can be drawn upon the lines of execution time. While this system works fairly fast for images of smaller dimensions, the routine is quite slow in processing larger images. This is primarily because the scrambling and unscrambling technique used is computationally expensive.
- This can be solved to some extent by using some other algorithms like 2D mapping and Fibonacci Transform.

- However, still the process is not that fast so that it is in harmony with the current fast technologies, it just might be the deal breaker. This follows to be a great area of research and we can hope future innovations make this or similar processing fast enough for commercial use.

Further Innovations / Discussions-

1. Use of Image Compression Algorithms

- The current approach involving pixel scrambling followed by encryption works fairly fast on images with small dimensions. But when applied to images of larger dimensions, the encryption and decryption was slow.
- In the fast pacing world, technology is aimed at making everything run faster and more economical. Thus in order to speed up the process, image compression before scrambling can be employed.
- However fully lossless image encryption hasn't been invented yet so this technique is limited to the applications in which some loss in data is not a big deal. (The original and compressed image look same to the naked eye but when pixel densities are computed there's a difference) Thus this technique cannot be used in military, high-priority, steganographic images.

2. Scrambling Algorithm alternatives

The algorithm used in the project is Row and Column Shifting with Circular Rotation. However various other algorithms exists too.

Some of them are:

-> Scrambling using Fibonacci Transformation

-> Scrambling using Arnold's Transformation

-> Gray code bit plane

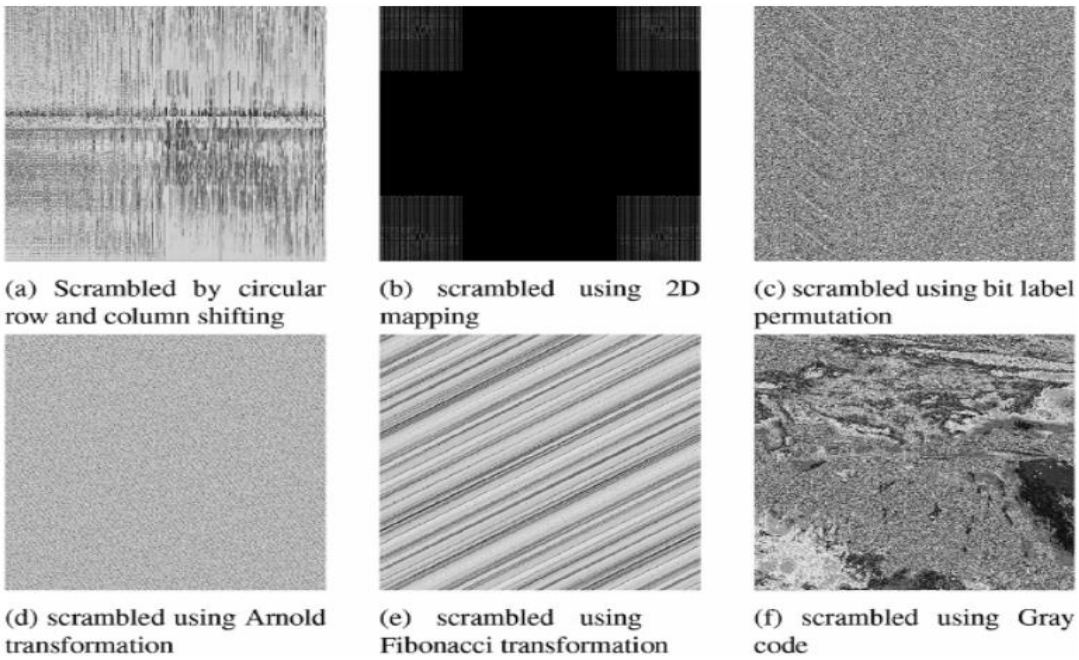
-> Row and Column Shifting with bit plane scrambling

-> 2D mapping

-> Generalized matrix based transformation

Eg: Original Image:

Scrambled Images using different algorithms:



(a) Scrambled by circular row and column shifting

(b) scrambled using 2D mapping

(c) scrambled using bit label permutation

(d) scrambled using Arnold transformation

(e) scrambled using Fibonacci transformation

(f) scrambled using Gray code

The selection of the scrambling technique can be chosen according to the application and requirements of further operations.

3. Autoencoders

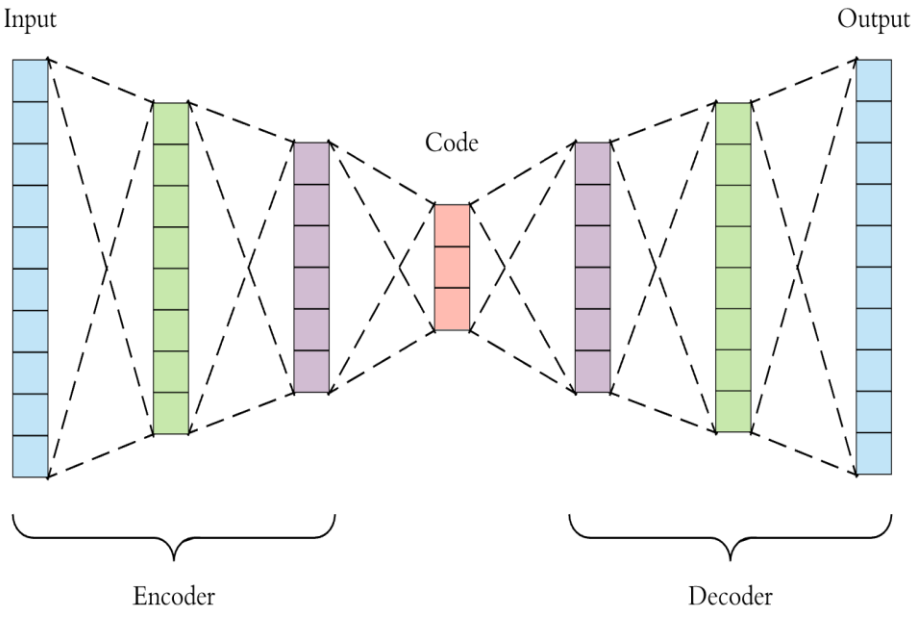
- An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction. Autoencoders have two parts: an encoder and a decoder.
- The aim of an autoencoder is to encode a given image into a feature vector and decode to it reconstruct the original image.

- Autoencoders are mainly used for noise reduction in images, creating super resolution images and constructing GANs (Generative Adversarial Networks).

- But we can employ the technique here to compress the image data before scrambling and ecrypting the image for speeding up the process exponentially and making it more secure!

- Although this sounds very fascinating, autoencoders do not perform 100% image reconstruction, i.e: the final image might be slightly different from the original image and hence the application of this concept is very limited.

- But we can surely hope that further research and development might lead us to our desired type of autoencoders for this purpose or creation of an entirely new technology.



7-REFERENCES

[i] "The Blowfish EncryptionAlgorithm"

Dr. Dobb's Journal, Vol. 19, No. 4, April 1994, pp. 38- 40, published by Bruce schneier

[ii] "Performance Enhancement of Blowfish Encryption Using RK-Blowfish Technique"

International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 9236-9244 ,published by B. Shamina Ross and V. Josephraj

[iii] "Enhancing Image Security by employing Blowfish Algorithm further embedding text and Stitching the RGB components of a Host Image" International Journal of Scientific & Engineering Research, Volume 4, Issue 7,July- 2013 ISSN 2229-5518, published by Deepanshu Agarwal, Pravesh Panwar, Purva Vyas, T.B. Patil andS.D.Joshi

[iv]"A New Chaotic Algorithm for Image Encryption and Decryption ofDigital ColorImages"
International Journal of Information and Education Technology, Vol. 1, No. 2, June 2011, published by K. Sakthidasan and B. V. Santhosh Krishna

[v] "BlowfishAlgorithm"
IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. X (Mar-Apr. 2014), PP 80-83, published by Ms NehaKhatri Valmik1 and Prof. V. K Kshirsagar

[vi] "Image Encryption And Decryption Using Blowfish Algorithm InMatlab"
International Journal of Scientific & Engineering Research, Volume 4, Issue 7, July- 2013ISSN 2229-5518, published by Pia Singh and Prof. Karamjeet

8. Source Code:

You can find our main source code and code for histogram analysis on this github link.

https://github.com/ma1VAR3/image-encryption-using-blowfish