# CHAPTER 1

# INTRODUCTION

With the rapid growth in digital technology and the growth in the usage of the internet, there is an unprecedented exchange of digital images nowadays. The popularity of the internet is growing day by day, and with this growth, digital technologies are simply soaring, and digital photographs are now shared at a pace that was previously unimaginable. Since important information is often contained in these pictures, strong security measures should be taken to protect the information from privacy, illegal access, and modification. The specific features of digital images, like high redundancy, enormous amounts of data, and profound correlations between neighboring pixels, make conventional ways of encryption like DES, AES, and RSA pretty unsuitable for image encryption, although quite effective in word and binary text. Effective management of the unique characteristics of digital photographs necessitates specialized methods of encryption.

One of the interesting methods is chaotic system image encryption. Chaos theory gives a location for predictable structures that are behaving randomly. Traditional methods of cryptography are not as fast compared to digital images. Therefore, fast encryption and decryption are required for real-time communication. Due to the nature of the image, the process of image-based encryption differs from text-based cryptography. For picture encryption, the pixels are regarded as the basic building blocks of an image. The location and values of the pixels in a picture are encrypted. Currently, picture encryption ought to be enough to stop these kinds of assaults. The encryption picture should be designed so that, after decryption, the initial picture may be recovered. A two-step iterative logistic technique is used to generate a chaotic sequence for encrypting the image. Sub-keys are used to alter the values and positions of pixels.

- It assigns a random location to the specific pixel.
- The cipher text has to be key-sensitive.

This enhanced picture encryption approach utilizes the chaotic properties of the logistic and Henon maps. This proposed scheme essentially comprises two parts: diffusion and permutation. Under the permutation phase, the pixels are rearranged based on the duffing

map, which ultimately destroys the relationship between the neighboring pixels. At the diffusion stage, logistic and Henon maps are used to alter the pixel values of the permuted image in a two-step iterative process.

## Need for Image-Specific Encryption Techniques

Since conventional cryptographic methods are not effective for digital images, specific encryption techniques are required. These should efficiently work on large bodies of data without fearing vulnerability to attack.

## Role of Chaos Theory in Image Encryption

The framework of image encryption with chaos theory can be considered excellent because of its transportation to the initial conditions and the nature being deterministic but unpredictable. In image processing, pixels are considered as the basic unit. Pixels in an image come in a two-dimensional form, and their values may be used in order to generate special patterns on the screen or in print. One can also revert the obtained 2-D array of pixels into an image.

## Two-Step Iterated Logistic Map

This is a proposed project on the adoption of a two-step iterated logistic map system, due to its simple and effective attributes with respect to image encryption for the first time. The characteristics, among others, include:

- **Mapping of Simple Chaotic Sequence:** Logistic map gives birth to a chaotic sequence with which communication will be encrypted.
- The process of encryption is basically done in 2 major steps, namely permutation and diffusion, for the algorithm to come about. Permutation is a step that rearranges the pixels, while diffusion changes the pixel values by a sequence of random numbers.
- Chaotic sequence sub-keys created are used to feed the encryption algorithm, which would guarantee that it is very sensitive to initial conditions, making it very hard to allow any attacker to decrypt the image without the right key.

## 1.1. Objectives, Scope and Purpose

### 1.1.1. Objectives

- This project is primarily looking at improving the safety of digital pictures, using chaos theory when developing encryptions, ensuring instantaneous operation and preservation of image state after decoding among others.
- In particular, a very secure algorithm has to be created whose strength would prevent an outsider from trying to open or decode them.
- The main feature of the project is the use of chaotic maps such as the two-step iterated logistic map to exploit their properties of high initial condition sensitivity and unpredictability in secure image encryption.

### 1.1.2. Scope

This encryption technique is used to several digital picture formats, most notably gray scale and color photos, which are utilized in a variety of fields including personal pictures, military purposes, and healthcare. Because of this suggested cryptographic scheme's extreme resistance to common cryptanalysis methods, such as brute force, statistical, and differential attacks, the security of the encrypted pictures will be guaranteed for an extended length of time.

### 1.1.3. Purpose

Sensitive data must be protected from unwanted access and potential exploitation in this project, and digital imagery such as military intelligence and medical records must be properly encrypted. The project will thereby raise user confidence in a system for digital communication and strengthen the strategies for protecting privacy and the accuracy of digital photographs. The study looks at chaotic maps for image encryption, which advances both the processing of images and cryptography. With any luck, it will stimulate more investigation into robust encryption methods utilizing different chaotic maps.

## 1.2. Project Description

In this project, a two-stage iterative logistic map based on Chaos Theory is used to construct a new picture encryption method. As the exchange of digital information through the internet is ever increasing, sensitive data like medical and military images should be secured. The conventional cryptographic algorithms, like AES, IDEA, and RSA, are not satisfactory for the treatment of digital images with bulk data and high correlation features requiring high computational power and real-time processing capability. The suggested method increases the security of picture encryption by taking use of chaos' natural properties. Specifically, great sensitivity to beginning circumstances and unpredictable behavior has been used.

The image is first permuted and then encrypted as part of the single-scan encryption technique that is described here. Two-step iterative logistic maps have been employed in this iteration procedure. The encrypted image is significantly altered by minor modifications in the key or the beginning circumstances because two-step iterative logistic maps can offer a huge key space. Other than improved security, this approach offers superior performance when compared to various other strategies available to achieve the same goal. In order to prevent unauthorized parties from readily predicting or reconstructing the original image, the created chaotic sequence is utilized to change the pixel locations and values during this process. The suggested method performs better and offers greater security, according to experimental results, demonstrating its suitability for real-world uses, including the safe transfer of private digital photos.

## 1.3. Problem Statement

Chaos has topological transitivity and sensitive initial condition properties that make it suitable for cryptosystems. The cryptographic system differs from the chaotic system in that, say, a cryptographic operates on integers of a large order of finitude while chaotic systems operate on real numbers.

Topological transitivity is used for diffusion in encryption, and the initial conditions act as keys. There the beginning conditions serve as keys in encoding, which use topology transitivity for dissemination. Stream and block ciphers can therefore be implemented using a chaotic system. As a result, blocking and stream encryption can both use a chaotic system. The complexity of the Brute Force assault will be attributed to its sensitivity to beginning conditions, and the chaotic sequence is very sensitive to these settings Initial values in chaos maps, the chaotic cryptography technique originates from the input picture's word streams; concurrently, the resulting bit streams create a cipher image. A tiny variation in the input causes an enormous modification in the final result after a few cycles. Furthermore, the cipher picture changes dramatically in response to even a little alteration in the key value.

# CHAPTER 2

# LITERATURE SURVEY

## [1] Priya R., Sankalp, and P a Vijaya, "Image Encryption using chaotic maps: A Survey" 5 International Conference on Signal and Image Processing, 2014, 10.1109/ICSIP.2014.80.

This paper applies the complicated Chao maps in its process. It involves two different phases: a phase of uncertainty and a phase of filtering. In the first case, initial the parameter is used to create secret key. Some problems are identified to be selected for the diffusion stage; one chaotic system will be employed for the diffusion, another one for confusion, and likewise, the third-dimensional chaotic system shall also apply for confusion.

## [2] S A. Soleymani, Z. Md Ali, and Md. J. Nordin," A Survey on Principal Aspects of Secure Image Transmission", World Academy of Science, Engineering and Technology 66 2012, pp. 247 – 254.

They proposed a two-unit encryption strategy, which creates a stream cipher and uses the other unit for pixel permutation. In that technique, the W7 method produces the main stream, while the Henton map produces the pseudo-random sequence. Here, there is the change in place as well as in value at a given time. By using a permutation map, pixel shuffling can reduce the correlation coefficient. The stream chosen and permuted picture are XORed to get the encrypted picture.

## [3] N.K. Pareek, Vinod Patidar and K.K. Sud, "Image Encryption using chaotic logistics Map", Image and Vision Computing Volume 24, Issue 9, 1 September 2006, Pages 926–934

In this, significant mapping tables based on logistic maps were constructed, which performed pixel value conversions without any change in the pixel's locational value. It applies two techniques: the first is applying a map table and a random value to map the values of the pixels, and the second is applying a map table generated from low-processed pixel values. The second is the XOR action, with the help of sequences generated by the logistic

map and pixels. Then, the procedure described above can be reversed, and the encrypted picture can be decrypted.

## [4] K. Gupta, S. Silakari, "New Approach for Fast Color Image Encryption Using Chaotic Map", JIS, 2011, 2, 139-150.

Gupta and S. Silakari proposed a method in which, from an original image, three colorful images are generated: those primary colors that are red, green, and blue; the green and red images are decolored in horizontal and vertical planes, respectively, while the blue image stays intact. As for the first level of confusion, a map of a cat in 2D is applied. From each of the three photos, rows are selected, and as this is being done, each of the selected rows appears to be confused.

## [5] Xu Shu-Jiang, Wang Ji-Zhi and Yang Su-Xiang, "An improved image Encryption Algorithm Based on Chaotic Maps" 2008 Chin. Phys. Soc. and IOP Publishing Ltd Chinese Physics B, Volume 17, Number 11.

A new encryption algorithm proposed by Xu Shu Jiang, Wang Ji, and Su-Xiang Yang used chaotic maps to enhance uncertainty and distribution. The Lorenz system and Baker map are used to change the coordinates of points and the intensities of individual pixels. Hence, confusion is not diffusion and requires another key. At the receiver end, the inverse process is applied to reconstruct the original image from the encrypted signal.

## 2.1. Existing System and Proposed System

### 2.1.1. Existing System

The system developer for image encryption intends to draw from a discrete array of cryptographic traditional algorithms, of which the most famous are AES, IDEA, and RSA. It has also got several limitations while working with digital images by the algorithms because of the bulky data and high-correlated features that make them less suitable for the purpose of encryption and decryption. These algorithms are slow in the case of digital images. Therefore, there is a dire need for developing high-speed algorithms for real-time communications in the processes of image encryption and decryption.

### 2.1.2. Proposed System

It is proposed that the system be designed with a view to rectifying the limitations of the existing system in just a single scan through image encryption. The sub-keys can be used on pixel values while modifying the pixel positions at pixel level in order to produce a chaotic sequence, which is realized via the logistic map. The proposed system combines elements of traditional cryptography with the power of chaos-based encryption. The Two-Step Logistic Map provides an extra layer of security, making the encryption process much more robust and resistant to attacks. This approach effectively addresses the limitations of traditional methods while introducing enhanced security features.

## 2.2. Tools and Technologies

### 2.2.1. Tools

- **Logistic Map:** Logistic-based generation of the chaotic sequence in two-step iteration is considered the primary tool, and this sequence played a vital role in the permutation and diffusion processes of encryption. In other words, the logistic map is ruled out by $Xn+1=r \cdot X\ n \cdot (1- X\ n)$, where $0 < r < 4$ and $0 < X < 1$.

- **Chaos Theory:** In this way, properties such as sensitivity to initial conditions, periodical transitivity, and non-sensitive topological transitivity of the system are exploited to make the encryption more secure. Chaotic maps will be used at once for processes of confusion and diffusion.

- This means that the sub-keys also participate in the process of diffusion to change pixel values. Therefore, encrypted images are very sensitive to the key and, hence, more secure.
- During the process of encryption, the permuted image and key stream are linked with the XOR operation. Simultaneously, the opposite happens in decryption for the retrieval of the original image.
- **Permutation and Diffusion:** The algorithm presented is thus, in some sense, a two-step process: permutation of the image pixel positions with the chaotic sequence and pixel value diffusion using sub-keys and the chaotic sequence.
- **Digital Signal Processing:** In this project, DSP techniques shall be used for the required computational power during the real-time process of digital image encryption.

### 2.2.2. Technology:

**Mat Lab:**

Matrix Laboratory, or MATLAB for short, is an interactive environment and high-level programming language created by Math Works. It is renowned for its powerful numerical computing capabilities, particularly in matrix computations, which makes it ideal for tasks involving linear algebra and Fourier analysis. Aside from numerical computation, MATLAB offers a wide range of data analysis and visualization capabilities, enabling users to create detailed plots, graphs, and other visual representations effortlessly. This makes it a popular choice for researchers and engineers who need to analyze and visualize large datasets.

## 2.3. Software and Hardware Requirement

### Hardware Requirement:

- Processor : Intel i3
- RAM : 64GB
- Input Device : Keyboard
- Output Device : Screens of Monitor or a Laptop

### Software Requirement:

- Operating system : Windows 10 and more
- Tool : Mat Lab R2018a version
- Plat Form : Mat Lab.

# CHAPTER 3

# SOFTWARE REQUIREMENT SPECIFICATION

## 3.1. User Flow

1. **User Input:**

    - The user selects the gray scale image file for encryption.
    - User provides an encryption key of 144-bit.

2. **Initialization**:

    - The system uses a comprehensively adjusted logistic map to initialize the chaotic sequences.
    - It generates initial values for the logistic map based on the input key.

3. **Permutation Stage**:

    - The pixel of the image is permuted based on the chaotic sequence generated by the logistic map.
    - Chaotic map equations are used to calculate new pixel positions.

4. **Diffusion Stage**:

    - Diffusion: The permuted image undergoes diffusion, which involves changing pixel values.
    - These sub-keys, which are obtained from the initial key, are used to have high sensitivity to the key and perform an XOR operation with pixel values.

5. **Encryption Output**:

    - The permutation and diffusion steps construct the final encrypted image.
    - The encrypted image is then displayed or saved according to the user's will.

6. **Decryption Process**:

    - The encrypted image is taken as input.
    - In this process, the same chaotic sequences are used with the initial key.
    - Inverse diffusion, followed by inverse permutation, is done by the system to retrieve the original image.

7. **Decryption Output**:

    - Finally, the decrypted image is presented to the user, which proves that both encryption and decryption were successful.

## 3.2. Functional Requirement

1. **User Interface:**

    - Provide an interface for users to upload and select an image for encryption.
    - Allow users to input a 144-bit encryption key.
    - Display the encrypted image to the user.
    - Provide an option for users to download the encrypted image.

2. **Encryption Process:**

    - Initialize the encryption key and use it to derive initial values for the chaotic map.
    - Implement the permutation stage using a modified logistic map to rearrange the pixels.

3. **Decryption Process:**

    - Reverse the encryption process using the same initial key and chaotic sequences.
    - Perform the inverse diffusion to retrieve the permuted image.
    - Perform the inverse permutation to retrieve the original image.

4. **Performance and Security:**

    - Ensure that the encryption and decryption processes are performed efficiently.
    - Implement measures to protect against brute-force attacks, ensuring a large key space.

5. **System Requirements:**

    - The system should support various image formats for input and output.
    - It must be able to handle large images without degradation in performance.

## 3.3. Non Functional Requirement

1. **Performance:**

    - The algorithms for encryption and decryption must be designed such that the latency is minimum and the processing time is as fast as possible for both small and large images.

2. **Scalability:**

    - The system has to be designed in a way that scaling happens on-demand and has no performance problems with the high number of requests binding it simultaneously to encryption and decryption.

3. **Security:**

    - Enough key space to prevent brute-force attacks one needs at least 144-bit keys for more sound security.
    - Differential analysis: a small change either in the plain image should result in enormous changes in the cipher image; that is, it should be resistant to differential attacks.

4. **Reliability:**

    - The system shall support reliable encryption and decryption. Integrity and accuracy of the data shall be mainstays in the output images.

5. **Usability:**

    - The user interface shall be friendly and intuitive so that users may navigate to either encrypt or decrypt an image.

6. **Maintainability**:

    - The system shall follow modular design and clear documentation for easier maintenance and update.

# CHAPTER 4

# SYSTEM DESIGN

## 4.1. System Architecture

This image is transformed using a duffing map. The image in question has undergone duffing map alteration using each of the values from its hidden key. Both values derived from the secret key represent the beginning variables of the Henon map. By XORing each pixel value with the array generated from the Henon map values, one diffusion cycle is achieved. Starting locations on the map of Henon. All pixel values are XORed with an array generated from the Henon mapping values to get one diffusion iteration. The logistic map's starting value is represented by the following value in the Henon map that is produced. Consequently, the following stage of transmission is driven by the urge to replicate information, as demonstrated. One chooses the subsequent quantity of the Henon map for the other picture in order to obtain the logistic map's beginning value.

Encryption process has been illustrated in figure 1 while decryption is the reversal of the encryption process illustrated in figure 2.

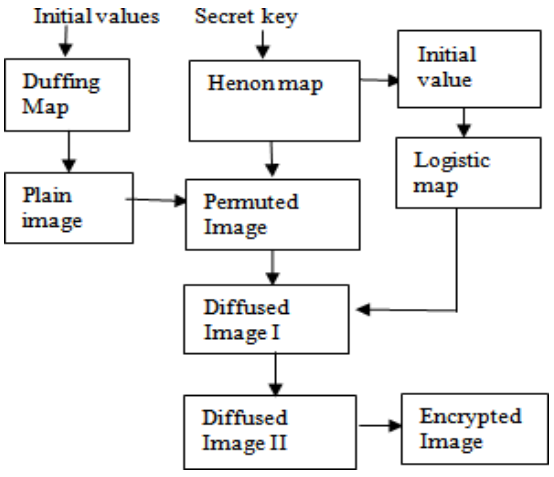


**Figure 4.1.1: Encryption Process**

1. **Initial Values and Duffing Map**:

   - **Initial Values**: These are starting values required for generating chaotic sequences.
   - **Duffing Map**: A chaotic map that generates initial chaotic sequences using the initial values.
   - **Plain Image**: The Original image that needs to be encrypted.

2. **Permutation Stage**:

   - The chaotic sequences generated by the Duffing map are used to permute the pixels of the plain image, resulting in a **Permuted Image**.

3. **Diffusion Stage I**:

   - **Secret Key and Henon Map**: A secret key is used to initialize the Henon map, another chaotic map.
   - The Henon map generates a chaotic sequence that is used to perform the first diffusion, resulting in **Diffused Image I**.

4. **Logistic Map**:

   - A logistic map, defined by $X_{n+1} = r \cdot X_n \cdot (1 - X_n)$, is used to further permute the pixels of the Diffused Image I.

5. **Diffusion Stage II:**

   - The permuted image from the Logistic map undergoes a second diffusion using the chaotic sequence, resulting in **Diffused Image II**.

6. **Final Encrypted Image:** The final output will be encrypted Image.

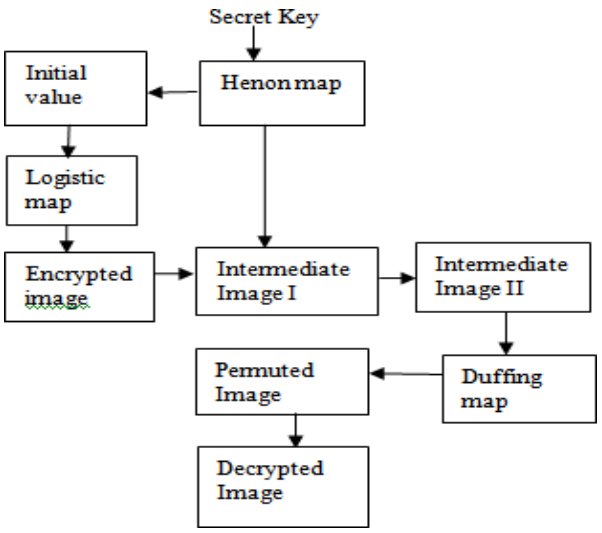


**Figure 4.1.2: Decryption Process**

## 1. Secret Key and Henon Map:

- Initial Value: The same initial value was used during encryption.
- Henon Map: The secret key is used to initialize the Henon map, generating the same chaotic sequences used in the encryption process.

- **Logistic Map:**

- The image is first processed through the Logistic map to generate Intermediate Image I.

## 2. Inverse Diffusion Stage II:

- The Intermediate Image I undergoes inverse diffusion to generate **Intermediate Image II**.

## 3. Inverse Permutation Stage:

- The Duffing map, initialized with the same initial values, is used to reverse the permutation of pixels, resulting in a **Permuted Image**.

### 4. Inverse Diffusion Stage I:

- The Permuted Image undergoes inverse diffusion using the Henon map sequences to retrieve the **Decrypted Image**, which should be the original plain image.

## 4.2. Algorithm

### Chaos-based image Algorithm:

**Step 1: Generation of Sequences Using Duffing Map**

**Duffing Map:** A nonlinear second-order differential equation used to generate chaotic sequences.

**Generate Sequences k and l:** Use the Duffing map to generate two sets of sequences, k and l.

These sequences are:

$K = \{k1, k2 ,\ldots, km\}$

$l = \{l1, l2\ldots ln\}$

**Step 2: Permutation Using Sorted Sequences**

**Sorting Sequences:** Sort the sequences k and l.

**Permutation:** Use the sorted sequences to rearrange (permute) the pixels of the image.

The pixel at position (i, j) in the original image I(i, j) is moved to a new position (k, l):

$I(i, j)=I(k, l)$

**Step 3: Diffusion Using Henon Map**

The Henon map sequence is used to dilute the permuted picture. The starting values and secret keys $a$, $d$, and others are selected so that (0) and (0) produce the Henon map sequence.

**Step 4: Further Diffusion Using Logistic Map**

Henon map ($i$), where $\boldsymbol{U}$ is, is used to select the initial value (0).

Further Diffusion: Use the logistic map sequence to further diffuse the image obtained from Step 3.

**Step 5: Encrypted Image**

This second diffuse image gotten from the application of logistic map will be the final encrypted image, C (i, j).

- Derive random-like and noisy signals with the help of the Duffing map.
- Dispersing the permuted image utilizing the Henon map.
- Disperse the image even more with the help of the logistic map.
- Thereby it fulfills the desired output, which is the encrypted image.

# CHAPTER 5

# DETAILED DESIGN

## 4.1. Data Flow Diagram

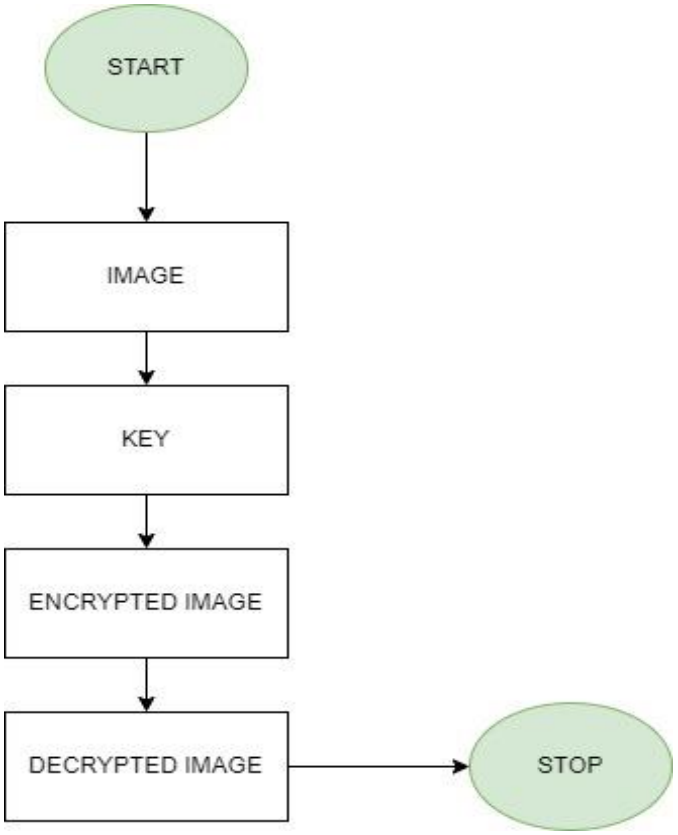


**Figure 5.1.1: Image Encryption and Decryption**

### 4.1.1. Encryption Process

1. **Initial Values and Duffing Map:**

- **Initial Values**: These are starting values required for generating chaotic sequences.
- **Duffing Map**: A chaotic map that generates initial chaotic sequences using the initial values.
- **Plain Image**: The original image that needs to be encrypted.

2. **Permutation Stage:**

    - The chaotic sequences generated by the Duffing map are used to permute the pixels of the plain image, resulting in a **Permuted Image**.

3. **Diffusion Stage I:**

    - **Secret Key and Henon Map**: A secret key is used to initialize the Henon map, another chaotic map.
    - The Henon map generates a chaotic sequence that is used to perform the first diffusion, resulting in **Diffused Image I**.

4. **Logistic Map:**

    - A logistic map, defined by $X_{n+1}=r \cdot X_n \cdot (1-X_n)$, is used to further permute the pixels of the Diffused Image I.

5. **Diffusion Stage II:**

    - The permuted image from the Logistic map undergoes a second diffusion using the chaotic sequence, resulting in **Diffused Image II**.

6. **Final Encrypted Image:**

    - The final output of the encryption process is the **Encrypted Image**.

### 4.1.2. Decryption Process

1. **Secret Key and Henon Map:**

    - **Initial Value**: The same initial value used during encryption.
    - **Henon Map**: The secret key is used to initialize the Henon map, generating the same chaotic sequences used in the encryption process

2. **Logistic Map:**

    - The encrypted image is first processed through the Logistic map to generate Intermediate Image I.

3. **Inverse Diffusion Stage II:**

    - The Intermediate Image I undergoes inverse diffusion to generate **Intermediate Image II**.

4. **Inverse Permutation Stage:**

    - The Duffing map, initialized with the same initial values, is used to reverse the permutation of pixels, resulting in a **Permuted Image**.

5. **Inverse Diffusion Stage I:**

    - The Permuted Image undergoes inverse diffusion using the Henon map sequences to retrieve the **Decrypted Image**, which should be the original plain image.

# CHAPTER 6

# IMPLEMENTATION

## 6.1. Coding

```
function varargout = recovermsg(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton,...
                   'gui_OpeningFcn', @recovermsg_OpeningFcn, ...
                   'gui_OutputFcn',  @recovermsg_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 recovermsg_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = recovermsg_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
 [fn,fp]=uigetfile('Select Image to Encrypt');
fpath=strcat(fp,fn);
set(handles.edit1,'String',fpath);
img=imread(fpath)
axes(handles.axes1)
imshow(img)
title('Original Image')
 function pushbutton2_Callback(hObject, eventdata, handles)
fname=get(handles.edit1,'String');
A=imread(fname);
K=10001;
lambda=2;
IK=mod(sum(sum(sum(A))),K);
MK=IK/K;
handles.MK=MK;
handles.IK=IK;
S=zeros(size(A,1),size(A,2),3);
for i=1:2
   su(i)=sum(sum(A(:,:,i)));
end
 for k=1:2
       n=mod(su(k),6)+1;
       switch n
          case 1
             xl=0.1*MK;
             for i=1:(IK+size(A,k)-1)
                xl(i+1)=4*xl(i)*(1-xl(i));
             end
             for i=1:size(A,k)
                x(1,i)= xl(IK+i-1);
             end
          case 2
             xc=0.1*MK;
```

```
for i=1:(IK+size(A,k)-1)
            xc(i+1)=cos(5*acos(xc(i)));
        end
        for i=1:size(A,k)
            x(2,i)= xc(IK+i-1);
        end
    case 3
        xg=0.1*MK;
        for i=1:(IK+size(A,k)-1)
            xg(i+1)=exp(-4.9*xg(i)*xg(i))+(-0.58);
        end
        for i=1:size(A,k)
            x(3,i)= xg(IK+i-1);
        end
    case 4
        xl=0.1*MK;
        for i=1:(IK+size(A,k)-1)
            xl(i+1)=4*xl(i)*(1-xl(i));
        end
        for i=1:size(A,k)
            x(4,i)= xl(IK+i-1);
        end
    case 5
        xb=0.1*MK;
        for i=1:(IK+size(A,k)-1)
            xb(i+1)=mod(2*xb(i),1);
            if xb(i+1)==0
                xb(i+1)=0.1*MK;
            end
        end
        for i=1:size(A,k)
            x(5,i)= xb(IK+i-1);
        end
```

```
 case 6
            xs=0.1*MK;
            for i=1:(IK+size(A,k)-1)
               xs(i+1)=0.9*sin(pi*xs(i));
            end
            for i=1:size(A,k)
               x(6,i)= xs(IK+i-1);
            end
      end
            switch k
               case 1
                  for i=1:size(A,1)
                     S(i,1,1)=x(n,i);
                  end
               case 2
                  for i=1:size(A,1)
                     S(1,i+1,1)=x(n,i).';
                  end
            end
 end
for k=1:3
   for j=1:(size(A,1)-1)
      for i=2:(size(A,2)+1)
         S(j+1,i,k)=        ((1-0.2)*(1-(lambda*S(j,i,k)*S(j,i,k))))+        (0.2*(1-
(lambda*S(j,i-1,k)*S(j,i-1,k))));
      end
   end
   S1(:,:,k)=S(:,2:(size(A,2)+1),k);
end
S1=mod(round(S1*(10^14)),256);
xh=0.1*MK; yh=0.1*MK;
for i=1:(max(size(A,1),size(A,2))+IK)
   xh(i+1)=yh(i)+1-(1.4*xh(i)*xh(i));
```

```
  yh(i+1)=0.3*xh(i);
end
for i=1:size(A,1)
    xh1(i)= xh(IK+i-1);
end
 for i=1:size(A,2)
    yh1(i)= yh(IK+i-1);
end
[temp,index1]=sort(abs(xh1));
[temp,index2]=sort(abs(yh1));
for k=1:3
    for i=1:size(A,1)
        for j=1:size(A,2)
            A1(i,j,k)=A(index1(i),index2(j),k);
        end
handles.su=su
encrypt=bitxor(A1,uint8(S1));
imwrite(encrypt,'encrypt1.tiff');
img=imread('encrypt1.tiff')
axes(handles.axes2)
imshow(img)
title('Encrypted Image')
 guidata(hObject,handles)
function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function pushbutton3_Callback(hObject, eventdata, handles)
 [fn,fp]=uigetfile('Select Image to Encrypt');
fpath=strcat(fp,fn);
set(handles.edit2,'String',fpath);
```

```
img=imread(fpath)
axes(handles.axes3)
imshow(img)
title('Encrypted  Image')
function pushbutton4_Callback(hObject, eventdata, handles)
fname=get(handles.edit2,'String');
A=imread(fname);
MK=handles.MK
IK=handles.IK
su=handles.su
K=10001;
lambda=2;
MK=IK/K;
disp('decryption')
imwrite(decrypt1,'recovered_image.jpg')
img=imread('recovered_image.jpg')
axes(handles.axes4)
imshow(img)
title('Retrieved Image')
fname=get(handles.edit1,'String');
img1=imread(fname)
[peaksnr, snr] = psnr(A, img);
set(handles.edit4,'String',peaksnr);
set(handles.edit5,'String',snr);
function edit4_Callback(hObject, eventdata, handles)
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)
end
```

# CHAPTER 7

# RESULTS

## 7.1. Screenshots

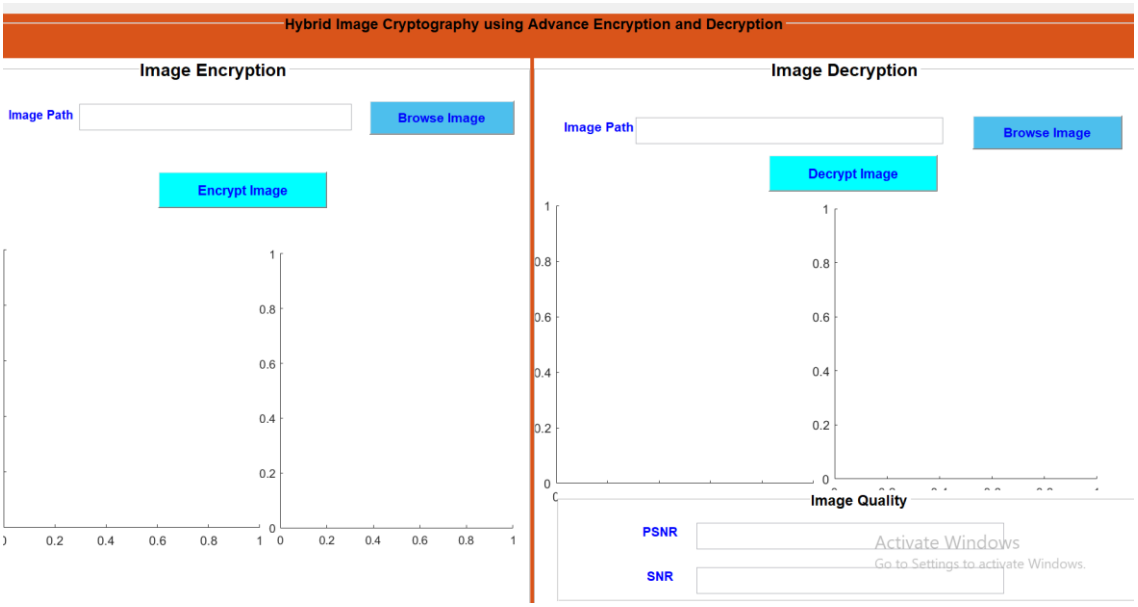


**Figure 7.1.1: Encryption and Decryption Process**

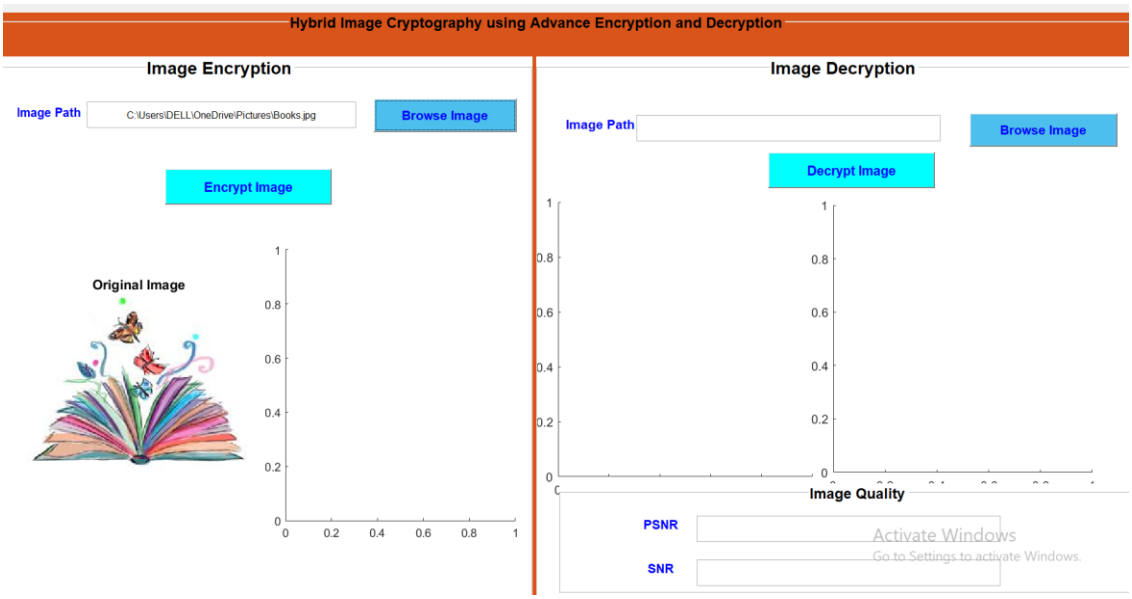


**Figure 7.1.2: Original Image**

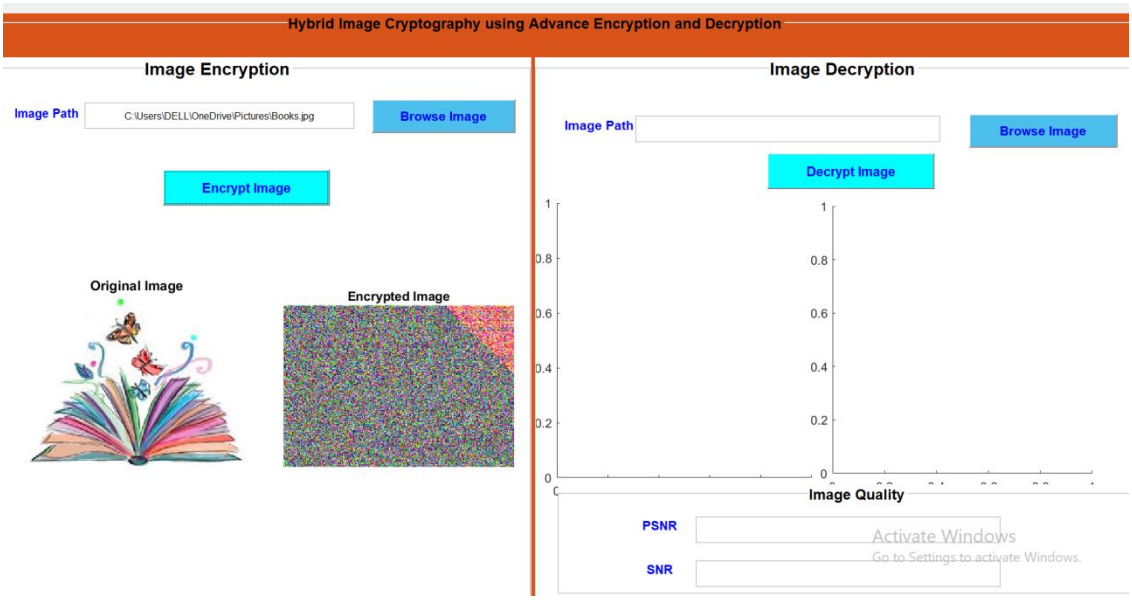


**Figure 7.1.3: Encryption Process**

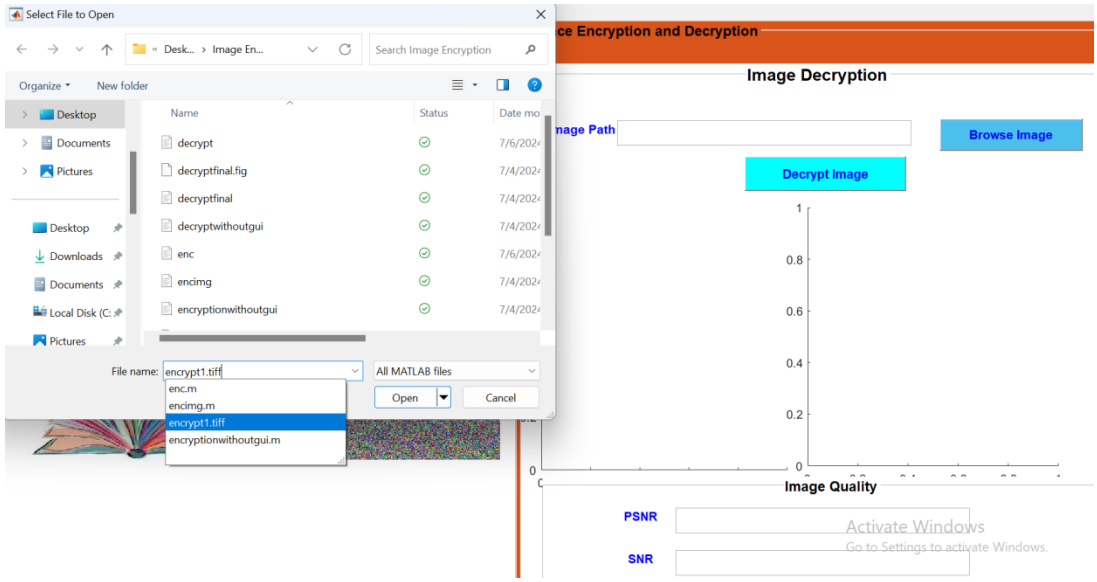


**Figure 7.1.4: Decryption Process**


**Figure 7.1.5: Decryption Process**

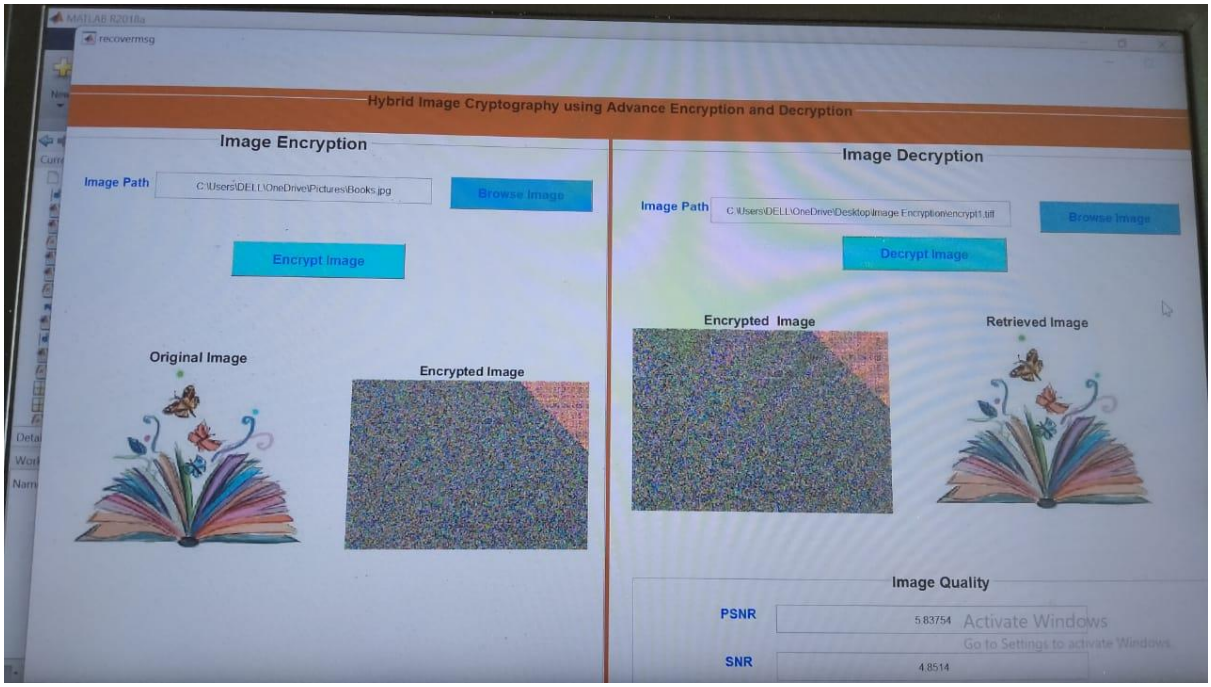


**Figure 7.1.6: Retrieved Image**

# CHAPTER 8

# SOFTWARE TESTING

## Purpose of Testing

It provides a detailed evaluation of the requirements design, that is, the code, and is an integral part of any software quality management. In particular, system testing is an activity of prime importance to this procedure since it verifies the functionality and behaviors of the software under test in a replicated setting. After all, testing is a fundamental step that serves up insightful details and places abnormalities in the limelight to attract attention.

### Verification

Verification ensures that the product does what it is supposed to do and conforms to specifications laid out at the start of the design process. It essentially serves to check whether the product works as projected and satisfies relevant needs. Testing, thus, ensures something or a program does exactly what it is designed for and meets performance requirements.

### Validation

Testing at the end of the design process is to make sure that the final item produced meets the criteria. To put it simply, it guarantees that something is built in compliance with the needs of the client.

## Types of Testing

- **Black Box Testing**

  Applications is tested for functioning using a method called "black box" testing, in which the underlying code structure, technical specifics, and internal routes of the program are not known. The emphasis is on input and output in this case, and it is examined if the program acts in accordance with the required specifications.

- **White Box Testing**

  Evaluation of software using the "white box" approach allows the user to examine the inner architecture, code, and development of the program. Ensuring that internal activities run as planned might be the goal. It has to do with checking that inputs flow through the code which includes checking for pathways, branching, and loops.

## Levels of Testing

- Unit Testing.
- Integration Testing.
- Functional Testing.
- System Testing.
- Stress Testing.
- Performance Testing.

1. **Unit Testing**: Unit testing is the verification process for every single software unit or component. The present study will employ unit evaluation for all functions, including logistic maps, henon maps, and duffing maps, that are integral to both of these processes.
2. **Integration Testing**: This testing phase ensures that different modules or services used by the application work together in fine cooperation. For example, permutation and diffusion processes would have to be integrated and tested so that smoothness is assured in terms of the flow and interaction of data.
3. **System Testing**:. In the project, it would involve system testing that checks whether, from input to output, everything in between is working correctly in all scenarios of encryption and decryption.
4. **Performance Testing**: Through testing, the performance of the program various workloads is determined. Performance assessments will be utilized in this project to evaluate both of these procedures' speed and effectiveness to make sure they can handle large picture files quickly.
5. **Security Testing**: Encryption initiatives must prioritize this in order to guarantee system security against intrusions and weaknesses. Therefore, security testing would

be necessary to make sure the encryption technique is robust and resistant to a variety of assaults, ensuring the confidentiality and integrity of the photos.

6. **User Interface Testing**: This evaluation will guarantee that the software's user interface operates and displays information in an easy-to-read manner. This has to be verified that all buttons, input fields, and graphical components function flawlessly and that the user experience is designed to be seamless and straightforward.

## 8.1. Test Cases

| Test Case | Test Case Description | Steps to Execute | Expected Result |
|---|---|---|---|
| TC1 | Validate logistic map generation | 1. Call logistic map function with seed value.<br>2. Generate chaotic sequence. | Chaotic sequence generated with values between 0 and 1. |
| TC2 | Validate image encryption process | 1. Load input image.<br>2. Execute complete encryption process. | Encrypted image generated with visually unrecognizable content. |
| TC3 | Validate image decryption process | 1. Load encrypted image.<br>2. Execute complete decryption process. | Decrypted image matches the original input image. |
| TC4 | Validate encryption and decryption speed | 1. Load input image.<br>2. Measure time taken for encryption.<br>3. Measure time taken for decryption. | Encryption and decryption complete within acceptable time limits (e.g., less than 1 second for standard image size). |
| TC5 | Validate user interface for image upload functionality | 1. Open the application.<br>2. Upload an image through the UI. | Image is successfully uploaded and displayed in the application. |

| | | | |
|---|---|---|---|
| TC6 | Validate error handling for invalid image format | 1. Open the application.<br>2. Attempt to upload an invalid image format. | Application displays an error message indicating invalid image format. |
| TC7 | Validate system security against brute-force attacks. | 1. Attempt brute force attack on encrypted image.<br>2. Measure the time and resources required to succeed. | Encryption withstands brute force attacks, requiring impractically long time/resources to break. |
| TC8 | Validate compatibility across different operating systems. | 1. Install application on different OS.<br>2. Run encryption and decryption tests. | Application performs consistently across all tested operating systems. |
| TC9 | Validate system's behavior with edge case inputs (e.g., very small or very large images) | 1. Load edge case images.<br>2. Execute encryption and decryption processes. | System handles edge cases without crashes, producing correct encrypted and decrypted images. |
| TC10 | Validate system's acceptance criteria | 1. Review all project requirements.<br>2. Execute final acceptance tests. | The system meets all specified requirements and performs as expected in real-world scenarios. |

**Figure 8.1.1: Test Cases**

# CHAPTER 9

# FUTURE ENHANCEMENT

This project include the integration of Service with cloud storage, enhanced security features, Multi-Factor Authentication, secure key management, and granular error handling to leave no loose ends. Further work in enhancing robustness against errors and attacks that are cryptanalytic in nature will ensure system reliability. This will also enhance the usability if a user-friendly interface is developed, cross-platform compatibility ensured, and integration into cloud services accorded. Investigation of new chaotic maps, hybrid cryptographic techniques, and implementation of performance metrics and analytics will let the system remain secure, efficient, and effective.

# CHAPTER 10

# CONCLUSION

This proposed encrypted image solution is directly dependent on whether the diffusion and confusion processes are changed in a revised logistic map. By doing so, this approach shall make the chaotic sequence much safer by using a two-step logistic map. Hybrid image cryptography via the two-step logistic map has thus been a key advantage over traditional methods; it is robust and secure in protecting sensitive visual data. In fact, it has combined the strengths of conventional cryptography with the power of chaos-based encryption in order to yield an effective and resilient system.

# REFERENCES

[1] Priya R., Sankalp, and P a Vijaya, “Image Encryption using chaotic maps: A Survey” 5 International Conference on Signal and Image Processing, 2014, 10.1109/ICSIP.2014.80.

[2] S A. Soleymani, Z. Md Ali, and Md. J. Nordin, “A Survey on Principal Aspects of Secure Image Transmission”, World Academy of Science, Engineering and Technology 66 2012, pp. 247 – 254.

[3] N.K. Pareek, Vinod Patidar and K.K. Sud, “Image Encryption using chaotic logistics Map”, Image and Vision Computing Volume 24, Issue 9, 1 September 2006, Pages 926–934

[4] K. Gupta, S. Silakari, “New Approach for Fast Color Image Encryption Using Chaotic Map”, JIS, 2011, 2, 139-150.

[5] Xu Shu-Jiang, Wang Ji-Zhi and Yang Su-Xiang, “An improved image Encryption Algorithm Based on chaotic Maps” 2008 Chin. Phys. Soc. and IOP Publishing Ltd Chinese Physics B, Volume 17, Number 11.

[6] Sankaran K.S and Krishna B.V.S., “A New Chaotic Algorithm for Image Encryption and Decryption of Digital Color Images”, IJIET, Vol. 1, No. 2, June 2011.

[7] Xing Yuan Wang, Jianfeng Zhao, Hongjun Liu, “A New Image Encryption Algorithm Based On Chaos”, Optics Communications Volume 285, Issue 5, 1 March 2012, Pages 562 566.

[8] Zhi-Hong Guano, Fang Jun Huanga, Wenjie Guanb, “Chaos-based image encryption algorithm” Physics Letters a Volume 346, Issues 1 3, 10 October 2005, Pages 153–157.

[9] Di Xiao, Xi Liaob, P. Weib, “Analysis and Improvement of a Chaos Based Image Encryption Algorithm”, Chaos, Solitons & Fractals Volume 40, Issue 5, 15 June 2009, Pages 2191–2199.

[10] Xing Yuan Wang, Jianfeng Zhao, Hongjun Liu, “A New Image Encryption Algorithm Based on Chaos”, Optics Communications Volume 285, Issue 5, 1 March 2012, Pages 562 566.