

1. INTRODUCTION

The rapidly developing technology has given many new benefits. These technologies are aimed at making our lives easier by making communication easy. But like every other thing, technology has its pros and cons. Many users try to make use of the data available on the internet for their own personal gain and malicious purposes. To protect user data and privacy from such malicious users, certain security measures are necessary.

Communication is sharing of information between two or more parties. This sharing can be in the form of sentences, data exchange or multimedia exchange like image exchange. Popular forms of communication include exchange or sharing of image by social media or electronic mail (e-mail). Therefore, security measures should include protection of user information and protection of user data. Protection of the images a user is sharing becomes a very vital part of protection of user data.

But why is protection of user shared images necessary? It is to protect the user itself: a malicious user can gain information regarding an organization by accessing his pictures itself. Images can hold sensitive data which must not be shared by external sources.

1.1 Classification of Image Security

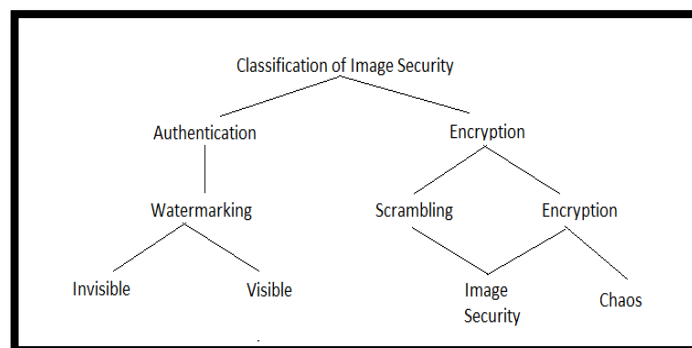


Figure 1.1 – Classification Image Security

Image Security can be classified into the following parts –

1. **Authentication** – Authentication involves verifying the validity of the truthfulness of a particular image. One of the methods of image authentication is Watermarking. Watermarking is a the procedure of leaving a recognizable design on an image, which can be used to determine its source.

Watermarking can be of the following forms:-

- A. Invisible – When the watermark is not visible in the image through naked eye.
- B. Visible – When the watermark is visible directly in the image.

2. **Encryption** – One of the most important methods of image security is image encryption. Encryption means hiding of data so that the actual value is hidden from an attacker. Image encryption includes methods to hide image information, such that the actual value is not clear to the user. Its purpose is to create confusion and diffusion. Encryption is needed to maintain confidentiality of the data and the user. Scrambling is a method in encryption where the pixels or the positions of the images are shuffled so as to create diffusion and confusion in the image. The shuffling is done using a key. Key management is an important part of image scrambling. Image scrambling can be used along with various other techniques to create a more stronger encryption technique. Chaos theory is another way of image encryption.

It has become very important to provide security to digital images over the internet. With the advent of the internet networks, systems and data transmitted over the network from one place to another are highly vulnerable to attack by malicious hackers. Image encryption has been used for this purpose to transmit data across the network without the attacker being able to see the actual content of it as it is encrypted. Several algorithms have been devised by researches to tackle this problem where a unique transform scrambles the image and makes it unintelligible to the attacker. Only the person who sends the image knows the transform used to encrypt the image and hence it becomes difficult to decode.

The Method of encryption needs to have the following quality:

- Diffusion – The scrambling should take place over the entire image to provide a good quality encryption. Not even one region should be recognizable by the attacker.
- Confusion – The image should be difficult to decipher by either scrambling the pixels at different positions or changing the values themselves.

1.2 Architecture

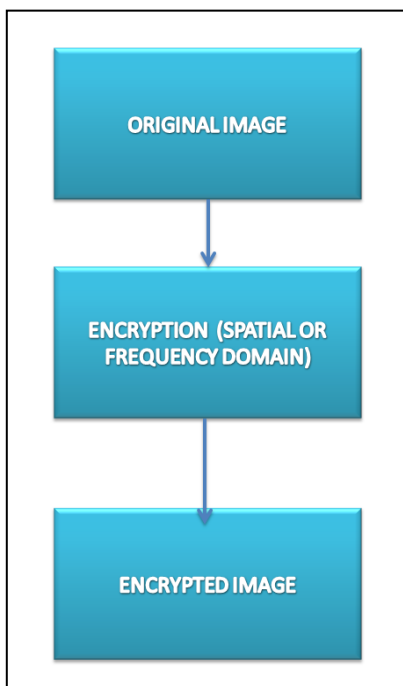


Fig1.2: Encryption

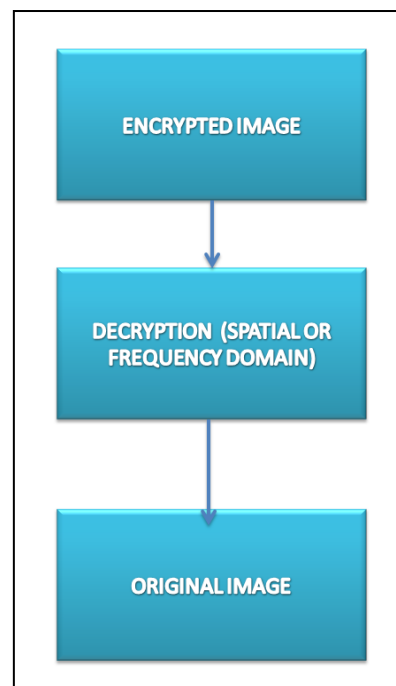


Fig 1.3: Decryption

- **Encryption:** The encryption involves confusion and diffusion of the image, such that the meaning of the image is not understandable visually. The basic steps followed in encryption are given below:
 1. **Input image:** An input image of size 256x256 is to be considered. The input images we have considered for analysis purpose are Baboon, Baby, Barbara, Barbie, Cartoon, Flower, Fruits, Lena, Pepper, Puppy, Scenery, Smiley, Taj Mahal, Temple, Watch. Each of these images have a different visual

representation. Computation of values for each of these images will help us provide an average estimate of an algorithm.

2. **Encryption Algorithm:** The encryption algorithm can be performed in spatial domain or frequency domain.
 - a. **Spatial Domain:** In spatial domain the pixel positions are shuffled and changed so that the visual representation of the image itself changes. Algorithms implemented in spatial domain are scrambling algorithms: Arnolds transformation, Key-based scrambling, Lucas transformation, Fibonacci transformation, Gray Code, n-p-k Gray Code, New Linear Transformation, Queue Transformation.
 - b. **Frequency Domain:** In frequency domain, the pixel values itself is changed according to the transform like sine transform and cosine transform so as to change the value of the image itself.
3. **Encrypted Image:** Encrypted image is the resulting image obtained after the input image is encrypted. It should not contain parts of the original image. The original image should be recognizable in the encrypted image at all. A good quality encrypted image is the one which is not recognizable or visible at all.
 - **Decryption:** Decryption of the image is as important as the encryption algorithm. If the decryption algorithm does not return the original image which is an exact match of the input image, then the encryption algorithm does not work. The encryption algorithm only has to change the values for confusion and diffusion temporarily, and not change these values permanently. Steps followed in image encryption are given below:
 - A. **Encrypted image:** An encrypted image of size 256x256 is to the result of the encryption algorithm. The input images we have considered for analysis purpose are Baboon, Baby, Barbara, Barbie, Cartoon, Flower, Fruits, Lena, Pepper, Puppy, Scenery, Smiley, Taj Mahal, Temple, Watch. For each of the

different input images the encrypted image may appear similar, but on decryption the original image is retained.

B. Decryption Algorithm: The decryption algorithm is the reverse of encryption algorithm. This can be performed in spatial domain or frequency domain.

- **Spatial Domain:** In spatial domain the pixel positions are shuffled and changed so that the visual representation of the image itself changes. Algorithms implemented in spatial domain are scrambling algorithms: Arnold transformation, Key-based scrambling, Lucas transformation, Fibonacci transformation, Gray Code, n-p-k Gray Code, New Linear Transformation, and Queue Transformation.
- **Frequency Domain:** In frequency domain, the pixel values itself is changed according to the transform like sine transform and cosine transform so as to change the value of the image itself.

C. Resultant Image: Only if the resultant image is exactly matching the input image, does the encryption algorithm work.

1.3 Hardware Specification

1. Operating System: Windows XP/7/8/8.1
2. Processor: Intel(R) Core(TM) i5-3230M
3. CPU at 2.60Ghz
4. RAM: 4.00GB (above 1024MB is required)
5. System type: 64bit operating system, x64 based processor

1.4 Software Specification

1. Programming: MATLAB 2013
2. Documentation: MS Excel 2007, MS Word 2007

2. LITERATURE REVIEW

3. IMAGE ENCRYPTION IN THE SPATIAL DOMAIN

1. Key- Based Scrambling:

In key based scrambling the rows and columns of the image are shuffled according to a randomly generated key.

Algorithm – Scrambling:

Consider an image of size of size $m \times m$

- Generate a random key 'y' (say) of size m
- Consider each element of the y and follow the steps III to
- For each element 'e' of y, shuffle row i and e such that $y_i = e$
- For each element 'e' of y, shuffle column-wise i and e such that $y_i = e$
- Rotate the rows by a fixed size in either clockwise or anticlockwise direction
- Rotate columns of the image in either clockwise or anticlockwise direction
- Obtain the scrambled image which is not readable and scrambled.

Algorithm – Unscrambling:

Consider a scrambled image of size of size $m \times m$

- Generate a random key 'y' (say) of size m
- Consider each element of the y and follow the steps III to
- Rotate the columns by a fixed size in either clockwise or anticlockwise direction
- Rotate rows of the image in either clockwise or anticlockwise direction
- For each element 'e' of y, shuffle column-wise i and e such that $y_i = e$
- For each element 'e' of y, shuffle row-wise i and e such that $y_i = e$
- The obtained image should match with the original image

2. (n,p,k)-Graycode Scrambling:

Graycode scrambling involves conversion of each pixel into a binary value and shuffling the binary equivalent according to the required parameters.

The sequence $(a_{k-1}, \dots, a_1, a_0)$ and $(g_{k-1}, \dots, g_1, g_0)$ are k-digits based on n-sequence of non-negative integers A and G integers where $A = \sum a_i n^i$ and $G = \sum g_i n^i$. G is called (n,p,k)-Gray code of A if the sequences are satisfied with

$$g_i = \begin{cases} a_i & \text{if } i > k-p-2 \\ (a_i + a_{i+p+1}) \bmod n & \text{if } 0 \leq i \leq k-p-2 \end{cases}$$

It can be represented in matrices format using the following formula:

$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{k-2} \\ g_{k-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{k-2} \\ a_{k-1} \end{pmatrix} \bmod n$$

If $p=2$ then the values after two positions are multiplied. It can be represented using the following

$$(g_0, g_1, \dots, g_{k-1}) =$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & \dots & \dots \\ 0 & 1 & 0 & 0 & \dots & \dots \\ \cdot & & & & & \\ \cdot & & & & & \\ \cdot & & & & & \\ 0 & \dots & \dots & 0 & 1 & \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{k-1} \end{pmatrix}$$

ALGORITHM - SCRAMBLING

Consider an input image of size $m \times n$

1. Enter the value of n and k in order to run the algorithm

2. For each pixel value at position (i,j) of the input image I, convert the decimal value of base 10 to binary value B of base 2
3. For each B binary value in the input image of size mxn obtain its gray code value G
4. Convert this gray code value G into decimal value D'
5. Combine these new decimal values D' to obtain the scrambled image.

ALGORITHM - UNSCRAMBLING

Perform the exact reverse of the above given scrambling procedure to obtain the unscrambled image R. This image should be similar to the original image.

3. Arnold Transform:

A 2-dimensional Arnold transform also known as the Arnold cat map is used to scramble the original image. The transform is as follows:

$$\begin{matrix} x' \\ y' \end{matrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \pmod{n}$$

The periodicity is defined as the number of iterations it takes to obtain the original image(m_n).

While descrambling the encrypted image, we multiply by the inverse of the Arnold Transform to obtain the original image.

This can also be performed on RGB images as follows using the 3-D Arnold transformation:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \pmod{N}$$

4. Fibonacci Transform:

A p-Fibonacci series based on user's input is generated which is used to scramble an image to make it unintelligible to any attacker.

The Fibonacci series is generated using the following formula:

$$F_p(n) = \begin{cases} 0 & n < 1 \\ 1 & n = 1 \\ F(n-1) + F(n-p-1) & n > 1 \end{cases}$$

Where p is a non-negative integer

The Fibonacci series will be as follows:

1. For $p=0$

the sequence is powers of two, 1, 2, 4, 8, 16...;

2. For $p=1$

the sequence is 1, 1, 2, 3, 5, 8, 13, 21...;

3. For $p>1$

For the large values of p the sequence starts with consecutive 1's and immediately after that 1, 2, 3, 4.... p ...

Here, $F_p(n)$ and $F_p(n+1)$ are consecutive terms of the Fibonacci series. A permutation $\{T_1, T_2, T_3, \dots, T_{F_p(n+1)-1}\}$ on an input sequence $\{1, 2, 3, \dots, F_p(n+1)-1\}$ called the p -fibonacci transform is as follows:

$$T_k = k[F_p(n)+i] \bmod F_p(n+1)$$

Where $k=0, 1, \dots, F_p(n+1)-1$ and $i=-3, -2, -1, 0, 1, 2, 3$ and $F_p(n)+i < F_p(n+1)$

The column coefficient matrix is computed which is as follows:

$$T_c(i, j) = \begin{cases} 1 & (T_{pi}, i) \\ 0 & \text{otherwise} \end{cases}$$

The row coefficient matrix is computed as follows:

$$T_r(i, j) = \begin{cases} 1 & (i, T_{pi}) \\ 0 & \text{otherwise} \end{cases}$$

Scrambling:

$$S = T_r * D * T_c$$

Descrambling:

$$D = T_r * S * T_c$$

5. Lucas Transform

A p-Lucas series based on user's input is generated which is used to scramble an image to make it unintelligible to any attacker.

The Lucas series is generated using the following formula:

$$L_p(n) = \begin{cases} 2 & n < 1 \\ 2 & n = 1 \\ 1 & n = 2 \\ L(n-1) + L(n-p-1) & n > 2 \end{cases}$$

Where p is a non-negative integer

Here, $L_p(n)$ and $L(n+1)$ are consecutive terms of the Lucas series. A permutation $\{T_1, T_2, T_3, \dots, T_{L_p(n+1)-1}\}$ on an input sequence $\{1, 2, 3, \dots, L_p(n+1)-1\}$ called the p-Lucas transform is as follows:

$$T_k = k[L_p(n)+i] \bmod L_p(n+1)$$

Where $k=0, 1, \dots, L_p(n+1)-1$ and $i=-3, -2, -1, 0, 1, 2, 3$ and $L_p(n)+i < L_p(n+1)$

The column coefficient matrix is computed which is as follows:

$$T_c(i, j) = \begin{cases} 1 & (T_{pi}, i) \\ 0 & \text{otherwise} \end{cases}$$

The row coefficient matrix is computed as follows:

$$T_r(i, j) = \begin{cases} 1 & (i, T_{pi}) \\ 0 & \text{otherwise} \end{cases}$$

Scrambling:

$$S = T_r * D * T_c$$

Descrambling:

$$D = T_r * S * T$$

6. Queue Transform

Consider an image of the size $M \times N$ where there are M rows and N columns. Now, either the row of the matrix can be considered as a queue or even the column can be considered as a queue.

A scrambled image can be obtained by shuffling the rows and columns by using queue transformation. The first step is to take a transform reference point (I, J) anywhere in the image. Then w.r.t this point the other pixels will be shuffled.

First, the rows will be transformed using the following rules:

- If row number is greater than I then move left, if smaller than I then move right and row number I remains constant.
- Second, the columns will be transformed using the following rules: If the column number is greater than J move it up, if smaller than J then move down and column number J remains constant.
- These steps are repeated until the required result is obtained. One can perform column operations first and then row operations as well.

The formula to scramble the image row-wise is as follows:

$$(i', j') = ((i+j'-J), (j+i-I))$$

The formula to scramble the image column-wise is as follows:

$$(i', j') = ((i+j-J), (j+i'-I))$$

The scrambled image can be descrambled by performing the exact operations in the opposite order.

7. New Linear Transform

In this algorithm, the linear transform needs to be calculated first. In order to do that a permutation π must be generated.

π contains elements from $\{0, 1, 2, \dots, N-1\}$ for an image of size $N \times N$ and is represented as follows:

$$\begin{pmatrix} 0 & 1 & 2 & \dots & (N-1) \\ \pi(0) & \pi(1) & \pi(2) & \dots & \pi(N-1) \end{pmatrix}$$

$$\text{Then, } \pi(x) = \begin{cases} x/2 & \text{if } x \text{ is even} \end{cases}$$

$$= N-(x/2)-1 \quad \text{if } x \text{ is odd}$$

For example if $N=4$, then

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}$$

And the linear transform for $N=4$ is hence.

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Consider an image I of size $N \times N$. The image can be scrambled as follows:

$$E = L * I * L^T$$

The image can be descrambled by performing the following operations:

$$I = L^T * E * L$$

8. Gray Code

4. IMAGE ENCRYPTION IN THE TRANSFORM DOMAIN

Sinusoidal and Non-Sinusoidal Transforms:

1. Discrete Cosine Transform:

A Discrete Cosine Transform (DCT) is a sequence of finite data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, such as lossy compression of audio and images.

The equation for a 2D for N data items is given below

$$F(m, n) = \frac{2}{\sqrt{MN}} C(m)C(n) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \cos \frac{(2x+1)m\pi}{2M} \cos \frac{(2y+1)n\pi}{2N}$$

Figure 4.1: DCT formula

2. Discrete Sine Transform:

The Discrete Sine Transform (DST) is a member of sinusoidal unitary transforms family. DST is real, symmetric, and orthogonal.

The equation for DST for N points is given below

$$\varphi(k, n) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(k+1)(n+1)}{N+1}$$

Figure 4.2: DST formula

3. Discrete Fourier Transform:

The discrete Fourier transform (DFT) converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has those same sample values. It can be said to convert the sampled function from its original domain (often time or position along a line) to the frequency domain.

The equation of DFT can be given as

$$F(u, v) = \frac{1}{NM} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{-2\pi i (\frac{xu}{N} + \frac{yv}{M})}$$

Figure 4.3: DFT formula

4. Slant transform:

Slant transform has its first basis function as constant and second basis function as linear. The slant function for NxN where N = 2n can be given as

$$S(1) = 1/\sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Slant transform has the following properties:

- It is real
- It is orthogonal
- It is a fast algorithm

5. Kekre transform:

Kekre transform for NxN matrix can be given as:

$$K(x, y) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -N+1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -N+2 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & \dots & -N+(N-1) & 1 \end{bmatrix}$$

Figure 4.4: Kekre transform formula

5. IMPLEMENTATION AND RESULTS

For computation of the results for each of these algorithms, the following parameters are considered –

- **Correlation:** In an image a pixel is correlated to its neighboring pixels, which is why the image is recognizable visually. Thus, for an encrypted image the correlation should be least possible value. Less correlation means that the pixel is not related to its neighboring pixels and the encryption is of good quality.
- **Distance Scrambling Factors [DSF]:** DSF is the distance of a pixel with its neighboring pixels after encryption. Greater the DSF, the better the quality of encryption.
- **Average Moving Distance[AMD]:**
- **SSIM**
- **Adjacent Row Pixel Correlation [ARPC]:** ARPC is the correlation between two adjacent pixels in a row of an image.
- **Adjacent Column Pixel Correlation [ACPC]:** ACPC is the correlation between two adjacent pixels in a column of an image.
- **Adjacent Diagonal Pixel Correlation [ADPC]:** ADPC is the correlation between two adjacent pixels in the diagonal of an image.
- **Adjacent Anti-Diagonal Pixel Correlation [AADPC]:** AADPC is the correlation between two adjacent pixels in the anti-diagonal of an image.

1. Key- Based Scrambling:

In key based scrambling the rows and columns of the image are shuffled according to a randomly generated key.

Algorithm - Scrambling

Consider an image of size of size $m \times m$


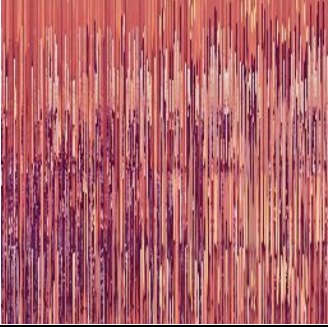

- Generate a random key 'y' (say) of size m
- Consider each element of the y and follow the steps III to
- For each element 'e' of y, shuffle row i and e such that $y_i = e$
- For each element 'e' of y, shuffle column-wise i and e such that $y_i = e$
- Rotate the rows by a fixed size in either clockwise or anticlockwise direction
- Rotate columns of the image in either clockwise or anticlockwise direction
- Obtain the scrambled image which is not readable and scrambled.

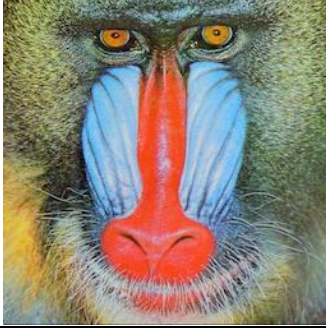
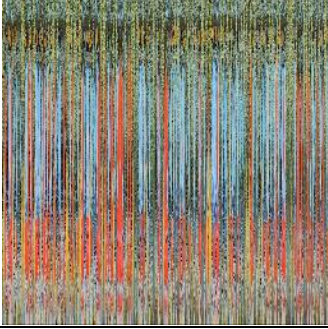
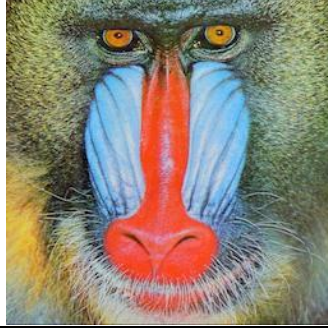
Algorithm - Unscrambling


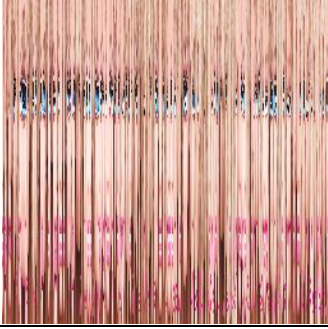

Consider a scrambled image of size of size $m \times m$

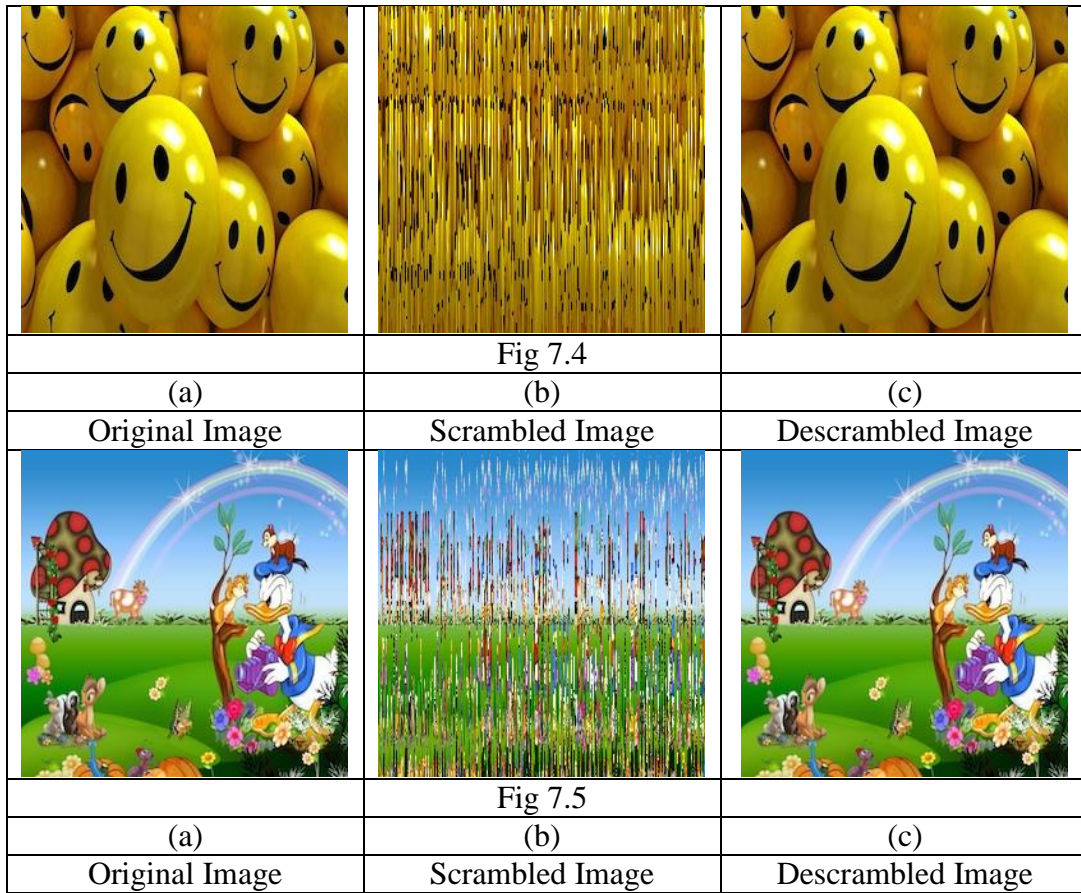
- Generate a random key 'y' (say) of size m
- Consider each element of the y and follow the steps III to
- Rotate the columns by a fixed size in either clockwise or anticlockwise direction
- Rotate rows of the image in either clockwise or anticlockwise direction
- For each element 'e' of y, shuffle column-wise i and e such that $y_i = e$
- For each element 'e' of y, shuffle row-wise i and e such that $y_i = e$
- The obtained image should match with the original image

Results:

		
	Fig 7.1	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

		
	Fig 7.2	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

		
	Fig 7.3	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image



Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.1286	0.8617	0.1233	0.1256
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.1135	0.9834	0.1139	0.1129
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.1387	0.9522	0.1363	0.1370
4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.2212	0.9771	0.2196	0.2193
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.3526	0.9407	0.3502	0.3500
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.1592	0.9329	0.1560	0.1575
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.1768	0.9554	0.1720	0.1730
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0195	0.9593	0.0188	0.0195
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.0829	0.9658	0.0823	0.0805

10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.1890	0.9912	0.1876	0.1870
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.4014	0.9063	0.3986	0.4003
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.0361	0.9581	0.0365	0.0345
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.5896	0.9676	0.5868	0.5879
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.3535	0.9311	0.3507	0.3516
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.3331	0.9020	0.3311	0.3315


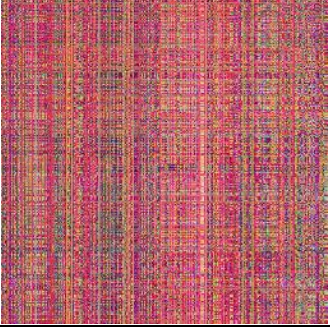

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0414
2.	Baby			0.1178
3.	Barbara			0.0332
4.	Barbie			0.0505
5.	Cartoon			0.1104
6.	Flower			0.0576
7.	Fruits			0.0341
8.	Lena			0.0586
9.	Pepper			0.0422
10.	Puppy			0.0582
11.	Scenery			0.0702
12.	Smiley			0.1217
13.	Taj Mahal			0.1640
14.	Temple			0.0333
15.	Watch			0.1065

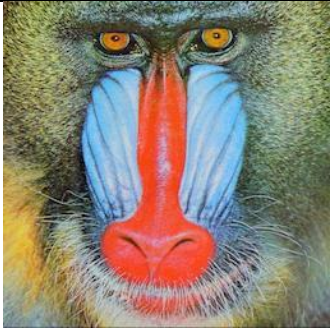
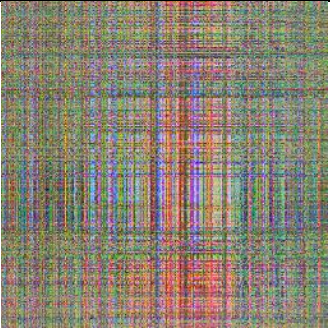
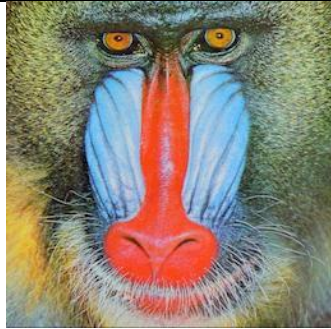
2. (n-p-k) Gray Code:


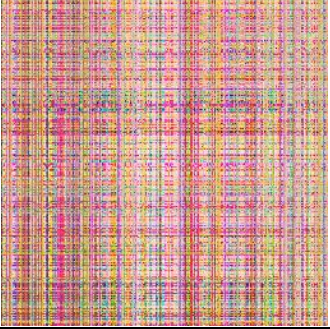

Graycode scrambling involves conversion of each pixel into a binary value and shuffling the binary equivalent according to the required parameters.

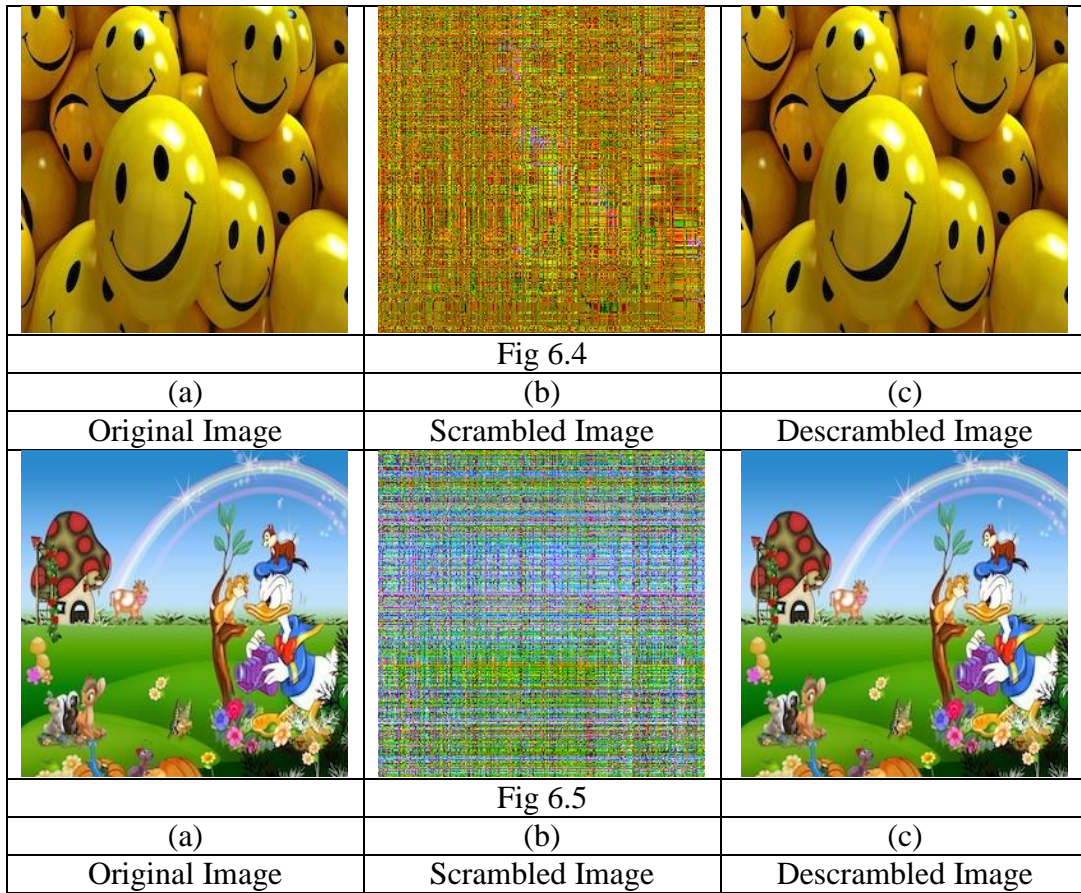
Results:

K=8; n=2; p=1

		
	Fig 6.1	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

		
	Fig 6.2	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

		
	Fig 6.3	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image



Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.2018	0.2833	0.0416	0.0412
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.1217	0.1472	0.0120	0.0120
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.2006	0.2250	0.0456	0.0455
4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.2203	0.3717	0.0356	0.0349
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.3504	0.1260	0.0275	0.0293
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.1333	0.2064	0.0185	0.0158
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.2060	0.1773	0.0297	0.0297
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0607	0.2306	0.0229	0.0235
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.1240	0.1514	0.0148	0.0136

10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.2466	0.3047	0.0443	0.0437
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.4194	0.0735	0.0252	0.0260
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.1956	0.2417	0.0475	0.0448
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.6374	0.0615	0.0276	0.0289
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.3719	0.1528	0.0288	0.0303
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.4414	0.2671	0.1417	0.1408

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0240
2.	Baby			0.0628
3.	Barbara			0.0278
4.	Barbie			0.0259
5.	Cartoon			0.0172
6.	Flower			0.0384
7.	Fruits			0.0131
8.	Lena			0.0369
9.	Pepper			0.0242
10.	Puppy			0.0188
11.	Scenery			0.0164
12.	Smiley			0.1081
13.	Taj Mahal			0.0200
14.	Temple			0.0126
15.	Watch			0.0468

3. Queue Transformation

Consider an image of the size $M \times N$ where there are M rows and N columns. Now, either the row of the matrix can be considered as a queue or even the column can be considered as a queue.

A scrambled image can be obtained by shuffling the rows and columns by using queue transformation. The first step is to take a transform reference point (I, J) anywhere in the image. Then w.r.t this point the other pixels will be shuffled.

First, the rows will be transformed using the following rules:

- If row number is greater than I then move left, if smaller than I then move right and row number I remains constant.

- Second, the columns will be transformed using the following rules: If the column number is greater than J move it up, if smaller than J then move down and column number J remains constant.
- These steps are repeated until the required result is obtained. One can perform column operations first and then row operations as well.

The formula to scramble the image row-wise is as follows:

$$(i', j') = ((i+j'-J), (j+i-I))$$





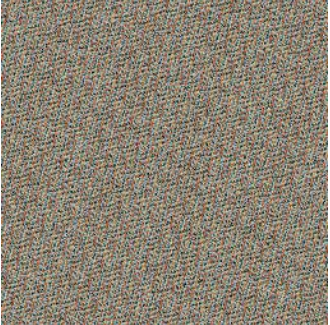
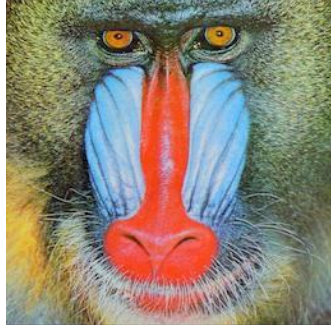
The formula to scramble the image column-wise is as follows:


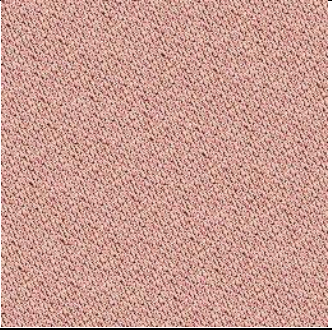


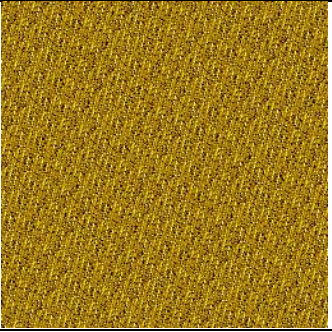


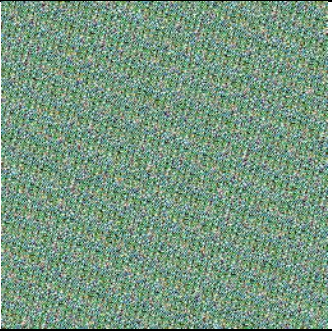

$$(i', j') = ((i+j-J), (j+i'-I))$$

The scrambled image can be descrambled by performing the exact operations in the opposite order.

$$(I, J, r) = (30, 30, 7)$$

Results:

		
	Fig 1.1	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 1.2	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

		
	Fig 1.3	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 1.4	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 1.5	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.0658	0.0373	0.0431	0.0276
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.2079	0.0731	0.0537	0.1463
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.0334	0.0369	0.0519	0.0227

4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.1889	0.0452	0.1357	0.1442
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.2461	0.1286	0.0685	0.1843
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.0813	0.1275	0.0419	0.0543
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.1020	0.0329	0.0651	0.0660
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0400	0.0389	0.0713	0.1182
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.1326	0.0265	0.0349	0.0475
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.0766	0.0783	0.1902	0.0560
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.0928	0.0505	0.0831	0.0729
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.0466	0.0984	0.0547	0.0584
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.2567	0.1994	0.1348	0.3059
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.1759	0.1297	0.0559	0.1114
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.3331	0.1236	0.0220	0.1126

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0174
2.	Baby			0.0528
3.	Barbara			0.0202
4.	Barbie			0.0187
5.	Cartoon			0.0111
6.	Flower			0.0317
7.	Fruits			0.0071
8.	Lena			0.0276
9.	Pepper			0.0209
10.	Puppy			0.0165
11.	Scenery			0.0110
12.	Smiley			0.0524
13.	Taj Mahal			0.0170
14.	Temple			0.0082
15.	Watch			0.0262

4. New Linear Transform:

In this algorithm, the linear transform needs to be calculated first. In order to do that a permutation π must be generated.

π contains elements from $\{0,1,2,\dots,N-1\}$ for an image of size $N*N$ and is represented as follows:

$$\begin{pmatrix} 0 & 1 & 2 & \dots & (N-1) \\ \pi(0) & \pi(1) & \pi(2) & \dots & \pi(N-1) \end{pmatrix}$$

Then,

$$\begin{aligned} \pi(x) &= x/2 && \text{if } x \text{ is even} \\ &= N-(x/2)-1 && \text{if } x \text{ is odd} \end{aligned}$$

For example if $N=4$, then

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 1 & 2 \end{pmatrix}$$

And the linear transform for $N=4$ is hence.

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Consider an image I of size $N*N$. The image can be scrambled as follows:

$$E = L * I * L^T$$

The image can be descrambled by performing the following operations:

$$I = L^T * E * L$$

Results:

Iteration value=20

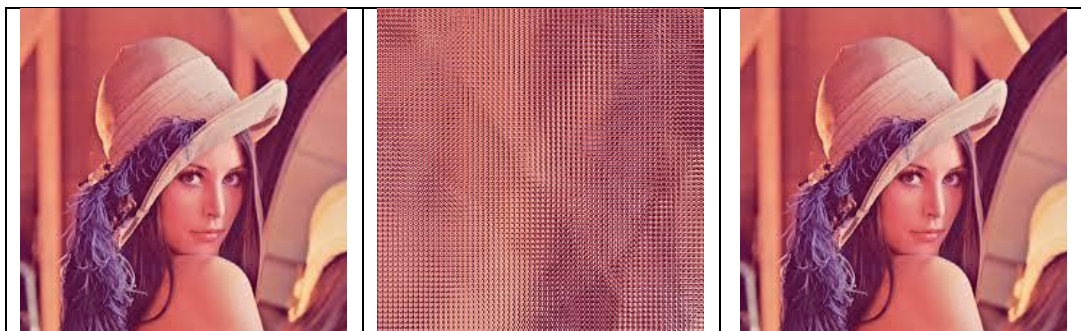
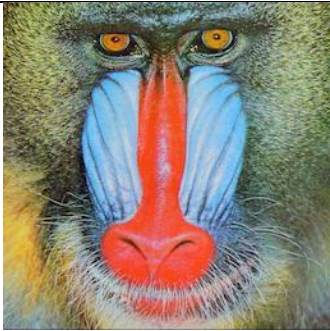
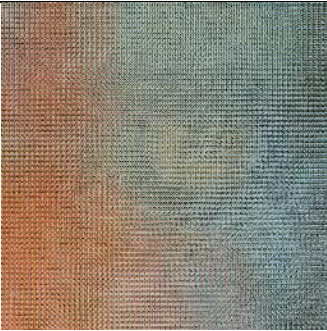
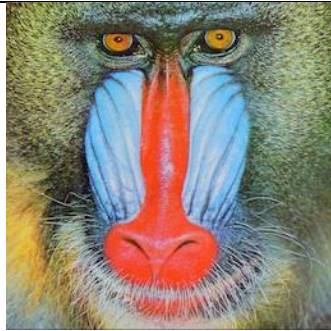

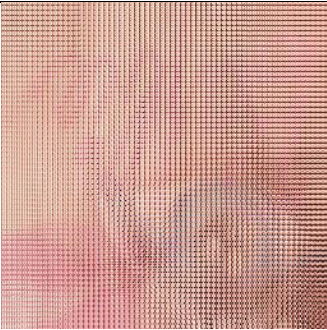


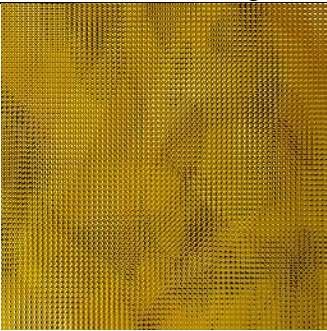

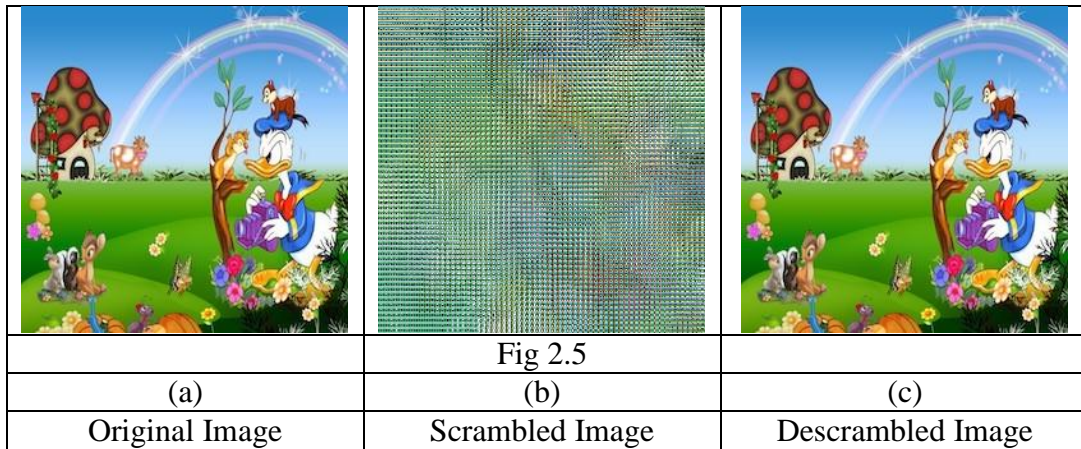


Fig 2.1		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
Fig 2.2		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
Fig 2.3		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
Fig 2.4		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image



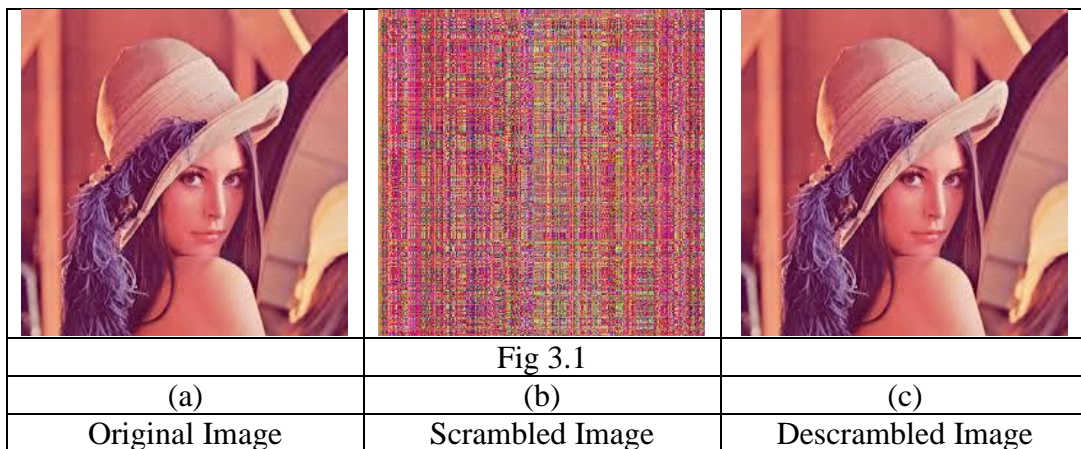
Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.0649	0.1373	0.0356	0.0374
2.	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.0290	0.0451	0.0770	0.0303
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.0645	0.0779	0.0457	0.0742
4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.0490	0.2558	0.0805	0.0793
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.3159	0.0635	0.1037	0.0681
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.0668	0.1011	0.1613	0.1542
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.1835	0.0536	0.0098	0.0306
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0449	0.1142	0.0203	0.0504
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.0384	0.0495	0.0337	0.0742
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.0269	0.2214	0.0730	0.0561
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.3865	0.0254	0.0863	0.0857
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.0682	0.0459	0.0402	0.0298
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.5711	0.0892	0.0130	0.0131
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.3328	0.0683	0.0683	0.0586
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412

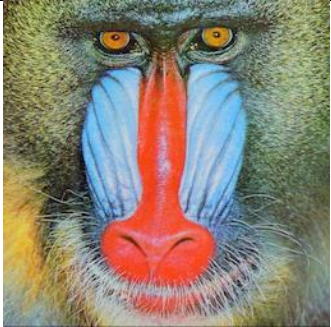
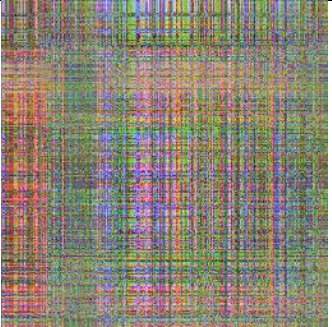
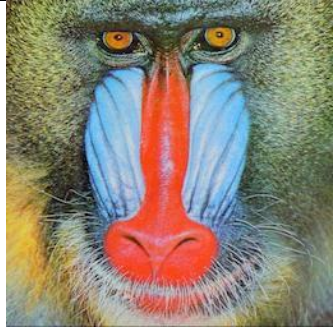

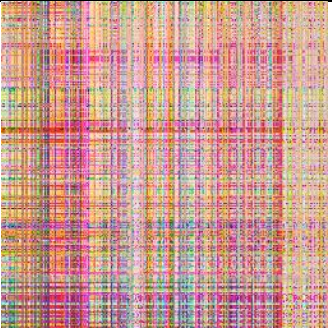


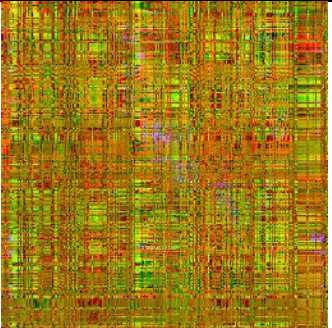


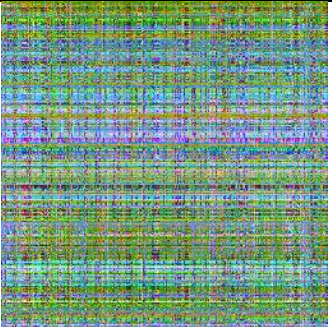

		Scrambled	0.2975	0.1812	0.1704	0.1236
--	--	-----------	--------	--------	--------	--------

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0209
2.	Baby			0.0655
3.	Barbara			0.0205
4.	Barbie			0.0220
5.	Cartoon			0.0114
6.	Flower			0.0326
7.	Fruits			0.0087
8.	Lena			0.0263
9.	Pepper			0.0215
10.	Puppy			0.0168
11.	Scenery			0.0148
12.	Smiley			0.0725
13.	Taj Mahal			0.0134
14.	Temple			0.0053
15.	Watch			0.0272

5. Gray Code:

Number of times=2; base value=10



		
	Fig 3.2	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 3.3	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 3.4	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 3.5	

(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.2260	0.3004	0.0570	0.0552
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.3032	0.3099	0.0897	0.0908
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.2592	0.2919	0.0728	0.0726
4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.3089	0.4750	0.1378	0.1381
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.3452	0.1452	0.0380	0.0393
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.1718	0.2219	0.0332	0.0320
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.2100	0.1778	0.0406	0.0415
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.1454	0.2905	0.0424	0.0426
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.1954	0.2414	0.0465	0.0437
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.3812	0.4269	0.1481	0.1491
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.4095	0.0858	0.0238	0.0230
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.2906	0.3130	0.0925	0.0902
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.6810	0.2213	0.1368	0.1355
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.4591	0.3195	0.1579	0.1612
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.4507	0.2777	0.1376	0.1375

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0239
2.	Baby			0.0834
3.	Barbara			0.0295
4.	Barbie			0.0329
5.	Cartoon			0.0137
6.	Flower			0.0374

7.	Fruits			0.0101
8.	Lena			0.0373
9.	Pepper			0.0256
10.	Puppy			0.0232
11.	Scenery			0.0166
12.	Smiley			0.1108
13.	Taj Mahal			0.0395
14.	Temple			0.0143
15.	Watch			0.0422

6. Fibonacci Transform:

A p-Fibonacci series based on user's input is generated which is used to scramble an image to make it unintelligible to any attacker.

The Fibonacci series is generated using the following formula:

$$F_p(n) = \begin{cases} 0 & n < 1 \\ 1 & n = 1 \\ F(n-1) + F(n-p-1) & n > 1 \end{cases}$$

Where p is a non-negative integer

The Fibonacci series will be as follows:

1. For p=0
the sequence is powers of two, 1, 2, 4, 8, 16...;
2. For p=1
the sequence is 1, 1, 2, 3, 5, 8, 13, 21...;
3. For p>1

For the large values of p the sequence starts with consecutive 1's and immediately after that 1, 2, 3, 4....p...

Here, $F_p(n)$ and $F_p(n+1)$ are consecutive terms of the Fibonacci series. A permutation $\{T_1, T_2, T_3, \dots, T_{F_p(n+1)-1}\}$ on an input sequence $\{1, 2, 3, \dots, F_p(n+1)-1\}$ called the p-fibonacci transform is as follows:

$$T_k = k[F_p(n)+i] \bmod F_p(n+1)$$

Where $k=0,1,\dots,Fp(n+1)-1$ and $i=-3,-2,-1,0,1,2,3$ and $Fp(n)+i < Fp(n+1)$

The column coefficient matrix is computed which is as follows:

$$T_c(i,j) = \begin{cases} 1 & (T_{pi},i) \\ 0 & \text{otherwise} \end{cases}$$

The row coefficient matrix is computed as follows:

$$T_r(i,j) = \begin{cases} 1 & (i, T_{pi}) \\ 0 & \text{otherwise} \end{cases}$$

Scrambling:

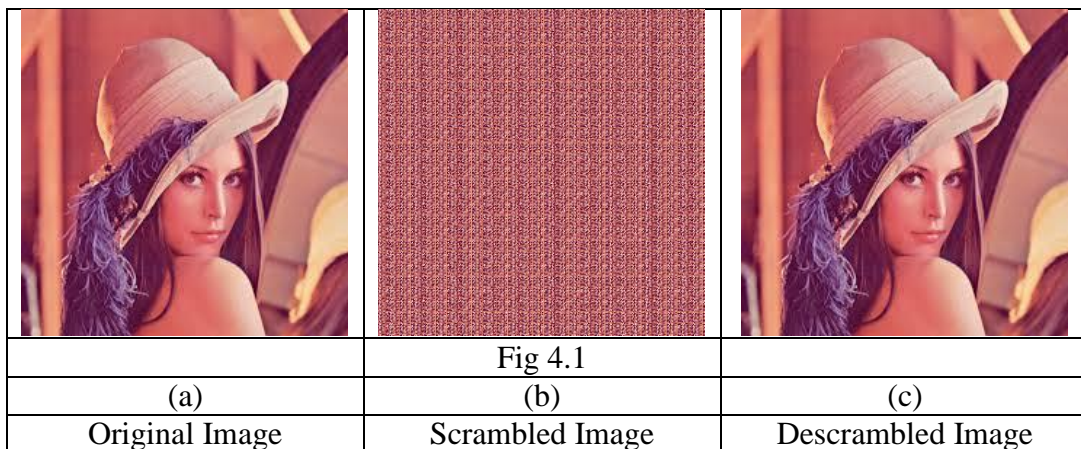
$$S = T_r * D * T_c$$

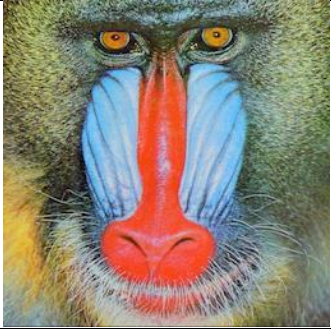
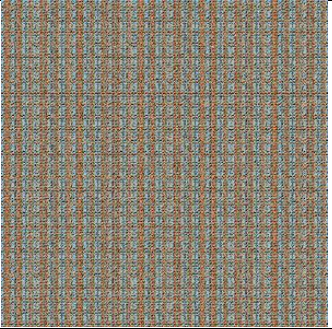
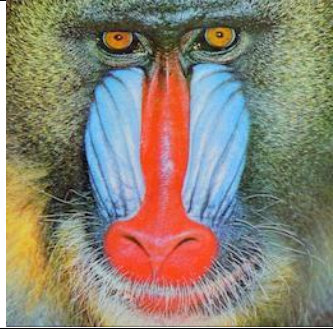

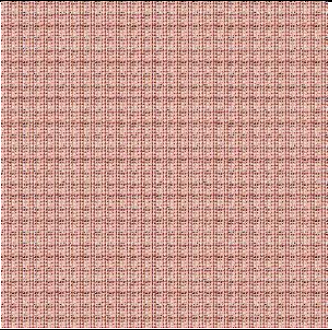


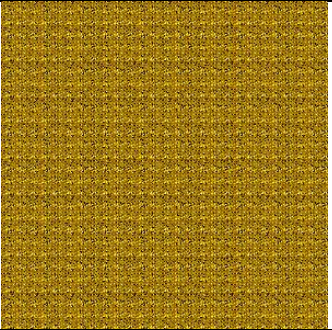


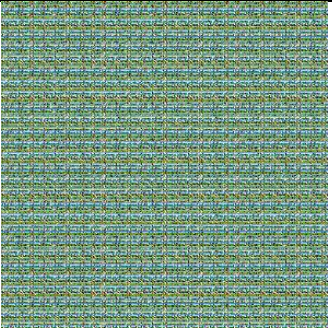

Descrambling:

$$D = T_r * S * T_c$$

Results:

P=8



		
(a)	Fig 4.2 (b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	Fig 4.3 (b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	Fig 4.4 (b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 4.5 (b)	
	Scrambled Image	Descrambled Image

(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.1183	0.0874	0.1097	0.0826
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.0362	0.0188	0.1563	0.1441
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.0707	0.0435	0.0869	0.0557
4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.0527	0.2630	0.2365	0.1798
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.2662	0.0754	0.0635	0.1241
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.0184	0.1070	0.0786	0.1064
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.0459	0.0620	0.0524	0.0513
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0367	0.0787	0.1082	0.0138
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.0662	0.0738	0.0341	0.0366
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.0805	0.1152	0.1292	0.1385
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.3405	0.0947	0.1529	0.1443
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.0337	0.0133	0.0307	0.0394
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.5349	0.2457	0.2007	0.1979
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.2889	0.1071	0.1723	0.0891
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.2490	0.0984	0.0107	0.2087

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			-Inf
2.	Baby			-Inf
3.	Barbara			-Inf
4.	Barbie			-Inf
5.	Cartoon			-Inf
6.	Flower			-Inf

7.	Fruits			-Inf
8.	Lena			-Inf
9.	Pepper			-Inf
10.	Puppy			-Inf
11.	Scenery			-Inf
12.	Smiley			-Inf
13.	Taj Mahal			-Inf
14.	Temple			-Inf
15.	Watch			-Inf

7. Lucas Transform:

A p-Lucas series based on user's input is generated which is used to scramble an image to make it unintelligible to any attacker.

The Lucas series is generated using the following formula:

$$L_p(n) = \begin{cases} 2 & n < 1 \\ 2 & n = 1 \\ 1 & n = 2 \\ L(n-1) + L(n-p-1) & n > 2 \end{cases}$$

Where p is a non-negative integer

Here, $L_p(n)$ and $L(n+1)$ are consecutive terms of the Lucas series. A permutation $\{T_1, T_2, T_3, \dots, T_{L_p(n+1)-1}\}$ on an input sequence $\{1, 2, 3, \dots, L_p(n+1)-1\}$ called the p-Lucas transform is as follows:

$$T_k = k[L_p(n) + i] \bmod L_p(n+1)$$

Where $k=0, 1, \dots, L_p(n+1)-1$ and $i=-3, -2, -1, 0, 1, 2, 3$ and $L_p(n) + i < L_p(n+1)$

The column coefficient matrix is computed which is as follows:

$$T_c(i, j) = \begin{cases} 1 & (T_{pi}, i) \\ 0 & \text{otherwise} \end{cases}$$

The row coefficient matrix is computed as follows:

$$T_r(i,j)= \begin{cases} 1 & (i, T_{pi}) \\ 0 & \text{otherwise} \end{cases}$$

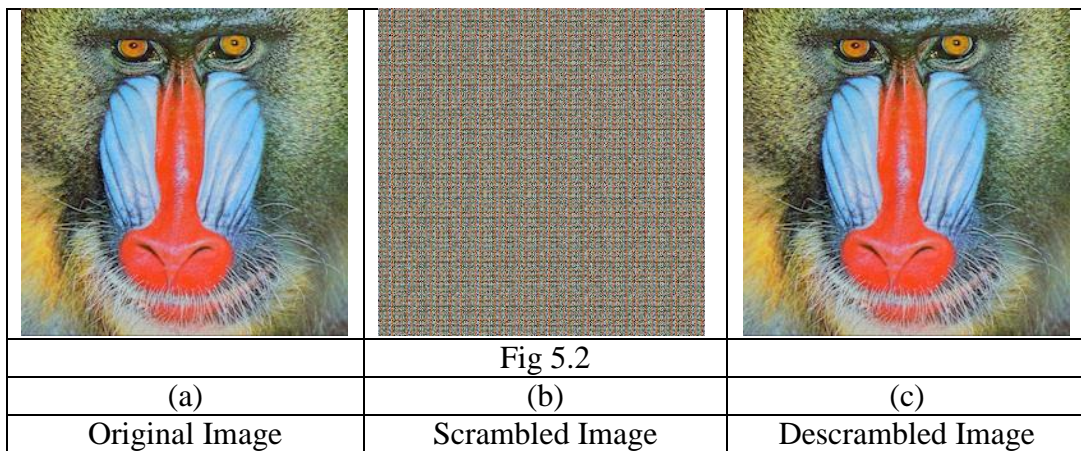
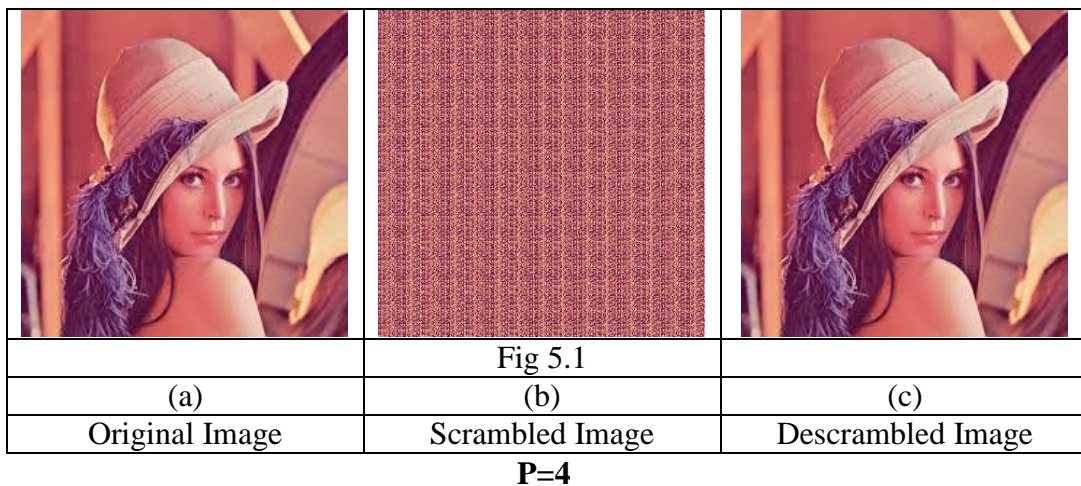
Scrambling:


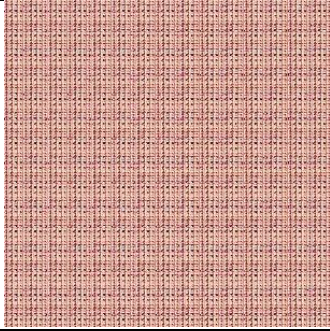


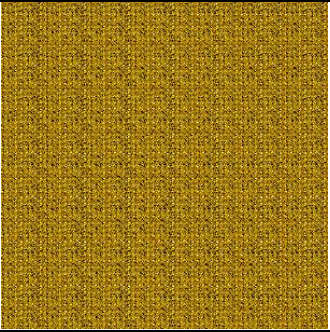


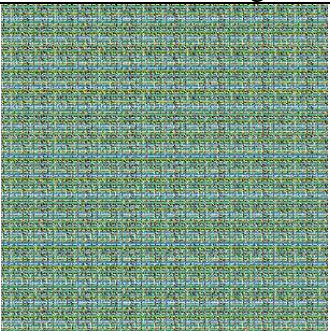

$$S = T_r * D * T_c$$

Descrambling:

$$D = T_r * S * T$$

Results:



		
	Fig 5.3	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 5.4	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
	Fig 5.5	
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.0304	0.1062	0.0322	0.0245
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.1074	0.1547	0.0573	0.1408
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.0243	0.1071	0.0683	0.0172

4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.1201	0.2425	0.1008	0.0773
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.2609	0.1401	0.1401	0.1866
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.0493	0.0834	0.1489	0.1555
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.0218	0.0525	0.0355	0.0816
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.0422	0.0424	0.0571	0.0261
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.1225	0.1273	0.0308	0.0779
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.0439	0.1325	0.0797	0.0624
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.3280	0.0368	0.0614	0.0370
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.0366	0.0183	0.0402	0.0104
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.5500	0.2787	0.1885	0.1985
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.2021	0.1941	0.1321	0.0996
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.1822	0.2591	0.0991	0.1687

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			-Inf
2.	Baby			-Inf
3.	Barbara			-Inf
4.	Barbie			-Inf
5.	Cartoon			-Inf
6.	Flower			-Inf
7.	Fruits			-Inf
8.	Lena			-Inf
9.	Pepper			-Inf
10.	Puppy			-Inf
11.	Scenery			-Inf
12.	Smiley			-Inf
13.	Taj Mahal			-Inf
14.	Temple			-Inf
15.	Watch			-Inf

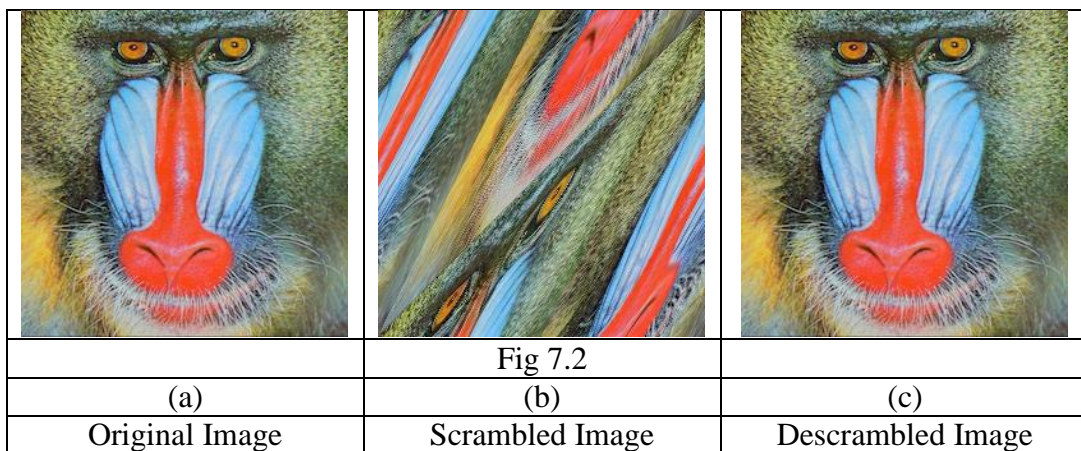
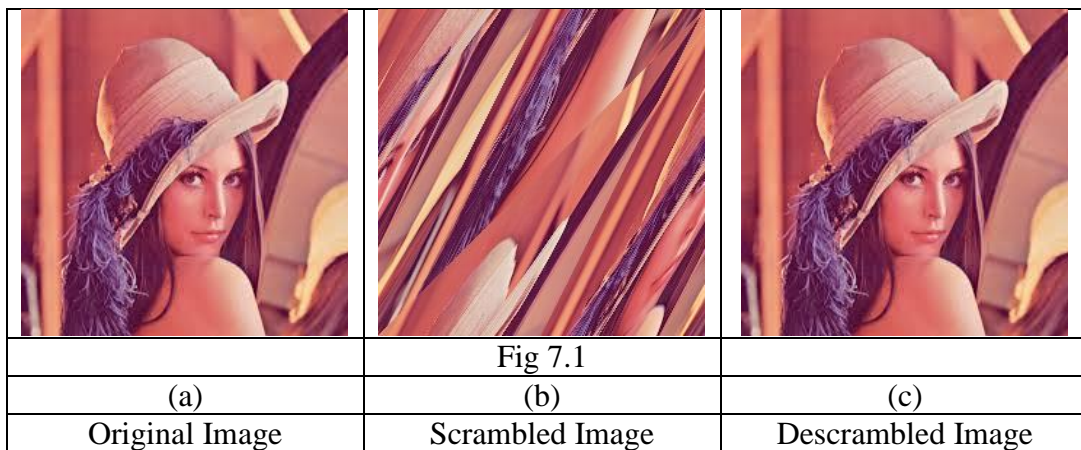
8. Arnold Transform:





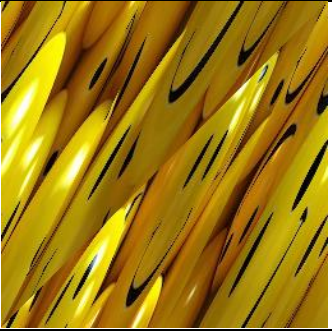




A 2-dimensional Arnold transform also known as the Arnold cat map is used to scramble the original image. The transform is as follows:

$$\begin{matrix} x' \\ y' \end{matrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \pmod{n}$$

Where $x, y \in \{0, 1, 2, \dots, N-1\}$

Results:



		
Fig 7.3		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
Fig 7.4		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
Fig 7.5		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Sr. No.	Image Name	Type of Image	ARPC	ACPC	ADPC	AADPC
1.	Baboon	Original	0.8947	0.8617	0.8340	0.8274
		Scrambled	0.7811	0.8301	0.7277	0.8928
2	Baby	Original	0.9763	0.9834	0.9657	0.9654
		Scrambled	0.9231	0.9610	0.8690	0.9731
3.	Barbara	Original	0.9417	0.9522	0.9016	0.9241
		Scrambled	0.8233	0.8948	0.7368	0.9373

4.	Barbie	Original	0.9551	0.9771	0.9438	0.9441
		Scrambled	0.8940	0.9385	0.8532	0.9548
5.	Cartoon	Original	0.9251	0.9407	0.8940	0.9013
		Scrambled	0.8246	0.8882	0.8882	0.9241
6.	Flower	Original	0.9184	0.9329	0.8689	0.8800
		Scrambled	0.7592	0.8622	0.6383	0.9144
7.	Fruits	Original	0.9461	0.9554	0.9268	0.9156
		Scrambled	0.8602	0.9168	0.7842	0.9449
8.	Lena	Original	0.9238	0.9593	0.8956	0.9207
		Scrambled	0.8010	0.8893	0.7173	0.9192
9.	Pepper	Original	0.9599	0.9658	0.9320	0.9396
		Scrambled	0.8652	0.9259	0.7762	0.9567
10.	Puppy	Original	0.9874	0.9912	0.9814	0.9821
		Scrambled	0.9585	0.9786	0.9248	0.9866
11.	Scenery	Original	0.8648	0.9063	0.8334	0.8292
		Scrambled	0.7425	0.8217	0.6871	0.8625
12.	Smiley	Original	0.9199	0.9581	0.9016	0.8974
		Scrambled	0.8013	0.8951	0.6985	0.9182
13.	Taj Mahal	Original	0.9500	0.9676	0.9292	0.9300
		Scrambled	0.8736	0.9193	0.8206	0.9480
14.	Temple	Original	0.9059	0.9311	0.8649	0.8668
		Scrambled	0.7894	0.8549	0.7177	0.9030
15.	Watch	Original	0.8961	0.9020	0.8793	0.8412
		Scrambled	0.8000	0.8747	0.7509	0.8943

Sr. No.	Image Name	AMD	DSF	SSIM
1.	Baboon			0.0603
2.	Baby			0.3650
3.	Barbara			0.1071
4.	Barbie			0.2061
5.	Cartoon			0.1293
6.	Flower			0.0846
7.	Fruits			0.0828
8.	Lena			0.1659
9.	Pepper			0.1434
10.	Puppy			0.2541
11.	Scenery			0.0795
12.	Smiley			0.2282
13.	Taj Mahal			0.1754
14.	Temple			0.0523
15.	Watch			0.1661

6. PARTIAL IMAGE ENCRYPTION USING VECTOR QUANTIZATION

7. CONCLUSION

REFERENCES

- [1] Zhenwei Shang Honge Ren Jian Zhang, "A Block Location Scrambling Algorithm of Digital Image Based on Arnold Transformation", The 9th International Conference for Young Computer Scientists.
- [2] Yicong Zhousa, Sos Agaianb, Valencia M. Joynera, Karen Panettaa, "Two Fibonacci P-code Based Image Scrambling Algorithms", The International Society for Optical Engineering.
- [3] Prashan Premaratne, "Key-based scrambling for secure image communication", University of Wollongong, University of Wollongong Research Online
- [4] Yicong Zhou, Karen Panetta, Sos Agaian, "An Image Scrambling Algorithm Using Parameter Based M-Sequences", Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008
- [5] Naeem, E.A.; Elnaby, M.M.A.; Hadhoud, M.M., "Chaotic image encryption in transform domains," International Conference on Computer Engineering & Systems, 2009. ICCES 2009., vol., no., pp.71-76, 14-16 Dec. 2009.
- [6] Sapna Anoop, Anoop Alakkaran, "A Full Image Encryption Scheme Based on Transform Domains and Stream Ciphers", International Journal of Advanced Information Science and Technology (IJAIST), Vol. 17, pp. 5-10, September 2013

- [7] Long Bao , Yicong Zhou , C. L. Philip Chen, "Image encryption in the wavelet domain", in Proc. SPIE 8755, Mobile Multimedia/Image Processing, Security, and Applications 2013
- [8] Y. Zhou, K. Panetta, S. Agaian, and C. L. P. Chen, "Image encryption using P- Fibonacci transform and decomposition," Optics Communications, vol. 285, pp. 594-608, 2012.
- [9] K. Wong and K Tanaka, "Scalable Image Scrambling Method Using Unified Constructive Permutation Function on Diagonal Blocks," IEEE Proc. 28th Picture Coding Symposium PCS, pp. 138-141, 2010.
- [10] Liu, S., Sheridan, J.T.: Optical Information Hiding by Combining Image Scrambling Techniques in Fractional Fourier Domains. In: Irish Signal and Systems Conference, pp. 249–254, 2001.
- [11] M François, T Grosge, D Barchiesi, R Erra, " Image Encryption Algorithm Based on a Chaotic Iterative Process", Applied Mathematics, Vol. 3, No. 12, 2012, pp. 1910-1920.
- [12] Mishra, Minati, Priyadarsini Mishra, M. C. Adhikary, and Sunit Kumar. "Image Encryption Using Fibonacci-Lucas Transformation." International Journal on Cryptography and Information Security (IJCIS), Vol.2, No.3, September 2012 pp 131-141
- [13] M. Cohn, and A. Lempel, "On fast M-sequence transforms (Corresp.)," Information Theory, IEEE Transactions on, vol. 23, pp. 135-137, 1977.
- [14] R. Pickholtz, D. Schilling, and L. Milstein, "Theory of Spread-Spectrum Communications- A Tutorial," Communications, IEEE Transactions on, vol. 30, pp. 855-884, 1982.

- [15] F. J. MacWilliams, and N. J. A. Sloane, "Pseudo-Random Sequences and Arrays," Proceedings of the IEEE, vol. 64, pp. 1715-1729, 1976.
- [16] Fuhao Zou, Zhengding Lu, Hefei Ling and Yanwei Yu, "Real-time video watermarking based on extended m-sequences," in Multimedia and Expo, 2006 IEEE International Conference on, Toronto, Canada, 2006, pp. 1561-1564.
- [17] Hong Wang, Ling Lu, Dashun Que, and Junbo Huang, "Adaptive audio digital watermark algorithm based on m-sequence modulation," in Signal Processing, 7th International Conference on, Beijing, China, 2004, pp. 2401-2404.
- [18] G. T. Buracas, and G. M. Boynton, "Efficient Design of Event-Related fMRI Experiments Using M-sequences," NeuroImage, vol. 16, pp. 801-813, 2002.
- [19] Yicong Zhou, Karen Panetta and Sos Agaian, "P-recursive sequence and key-dependent multimedia scrambling," in Mobile Multimedia/Image Processing for Military and Security Applications, SPIE Defence and Security Symposium 2008, Orlando, FL USA 2008.
- [20] J.-L. Fan and X.-F. Zhang, "Image encryption algorithm based on chaotic system," in International Conference on Computer-Aided Industrial Design and Conceptual Design CAIDCD, 2006, pp. 1–6.

APPENDIX

IMAGE DATABASE

Image Name	Image
Baboon	
Baby	
Barbara	
Barbie	

Cartoon



Flower



Fruits



Lena



Pepper



Puppy



Scenery



Smiley



Taj Mahal



Temple



Watch

