CHAPTER 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 again 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

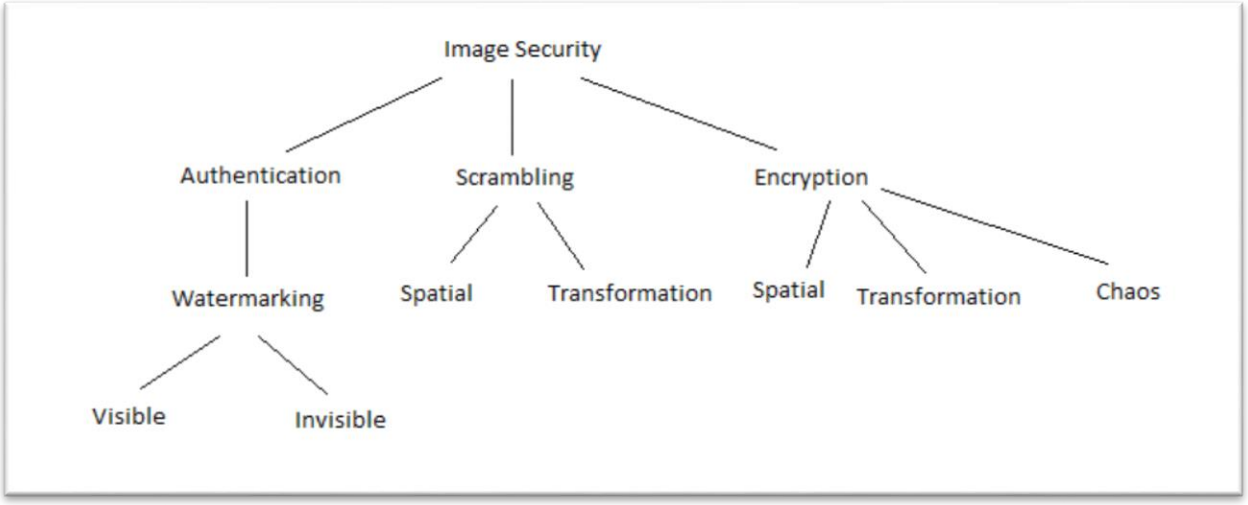


Figure 1.1 – Classification Image Security

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 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 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

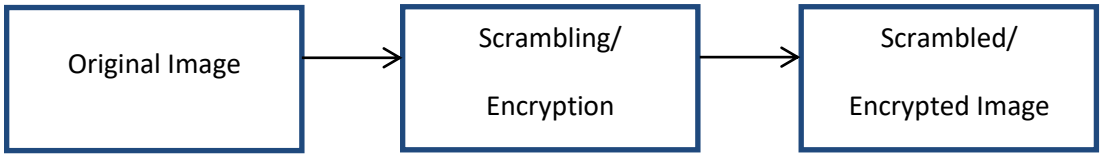


Fig 1.2: Encryption

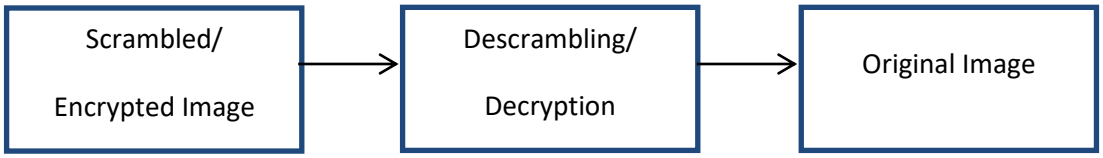


Fig 1.3: Decryption

- **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, TajMahal, 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: Arnold 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, TajMahal, 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.

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

- **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.

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

CHAPTER 2
LITERATURE REVIEW

Nowadays with the spreading technology and its benefits, the security risks and threats are rising as well. Social media platforms and electronic mail services have become centers for sharing and saving images. This not only increases connectivity but also exponentially increases privacy risks. News of image database hacking and unauthorized access to private images very common these days. In such a scenario it is necessary to implement a strong image encryption technique for images that creates confusion and diffusion. Recently, hacking of technology companies like Sony, Snapchat, Apple, etc has been circulating the news. A social media site like Facebook, in spite of its privacy and security settings, is vulnerable to security attacks and unauthorized uses. A strong encryption technique will ensure that even if unauthorized personnel gains access to certain images, he is unable to read the information the image contains. Image encryption is one of the many layers of security that can be applied for security purposes.

Image encryption is a growing field that has been extensively studied. An encryption technique based on pixels has been implemented in [1]. In this technique, first scrambling of the image takes place. Watermarking technique is then applied to increase the difficulty of decoding. Finally camouflaged image is to vision or the pixel of the true image is chosen, which gives us the encrypted image. The key parameters are encrypted using Elliptic Curve Cryptography (ECC). A region based selective encryption has been suggested in [2]. Region based encryption facilitates selective encryption and simultaneous reconstruction of images.

Recently chaos theory has also been utilized for the purpose of image encryption. Non-linear chaotic map[3], chaotic standard map[4] are used for chaotic stream cryptosystems or chaotic block cryptosystems. [6] has solved the problem of requirement of probability distribution of chaotic orbit for chaotic image scrambling. Li and Zhao have proposed an image scrambling technique using chaos theory and Vigen`ere cipher in [5].

Chaotic theory has been applied to image encryption as an effective and robust technique due to its unique properties. In [13] , we Chen et al, introduce a new combined chaotic system, which shows better chaotic behaviors than the traditional ones. Applying this chaotic system to image processing, a new image encryption algorithm is introduced based on the confusion and diffusion in

encryption procedure. Experimental results show that the proposed algorithm has a higher security level and excellent performance in image encryption.

In [7], firstly textual message is encrypted using a suitable key, to obtain some suitable nonlinear pixel and bit positions about the entire image. As a result, a watermarked image is obtained. Three different image shares are obtained by combining any two components R, G or B of the watermarked image. The key is also divided into three different logical blocks by digits. The key shares are assigned to image shares. Out of the three shares only a combination of two shares will result in the full image or key. At the decryption side, appropriate arrangement of shares of key and image makes the retrieval of hidden data.

Yen and J.-I.Guo [8] proposed a novel image encryption algorithm called BRIE (Bit Recirculation Image Encryption). This paper points out that BRIE is not secure enough from strict cryptographic viewpoint. It has been found that some defects exist in BRIE, and a known chosen-plaintext attack can break BRIE with only one known chosen plain-image. Experiments are made to verify the defects and the feasibility of the attack. Security is very important in transmission of digital images and video conferencing. There is a huge increase in the use of digital images in industrial process, security of these images with confidential data from unauthorized access is very important.

Advanced encryption standard(AES) is a well known block cipher method , however it is not suitable for real time applications. Various data encryption algorithms have been proposed and widely used, such as AES, RSA, or IDEA [9,10], most of which are used in text or binary data. It is difficult to use them directly in multimedia data, for multimedia data [11] are often of high redundancy, of large volumes and require real-time interactions, such asdisplaying, cutting, copying, bit rate conversion, etc. In [12], authors analyze and present a modification to the Advanced Encryption Standard (MAES) to reflect a high level security and better image encryption. The modification is done by adjusting the ShiftRow Transformation. Detailed results in terms of security analysis and implementation are given. Experimental results verify and prove that the proposed modification to image cryptosystem is highly secure from the cryptographic viewpoint. The results also prove that with a comparison to original AES encryption algorithm the modified algorithm gives better encryption results in terms of security against statistical attacks.

A logistic based image encryption techniques is been proposed in [14]. A Haar wavelet is applied over the image to decompose it and decorrelate its pixels into averaging and differencing components. The method produces cipher of the digital image that has good confusion and diffusion properties. The differencing components are compressed using a wavelet transform. Key transmission is done using steganography concept. NPCR, UACI and PSNR and other tests are carried out for experimental analysis. An upgraded method of SD-EI is proposed in [15] called as SD-AEI. This method has three stages: firstly each pixel is converted to its binary equivalent. In eight bit number, the number of bits equivalent to the length of the password is rotated and reversed. In second stage extended, hill cipher technique is applied by using evolutionary matrix which is generated by using the same password. In the final stage the whole image is randomized using Modified MSA Randomization encryption technique. The proposed technique is very effective in encrypting any type of image. Also experimental results show that this encryption technique takes optimal amount of time, as compared to various other traditional encryption techniques. This saves time and energy.

In [16] a full image encryption scheme based on DWT and Stream Ciphers is proposed. A competitive study on image encryption schemes in transform domains (DCT, DWT) is also discussed in this paper. A major recent trend is to minimize the computational requirements for secure multimedia distribution by selective encryption (SE) where only parts of the data are encrypted [17]. Selective encryption aims at avoiding the encryption of all bits of a digital image and yet ensuring a secure encryption [18].

In this project, we have created a basic framework using transform and scrambling technique which is effective in creating confusion and diffusion.

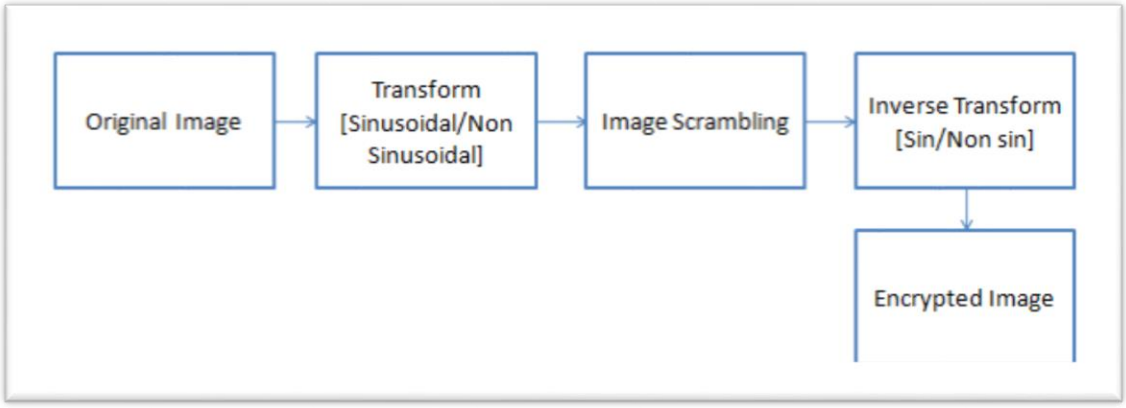


Figure 2.1: Encryption

For encryption (as shown in the above figure), first we take the original image of any format. The image is then converted into a grayscale image, on which the transform is then applied. Transform may include sinusoidal [DCT, DST, DFT, Fourier, Hartley] or non sinusoidal transforms [Hadamard, Walsh, Slant and Kekre]. We have also obtained hybrid transforms which involves combination of two transforms. Experimental results show that hybrid transform gives better results. After applying transform, an image scrambling technique is applied. We have used keybased scrambling as the basic scrambling technique in our approach, since this scrambling technique gives the best results. Inverse transform of this is obtained, which is the encrypted image.

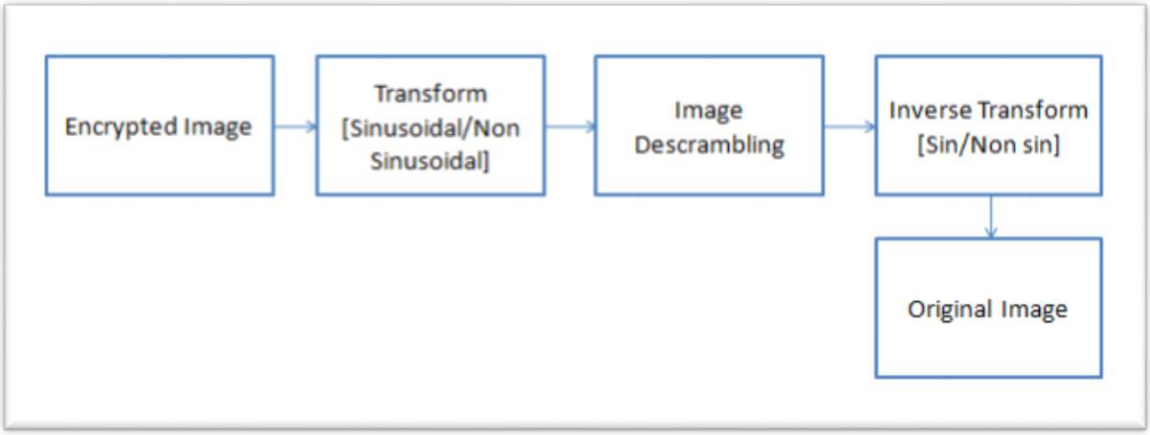


Figure 2.2: Decryption

Decryption procedure has similar framework as encryption, only descrambling technique is applied instead of scrambling. The inverse transform should give back the original image. The advantage of this framework over traditional scrambling or transform method is that it gives better results (lesser neighbouring correlation, more confusion and diffusion in the encrypted images, etc). The only disadvantage this framework might have is that it requires slightly greater computational time, as compared to the traditional techniques. How much time an algorithm takes depends on the kind of transform and scrambling algorithm that has been used in the framework.

CHAPTER 3
PERFORMANCE EVALUATION PARAMETERS

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.

$$d(x,y)=\sqrt{(x-x')^2+(y-y')^2}$$

Fig 3.1: DSF Formula

$$E_{max}(d)=\sqrt{(M-1)^2+(N-1)^2}$$

Fig 3.2: Mean moves distance of image

- **Structural Similarity [SSIM]:** Structural Similarity index is a method to measure the similarity between two images. It is given by the following formula:

$$\mathrm{SSIM}(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

Fig 3.3: SSIM Formula

Where
μ_x= average of x
μ_y= average of y
σ^2_x= variance of x
σ^2_y= variance of y
σ_{xy}= covariance of x and y
c_1 and c_2 are constants

- **NPCR:** Suppose ciphertext images before and after one pixel change in a plaintext image are and , respectively; the pixel value at grid in and are denoted as and ; and a bipolar array is defined in Eqn. (1). Then the NPCR can be mathematically defined by Eqns. (2) where symbol denotes the total number pixels in the ciphertext, symbol denotes the largest supported pixel value compatible with the ciphertext image format, and denotes the absolute value function.

$$D(i,j) = \begin{cases} 0, if\, C^1(i,j) = C^2(i,j) \\ 1, if\, C^1(i,j) \neq C^2(i,j) \end{cases} \quad (1)$$

$$\text{NPCR: } \mathcal{N}(C^1, C^2) = \sum_{i,j} \frac{D(i,j)}{T} \times 100\% \quad (2)$$

Figure 3.4: NPCR Formula

- **Average fractional change in pixel value (AFCPV):** Average fractional change in pixel value is given by the formula

$$\text{AFCPV} = \frac{\sum_{i=1}^{i=p} \sum_{j=1}^{j=q} (|x_{ij} - y_{ij}|) / x_{ij}}{p * q}$$

Figure 3.5 AFCPV formula

Where, x is the original image and y is the encrypted image.

- **Peak Average fractional Change in Pixel Value (PAFCPV):** Peak Average fractional change in Pixel value is the maximum fractional average change in pixel value of an image.
- **Entropy:** Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. Entropy is defined as

  *-sum(p.*log2(p))*

  where p contains the histogram counts

- **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.

CHAPTER 4
IMAGE ENCRYPTION IN THE SPATIAL DOMAIN

Introduction

This chapter explains the steps involved in the implementation of the algorithms in the spatial domain. The following algorithms have been implemented: Key-based scrambling, New Linear Transform, Queue Transformation, Fibonacci Transform, Lucas Transform, Arnold Transform, Gray Code and (n,p,k) Gray Code. The results obtained for each algorithm based on the parameters mentioned in the previous chapter have been tabulated and the outputs obtained for each have also been shown.

4.1 Key- Based Scrambling:

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

4.1.1 Scrambling Algorithm

Consider an image of size of size mxm

- 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.

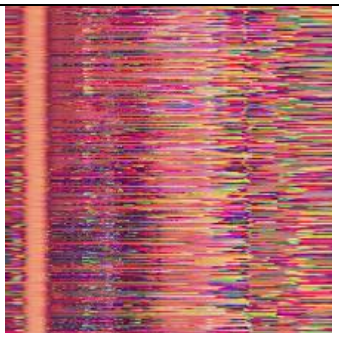
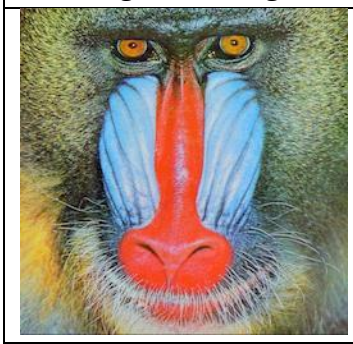
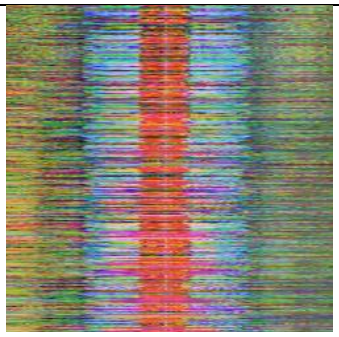
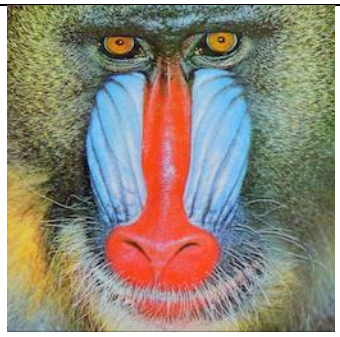
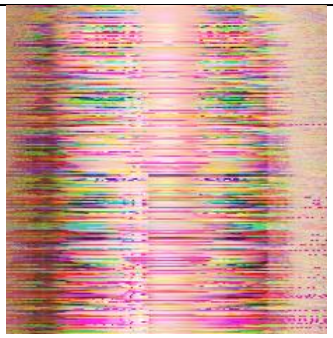
4.1.2 Unscrambling Algorithm:

Consider a scrambled image of size of size mxm

- 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

4.1.3 Results:

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

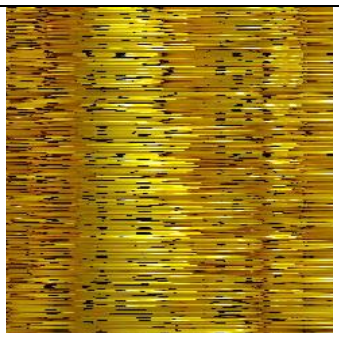
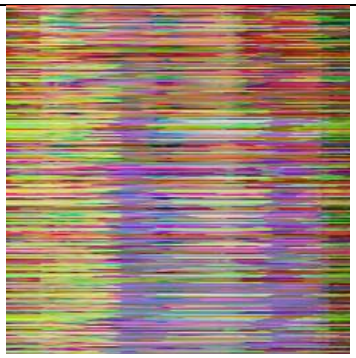
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.1.1: Output

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.	TajMahal	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

Table 4.1.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	0.1640
14.	Temple	0.0333
15.	Watch	0.1065

Table 4.1.3: SSIM Results

4.2 (n,p,k) Gray Code Scrambling:

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

The sequence $(a_{k-1},....,a_1, a_0)$ and $(g_{k-1},....,g_1, g_0)$ 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

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

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

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

Fig 4.1: Gray Code Formula

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

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

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

Fig 4.2: Gray Code Formula when p=2

4.2.1 Scrambling Algorithm:

Consider an input image of size mxn

- Enter the value of n and k inorder to run the algorithm
- 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
- For each B binary value in the input image of size mxn obtain its gray code value G
- Convert this gray code value G into decimal value D'
- Combine these new decimal values D' to obtain the scrambled image.

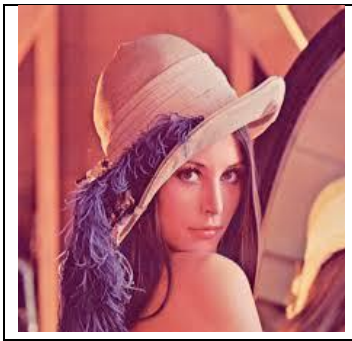
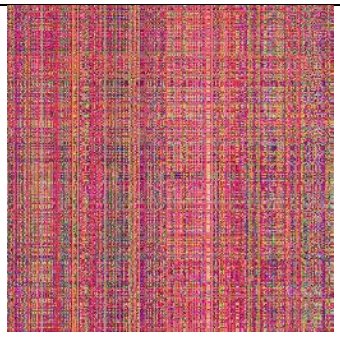
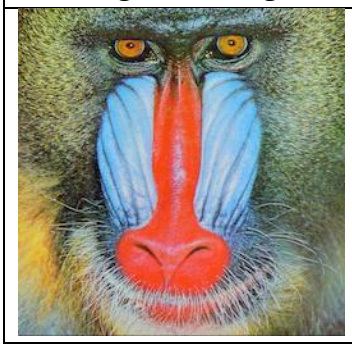
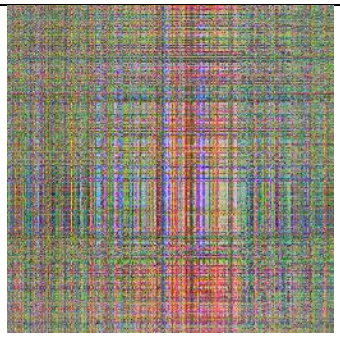
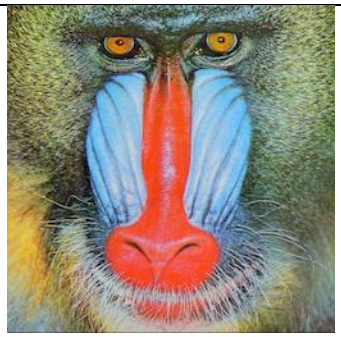
4.2.2 Unscrambling Algorithm:

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.

4.2.3 Results:

Parameters entered while execution:-

K=8; n=2; p=1

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

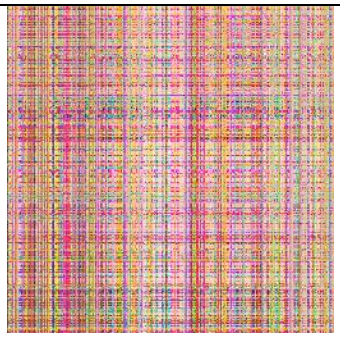
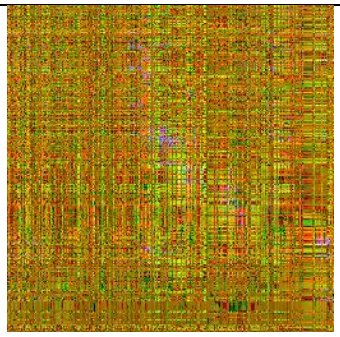
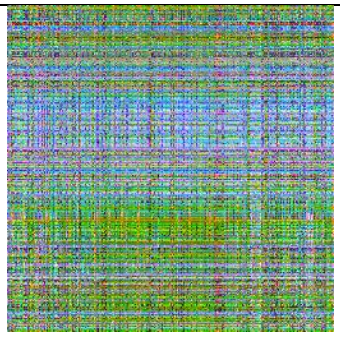
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.2.1: Output

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.	TajMaha l	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

Table 4.2.2: Correlation Results

Sr.No.	**Image Name**	**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.	TajMahal	0.0200
14.	Temple	0.0126
15.	Watch	0.0468

Table 4.2.3: SSIM Results

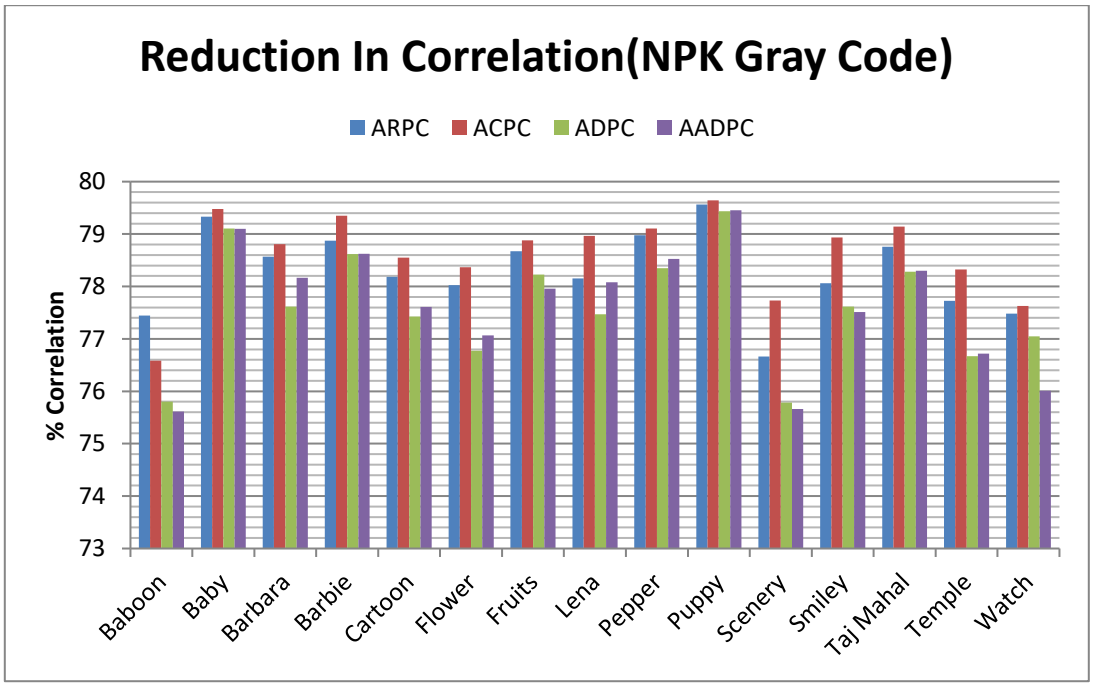


Fig 4.3: Reduction In Correlation(NPK Gray Code)

4.3 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:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} (\text{mod } n)$$

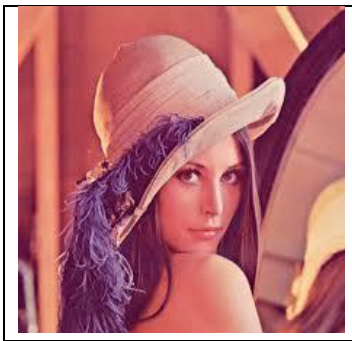
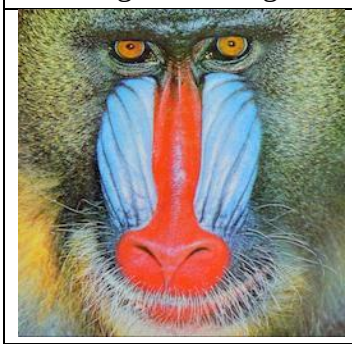
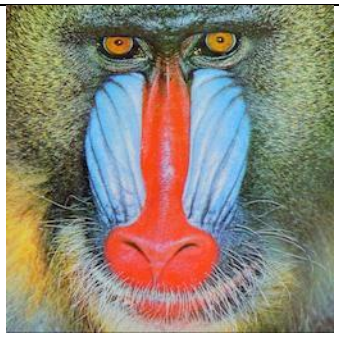
The periodicity is defined as the number of iterations it takes to obtain the original image(m_n).

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:

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

4.3.1 Results:

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

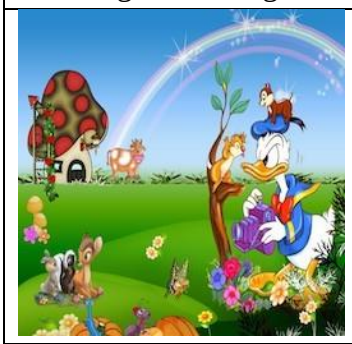
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.3.1: Output

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.	TajMahal	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

Table 4.3.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	0.1754
14.	Temple	0.0523
15.	Watch	0.1661

Table 4.3.3: SSIM Results

4.4 Fibonacci Transform:

A p-Fibonacci serried 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 $\{T1, T2, T3,\ldots T_{Fp(n+1)-1}\}$ on an input sequence $\{1, 2, 3\ldots..F_{p(n+1)-1}\}$ called the p-Fibonacci transform is as follows:

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

Where $k=0,1,\ldots F_p(n+1)-1$ 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:

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

4.4.1 Scrambling Algorithm:

$$S = T_r * D * T_c$$

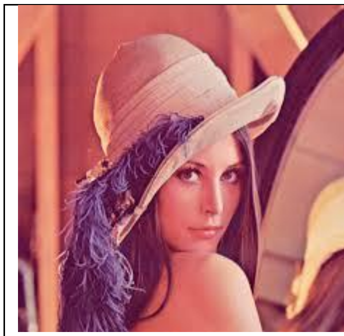
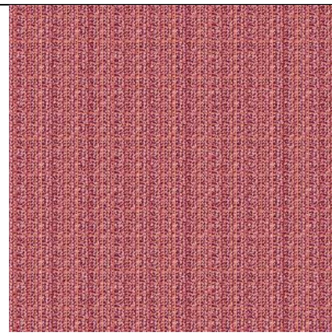
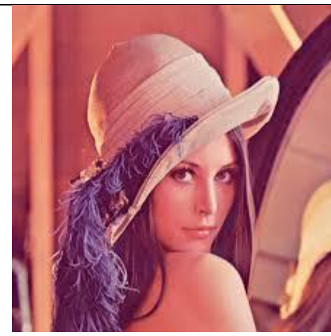
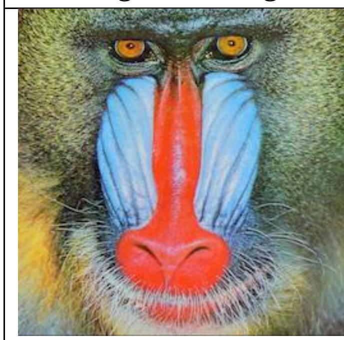
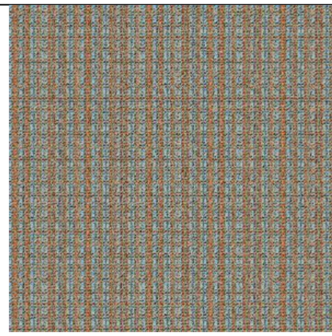
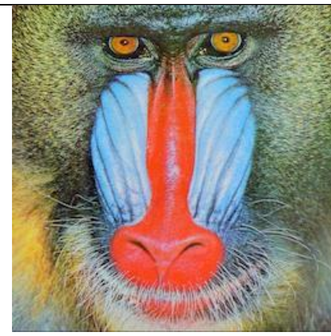
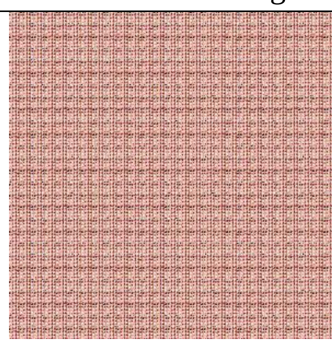
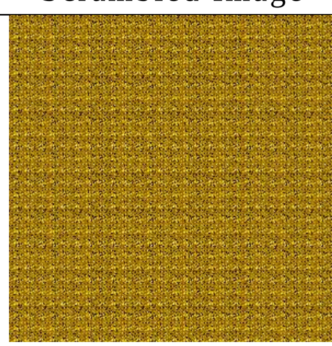
4.4.2 Unscrambling Algorithm:

$$D = T_r^{-1} * S * T_c^{-1}$$

4.4.3 Results:

Parameters taken during execution:-

P=8

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

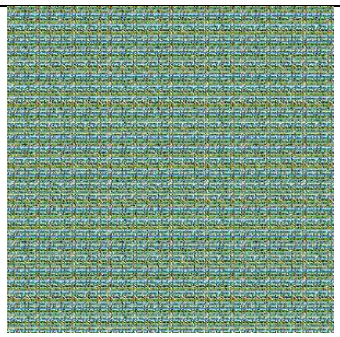
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.4.1: Output

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.	TajMahal	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

Table 4.4.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	-Inf
14.	Temple	-Inf
15.	Watch	-Inf

Table 4.4.3: SSIM Results

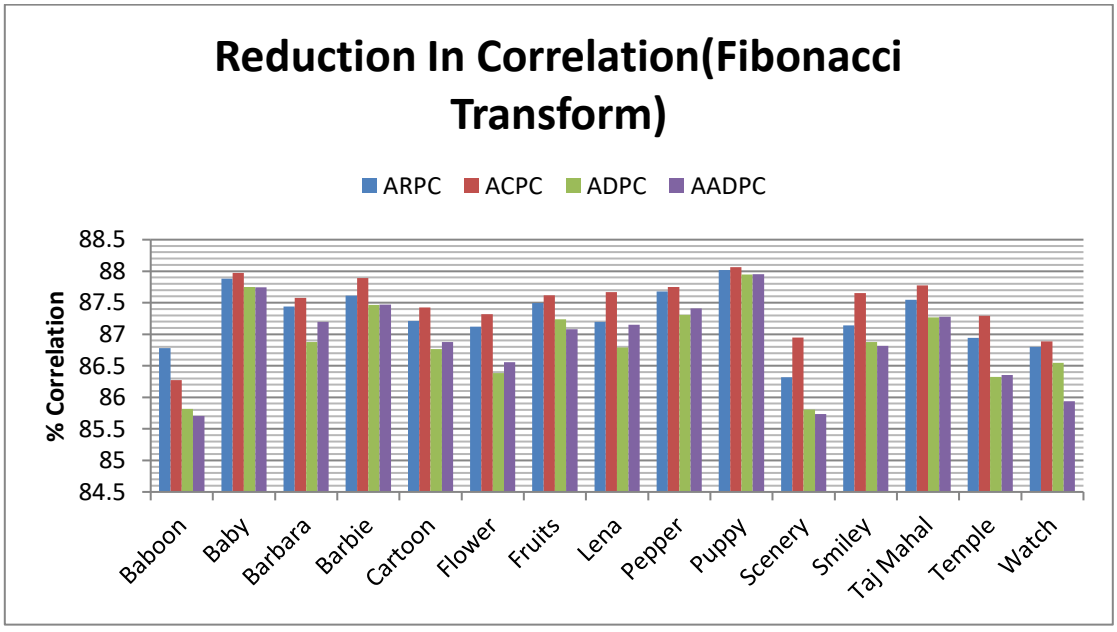


Fig 4.4: Reduction In Correlation(Fibonacci Transform)

4.5 Lucas Transform

A p-Lucas serried 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 {T1, T2, T3,...$T_{Lp(n+1)-1}$} on an input sequence {1, 2, 3.....$L_{p(n+1)-1}$} called the p-Lucas transform is as follows:
$T_k = k[L_p(n)+i] \bmod L_p(n+1)$

Where $k=0,1,...L_p(n+1)-1$ 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:

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

4.5.1 Scrambling Formula:

$$S = T_r * D * T_c$$

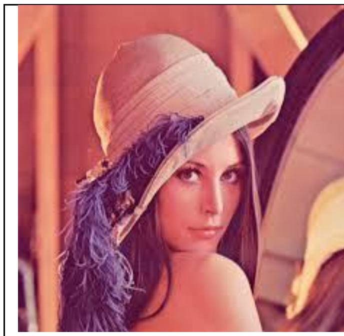
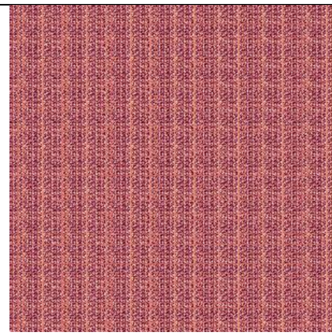
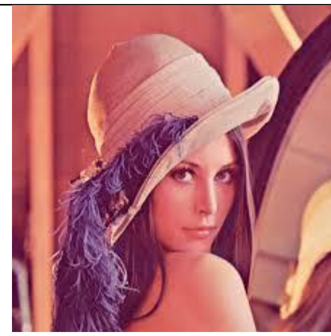
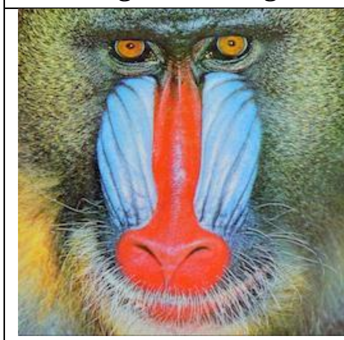
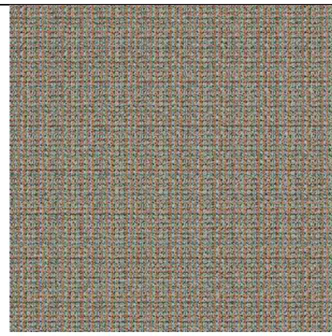
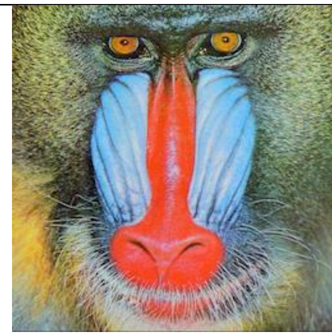
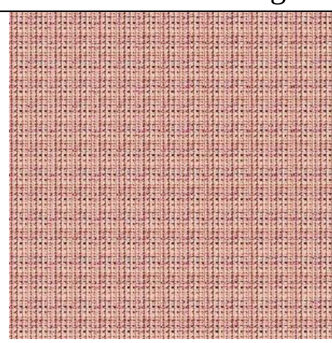
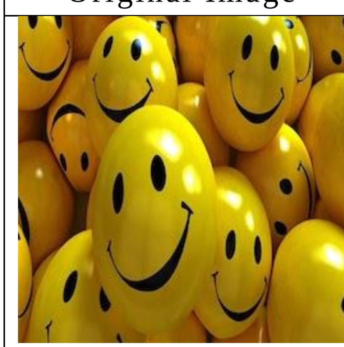
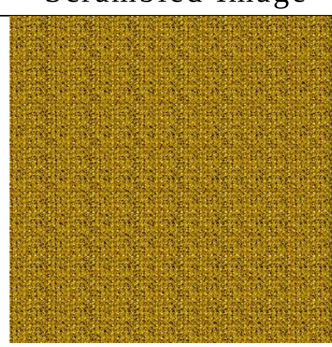
4.5.2 Unscrambling Formula:

$$D = T_r^{-1} * S * T_c^{-1}$$

4.5.3 Results:

Parameters taken during execution:-

P=4

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

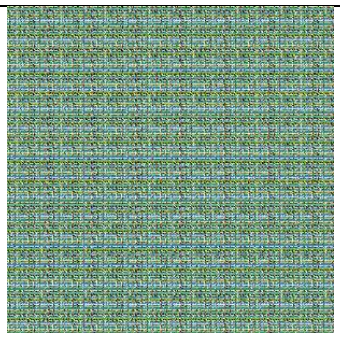
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.5.1: Output

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.	TajMahal	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

Table 4.5.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	-Inf
14.	Temple	-Inf
15.	Watch	-Inf

Table 4.5.3: SSIM Results

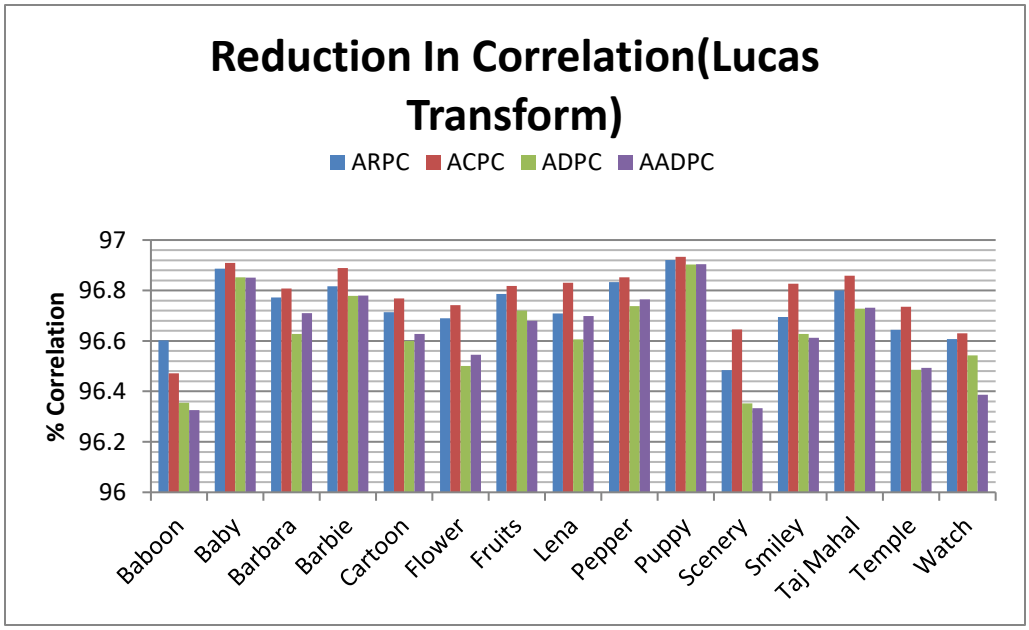


Fig 4.5: Reduction In Correlation(Lucas Transform)

4.6 Queue Transform

Consider an image of the size M*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:

4.6.1 Scrambling Formula:

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

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

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

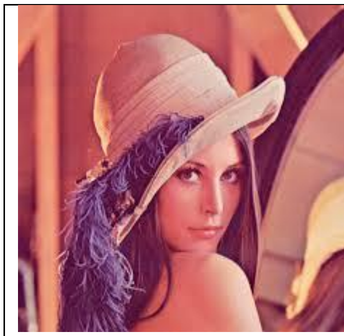
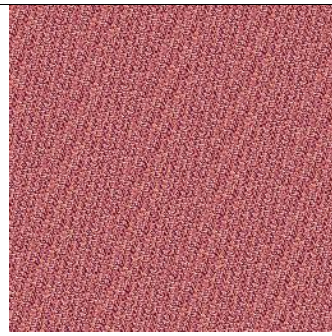
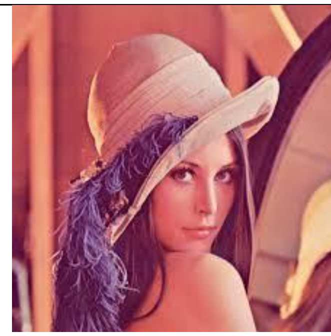
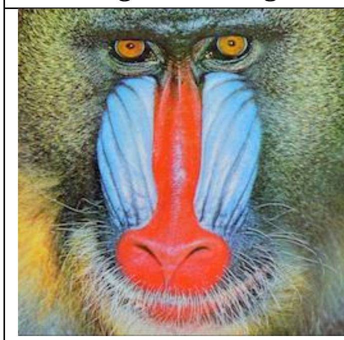
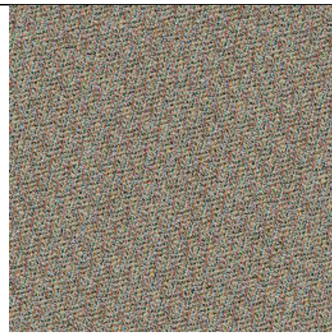
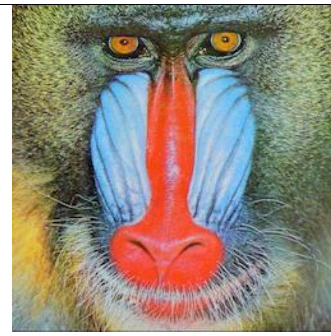
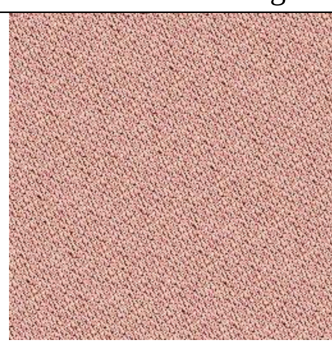
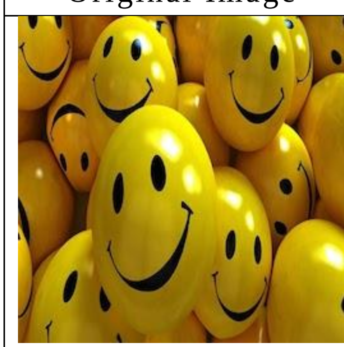
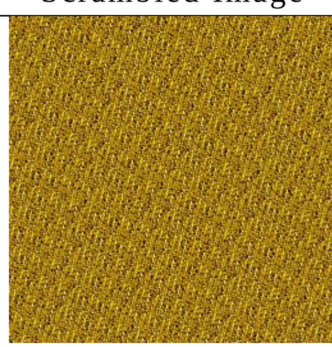
4.6.2 Unscrambling Algorithm:

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

4.6.3 Results:

Parameters taken during execution:-

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

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

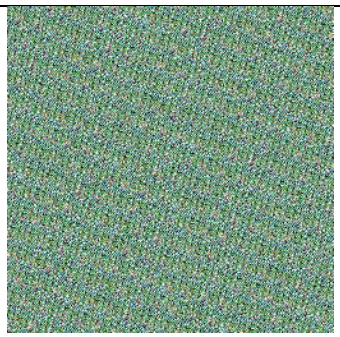
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.6.1: Output

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.	TajMahal	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

Table 4.6.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	0.0170
14.	Temple	0.0082
15.	Watch	0.0262

Table 4.6.3: SSIM Results

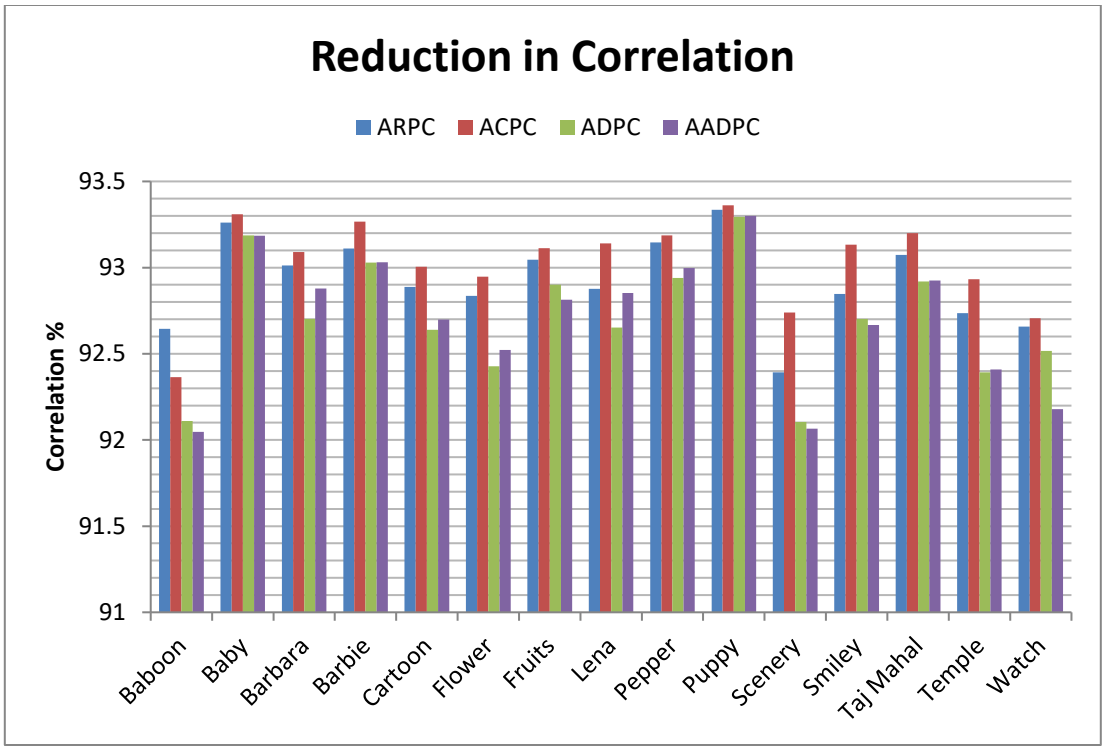


Fig 4.6: Reduction In Correlation(Queue Transform)

4.6 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,……N-1} for an image of size N*N and is represented as follows:

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

Then,

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

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.

4.7.1 Scrambling Formula:

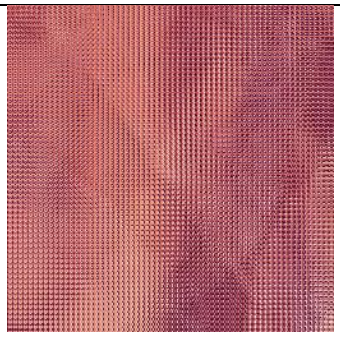
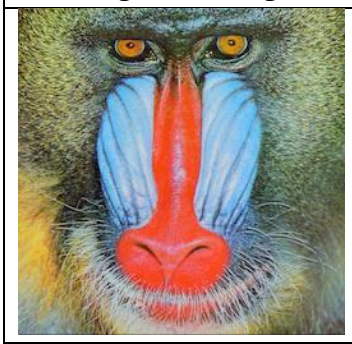
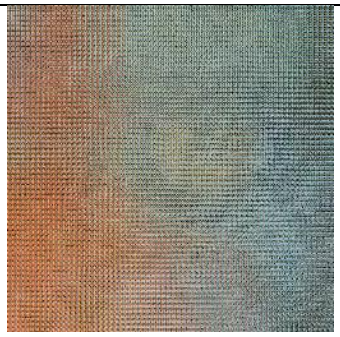
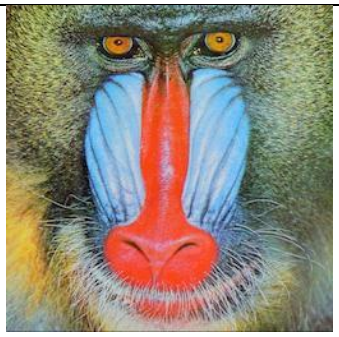
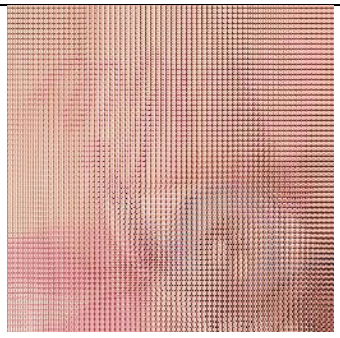
$$E = L * I * L^T$$

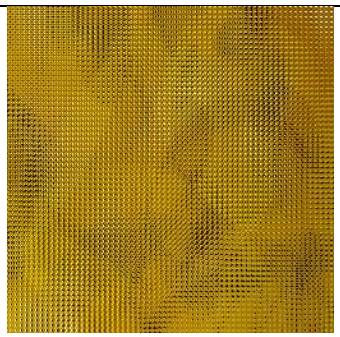
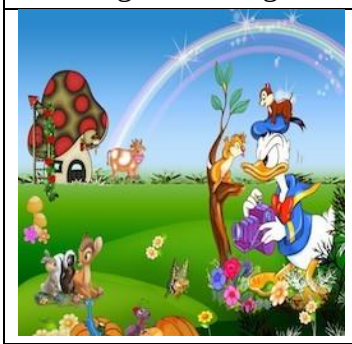
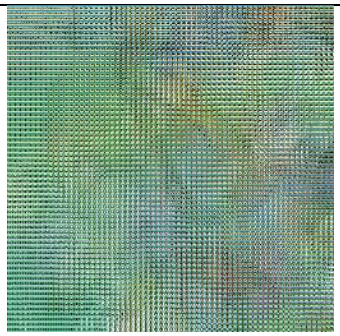
4.7.2 Unscrambling Formula:

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

4.7.3 Results:

Iteration value=20

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

		
(a)	(b)	(c)
Original Image	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	0.0649	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.	TajMahal	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

Table 4.7.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	0.0134
14.	Temple	0.0053
15.	Watch	0.0272

Table 4.7.3: SSIM Results

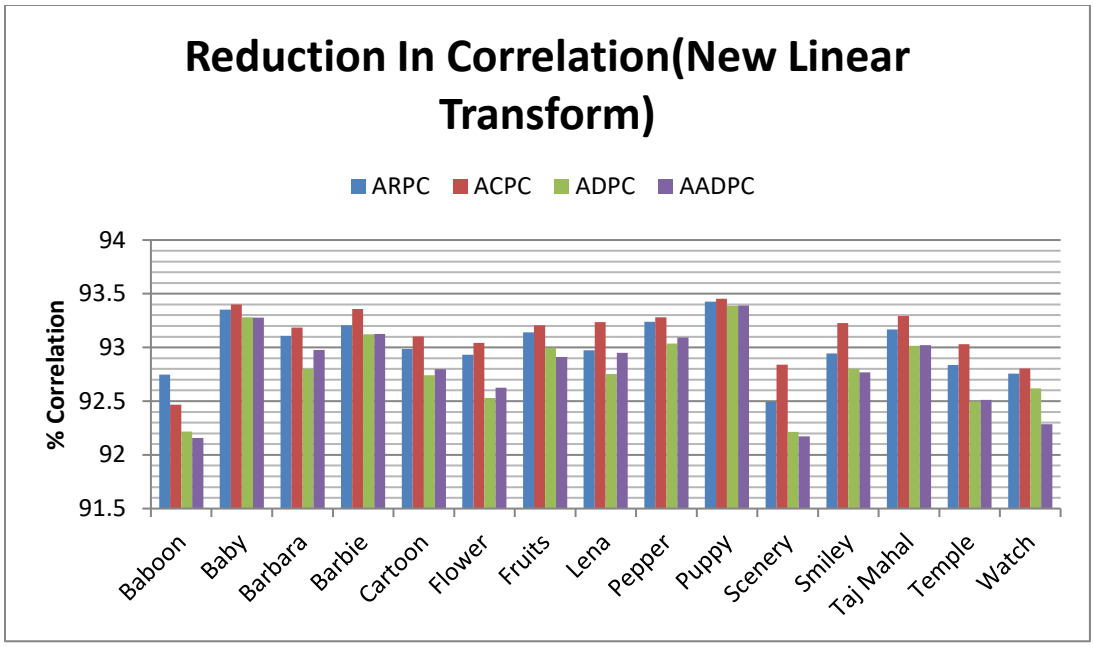


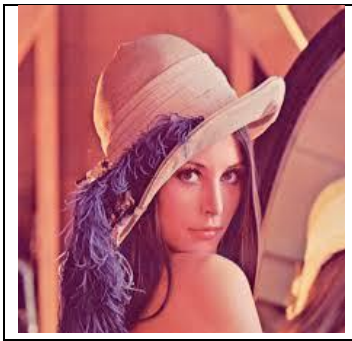
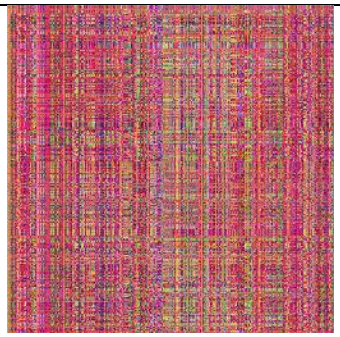
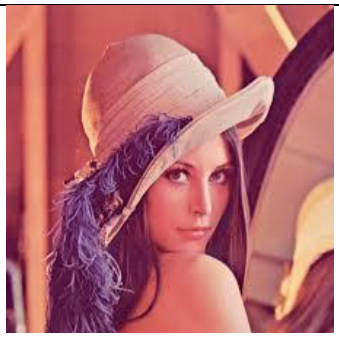
Fig 4.7: Reduction In Correlation(New Linear Transform)

4.8 Gray Code

4.8.1 Results:

Parameters taken during execution:-

Number of times=2; base value=10

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

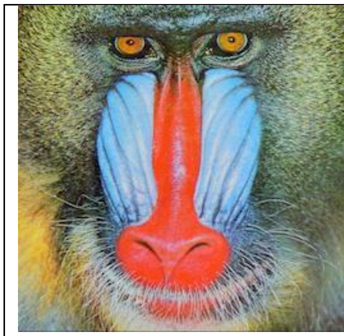
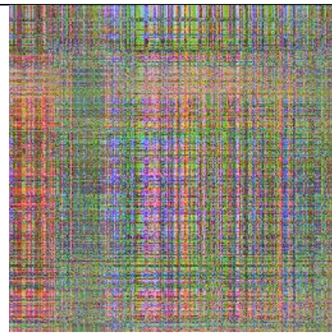
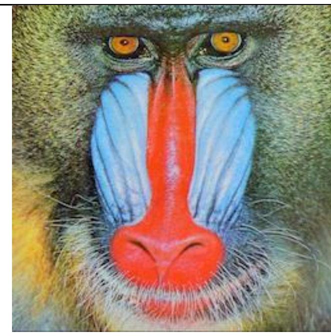
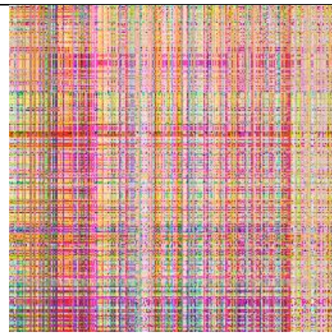
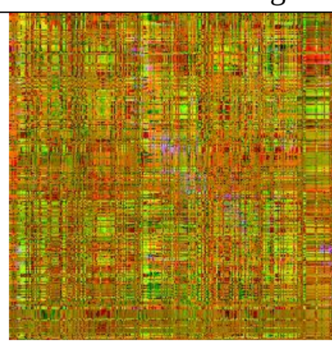
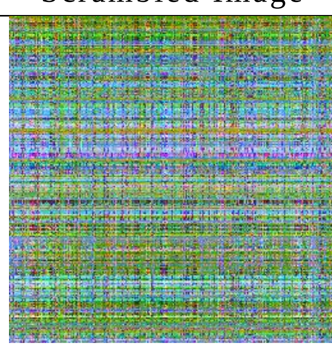
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image
		
(a)	(b)	(c)
Original Image	Scrambled Image	Descrambled Image

Table 4.8.1: Output

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.	TajMahal	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

Table 4.8.2: Correlation Results

Sr.No.	Image Name	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.	TajMahal	0.0395
14.	Temple	0.0143

Table 4.8.3: SSIM Results

CHAPTER 5

IMAGE ENCRYPTION IN THE TRANSFORM DOMAIN

5.1 Introduction

This chapter explains image encryption in the transform domain which include sinusoidal and non-sinusoidal transforms such as Discrete Cosine transform, Discrete Sine Transform, Discrete Fourier transform, Slant Transform and Kekre transform. Results for the above mentioned transforms have also been shown as implemented on input image Lena.

5.2 Sinusoidal Transforms:

5.2.1 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 5.1: DCT formula

5.2.2 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

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

Figure 5.2: DST formula

5.2.3 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 5.3: DFT formula

5.2.4 Hartley Transform

The Discrete Hartley Transform pair is defined for a real- valued length-N sequence x(n), 0≤ n ≤ N-1, by the following equation

$$H(k) = \sum_{n=0}^{N-1} x(n) cas \left(\frac{2\pi}{N} kn\right) \quad 0 \leq k \leq N-1)$$

Figure 5.4: Hartley formula

5.2.5 Real Fourier Transform

The real Fourier transform is given by

$$\left.\begin{aligned} F(k) &= \frac{1}{N} \sum_{m=0}^{N-1} \sin \frac{\pi(k+1)(2m+1)}{2N} \\ F(k+1) &= \frac{1}{N} \sum_{m=0}^{N-1} \cos \frac{\pi(k+1)(2m+1)}{2N}, k = 0,2,\dots,(N-2) \end{aligned}\right\}$$

Where,

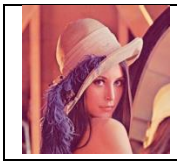
$$f(m) = 2 \sum_{\substack{k=0 \\ k=even}}^{\cdot\cdot\cdot} F(k) \sin \frac{\pi(k+1)(2m+1)}{2N} +$$

$$2 \sum_{\substack{k=0 \\ k=even}}^{N-2} F(k+1) X \cos \frac{\pi(k+1)(2m+1)}{2N}, m = 0,1, \ldots., (N-1)$$

Figure 5.5: Real fourier transform formula

5.3 Results for Sinusoidal based transforms and key based scrambling

5.3.1 Results for Lena

	**Original Image : Lena Row: 0.8439 Col:0.6937**					
Transforms	Row Transform	Row Transform Scrambled	Column Transform	Column Transform Scrambled	Full Transform	Full Transform Scrambled
DCT	Row: 0.9938 Col: 0.1958	Row: 0.5546 Col: 0.3890	Row: 0.1907 Col: 0.9932	Row: 0.4802 Col: 0.6521	Row: 0.2062 Col: 0.2202	Row: 0.5576 Col: 0.5835
DST	Row: 0.9940 Col: 0.2079	Row: 0.5383 Col: 0.3316	Row: 0.1910 Col:0.9909	Row: 0.4543 Col: 0.5676	Row: 0.3048 Col: 0.3395	Row: 0.5271 Col:0.5232
Real Fourier	Row: 0.9956 Col:0.2131	Row: 0.5771 Col: 0.2237	Row: 0.1901 Col:0.9930	Row: 0.4507 Col: 0.2054	Row: 0.2093 Col:0.2345	Row: 0.3222 Col:0.2601
Hartley	Row: 0.9947 Col:0.2136	Row: 0.2385 Col: 0.2224	Row: 0.2160 Col:0.9931	Row: 0.1969 Col:0.2175	Row: 0.3966 Col: 0.4995	Row: 0.2096 Col:0.2048
DFT	Row:	Row:	Row:	Row:	Row:	Row:

	0.9961 Col: 0.1736	0.2697 Col: 0.1677	0.1860 Col: 0.9961	0.1308 Col:0.2117	0.4178 Col: 0.4599	0.1877 Col: 0.1780

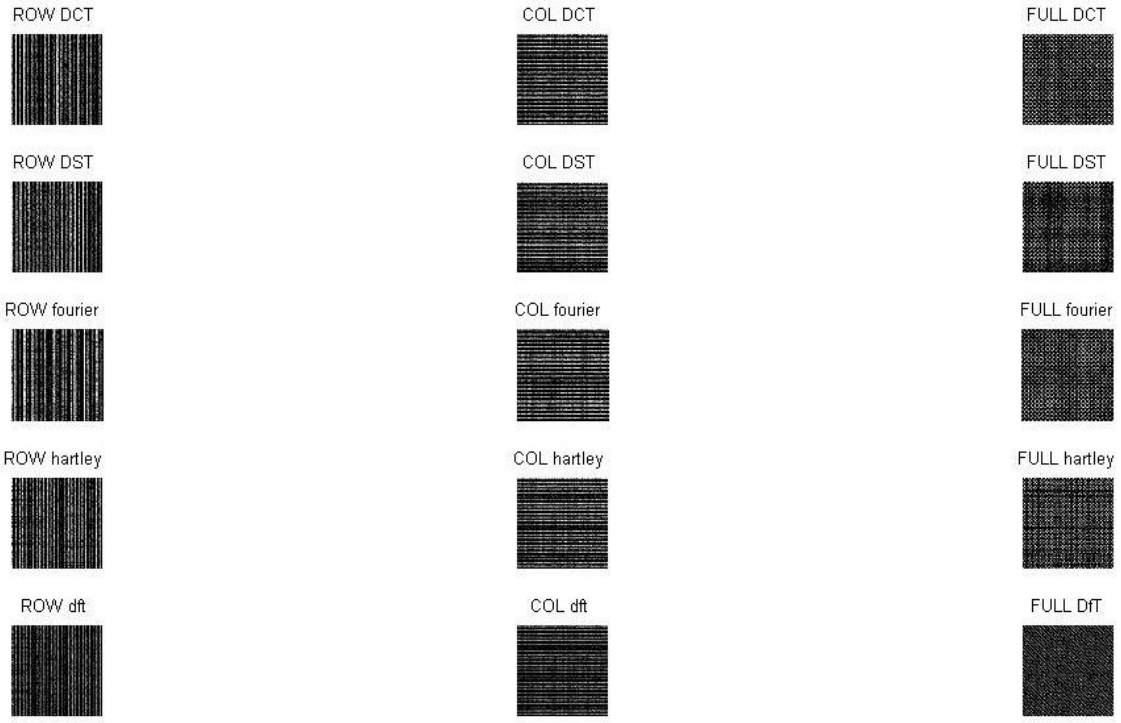


Fig 5.6: Output for Lena(Sinusoidal)

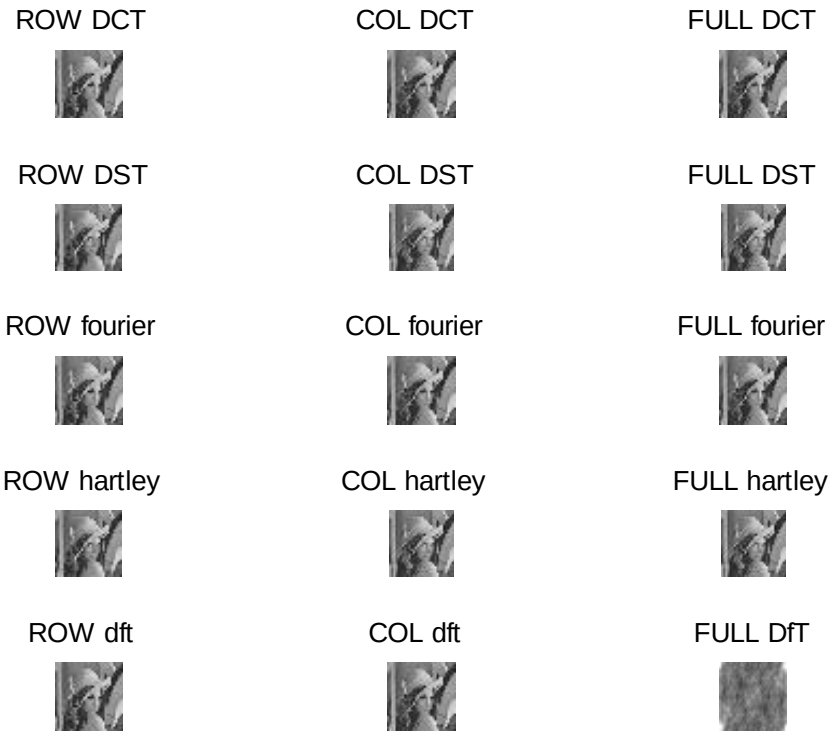


Fig 5.7: Output for Recovered Lena(Sinusoidal)

5.3.2 Results for Baboon

	**Original Image : Baboon Row: 0.7179 Col:0.6935**					
Transforms	Row Transform	Row Transform Scrambled	Column Transform	Column Transform Scrambled	Full Transform	Full Transform Scrambled
DCT	Row: 0.9912 Col: 0.2006	Row: 0.6365 Col: 0.3851	Row: 0.2721 Col: 0.9935	Row: 0.3866 Col: 0.6583	Row: 0.2778 Col: 0.2159	Row:0.5723 Col: 0.5739
DST	Row: 0.9923 Col: 0.2029	Row: 0.7196 Col: 0.3702	Row: 0.2912 Col: 0.9927	Row: 0.3789 Col: 0.5446	Row: 0.3865 Col: 0.3444	Row:0.4690 Col: 0.5042
Real Fourier	Row: 0.9953	Row: 0.6600	Row: 0.1987	Row: 0.3794	Row: 0.2179	Row:0.2420 Col: 0.2749

	Col: 0.2096	Col: 0.2457	Col: 0.9937	Col: 0.1922	Col: 0.2179	
Hartley	Row: 0.9928 Col: 0.2144	Row: 0.2002 Col: 0.2370	Row: 0.2633 Col: 0.9937	Row: 0.1911 Col: 0.1918	Row: 0.4242 Col: 0.3475	Row:0.1825 Col: 0.1937
DFT	Row: 0.9961 Col: 0.1684	Row: 0.1945 Col: 0.1575	Row: 0.2569 Col: 0.9961	Row: 0.1350 Col: 0.1754	Row: 0.3952 Col: 0.2750	Row:0.1422 Col: 0.1430

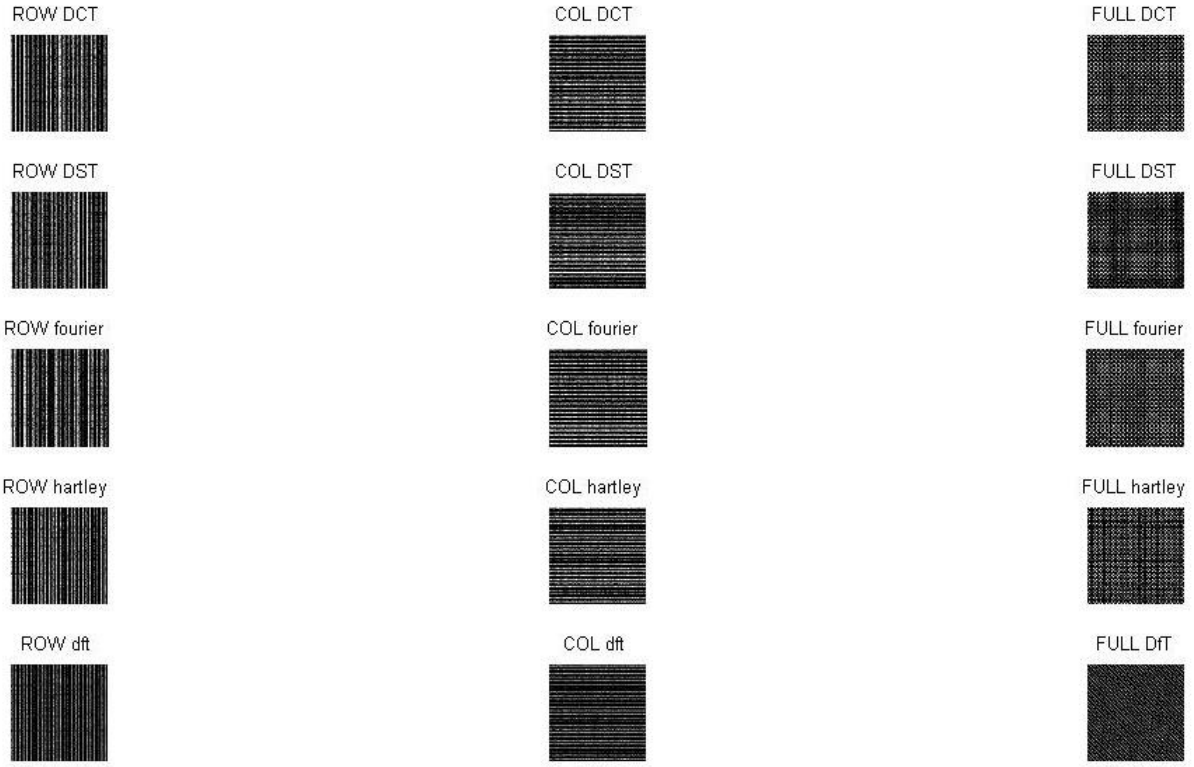


Fig 5.8: Output for Baboon(Sinusoidal)

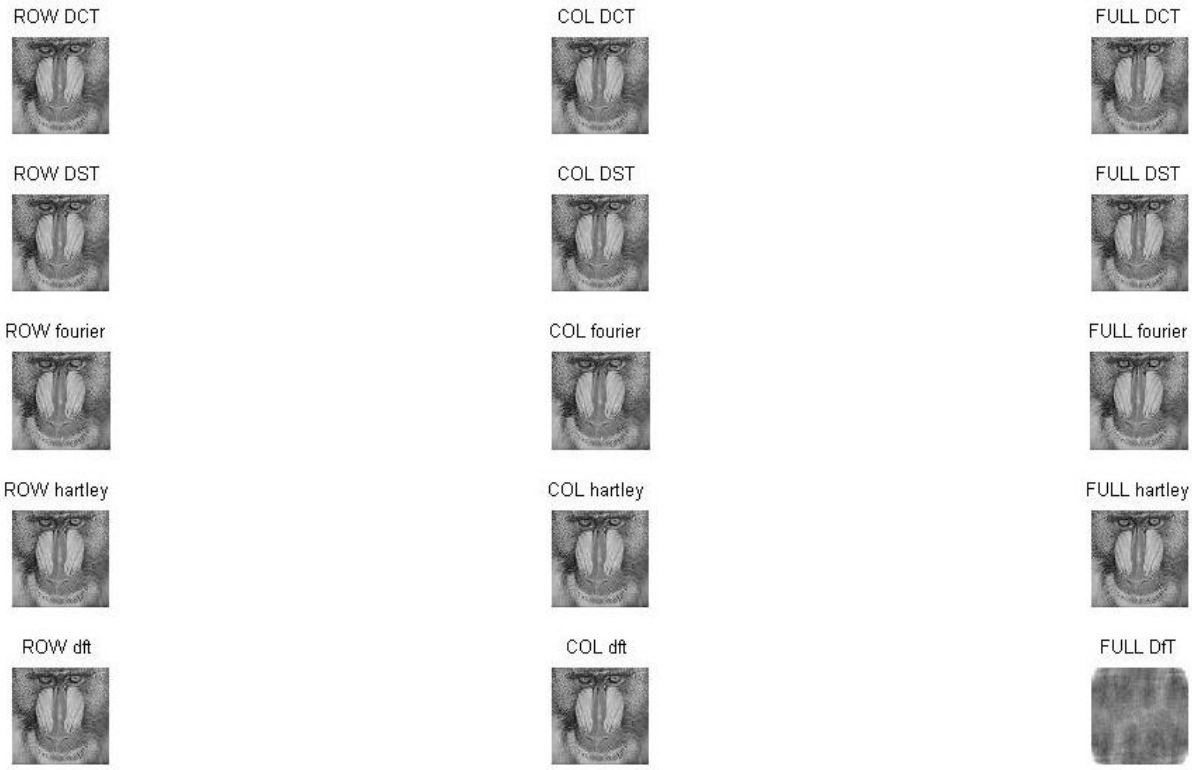


Fig 5.9: Output for Recovered Baboon(Sinusoidal)

5.3.3 Results for Pepper

	**Original Image : Pepper Row: 0.8891 Col: 0.8468**					
Transforms	Row Transform	Row Transform Scrambled	Column Transform	Column Transform Scrambled	Full Transform	Full Transform Scrambled
DCT	Row: 0.9926 Col: 0.1890	Row: 0.5381 Col: 0.3483	Row: 0.1888 Col: 0.9932	Row: 0.4156 Col: 0.6019	Row: 0.2278 Col: 0.3145	Row: 0.5571 Col: 0.5680
DST	Row: 0.9919 Col: 0.1892	Row: 0.5707 Col: 0.3907	Row: 0.2013 Col: 0.9899	Row: 0.3830 Col: 0.5207	Row: 0.3547 Col: 0.2044	Row: 0.4879 Col: 0.5080
Real Fourier	Row: 0.9955	Row: 0.5126	Row: 0.1892	Row: 0.3722	Row: 0.2384	Row: 0.4108

	Col: 0.2018	Col: 0.2291	Col: 0.9929	Col: 0.2334	Col: 0.2374	Col: 0.3003
Hartley	Row: 0.9941 Col: 0.2004	Row: 0.3166 Col: 0.1965	Row: 0.2032 Col: 0.9933	Row: 0.1857 Col: 0.2439	Row: 0.5053 Col: 0.5722	Row: 0.2656 Col: 0.2227
DFT	Row: 0.9961 Col: 0.1559	Row: 0.3901 Col: 0.1383	Row: 0.1605 Col: 0.9961	Row: 0.1265 Col: 0.2696	Row: 0.4932 Col: 0.5445	Row: 0.2429 Col: 0.2116

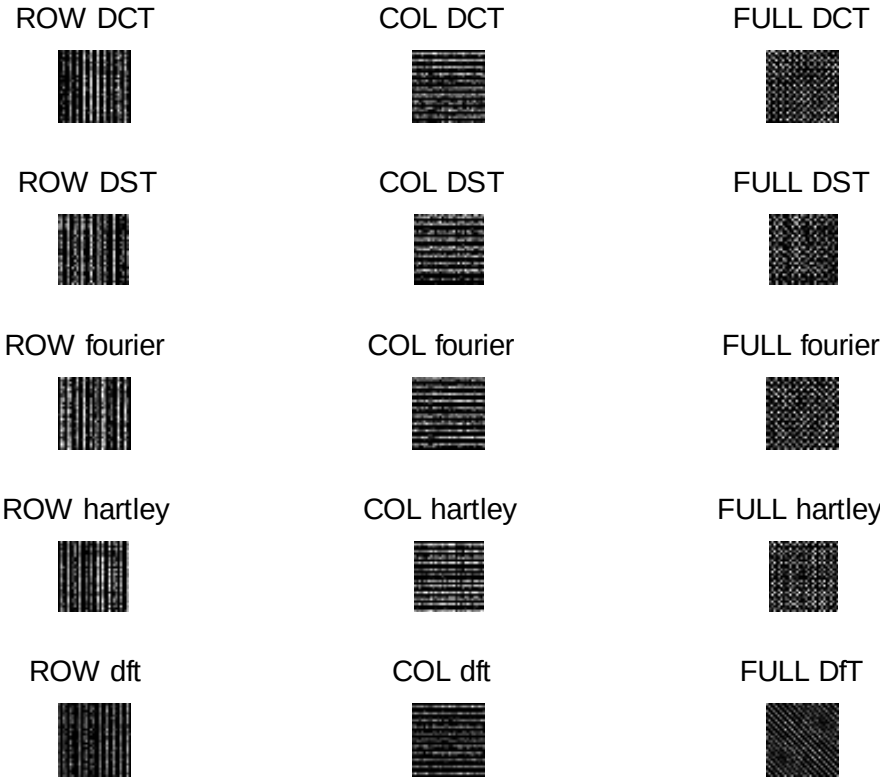


Fig 5.10: Output for Pepper(Sinusoidal)

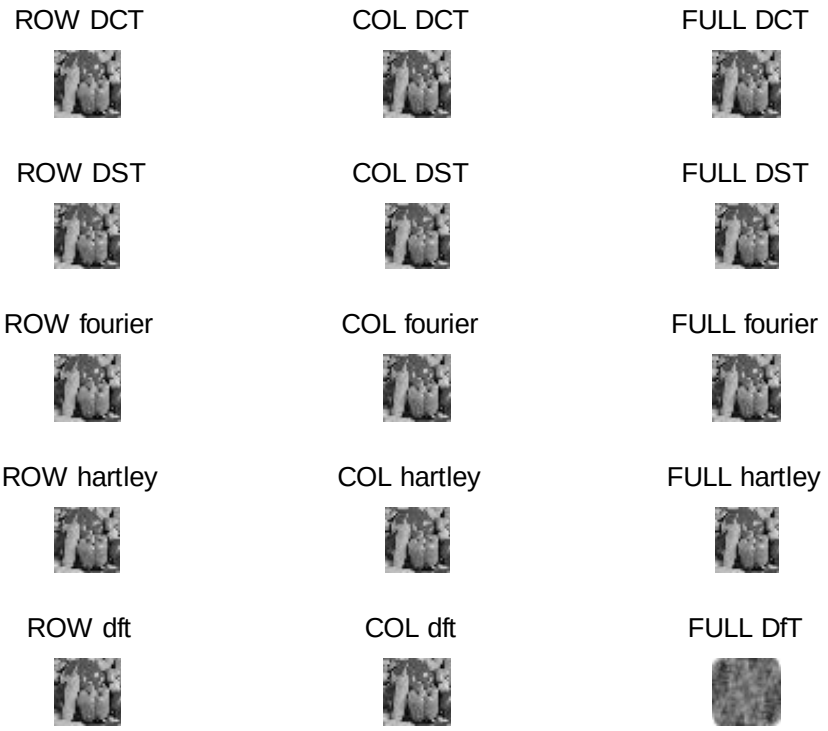


Fig 5.11: Output for Recovered Pepper(Sinusoidal)

5.3.4 Results for Lotus

	**Original Image : Lotus Row: 0.8163 Col:0.8461**					
Transforms	Row Transform	Row Transform Scrambled	Column Transform	Column Transform Scrambled	Full Transform	Full Transform Scrambled
DCT	Row:0.9839 Col: 0.1847	Row: 0.3744 Col: 0.4335	Row: 0.1950 Col: 0.9915	Row: 0.3806 Col: 0.4547	Row: 0.2391 Col: 0.2292	Row: 0.4567 Col: 0.5137
DST	Row: 0.9795 Col: 0.1901	Row: 0.4459 Col: 0.4601	Row: 0.1996 Col: 0.9879	Row: 0.3821 Col: 0.2824	Row:0.3615 Col: 0.3170	Row: 0.4182 Col: 0.4148
Real Fourier	Row: 0.9918 Col: 0.1914	Row: 0.3810 Col:	Row: 0.2012 Col:	Row: 0.3649 Col:	Row: 0.2531 Col: 0.2311	Row: 0.2906 Col:

		0.2458	0.9892	0.2126		0.3091
Hartley	Row: 0.9884 Col: 0.2290	Row: 0.2134 Col: 0.2146	Row: 0.1860 Col: 0.9915	Row: 0.2061 Col: 0.2148	Row: 0.3902 Col: 0.4168	Row: 0.2153 Col: 0.1902
DFT	Row: 0.9939 Col: 0.1908	Row: 0.2007 Col: 0.1426	Row: 0.1369 Col: 0.9954	Row: 0.1415 Col: 0.2143	Row: 0.3442 Col: 0.3701	Row: 0.1844 Col: 0.1405

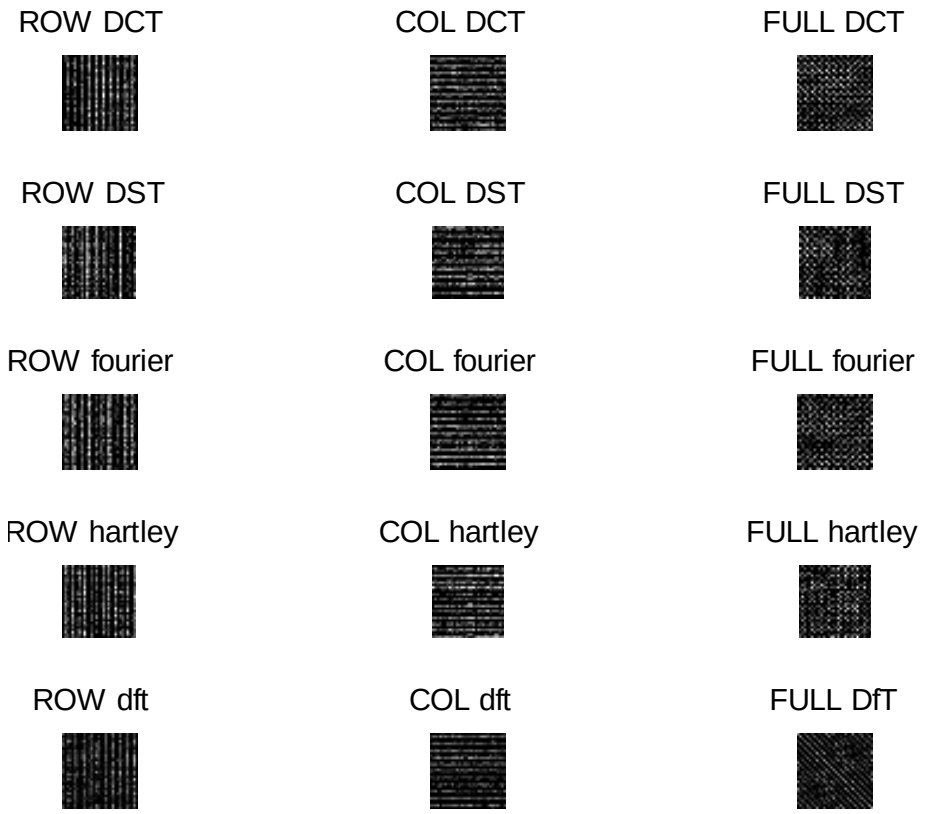


Fig 5.12: Output for Lotus(Sinusoidal)

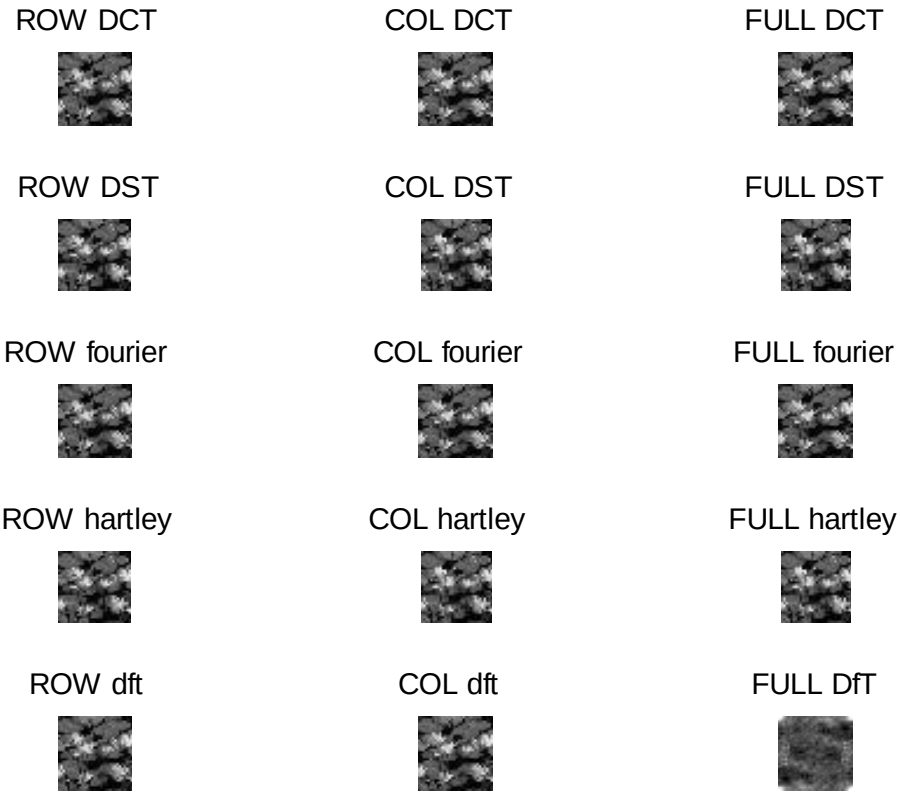


Fig 5.13: Output for Recovered Lotus(Sinusoidal)

5.3.5 Results for Cartoon

	**Original Image : Cartoon Row: 0.8027 Col:0.8070**					
Transforms	Row Transform	Row Transform Scrambled	Column Transform	Column Transform Scrambled	Full Transform	Full Transform Scrambled
DCT	Row: 0.9945 Col: 0.1887	Row: 0.7014 Col: 0.5178	Row: 0.2556 Col: 0.9955	Row: 0.5230 Col: 0.7489	Row: 0.2897 Col: 0.2238	Row:0.6601 Col: 0.6596
DST	Row: 0.9944 Col:0.2056	Row: 0.8154 Col: 0.4576	Row: 0.2646 Col: 0.9953	Row: 0.4515 Col: 0.6254	Row: 0.3048 Col: 0.5047	Row: 0.5924 Col: 0.5571

Real Fourier	Row: 0.9959 Col: 0.2185	Row: 0.7298 Col: 0.2971	Row: 0.2217 Col: 0.9954	Row: 0.4903 Col: 0.2254	Row: 0.2613 Col: 0.2383	Row: 0.3178 Col: 0.3187
Hartley	Row: 0.9949 Col: 0.2605	Row: 0.2700 Col: 0.2675	Row: 0.1983 Col:0.9954	Row: 0.2046 Col: 0.2337	Row: 0.3954 Col: 0.5162	Row:0.2412 Col: 0.2146
DFT	Row: 0.9961 Col: 0.2406	Row: 0.3027 Col: 0.1623	Row: 0.1492 Col: 0.9961	Row: 0.1299 Col: 0.2483	Row: 0.3068 Col: 0.4806	Row: 0.2410 Col: 0.1872

Fig 5.14: Output for Cartoon(Sinusoidal)

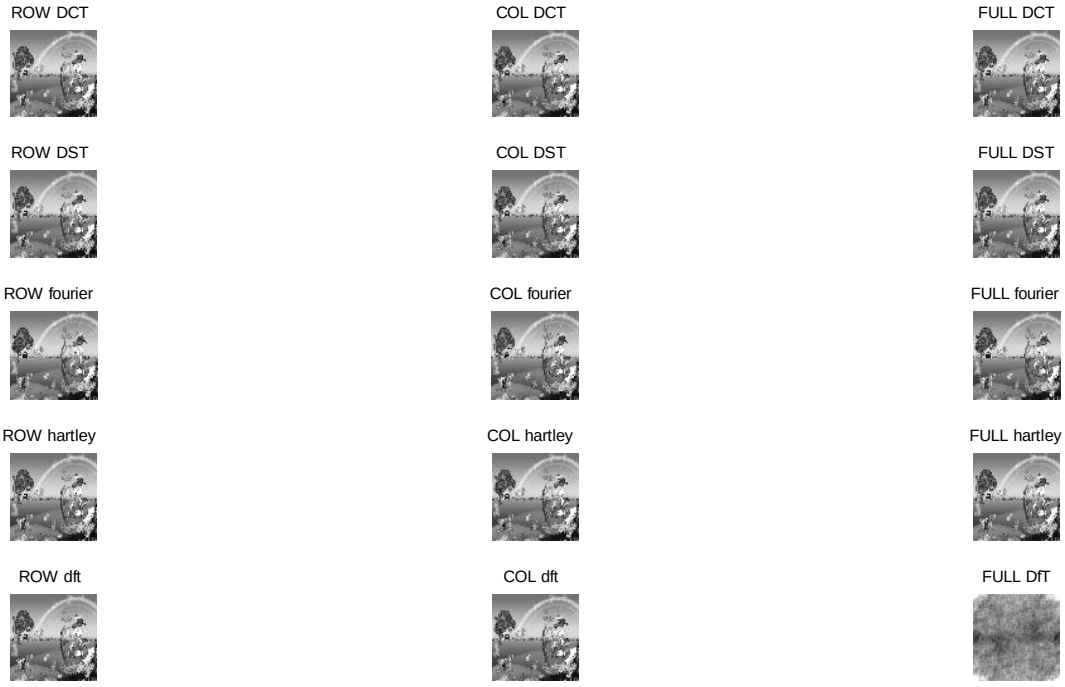


Fig 5.15: Output for Recovered Cartoon(Sinusoidal)

5.4 Non-Sinusoidal Transforms:

5.4.1 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

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

Slant transform has the following properties:

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

5.4.2 Kekre transform:

Kekre transform for NxN matrix can be given as:

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

Figure 5.16: Kekre transform

5.4.3 Hadamard transform:

Hadamard transform is an example of a Fourier transform. It is orthogonal and symmetric in nature. It is given by

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Figure 5.17: Hadamard transform

5.4.4 Walsh transform:

Walsh transform is an example of a Fourier transform. It is orthogonal and symmetric in nature. It is given by

$$W1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Figure 5.18: Walsh Transform

5.4.5 Haar transform:

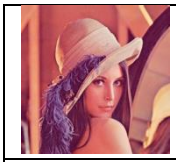
It is a sequence of square shaped functions which together form a wavelet family. It is orthogonal in nature. It is given by

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2}, \\ -1 & \frac{1}{2} \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 5.19: Haar Transform

5.5 Results for Non-Sinusoidal based transforms and key based scrambling

5.5.1 Results for Lena

	**Original Image : Lena Row: 0.8439 Col:0.6937**					
Transforms	**Row Transform**	**Row Transform Scrambled**	**Column Transform**	**Column Transform Scrambled**	**Full Transform**	**Full Transform Scrambled**
Walsh	Row: 0.9947 Col:0.2088	Row: 0.3520 Col:0.3283	Row: 0.1938 Col:0.9928	Row: 0.4249 Col:0.3380	Row: 0.2026 Col: 0.2351	Row: 0.2026 Col: 0.2351
Slant	Row:0.8916 Col: 0.3899	Row: 0.8429 Col:0.2065	Row: 0.3303 Col: 0.9931	Row: 0.6336 Col:0.8118	Row: 0.4060 Col:0.4289	Row: 0.6272 Col:0.5867
Kekre	Row: 0.9924 Col:0.7046	Row: 0.8102 Col:0.2284	Row: 0.7884 Col:0.9924	Row: 0.2517 Col:0.9359	Row: 0.8944 Col:0.8831	Row: 0.2515 Col:0.2848
Haar	Row: 0.9122 Col:0.1969	Row: 0.8345 Col:0.2048	Row: 0.2382 Col:0.9928	Row: 0.6088 Col:0.6960	Row: 0.2466 Col: 0.2794	Row: 0.5935 Col:0.5548

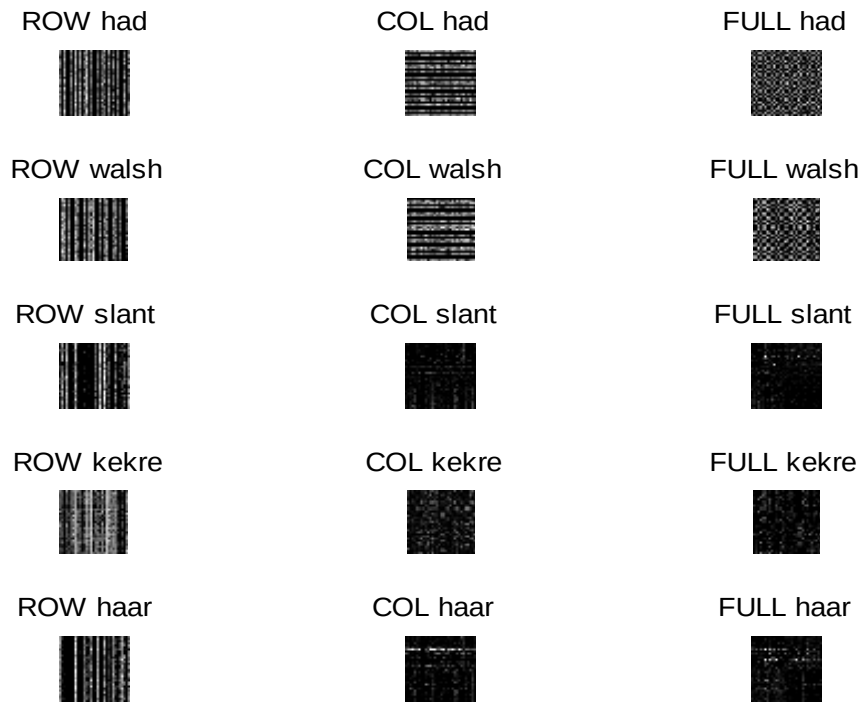


Fig 5.20: Output for Lena(Non-Sinusoidal)

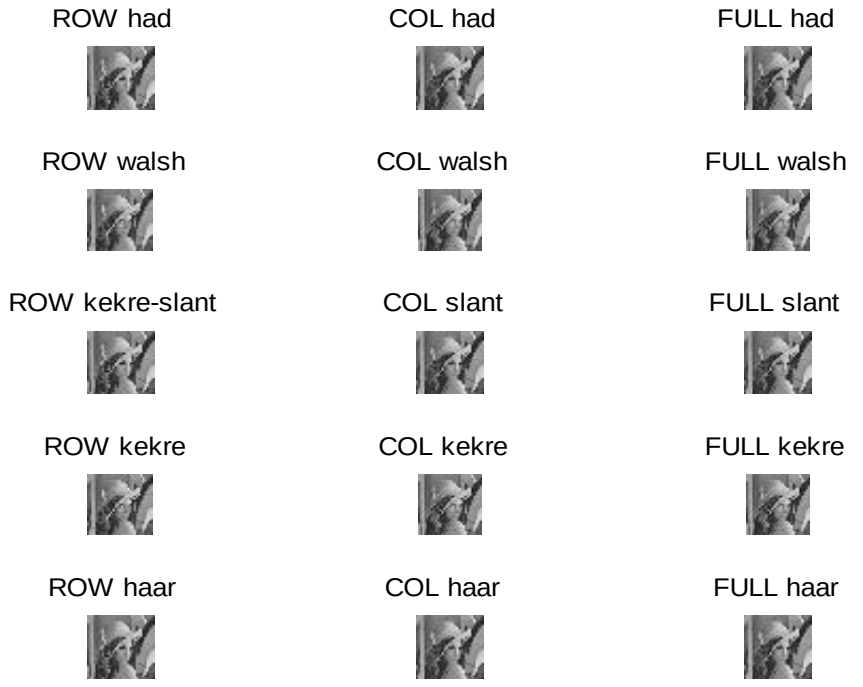


Fig 5.21: Output for Recovered Lena(Non-Sinusoidal)

5.5.2 Results for Baboon

	**Original Image : Baboon Row: 0.7179 Col:0.6935**					
Transforms	**Row Transform**	**Row Transform Scrambled**	**Column Transform**	**Column Transform Scrambled**	**Full Transform**	**Full Transform Scrambled**
Walsh	Row: 0.9932 Col:0.2117	Row: 0.3331 Col:0.3686	Row: 0.2436 Col:0.9937	Row: 0.3087 Col:0.3147	Row: 0.2564 Col:0.2250	Row: 0.3766 Col: 0.4097
Slant	Row: 0.7745 Col:0.4015	Row: 0.8865 Col:0.2220	Row: 0.2910 Col:0.9937	Row: 0.4912 Col:0.8591	Row: 0.3603 Col:0.3813	Row: 0.4948 Col:0.5261
Kekre	Row: 0.9875 Col: 0.6756	Row: 0.8610 Col: 0.2429	Row: 0.6758 Col:0.9934	Row: 0.2563 Col:0.9601	Row: 0.8523 Col:0.8342	Row: 0.3000 Col: 0.2667
Haar	Row: 0.8299 Col: 0.1984	Row: 0.8725 Col:0.2215	Row: 0.1940 Col:0.9937	Row: 0.4702 Col: 0.7401	Row: 0.2026 Col:0.2035	Row: 0.4734 Col: 0.4943

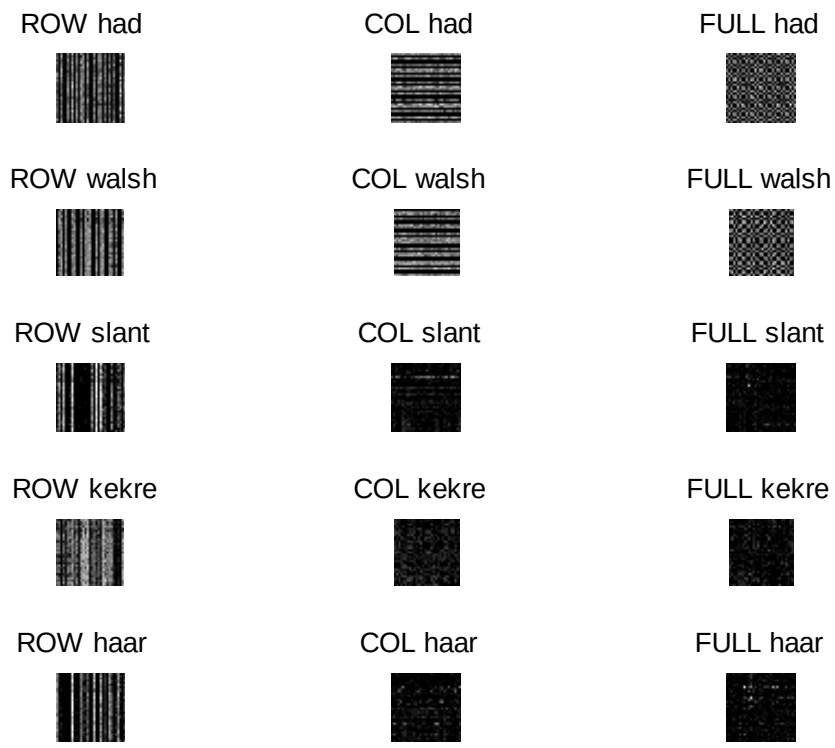


Fig 5.22: Output for Baboon(Non-Sinusoidal)

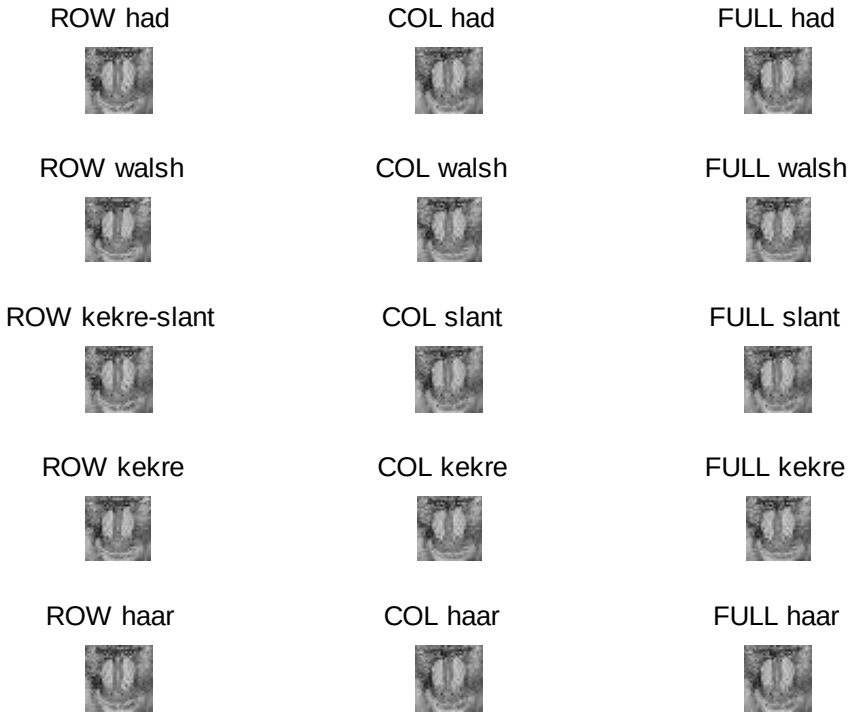


Fig 5.23: Output for Recovered Baboon(Non-Sinusoidal)

5.5.3 Results for Pepper

	**Original Image : Pepper Row: 0.8891 Col: 0.8468**					
Transforms	**Row Transform**	**Row Transform Scrambled**	**Column Transform**	**Column Transform Scrambled**	**Full Transform**	**Full Transform Scrambled**
Walsh	Row: 0.9940 Col: 0.1990	Row: 0.3498 Col:0.3344	Row: 0.1965 Col: 0.9929	Row: 0.3368 Col: 0.3353	Row: 0.2415 Col: 0.2686	Row: 0.4102 Col:0.4171
Slant	Row: 0.9153 Col:0.5000	Row: 0.7708 Col: 0.2248	Row: 0.3037 Col:0.9922	Row: 0.6537 Col:0.7370	Row: 0.4033 Col:0.4236	Row: 0.6477 Col:0.5913
Kekre	Row: 0.9933 Col:0.8470	Row: 0.7393 Col: 0.2306	Row: 0.8334 Col:0.9927	Row: 0.2766 Col: 0.8963	Row: 0.9300 Col:0.9159	Row: 0.2808 Col:0.2462
Haar	Row: 0.9381 Col:0.2251	Row: 0.7748 Col:0.2226	Row: 0.2202 Col:0.9923	Row: 0.6153 Col:0.6225	Row: 0.2963 Col: 0.3150	Row: 0.6205 Col: 0.5891

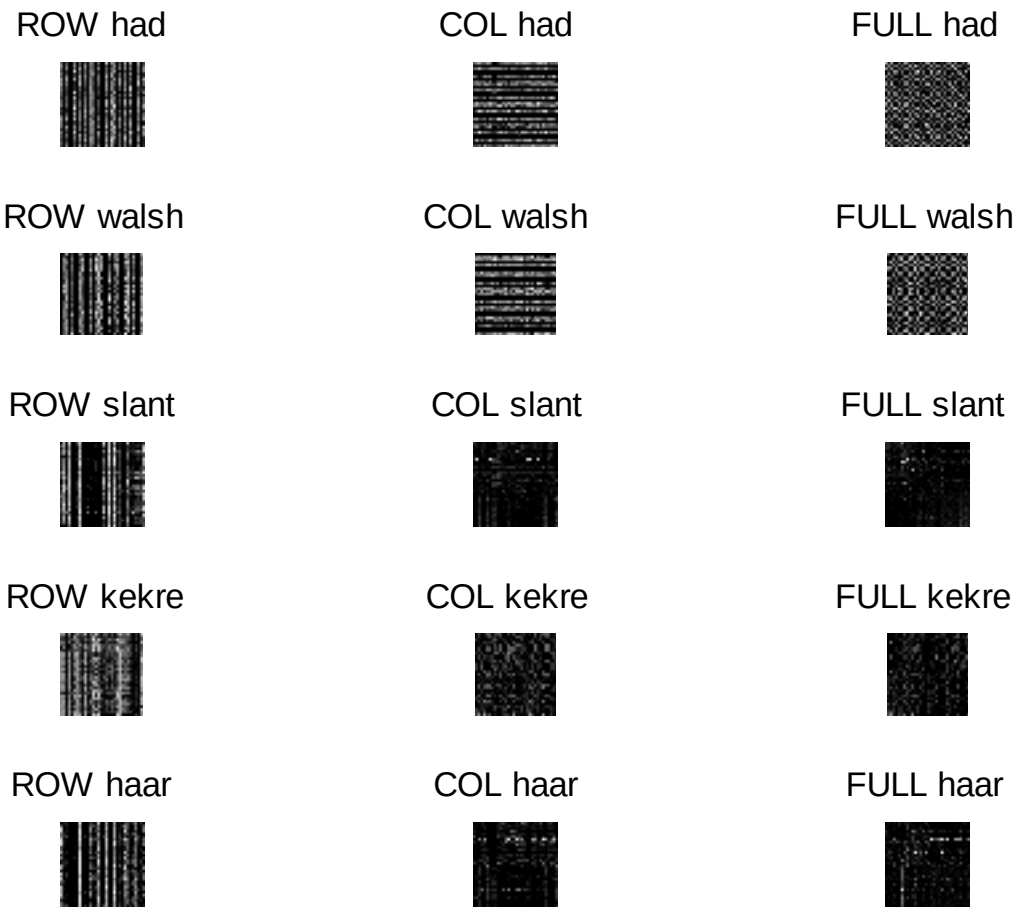


Fig 5.24: Output for Pepper (Non-Sinusoidal)

Fig 5.25: Output for Recovered Pepper (Non-Sinusoidal)

5.5.4 Results for Lotus

	**Original Image : Lotus Row: 0.8163 Col:0.8461**					
Transforms	**Row Transform**	**Row Transform Scrambled**	**Column Transform**	**Column Transform Scrambled**	**Full Transform**	**Full Transform Scrambled**
Walsh	Row: 0.9857 Col:0.1934	Row: 0.2906 Col: 0.4174	Row: 0.2093 Col:0.9885	Row: 0.3509 Col:0.3108	Row: 0.2333 Col:0.2295	Row: 0.3844 Col:0.4271
Slant	Row: 0.8693 Col:0.4610	Row: 0.6497 Col:0.2216	Row: 0.3401 Col:0.9858	Row: 0.5389 Col:0.5948	Row: 0.4066 Col:0.4098	Row: 0.5541 Col: 0.5805
Kekre	Row:	Row:	Row:	Row:	Row:	Row:

	0.9796 Col:0.8487	0.6158 Col:0.2260	0.8196 Col:0.9892	0.2323 Col:0.7988	0.9242 Col:0.8973	0.2528 Col: 0.2409
Haar	Row: 0.8871 Col:0.2111	Row: 0.6503 Col:0.2331	Row: 0.2210 Col: 0.9847	Row: 0.5430 Col:0.5760	Row: 0.2869 Col:0.2730	Row: 0.5479 Col: 0.5605

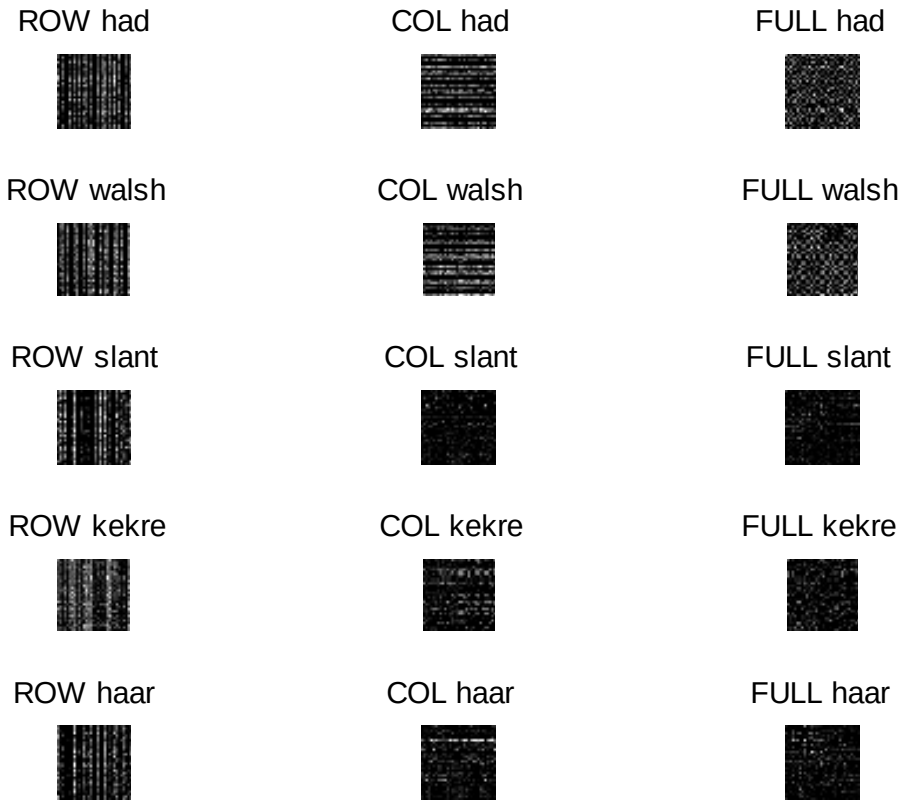


Fig 5.26: Output for Lotus(Non-Sinusoidal)

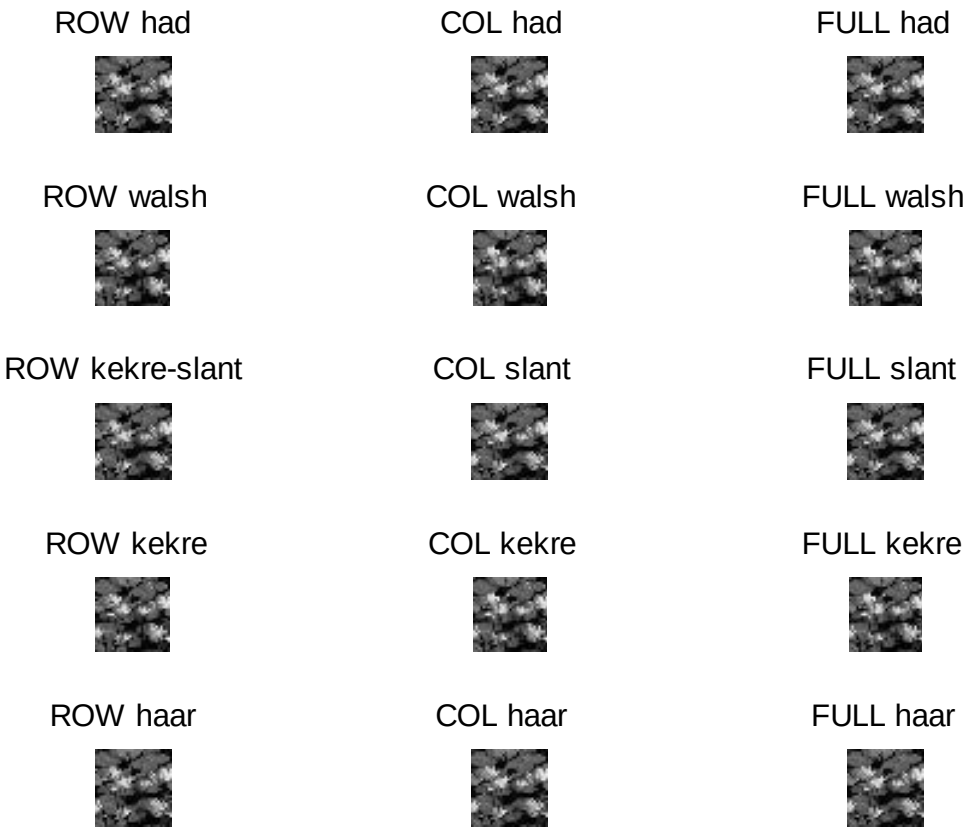
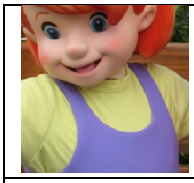


Fig 5.27: Output for Recovered Lotus (Non-Sinusoidal)

5.5.5 Results for Cartoon

	**Original Image : Cartoon Row: 0.8027 Col:0.8070**					
Transforms	**Row Transform**	**Row Transform Scrambled**	**Column Transform**	**Column Transform Scrambled**	**Full Transform**	**Full Transform Scrambled**
Walsh	Row: 0.9951 Col:0.2066	Row: 0.3754 Col:0.4798	Row: 0.2734 Col:0.9954	Row: 0.4359 Col:0.4656	Row: 0.3931 Col:0.2410	Row: 0.4824 Col:0.5344
Slant	Row: 0.8822 Col:0.4816	Row: 0.8810 Col: 0.2522	Row: 0.3171 Col:0.9954	Row: 0.6384 Col:0.8893	Row: 0.3923 Col:0.3992	Row: 0.7195 Col:0.6217

Kekre	Row: 0.9942 Col: 0.7956	Row: 0.8522 Col: 0.2878	Row: 0.7676 Col:0.9953	Row: 0.2629 Col: 0.9591	Row: 0.9577 Col:0.9247	Row: 0.9577 Col:0.9247
Haar	Row: 0.9252 Col:0.2525	Row: 0.8542 Col:0.2626	Row: 0.2592 Col:0.9956	Row: 0.6063 Col:0.8189	Row: 0.3065 Col: 0.2804	Row: 0.6822 Col:0.6449

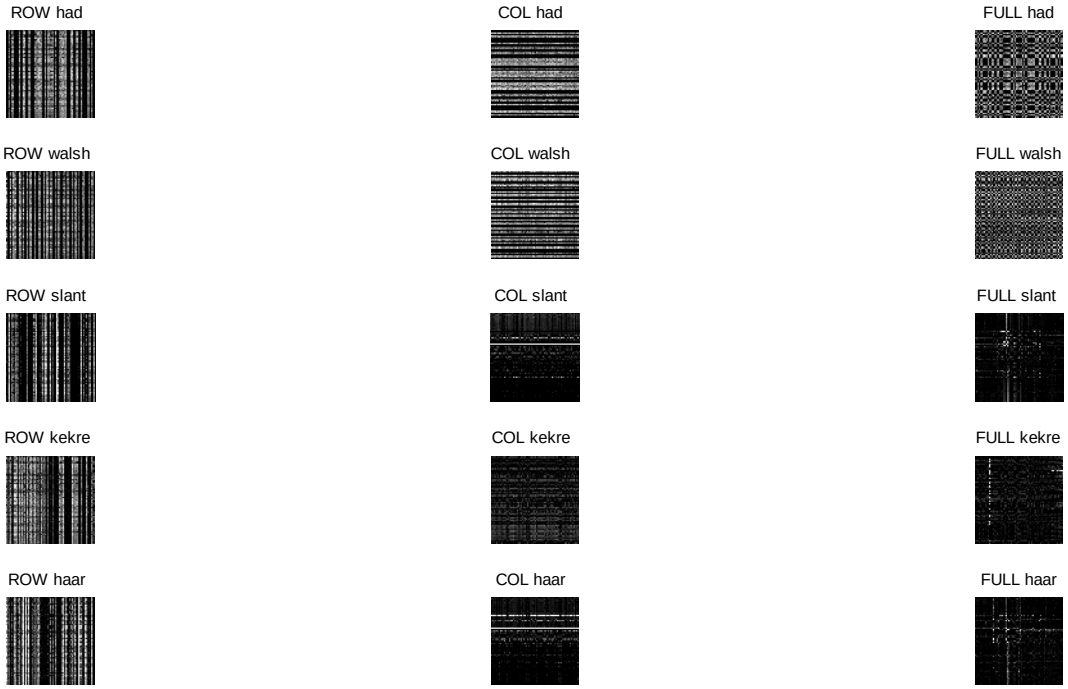


Fig 5.28: Output for Cartoon(Non-Sinusoidal)

Fig 5.29: Output for Recovered Cartoon(Non-Sinusoidal)

5.6 HYBRID TRANSFORMS:

A hybrid transform matrix involves a combination of two transforms. This resultant transform is orthogonal. When this hybrid transform is applied to an image, it gives better encryption as compared to the traditional sinusoidal and non-sinusoidal transforms. Let H be the resultant hybrid transform of size 256x256(say). Let H be the combination of two transforms (either sinusoidal or non sinusoidal) A and B. Then H can be as follows:

H = A*B
where,
A = 16x16 and B = 16x16
A = 8x8 and B = 32x32
A = 4x4 and B= 64x64
A = 2x2 and B = 128x128

When combined this gives us a matrix H of size 256x256. Thus when H is of size ‘n’, then the factors of n are taken into consideration. The size of A and B should be the combination of these factors such that the

multiplication of the two factors gives the size n. Hybrid transforms can include the following transforms:

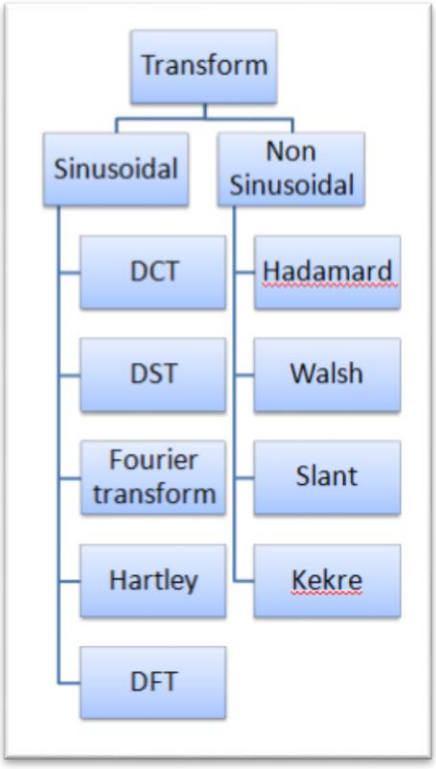


Figure 5.30: Types of transforms that can be used in Hybrid transform

Note that in hybrid transform a sinusoidal transform is only combined with another sinusoidal transform. Similarly, a non-sinusoidal transform is combined with a non-sinusoidal transform only. For sinusoidal, we have considered Hartley as the base transform and have combined it with DCT, DST and Fourier and DFT to obtain Hartley-DCT, Hartley-DST, Hartley-DFT and Hartley-Fourier transforms. For non-sinusoidal, we have considered Kekre as the base transform and have combined it with Hadamard, Walsh and Slant transform to obtain Kekre-Hadamard, Kekre-Walsh and Kekre-Slant transform. The block size of the two can be 16x16, 8x32, 4x64 and 2x128 for the image size of 256x256.

5.6.1 Results for Non- Sinusoidal Hybrid Transforms with Base Kekre

5.6.1.1 Results for Lena

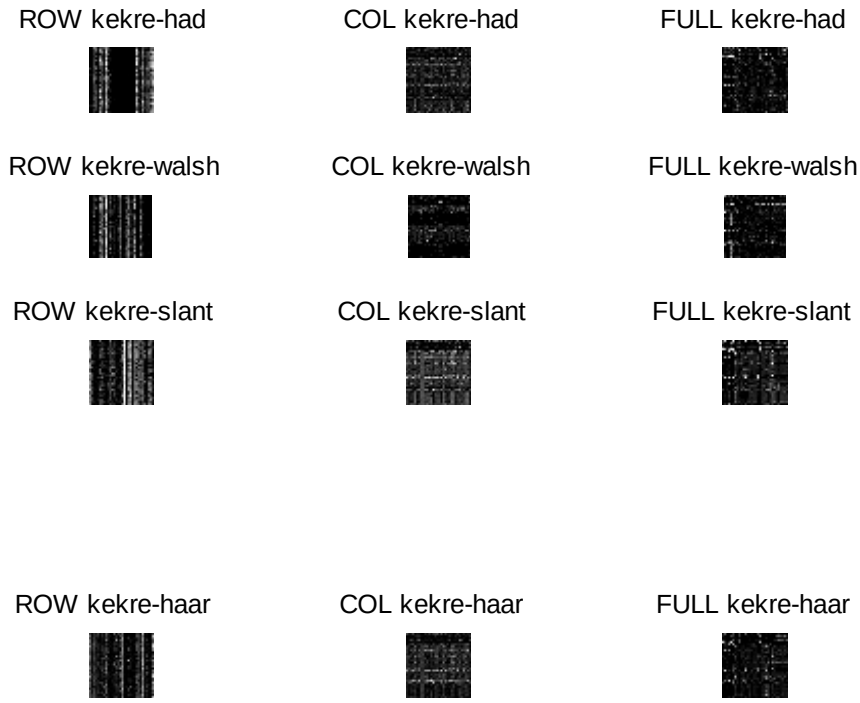


Fig 5.31: Output for Lena Hybrid Transform

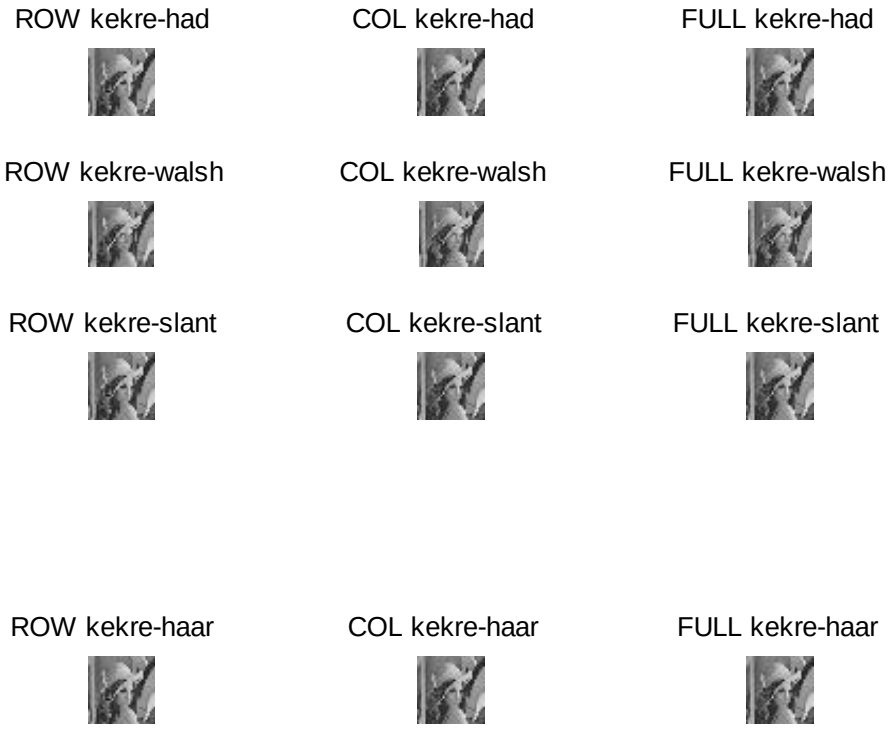


Fig 5.32: Output for Descrambled Lena Hybrid Transform

5.6.1.2 Results for Baboon

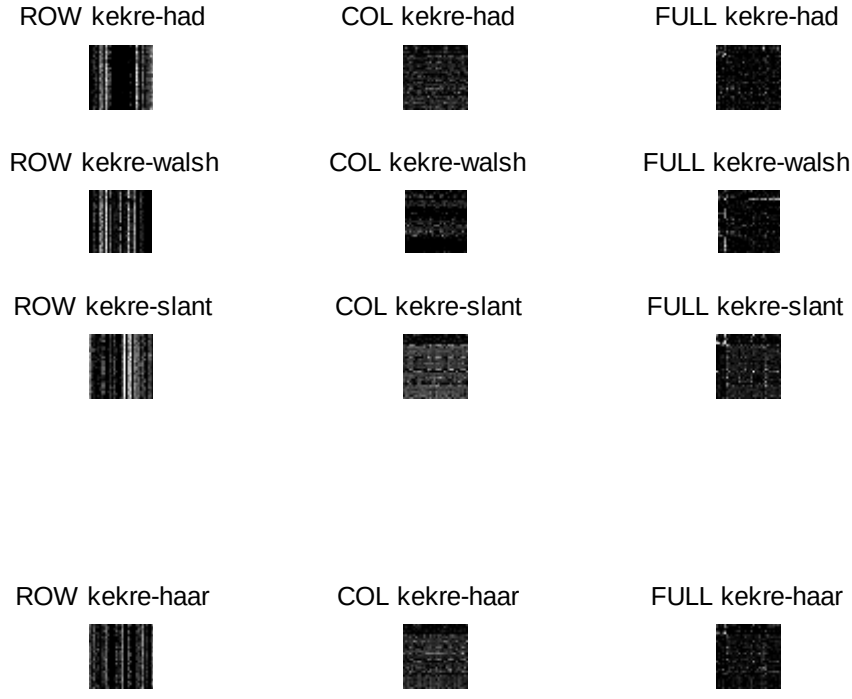


Fig 5.33: Output for Baboon Hybrid Transform

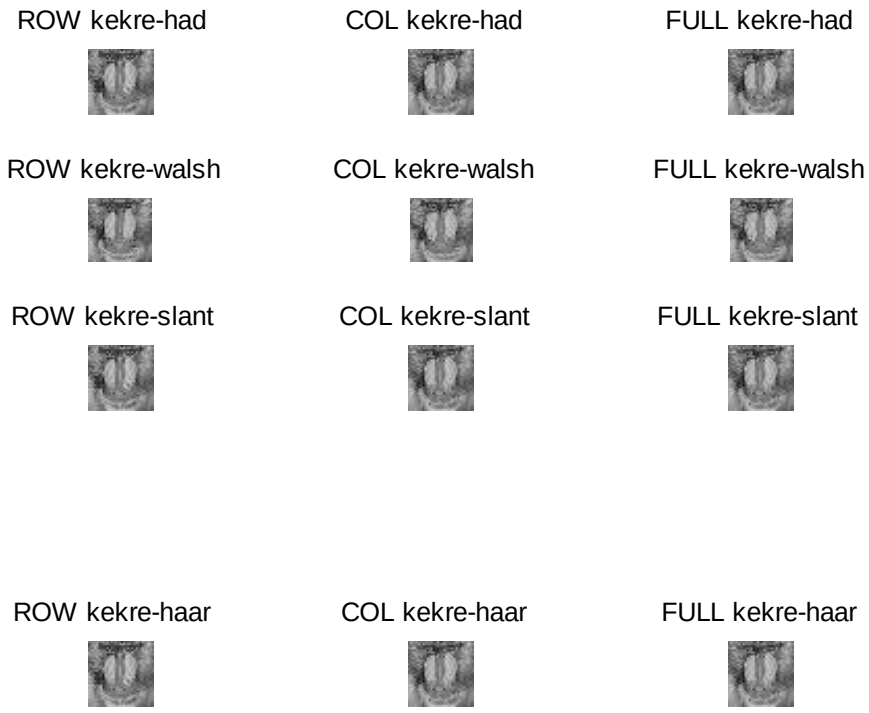


Fig 5.34: Output for Descrambled Baboon Hybrid Transform

5.6.1.3 Results for Cartoon

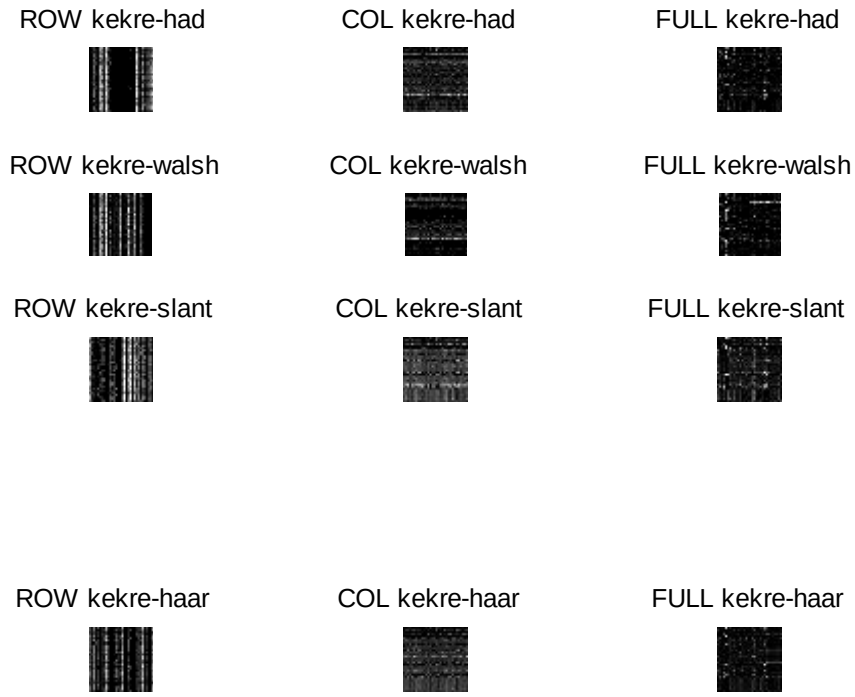


Fig 5.35: Output for Cartoon Hybrid Transform

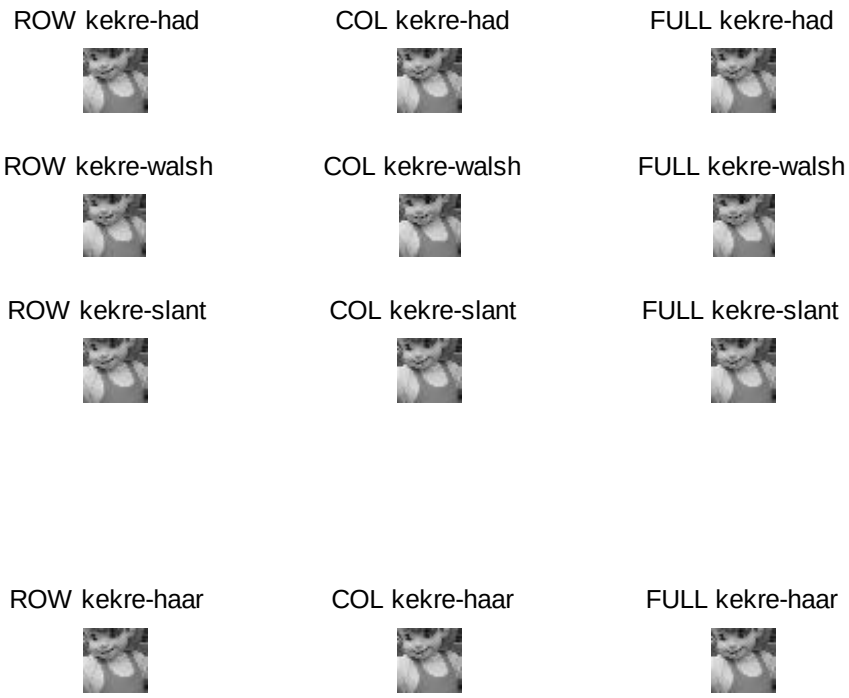


Fig 5.36: Output for Descrambled Cartoon Hybrid Transform

5.6.1.4 Results for Pepper

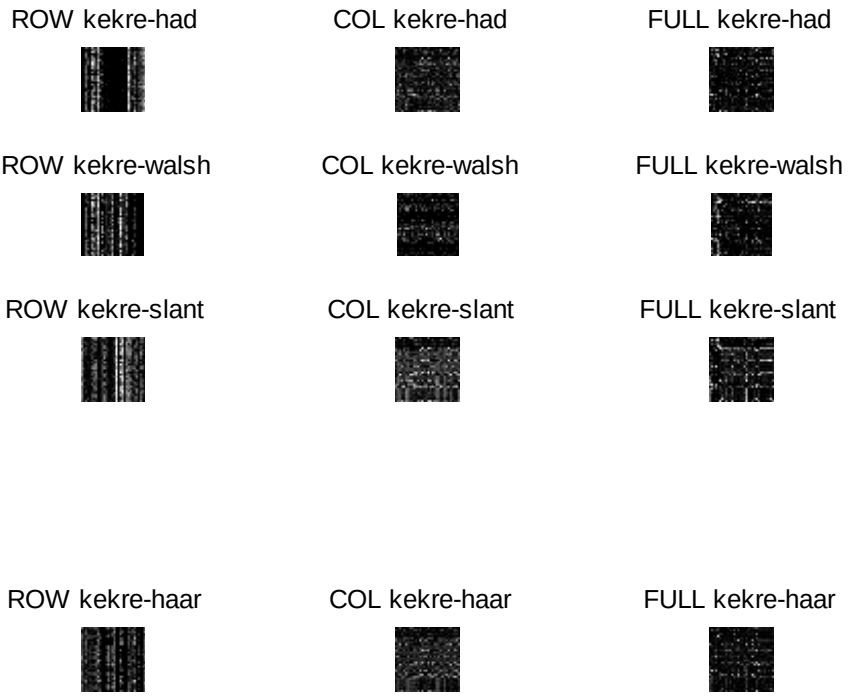


Fig 5.37: Output for Pepper Hybrid Transform

Fig 5.38: Output for Descrambled Pepper Hybrid Transform

5.6.1.5 Results for Lotus

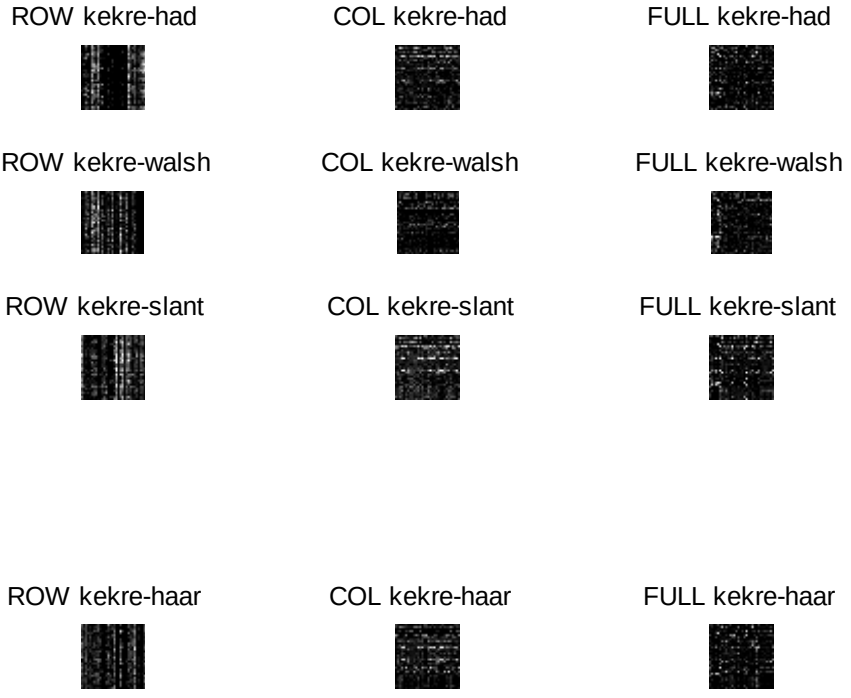


Fig 5.39: Output for Lotus Hybrid Transform

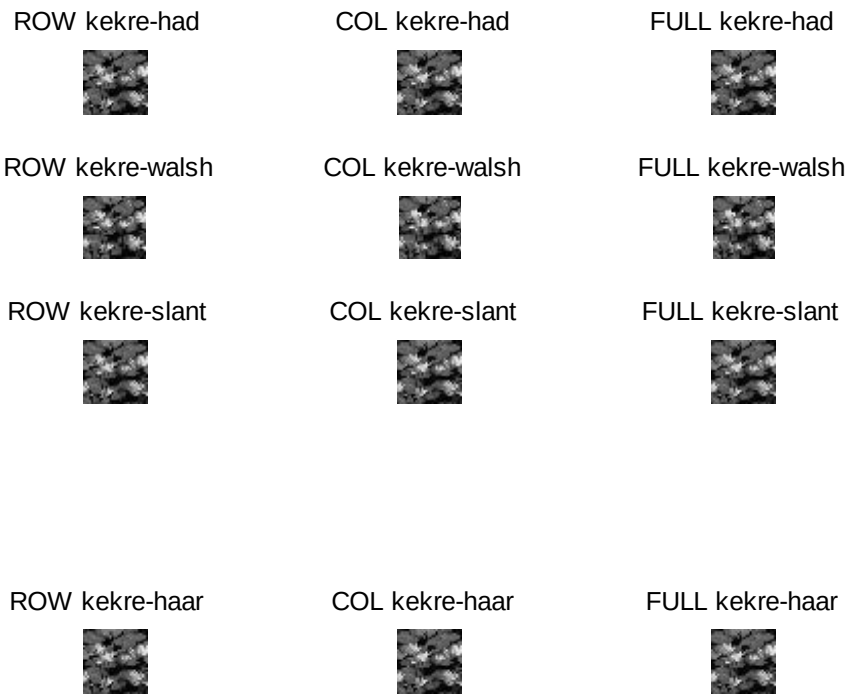


Fig 5.40: Output for Descrambled Lotus Hybrid Transform

5.6.2 Experimental Results for Hybrid Transform Base Kekre (Non Sinusoidal)

Experimental Results for 8x32 Base Kekre (Local Transforms WALSH , SLANT and HAAR)

	Row Transform	**ROW TRANSFORM SCRAMBLED**	Col Transform	**Col TRANSFORM SCRAMBLED**	Full Transform	**FULL TRANSFORM SCRAMBLED**
Pepper 7.5707	Kekre – Walsh					
	4.7278	3.5401	4.5080	3.4935	4.6044	3.0340
	Kekre – Slant					
	5.0187	4.7004	4.2333	3.1383	4.0830	2.7229
	Kekre – Haar					
	4.8146	4.5618	4.3044	3.2175	4.1156	2.9691

	Row Transform	**ROW TRANSFORM SCRAMBLED**	Col Transform	**Col TRANSFORM SCRAMBLED**	Full Transform	**FULL TRANSFORM SCRAMBLED**

	Kekre – Walsh

	Row Transform	**ROW TRANSFORM SCRAMBLED**	Col Transform	**Col TRANSFORM SCRAMBLED**	Full Transform	**FULL TRANSFORM SCRAMBLED**
	4.7912	3.0355	4.3580	3.2929	4.5432	2.6479
	Kekre – Slant					
	5.0493	4.6869	4.0754	2.7209	3.8216	2.3409
	Kekre – Haar					
	4.9454	4.5441	4.2728	3.1947	3.8131	2.5413

	Row Transform	**ROW TRANSFORM SCRAMBLED**	Col Transform	**Col TRANSFORM SCRAMBLED**	Full Transform	**FULL TRANSFORM SCRAMBLED**
Baboon 7.1791	Kekre – Walsh					
	4.7336	3.5350	4.2841	3.5750	4.5075	3.2289
	Kekre – Slant					
	5.0583	4.6861	4.0529	3.5730	3.8593	3.1266
	Kekre – Haar					
	4.9152	4.5161	4.1685	3.6960	3.8859	3.2515

	Row Transform	**ROW TRANSFORM SCRAMBLED**	Col Transform	**Col TRANSFORM SCRAMBLED**	Full Transform	**FULL TRANSFORM SCRAMBLED**
Lotus 7.3682	Kekre – Walsh					
	4.7655	3.6566	4.6469	3.8028	4.5156	**3.2481**
	Kekre – Slant					
	4.9039	4.7692	4.9039	3.4996	4.1500	2.9797
	Kekre – Haar					
	4.6972	4.6315	4.3361	3.7218	4.1542	3.2221

Lena 7.4522	Kekre – Walsh					
	4.8065	3.4981	4.4550	3.2979	4.5562	3.0438
	Kekre – Slant					
	5.1058	4.3758	4.1176	3.0106	4.0096	2.8197
	Kekre – Haar					
	4.8139	4.3787	4.1665	3.1728	4.0322	3.0931

5.6.3 Image Results for Sinusoidal Hybrid Transforms with Base Hartley

5.6.3.1 Results for Lena

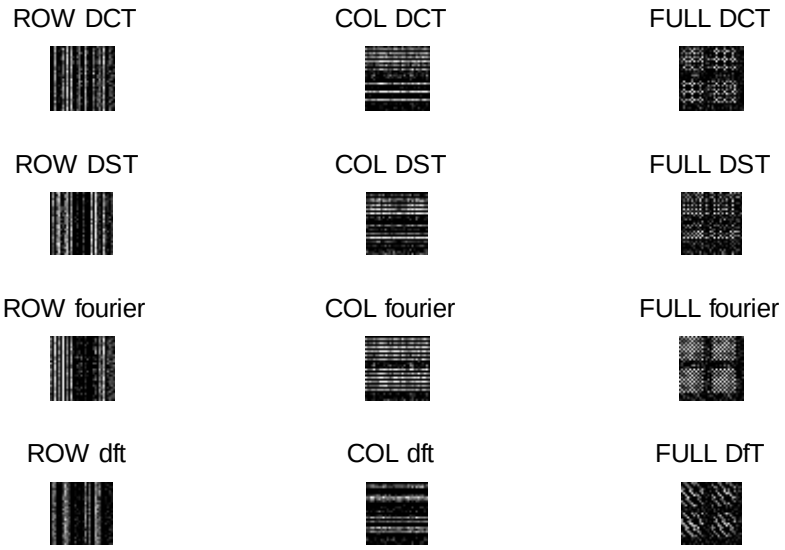


Fig 5.41: Output for Lena Hybrid Transform

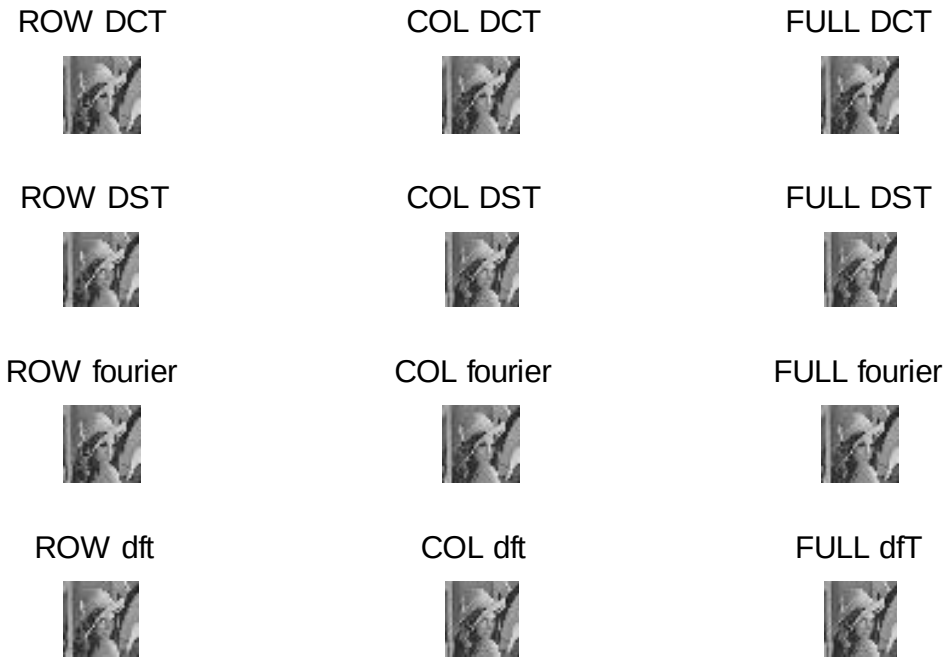


Fig 5.42: Output for Descrambled Lena Hybrid Transform

5.6.3.2 Results for Cartoon

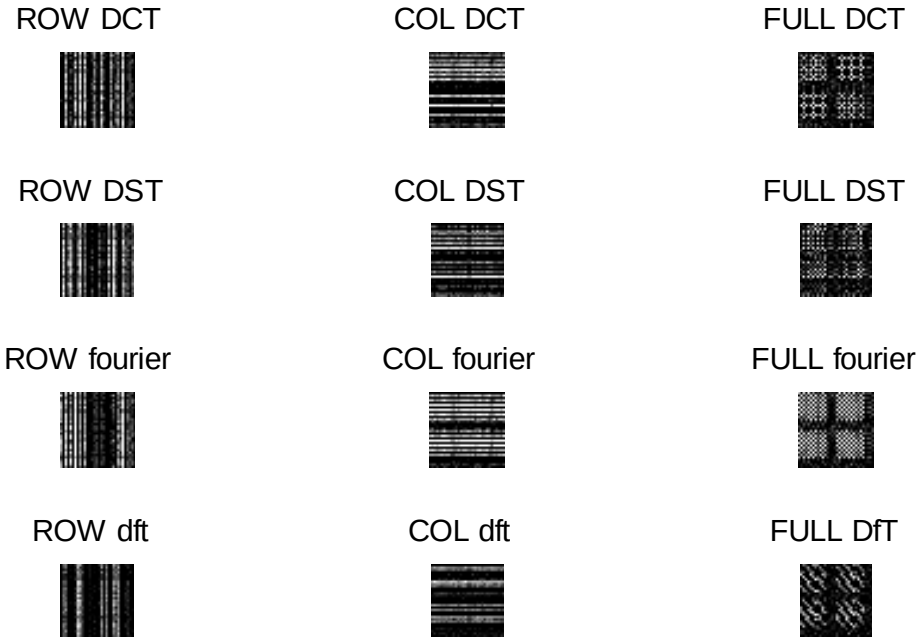


Fig 5.43: Output for Cartoon Hybrid Transform

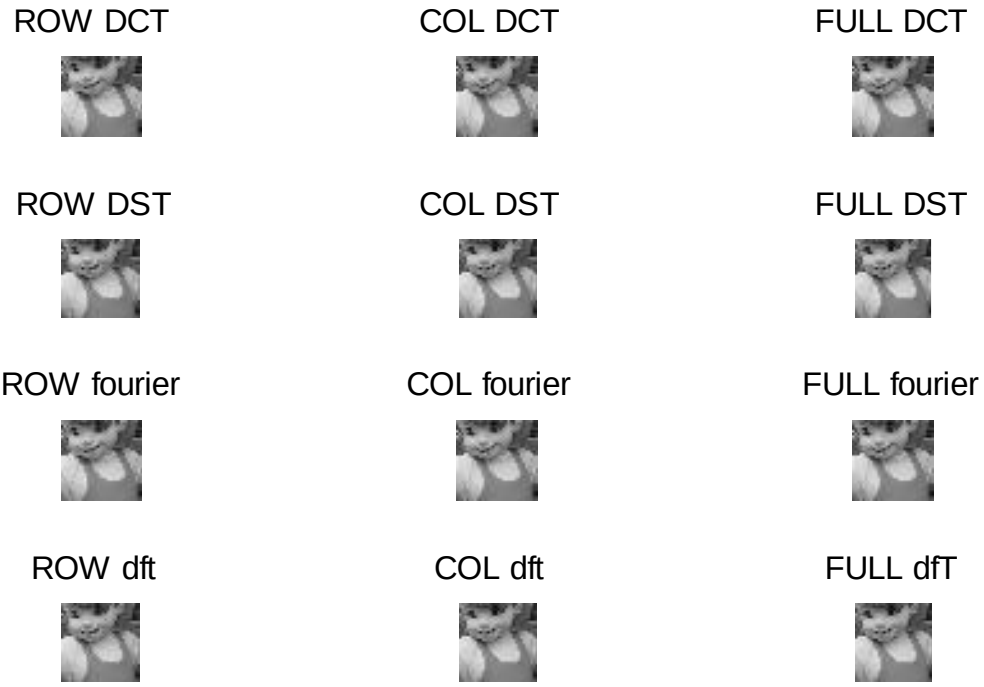


Fig 5.44: Output for Descrambled Cartoon Hybrid Transform4

5.6.3.1 Results for Baboon

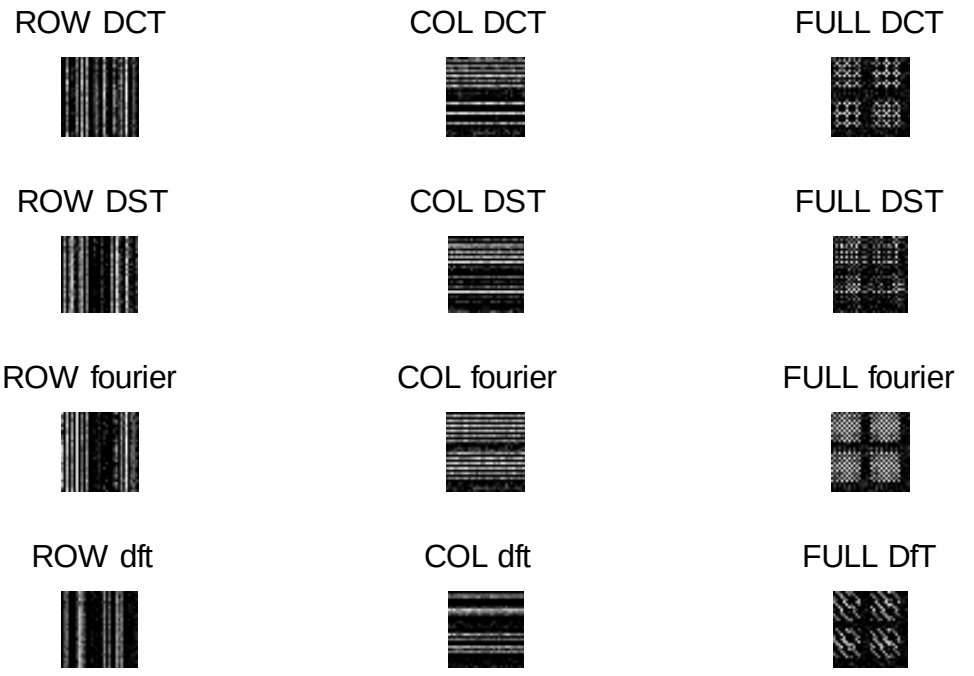


Fig 5.45: Output for Baboon Hybrid Transform

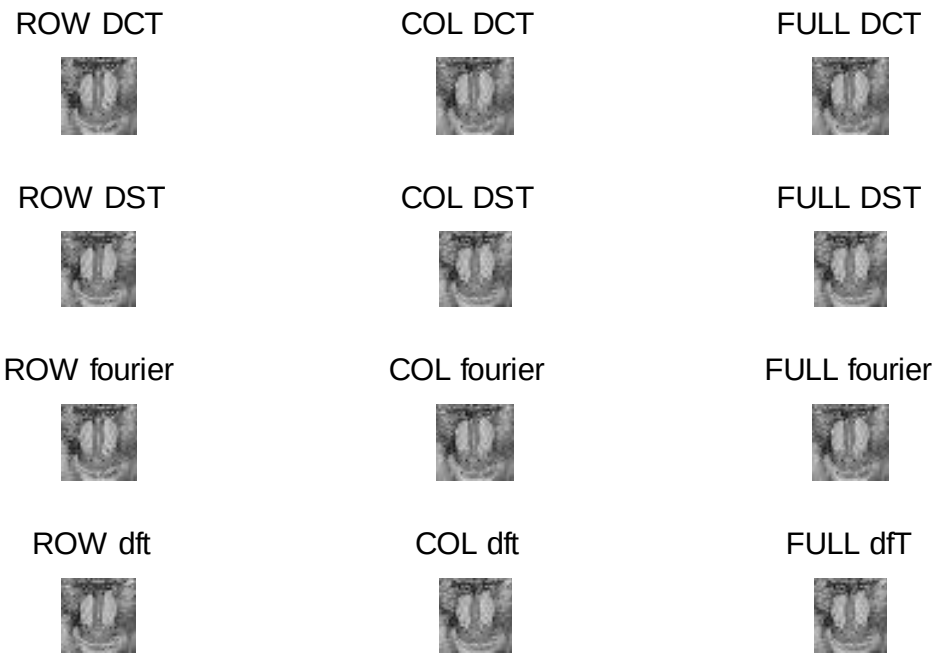


Fig 5.46: Output for Descrambled Baboon Hybrid Transform

5.6.3.1 Results for Lotus

Fig 5.47: Output for Lotus Hybrid Transform

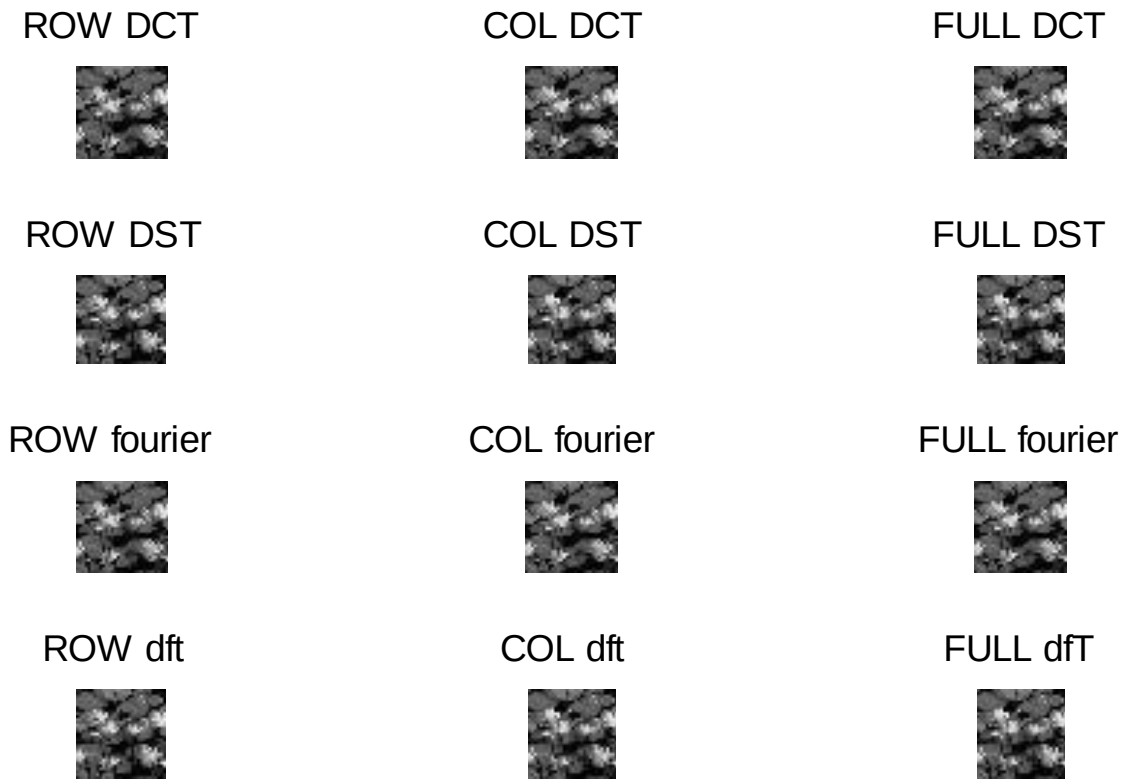


Fig 5.48: Output for Descrambled Lotus Hybrid Transform

5.6.4 Experimental Results for Sinusoidal Hybrid Transforms with Base Hartley

Lena	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
(Row)	0.9938	0.4954	0.194	0.1894	0.2078	0.2086
(Col)	0.2059	0.1998	0.9929	0.3085	0.2236	0.2191
Hartley - DST						
(Row)	0.9933	0.4849	0.1917	0.1933	0.2899	0.2134
(Col)	0.2076	0.2103	0.9916	0.4761	0.2792	0.2219
Hartley - Real Fourier						

(Row)	0.9932	0.491	0.1896	0.1924	0.212	0.1999
(Col)	0.2209	0.207	0.9927	0.2187	0.2318	0.206
Hartley - DFT						
(Row)	0.9946	0.2297	0.262	0.1161	0.5488	0.1389
(Col)	0.2409	0.124	0.9929	0.1905	0.5211	0.1524

Correlation table for Lena

cartoon	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
(Row)	0.995	0.5721	0.1876	0.1857	0.2069	0.2283
(Col)	0.18	0.2263	0.9954	0.3523	0.2182	0.2318
Hartley - DST						
(Row)	0.994	0.5993	0.196	0.1891	0.37	0.2215
(Col)	0.1931	0.2003	0.9947	0.5664	0.3576	0.2398
Hartley - Real Fourier						
(Row)	0.9948	0.5516	0.1922	0.189	0.2312	0.1976
(Col)	0.1976	0.2207	0.9952	0.2979	0.2285	0.2152
Hartley - DFT						
(Row)	0.9953	0.2544	0.1836	0.1255	0.5849	0.1526
(Col)	0.2656	0.1067	0.9957	0.297	0.5492	0.1421

Correlation table for Cartoon

Baboon	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
(Row)	0.9919	0.4963	0.2066	0.2007	0.2225	0.1981

(Col)	0.2071	0.2014	0.9934	0.2703	0.2097	0.2033
Hartley - DST						
(Row)	0.9916	0.5317	0.2044	0.2077	0.2858	0.216
(Col)	0.2	0.2051	0.9922	0.5093	0.277	0.2211
Hartley - Real Fourier						
(Row)	0.9924	0.4756	0.1879	0.2057	0.2237	0.1901
(Col)	0.2037	0.2008	0.9935	0.2144	0.2159	0.1904
Hartley - DFT						
(Row)	0.9939	0.1876	0.2558	0.121	0.4112	0.1314
(Col)	0.243	0.1168	0.9944	0.1767	0.4068	0.1294

Correlation table for Baboon

Lotus	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
(Row)	0.9814	0.346	0.203	0.195	0.2316	0.21
(Col)	0.1898	0.1902	0.9899	0.2792	0.2275	0.2036
Hartley - DST						
(Row)	0.9803	0.3537	0.2112	0.2017	0.2848	0.2034
(Col)	0.19	0.1876	0.9861	0.348	0.2778	0.1975
Hartley - Real Fourier						
(Row)	0.9842	0.3473	0.2041	0.1966	0.2399	0.2015
(Col)	0.192	0.196	0.9886	0.2358	0.2419	0.1979
Hartley - DFT						
(Row)	0.9876	0.1928	0.2783	0.1321	0.4622	0.1456
(Col)	0.2615	0.1168	0.9902	0.2047	0.5112	0.1386

Correlation table for Lotus

		ROW TRANSFORM SCRAMBLED	Col TRANSFORM SCRAMBLED	FULL TRANSFORM SCRAMBLED
Lena	Hartley - DCT	0.5839	0.5878	0.5924
	Hartley - DST	0.5845	0.5872	0.59
	Hartley - Real Fourier	0.6007	0.5864	0.5908
	Hartley - DFT	0.6528	0.6514	0.6531
Pepper	Hartley - DCT	0.5694	0.5742	0.5762
	Hartley - DST	0.5657	0.5705	0.5723
	Hartley - Real Fourier	0.5823	0.5743	0.5751
	Hartley - DFT	0.6351	0.6356	0.636
Cartoon	Hartley - DCT	0.6508	0.646	0.6575
	Hartley - DST	0.6419	0.6498	0.6546
	Hartley - Real Fourier	0.6613	0.6445	0.6548
	Hartley - DFT	0.7233	0.7223	0.724
Baboon	Hartley - DCT	0.5978	0.5946	0.604
	Hartley - DST	0.5866	0.5981	0.6025
	Hartley - Real Fourier	0.6125	0.5939	0.6028
	Hartley - DFT	0.664	0.6635	0.6641

Lotus	Hartley - DCT	0.4381	0.4464	0.4472
	Hartley - DST	0.4365	0.4426	0.4425
	Hartley - Real Fourier	0.4446	0.4456	0.4457
	Hartley - DFT	0.4878	0.4905	0.491

Table 5.7.1: PAFCPV Values for all the Images

		ROW TRANSFORM SCRAMBLED	Col TRANSFORM SCRAMBLED	FULL TRANSFORM SCRAMBLED
Lena	Hartley - DCT	100	100	100
	Hartley - DST	100	100	100
	Hartley - Real Fourier	100	100	100
	Hartley - DFT	100	100	100
Pepper	Hartley - DCT	100	100	100
	Hartley - DST	100	100	100
	Hartley - Real Fourier	100	100	100
	Hartley - DFT	100	100	100
Cartoon	Hartley - DCT	100	100	100
	Hartley - DST	100	100	100

	Hartley - Real Fourier	100	100	100
	Hartley - DFT	100	100	100
Baboon	Hartley - DCT	100	100	100
	Hartley - DST	100	100	100
	Hartley - Real Fourier	100	100	100
	Hartley - DFT	100	100	100
Lotus	Hartley - DCT	100	100	100
	Hartley - DST	100	100	100
	Hartley - Real Fourier	100	100	100
	Hartley - DFT	100	100	100

Table 5.7.2: NPCR Results for all image

Lena: 7.4522	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
	3.6687	4.8155	2.7735	4.9391	2.2495	4.8553
Hartley - DST						
	3.9788	4.9085	4.4094	4.7641	3.0155	4.7627
Hartley - Real Fourier						
	3.9682	4.8254	3.1512	4.9323	2.5834	4.8781
Hartley - DFT						
	3.8816	4.6859	4.365	4.7003	4.3824	4.7085

Table 5.7.3: Entropy for Lena

Pepper: 7.5707	ROW TRANSFORM	ROW TRANSFORM SCRAMBLED	Col TRANSFORM	Col TRANSFORM SCRAMBLED	FULL TRANSFORM	FULL TRANSFORM SCRAMBLED
Hartley - DCT						
	3.6511	4.877	3.0833	4.9249	2.1885	4.869
Hartley - DST						
	3.9534	4.9545	4.6123	4.8242	2.6934	4.7684
Hartley - Real Fourier						
	3.8073	4.7539	3.5327	4.9102	2.3617	4.8825
Hartley - DFT						
	3.1333	4.6812	4.6702	4.6998	4.0406	4.701

Table 5.7.4: Entropy for Pepper

cartoon: 7.4201	ROW TRANSFORM	ROW TRANSFORM SCRAMBLED	Col TRANSFORM	Col TRANSFORM SCRAMBLED	FULL TRANSFORM	FULL TRANSFORM SCRAMBLED
Hartley - DCT						
	3.6358	4.7185	2.7647	4.9155	2.0845	4.8352
Hartley - DST						
	4.1008	4.8448	3.3785	4.7466	2.5344	4.7362
Hartley - Real Fourier						
	3.7137	4.7546	2.7369	4.9355	2.3629	4.8456
Hartley - DFT						
	3.8261	4.7711	3.6028	4.7662	4.2663	4.7645

Table 5.7.5: Entropy for Cartoon

Baboon: 7.1791	ROW TRANSFORM	ROW TRANSFORM SCRAMBLED	Col TRANSFORM	Col TRANSFORM SCRAMBLED	FULL TRANSFORM	FULL TRANSFORM SCRAMBLED
Hartley - DCT						
	3.925	4.7915	3.2489	4.9099	2.6391	4.8901
Hartley - DST						
	4.3993	4.9721	4.2059	4.7524	2.9379	4.7686
Hartley - Real Fourier						
	4.0224	4.7857	3.3324	4.9145	2.7406	4.8999

Hartley - DFT						
	4.855	4.6589	4.379	4.6743	3.9084	4.6788

Table 5.7.6: Entropy for Baboon

Lotus : 7.3682	**ROW TRANSFORM**	**ROW TRANSFORM SCRAMBLED**	**Col TRANSFORM**	**Col TRANSFORM SCRAMBLED**	**FULL TRANSFORM**	**FULL TRANSFORM SCRAMBLED**
Hartley - DCT						
	3.7343	4.7871	3.1415	4.8154	2.3728	4.8055
Hartley - DST						
	4.0392	4.8424	4.1858	4.7993	2.7835	4.7329
Hartley - Real Fourier						
	3.956	4.7028	3.4084	4.7917	2.6586	4.7904
Hartley - DFT						
	4.1417	4.5474	4.2649	4.553	4.138	4.5554

Table 5.7.7: Entropy for Lotus

CHAPTER 6

CONCLUSION

Based on the experimental results obtained(shown in the previous chapter), our framework for transform along with image scrambling technique gives better results as compared to the traditional techniques.

First, various kinds of image scrambling techniques were introduced, followed by a description of different kinds of transforms (sinusoidal and non-sinusoidal). We have set a basic framework for transform and image scrambling/encryption techniques, in which both sinusoidal transform and non-sinusoidal transform have been utilized.

For non-sinusoidal transforms, Kekre transforms provides the best results out of the four transforms. For sinusoidal transforms, DHT and Fourier provide the best results. Energy distribution were found to be similar in all the four non-sinusoidal transforms with original image, row, column transformed image and row scrambled image having a linear increase, whereas energy was found to be in the range of 80% in full transformed image in the initial blocks and then a small increase till the last block of the image.

In the full transformed scrambled image, energy was very less approx. to 0 and then a sudden increase in the energy after 22^{nd} block of the image. This energy distribution in the scrambled image shows a common property for all the full transforms which can be very useful to check whether the image is scrambled.

For hybrid transforms, we have combined two transforms-sinusoidal and non-sinusoidal transforms which provide even better results as compared to the previously discussed one transform. Thus, the proposed approach is a combination of both transform as well as spatial domain. Hence it is very useful for image scrambling and provides more security.

REFERENCES

[1] Zhenwei Shang HongeRenJian Zhang, “A Block Location Scrambling Algorithm of Digital Image Based on Arnold Transformation”, The 9th International Conference for Young Computer Scientists.

[2] YicongZhousa, SosAgaianb, Valencia M. Joynera, Karen Panettaa, “Two Fibonacci P-code Based Image Scrambling Algorithms”, The International Society for Optical Engineering.

[3] PrashanPremaratne, “Key-based scrambling for secure image communication”, University of Wollongong, University of Wollongong Research Online

[4] Yicong Zhou, Karen Panetta, SosAgaian, “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] SapnaAnoop, AnoopAlakkaran, “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 Grosges, 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] FuhaoZou, 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, DashunQue, 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.

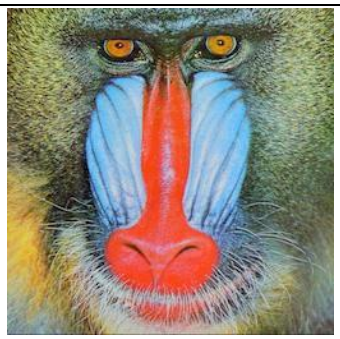
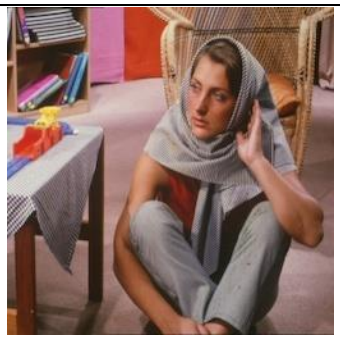
[18] G. T. Buracas, and G. M. Boynton, "Efficient Design of Event-Related fMRI Experiments Using M-sequences," NeuroImage, vol. 16, pp. 801-813, 2002.

[19] Yicong Zhou, Karen Panetta and SosAgaian, "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 1	
Cartoon 2	
Flower	
Fruits	
Lena	

Pepper	
Puppy	
Scenery	
Smiley	
TajMahal	

Temple	
Watch	
Lotus