Image Encryption using MVK Algorithm, El-Gamal and Chaotic Systems

Mrs. T Madhavi Kumari [1], Dr. A. Vinaya Babu[2], K S Goutham[3]

[1](ECE, JNTUH College of Engineering Hyderabad, India)
[2](Rtd. Prof. Of CSE, JNTU & Dean Academics, Stanley College of Engg. Tech, Hyderabad, India)
[3](ECE, JNTUH College of Engineering Hyderabad, India)

---***---

Abstract - Digital images are often transmitted in this modern era through various networks. Digital images are used in various fields like Medical, Military, etc. These images require high-security levels to prevent misusing and unauthorized access. There are several Image Encryption Techniques which are used to achieve image confidentiality; this paper presents a newer approach for Image Encryption which consists of 3 stages. MVK Algorithm is the first stage of encryption which is used to construct a permuted image and demolish the relationships among the adjacent pixels. El-Gamal Encryption Algorithm is the second stage of encryption which modifies permuted image; it is an alternative for RSA public key cipher. Third stage of encryption uses Lorenz system and Rossler system to achieve confusion and diffusion operations. To evaluate the effectiveness and security of the suggested strategy, a thorough analysis has been done. Visual and numerical results show that the proposed image cryptosystem can defend against a number of well-known attacks.

Key Words: El-Gamal, Lorenz, Rossler.

1.INTRODUCTION

Digital images are mostly used to convey information via internet. Ensuring security to these images is a main concern. There is always a possibility of threat from an unauthorized person who tries to steal or modify its contents. Cryptographic encryption/decryption is the best and most widely used technique to prevent such attacks. Cryptography uses either symmetric encryption or asymmetric encryption for encrypting data.

In symmetric encryption, sender and receiver exchange keys with each other over a secure channel to maintain confidentiality. Using this key, both sender and receiver encrypt and decrypt data exchanged between them. Symmetric encryption is categorized into Block algorithms and Stream algorithms. In block algorithms, the data is divided into small blocks and each block is encrypted using a designated key whereas in stream algorithms data is encrypted as it streams. AES, DES, IDEA, Blowfish and RC4 are some of the popular and most widely used symmetric encryption algorithms. Asymmetric encryption uses two keys: public key is used for encryption and private key is used for decryption. It is more complex, time-consuming and requires more computational power than symmetric encryption. RSA, ECC and El-Gamal are some of the popular asymmetric encryption algorithms. Its main applications are confidentiality of data, Non-repudiation, Integrity of information exchange.

Recently, chaotic systems are heavily used to develop cryptographic algorithms. Chaotic systems provide efficiency, security, speed and strong defenses against different kind of attacks. Adopting chaotic systems with high dimensionality will improve security by increasing the nonlinearity. This is the reason why chaotic systems are widely used in image security.

2. BACKGROUND

In this paper, we present a newer encryption system to secure images by integrating MVK Algorithm, El-Gamal and chaotic systems. This section elaborates each component present in our approach.

MVK Algorithm: MVK algorithm is a newer algorithm proposed by one of our authors Mrs. T Madhavi Kumari in the year 2021. Predicting a pixel value from its neighboring pixel values will be easier if we have tight relationships among the adjacent pixels. This algorithm is employed to reduce the tight relations among pixels and improve the entropy value of the image. According to MVK Algorithm, we divide an image into a certain number of blocks and later we shuffle these block positions to break neighboring relations. Before applying the MVK algorithm to any image we need to make sure that length and breadth of input image must be equal and also should be multiple of 25. This is the only constraint we need to follow before performing this algorithm. Consider an input matrix as shown in Fig 2.1, consider each element in the matrix as a block containing pixels of an image. To get output, first create an empty matrix with the same shape as the input matrix. Below are the rules we need to follow to shuffle these blocks to get an output:

a) Initially place the first element of the input matrix at the middle of the top row.
b) Consider it as the starting position and move towards the top right diagonally after placing the element.

Diagonal path is shown in Fig2.2 with black coloured dotted line.

c) While iterating, if we reach the top most row then move towards the immediate right column and start filling elements from bottom as shown in Fig2.2, path is shown with red coloured line.

d) While iterating, if we reach the rightmost column then place the next element immediately above the row in the left most column and continue the process of moving diagonally right. Path is shown in Fig2.2 with a blue coloured line.

e) If the diagonal element is already filled then move towards immediate down, path is shown in Fig2.2 with green coloured arrow.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 2.1 Input Matrix

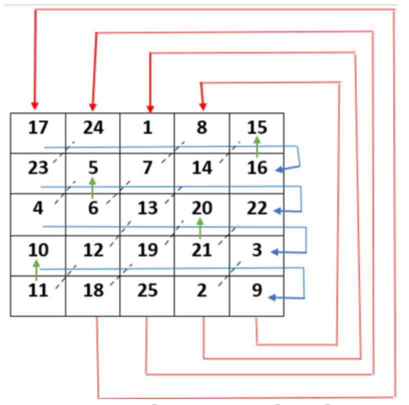


Figure 2.2 Encrypted matrix with Relocation Paths

Similarly, there are certain rules to get back the input matrix from the shuffled matrix. At first create an empty matrix of same shape as shuffled matrix and start filling positions in empty matrix according to these rules:

a) Consider the element from bottom row middle position and place it at the last row last column position of the reconstructed matrix.

b) Now consider that place of the shuffled matrix as the starting point and iterate diagonally bottom left.

c) While iterating, if we reach the bottom most row then move towards the immediate left column and start filling elements from top as shown in Fig2.3, path is shown with red coloured line.

d) While iterating, if we reach the left most column then place the next element immediately below the row in the right most column and continue the process of moving diagonally left. Path is shown in Fig2.3 with a blue coloured line.

e) If the diagonal element is already filled then move towards the immediate top path as shown in Fig2.3 with green coloured arrow.

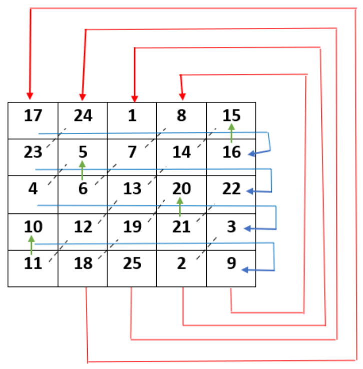


Figure 2.3 Matrix containing Decrypting Relocation Paths

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure2.4 Decrypted matrix

Fig2.3 shows all the relocation paths and Fig 2.4 represents the decrypted matrix same as the input matrix in encryption phase. This is the basic structure of the MVK Algorithm. Its usage will be robust in the implementation of the Encryption system to further break the strong relations between neighboring pixels. The degree of scrambling can be improved by using different beginning points, which also improves the algorithm's security.

El-Gamal Encryption: El-Gamal encryption was first proposed in 1985 by Taher El-Gamal. It is an asymmetric key cryptosystem. El-Gamal supports homomorphic multiplication operations on encrypted data. Security of El-Gamal is based on the hardness of solving discrete algorithms. It is almost unattainable to break this encryption system. For a given plain image, El-Gamal encryption provides different encrypted images every time we encrypt. This encryption process consists of 3 phases:

a) Key Production Phase:

1. Select a prime number ‘p’ and an integer ‘e_1’ such that e_1 is the primitive root of p.
2. Select a secret key ‘d’ which is a random integer from the interval [1, …., p-2].
3. ‘e_2’ is calculated as: $e_2 = e_1^d \bmod p$

Private key is ‘d’ and the public key contains the triple (p, e_1, e_2).

b) Encryption Phase:

1. Public keys are obtained from the sender.
2. Select a random integer ‘r’ from the interval [1, …., p-2].
3. Encrypted messages contain a pair (c_1, c_2). ‘m’ represents a message to be encrypted.

4. c_1 is calculated as: $c_1 = e_1^{\,r} \bmod p$, c_2 is calculated as: $c_2 = m \times (e_2^{\,r} \bmod p)$.

c) Decryption Phase: Decrypt m by following computation: $m = c_2/(c_1)^d \bmod p$.

Chaotic Systems: Chaotic systems are highly sensitive to its initial and control parameters. These qualities of chaotic systems are helpful for improving the security of images. In our methodology, we use the Lorenz system and Rossler system. Below is the description of these two chaotic systems:

a) Lorenz System: This system was established in the year 1963 by scientist Lorenz. It is described via three-dimensional independent equations which are given below:

$$\dot{x} = a(y - x),\ \dot{y} = cx - y - xz,\ \dot{z} = xy - bz \qquad (1)$$

Here a, b, c are control parameters. x_0, y_0, z_0 are initial states. In order to attain chaotic behavior, values of control parameters (a, b, c) should be (10, 8/3, 28) respectively.

b) Rossler System: This system was established in the year 1970 by Rossler. It is described via three-dimensional independent equations which are given below:

$$\dot{x} = -y - z,\ \dot{y} = x + ay,\ \dot{z} = b + z(x - c) \qquad (2)$$

Here a, b, c are control parameters. x_0, y_0, z_0 are initial states. In order to attain chaotic behavior, values of control parameters (a, b, c) should be (0.2, 0.2, 5.7) respectively.

3.METHODOLOGY

In this section, we elaborate on our proposed methodology for securing digital images. Our methodology is composed of MVK Algorithm, El-Gamal Encryption and Chaotic Systems. Before the encryption of image make sure that length and breadth of image are equal and also should be multiple of 25. In the first stage of encryption, we apply the MVK Algorithm on the entire image twice. Next we split the image vertically into left half and right half, then we create a new image by merging alternative columns of left half and right half, and finally flip the image upside down. After that we apply the MVK Algorithm on each sub block of the image twice. This is about the first stage of encryption using MVK Algorithm. In the second stage, we use El-Gamal encryption, which is applied to produce a secure digital image. Chaotic systems are used to carry out primitive cryptographic operations in order to enhance system security and increase entropy (by reducing correlation among image pixels) . In order to achieve the confusion operation by randomly varying the pixel locations, a three-dimensional Lorenz chaotic system is applied on the encrypted image obtained from the second phase. The diffusion operation is then carried out using a three-dimensional Rössler system by altering the image pixel values, leading to the creation of the ciphered image. Block diagram of the proposed methodology is shown in Fig 5.

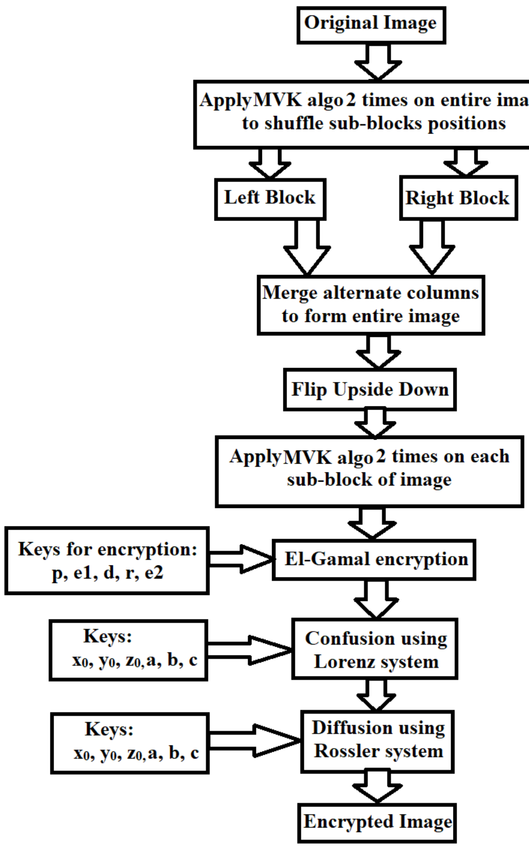

Original Image
ApplyMVK algo 2 times on entire image to shuffle sub-blocks positions
Left Block
Right Block
Merge alternate columns to form entire image
Flip Upside Down
ApplyMVK algo 2 times on each sub-block of image
Keys for encryption: p, e1, d, r, e2
El-Gamal encryption
Keys: x0, y0, z0, a, b, c
Confusion using Lorenz system
Keys: x0, y0, z0, a, b, c
Diffusion using Rossler system
Encrypted Image

Figure 3 Block Diagram of proposed methodology

Encryption Procedure of Color Image:

Step1: The original image O (n, n × 3) is divided into (n, n), G (n, n) and B (n, n) respectively.

Step2: Divide R (n, n) into 25 sub-blocks of equal size. Consider each sub-block as a unit and apply New Algo on R (n, n) for 2 times such that all these units are shuffled to get D (n, n).

Step3: Split D (n, n) vertically into 2 equal blocks to get D_1(n, n/2) and D_2(n, n/2).

Step4: To get M (n, n), merge D_1(n, n/2) and D_2(n, n/2) in such a way that columns of D_1 represent the odd columns in M (1,3,5…) and the columns of D_2 represent the even columns in M (2,4, 6…).

Step5: Flip the M (n, n) upside down to get U (n, n).

Step6: Now apply the MVK Algorithm on each of the 25 sub blocks of U(n, n) twice to get F(n, n).

Step7: El-Gamal encryption is performed on F (n, n) with the help of keys p, e_1, d, r, e_2 to get:

$$E_1 = e_1^{\,r} \bmod p,$$
$$E_2(i, j) = F(i, j) \times (e_2^{\,r} \bmod p).$$

Step6: Generate x, y, z chaotic sequences from (1) using x_0, y_0, z_0, a, b, c. Reshape x with the same size of E_2 to get X. Pixels of E_2 are scrambled as follows:

$$[I_x, V_x] = \text{sort}(X),$$
$$L_1(i, j) = E_2(I_x(i), I_x(j),) \qquad (3)$$

X is sorted in ascending order, V_x represents the new sequence after sorting X and I_x represent index value of V_x. Finally, L_1 (n, n) is obtained by permuting E_2 according to (3).

Step7: Generate x', y', z' chaotic sequences from (2) using x_0, y_0, z_0, a, b, c. Reshape x' with the same size of L_1 to get X'. Perform the following bitwise operation to diffuse pixels of L_1.

$R_1 = bitxor\,(R_1\,(i, j - 1), L_1\,(i, j), X'\,(i, j))$ (4)

Step8: Perform step2 to step7 on G (n, n) and B (n, n) to get R_2 (n, n) and R_3 (n, n) respectively. Combine all the 3 channels output to form R (n, n) to get the final encrypted image.

Encryption Procedure of Color Image:

For the Decryption process, we employ encryption steps in reverse order to recover the original image. This process requires all the secret keys of El-Gamal, Lorenz and Rossler for decrypting the image.

4.RESULTS

In our analysis, handwritten scripts of Dravidian languages are used as sample test images as shown in Fig 4.1. These images are of 4 different resolutions. Fig4.1(a) is Malayali script with resolution 250 x 250, Fig4.1(b) is Kannada script with resolution 500 x 500, Fig4.1(c) is Telugu script with resolution 750 x 750, Fig4.1(d) is Tamil script with resolution 1000 x 1000. Due to space limitations, all the images are represented here with the same size. PyCharm Community Edition 2022.1.2 software is used to implement this Image Cryptosystem under Windows 12 operating system environment. In addition, a personal computer (laptop) with an Intel Core I5 and 8GB installed memory (RAM), is used to perform this experiment. Secret parameters for El-Gamal are randomly selected for each image. Secret parameters for Lorenz system are set as: $x_0 = 1.0$, $y_0 = 1.0$, $z_0 = 1.0$, $a = 10$, $b = 28$, and $c = 8/3$ whereas secret parameters for Rossler system are set as: $x_0 = 1.0$, $y_0 = 1.0$, $z_0 = 1.0$, $a = 0.2$, $b = 0.2$, and $c = 5.7$. The encrypted and decrypted versions of all images in Fig. 4.1 are shown in Figs. 4.2 and 4.3, respectively.

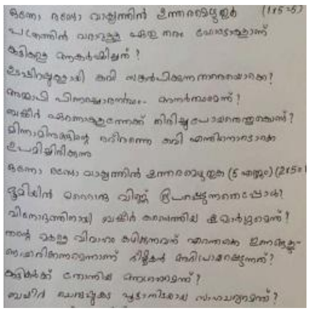


Figure 4.1(a)

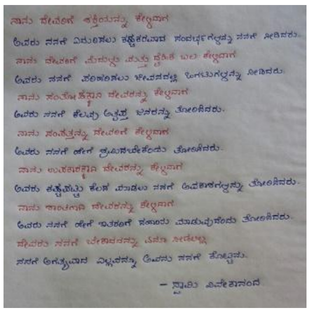


Figure 4.1(b)

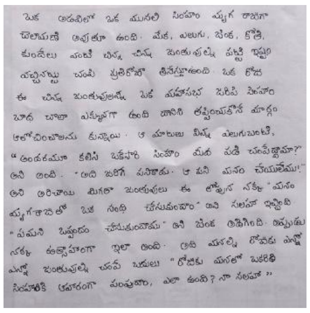


Figure 4.1(c)

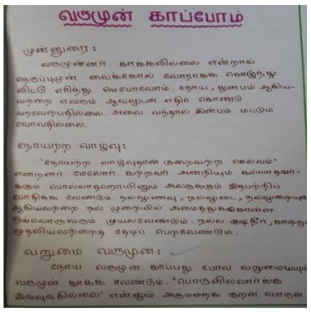


Figure 4.1(d)

Figure 4.1 Sample test images: (a) Malayali script, (b) Kannada script, (c) Telugu script and (d) Tamil script.

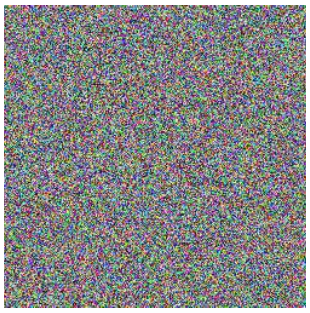


Figure 4.2(a)

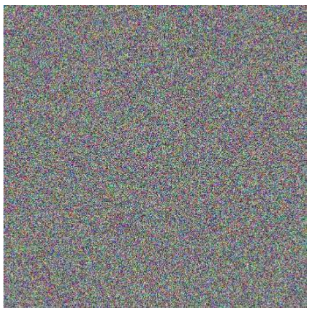


Figure 4.2(b)

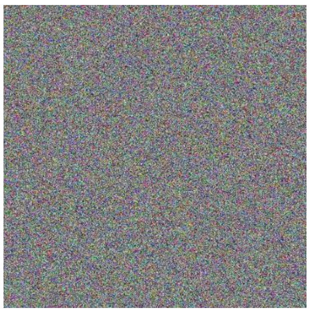


Figure 4.2(c)

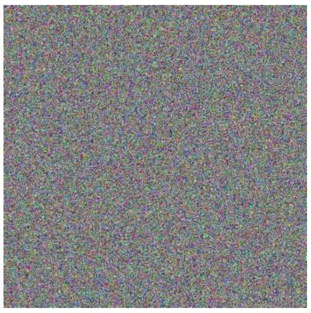


Figure 4.2(d)

Figure 4.2 Encrypted images: (a) Malayali script, (b) Kannada script, (c) Telugu script and (d) Tamil script.

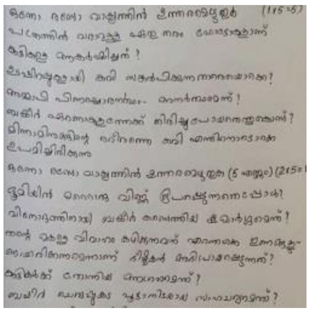


Figure 4.3(a)

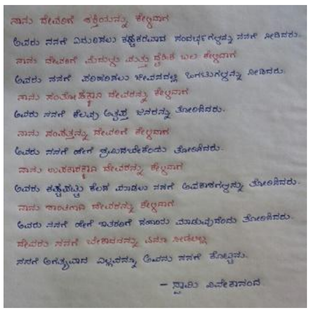


Figure 4.3(b)

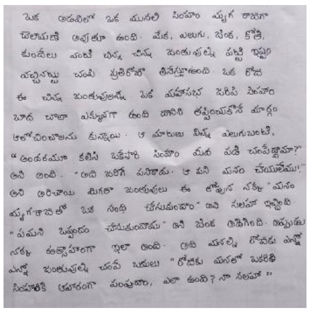
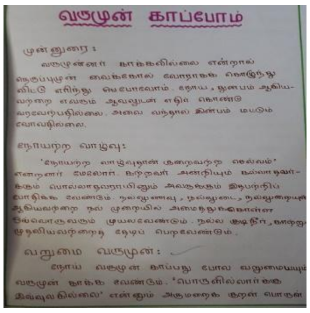

[illegible]

Figure 4.3(c)


வருமுன் காப்போம்

[illegible]

Figure 4.3(d)

Figure 4.3 Decrypted images: (a) Malayali script, (b) Kannada script, (c) Telugu script and (d) Tamil script.

Histogram Analysis: Histogram Analysis provides the visual representation of distribution of image pixels. For a good cipher technique, histogram should be flat and uniform. Fig. 4.4 shows histogram plot of plain images and Fig 4.5 shows histogram plot of encrypted images. In each plot, it contains 3 different curves representing each channel of image in red, green and blue color. As we can observe from the graphical results that all the encrypted images have a flatten histogram. Hence our proposed method can prevent any statistical attacks.

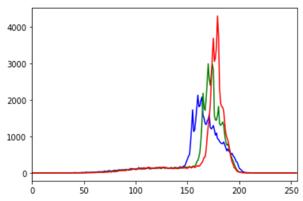


Figure 4.4(a)

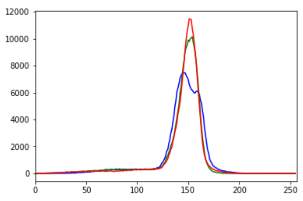


Figure 4.4(b)

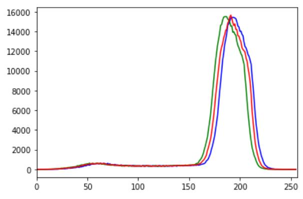


Figure 4.4(c)

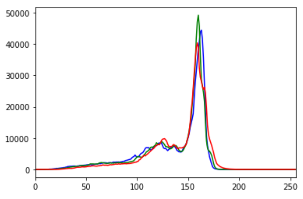


Figure 4.4(d)

Figure 4.4 Plain image histograms: (a) Malayali script, (b) Kannada script, (c) Telugu script and (d) Tamil script.

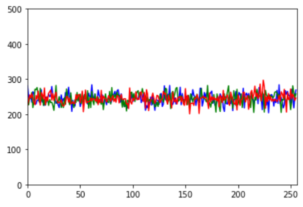


Figure 4.5(a)

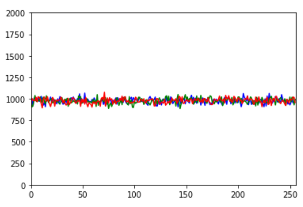


Figure 4.5(b)

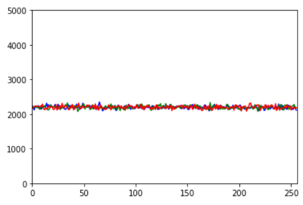


Figure 4.5(c)

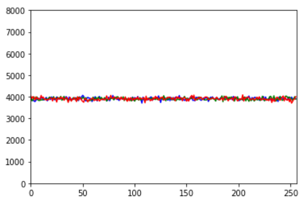


Figure 4.5(d)

Figure 4.5 Encrypted image histograms: (a) Malayali script, (b) Kannada script, (c) Telugu script and (d) Tamil script.

Image Entropy Analysis: This analysis helps to test randomness of plain and encrypted images. It is calculated as follows: H(X) = - $\sum_{i=0}^{n}$ $P(x_i)log_2P(x_i)$ where X is a random variable, which is defined as X = $\{x_0, x_1, \dots x_n\}$. P is probability of x_i and n denotes the size. If the entropy value is higher then it indicates that the encrypted image is highly secured. Theoretical value is 8. As per the results shown in Table 1, all the encrypted image entropy values are close to 8. Hence our proposed cryptosystem has high randomness.

Table no 1: shows entropy values of plain and encrypted images

Image	Plain Image Entropy	Encrypted Image Entropy
Malayali script	6.1156	7.9989
Kannada script	5.8710	7.9997
Telugu script	6.2091	7.9998
Tamil script	6.4203	7.9999

Peak Signal to Noise Ratio (PSNR): The peak signal to noise ratio is used for measuring quality of an image with the help of mean square error (MSE). Formula of MSE and PSNR are given below:

$$\text{MSE} = \frac{1}{N \times N}\sum_{i=1}^{N} \sum_{j=1}^{N} [X(i,j) - Y(i,j)]^2 \quad (3),$$

$$\text{PSNR} = 10 \times log_{10}[\frac{255 \times 255}{MSE}] \quad (4)$$

here X denotes plain image and Y denote encrypted image. If MSE value is higher and PSNR is lower then it indicates that there is great difference between input and output images. As per the results shown in Table 2 MSEs are large and PSNRs are small for every tested image. Hence our results prove that encrypted images obtained from our cryptosystem are highly distorted.

Table no 2: shows MSE and PSNR values between plain and encrypted images

Image	MSE between Plain and Encrypted images	PSNR between Plain and Encrypted images
Malayali script	7347.78	9.370
Kannada script	5143.08	10.186
Telugu script	9699.43	8.264
Tamil script	6561.55	9.961

Structural Similarity Index Measure (SSIM): SSIM is used for computing the visual difference between two given images. Here we compute SSIM between plain & encrypted and plain & decrypted. SSIM formula is given below:

$$\text{SSIM}\ (I_1, I_2) = \frac{(2\mu_{I_1}\mu_{I_2} + \alpha)(2\sigma_{I_1 I_2} + \beta)}{(\mu_{I_1}^2 + \mu_{I_2}^2 + \alpha)(\sigma_{I_1}^2 + \sigma_{I_2}^2 + \beta)} \qquad (5)$$

Here I_1 and I_2 are plain and encrypted images respectively. μ_{I_1} *and* μ_{I_2} are averages of I_1 and I_2 respectively. $\sigma_{I_1 I_2}$ is the covariance between I_1 and I_2. $\sigma_{I_1}^2$ *and* $\sigma_{I_2}^2$ are the variances of I_1 and I_2 respectively. SSIM value should be lower between plain and encrypted images. As per the results shown in Table 3, we can conclude that the proposed system can break the similarity between input and encrypted image.

Table no 3: shows SSIM between plain & encrypted, plain and decrypted

Image	SSIM between Plain and Encrypted images	SSIM between Plain and Decrypted images
Malayali script	0.0099	1.0
Kannada script	0.0104	1.0
Telugu script	0.0097	1.0
Tamil script	0.0107	1.0

Correlation Analysis: Correlation analysis is used to measure the degree of association among the pixels. Neighboring pixels possess tight relations between them, a good cryptosystem will lessen this relation to prevent any attacks. Correlation coefficient r_{xy} can be calculated for horizontal and vertical directions. r_{xy} can be defined as:

$$\text{E}(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (6),$$

$$\text{D}(x) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x_i))^2 \quad (7),$$

$$\text{cov}\ (x, y) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x_i))(y_i - E(y_i)) \quad (8)$$

$$r_{xy} = \frac{cov\ (x,y)}{\sqrt{D(x)} \times \sqrt{D(y)}} \quad (9)$$

5000 neighboring pixel pairs are randomly selected for calculating correlation coefficient. As per the results shown in Table 4, we can conclude that the relation among pixel pairs of a plain image can be broken in an encrypted image by the proposed system.

Table no 4: shows Correlation analysis in plain and encrypted images

Image	Plain Image		Encrypted Image	
	Horizontal	Vertical	Horizontal	Vertical
Malayali script	0.6050	0.6767	-0.0660	0.0412
Kannada script	0.7107	0.6283	-0.0262	-0.0259
Telugu script	0.7971	0.7971	-0.0117	-0.0625
Tamil script	0.9463	0.9600	-0.0374	-0.0558

NPCR Analysis: Number of Pixels Change Rate (NPCR) is used to measure a system's ability to resist any differential attacks. Formula of NPCR is given below:

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{N \times N} \times 100\%$$

Where D(i) = {0 if C_1(i, j) = C_2(i, j),

1 if C_1(i, j) ≠ C_2(i, j)} (10)

Here C_1 represents the first encrypted image and C_2 represents the second encrypted image after modifying one of the pixels in the unencrypted input image. As per the results shown in Table 5, they are compared with NPCR critical values and all cases are larger than critical values. Hence our proposed system can resist any differential attacks.

Table no 5: shows NPCR analysis between two different encrypted images

Image	NPCR (%)	Critical values for NPCR		
		$N_{0.05}$ = 99.5893%	$N_{0.01}$ = 99.5810%	$N_{0.001}$ = 99.5717%
Malayali script	99.6042	Pass	Pass	Pass
Kannada script	99.6122	Pass	Pass	Pass
Telugu script	99.6905	Pass	Pass	Pass
Tamil script	99.6224	Pass	Pass	Pass

Key Sensitivity Analysis: Key sensitivity analysis guarantees that any information cannot be recovered from a plain image if there is any modification in any keys. In our analysis we modify one of El-Gamal's key, one of Lorenz's key and one of Rossler's key, later we perform decryption and check PSNR, MSE, SSIM between decrypted image and plain image. In Fig 4.6, we presented all the decrypted images after modifying their keys. As per the results shown in Table 6, we can conclude that our system is very sensitive to any change in confidential keys.

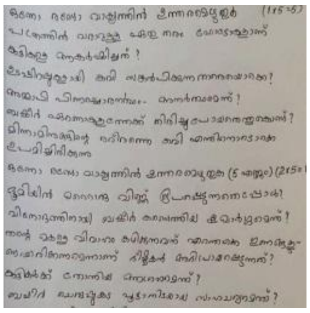

[illegible]

Figure 4.6(a)

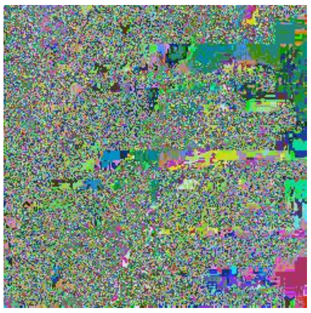


Figure 4.6(b)

Figure 4.6(c)

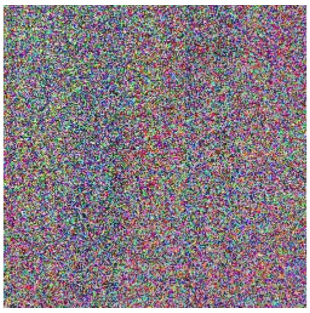


Figure 4.6(d)

Figure 4.6 Decrypted images: (a) plain image, (b) decrypted image of (a) after modifying'd' key of El-Gamal, (c) decrypted image of (a) after modifying 'x_0' of Lorenz system and (d) decrypted image of (a) after modifying 'z_0' of Rossler system.

Table no 6: shows PSNR, MSE and SSIM between plain and decrypted images after modifying keys

Parameter	Decrypted image after modifying 'd' key of El-Gamal	Decrypted image after modifying 'x_0' of Lorenz system	Decrypted image after modifying 'z_0' of Rossler system
MSE between plain and decrypted	5847.67	229.66	6743.93
PSNR between plain and decrypted	10.461	24.52	9.842
SSIM between plain and decrypted	0.0245	0.0962	0.0103

Occlusion Attack Analysis: In occlusion attack, different sized blocks are removed from the encrypted image and we decrypt these attacked images. We perform analysis like MSE, PSNR, SSIM between plain image and decrypted image to check how quality of image decreases after attack. In Fig4.7 we can observe the attacked encrypted images and in Fig 4.8 we can observe the respective decrypted images. As per the results shown in Table 7, we can conclude that quality of decrypted images decreases after the occlusion attack on encrypted images.

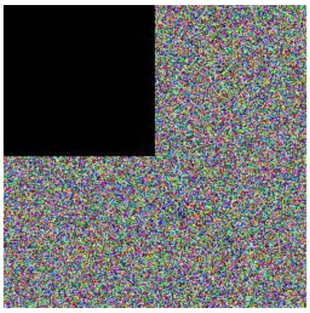


Figure 4.7(a)

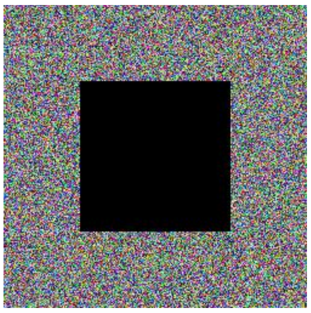


Figure 4.7(b)

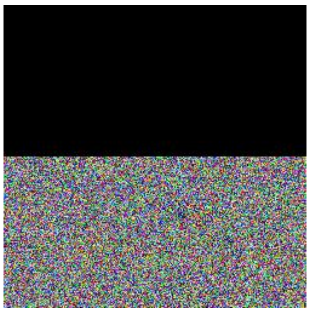


Figure 4.7(c)

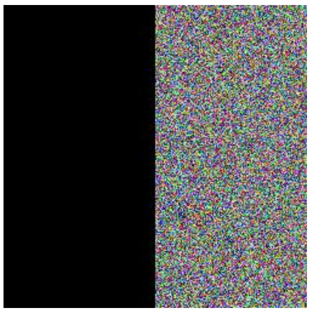


Figure 4.7(d)

Figure 4.7 Encrypted images with occlusion attacks: (a) ¼ data loss at top left, (b) ¼ data loss at center, (c) ½ data loss at top half and (d) ½ data at left half.

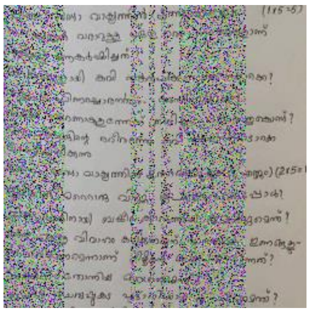


Figure 4.8(a)

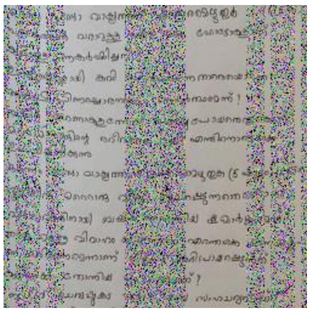


Figure 4.8(b)

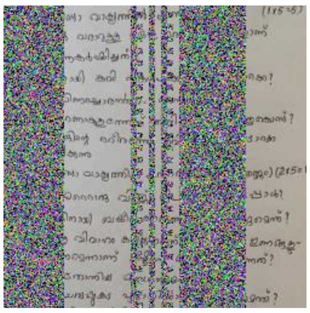


Figure 4.8(c)

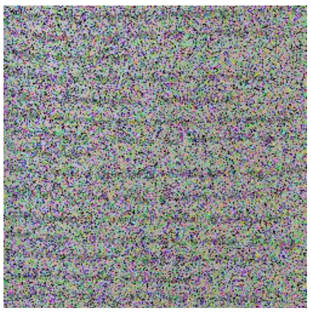


Figure 4.8(d)

Figure 4.8 Decrypted images: (a) of ¼ data loss at top left, (b) of ¼ data loss at center, (c) of ½ data loss at top and (d) of ½ data at left.

Table no 7: shows MSE, PSNR and SSIM between plain images and attack affected decrypted images.

Data portion	MSE between plain and decrypted	PSNR between plain and decrypted	SSIM between plain and decrypted
¼ data loss at top left	2399.35	14.33	0.4863
¼ data loss at center	2355.55	14.41	0.4762
½ data loss at top	4776.33	11.34	0.4191
½ data loss at left	4787.23	11.33	0.1001

Noise Attack Analysis: Noise is the most common attack on encrypted images. Some of the common noises are Gaussian, Salt & Pepper, Poisson, etc. In our analysis we apply gaussian noise with 2 different noise densities to one of the encrypted images similarly we apply salt & pepper noise with 2 different noise densities to one of the encrypted images. Fig4.9 shows the noise affected encrypted images and Fig4.10 shows the corresponding decrypted images. As per the results shown in Table 8, we can conclude that quality of decrypted image increases with decrease in noise intensity in encrypted image.

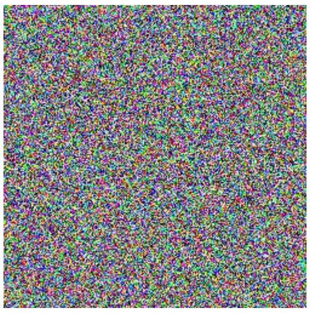


Figure 4.9(a)

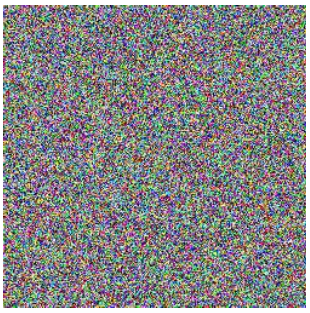


Figure 4.9(b)

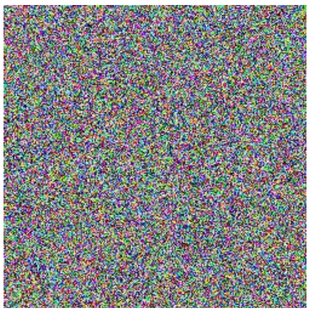


Figure 4.9(c)

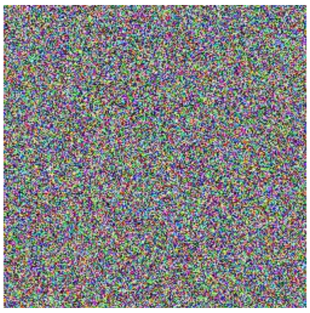


Figure 4.9(d)

Figure 4.9 Encrypted images with noise: (a) gaussian noise with density 0.2, (b) gaussian noise with density 0.1, (c) salt & pepper noise with density 0.2 and (d) salt & pepper noise density with density 0.1

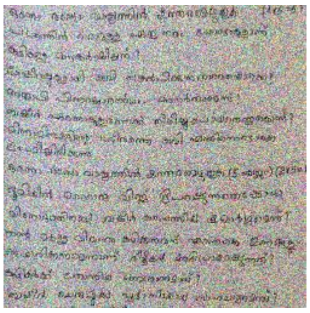


Figure 4.10(a)

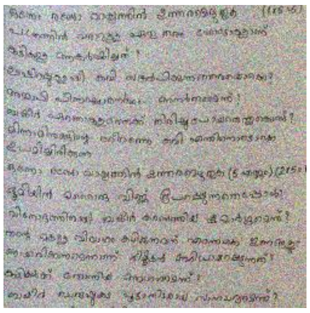


Figure 4.10(b)

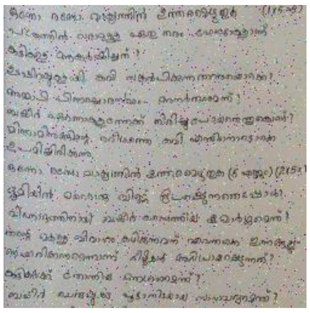


Figure 4.10(c)

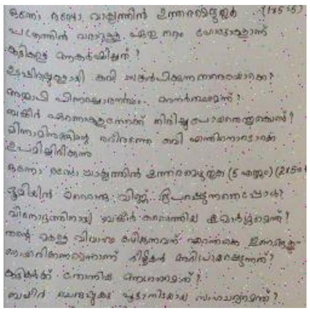


Figure 4.10(d)

Figure 4.10 Decrypted images: (a) for gaussian noise with density 0.2, (b) for gaussian noise with density 0.1, (c) for salt & pepper noise with density 0.2 and (d) for salt & pepper noise density with density 0.1

Table no 8: shows MSE, PSNR and SSIM between plain images and decrypted images of noise attacked encrypted images.

parameter	Gaussian noise with density 0.2	Gaussian noise with density 0.1	Salt & Pepper noise with density 0.2	Salt & Pepper noise with density 0.1
MSE between plain and decrypted	4028.81	2056.45	3692.50	1910.25
PSNR between plain and decrypted	12.08	15	12.46	15.32
SSIM plain and decrypted	0.3	0.41	0.32	0.46

Time and Complexity Analysis: In time and complexity analysis, we calculate the time taken to complete each stage of encryption for a given image. As per results shown in Table 9, we can conclude that as resolution increases time increases. We can also observe that our proposed method in stage-1 of encryption took very less time compared to other stages. The Lorenz system took the most amount of time for computation due to sorting of sequences. So, our proposed method takes more time if the resolution of the input image increases.

Table no 9: shows time taken for each stage of encryption for various resolutions

Resolution	MVK Algorithm Time (s)	El-Gamal encryption time (s)	Lorenz encryption time (s)	Rossler encryption time (s)	Total time (s)
250 X 250	0.5s	3s	1min 23s	14s	1min 41s
500 X 500	1.5s	5s	6min 34s	1min 13s	7min 54s
750 X 750	2.5s	8s	15min 47s	2min 45s	18min 43s
1000 X 1000	3s	13s	25min 44s	5min 26s	31min 15s

5.CONCLUSIONS

This research provides the newer approach to secure images by using MVK Algorithm, El-Gamal and Chaotic Systems. Initial stage performs the shuffling process which breaks the relations between neighboring pixel pairs. Chaotic systems improved the security by confusion and diffusion process. All the numerical results suggest that it can resist from any statistical, differential and exhaustive attacks. In future work, we can still make use of other chaotic systems to reduce time taken for encryption and further increase its security and make it more robust.

REFERENCES

[1]. R. M. Rad, A. Attar, and R. E. Atani, ''A new fast and simple image encryption algorithm using scan patterns and XOR,'' Int. J. Signal Process., Image Process. Pattern Recognit., vol. 6, no. 5, pp. 275–290, Oct. 2013.
[2]. X. Chai, Z. Gan, K. Yuan, Y. Chen, and X. Liu, ''A novel image encryption scheme based on DNA sequence operations and chaotic systems,'' Neural Comput. Appl., vol. 31, no. 1, pp. 219–237, 2019.
[3]. A. S. Hameed, ''Image encryption based on fractional order Lorenz system and wavelet transform,'' Diyala J. Eng. Sci., vol. 10, no. 6, pp. 81–91, Mar. 2017.
[4]. J. Wu, X. Liao, and B. Yang, ''Color image encryption based on chaotic systems and elliptic curve ElGamal scheme,'' Signal Process., vol. 141, pp. 109–124, Dec. 2017.
[5]. Z. Tang, Y. Yang, S. Xu, C. Yu, and X. Zhang, ''Image encryption with double spiral scans and chaotic maps,'' Secur. Commun. Netw., vol. 2019, pp. 1–15, Jan. 2019.
[6]. F. Abundiz-Pérez, C. Cruz-Hernández, M. A. Murillo-Escobar, R. M. López-Gutiérrez, and A. Arellano-Delgado, ''A fingerprint image encryption scheme based on hyperchaotic Rössler map,'' Math. Problems Eng., vol. 2016, pp. 1–15, Jan. 2016.