International Journal of Cyber-Security and Digital Forensics (IJCSDF) 10(1): 17-27
The Society of Digital Information and Wireless Communications (SDIWC), 2021 ISSN: 2305-0011

A Comparative Study of Different Image Encryption and Decryption Techniques

Yash Raj, Simriti Koul, Simran Koul
School of Computer Science and Engineering
VIT University, Vellore
Tamil Nadu, India
yashraj.vit01@gmail.com, simriti.koul@yahoo.com, simran.koul@yahoo.com

ABSTRACT

Currently the digital communication generates millions of digital data in the form of digital images. These confidential images must be protected from intruders over transmission in network channels. Various Image Encryption techniques exist which make use of symmetric as well asymmetric cryptography algorithms to secure the digital media. In this paper, we present 4 different methods to perform Image Encryption – Elliptic Curve Cryptography (ECC) with Hill Cipher, ECC with Advanced Encryption Standard (AES), ECC using a mapping table, and ElGamal with Double Playfair Cipher. The algorithms are tested over both grayscale and RGB images. Comparisons are made through parameters like time taken for encryption and decryption, entropy of encrypted image, loss in intensity of decrypted image, Peak Signal to Noise Ratio (PSNR), Number of Pixels Change Rate (NPCR), and Unified Average Changing Intensity (UACI).

KEYWORDS

Artificial neural networks; Cryptography; Cyber-security; Decryption; Encryption

1 INTRODUCTION

A lot of information is apparent when we observe an image. It has been observed that nearly one-third of our cortical region of brain is dedicated to visual processing of the information perceived. Images have become an important source of information. Images have various applications in a variety of fields such as storing medical information of patients, satellite imaging capturing the aerial images, telescopes capturing the inter-planetary motion images, storing an individual's identity in form of finger prints or iris images and much more.

When images are to be kept private and need to be transferred safe and securely, image encryption comes into play. The encryption task involves distorting the pixel intensities of the input image to obtain a cipher image which visually completely different from the original image. The receiver decrypts the cipher image using the secret keys and obtains the original image back [6].

In asymmetric key cryptography, there are different private keys used by sender and receiver which are further used to generate the shared secret key. On the other hand the symmetric key cryptography involves encryption and decryption with a single key secretly known to sender and receiver [2][6]. From several ways of protecting the information stored in the images, most common processes involve symmetric approaches like AES, DES, Hill cipher, etc. Although they are easy to implement and quick in their processing, these methods lack the amount of security provided to the image. This is overcome by incorporating asymmetric techniques like RSA, ECC, ElGamal, etc. which provide more security but with a trade-off in ease of implementation and computational feasibility.

Several other methods combine intra-disciplinary algorithms from Machine Learning like Artificial Neural Networks (ANN) to encrypt the image [18]. However, such processes get more difficult to deploy when it comes to time complexity of the algorithm and having the disadvantage of communicating several encryption and decryption parameters through an unsecure channel.

This allows up to explore hybrid processes which involve the speed and ease of implementation from symmetric algorithms as well as increased security from the asymmetric ones. Elliptic curve and ElGamal cryptosystems involve asymmetric key cryptography while Hill Cipher, AES, and Double Playfair Cipher are

symmetric key algorithms. Asymmetric key cryptography is highly secure but involves intense computation complexities which are further eased using a symmetric key cryptography algorithm in addition to asymmetric key cryptography.

2 LITERATURE SURVEY

Elliptic Curve Cryptography has been a growing technique in protecting the images being transferred through the web. The asymmetric mechanism works on an elliptic curve over finite prime fields [4]. Li Li et al. [5] selected the parameter of the elliptic curve to resist Pollard's rho, isomorphic and Pohlig Hellman attack. The experimental result indicated that it is better than encryption using RSA and ElGamal. S. Maria Celestin Vigila et al. proposed a procedure to perform image encryption using ECC with a coupled linear congruential generator to generate the private key and a random integer k. However, the algorithm fails to optimize the complexities involved in mapping every pixel to a point on the curve [17].

Various works have been done in [14], [15], [16], [17] to allow ECC encryption after mapping every pixel to a point on a pre-defined elliptic curve. The algorithm serves to be faster in performance but it involves complex pre-computations of finding every point on the elliptic curve for a large value of prime number p used to generate the finite field. It also includes communication of large mapping table over insecure channels for decryption process to be performed correctly.

The algorithms used in [1], [3], [19] try to optimize such computational complexities by avoiding the use of mapping table. Instead the pixels are grouped in pairs and are treated as cartesian co-ordinates. These pairs are then used to perform further calculations with the shared secret key of the elliptic curve. Several research works have focused only on the use of AES as the encryption and decryption algorithm [7], [8], [9]. Although the algorithm in [7] claims to perform better over other methods, it did not include concrete results for the same through various security and encryption quality metrics like entropy, NPCR, UACI, etc. Another approach in [8] relates AES with visual cryptography and establishes good results by encrypting the image using AES and the original key using visual cryptography by converting it to an image. The algorithm is still susceptible to an attack over the image shares created for the key.

Hayder Raheem Hashima et al. [10] used ElGamal encryption as the asymmetric encryption algorithm and was tested using MATLAB. The work concluded that it requires increasing time for computation with the use of large prime number as the encryption parameter. One of the researches [11] was done to improve the security of text encryption provided by Double Playfair Cipher by using 6 × 6 key matrices over the standard 5 × 5 key matrices. However, the algorithm failed due to data loss over certain characters like spaces and special symbols. The work done by S. M. Hardi combined the use of ElGamal cryptosystem and Double Playfair cipher for securing the text data with the use of standard keys for the symmetric process. Although, the algorithm claims to work on digital media too, it fails to analyse the security measures and metrics [12].

Safwat Hamad and his team has worked on increasing the security of image encryption through standard Playfair cipher with a modified key of size 16 by 16 based on the range of a 8-bit image pixel value. The results are further improved by performing XOR operation with the help of a randomly generated mask. The algorithm's additional security through the XOR function fails if the mask is eavesdropped by the intruder [13].

In the subsequent portion of the paper we would be analysing the performance of 4 methods – Elliptic Curve Cryptography (ECC) with Hill Cipher, ECC with Advanced Encryption Standard (AES), ECC using a mapping table, and ElGamal with Double Playfair Cipher. The performance analysis of each of these algorithms would be done on the basis of parameters like time taken for encryption and decryption, entropy of encrypted image, loss in intensity of decrypted image, Peak Signal to Noise Ratio (PSNR), Number of Pixels Change Rate (NPCR), and Unified Average Changing Intensity (UACI).

3 PROPOSED METHODOLOGY

In this section we discuss in detail about the proposed hybrid algorithms making use of various symmetric and asymmetric techniques for image encryption. For the comparison of all the different methods incorporated further, we set up certain guidelines – both general and specific to an algorithm. These guidelines may also include rules to be followed while choosing certain parameters so as to perform encryption and decryption correctly. The complete set of rules help us in doing a comparative study (through various performance measures as described in the next section) of each algorithm and also allows us to analyze the work done through identical parameters set up for each method.

The general constraints to be followed are:

•The size of image considered for encryption is 256 × 256 pixels. The size of encrypted image is same as input image except for algorithm using AES with ECC where size of encrypted image is slightly more. All decryption algorithms produce the decrypted image of the same size as that of original image.
•For keeping the type of images both gray-scale or RGB, as the choice of user, we deploy an algorithm that always encrypts and decrypts all 3 channels (RGB) of the image separately.
Before moving forward towards different algorithms, let us first mathematically define the Elliptic Curve Cryptography which will be used further in the different proposed algorithms.

3.1 An Overview of Elliptical Curve Cryptography:

We define elliptic curves over a finite prime field and the same would be used by different proposed algorithms. Mathematically the curve is defined as:
$E(Fp) = \{a, b, p, G\}$
$y2 \equiv x3 + ax + b\ (mod\ p)\ and\ 4a3 + 27b2 \not\equiv 0\ (mod\ p)$

where,
$Fp\ is\ the\ finite\ field\ over\ a\ prime\ number\ p\ with\ generator\ G$
$a, b\ are\ curve\ parameters$

The generator of the curve G is the point whose point multiplication with different scalars produce every point on the curve [20]. Further we define the order of elliptic curve n as the smallest integer whose scalar point multiplication with generator G gives us the point at infinity O for the curve i.e., $nG = O$.

For the proposed algorithms we need to generate the keys for elliptic curve cryptography. The key generation algorithm is given as:

3.1.1 Key generation of ECC

• Receiver establishes Elliptic curve parameters: $(Fp) = \{a, b, p, G\}$.
• Choose a private key nB in the range: $\{1, p - 1\}$.
• Find the public key: $PB = nBG$ using point multiplication over the curve.
• Publish the public key PB and the curve parameters $(Fp) = \{a, b, p, G\}$.
• Sender gets the public key of receiver PB along with associated curve parameters.
• Sender chooses a private key nA in the range: $\{1, p - 1\}$.
• Sender computes his public key as: $PA = nAG$.
• The shared secret key (SSK) as computed by sender will be: $SSK = nAPB = nAnBG = (x, y)$.

The steps of key generation are followed by the described algorithms that use ECC as the asymmetric approach.

3.2 Details of Proposed Algorithms

Algorithm 1: ECC with Hill Cipher

Guidelines and characteristics specific to the algorithm:
• The matrix generated for Hill Cipher is a self-invertible matrix i.e. $Km = K-1$. The size of self-invertible matrix used in literature is usually 2 × 2. On using 4 × 4 size self- invertible matrix performs the encryption and decryption process comparatively faster as well as produces more distortion in the encrypted image by using more number original image pixels to generate the corresponding cipher pixels. As the matrix is derived from the shared secret key, there is no need to send the matrix with the encrypted image.

• The algorithms in the literature focus on encryption of gray-scale images only while the generality of the following algorithm comes from its ability to encrypt color image as well, making it more useful.

Input:
The image of size 256 × 256 to be encrypted or decrypted. Elliptic curve parameters: $(Fp) = \{a, b, p, G\}$.

Output:
The corresponding cipher image or original image of size 256 × 256.

Method:
Step 1. Key Generation for ECC
The ECC keys are generated as described in the algorithm in Section 2.1.1.

Step 2. Computing the self-invertible matrix
1. Compute: $K1 = xG = (k11, k12)$ and $K2 = yG = (k21, k22)$.
2. Compute: $K12 = I2 - K11$; $K21 = I2 + K11$; $K11 + K22 = 0$.
3. The 4 × 4 self-invertible matrix is found as:
$Km = [\ 11\ K12\ K21\ K22\ ]$,
$where$
$K11 = [k11\ k12\ k21\ k22]$.

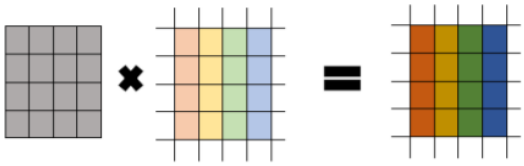
Step 3. Encryption process

1. Read the image to be encrypted and collect the image pixels separately for the channels R, G, and B.
2. Group every channel of pixels into 4 × 4 matrices and perform matrix multiplication with computed self-invertible matrix.
3. The encryption is done using the subsequent formula:

$$C_i = K_m . P_i = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

where,
$Km\ is\ the\ self - invertible\ matrix\ and$
$Pi\ is\ the\ current\ input\ image\ block\ to\ be\ encrypte$

4. Allocate the cipher pixels exactly to the same position as of the corresponding input image pixels. A cipher image is formed of size identical to the size of input image.
5. Send the cipher image and ECC public key to the receiver. The encryption process can be visualized as in Fig. 1 given below.



Step 4. Decryption process

1. Compute the shared secret key from the public key of sender as:
$SSK = nBPA = nBnAG = nAnBG = (x, y)$.
2. Compute the self-invertible matrix exactly as described above using the shared secret key.
3. Group the cipher pixels into 4 × 4 matrices and perform matrix multiplication with self- invertible matrix from step 2.
4. Allocate the plain text pixels to the same position as of the corresponding cipher pixels. The original image is formed back and retrieved as without any intensity loss.

Algorithm 2. ECC with AES

Guidelines and characteristics specific to the algorithm:

• AES key and initialization vector (IV) must be of equal length and it should be a multiple of 16 or 24 or 32 bytes respectively for AES-128, AES-192 or AES- 256 encryption/decryption.
• The AES encryption and decryption is performed using Cipher Feedback (CFB) mode. If Electronic Code Book (ECB) mode is used then IV is not required.
• The AES encrypted bytes are converted to large integers to save the number of operations by encrypting 2 × $group\ size$ number of bytes in one ECC operation.
• Base 256 representation is chosen as it would have values 0 to 255 in it and the algorithm is working on 8-bit image whose pixel intensities also range from 0 to 255.

• While performing ECC encryption, we simply make use of point addition formulae to encrypt any point on the XY-plane. The advantage of such an operation saves us from creating a mapping table (which becomes computationally impossible if a very large prime number is used for generating the finite field) and sharing it between the users.
• On performing decryption we simply reflect the SSK co-ordinates with respect x-axis and taking modulus p. The reflected point is then used with point addition formulae for performing the inverse operation used in encryption method.

Input:
The image of size 256 × 256 to be encrypted or decrypted. Elliptic curve parameters: $(Fp) = \{a, b, p, G\}$.
AES symmetric key and IV.

Output:
The corresponding cipher image or original image of size 256 × 256.

Method:
Step 1. Key Generation for ECC
The ECC keys are generated as described in the algorithm in Section 2.1.1.

Step 2. Key Generation for AES
1. Sender randomly generates 16 bytes long AES symmetric key and the Initialization Vector.
2. These parameters must be securely transmitted to receiver.

Step 3. AES Encryption
1. Read the image to encrypted and collect pixels for separate channels: R, G, B.
2. Convert the pixels in each channel to bytes using the function
$byte(PL)$ in Python where PL denotes the list of pixels. The function returns an immutable array of bytes.
3. Perform AES encryption for each of the channel bytes. This generates bytes in the encrypted form.

Step 4. ECC Encryption
1. Choose a base of representing the numbers. We have chosen 256 to represent the range of 8-bit pixel values.
2. Represent the prime number p in base 256 as a list of integers from 0 to 255. Measure the length of this representation as L. Initialize group size as: $L - 1$.
3. Group the cipher bytes from each channel separately with group size initialized as above. For each of the channels, convert each group of cipher bytes to big integers treating base of representation as 256. If number of big integers are odd, append with a random value (possibly less than 256).
4. Pair up the big integers to represent it as a point on the XY-plane. If the X co-ordinate of the pair and that of the shared secret key are identical, then we apply point doubling formula over the pair co-ordinates. Otherwise apply point addition formula with SSK co-ordinates.
5. From step-4, we get a point on XY plane (not at all necessary to be on the curve chosen). Represent its co-ordinates in base 256 digits. Each of these representations must be of size equal to $L = group\ size + 1$. Append 0 zeros otherwise to make the required length.
6. The base 256 representations of cipher points is treated as image pixels and cipher image is constructed correspondingly. The width of cipher image is kept same as width of input image and height is computed based on the total number of cipher pixels collected. We observe that size of cipher image is generally more than the input image.
7. Send the cipher image, original image size and ECC public key to receiver.

Step 5. ECC Decryption
1. Compute the shared secret key from the public key of sender as:

$SSK = nBPA = nBnAG = nAnBG = (x, y)$.

2. Represent the prime number p in base 256 as a list of integers from 0 to 255. Measure the length of this representation as L. Initialize group size as: $group\ size = L - 1$.
3. Collect the cipher pixels for each channel and group them with each group size equal to $group\ size + 1$.
4. Convert each group in respective channels to corresponding big integers taking base to be 256. Pair up the big integers treating them as points on the XY-plane.

5. Perform reflection of SSK point with respect x-axis: $(x, -y \ mod \ p)$.
6. Perform point doubling with reflected SSK co-ordinates if the x co- ordinates are same or perform point addition formula otherwise. The point obtained for each pair lies on XY-plane but need not be a point the elliptic curve.
7. Represent each of the points obtained in base 256 with each representation of length equal to $group \ size$. Append zeros if required.
8. Obtained values are bytes obtained from AES encryption.

Step 6. AES Decryption

1. Obtain the AES key and IV from secure communication.
2. Perform AES decryption for the bytes obtained from above for each of the respective image channels.
3. Represent the bytes as the plain text image of size as mentioned by the sender.

Algorithm 3: ECC with mapping table

Guidelines and characteristics specific to the algorithm:

• Find every point for defined Elliptic curve. Note that (0, 0) must not satisfy the curve as it will be treated as our point at infinity. The time complexity of finding every point on the curve is $(p2)$ [20]. Therefore, the prime number p is limited to smaller values.

• Start mapping every point to an intensity value and start from zeroth position if there are more than 256 points. Add (0, 0) as the point at infinity, mapped to a pixel value, to the mapping table. The mapping table created is shared securely with the receiver.

• For the encryption process follow the following rules:

- If the point to be encrypted is same as that of the shared secret key then perform point doubling of the point to be encrypted.
- If only the x-coordinate of point to be encrypted is same as that of the shared secret key then the cipher point becomes point at infinity i.e. (0, 0).
- Otherwise perform point addition between the point to be encrypted and shared secret key.

• Initialize SSK' as reflection of shared secret key point. Now for the decryption process follow the following rules:

- If the point to be decrypted is same as that of SSK' then perform point doubling of the point to be decrypted.
- If the point to be decrypted is (0, 0) i.e. the point at infinity then the decrypted point will be same as SSK'.
- Otherwise perform point addition between the point to be decrypted and the shared secret key point.

• Two maps are generated based on the mapping table – first one maps pixel intensities to an array of mapped points (in tuples) and second one maps points as tuples to the corresponding mapped intensity value. The use of 2 maps optimizes the time complexity of algorithm by returning the pixel value corresponding to a point in O(1) while finding the cipher pixels in encryption or original pixels in decryption.

Input:

• The image of size 256 × 256 to be encrypted or decrypted.

• Elliptic curve parameters: $(Fp) = \{a, b, p, G\}$.

• Mapping table corresponding to the elliptic curve defined.

Output:

• The corresponding cipher image or original image of size 256 × 256.

Method:

Step 1. ECC key generation

The ECC keys are generated as described in the algorithm in Section 2.1.1. It is made sure that the point (0, 0) does not satisfy the chosen elliptic curve.

Step 2. Generating the mapping table

1. For every value of x from 0 to p-1 and every value of y from 0 to p-1 find the RHS as $x3 + ax + b \ (mod \ p)$ and the LHS as $y2 \ (mod \ p)$.
2. If $LHS \equiv RHS$ then add (x, y) and $(x, -y \ mod \ p)$ to the list of all points of the curve. Add (0, 0) to the list of points.

Find the total number points generated on the curve.
3. Create random shuffle of all pixel values as obtained by a 8-bit representation.
4. Starting from the first pixel intensity map one by one every pixel intensity to a point from the list of points. If there are more than 256 points in the list, start mapping it from first pixel intensity value. Repeat till all points are mapped.

Step 3. Encryption process
1. Read the mapping table and create a map with keys to be pixels and values to be an array containing all those points (in tuples) that were mapped to a given intensity value. Simultaneously, create a second map that has keys as point tuples and values as pixels.
2. Read the input image and group the pixels based upon image channels: R, G, B.
3. Map each channel separately to corresponding points on the curve using the first map which has unique keys as pixel intensities and corresponding values as points of the curve.
4. Perform encryption as stated in the specific guidelines to this algorithm. Use the second map to obtain the encrypted point values corresponding to pixel intensities. Create the cipher image using the encrypted pixel values.
5. Send the cipher image, ECC public key of sender and the mapping table to the receiver.

Step 4. Decryption process

1. Compute the shared secret key from the public key of sender as: $SSK = nBPA = nBnAG = nAnBG = (x, y)$.
2. Read the mapping table and create a map with keys to be pixels and values to be an array containing all those points (in tuples) that were mapped to a given intensity value. Simultaneously, create a second map that has keys as point tuples and values as pixels.
3. Read the cipher image and group the pixels based upon image channels: R, G, B.
4. Map each channel separately to corresponding points on the curve using the first map which has unique keys as pixel intensities and corresponding values as points of the curve.
5. Perform reflection of SSK point with respect x-axis: $SSK' = (x, -y \ mod \ p)$. Perform decryption as stated in the specific guidelines to this algorithm.
6. Use the second map to obtain the point values corresponding to pixel intensities. Create the original image using the decrypted pixel values.

Algorithm 4. ElGamal with Double Playfair Cipher
Guidelines and characteristics specific to the algorithm:
• The algorithm uses 2 key matrices for Double Playfair Cipher but the size of these key matrices used is 16 × 16 instead of the traditional 5 × 5 keys used in text encryption. The matrix sizes correspond to the number of distinct pixel values represented by any 8-bit image which is 0 to 255 or 256 distinct values.
• The use of key maps increases the speed of the algorithm to a great extent by saving the time to traverse the entire key matrices twice in the encryption or decryption process of a single pixel. The position of pixel in key matrices is answered in (1) with use of just $O(2 \times 3 \times 356)$ of extra space and a pre-computation time of $O(256)$. The time optimization of the algorithm clearly reflects when encrypting images for large sizes.
• Both key matrices are encrypted using ElGamal encryption before sending it to the receiver. We need not send the key maps as they can be easily generated on the receiver end once the symmetric keys are shared and derived.

Input:
• The image of size 256 × 256 to be encrypted or decrypted.
• The key matrices corresponding to Double Playfair Cipher.
• The ElGamal cryptosystem parameters.

Output:
• The corresponding cipher image or original image of size 256 × 256

Method:
Step 1. Key Generation for Double Playfair Cipher

1. A modified key space of 2 matrices, each of size 16 × 16 is used instead of standard 5 × 5 key matrices used in Double Playfair cipher.
2. Create 2 key maps, 1 corresponding to each of the keys. The key maps have pixel intensities as keys and their location in the respective key matrix as the corresponding value (represented by a tuple of row and column).

Step 2. Key Generation for ElGamal Cryptosystem

1. Receiver chooses a prime number p and computes its generator $g.\equiv gd\ (mod\ p)$.
2. Publish the public key, prime number p, and generator g.
3. Sender chooses a random integer i in the range: $\{2,\ p\ -\ 2\}$. Sender computes the temporary key or ephemeral key as: $KE \equiv gi\ (mod\ p)$.
4. The masking key is computed by sender as: $KM\ \equiv Kpub\ (mod\ p)$.

Step 3. Encryption process
1. Read the image to be encrypted and separate it to different channels as R, G, B. Pair up pixels in 2 for each of the channels separately.
2. To apply Vertical Double Playfair Cipher first we find the location of first value from pixel pair from the first key map and the location of second pixel value from the second key map. Followed by which, if the pixels are in same column then keep them unchanged. Otherwise, find the pixel intensities at the opposite corners of the rectangle formed by the pair of input pixels considered. Place them in the same order as they correspond to the order of pixel values in the input pair.
3. Substitute the cipher pixel values to create the cipher image of size identical to that of the input image.
4. Encrypt each of the key matrices using the masking key from ElGamal Encryption as:
$cij \equiv pij \times Km\ (mod\ p)$
Where, pij denotes the corresponding pixel intensity in a cell of a key matrix.
5.Send the encrypted image and encrypted key matrices along with ElGamal public key (ephemeral key) of sender to the receiver.

Step 4. Decryption process
1. The masking key will be computed by receiver as: $KM \equiv KEd\ (mod\ p)$.
2. The key matrices for Double Playfair Cipher are decrypted as: $pij \equiv cij \times Km{-}1\ (mod\ p)$.
3. Read the image to be decrypted and separate it to different channels as R, G, B. Pair up pixels in 2 for each of the channels separately.
4. To apply Vertical Double Playfair Cipher first we find the location of first value from pixel pair from the first key map and the location of second pixel value from the second key map. Followed by which, if the pixels are in same column then keep them unchanged. Otherwise, find the pixel intensities at the opposite corners of the rectangle formed by the pair of input pixels considered. Place them in the same order as they correspond to the order of pixel values in the input pair.
5. Substitute the decrypted pixel values to create the original image.

4 RESULTS

4.1 Sample Input and Output

The following test cases help us in visualizing the input image, encrypted image and the decrypted image for each of the algorithms.

Image Name	Original Image	Encrypted Image (ECC with Hill Cipher)	Encrypted Image (ECC with AES)	Encrypted Image (ECC with mapping table)	Encrypted Image (ElGamal with Double Playfair cipher)	Decrypted Image
Mona Lisa (Grayscale)						
Mona Lisa (Coloured)						
Eggs (Grayscale)						
Eggs (Coloured)						

4.2 Metric Measures

The tuple, if present in the table, denotes metrics value in the order of (R, G, B).

4.2.1 Eggs

Grayscale				
Evaluation metrics	**ECC with Hill Cipher**	**ECC with AES**	**ECC with mapping table**	**ElGamal with Double Playfair cipher**
Encryption Time	0.33418	2.82401	1.59456	0.18775
Decryption Time	0.36932	2.75127	1.26276	0.17856
Entropy	7.9434	7.99653	7.2532	7.78708
Squared Error in decrypted image	0	0	0	0
PSNR	9.45457	9.129	9.4518	9.36168
NPCR	86.91254	99.60632	100	94.81812
UACI	26.06152	28.90828	27.39494	27.91177

Coloured				
Evaluation metrics	**ECC with Hill Cipher**	**ECC with AES**	**ECC with mapping table**	**ElGamal with Double Playfair cipher**
Encryption Time	0.32507	2.53642	1.65586	0.13366
Decryption Time	0.35904	2.52728	1.17281	0.16954
Entropy	(7.98534, 7.98371, 7.98925)	(7.99652, 7.99585, 7.99638)	(7.11174, 7.31097, 7.41393)	(7.85019, 7.87646, 7.90649)
Squared Error in decrypted image	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
PSNR	(9.41053, 8.95845, 8.74391)	(9.39128, 8.98602, 8.7218)	(9.63232, 9.32158, 9.37724)	(9.86406, 9.45456, 9.069)
NPCR	(96.67358, 96.81854, 96.99249)	(99.646, 99.63684, 99.62463)	(100, 100, 100)	(94.05823, 94.08875, 94.1803)
UACI	(27.87552, 29.14918, 29.73587)	(28.15297, 29.32043, 30.11092)	(27.15456, 27.80935, 27.80935)	(26.05094, 27.09534, 28.17803)

4.2.2 Mona Lisa

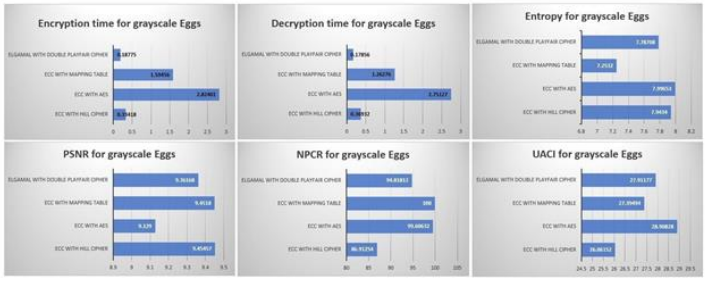
Grayscale				
Evaluation metrics	**ECC with Hill Cipher**	**ECC with AES**	**ECC with mapping table**	**ElGamal with Double Playfair cipher**
Encryption Time	0.44584	2.84150	1.59667	0.24519
Decryption Time	0.35107	2.7866	1.19981	0.20224
Entropy	7.99442	7.99671	7.44896	7.92129
Squared Error in decrypted image	0	0	0	0
PSNR	8.5959	8.62078	9.07994	9.12066
NPCR	97.41669	99.58191	100	94.13757
UACI	30.26131	30.37332	28.76783	27.97823

Coloured				
Evaluation metrics	**ECC with Hill Cipher**	**ECC with AES**	**ECC with mapping table**	**ElGamal with Double Playfair cipher**
Encryption Time	0.32915	2.52632	1.62	0.13013
Decryption Time	0.35193	2.50829	1.10915	0.1486
Entropy	(7.99701, 7.9971, 7.99723)	(7.99667, 7.99659, 7.99672)	(7.56468, 7.51776, 6.63265)	(7.95568, 7.96154, 7.8657)
Squared Error in decrypted image	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
PSNR	(8.80367, 8.45589, 7.91148)	(8.86691, 8.54116, 7.9478)	(9.17872, 9.10195, 8.67261)	(9.19181, 8.8655, 8.27107)
NPCR	(99.48273, 99.38507, 99.47968)	(99.63379, 99.64142, 99.6109)	(100, 100, 100)	(93.57605, 93.23425, 93.7561)
UACI	(29.85083, 31.02715, 32.87405)	(29.60341, 30.65091, 32.69575)	(28.56394, 28.8705, 29.93446)	(27.70115, 28.5317, 30.3973)

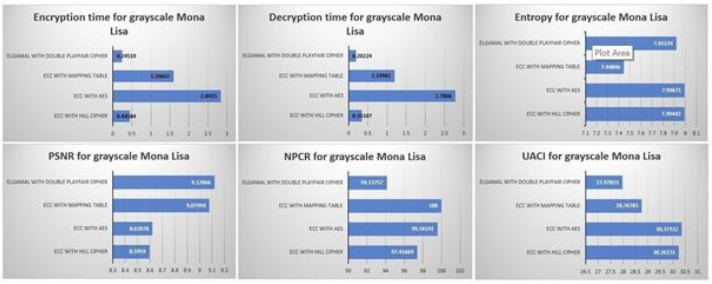
4.3 Performance Analysis through Charts

The tabulated data is visualized for the sample gray-scale images to get a better perception of the results.

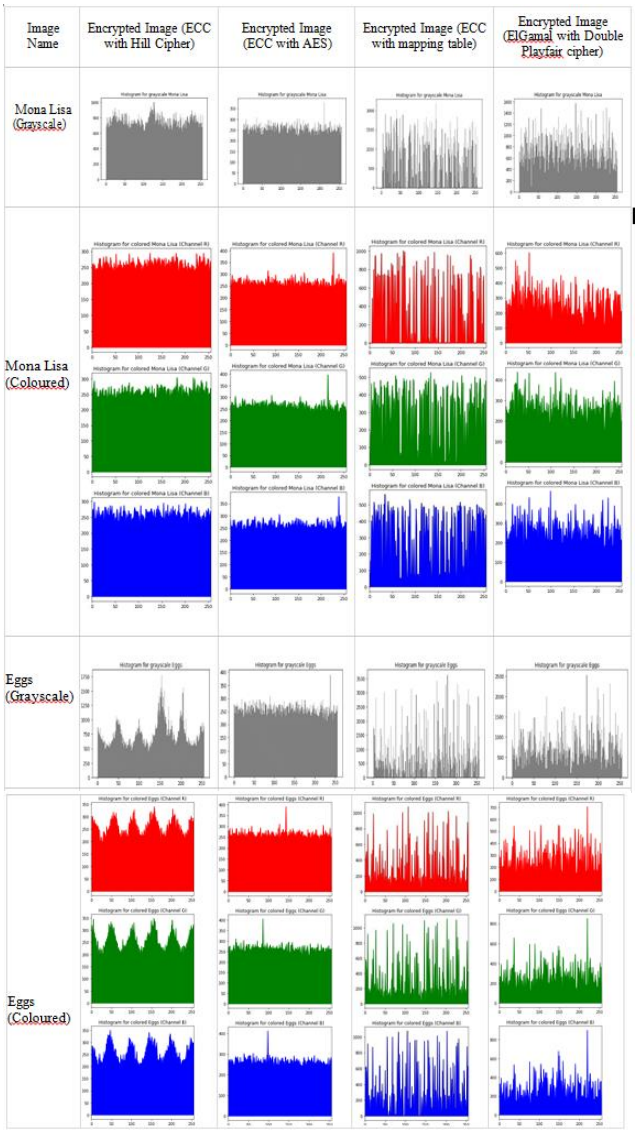
4.3.1 Eggs



4.3.2 Mona Lisa



4.4 Histogram analysis of the encrypted images



5 CONCLUSION

Various performance measures determine the security of the encryption provided by an algorithm to the input image. The entropy signifies the randomness of pixel intensities in the encrypted image. For an 8-bit image, an entropy value close to 8 implies a good

encryption algorithm. Similarly, a lower value of PSNR implies a higher randomness in the encryption process. Using the entropy and PSNR measures we can come to know that ECC with AES and ECC with Hill Cipher introduce considerably high amount of randomness in the encryption of the image with respect to the other 2 algorithms.

NPCR is also an indicative measure of the number of pixels that are different in encrypted image as compared to input image and is represented as a percentage value. Although the NPCR value for the algorithm ECC with mapping table is 100%, the lower entropy indicates that the probability distribution of the different pixel intensities is not uniform. This is because of the limitation of choosing a smaller value of prime number p as the complexity of algorithm increases exponentially (in $(p2)$) with respect to p. In most of the test samples the NPCR values for ECC with Hill Cipher and ECC with AES are very close to 100%.

The UACI measure indicates the algorithm's security against the differential attacks like chosen plaintext attack, cipher-text-only attack, or known plaintext attack. A higher UACI value for ECC with AES and ECC with Hill Cipher suggest that the algorithms are more secure against such attacks. Although, we can also infer that the UACI values for other 2 algorithms is very close to the former algorithms.

Based on the time taken for encryption and decryption we can identify that ECC with AES is computationally intensive and not feasible in applications involving protection of a large database of images. ECC with AES comes out to be good for remote or private communications involving a smaller image size. We can say that ECC with Hill Cipher turns out to be an overall good choice of encryption algorithm.

But we can also deploy the algorithm ElGamal with Double Playfair Cipher for applications demanding ease of implementation, speed, and large image sizes. However, the algorithm fails against brute-force attacks involving knowledge of large number of cipher texts. This can be minimalized by deploying a new pair of symmetric keys for every set of communication with the protection of ElGamal cryptosystem. Although the communication overhead increases a little, the algorithm can be found very useful for applications involving less frequency of communication between parties.

6 FUTURE WORK

In analyzing the working of each algorithm, we learn that there is a scope of improvement in the algorithm using ECC with a mapping table where, we could use task or data parallelism to identify every point on the chosen elliptic curve in finite field to allow this algorithm to be used for large values of p. This could lead to a significant improvement in the entropy values for the algorithm as the same pixel intensities (in a single image) would be mapped to different points on the curve leading to different encryption of the identical pixel intensities. Apart from this, we could also test the security of the algorithm – ElGamal with Double Playfair cipher using Four-Square cipher with a little more memory requirements.

REFERENCES

[1] Laiphrakpam Dolendro Singh and Khumanthem Manglem Singh: Image Encryption using Elliptic Curve Cryptography. Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015). Procedia Computer Science 54 (2015) pp. 472 – 481.

[2] Ziad E. Dawahdeh, Shahrul N. Yaakob, Rozmie Razif bin Othman: A new image encryption technique combining Elliptic Curve Cryptosystem with Hill Cipher. Journal of King Saud University – Computer and Information Sciences 30 (2018) pp. 349–355.

[3] Srinivasan Nagaraj, Dr. G. S. V. P. Raju, K.Koteswara Rao: Image Encryption Using Elliptic Curve Cryptography and Matrix. International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015). Procedia Computer Science 48 (2015) pp. 276 – 281.

[4] Alka Sawlikar: Point Multiplication Methods for Elliptic curve Cryptography. International Journal of Engineering and Innovative Technology (IJEIT) Volume 1, Issue 1, January 2012 pp. 1-4. ISSN 2277-3754.

[5] Li Li, Ahmed A. Abd El-Latif, Xiamu Niu: Elliptic curve ElGamal based homomorphic image encryption scheme for sharing secret images. Signal Processing, Volume 92 Issue 4, April (2012) pp. 1069-1078.

[6] Bibhudendra Acharya, Girija Sankar Rath, Sarat Kumar Patra, and Saroj Kumar Panigrahy: Novel Methods of Generating Self-Invertible Matrix for Hill Cipher Algorithm. International Journal of Security, Volume 1 : Issue (1), June (2007) pp. 14-21.

[7] Priya Deshmukh: An image encryption and decryption using AES algorithm. International Journal of Scientific & Engineering Research, Volume 7, Issue 2, February-2016, pp. 210-213. ISSN 2229-5518.

[8] Venkata Krishna Pavan Kalubandi, Hemanth Vaddi, Vishnu Ramineni, Agilandeeswari Loganathan: A Novel Image Encryption Algorithm using AES and Visual Cryptography. 2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016) pp. 808-813.
[9] Samiksha Sharma, Vinay Chopra: Analysis of AES encryption with ECC. Proceedings of International Interdisciplinary Conference On Engineering Science & Management (2016) pp. 195-201. ISBN: 9788193137383.
[10] Hayder Raheem Hashima, Irtifaa Abdalkadum Neamaa: Image Encryption and Decryption in A Modification of ElGamal Cryptosystem in MATLAB. International Journal of Sciences: Basic and Applied Research (IJSBAR) (2014) Volume 14, No 2, pp. 141-147. ISSN 2307-4531.
[11] Anirban Bhowmick, Anand Vardhan Lal, Nitish Ranjan: Enhanced 6x6 Playfair Cipher using Double Myszkowski Transposition. International Journal of Engineering Research & Technology (IJERT) Vol. 4 Issue 07, July 2015 pp. 1100-1104. ISSN: 2278-0181.
[12] S M Hardi, J T Tarigan and N Safrina: Hybrid cryptosystem for image file using elgamal and double playfair cipher algorithm. 2nd International Conference on Computing and Applied Informatics 2017 pp. 1-6.
[13] Safwat Hamad, Amal Khalifa, Ahmed Elhadad, S. Z. Rida: A Modified Playfair Cipher for Encrypting Digital Images. Journal of Communication & Computer Engineering (2013) Volume 3, Issue 2, pp. 1-9. ISSN 2090-6234.
[14] Kamlesh Gupta, Sanjay Silakari, Ranu Gupta, Suhel A. Khan: An Ethical way for Image Encryption using ECC. First International Conference on Computational Intelligence, Communication Systems and Networks (2009) pp. 342-345.
[15] Ali Soleymani, Md Jan Nordin, Azadeh Noori Hoshyar, Zulkarnain Md Ali, Elankovan Sundararajan: An Image Encryption Scheme Based on Elliptic Curve and a Novel Mapping Method. International Journal of Digital Content Technology and its Applications (JDCTA) Volume 7, No. 13, September 2013 pp. 85-94.
[16] Srinivasan Nagaraj and G. S. V. P. Raju: Image Security using ECC Approach. Indian Journal of Science and Technology, Vol 8(26), October 2015 pp. 1-5.
[17] S. Maria Celestin Vigila and K. Muneeswaran: Nonce Based Elliptic Curve Cryptosystem for Text and Image Applications. International Journal of Network Security, Vol. 14, no. 4, July (2012) pp. 236-242.
[18] I. A. Ismail, G. H. Galal-Edeen, S. Khattab and M. A. E. M. E. Bahtity, "Satellite image encryption using neural networks backpropagation," 2012 22nd International Conference on Computer Theory and Applications (ICCTA), Alexandria, 2012, pp. 148-152.
[19] Dr. Parmanand Astya, Ms. Bhairvee Singh, Mr. Divyanshu Chauhan: Image Encryption and Decryption using Elliptic Curve Cryptography. International Journal of Advance Research In Science And Engineering (IJARSE), Vol. No. 3, Issue No. 10, October (2014) pp. 198-205. ISSN- 2319-8354.
[20] Moumita Roy, Nabamita Deb, Amar Jyoti Kumar: Point Generation And Base Point Selection In ECC: An Overview. International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 5, May 2014 pp. 6711-6713. ISSN (Online): 2278- 1021