DOI: 10.24996/ijs.2023.64.11.41

ISSN: 0067-2904

# A Lightweight Image Encryption Algorithm Based on Elliptic Curves and a 5D Logistic Map

**Abdullah A. Rashid *, Khalid A. Hussein**

*Department of Computer Science, College of Education, AL-Mustanisiryah University, Baghdad, Iraq*

Received: 7/9/2022 Accepted: 5/1/2023 Published: 30/11/2023

## Abstract

Cryptography can be thought of as a toolbox, where potential attackers gain access to various computing resources and technologies to try to compute key values. In modern cryptography, the strength of the encryption algorithm is only determined by the size of the key. Therefore, our goal is to create a strong key value that has a minimum bit length that will be useful in light encryption. Using elliptic curve cryptography (ECC) with Rubik's cube and image density, the image colors are combined and distorted, and by using the Chaotic Logistics Map and Image Density with a secret key, the Rubik's cubes for the image are encrypted, obtaining a secure image against attacks. ECC itself is a powerful algorithm that generates a pair of public and private elements, and we generate secret key values with the help of the above key pair. This algorithm is applied to smart cards, sensors, and wireless network security. Whereas, the test results indicate that the highest value for PSNR is 7.9641, while the highest values for UACI and NPCR are 35.3721 and 99.9992, respectively. The encoding and decoding speeds reach 0.1561 and 0.1532 seconds, respectively.

**Keywords:** Chaotic, Cryptography, Density, NIST, Magic cube.

# خوارزمية تشفير الصور خفيفة الوزن المعتمدة على المنحنيات الإهليجية والخرائط اللوجستية *5D*

**عبد الله عادل رشيد *, خالد علي المياحي**

قسم علوم الحاسوب، كلية التربية، الجامعة المستنصرية، بغداد، العراق

## الخلاصة

يمكن اعتبار التشفير بمثابة صندوق أدوات، حيث يتمكن المهاجمون المحتملون من الوصول إلى موارد وتقنيات الحوسبة المختلفة لمحاولة حساب القيم الأساسية. في التشفير الحديث، قوة خوارزمية التشفير هي فقط في حجم المفتاح. لذلك، هدفنا هو إنشاء قيمة مفتاح قوية لها حد أدنى لطول البت والتي ستكون مفيدة في التشفير الخفيف. باستعمال تشفير منحنى إهليلجي (ECC) مع مكعب روبيك وكثافة الصورة، يتم مزج ألوان الصورة وتشويهها، وباستعمال خريطة الخدمات اللوجستية الفوضوية وكثافة الصورة بمفتاح سري، يتم تشفير مكعبات روبيك للصورة، والحصول على صورة آمنة ضد الهجمات. تعد ECC نفسها خوارزمية قوية تنشئ زوجًا من العناصر العامة والخاصة، ونقوم بإنشاء قيم مفاتيح سرية بمساعدة زوج المفاتيح أعلاه. تنطبق هذه الخوارزمية على البطاقات الذكية وأجهزة الاستشعار وأمان الشبكة اللاسلكية. حيث تشير نتائج الاختبار إلى أن

*Email: abdullah.adil@ihcoedu.uobaghdad.edu.iq

أعلى قيمة لـ PSNR هي 7.9641 بينما أعلى قيمة لـ UACI وNPCR هي 35.3721 و99.9992 على
التوالي. تصل سرعات التشفير وفك التشفير إلى 0.1561 و0.1532 ثانية على التوالي.

## 1. Introduction

One of the public-key cryptosystem algorithms that Koblitz and Miller presented is elliptic curve cryptography (ECC) [1]. Finding the discrete logarithm of a random elliptic curve element concerning a given base point is considered to be computationally infeasible for elliptic-curve-based protocols; this is known as the "Elliptic Curve Discrete Logarithm Problem" (ECDLP) [2]. Elliptic curve cryptography's security hinges on being able to compute a point multiplication but being unable to do so given the original and product points. ECC offers the same level of security as RSA but uses keys with fewer bit values, meaning faster performance and less computational complexity [1, 3]. The security level of a 160-bit ECC key is equivalent to that of a 1024-bit RSA key [4]. Text encryption is not the same as image encryption [5]. Traditional encryption algorithms such as Advanced Encryption Standard (AES) and others are not suitable for multimedia files due to their enormous data capacity, significant pixel correlation, and high redundancy [6].

In 1963, Edward Lorenz became the first person to apply chaos theory to a computer system. Because of the unauthorized person's noise-like signal, ergodicity, mixing, and sensitivity to the initial conditions, chaos-based cryptography has gotten a lot of attention in the past decade. These properties may be linked to those of excellent ciphers, such as confusion and diffusion [5]. Many picture encryption techniques based on chaotic systems have been presented in the field of information security research [7]. Many picture encryption methods have been developed based on chaotic maps, such as the logistic map. Cryptoanalytic attacks are significantly less effective on higher-dimensional chaotic functions [8, 9]. The main problem that may face the process of sending data such as images, texts, and other types of data is that it is exposed to many strong attacks, so we need to protect and preserve data by encrypting it. This paper presents an effective and robust method for encoding color images based on the use of density in elliptic curves with a Rubik's cube to blend some of the RGB image channels with them and by using 5D logistic chaos with image density to generate the main and encoded images using the XOR process.

The remaining sections of the paper are organized as follows: Section 2 focuses on related work. Section 3 explains the elliptic curve technique. Section 4 explains the proposed system. Section 5 presents the results and discussion. Section 6 presents the National Institute of Standards and Technology (NIST) randomness test. Section 7 gives the conclusion.

## 2. Related Work

The researchers focus on chaotic systems and elliptic curves by applying different algorithms, including the Rubik's cube, for image encoding and decoding operations. **D. S. Laiphrakpam** [10] proposed an image encryption scheme based on a chaotic system and an elliptic curve over a finite field. Using the Diffie-Hellman public key sharing method, the sender and receiver settle on an elliptic curve point. The logistic map is used to build a chaotic sequence with starting conditions determined by the common elliptic curve point. The shared elliptic curve point is used to multiply the points in the chaotic sequence once it has been converted to integers. To create a random sequence, the resultant elliptic curve points are translated to byte values. Arnold’s transform is used to scramble the encrypted picture, with the shared elliptic curve point determining the number of scrambling rounds. The scrambled

image’s pixel value is XORed with the random sequence to generate the cipher image. The greatest entropy value was 7.9997, while it takes 0.36 seconds to encrypt an image.
**J. Wu et al**. [11] proposed an asymmetric picture encryption technique with the benefit of having extremely tiny key groups and key numbers, as well as a relatively straightforward and safe key transmission method. The approach first compresses the plain picture, then uses the enhanced 4D cat map to encrypt the color image, then performs asymmetric encryption based on elliptic curve ElGamal encryption, and finally diffuses the encrypted image worldwide. The test results indicate Entropy has the highest value of 7.991596, while the highest values of the unified averaged changed intensity (UACI) and the number of changing pixels rate (NPCR) are 33.47% and 100%, respectively.

**J. Wu** [12] For handling color photographs utilized in the medical and forensic fields, a multilayered hybrid model known as the Hybrid multilayered hyper-chaotic hyper-elliptic curve-based image encryption (HHH) system has been presented. In order to produce chaotic sequences, hyper-chaotic-based DNA encoding is implemented in the top layer. Substitution permutation increases the amount of confusion spread to strengthen the cryptosystem's security.Genus-2 hyperelliptic curve cryptography over GF(p), which stresses the spatial domain for encryption, has been used due to its mathematical power in the second layer. As a hybrid system that incorporates symmetric and asymmetric cryptosystems, the proposed HHH system provides an NPCR of 99.71% and a UACI of 33.36%, while the entropy value is 7.9995.

**Oravec et al.** [13] proposed a plaintext-related picture encryption technique that changes the parameter values of the logistic map in response to pixel intensities in the plain image. Row by row, the parameter values are changed, allowing the same encryption and decryption techniques to be used. The parameter modification technique takes into consideration prior knowledge of the logistic map, its fixed points, and any periodic cycles. Because the resultant interval of parameter values contains large positive Lyapunov exponents, the logistic map's chaotic character should be clear. The test results indicate the highest values of UACI and NPCR are 33.4857% and 99.6143%, respectively.

**R. Vidhya** [14] To safely transmit multimedia information (images) through an untrusted channel, a Rubik's cube-based pixel-level scrambling method and a straightforward XOR-based diffusion method are proposed. Initial random value generation is also used to achieve high plain image sensitivity and fend off attacks related to plain images. The beginning vectors of the Henon map are derived from this random value, and this procedure is repeated to derive the key sequences to be used across the Rubik's cube row and column confusion processes. The key generation technique based on prime factorization to be used in diffusion also uses the same random seed. The random list is dynamically altered for each encryption of various plain pictures, and it is demonstrated that the CIERPF approach is safe against differential attacks. The test results indicate the highest values of UACI and NPCR being 33.4670% and 99.6261%, respectively, while the entropy value is 7.9995.

## 3. Research Methods

This section explains the methods used in the proposed system as follows:

### 3.1. Elliptic Curve Technique

Elliptic Curve Cryptosystem (ECC) is a practical encryption method that may be employed in, for instance, embedded systems and mobile devices since it can provide high security with a small key size, fewer calculations, less memory utilization, and reduced power consumption [15].

▪ Definition: Ep (a, b): $y^2=x^3+ax+b$ mod p defines an elliptic curve over a prime field Fp where p>3, a, b ∈ Fp and meet the condition $4a^3+27b^2$ mod p ≢ 0. The point at infinity O∞ and all points (x, y) that satisfy the elliptic curve Ep (a, b) make up the elliptic curve group E(Ep) [15].

▪ Elliptic curve operations: The elliptic curve scalar multiplication is the major elliptic curve operation on the elliptic curve that takes the most time during encryption and decryption processes. Calculating the elliptic curve scalar multiplication requires two operations: point addition and point doubling [15].

▪ Point addition: Suppose $P1= (x1, y1)$ and $P2= (x2, y2)$, where $P1\neq P2$, are two points lie on an elliptic curve $Ep\ (a, b)$. Adding the two points $P1$ and $P2$ giving a third point $P3= (x3, y3)$, as $x3\equiv(s2-x1-x2)\ mod\ p$, $y3\equiv(s(x1-x3)\ -y1))\ mod\ p$ and $s=y2-y1x2-x1\ mod\ p$. $P3$ should lie on the same curve $Ep\ (a, b)$.

▪ Point doubling: Suppose $P= (x1, y1)$ is a point on an elliptic curve $Ep\ (a, b)$, the point $R$=2 $P= (x2, y2)$ that results from doubling the point $P$ as $x2\equiv(s2-2x1)\ mod\ p$, $y2\equiv(s(x1-x2)\ -y1))\ mod\ p$ and $s\ =3x12+a2y1\ mod\ p$. $R$ is also a point on an elliptic curve $Ep\ (a, b)$ [1][15].

▪ Scalar multiplication on an elliptic curve: Let P be any point on the elliptic curve Ep (a, b). The repeated addition kP=P+P++Pk times defines the scalar multiplication operation over the elliptic curve P [1][15].

### 3.2. Magic cube (Rubik's cube)

The Rubik's cube was created in 1974 by Hungarian artist and lecturer Ern Rubik. The magic cube comprises three layers, six faces of various colors, and nine cells of the same color on each face [16].

$$B_{size} = (3 * 3)_{paces} * 6_{\text{faces}} = 54 \tag{1}$$

The magic cube can rotate in a variety of ways, with one potential rotation type being (NR). Typically, there are eighteen possible rotations for 3 × 3 × 3 cubes [17].

$$\text{NR} = (3)_{layers} * 6_{\text{faces}} = 18 \tag{2}$$

Variations on the original 3 × 3 × 3 Rubik's Cube are equivalent to being $8! \times 3^8 \times (12! \times 2) \times 2^{11}$ this is close to 43 quintillion. The Rubik's cube is more challenging to solve since there are so many alternatives. As a result, it was intended to use this complexity in the recommended encryption method, which may make it harder for hackers to decipher the original message [16]. It has layers like front (F), front-inverse (F'), back (B), back-inverse (B'), upper (UP), upper inverse (UP'), down (D), down inverse (D'), left (L), left-inverse (L'), right ®, right inverse'(R'), middle (M), middle inverse'(M'), horizontal (H), horizontal-inverse'(H'), and vertical (V), vertical-inverse'(V'). The Rubik's cube can be rotated [17, 18], as shown in Figure 1.

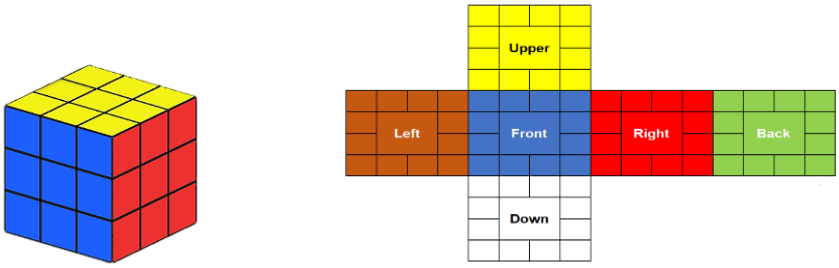

**Figure 1:** The Rubik's cube and its unfolding structure based on face values

## 4. Proposed Method

The main objective of this procedure is to secure the images using a new lightweight coding technique based on building two Rubik's cubes. The rows and columns of each cube are swapped according to the values of the points of the elliptical curve, whose point generation process depends on the image density, thus obtaining a mixture of pixels for the image to be created. Then the layers of each cube are encrypted based on a five-dimensional chaotic logistics map and a master encryption key, which overcomes the weakness of the logistics security, as shown in Figure 2.

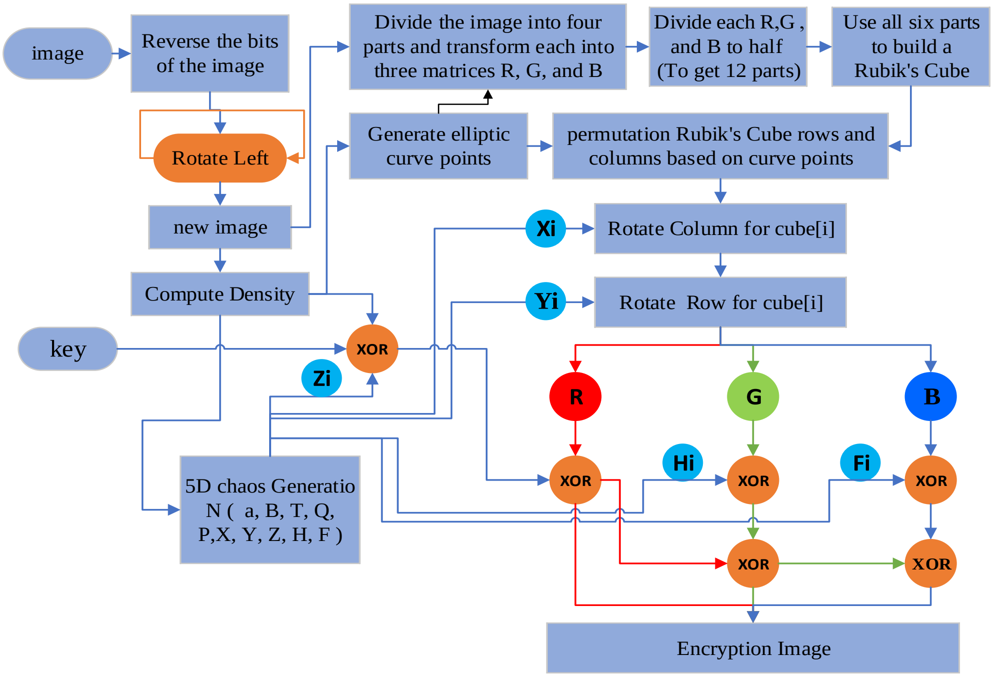
image
Reverse the bits of the image
Divide the image into four parts and transform each into three matrices R, G, and B
Divide each R,G , and B to half (To get 12 parts)
Use all six parts to build a Rubik's Cube
Rotate Left
Generate elliptic curve points
permutation Rubik's Cube rows and columns based on curve points
new image
Xi
Rotate Column for cube[i]
Compute Density
Yi
Rotate Row for cube[i]
key
XOR
Zi
R
G
B
Hi
Fi
5D chaos Generatio N ( a, B, T, Q, P,X, Y, Z, H, F )
XOR
XOR
XOR
XOR
XOR
Encryption Image

**Figure 2**: A block diagram illustrating the steps of the proposed method.

## 4.1. Encryption Image Stage

The encryption stage is divided into eight steps, which are as follows:

### Step 1: Image Density Calculation

The following equation was used to compute the density of the original image [19]:

$$density = \frac{\sum_{x=0}^{N}\sum_{y=0}^{M} image(x,y)}{N*M} \quad (3)$$

where N and M represent the width and height of the image, while image (x, y) represents the image pixel value for points x and y.

### Step 2: Reverse Image Bits and Rotate Left

We first use an algorithm to invert the image bits and then rotate them to the left in one-bit increments to improve the method's security, which makes the algorithm a good anti-noise attack and has the capacity to minimize assaults. This step is shown as an example: we have the value 197 as it is converted to the binary value 1100 0101, the bits are inverted to get 1010 0011, and after doing a left rotation, the value becomes 0100 0111 and returns to the integer

71. Figure 3 displays the results of the first step of the process of inverting and rotating the pixels in the image so that the image becomes semi-encrypted, and this increases the encryption and security of the image against attacks.

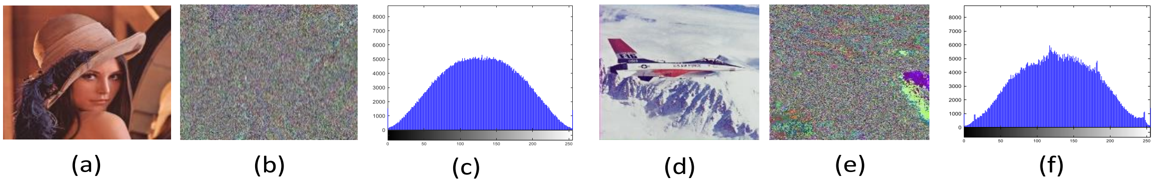
(a) (b) (c) (d) (e) (f)

**Figure -3** Histogram analyses of different images:(a, d) original image, (b, e): inverting and rotating, (c, f): histogram of image (b, e).

### Step 3: Rubik's cube building

In this section, the steps of the proposed mechanism for creating two cubes from the original image are described as follows:

1- Calculate the image size where $size\ = M * N$

where $M\ =$ image width; $N =$ image height.

2- Divide the image size into four parts and create two cubes of the resulting volume, where each cube consists of six layers (upper, lower, front, back, right, and left).

3- The layers of the cube are filled with image pixels; the image is divided into four parts, and each part is divided into three channels: R, G, and B. The first cube is filled from the first and fourth parts of the image; the second cube is filled from the second and third parts of the image; the upper and lower layers are filled with red; the right and left layers are filled with green; and the front and back layers are filled with blue for each cube, as shown in Figure 4.

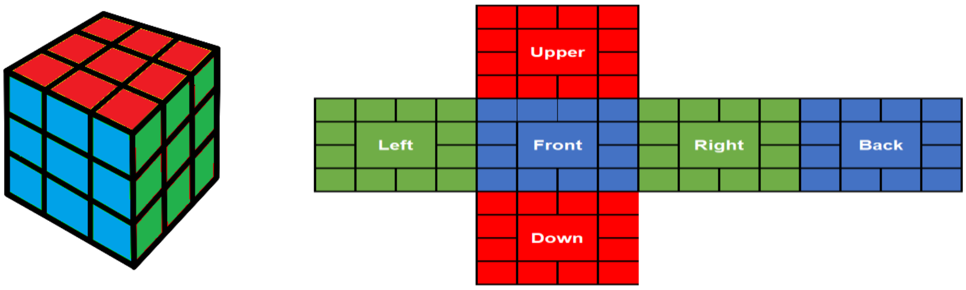

**Figure 4:** Rubik’s cube and color image distribution

4- If there is a data shortage when the image is divided into four parts, this deficiency will be compensated by a value of zero. This is done by rechecking if the $Mo =$ size is mod 4! = 0, it will add zero values according to the return $Na =$ size – Mo

5- where $Mo\ =$ The remainder of the value; $Na =$ The number of zero to be added.

### Step 4: Generate elliptic curve points

In this part, the points of the ellipsoidal curve are generated using multiplication, which depends on addition and repetition. Curve points are generated depending on the density of the image, and by increasing it and applying the multiplication process, all the required points are created in less time, and these points change dynamically for each image because they depend on image density.

**Algorithm 1. Elliptical Curve point generation**

**INPUT:** Elliptical curve is in the form of $y^2 = x^3 + ax + b \bmod p$; density, G (point in EC)

**OUTPUT:** Elliptical Curve point

**Step:**

```
x = 0
while x < p do
    curve(x) = Scalar_multiplication (Density, G)
    density = density + 1; x = x + 1
End of while loop
```

**Step 5: permutation Rubik's Cube rows and columns based on elliptic curve points**

In this section, the rows and columns of each cube are rotated based on the elliptical curve point values, and the cube's rotation angle is between 0 and 360 degrees according to the curve point values. The direction of rotation is either clockwise or counterclockwiseshows the images after rotating the first and second cubes according to the points of the elliptic curve.

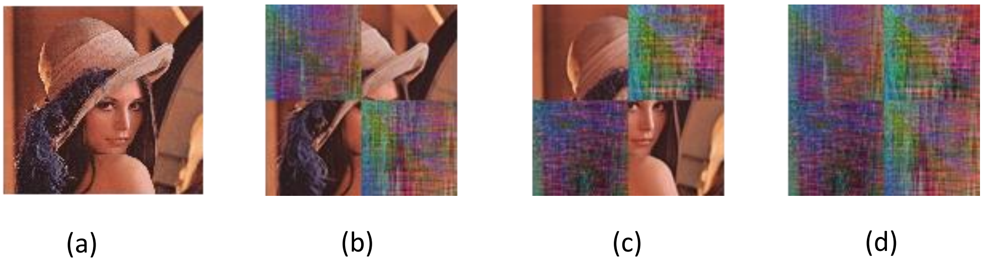

**Figure 5:** Rubik's cube permutation (a) original image (b) permutated first cube (c) permutated second cube (d) after permutated first and second cube.

Steps 2 and 5 are used, as well as the process of switching the layers of the two cubes between them, where the upper layer of the first cube is replaced with the upper layer of the second cube, the front layer of the first cube is replaced with the down layer of the second cube, the left layers of the first cube are replaced with the back layer of the second cube, and finally, the lower layer of the first cube is replaced with the front layer of the second cube.

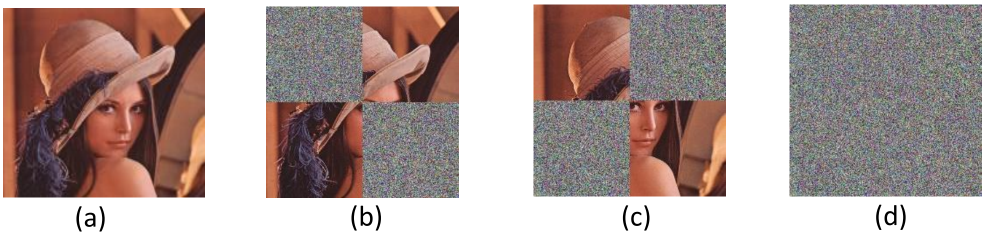

**Figure 6 :** After inverting and rotating the bits and switching the Rubik's cube. (a) the original image (b) the first cube (c) toggled the second cube (d) after switching between the first and second cubes.

**Step 6: 5D Logistic Chaotic Map**

The logistic map is the simplest process of chaos generation given by Eq. (4) [20].

$$x_{i+1} = r\, x_i(1 - x_i) \tag{4}$$

The requirements to make this equation chaotic are $0 < x_i < 1$ and $r = 4$ [20]. Hongjuan Liu. et al. suggest a 2D logistics map using quadratic coupling to enhance security [21].

Hossain, M. B. et al. suggest a 3D logistics map using quadratic-cubic coupling for added safety [8]. The extended 5D version is suggested by the following formula:

$$x_{i+1} = ax_i(1-x_i) + Bf_i^2x_i + Yh_i^3 + Qz_i^4x_i + Py_i^5 \quad (5)$$

$$y_{i+1} = ay_i(1-y_i) + Bx_i^2y_i + Yf_i^3 + Qh_i^4y_i + Pz_i^5$$

$$z_{i+1} = az_i(1-z_i) + By_i^2z_i + Yx_i^3 + Qf_i^4z_i + Ph_i^5 \quad (6)$$

$$h_{i+1} = ah_i(1-h_i) + Bz_i^2h_i + Yy_i^3 + Qx_i^4h_i + Pf_i^5$$

$$f_{i+1} = af_i(1-f_i) + Bh_i^2f_i + Yz_i^3 + Qy_i^4f_i + Px_i^5 \quad (7)$$

The chaotic behavior in the preceding equations may be seen here in 3.57<a<4, 0<B<0. 17* 10-11, 0<Y< 0. 085* 10-11, 0<Q< 0.069* 10-11, 0<P< 0. 032* 10-11, and the starting value of x, y, z, h, f, and k between 0 and 1. The presence of hexagonal quadratic coupling and six constant terms complicates and secures the 5D logistic map. Figure 7 shows the chaotic sequences formed using Equations 5, 6, 7, 8, and 9 and a starting value of x (1) = 0.07; y (1) = 0.09; z (1) = 0.02; h (1) = 0.03; f (1) = 0.04; a = 4; B = 1 / Density + 0.128*10-12; Y = 0. 1253* 10-11; Q = 0. 1573 * 10-11; P = 0. 837* 10-12, and the number of iterations is 100.

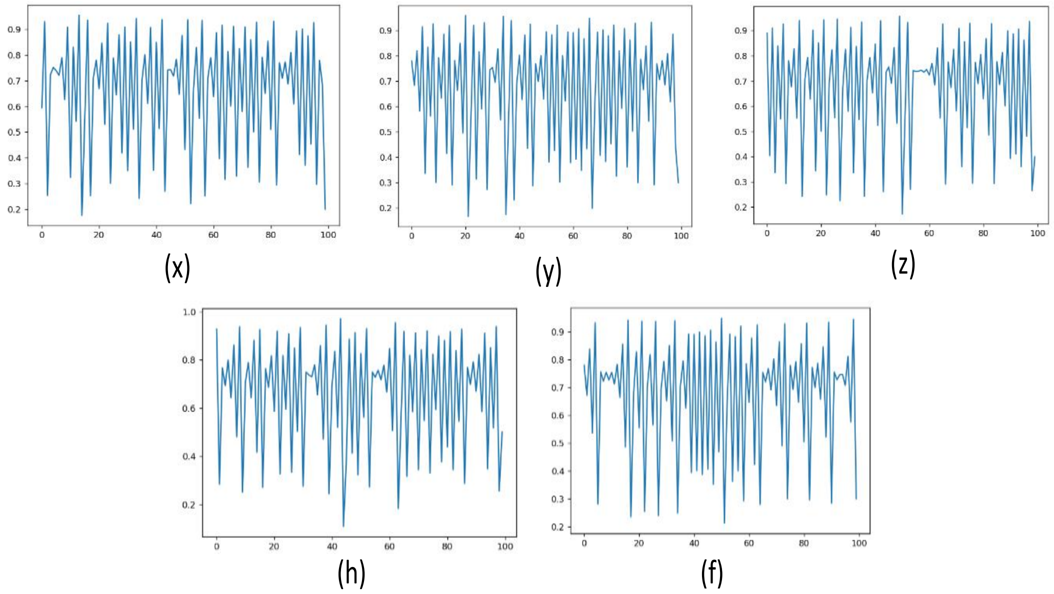

**Figure 7:** 5D Logistic Chaotic Map

**Step 7: Column Rotation and Row Rotation**

The image pixel values are rotated depending on the chaos, according to the chaos x value of Eq (5), to enhance security by switching the image pixel values. We introduce a new way to rotate values based on the sequence of images and clutter X columns. We go left when clutter is even and right if clutter is odd. Row rotation is the same as column rotation in that the image pixel values are rotated based on the chaos, according to the chaos y value of Eq (6), and security is further enhanced by switching the image pixel values and rotating the values based on the sequence of rows of images and the chaos y.

**Step 8: XOR operation**

The XOR technique is the final step in the encryption process. The XOR operation changes the value of a pixel to a new value and cannot be reversed without knowing the chaos key [22]. In the beginning, three of the chaos values Z, H, and F are generated, and the image is divided into three matrices: R, G, and B. An XOR operation was performed between the chaos value Z, the master key, and the image density, and then the XOR operation was performed with the R matrix. An XOR operation was performed between the values of the chaos matrix H and the G matrix, and the resulting value was XORed with the value of the R matrix. Finally, perform an XOR operation between the chaos value of F and the B matrix, and XOR the resulting value with the G matrix value. By repeating these operations on all the pixels of the image, we get encrypted images. Algorithm 2 explains the steps of the proposed system.

**Algorithm 2. Encryption Algorithm**

```
Input: image, Initial (a, B, Y, Q, P, x, y, z, h, f), key, EC (a, b, p, G)
Output: image encryption
Step 1: Density Calculation
    Density = Get_Density(image)
Step 2: Reverse Image Bits and Rotate Left
   rImg = RotateLeft_Reverse(image)
Step 3: Rubik's cube building
 Size = M * N;
 Cubes = Cube_building (rImg, size)
Step 4: Generate elliptic curve points
Curve = Generate_EC (a, b, p, G, Density)
Step 5: Permutation Rubik's Cube based on elliptic curve points
Cube_Rotate_X_Y (cubes, Curve.X, Curve.Y)
SwapLayers(cubes)

Step 6: Generate Logistics Chaos Keys
  Logis5D = Generate (a, B, Y, Q, P, x, y, z, h, f, Density)
Step 7: Column and Row Rotation
Column_RowRotation (cubes, Logis5D.x, Logis5D.y)
Step 8: XOR Operation
For all x, y Do {where 0 ≤ x ≤ Size, 0 ≤ y ≤ Size, 0 ≤ t ≤ 1}
cubes [t].UPP[x][y]  = cubes [t].UPP[x][y] ⊕ Mod (Logis5D.z⊕[i]      key        Density,256)
cubes [t].RGT[x][y] = cubes [t].RGT[x][y]⊕    Mod(Logis5D⊕h[i]        cubes [t].UPP[x][y]
,256)
cubes [t].FOT[x][y]   = cubes [t].FOT[x][y⊕      Mod(Logis5⊕D.f[i]       cubes [t].UPP[x][y]
,256)
 I =  i + 1
cubes [t]. DWN [x][y] = cubes [t]. DWN [x⊕][y]          Mod (⊕Logis5⊕D.z[i]            key
Density,256)
cubes [t]. LFT [x][y] = cubes [t]. LFT [x][y]⊕  Mod(Logis5D.h⊕[i]     cubes [t]. DWN [x][y]
,256)
cubes [t]. FOT [x][y] = cubes [t]. FOT [x][y]⊕       Mod(Logis5⊕D.f[i]        cubes [t]. LFT
[x][y],256)
                                                                        End For// Size, Size
```

### 4.2. Decryption Image Stage

The decoding process is the opposite of the encoding process, in which the key value is entered together with the image density value, In the same way, two cubes are created from the encrypted image, and the encryption steps are applied inversely to obtain the decoded image. If the image contains padding equal to zero, this padding is removed.

$$MSE = \frac{1}{M\,x\,N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} [I_1(i,j) - C_2\,(i,j)]^2 \qquad (10)$$

## 5. Results and Discussion

Experiments with the suggested approach were carried out on a machine with a 3.3 GHz CPU and 16 GB of RAM, which was running GO Language at Windows 11 Home, 64-bit. Figure 8 displays the tested images utilized in experiments. All of these images have a 512 × 512-pixel resolution. Color depths of up to 24 bits per pixel are used in the Lena and Airplane images. The proposed system is tested for ten images from the dataset (i.e., all dataset images), but this work only shows the results of two images as described in Figure 8.

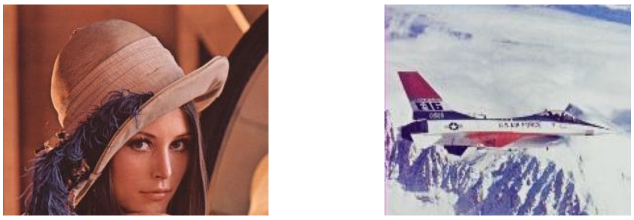

**Figure 8:** A set of experimental images.

The following criteria are utilized to measure the performance of the presented cryptosystem:

- **Histogram**

The image histogram displays the number of pixels (along the y-axis) at each intensity level (along the x-axis) to demonstrate how the pixels are distributed in the image [1]. A successful image coding system should provide all encoded images with a uniform image outline, regardless of the original planar image structure [9, 23].

- **MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ration)**

The difference between the input and output images in terms of pixel intensity levels is measured using MSE. The MSE value is higher for an ideal fully encrypted cipher image [12][24]. The MSE is computed using Eq. (10).

.

Where M and N are the width and height of images, $I_1$ is the original image, and $C_2$ is the cipher image. PSNR analysis The Peak Signal-to-Noise Ratio (PSNR) reflects the encryption quality. The lower the PSNR value, the better the encryption quality [11] [10].

$$PSNR = 20 \text{ x } \log_{10}\left(\frac{255}{Sqrt\,(MSE)}\right). \qquad (11)$$

- **NPCR (Number of Pixel Change Rate) and UACI (Unified Average Changing Intensity)**

The NPCR value tells the rate of change of a number of pixels in an encrypted image when a pixel of the original image is changed [13, 14]. The higher the value, the better the encryption [12], with Eq. (12) defining the NPCR of an image.

$$D(i,j) = \begin{cases} 1, & C_1\,(i,j) \neq C_2(i,j) \\ 0, & C_1\,(i,j) = C_2(i,j) \end{cases}$$

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M*N} * 100\%. \qquad (12)$$

The UACI value calculates the average intensity of the differences between the original image and the encrypted image. The higher the value, the better the encryption [11, 12]. The UACI of an image can be defined as Eq (13).

$$UACI = \frac{1}{M*N}\left[\sum_{i,j}\frac{|C_1(i,j) - C_2(i,j)|}{15}\right] * 100\ \%. \qquad (13)$$

where C1 and C2 are two cipher images encrypted without a change in the original image and with a slight change in the original image, respectively.

- **Entropy Analysis**

The entropy (H) of a symbol source (S) can be calculated by following Eq. (14) [3].

$$H(s) = -\sum_{i=0}^{M-1} P(s_i)\log_2\frac{1}{P(s_i)}. \qquad (14)$$

Where Si is the chance that a pixel will appear in an image and N is the length of a pixel's binary number. The cryptosystem's ability to withstand entropy assaults is one of its most essential features; the optimal entropy value of encrypted pictures is 8 bits/pixel[9].

- **Correlation Coefficient Analysis**

The correlation coefficient of a visible image with correct brightness is one, but it is much lower for a ciphered image (almost equal to zero). An encryption technique produces ciphered images with randomly dispersed pixels of various brightness and a low correlation coefficient between neighboring pixels [13]. The correlation coefficient of any two-pixel color value at the same place in the original and cipher pictures is calculated using Eq. (15) [25].

$$Corr(x,y) = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma_x\,\sigma_y}. \qquad (15)$$

Where $\mu x$ and $\mu y$ represent the mean values of x and y, $\sigma x$ and $\sigma y$ are the standard deviations of x and y, and $E$ [·] is the expectation function.

## 5.1. Results of image encryption and decryption

A picture encryption solution should be resistant to all known forms of attack and should not be dependent on the plaintext or encryption key. If the encryption key is to be utilized consistently, a suitable picture encryption algorithm should be able to encrypt any plain text image into a randomly generated ciphertext [9]. The suggested picture encryption approach is put to the test in this part by utilizing a six-dimensional logistics map and security analysis. Figure 9 shows a histogram analysis of several images, including encoded and non-encoded images.

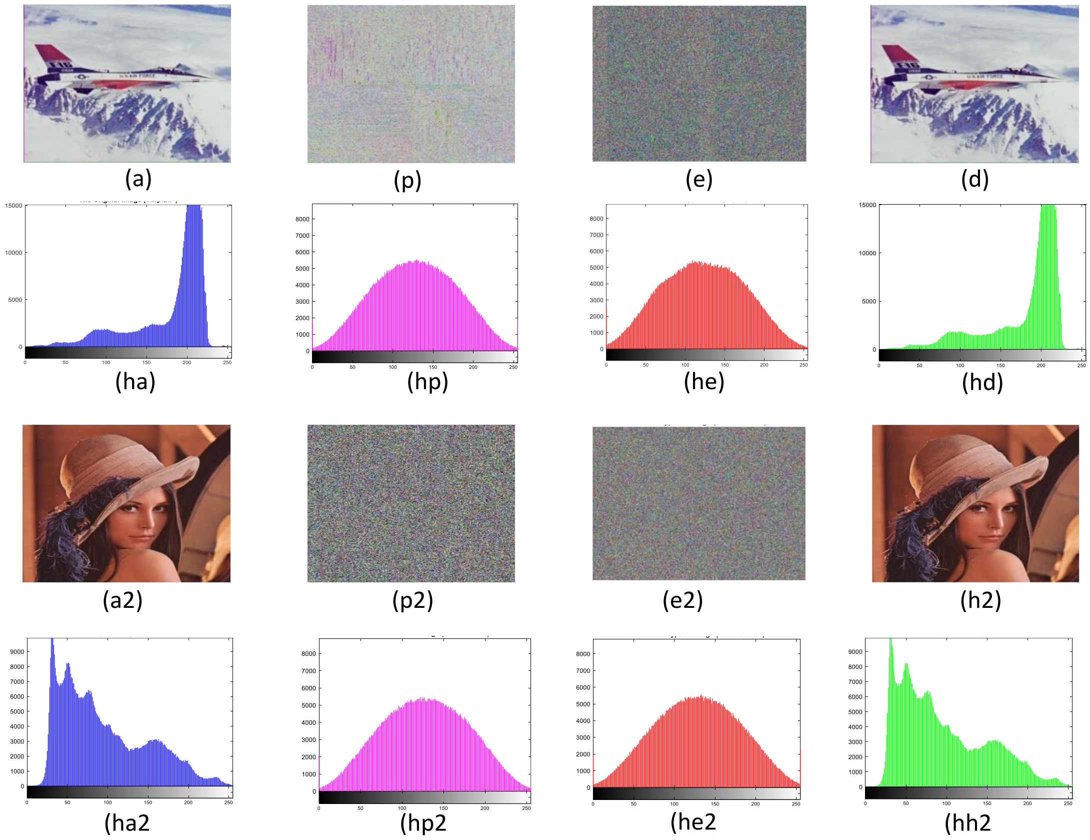

**Figure 9**: Histogram analysis of different images and their encrypted images (horizontally), (eg, (a) original image, (ha) its histogram, (p) images after inverting and rotating the bits, (hp) its histogram, (e) encrypted (p), (he) histogram of the encrypted image, (d) decrypted (e), (hd) histogram of the decrypted image.

### 5.2. MSE and PSNR analysis

Table 1 shows the MSE and PSNR values of the original and encoded images of Airplane and Lena. The highest value of MSE indicates a better result in encryption, while the lower value of the PSNR indicates better encryption quality. MSE and PSNR values are equal to zero and ∞ because not distortion any data in the decryption image.

**Table 1:** MSE and PSNR values for plain and cipher images of Airplane, and Lena.

| Image Name | Encrypt | | Decrypt | |
|---|---|---|---|---|
| | **MSE** | **PSNR** | **MSE** | **PSNR** |
| **Airplane** | 10391.3008 | 7.9641 | 0 | ∞ |
| **Lena** | 9451.3217 | 8.3759 | 0 | ∞ |

### 5.3. Correlation analysis and entropy analysis

The proposed algorithm is analyzed using the two test images. The encrypted images are obtained by applying five levels of encryption to the input image, which include reverse bits, rotate Left, 5D chaos generation, pixel permutation, column rotation, row rotation, and XOR logical operation. Table 2 shows the results of the entropy and correlation of the two test images encrypted in the proposed system.

**Table 2:** shows the results of the entropy and correlation test of the encrypted images

| Image Name | | Entropy | | | | Correlation | | |
|---|---|---|---|---|---|---|---|---|
| | | **Plain** | **Encrypt** | **Decrypt** | | **Horizontal** | **Vertical** | **Diagonal** |
| **Airplane** | | 6.6963 | 7.7056 | 6.6963 | | -0.00080 | -0.00127 | -0.00302 |
| **Lena** | | 7.6037 | 7.7031 | 7.6037 | | -0.00126 | -0.00052 | -0.00266 |

Figures 10 and 11 illustrate the scatter plots of correlation for plain and cipher images for all of the tested images in the horizontal, vertical, and diagonal directions, respectively (i.e., Airplane and Lena).

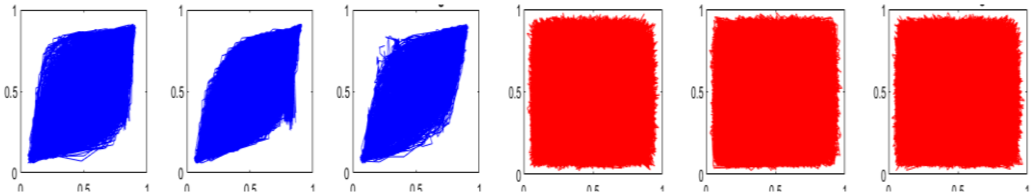

**Figure 10**: Horizontal, vertical, and diagonal correlation scatter plots for the plain Airplane image (a–c) and the encrypted Airplane image (d–f).

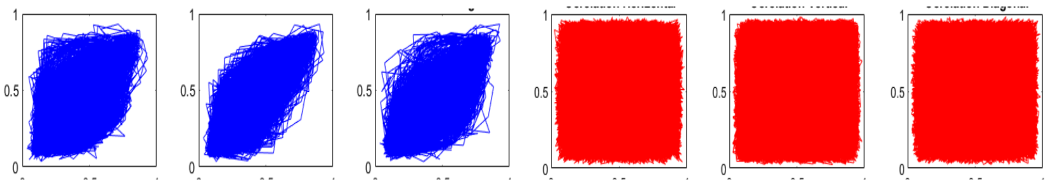

**Figure 11:** Horizontal, vertical, and diagonal correlation scatter plots for the plain Lena image (g–i)
and the encrypted Lena image (j–l).

### 5.4. Differential attacks analysis

The proposed system resulted in two encoded images: one by encoding it directly and the other by modifying one pixel of the original image and then encoding it. These images are necessary to calculate the NPCR and UACI values to evaluate the performance of the proposed system. NPCR and UACI values are listed in Table 3 for tested images. The highest value for NPCR is 99.9992 percent, which means that changing one pixel of the original image dramatically changes the encoded image up to 100% in some test images. The highest value for UACI is 35.3721 percent. Since the experimental values match the theoretical values, the proposed method is resistant to differential assaults.

**Table 3:** NPCR and UACI values of Panda, Lena

| Image Name | NPCR | UACI |
|---|---|---|
| **Airplane** | 99.9992 | **35.3721** |
| **Lena** | **99.9989** | **33.6584** |

### 5.5. Analysis of speed

The encryption time for a cryptosystem should be as short as feasible. The time taken to implement the suggested method for all test images and the external test image is shown in Table 4. Given the proposed scheme's high level of security, these running speeds are acceptable. Figure 12 shows the result of applying proposed system on the external test image of size (1024 * 1024).

**Table 4:** Encryption and Decryption speed test

| Image Name | Image size | Encryption Speed (second) | Decryption Speed (second) |
|---|---|---|---|
| **Airplane** | 512 * 512 | 0.1561 | 0.1532 |
| **Lena** | 512 * 512 | 0.1583 | 0.1607 |
| **External test** | 1024 * 1024 | 0.3248 | 0.3054 |

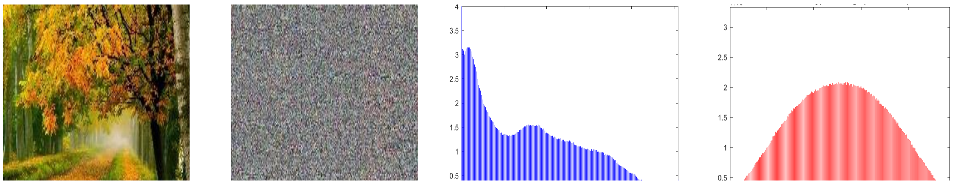

**Figure 12**: External test (a) original image, (e2) its histogram, (ha) encrypted (a), (he) histogram (ha).

### 5.6. Comparison

Table 5 compares the performance evaluation metrics attained by our proposed method with those given in previous studies and also shows that the proposed method has given better results than methods in the previous studies.

**Table 5:** Comparison of the proposed method for encryption images with the previous studies

| Measurements | Proposed | [11] | [10] | [12] | [13] | [14] |
|---|---|---|---|---|---|---|
| **PSNR** | 7.9641 | 27.5912 | 8.63193 | 8.65 | - | - |
| **NPCR** | 99.9992 | 100 | - | 99.71 | 99.6143 | 99. 6252 |
| **UACI** | 35.3721 | 33.47 | - | 33.36 | 33.4857 | 33. 5487 |
| **Entropy** | 7.7056 | 7.991226 | 7.9992 | 7.9995 | 7.9992 | 7.9995 |
| **Horizontal Correlation** | -0.00126 | -0.0001 | −0.0034 | −0.056 | −0.0019 | 0.00002 |
| **Vertical Correlation** | -0.00052 | 0.0089 | -0.0003 | −0.047 | 0.002 | 0.00045 |
| **Diagonal Correlation** | -0.00266 | 0.0091 | 0.008 | - | −0.0012 | 0.00081 |
| **Encryption speed (second)** | 0.1561 | - | 0.36 | - | 1.5211 | - |
| **Decryption speed (second)** | 0.1532 | - | - | - | 1.5371 | - |

## 6. NIST randomness test

In randomness tests, the NIST test is the most popular of all sequence randomness tests [26]. We run NIST tests using the random sequence of logistic map keys used for encryption to get the results. Table 6 shows the results of the NIST test and shows that all the generated encryption keys are random for the 16 items.

**Table 6:** NIST Testing for bit sequences that have a length 100000 bits.

| Type of Test | P-Value | Status | Type of Test | P-Value | Status |
|---|---|---|---|---|---|
| **Frequency test (Monobit)** | 0.9605226524 | Random | Maurer's Universal Statistical | 0.2487526006 | Random |
| **Frequency test within a Block** | 0.0290851864 | Random | Linear Complexity | 0.4604387067 | Random |
| **Runs test** | 0.9582660454 | Random | Serial test | 0.0407380230 | Random |
| **Longest Run of Ones in a Block** | 0.5973225710 | Random | Approximate Entropy | 0.6553999692 | Random |
| **Binary Matrix Rank** | 0.5608014420 | Random | Cumulative Sums (Forward) | 0.9085205277 | Random |
| **Discrete Fourier Transform (Spectral)** | 0.6531751671 | Random | Cumulative Sums (Reverse) | 0.9144078012 | Random |
| **Non-Overlapping Template Matching** | 0.5214836922 | Random | Random Excursions (+1) | 0.4939534908 | Random |
| **Overlapping Template Matching** | 0.1631537766 | Random | Random Excursions Variant (+1) | 0.3326976824 | Random |

## 7. Conclusion

In previous studies on light cipher using elliptic curves and logistic chaos, this topic has been extensively researched and dealt with. But this paper presents an effective and powerful method for encoding images based on mixing and swapping image colors using elliptic curves with image density in a Rubik's cube and using 5D logistic chaos with image density to generate master and encoded images using the XOR process. The proposed system has been tested on all types of images of different sizes, and from the results obtained in Section 5, the efficiency of the proposed system has been demonstrated. The experimental results showed that the highest value obtained when PSNR was 7.9641 and UACI and NPCR were 35.3721 and 99.9992, respectively, was according to the test data. Based on the test results, we can conclude that our algorithm has a good level of security and can effectively withstand a variety of attacks. All of these results demonstrate that our method competes with previously developed clutter-based image coding techniques.

## REFERENCES

**[1]** S. Banerjee and A. Patil, “ECC Based Encryption Algorithm for Lightweight Cryptography,” *Intell. Syst. Des. Appl.*, vol. 940, pp. 600–609, 2020, doi: 10.1007/978-3-030-16657-1_56.

**[2]** Z. K. Obaidand, N. Falah, and H. Al, “Image encryption based on elliptic curve cryptosystem,” vol. 11, no. 2, pp. 1293–1302, 2021, doi: 10.11591/ijece.v11i2.pp1293-1302.

**[3]** K. A. Hussein and T. B. Kareem, “Proposed Parallel Algorithms to Encryption Image Based on Hybrid Enhancement RC5 and RSA,” *2019 Int. Eng. Conf.*, pp. 101–106, 2019.

**[4]** B. Jasra, M. Saqib, and A. H. Moon, “Image Encryption Using Logistic-Cosine-Sine Chaos Map And Elliptic Curve Cryptography,” *J. Theor. Appl. Inf. Technol.*, vol. 99, no. 16, 2021.

**[5]** S. Anwar and S. Meghana, “A pixel permutation based image encryption technique using chaotic map,” *Multimed. Tools Appl.*, vol. 78, no. 19, pp. 27569–27590, 2019, doi: 10.1007/s11042-019-07852-2.

**[6]** M. François *et al.*, “A Pseudo-Random Bit Generator Using Three Chaotic Logistic Maps,” 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00785380/document

**[7]** S. Patel, K. P. Bharath, and R. K. M, “Symmetric keys image encryption and decryption using 3D chaotic maps with DNA encoding technique,” *Multimed. Tools Appl.*, vol. 79, no. 43, pp. 31739–31757, 2020, doi: 10.1007/s11042-020-09551-9.

**[8]** B. Hossain, T. Rahman, A. B. M. S. Rahman, and S. Islam, “A New Approach of Image Encryption Using 3D Chaotic Map to Enhance Security of Multimedia Component,” *2014 Int. Conf. Informatics, Electron. Vis.*, pp. 1–6, 2014, doi: 10.1109/ICIEV.2014.6850856.

**[9]** J. P. Noonan and J. P. Noonan, “Image encryption using the two- dimensional logistic chaotic map,” *J. Electron. Imaging*, vol. 21, no. 1, p. 013014, 2012, doi: 10.1117/1.JEI.21.1.013014.

**[10]** D. S. Laiphrakpam and M. S. Khumanthem, “A robust image encryption scheme based on chaotic system and elliptic curve over finite field,” *Multimed. Tools Appl.*, vol. 77, pp. 8629–8652, 2018, doi: 10.1007/s11042-017-4755-1.

**[11]** J. Wu, L. Xiaofeng, and B. Yang, “Color image encryption based on chaotic systems and elliptic curve ElGamal scheme,” *Signal Processing*, vol. 141, pp. 109–124, 2017, doi: 10.1016/j.sigpro.2017.04.006.

**[12]** N. Sasikaladevi, K. Geetha, K. Sriharshini, and M. D. Aruna, “H 3 -hybrid multilayered hyper chaotic hyper elliptic curve based image encryption system,” *Opt. Laser Technol.*, vol. 127, no. January, p. 106173, 2020, doi: 10.1016/j.optlastec.2020.106173.

**[13]** P. P. Values, J. Oravec, and L. Ovsenik, “An Image Encryption Algorithm Using Logistic Map with Plaintext-Related Parameter Values,” *Entropy*, vol. 23, no. 11, p. 1373, 2021, doi: 10.3390/e23111373.

**[14]** R. Vidhya and M. Brindha, “A chaos based image encryption algorithm using Rubik ’ s cube and prime factorization process ( CIERPF ),” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 5, pp. 2000–2016, 2022, doi: 10.1016/j.jksuci.2019.12.014.

**[15]** D. Vamsi, “Hybrid Image Encryption Using Elliptic Curve Cryptography , Hadamard Transform and Hill Cipher,” *Webology*, vol. 19, no. 1, pp. 2357–2378, 2022, doi: 10.14704/WEB/V19I1/WEB19160.

**[16]** R. M. Zaki and H. Bahjat, “A novel of substitution-box design using PLL algorithms in magic cube,” *Period. Eng. Nat. Sci.*, vol. 9, no. 4, pp. 674–684, 2021, doi: 10.21533/pen.v9i4.2402.

**[17]** R. Dhandabani, S. S. Periyasamy, P. Theagarajan, and A. K. Sangaiah, “Six-face cubical key encryption and decryption based on product cipher using hybridisation and Rubik ’ s cubes,” *IET Networks*, vol. 7, no. 5, pp. 313–320, 2018, doi: 10.1049/iet-net.2017.0196.

**[18]** H. Stitz, “Visualization of Rubik ’ s Cube Solution Algorithms,” *InEuroVA@ EuroVis*, pp. 19–23, 2019, doi: 10.2312/eurova.20191119.

**[19]** M. A. Rajab and L. E. George, “An Efficient Method for Stamps Recognition Using Histogram Moment with Haar Wavelet Sub-bands,” *Iraqi J. Sci.*, vol. 62, no. 9, pp. 3182–3195, 2021, doi: 10.24996/ijs.2021.62.9.32.

**[20]** H. Xiang and L. Liu, “An improved digital logistic map and its application in image encryption,” *Multimed. Tools Appl.*, vol. 79, no. 41, pp. 30329–30355, 2020, doi: 10.1007/s11042-020-09595-x An.

**[21]** H. Liu, Z. Zhu, H. Jiang, and B. Wang, “A Novel Image Encryption Algorithm Based on Improved 3D Chaotic Cat Map,” *2008 9th Int. Conf. Young Comput. Sci.*, pp. 3016–3021, 2008, doi: 10.1109/ICYCS.2008.449.

**[22]** 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, 2013, doi: 10.14257/ijsip.2013.6.5.25.

**[23]** S. A. Mahmood, K. A. Hussein, Y. N. Jurn, and E. A. Albahrani, “Parallelizable cipher of color image based on two-dimensional chaotic system,” *Indonesian Journal of Electrical Engineering and Computer Science,* Vol. 14, no. 1, pp. 101-111, April 2020, doi: 10.11591/ijeecs.v18.i1.pp101-111.

**[24]** M. R. Salman, khalid A. Hussein, and A. K. Farhan, “Color Image Encryption Depend on DNA Operation and Chaotic System,” *2019 First Int. Conf. Comput. Appl. Sci.*, pp. 267–272, 2019, doi: 10.1109/CAS47993.2019.9075458.

**[25]** K. A. Hussein, “A New Permutation-Substitution Scheme Based on Henon Chaotic Map for Image Encryption,” *2019 2nd Sci. Conf. Comput. Sci.*, pp. 63–68, 2019.

**[26]** R. A. Elmanfaloty, “An Image Encryption Scheme Using a 1D Chaotic Double Section Skew Tent Map,” *Complexity*, vol. 2020, pp. 1–18, 2020, doi: 10.1155/2020/7647421.