







## Genetic Algorithm Using Feistel and Genetic Operator Acting at the Bit Level for Images Encryption



Abdellah Abid<sup>\*</sup>, Mariem Jarjar, Mourad Kattass, Hicham Rrghout, Abdellatif Jarjar,  
Abdelhamid Benazzi

MATSI Laboratory, Mohamed First University, Oujda 60000, Morocco

Corresponding Author Email: [a.abid@ump.ac.ma](mailto:a.abid@ump.ac.ma)

Copyright: ©2024 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/ijss.140102>

### ABSTRACT

**Received:** 23 November 2023

**Revised:** 26 January 2024

**Accepted:** 12 February 2024

**Available online:** 29 February 2024

#### Keywords:

*image encryption, chaotic map, genetic algorithm, fitness function, genetic operator, Feistel network, hamming distance*

In this paper, a new medical image encryption technique based on genetic algorithms acting at the bit level will be developed. Initially, a transformation to a binary matrix notation of the original image is applied, followed by an evaluation function determined by the Hamming distance between the obtained image and another pseudo-random image generated from chaotic maps used. This discrimination function divides the image, viewed as a population where each row represents an individual, into two categories: a strong population and a weak population. An enhanced Feistel round will be implemented by introducing a chaotic mating between the two categories based on a circular shift for the right bloc and a pseudo-random permutation for the left bloc. Next, a genetic crossover adapted for image encryption will be performed with another pseudo-random vector under the control of a crossover table. To ensure the robustness of our approach, a genetic mutation will be applied at the end of the encryption. A multitude of images of different sizes and formats have been tested using our approach, with encouraging results.

## 1. INTRODUCTION

In today's digital age, the security of sensitive data, especially medical images, is of paramount importance. Safeguarding the integrity and the confidentiality of images is crucial in a digital world where the exchange and the storage of visual information are commonplace. Images may contain private, confidential, or strategic data, and unauthorized exposure can lead to several consequences for individuals and organizations. To address this growing concern, numerous image encryption methods have been developed, yet some still exhibit vulnerabilities to sophisticated attacks.

With advancements in mathematics, cryptography swiftly made its way into the realm of security. The emergence of standards such as Hill [1-4], Vigenère [5-8], DES [9], Feistel [10-13], AES [14, 15], and others has been observed. The development of chaos theory and the availability of multiple chaotic maps facilitate the creation of encryption algorithms. Furthermore, by following Shannon's guidelines [16] and incorporating feedback in the encryption mode, a strong resilience is conferred upon the new cryptographic systems.

With the rapid advancement of chaos theory, we are witnessing several improvements in these classical techniques. Additionally, some of the currently highly recommended solutions are genetic algorithms. In fact, genetic algorithms are among the most frequently used optimization methods, with numerous applications in various scientific fields [17-20]. The genetic algorithm, first proposed by John Holland in 1975, is one of the most renowned evolutionary algorithms, inspired by the evolutionary process of living species. The genetic

algorithm is widely used to solve a variety of optimization problems, starting with the initial random generation of individuals, followed by selection for reproduction of the new tribes, according to the value of their evaluation function. In other words, the best-performing individuals have the opportunity to be selected for the reproduction of future generations. The selected individuals will be subjected to genetic operators such as mutation, crossover, insertion and reversion (Figure 1). With this in mind, most researchers have developed encryption methods based on adaptation of genetic algorithm to image encryption. Ghazvini et al. [21] proposed an image encryption method based on genetic and chaos algorithms. The method consists of three main steps: a confusion phase using the Chen map, a second phase performed by the logistic-sine map, and the final phase involves selecting the encrypted image that yields the highest entropy and the lowest correlation. Mahmud et al. [22] had developed a novel cryptosystem based on the hybridization of Ribonucleic Acid (RNA) and Genetic Algorithm (GA). Initially, the logistic map and RNA codons are used to create initial populations for GA. Finally, GA optimizes the images using the fitness function based on entropy to find the optimized encrypted image. Ferdush et al. [23] proposed a new hybrid encryption system using PSO and GA with a chaotic map. An image is initially encrypted using a chaotic function, then PSO and GA are applied to select the best encrypted image that exhibits the highest entropy and the lowest correlation coefficient between adjacent pixels.

Our contribution in this work is to develop a robust encryption system based on genetic algorithms using chaos

and an enhanced Feistel scheme employing a circular shift for the right bloc and a pseudo-random permutation for the left bloc. Followed by the two most widely used genetic operators in cryptography, namely crossover and genetic mutation adapted to image encryption.

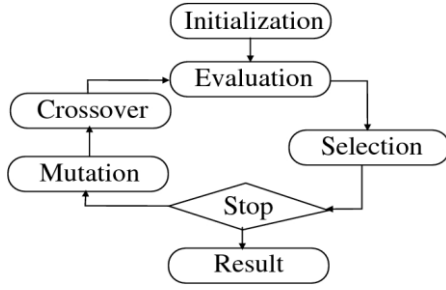


Figure 1. Genetic algorithm process

## 2. PROPOSED METHOD

Based on chaos [24], this new technology describes the construction of a genetic algorithm operating at the bit level. The implementation of this new system is based on the following axes:

### 2.1 Chaotic sequences development

The choice of chaotic maps is crucial for an encryption system. Therefore, our algorithm utilizes the two most used chaotic maps in cryptography, the logistic map, and the skewed tent map. This choice is due to the ease of implementation of both maps and their high sensitivity to initial conditions.

#### 2.1.1 The logistics map

The logistic map is a recurrent sequence defined by Eq. (1).

$$\begin{cases} s_0 \in [0,5 \ 1] & , \quad r \in [3,75 \ 4] \\ s_{n+1} = r s_n (1 - s_n) \end{cases} \quad (1)$$

$s_0$  is the initial value and  $r$  is the control parameter.

The Lyapunov exponent  $\lambda$  of the logistic map is  $\lambda = \log(2) > 0$ , which proves that this sequence is highly sensitive to initial conditions.

#### 2.1.2 The skew tent map

The Skew tent map is defined by Eq. (2).

$$\begin{cases} t_0 [0 \ 1] v \in [0,5 \ 1] \\ t_{n+1} = \begin{cases} \frac{t_n}{v} & \text{if } 0 < t_n < v \\ \frac{1-t_n}{1-v} & \text{if } v < t_n \end{cases} \end{cases} \quad (2)$$

$t_0$  is the initial value and  $v$  is the control parameter.

The combination of these two chaotic maps will be used to generate all the parameters necessary for the proper functioning and operation of our new technique.

#### 2.1.3 Chaotic table design

Our approach requires the use of a pseudo-random table (WC) of size (13nm, 3) constructed from the chaotic maps used with coefficients in  $G_{256}$  to ensure multiple confusions

(Algorithm 1), and a table (WB) of the same size for system action control (Algorithm 2).

---

### Algorithm 1: Chaotic vectors design

---

Begin

For  $i=1$  to 13nm

$$WC(i, 1) = \text{mod}(E(\min(s(i), t(i)) * 10^{11}, 254) +$$

1)

$$WC(i, 2) = \text{mod}\left(E\left(\frac{2*s(i)+3*t(i)}{5} * 10^{12}, 253\right) + 2\right)$$

$$WC(i, 3) = \text{mod}\left(E\left(\frac{2*s(i)+t(i)}{3} * 10^{13}, 252\right) + 3\right)$$

endFor

end

---



---

### Algorithm 2: Control table design

---

Begin

For  $i=1$  to 13nm

If  $WC(i, 1) \geq WC(i, 2)$  then

$$WB(i, 1) = 0$$

Else

$$WB(i, 1) = 1$$

EndIf

If  $WC(i, 2) \geq WC(i, 3)$  then

$$WB(i, 2) = 0$$

Else

$$WB(i, 2) = 1$$

EndIf

If  $WC(i, 3) \geq WC(i, 1)$  then

$$WB(i, 3) = 0$$

Else

$$WB(i, 3) = 1$$

EndIf

endFor

end

---

### 2.2 Population construction

#### 2.2.1 Image vectorization

Before starting the encryption process, an extraction of the 3RGB color channels is performed, followed by vectorization ( $W_r$ ), ( $W_g$ ), and ( $W_b$ ), each of size (nm), and then concatenation to obtain a vector ( $W$ ) of size (3nm). This process is illustrated in Figure 2 and described by Algorithm 3.

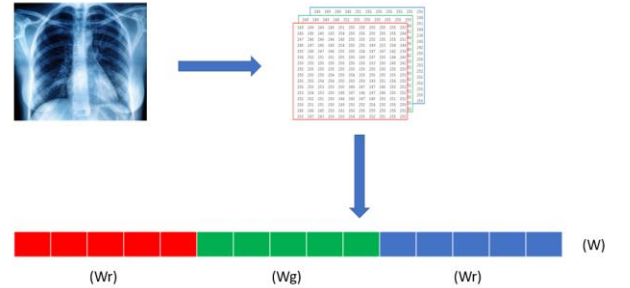


Figure 2. Image vectorization

---

### Algorithm 3: Vectorization process

---

Begin

For  $i=1$  to nm

$$W(i) = W_r(i)$$

$$W(i + nm) = W_g(i)$$

end

---

---

$W(i + 2nm) = W_b(i)$

**endFor**

---

### 2.2.2 Pseudo random transformation into binary

The vector ( $W$ ) of size ( $3nm$ ) will be converted into a binary vector ( $XB$ ) of size ( $24nm$ ) using 4 conversion tables (Figure 3). The choice of the conversion table is controlled by the control table ( $WB$ ). The binary conversion process is described by Algorithm 4. An example of conversion is illustrated by Figure 4.

After the pseudo random binary conversion, the vector ( $XB$ ) of size ( $24nm$ ) will be resized into a matrix ( $MB$ ) of size ( $4n, 6m$ ), this process is illustrated in Figure 5.

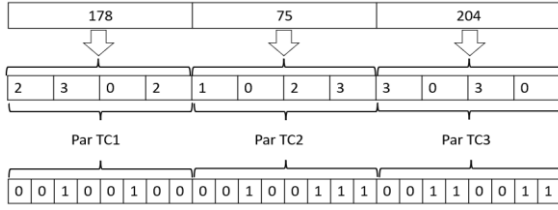
TC1	0	1
0	0	1
1	1	1
2	0	0
3	1	0

TC2	0	1
0	1	0
1	0	0
2	0	1
3	1	1

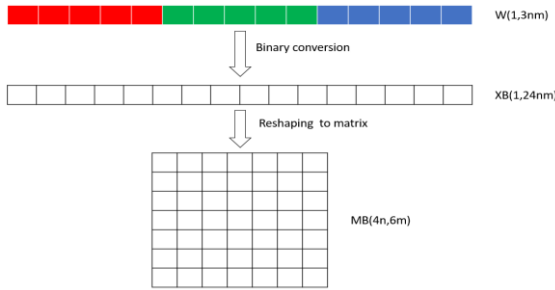
TC3	0	1
0	1	1
1	0	1
2	1	0
3	0	0

TC4	0	1
0	0	0
1	1	0
2	1	1
3	0	1

**Figure 3.** Conversion tables



**Figure 4.** Binary conversion example



**Figure 5.** Resizing process

---

#### Algorithm 4: Binary conversion

---

**Begin**

**For**  $i=1$  to  $3nm$

$x = W(i) \text{ div } 16$

$y = W(i) \text{ mod } 16$

$\alpha = x \text{ div } 4$

$\beta = x \text{ mod } 4$

$\delta = y \text{ div } 4$

$\gamma = y \text{ mod } 4$

**If**  $WB(i,1)=0$  and  $WB(i,2)=0$  **then**

$TC=TC1$

**Elif**  $WB(i,1)=0$  and  $WB(i,2)=1$  **then**

$TC=TC2$

**Elif**  $WB(i,1)=1$  and  $WB(i,2)=0$  **then**

$TC=TC3$

**Else**

$TC=TC4$

**EndIf**

---

$XB(8i - 7) = TC(\alpha, 0)$

$XB(8i - 6) = TC(\alpha, 1)$

$XB(8i - 5) = TC(\beta, 0)$

$XB(8i - 4) = TC(\beta, 1)$

$XB(8i - 3) = TC(\delta, 0)$

$XB(8i - 2) = TC(\delta, 1)$

$XB(8i - 1) = TC(\gamma, 0)$

$XB(8i) = TC(\gamma, 1)$

**end For**

**end**

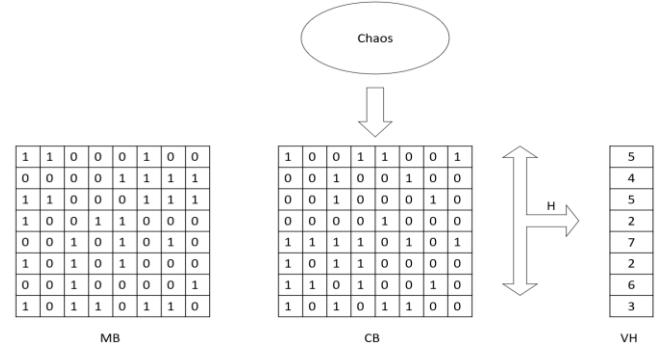
---

### 2.2.3 Fitness function definition

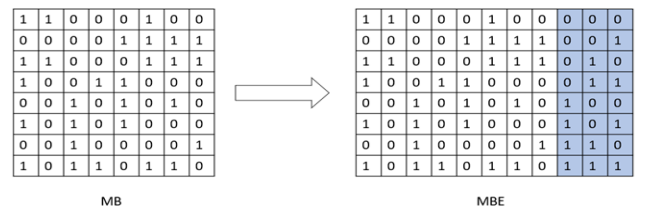
The matrix ( $MB$ ) will be considered as a population, where each row represents an individual. A fitness function will be applied to assess the population's integrity for future categorization into two groups: strong population and weak population. This evaluation function is defined by the Hamming distance between the original image ( $MB$ ) and the pseudo-random image ( $CB$ ) defined by Eq. (3). An example of hamming distance calculation is given in Figure 6.

$$H(MB(i,:), CB(i,:)) = \sum_{j=1}^{6m} MB(i,j) \oplus CB(i,j) \quad (3)$$

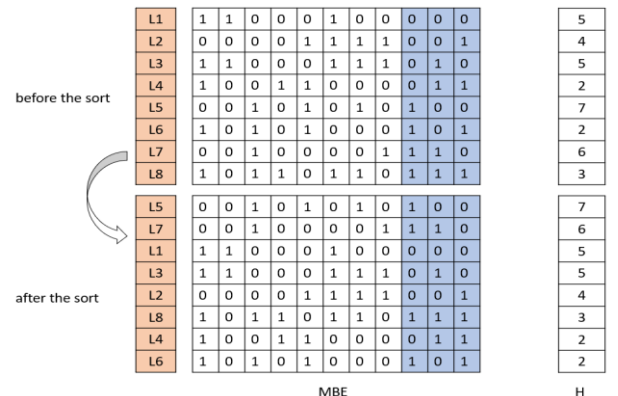
### 2.2.4 Construction of the categories



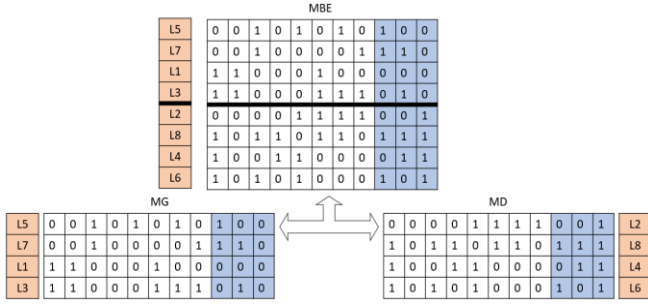
**Figure 6.** Hamming distance calculation



(a) row extension



(b) row sorting



(c) tribe generation

Figure 7. Construction of the row categories

Before partitioning the population, each row of the matrix (MB) will be augmented by 16 bits to store the identifier corresponding to its row number (Figure 7(a)). After that a descending sort of each row of (MBE) is established with respect to the weight vector (VH) to separate the population into 2 categories (Figure 7(b)). Finally the matrix (MBE) will be subdivided into two sub-matrices (MG) and (MD) each of size  $(2n, 6m+16)$  (Figure 7(c)).

### 3. ENCRYPTION PROCESS

Our encryption technique consists of coupling the rows of (MG) and (MD) matrices using an enhanced Feistel laps followed by a crossover and mutation operations.

#### 3.1 Feistel laps

The two matrices (MG) and (MD) will undergo coupling through an enhanced Feistel TF circuit defined by Eq. (4).

$$TF(G, D) = (G', D') = \begin{cases} G' = h_i(D) \\ D' = G \oplus f_i(D) \end{cases} \quad (4)$$

where,  $h_i$  is a binary permutation constructed by the binary vector  $(WB(:, 3))$  and  $f_i$  is a circular shift function determined by the coefficient  $WC(i, 3)$ . The coupling process is described by Algorithm 5.

#### Algorithm 5: Coupling process

1. **Begin**
2.  $G = MG(1, :)$
3.  $D = MD(1, :)$
4.  $XG = G \oplus VIG$
5.  $XD = D \oplus VID$
6.  $G' = h_i(XD)$
7.  $D' = XG \oplus f_i(XD)$
8.  $GM(1, :) = G'$
9.  $DM(1, :) = D'$
10. **For**  $i=2$  **to**  $2n$
11.  $G = MG(i, :)$
12.  $D = MD(i, :)$
13.  $XG = G \oplus G'$
14.  $XD = D \oplus D'$
15.  $G' = h_i(XD)$
16.  $D' = XG \oplus f_i(XD)$
17.  $GM(i, :) = G'$
18.  $DM(i, :) = D'$
19. **end For**
20. **end**

Using VIG and VID are obtained by applying the XOR operation on all the rows of the MG matrix and MD matrix, excluding the first row. The enhanced Feistel round is illustrated in Figure 8.

At the end of the Feistel round. The two matrices (GM) and (DM) of size  $(2n, 6m+16)$  are combined into a single matrix ( $M'$ ) of size  $(2n, 12m+32)$ . This matrix will be transformed into a vector (BX) of size  $(24nm+64n)$ .

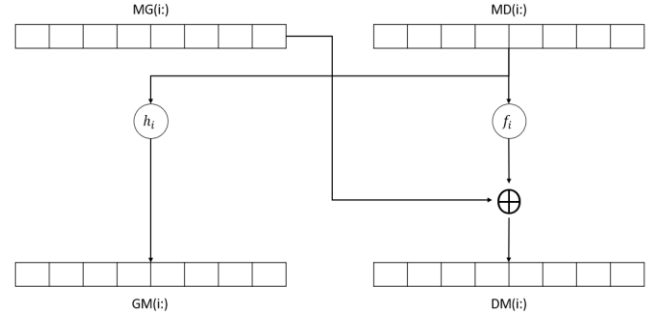


Figure 8. Feistel lap

### 3.2 Genetic crossover

The second encryption phase involves a genetic crossover adapted for image encryption. For this reason, a pseudo-random transformation at the DNA level is necessary. This operation is described by the following steps:

#### 3.2.1 Transition to $G_4$

Transition to  $G_4$  the binary vector (BX) of size  $(24nm+64n)$  will be converted into a vector (BY) of size  $(12nm+32n)$  taking values in  $G_4$  using Algorithm 6.

#### Algorithm 6: Transition to $G_4$

1. **Begin**
2. **For**  $i=1$  **to**  $12nm+32n$
3.  $BY(i) = 2 * BX(2 * i - 1) + BX(2 * i)$
4. **end For**
5. **end**

#### 3.2.2 Transition to DNA notation

The transition to DNA notation involves writing the vector (BY) into (YB) vector using the symbols  $\{A, C, T, G\}$ . For this, we will use two conversion tables T1 and T2 defined in Figure 9. The choice of the conversion table is determined by the control vector  $(WB(:, 3))$ . The process of converting to DNA notation is illustrated by Algorithm 7.

	0	1	2	3
T1	T	A	G	C

	0	1	2	3
T2	C	T	A	G

Figure 9. DNA conversion table

#### Algorithm 7: Conversion to DNA notation

1. **Begin**
2. **For**  $i=1$  **to**  $12nm+32n$
3.  $x = BY(i)$
4. **If**  $WB(i,3)=0$  **then**
5.  $YB(i) = T1(x)$

---

```

6.      Else
7.      YB(i) = T2(x)
8.      endFor
9.      end

```

---

### 3.2.3 Genetic crossover function

The vector (YB) will be genetically crossed with a pseudorandom vector (R) of size (12nm+32n) defined by Algorithm 8.

---

#### Algorithm 8: Generation of the vector R

---

```

1.      Begin
2.      For i=1 to 12nm+32n
3.      If s(i)<=0.25 then
4.      R(i)="A"
5.      Eilf s(i)<=0.5 then
6.      R(i)="C"
7.      Eilf s(i)<=0.75 then
8.      R(i)="T"
9.      Else
10.     R(i)="G"
11.     end For
12.     end

```

---

The genetic crossover operation is controlled by a crossover table TC of size (12nm+32n, 3) where each column is defined as follows:

- The first column: a permutation P obtained by sorting the 12nm+32n values of the sequence s, indicating the index of the pixel to be processed.
- The second column: a permutation Q obtained by sorting the 12nm+32n values of the sequence t, indicating the index of the factor of R to be used.
- The third column: the QoP permutation indicating the index of the pixel obtained through genetic crossover.

The process of genetic crossover is illustrated by Algorithm 9.

---

#### Algorithm 9: Genetic crossover

---

```

1.      Begin
2.      For i=1 to 12nm+32n
3.      x = YB(TC(i, 1))
4.      If WB(i,1)=0 then
5.      y = x ⊕ R(TC(i, 2))
6.      Else
7.      y = x ⊗ R(TC(i, 2))
8.      WQ(TC(i, 3)) = y
9.      end For
10.     end

```

---

⊕	A	C	T	G
A	A	C	T	G
C	C	A	G	T
T	T	G	A	C
G	G	T	C	A

⊗	A	C	T	G
A	T	G	A	C
C	G	T	C	A
T	A	C	T	G
G	C	A	G	T

Figure 10. DNA conversion table

With  $\oplus$  and  $\otimes$  are crossover operators defined at DNA level defined by Figure 10.

### 3.3 Genetic mutation

The third encryption phase involves establishing a diffusion operation ensured by a genetic mutation adapted for medical image encryption. This step will be described by the following axes:

#### 3.3.1 Transition to $G_4$

The vector (WQ) written in DNA notation of size (12nm+32n) will be reconverted into a vector (QW) with coefficients in  $G_4$  using conversion tables defined in Figure 11. The choice of the conversion table is determined by the control vector (WB(:, 2)). The process of conversion to  $G_4$  is illustrated by Algorithm 10.

	A	T	C	G
T3	0	1	2	3

	A	T	C	G
T4	1	0	3	2

Figure 11. DNA to  $G_4$  tables conversion

---

#### Algorithm 10: Transition to $G_4$

---

```

1.      Begin
2.      For i=1 to 12nm+32n
3.      x = WQ(i)
4.      If WB(i,2)=0 then
5.      QW(i) = T3(x)
6.      Else
7.      QW(i) = T4(x)
8.      end For
9.      end

```

---

#### 3.3.2 Transition to gray level ( $G_{256}$ )

The vector (QW) of size (12nm+32n) will be converted into a vector (QZ) of size (3nm+8n) with values in ( $G_{256}$ ). This conversion is illustrated by Algorithm 11.

---

#### Algorithm 11: Transition to $G_{256}$

---

```

1.      Begin
2.      For i=1 to 3nm+8n
3.      QZ(i) = QW(4 * i - 3) + 2 * QW(4 * i - 2) + 4 * QW(4 * i - 1) + 8 * QW(4 * i)
4.      end For
5.      end

```

---

#### 3.3.3 Initial value calculation

An initial value (WI) will be calculated to initialize the diffusion process. The calculation of the initial value is shown in Algorithm 12.

---

#### Algorithm 12: Calculation of the initial value

---

```

1.      Begin
2.      WI=0
3.      For i=1 to 3nm+8n
4.      WI = WI ⊕ QZ(i)
5.      end For
6.      end

```

---

### 3.3.4 Genetic mutation function

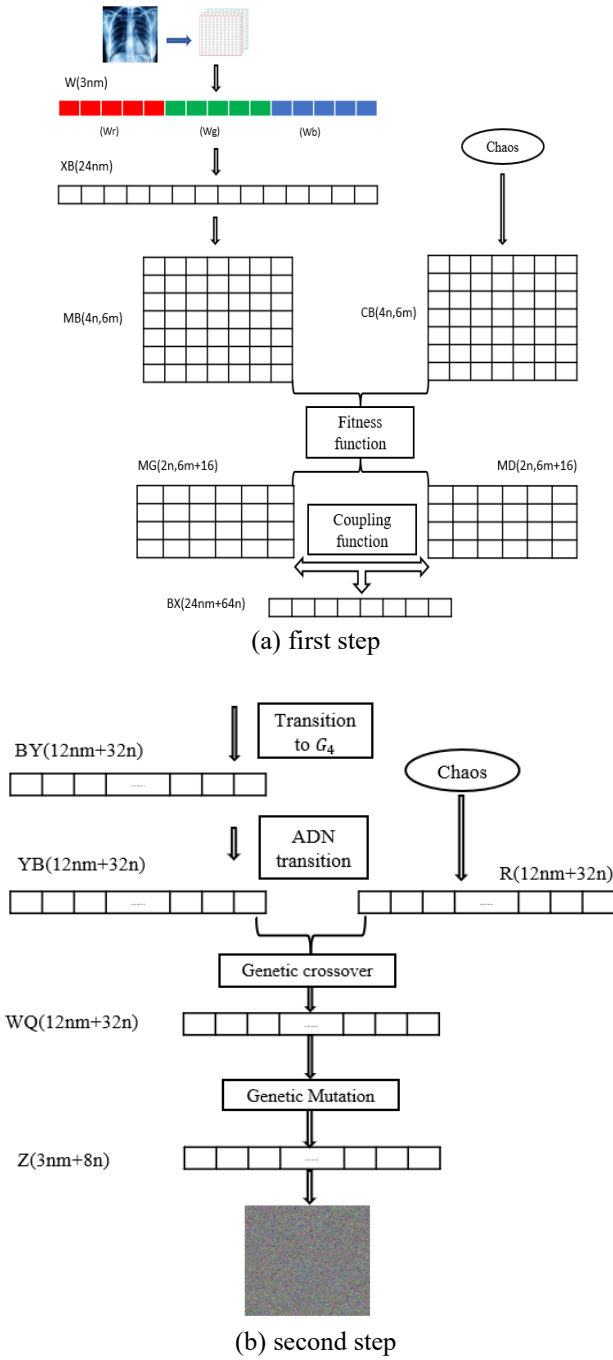


Figure 12. Encryption process

The vector (QZ) of size  $(3nm+8n)$  will undergo a genetic mutation function, defined by an inversion operation, and controlled by a mutation table of size  $(3nm+8n, 3)$ . Each column of the mutation table is defined as follows:

- The first column: a permutation P obtained by sorting the  $3nm+8n$  values of the sequence  $s+t$ , indicating the index of the pixel to be processed.
- The second column: a permutation Q obtained by sorting the  $3nm+8n$  values of the sequence  $t+2s$ , indicating the index of the pixel obtained through genetic crossover.
- The third column: is a pseudo-random control vector used to control genetic mutation. This vector is defined by Algorithm 13.

The diffusion process is illustrated by Algorithm 14.

#### Algorithm 13: Generation of the genetic mutation control vector.

```

1. Begin
2. For i=1 to 3nm+8n
3.   If s(i)≤0.1 then
4.     TM(i,3)=0
5.   Else
6.     TM(i,3)=1
7.   end For
8. end

```

#### Algorithm 14: Diffusion operation using genetic mutation.

```

1. Begin
2.   If TM(1,3)=1 then
3.     Z(TM(1,2)) = WI ⊕ 255
4.   Else
5.     Z(TM(1,2)) = WI
6.   For i=2 to 3nm+8n
7.     If TM(i,3)=1 then
8.       Z(TM(i,2)) = Z(TM(i-1,2)) ⊕
9.         QZ(TM(i,1)) ⊕ 255
10.    Else
11.      Z(TM(i,2)) = Z(TM(i-1,2)) ⊕
12.        QZ(TM(i,1))
13.    end For
14. end

```

The vector (Z) constitutes the encrypted image by our algorithm and the entire encryption process is presented in Figure 12.

## 4. DECRYPTION PROCESS

Our algorithm is a symmetric encryption algorithm. By applying the inverse encryption functions, in reverse order and using the same encryption key, we can restore the original image.

### 4.1 The inverse of the genetic mutation

The inverse function of genetic mutation is described by Algorithm 15.

#### Algorithm 15: Reverse of genetic mutation

```

1. Begin
2.   For i=3nm+8n to 2
3.     If TM(i,3)=1 then
4.       QZ(TM(i,1)) = Z(TM(i-1,2)) ⊕
5.         Z(TM(i,2)) ⊕ 255
6.     Else
7.       QZ(TM(i,1)) = Z(TM(i-1,2)) ⊕
8.         Z(TM(i,2))
9.     End If
10.   end For
11.   QZ(TM(1,1)) = Z(TM(1,2))
12.   For i=1 to 3nm+8n
13.     If i≠TM(1,1) then
14.       QZ(TM(1,1)) = QZ(TM(1,1)) ⊕
15.         QZ(i)
16.     end If
17.   End For

```

---

```

15.   If  $TM(1,3)=1$  then
16.      $QZ(TM(1,1)) = QZ(TM(1,1)) \oplus 255$ 
17.   Else
18.      $QZ(TM(1,1)) = QZ(TM(1,1))$ 
19.   End If
20.   end

```

---

#### 4.2 The inverse of the genetic crossover

The genetic crossover operation acts on the image at the DNA level, a transition from the vector (QZ) to notation is necessary. First, the vector will be converted to the vector (QW) of size  $(12nm+32n)$  with coefficients in  $G_4$ . This conversion is described by Algorithm 16.

---

##### Algorithm 16: Transition to $G_4$

---

```

1.   Begin
2.   For  $i=1$  to  $3nm+8n$ 
3.      $x = QZ(i) \bmod 16$ 
4.      $y = QZ(i) \div 16$ 
5.      $QW(4i - 3) = x \bmod 4$ 
6.      $QW(4i - 2) = x \div 4$ 
7.      $QW(4i - 1) = y \bmod 4$ 
8.      $QW(4i) = y \div 4$ 
9.   end For
10.  end

```

---

The vector (QW) will be converted to the vector (WQ) in DNA notation using inverse conversion tables: TI3 (inverse of T3) and TI4 (inverse of T4) described in Figure 13. The conversion process is illustrated by Algorithm 17.

	0	1	2	3
TI3	A	T	C	G

	0	1	2	3
TI4	T	A	G	C

Figure 13.  $G_4$  to DNA tables conversion

---

##### Algorithm 17: Transition to DNA notation

---

```

1.   Begin
2.   For  $i=1$  to  $12nm+32n$ 
3.      $x = QW(i)$ 
4.     If  $WB(i)=0$  then
5.        $WQ(i) = TI3(x)$ 
6.     Else
7.        $WQ(i) = TI4(x)$ 
8.     end For
9.   end

```

---

The inverse genetic crossover process is illustrated by Algorithm 18.

---

##### Algorithm 18: The inverse of genetic crossover

---

```

1.   Begin
2.   For  $i=12nm+32n$  to 1
3.      $x = WQ(TC(i, 1))$ 
4.     If  $TC(i,4)=0$  then
5.        $y = x \oplus R(TC(i, 2))$ 
6.     Else

```

---



---

```

7.      $y = x \otimes R(TC(i, 2))$ 
8.      $YB(TC(i, 3)) = y$ 
9.   end For
10.  End

```

---

#### 4.3 Feistel lap inverse

Before applying the inverse Feistel network, a conversion to the binary level is crucial. For this, the vector (YB) will be converted to a vector (BY) with values in  $G_4$  using the inverse tables TI1 (inverse of T1) and TI2 (inverse of T2) described in Figure 14. The conversion process is illustrated by Algorithm 19.

	A	T	C	G
TI1	1	0	3	2

	A	T	C	G
TI2	2	1	0	3

Figure 14.  $G_4$  to DNA tables conversion

---

##### Algorithm 19: Transition to $G_4$

---

```

1.   Begin
2.   For  $i=1$  to  $12nm+32n$ 
3.      $x = YB(i)$ 
4.     If  $WB(i,2)=0$  then
5.        $BY(i) = TI1(x)$ 
6.     Else
7.        $BY(i) = TI2(x)$ 
8.     end For
9.   end

```

---

The vector (BY) will then be converted to a binary vector (BX) using Algorithm 20. After that, the vector (BX) will be resized into a matrix ( $M'$ ) of size  $(2n, 12m+32)$ . This matrix will be subdivided into two sub-matrices (GM) and (DM) each of size  $(2n, 6m+16)$ . The inverse Feistel network is applied to each row of (GM) and (DM). This process is described by Algorithm 21.

---

##### Algorithm 20: Binary transition

---

```

1.   Begin
2.   For  $i=1$  to  $12nm+32n$ 
3.      $BX(2 * i - 1) = BY(i) \div 2$ 
4.      $BX(2 * i) = BY(i) \bmod 2$ 
5.   end For
6.   end

```

---



---

##### Algorithm 21: Feistel lap inverse

---

```

1.   Begin
2.   For  $i=2n$  to 2
3.      $G' = MG'(i, :)$ 
4.      $D' = MD'(i, :)$ 
5.      $XD = h_i^{-1}(G')$ 
6.      $XG = D' \oplus f_i(XD)$ 
7.      $G = XG \oplus MG'(i - 1, :)$ 
8.      $D = XD \oplus MD'(i - 1, :)$ 
9.      $MG(i, :) = XG$ 
10.     $MD(i, :) = XD$ 
11.  end For

```

---

---

```

21.     $D' = DM(1, :)$ 
22.     $G' = GM(1, :)$ 
23.     $XD = h_i^{-1}(G')$ 
24.     $XG = D' \oplus f_i(XD)$ 
25.     $D = XD \oplus VID$ 
26.     $G = XG \oplus VIG$ 
27.     $MG(1, :) = G$ 
28.     $MD(1, :) = D$ 
29.    end

```

---

The two matrices (MG) and (MD) will be combined into a single matrix (MBE) of size  $(4n, 6m+16)$ .

#### 4.4 Reconstruction of the clear image

The matrix (MBE) will be sorted based on the last 16 bits of each row, and then extraction of the matrix (MB) from (MBE) by removing the last 16 columns to construct the decrypted image. The binary matrix (MB) of size  $(4n, 6m)$  will be transformed into a vector (XB) of size  $(24nm)$ . The vector (XB) will be converted to grayscale to recover the encrypted image (W) using Algorithm 22.

---

#### Algorithm 22: Transition to $G_{256}$

---

```

1.    Begin
2.    For i=1 to 3nm
3.         $\beta_1 = XB(8 * i - 7)$ 
4.         $\beta_2 = XB(8 * i - 6)$ 
5.         $\beta_3 = XB(8 * i - 5)$ 
6.         $\beta_4 = XB(8 * i - 4)$ 
7.         $\beta_5 = XB(8 * i - 3)$ 
8.         $\beta_6 = XB(8 * i - 2)$ 
9.         $\beta_7 = XB(8 * i - 1)$ 
10.        $\beta_8 = XB(8 * i)$ 
11.       If  $WB(i, 1) = 0$  and  $WB(i, 2) = 0$  then
12.            $W(i) = 64 * TCI1(\beta_1, \beta_2) + 16 * TCI1(\beta_3, \beta_4) + 4 * TCI1(\beta_5, \beta_6) + TCI1(\beta_7, \beta_8)$ 
13.       Elif  $WB(i, 1) = 0$  and  $WB(i, 2) = 1$  then
14.            $W(i) = 64 * TCI2(\beta_1, \beta_2) + 16 * TCI2(\beta_3, \beta_4) + 4 * TCI2(\beta_5, \beta_6) + TCI2(\beta_7, \beta_8)$ 
15.       Elif  $WB(i, 1) = 1$  and  $WB(i, 2) = 0$  then
16.            $W(i) = 64 * TCI3(\beta_1, \beta_2) + 16 * TCI3(\beta_3, \beta_4) + 4 * TCI3(\beta_5, \beta_6) + TCI3(\beta_7, \beta_8)$ 
17.       Else
18.            $W(i) = 64 * TCI4(\beta_1, \beta_2) + 16 * TCI4(\beta_3, \beta_4) + 4 * TCI4(\beta_5, \beta_6) + TCI4(\beta_7, \beta_8)$ 
19.       end For
20.    end

```

---

The tables TCI1, TCI2, TCI3, and TCI4 are the inverse conversion tables described in Figure 15. The vector (W) represents the decrypted image.

	A	T	C	G
T11	1	0	3	2

	A	T	C	G
T12	2	1	0	3

Figure 15. Binary tables conversion

## 5. SIMULATION RESULTS

To evaluate the performance of our encryption system, we randomly select many reference images and then subject them to our encryption method. In this section, all experiments are conducted using the Python programming language on a personal computer based on i5, with 8GB of RAM and a 500GB hard drive, running Ubuntu 20.04.

### 5.1 Brutal attacks

Brutal attacks consist of reconstructing the encryption keys in a random manner by exploring all the key-space values.

#### 5.1.1 Key-space analysis

To ensure the robustness of a good image encryption algorithm, the key space should have at least 2100 possibilities. If the key space is not sufficiently large, the algorithm could be vulnerable to brute-force attacks. In this algorithm, the key space encompasses the initial condition and control parameters of the chaotic maps. The secret key of our system consists of:

- $s_0 = 0.79878796, r = 0.755654$  logistique map.
- $t_0 = 0.6789654, v = 0.98788755$  skew tente map.

In our case, we consider the precision to be of the order of 1016. Therefore, the total key space reaches 1064. Thus, our algorithm is capable of resisting brute-force attacks due to the sufficiently large size of its key space.

#### 5.1.2 Secret key's sensitivity analysis

The high sensitivity of our encryption key is manifested by an amplified reaction following a slight modification of a single parameter, resulting in a significant difference from the original image. This assertion is visually confirmed by Figure 16.

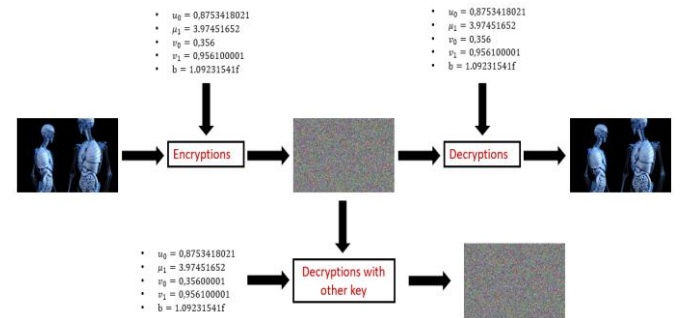


Figure 16. Encryption key sensitivity

### 5.2 Visual test

The visual test is the first test which aims to detect a certain resemblance between the original image and the image encrypted by the new crypto system. In our simulations, visually the encrypted image is totally different from the original image and does not reflect any information or resemblance, this initially ensures the robustness of our system is illustrated by the result of the Table 1.

### 5.3 Histogram analysis

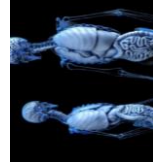

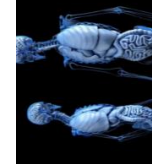

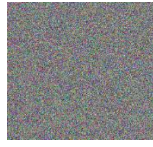


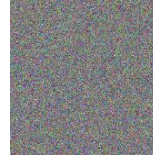
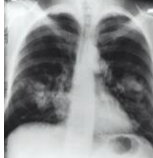





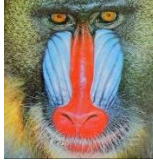
The histogram represents the distribution of pixels in an image. In the case of an encrypted image, the histogram should be uniformly distributed to prevent attackers from guessing information about the image. Furthermore, the histogram of



the encrypted image should not resemble that of the original image. Table 2 shows the histograms of some encrypted

images. As can be seen, the histograms of the encrypted images obtained with our algorithm are uniform.

**Table 1.** Visual test

Image Name	Clair Image	Encrypted Image	Decrypted Image
Img1 243×411			
Img2 1057×1200			
Img3 837×821			
Img4 512×512			
Img5 512×512			

### 5.4 Correlation analysis

In the original image, neighboring pixels demonstrate a pronounced correlation due to their closely similar values. The efficacy of the encryption algorithm resides in generating an encrypted image characterized by minimal correlation among adjacent pixels. Mathematically, the correlation coefficient between two neighboring pixels is calculated employing the Eq. (5).

$$r = \frac{cov(x, y)}{\sqrt{V(x)}\sqrt{V(y)}} \quad (5)$$

With:

- "x" and "y" represent adjacent pixels.
- "cov (x, y)": covariance between the random variables x and y.
- V(x): variance of x.

The Table 3 shows the correlation coefficients of some images encrypted by our method. All the images have a correlation close to zero.

### 5.5 Entropy analysis

Entropy is the measure of the disorder diffused by a source without memory. The entropy expression is determined by the Eq. (6).

$$H(img) = \sum_{i=0}^{255} -p(img_i) \log_2(p(img_i)) \quad (6)$$

$p(img_i)$ , is the probability of occurrence of level (i) in the image encrypted by our new method.

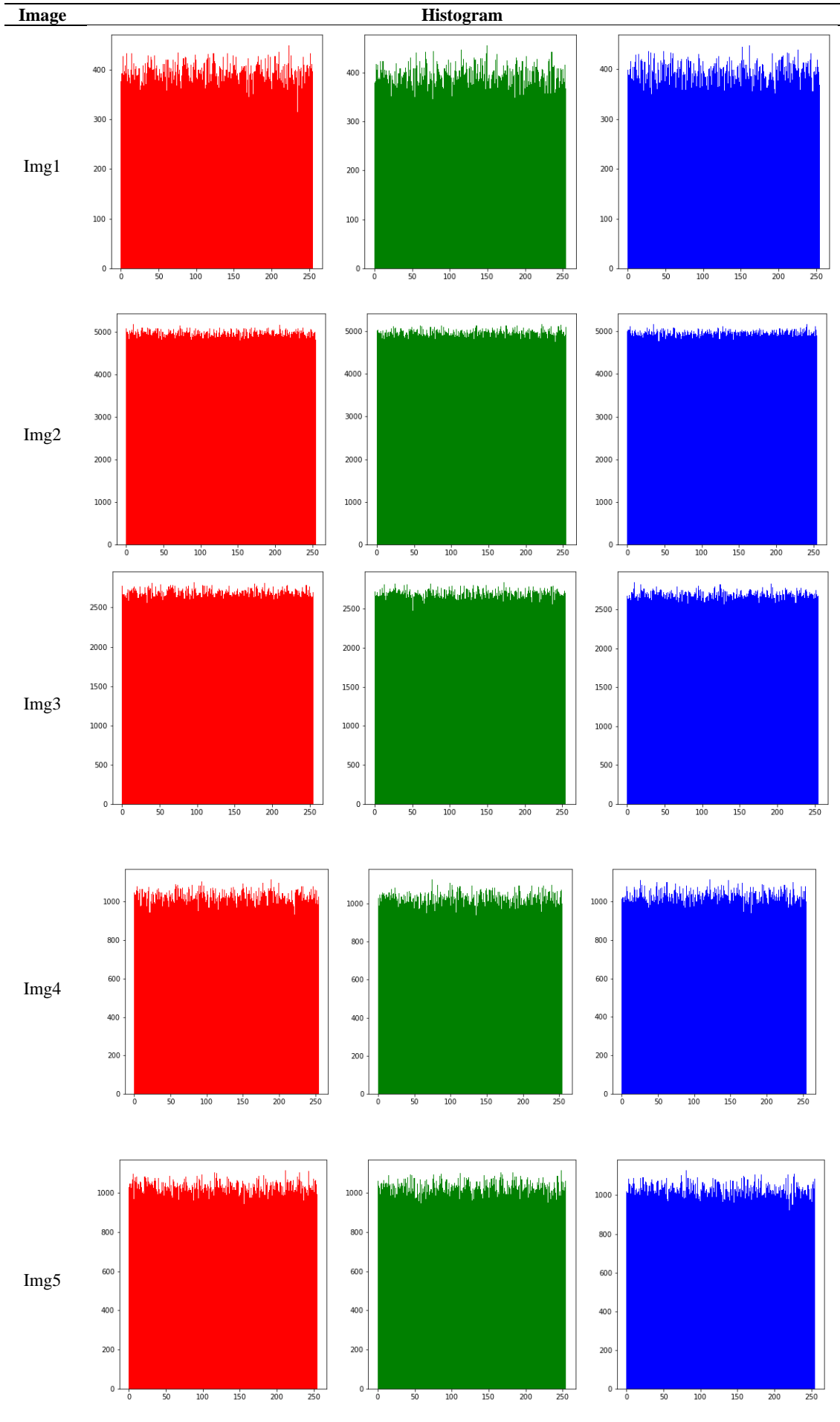
The maximum value of entropy is 8; the higher the entropy, the more randomness is present in the encrypted image. The Table 3 presents the entropy values obtained for different images encrypted by our algorithm. All the obtained values are close to the maximum value of 8.

### 5.6 Differential analysis

Differential attack is based on deducing information about an image by introducing a slight modification to the plaintext image and then encrypting both images using the same algorithm. A comparison between the two encrypted images is made to determine any correlation between the original image and its encrypted version. It is expected that a robust algorithm produces distinctly different encrypted images, even for the slightest modifications made to the plaintext image. The parameters used to measure the effectiveness of our method against differential attacks are: NPCR (Number of Pixel Change Rate) and UACI (Unified Average Changing Intensity).

NPCR (Number of Pixel Change Rate) is a measure that quantifies the percentage of different pixel values between two encrypted images, when the corresponding plaintext images differ by only one pixel. A high NPCR value indicates better resistance of the algorithm against differential attacks. NPCR can be calculated using Eq. (7).

Table 2. Histogram



**Table 3.** Correlation and entropy

Image		Vertical Correlation	Diagonal Correlation	Horizontal Correlation	Entropy
Img1	R	0.00194	-0.00264	-0.00064	7.99829
	G	0.00162	-0.00201	-9.11e-05	7.99806
	B	0.00014	-0.00307	0.00092	7.99811
Img2	R	-0.00122	0.00090	0.00098	7.99985
	G	0.00031	-7.75e-05	0.00093	7.99983
	B	-0.00056	0.00050	-0.00043	7.99987
Img3	R	0.00064	-0.00159	8.17e-05	7.99977
	G	-0.00177	-0.00156	0.00068	7.99971
	B	-0.00102	0.00112	0.00124	7.99975
Img4	R	0.00020	0.00161	0.00166	7.99930
	G	-0.00127	-0.00312	-0.00213	7.99930
	B	0.00023	0.00233	0.00138	7.99929
Img5	R	0.00125	0.00096	0.00232	7.99935
	G	0.00059	0.00178	-0.00224	7.99925
	B	-0.00464	-0.00014	-0.00173	7.99922

$$NPCR = \left( \frac{1}{nm} \sum_{i,j=1}^{nm} D(i,j) \right) * 100 \quad (7)$$

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

The UACI mathematical analysis of an image is given by Eq. (8).

$$UACI = \left( \frac{1}{255nm} \sum_{i,j=1}^{nm} Abs(C_1(i,j) - C_2(i,j)) \right) * 100 \quad (8)$$

The avalanche effect refers to the number of bits that change when a single bit in the original image is modified [24]. The mathematical representation of the avalanche effect is expressed by Eq. (9).

$$AE = \left( \frac{\sum_i \text{bit change}}{\sum_i \text{bit total}} \right) * 100 \quad (9)$$

**Table 4.** Differential parameters

Pictures	NPCR	UACI	EA
Img1	99.61%	33.46%	50.03%
Img2	99.61%	33.47%	50.00%
Img3	99.61%	33.48%	50.00%
Img4	99.60%	33.46%	50.01%
Img5	99.61%	33.46%	50.01%

Table 4 presents some results of the calculation of differential parameters for images encrypted using our method. All calculated NPCR values are greater than the threshold of 99.60%, UACI is greater than 33.46%, and AE is greater than 50%. This demonstrates that our algorithm can resist differential attacks.

### 5.7 Encryption time

The encryption time is a crucial benchmark for assessing the efficiency of an image encryption algorithm. Effectively encrypting substantial data, such as images, within a reasonable timeframe has become a challenging aspect of algorithm development. In our study, we present the encryption times for images of sizes 256×256 and 512×512 in (Table 5), along with a comparison to other recent works. Furthermore, the time complexity of our method for an image

of size (N, M) is O(NM).

**Table 5.** Encryption time

Encryption Technique	Images	
	256x256	512x512
Our algorithm	0.097	0.171
Ref. [25]	0.65	-----
Ref. [26]	8.22	-----
Ref. [27]	0.156	0.406

### 5.8 Comparison to other techniques

Table 6 shows that our technique gives satisfactory results compared to other method in the literature. The entropy of the tow images gives values extremely close to the ideal value of 8 and higher than the values obtained by most other methods. For the NPCR and UACI, the values of the proposed scheme are very close to the values of NPCR (99.6%) and UACI (33.4%) expected. Accordingly, from Table 6, we can conclude that the suggested method outperforms many recently published studies in terms of entropy, NPCR and UACI values.

**Table 6.** Comparison to other technique

Images	Methods	Entropy	NPCR	UACI
Lena	<b>Our Method</b>	<b>7.9993</b>	<b>99.61</b>	<b>33.46</b>
	Ref. [22]	7.9976	99.61	33.51
	Ref. [28]	7.9992	99.65	33.48
Baboon	<b>Our Method</b>	<b>7.9993</b>	<b>99.61</b>	<b>33.46</b>
	Ref. [29]	7.9984	99.60	33.26

## 6. CONCLUSIONS

Converting the image to be encrypted into binary notation and applying a discrimination function in our genetic algorithm resulted in the generation of two tribes of individuals with different properties. The implementation of a Feistel round and the integration of a diffusion function in the first round, for the mating of two individuals, made it possible to reproduce a new generation, inheriting the genes of each tribe in order to cope with any unforeseen attack. While crossover and mutation in the second round reinforced our encryption system. A wide range of images tested by our algorithm proved the robustness of our cryptosystem. Reformulating the fitness function and increasing the number

of operations at the DNA level may add further performance to our system.

## REFERENCES

- [1] Mfungo, D.E., Fu, X., Wang, X., Xian, Y. (2023). Enhancing image encryption with the kronecker xor product, the hill cipher, and the sigmoid logistic map. *Applied Sciences*, 13(6): 4034. <https://doi.org/10.3390/app13064034>
- [2] Qobbi, Y., Jarjar, A., Essaid, M., Benazzi, A. (2022). New image encryption scheme based on dynamic substitution and hill cipher. In *WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*. Springer Singapore, pp. 797-808. [https://doi.org/10.1007/978-981-33-6893-4\\_72](https://doi.org/10.1007/978-981-33-6893-4_72)
- [3] Arifin, S., Kurniadi, F.I., Yudistira, I.G.A., Nariswari, R., Murnaka, N.P., Muktyas, I.B. (2022). Image encryption algorithm through hill cipher, shift 128 cipher, and logistic map using python. In *2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, IEEE, IPOH, Malaysia, pp. 221-226. <https://doi.org/10.1109/AiDAS56890.2022.9918696>
- [4] Lone, M.A., Qureshi, S. (2023). Encryption scheme for RGB images using chaos and affine hill cipher technique. *Nonlinear Dynamics*, 111(6): 5919-5939. <https://doi.org/10.1007/s11071-022-07995-2>
- [5] Boussif, M., Aloui, N., Cherif, A. (2020). Securing DICOM images by a new encryption algorithm using Arnold transform and Vigenère cipher. *IET Image Processing*, 14(6): 1209-1216. <https://doi.org/10.1049/iet-ipt.2019.0042>
- [6] Mir, U.H., Lone, P.N., Singh, D., Mishra, D.C. (2023). A public and private key image encryption by modified approach of Vigenere cipher and the chaotic maps. *The Imaging Science Journal*, 71(1): 82-96. <https://doi.org/10.1080/13682199.2023.2175436>
- [7] Abid, A., Jarjar, M., Benazzi, A., Jarjar, A. (2022). Color image encryption using improved vigenère method followed by a permutation. In *The Proceedings of the International Conference on Smart City Applications*. Cham: Springer International Publishing, Springer, Cham, pp. 580-590. [https://doi.org/10.1007/978-3-031-26852-6\\_54](https://doi.org/10.1007/978-3-031-26852-6_54)
- [8] Bhavana, V., Banushree, J., Bhumika, D., Chaitanya, B., BIET, D., Raghu, B. (2021). A crypto system using vigenere and po-lybius cipher. *International Journal of Engineering Applied Sciences and Technology*, 6: 39-42.
- [9] Zhang, X., Wang, L., Cui, G., Niu, Y. (2019). Entropy-based block scrambling image encryption using DES structure and chaotic systems. *International Journal of Optics*, 2019. <https://doi.org/10.1155/2019/3594534>
- [10] Zhang, X., Zhou, Z., Niu, Y. (2018). An image encryption method based on the feistel network and dynamic DNA encoding. *IEEE Photonics Journal*, 10(4): 1-14. <https://doi.org/10.1109/JPHOT.2018.2859257>
- [11] JarJar, A. (2019). Two Feistel rounds in image cryptography acting at the nucleotide level exploiting dna and rna property. *SN Applied Sciences*, 1(11): 1411. <https://doi.org/10.1007/s42452-019-1305-7>
- [12] Abid, A., Qobbi, Y., Benazzi, A., Jarjar, M., Jarjar, A. (2022). Two enhanced Feistel schemes for medical image encryption. In *2022 IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pp. 1-4. <https://doi.org/10.1109/ICECOCS55148.2022.9982938>
- [13] Hraoui, S., JarJar, A. (2022). Single Feistel lapse acting on reduced ASCII codes followed by a genetic crossover. *SN Applied Sciences*, 4(4): 113. <https://doi.org/10.1007/s42452-022-04972-7>
- [14] Zhang, Y. (2018). Test and verification of AES used for image encryption. *3D Research*, 9: 1-27. <https://doi.org/10.1007/s13319-017-0154-7>
- [15] Arab, A., Rostami, M.J., Ghavami, B. (2019). An image encryption method based on chaos system and AES algorithm. *The Journal of Supercomputing*, 75: 6663-6682. <https://doi.org/10.1007/s11227-019-02878-7>
- [16] Maurer, U.M. (1993). The role of information theory in cryptography. In *Fourth IMA Conference on Cryptography and Coding*, pp. 49-71.
- [17] Mirjalili, S., Song Dong, J., Sadiq, A.S., Faris, H. (2020). Genetic algorithm: Theory, literature review, and application in image reconstruction. *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*, 69-85. [https://doi.org/10.1007/978-3-030-12127-3\\_5](https://doi.org/10.1007/978-3-030-12127-3_5)
- [18] Lambora, A., Gupta, K., Chopra, K. (2019). Genetic algorithm-A literature review. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, pp. 380-384. <https://doi.org/10.1109/COMITCon.2019.8862255>
- [19] Katoch, S., Chauhan, S.S., Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80: 8091-8126. <https://doi.org/10.1007/s11042-020-10139-6>
- [20] Gen, M., Lin, L. (2023). Genetic algorithms and their applications. In *Springer Handbook of Engineering Statistics*, London: Springer London, pp. 635-674. [https://doi.org/10.1007/978-1-4471-7503-2\\_33](https://doi.org/10.1007/978-1-4471-7503-2_33)
- [21] Ghazvini, M., Mirzadi, M., Parvar, N. (2020). A modified method for image encryption based on chaotic map and genetic algorithm. *Multimedia Tools and Applications*, 79: 26927-26950. <https://doi.org/10.1007/s11042-020-09058-3>
- [22] Mahmud, M., Lee, M., Choi, J.Y. (2020). Evolutionary-based image encryption using RNA codons truth table. *Optics & Laser Technology*, 121: 105818. <https://doi.org/10.1016/j.optlastec.2019.105818>
- [23] Ferdush, J., Mondol, G., Prapti, A.P., Begum, M., Sheikh, M.N.A., Galib, S.M. (2021). An enhanced image encryption technique combining genetic algorithm and particle swarm optimization with chaotic function. *International Journal of Computers and Applications*, 43(9): 960-967. <https://doi.org/10.1080/1206212X.2019.1662170>
- [24] Zolfaghari, B., Koshiba, T. (2022). Chaotic image encryption: State-of-the-art, ecosystem, and future roadmap. *Applied System Innovation*, 5(3): 57. <https://doi.org/10.3390/asi5030057>
- [25] Khan, J.S., Ahmad, J. (2019). Chaos based efficient selective image encryption. *Multidimensional Systems and Signal Processing*, 30: 943-961. <https://doi.org/10.1007/s11045-018-0589-x>
- [26] Ayoup, A.M., Hussein, A.H., Attia, M.A. (2016). Efficient selective image encryption. *Multimedia Tools*

- and Applications, 75: 17171-17186. <https://doi.org/10.1007/s11042-015-2985-7>
- [27] Laiphrakpam, D.S., Khumanthem, M.S. (2017). Medical image encryption based on improved ElGamal encryption technique. Optik, 147: 88-102. <https://doi.org/10.1016/j.ijleo.2017.08.028>
- [28] Mondal, B., Mandal, T. (2020). A secure image encryption scheme based on genetic operations and a new hybrid pseudo random number generator. Multimedia Tools and Applications, 79(25-26): 17497-17520. <https://doi.org/10.1007/s11042-019-08352-z>
- [29] Es-Sabry, M., El Akkad, N., Merras, M., Saaidi, A., Satori, K. (2022). A new color image encryption algorithm using multiple chaotic maps with the intersecting planes method. Scientific African, 16: e01217. <https://doi.org/10.1016/j.sciaf.2022.e01217>