

A Novel Cryptosystem Using Bitwise Permutations on Rows and Columns Followed by a Pseudo-Random Adapted Genetic Crossover

Mariam Jarjar¹, Hassan TABTI², Hicham Rrghout¹, Hamid El Bourakkadi¹, *Abdellatif Jarjar¹, Abdellhamid Benazzi¹

¹LSIA laboratory, Sidi Mohamed Ben Abdellah University, 30 000, Fez, Morocco

²MATSI Laboratory, Mohammed First University, 60 0000, Oujda, Morocco

ARTICLE INFO

ABSTRACT

Received: 29 Dec 2024

Revised: 12 Feb 2025

Accepted: 27 Feb 2025

Most image encryption algorithms operate at the pixel level, treating it as the primary element. However, bitwise permutations offer greater efficiency by altering both the pixel's value and its position within the image. In this study, we propose a novel color image encryption technique that applies multiple independent permutations to the rows and columns of the image after a Pseudo-Random transition to binary notation. Following a Pseudo-Random return to the grayscale level, a genetic crossover highly suited for color image encryption is applied to the resulting vector, integrating a substitution process using two S-Boxes and a diffusion mechanism to counter differential attacks. Extensive simulations on images of various sizes and formats confirm the robustness of our method against known attacks.

Keywords: Chaotic map; Broadcast function; P-Box; S-Box; Genetic Cross-over.

INTRODUCTION

Classical encryption systems (substitution, transposition, Vigenère, Hill, Feistel, etc.) have several shortcomings that make them vulnerable to modern cryptanalysis techniques. This is their main weakness:

Bit-level encryption is the fundamental building block of cryptography, linking mathematical concepts to software/hardware implementations. Its robustness depends on:

The quality of bit-by-bit operations (non-linearity, diffusion),

Key management,

Resistance to physical and software attacks.

The global Internet's rapid expansion and inherent insecurity mean that large volumes of data are constantly transmitted, leaving confidential information vulnerable to cyber threats. Recognizing the crucial and urgent need for information security, especially for data like images, which are valued for their clear communication, researchers have developed diverse encryption techniques to safeguard data exchange and transmission.

Recent advancements in number theory within mathematics provide researchers with novel opportunities for the design of image encryption algorithms rooted in chaos theory. Concurrently, a significant research effort is directed towards the enhancement of classical cryptographic techniques through the incorporation of chaotic elements. We have cited specific examples of such improvements implemented at the pixel level for the Vigenère, Hill, and Feistel ciphers [1-2-3-4]. Characterized by a public substitution matrix and a private key extended to the plaintext size, the classic Vigenère cipher demonstrated significant resistance to attacks until Babagh's advancements. Despite various attempts at improvement, the fundamental public matrix has generally remained unchanged [5-6-7-8].

While foundational to cryptography's history, classical encryption methods exhibit critical weaknesses against contemporary cryptanalysis, contrasting sharply with modern techniques. Substitution ciphers (e.g., Caesar, monoalphabetic) fail against frequency analysis by retaining language statistical patterns. Polyalphabetic ciphers

(e.g., Vigenère) are vulnerable to Chosen-plaintext attacks and short keys. Their small key spaces, exemplified by the Caesar cipher's 25 possibilities, render brute-force simple. Crucially, they lack diffusion and confusion, failing to produce the avalanche effect and leaving ciphertext patterns exposed. Designed for text, they are unsuited for modern binary data formats and succumb easily to advanced cryptanalysis powered by AI and high-performance computing. Therefore, they are no longer viable for securing sensitive data, unlike modern algorithms like AES, RSA, and ECC, which provide vastly improved security through larger keys, stronger attack resistance, and robustness against today's sophisticated computational power. Traditional encryption systems, primarily focused on text security, were vulnerable to statistical and frequency analysis. Their lack of chaining also exposed them to differential attacks, and small secret keys made them highly susceptible to brute-force attacks, revealing significant security limitations.

OBJECTIVES

The principal contribution of this research lies in an efficient, bit-level, action-centric methodology for image encryption. This is accomplished through the systematic application of distinct permutation operations on the rows and columns of the binary image. Finally, a genetic crossover mechanism, controlled by a Pseudo-Random crossover table, is employed to yield the final encrypted image, thereby establishing a robust and secure encryption framework.

METHODS

This section provides a comprehensive analysis of the proposed algorithm, which leverages multiple chaotic maps to enhance its functionality and security [9-10-11-12], This algorithm includes several steps.

Stage 1: Chaotic Sequences selection

To ensure synchronized sequence generation between transmitter and receiver, the proposed cryptographic system utilizes a Pseudo-Random number generator (PRNG) derived from three coupled chaotic maps. This PRNG exploits the three most widespread chaotic maps in cryptography, selected for their strong sensitivity to initial conditions and their seamless integration into the encryption process, providing a secure and efficient method for key generation.

1) The Logistics Map

The logistic map is a mathematical function from chaos theory, often used in cryptography to generate Pseudo-Random sequences. The logistic map (u_n) is a recurrent sequence described by a simple polynomial of second degree defined by the following equation:

$$\begin{cases} u_0 \in]0,5 \ 1[\quad , \quad \mu \in [3,75 \ 4] \\ u_{n+1} = \mu u_n(1 - u_n) \end{cases} \quad (1)$$

2) Piecewise Linear Chaotic Map (PWLCM)

The piecewise linear chaotic map (PWLCM) is a one-dimensional chaotic function that is widely used in cryptography due to its computational simplicity and diverse dynamic properties. It achieves an optimal balance between complexity and efficiency and is well suited for applications such as image encryption, key generation, and embedded security systems. This a real linear (w_n) sequence[13 – 14 – 15] by pieces defined by the equation below:

$$\{w_n = f(w_{n-1}) = \begin{cases} \frac{w_{n-1}}{d} & \text{if } 0 \leq w_{n-1} \leq d \\ \frac{w_{n-1}-d}{0.5-d} & \text{if } d \leq w_{n-1} \leq 0.5 \\ \text{Else} & \\ f(1 - w_{n-1}) & \end{cases} \quad (2)$$

3) The Skew Tent Map (SKTM)

Widely employed in cryptography, the Skew Tent Map (or asymmetric tent map) is a one-dimensional chaotic function valued for its mathematical simplicity, ergodic behavior, and strong dependence on initial values. Its characteristics are particularly beneficial for creating Pseudo-Random sequences and developing secure encryption algorithms. The Skew tent [16 – 17 – 18]map (v_n) will be redefined as the next equation:

$$v_0 \in]0, 1[\quad p \in]0, 5[\quad (3)$$

$$v_{n+1} = \begin{cases} \frac{v_n}{p} & \text{if } 0 < v_n < p \\ \frac{1-v_n}{1-p} & \text{if } p < v_n < 1 \end{cases}$$

These chaotic maps were chosen for our cryptosystem for the following reasons

- ✓ Sensitivity to Initial Conditions
- ✓ Larger key space
- ✓ Random Appearance
 - More complex chaotic behavior
 - Increased attack resistance
- ✓ Ease of Implementation

Improved ergodicity

4) Compare the three chaotic maps

Despite the reliability of the three chaotic maps, there are minimal differences between them, we quote:

Criteria	Logistic Map	PWLCM	Skew tent Map
Complexity	Quadratic (more complex)	Piecewise linear (simple)	Piecewise linear
uniformity	Possible bias for some r	Very uniform distribution	Good uniformity
Security	Vulnerability to weak parameters	Resists short cycle attacks	parameters Sensitive

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

Stage 2: Expanding encryption settings

For the optimal operation of our system, multiple subkeys need to be derived from the chaotic maps used.

1) Pseudo-Random table design

Our work entails constructing a (CT) chaotic table with coefficients in (G_{256}) of size $(24nm; 5)$, which is required for fusion and diffusion processes, and a (BT) binary table control of size $(3nm; 2)$. This setup can be seen in the diagram below:

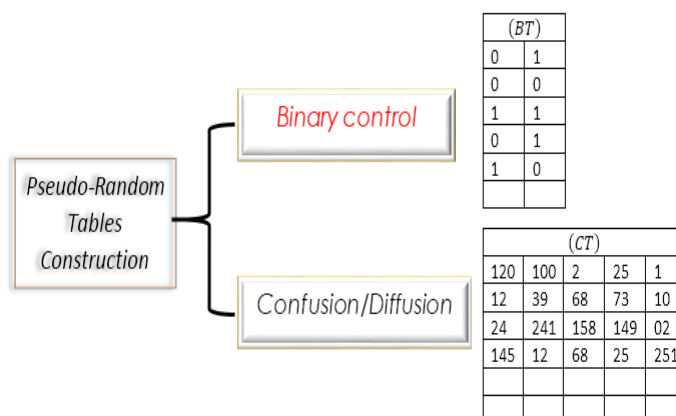


Figure1: Chaotics tables design

This figure is illustrated by the following algorithm:

Algorithm1:

-
1. For $i = 1$ to $24nm$
 - (CT) Confusion table in G_{256}
 2. $CT(i; 1) = \text{mod}(E(|u(i) - v(i)| * 10^{10}, 253) + 3)$
 3. $CT(i; 2) = \text{mod}(E(u(i) * w(i)) * 10^{10}, 253) + 2)$
 4. $CT(i; 3) = \text{mod}(i) = \text{mod}(E(u(i) * v(i) * 10^{11}, 253) + 1)$
 5. $CT(i; 4) = \text{mod}(E((u(i) + w(i) + v(i)) * 10^{10}, 253) + 1)$
 6. $CT(i; 5) = \text{sup}(CT(1; i); CT(4; i))$
 - (BT) Control table in G_2
 7. If $u(i) \geq v(i)$ then $BT(i; 1) = 0$ else $BT(i; 1) = 1$: end if
 8. If $w(i) \geq 0.5$ then $BT(i; 2) = 0$ else $BT(i; 2) = 1$: end if
 9. Next i
-

2) Building substitution tables

The Substitution S-Box constitutes a fundamental building block in symmetric encryption algorithms. Its role is to introduce non-linearity and enhance confusion, thereby bolstering the cipher's resilience against cryptanalytic attacks. A defining characteristic is that each of its rows represents an independent, nonlinear permutation. This encryption transformation signifies a profound improvement over the Vigenere and genetic techniques and requires the construction of two large S-Boxes, each with a size of (256; 256).

a. Convert a Pseudo-Random vector into permutation

The fact that every row in an S-Box is a permutation implies that creating this one-to-one mapping is done by sorting a specified non-Pseudo-Random vector. This process is exemplified below:

Example:

Rank	1	2	3	4	5	6	7	8	9	10	11
Value	8	23	21	8	5	9	32	25	14	7	9
range	9	3	4	8	11	7	1	2	5	10	6
Permutation	$Pr = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 9 & 3 & 4 & 8 & 11 & 7 & 1 & 2 & 5 & 10 & 6 \end{pmatrix}$										

b. (BW1) development

While the S-Box's non-linear nature is a cornerstone of modern cryptographic security, effectively disrupting predictable patterns and providing strong attack resistance, careful design remains paramount to eliminate exploitable vulnerabilities.

The construction of the substitution table (BW1) is determined by the following steps

i. First three rows Construction

- ✓ The first column is the permutation (P1) obtained by sorting on the first (256) values of the sequence (U).
- ✓ The first column is the permutation (P2) obtained by sorting on the first (256) values of the sequence (V)
- ✓ The first column is the permutation (P3) obtained by sorting on the first (256) values of the sequence (W)

Description:

- The first line of the table (BW1) is the permutation (P1)
- The second line of the table (BW1) is the permutation (P2)
- The third line of the table (BW1) is the permutation (P3)

This construction is performed by the algorithm below:

Algorithm2:
For $i = 1$ to 256
$BW1(1; i) = P1(i)$
$BW1(2; i) = P2(i)$

$$BW1(3;i) = P3(i)$$

$$\text{Next } i$$

j. Following rows Definition

According to the values of the vector obtained by the control vector (BT), the rank line ($i > 3$) is made up of the line ($i - 3$) and the line ($i - 2$) or the line ($i - 1$) by the line of rank ($i - 3$). The algorithm below serves as an explanation of this mechanism.

Algorithm3:

For $i = 3$ to 256
 For $j = 1$ to 256
 If $BT(i; 2) = 0$ Then

$BW1(i; j) = BW1(i - 3; BW1(i - 2; j))$
 Else
 $BW1(i; j) = BW1(i - 1; BW1(i - 1; j))$
 Next j, i

Example:

(SW)	1	2	3	4	5	6	7	8	CV
P1	2	5	1	8	3	6	4	7	0
P2	5	1	8	3	6	2	7	4	1
P3	6	5	7	1	3	2	4	8	1
P4	2	5	7	6	8	1	3	4	1
P5	5	3	4	2	8	6	7	1	0

c. ($BW2$) of size (256,256) Construction

To establish new confusion and diffusion mechanisms between the two substitution tables, a second S-Box is constructed. The creation of this replacement table ($BW2$) follows these steps:

i. The first-row construction

The first column is the permutation ($Q1$) obtained by sorting on the first (256) values of the sequence ($CT(: 5)$). The first line of the table ($BW2$) is the permutation ($Q1$)

Algorithm4:

For $i = 1$ to 256
 $BW2(1; i) = Q1(i)$
 Next i

j. The following rows Definition

The algorithm provided below describes the mechanism that explains how the row of rank ($i > 1$) in the control vector ($BT(: 1)$) is shifted based on the values of the vector. Depending on the value of $BT(: 2)$, the row is either a shift of row $CT(i; 2)$ from the previous row ($i - 1$) or a shift of row $CT(i; 4)$ from the previous line ($i - 1$).

Algorithm5:

For $i = 2$ to 256
 For $j = 1$ to 256
 If $BT(i; 1) = 0$ Then
 $BW2(i; j) = BW2(i - 1; j \oplus CT(i; 2))$

Else
 $BW2(i; j) = BW2(i - 1; j \oplus CT(i; 4))$
 Next j, i

In the novel encryption transformations, these two S-Boxes will function as the substitution tables.

d. Defining the substitution and diffusion function

These two S-Boxes serve as the foundation for the confusion functions, altering pixel values and enabling diffusion. This process creates a dependency chain between each encrypted pixel and the subsequent plaintext pixel, enhancing the avalanche effect and strengthening resistance against differential attacks. The following formulas define the new substitution and diffusion functions used in our proposed approach:

i. Confusion functions

The confusion function that provides Pseudo-Random substitution used in our approach is defined by the following formula

VG1 First Substitution function

$$VG1(X(i)) = Y(i) = BW1(CT(i; 2); BW2(CT(i; 4); X(i) \oplus CT(i; 3)))$$

VG2 second Substitution function

$$VG2(X(i)) = Y(i) = BW2(CT(i; 2); BW1(CT(i; 3); X(i) \oplus CT(i; 5)))$$

The confusion function applied is governed by the binary vector $B(: 1)$, as shown in the following formula

$$VG(X(i)) = \begin{cases} VG1(X(i)) & \text{if } BT(i, 1) = 0 \\ VG2(X(i)) & \text{if } BT(i, 1) = 1 \end{cases}$$

This confusion function is used to pseudo-randomly modify the value of the pixels in the image

j. Diffusion functions

The diffusion function that connects each encrypted pixel to the next clear pixel used in our algorithm is defined by the following formula

DF1 First diffusion function

$$DF1(X(i)) = BW1(CT(i; 1); Y(i - 1) \oplus X(i))$$

DF2 Second diffusion function

$$DF2(X(i)) = BW2(CT(i; 2); Y(i - 1) \oplus X(i))$$

Likewise, the diffusion function, which links the encrypted pixel to the next clear pixel, is controlled by the binary vector $B(: 2)$, as depicted in the following formula.

$$DF(X(i)) = \begin{cases} DF1(X(i)) & \text{if } BT(i, 2) = 0 \\ DF2(X(i)) & \text{if } BT(i, 2) = 1 \end{cases}$$

3) Building permutation tables

The P-Box (Permutation Box) is a fundamental component in symmetric encryption algorithms, particularly in block ciphers such as DES (Data Encryption Standard). Unlike the S-Box, which applies non-linear substitution, the P-Box performs a simple permutation of bits to ensure diffusion in the cipher. This transformation, which describes a profound improvement in the Vigenère technique, requires the construction of two ($PR1$) and ($PR2$) large S-Box of size ($4n; 6m$) each.

a. Line permutation table ($PR1$) design

The construction of the permutation line table ($PR1$) of size ($4n; 6m$) is determined by the following steps.

i. Four rows generation

- The first line is the permutation ($T1$) obtained by sorting on the first ($6m$) values of the sequence ($CT(: 3)$).
- The first line is the permutation ($T2$) obtained by sorting on the first ($6m$) values of the sequence ($CT(: 5)$).
- The first line is the permutation ($T3$) obtained by sorting on the first ($6m$) values of the sequence $CT(: 1)$
- The four line is the permutation ($T4$) obtained by sorting on the first ($6m$) values of the sequence $CT(: 2)$

Algorithm6:

For $i = 1$ to $6m$	$PR1(3; i) = T3(i)$
$PR1(1; i) = T1(i)$	$PR1(4; i) = T4(i)$
$PR1(2; i) = T2(i)$	Next i

j. Following rows Definition

According to the values of the vector obtained under the control vector ($BT(:2)$), the rank line ($i > 4$) is made up of the line ($i - 1$) and the line ($i - 3$) or the line ($i - 4$) by the line ($i - 2$). The algorithm below serves as an explanation of this mechanism.

Algorithm7:	
For $i = 5$ to $6m$	$PR1(i; j) = PR1(i - 1; PR1(i - 3; j))$
For $j = 1$ to $4n$	Else
If $BT(i; 2) = 0$ Then	$PR1(i; j) = PR1(i - 4; PR1(i - 2; j))$
	Next j, i

b. Columns permutation table (PR2) design

The construction of the permutation columns table ($PR2$) of size ($6m; 4n$) is determined by the following steps.

i. Four rows generation

- ✓ The first line is the permutation ($K1$) obtained by sorting on the first ($4n$) values of the sequence ($CT(:1)$).
- ✓ The first line is the permutation ($K2$) obtained by sorting on the first ($4n$) values of the sequence ($CT(:3)$).
- ✓ The first line is the permutation ($K3$) obtained by sorting on the first ($4n$) values of the sequence ($CT(:5)$).
- ✓ The four line is the permutation ($K4$) obtained by sorting on the first ($4n$) values of the sequence ($CT(:4)$).

Algorithm8:	
For $i = 1$ to $4n$	$PR2(3; i) = K3(i)$
$PR2(1; i) = K1(i)$	$PR2(4; i) = K4(i)$
$PR2(2; i) = K2(i)$	Next i

j. Following rows Definition.

According to the values of the vector obtained by the control vector ($BT(:1)$), the rank line ($i > 3$) is made up of the line ($i - 1$) and the line ($i - 3$) or the line ($i - 2$) by the line ($i - 1$). The algorithm below serves as an explanation of this mechanism.

Algorithm9:	
For $i = 5$ to $4n$	$PR2(i; j) = PR2(i - 1; PR2(i - 3; j))$
For $j = 1$ to $6m$	Else
If $BT(i; 1) = 0$ Then	$PR2(i; j) = PR2(i - 2; PR2(i - 4; j))$
	Next j, i

Stage 4: Preparing the image to encrypt

the original image must be prepared before any passage to the encryption operating room

1) Convert original image to vector

The three-color channels (RGB) are first extracted from the original image and transformed into three vectors: (V_r), (V_g), and (V_b), each of size ($1, nm$). These vectors are then concatenated and processed through a confusion step using a Pseudo-Random table, under the control of the decision vector $BT(:2)$. This process results in the generation of a lightly encrypted vector $X(x_1, x_2, \dots, x_{3nm})$ of size ($1, 3nm$). The following diagram illustrates this operation.

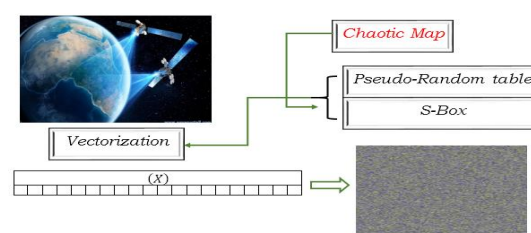




Figure2: Original Image vectorization

This pattern is interpreted by the algorithm below:

Algorithm10:

1. for $i = 1$ to nm
2. If $BT(i; 2) = 0$ Then
3. $X(3i - 2) = VG(Vb(i) \oplus CT(3i; 4))$
4. $X(3i - 1) = VG(Vr(i) \oplus CT(3i - 1; 3))$
5. $X(3i) = VG(i) \oplus CT(3i; 2)$
6. Else
7. $X(3i - 2) = Vr(i) \oplus CT(3i - 1; 4)$
8. $X(3i - 1) = Vb(i) \oplus CT(3i - 2; 5)$
9. $X(3i) = VG(Vg(i) \oplus CT(3i; 3))$
10. End if
11. Next i

This initial encryption attempt produced a lightly secured image that is vulnerable to statistical and frequency attacks. This result can be seen by the next table:

Original Image	Cipher image	Parameters values		
		Vertical correlation	R	0.00320
			G	-.00036
			B	-.00150
		Horizontal correlation	R	0.00016
			G	0.00184
			B	-.00237
		Diagonal correlation	R	-.00055
			G	0.00072
			B	0.00325
		Entropy	R	7.99870
			G	7.99873
			B	7.99876

To protect the cryptosystem against any differential attack, another round of encryption is required

Stage 5: Transform the resulting image for encoding

The resulting image from this initial phase, will be subjected to the following transformations to achieve a higher level of encryption

1) Switching to binary

The vector (X) is converted into a binary vector (XB) using the four transformation tables (TR1), (TR2) (TR3), TR4) listed below

(TR1)		
	0	1
0	0	0
1	0	1
2	1	0
3	1	1

(TR2)		
	0	1
0	0	0
1	1	1
2	1	0
3	0	1

(TR3)		
	0	1
0	0	1
1	0	0
2	1	1
3	1	0

(TR4)		
	0	1
0	1	0
1	1	1
2	0	0
3	0	1

and under the vector control (CP) defined by the following algorithm

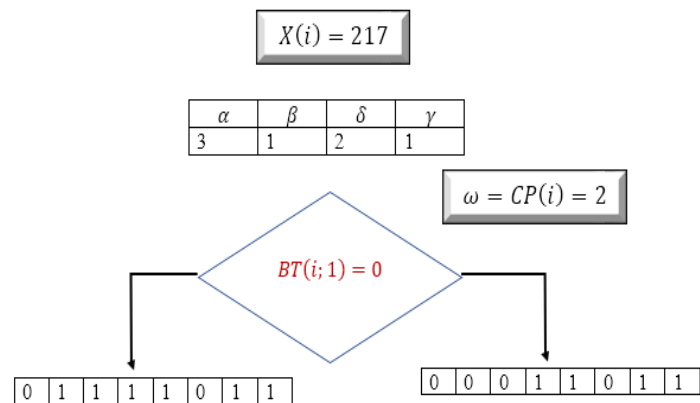
Algorithm11:

- | | |
|---------------------------------------|---------------------------------------|
| For $i = 1$ to $24nm$ | Else |
| If $BT(i; 1) = 0$ Then | $CP(i) = \text{Mod}(CT(i; 3); 4) + 1$ |
| $CP(i) = \text{Mod}(CT(i; 4); 4) + 1$ | Next i |

The algorithm below details the Pseudo-Random conversion of vector (X) into a binary vector (XB), controlled by the decision vector (CP)

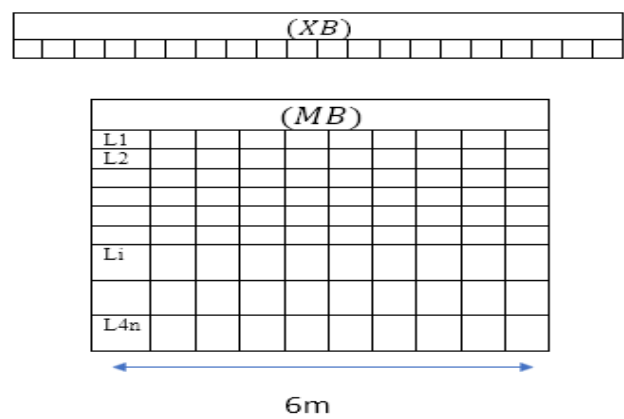
Algorithm12:		
For $i = 1$ to $3nm$	$\omega = CP(i)$	Else
$x = E\left(\frac{X(i)}{16}\right)$	If $BT(i; 1) = 0$ then	$XB(8 * i - 7) = T\omega(\delta, 1)$
$y = X(i) - 16 * x$	$XB(8 * i - 7) = T\omega(\beta, 1)$	$XB(8 * i - 6i) = T\omega(\delta, 2)$
$\alpha = E\left(\frac{x}{4}\right)$	$XB(8 * i - 6i) = T\omega(\beta, 2)$	$XB(8 * i - 5) = T\omega(\alpha, 1)$
$\beta = x - 4 * \alpha$	$XB(8 * i - 5) = T\omega(\delta, 1)$	$XB(8 * i - 4) = T\omega(\alpha, 2)$
$\gamma = E\left(\frac{y}{4}\right)$	$XB(8 * i - 4) = T\omega(\delta, 2)$	$XB(8 * i - 3) = T\omega(\gamma, 1)$
$\delta = y - 4 * \gamma$	$XB(8 * i - 3) = T\omega(\alpha, 1)$	$XB(8 * i - 2) = T\omega(\gamma, 2)$
	$XB(8 * i - 2) = T\omega(\alpha, 2)$	$XB(8 * i - 1) = T\omega(\beta, 1)$
	$XB(8 * i - 1) = T\omega(\gamma, 1)$	$XB(8 * i) = T\omega(\beta, 2)$
	$XB(8 * i) = T\omega(y, 2)$	Next i

Example:



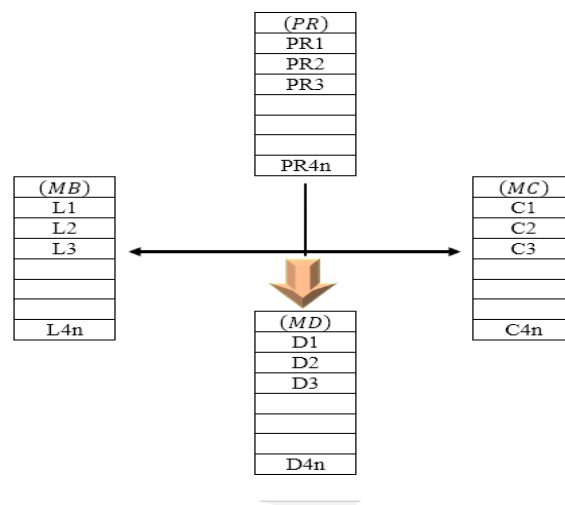
2) Switching to (MB) binary matrix

The vector (XB) is transformed into a binary matrix (MB) of size $(4n; 6m)$. This passage is illustrated by the following figure:



a. Permutation on (MB) lines

The vector $BT(: 2)$ is also converted into a binary matrix (MC) of size $(4n; 6m)$. The row permutation applied to the matrix (MB) is depicted in the figure below.



This transformation is determined by the following algorithm:

First line transformation

$$D1 = PR1(L1) \oplus C1$$

subsequent lines Transformation

$$Di = PRi(Di - 1 \oplus Li) \oplus Ci$$

Example:

(PR)									
PR1	3	5	1	6	8	4	2	7	
PR2	2	1	8	5	7	4	6	3	
PR3	5	6	3	1	8	4	7	2	
PR4	5	3	8	2	7	4	6	1	
PR5	8	1	5	2	6	4	7	3	

$$D1 = PR1(L1) \oplus C1$$

$$Di = PRi(Di - 1 \oplus Li) \oplus Ci$$

(MB)									
L1	0	0	1	1	1	0	1	1	
L2	1	1	0	0	0	1	1	0	
L3	0	1	1	0	0	1	1	0	
L4	1	1	1	0	1	1	1	0	
L5	0	0	0	1	0	1	1	0	

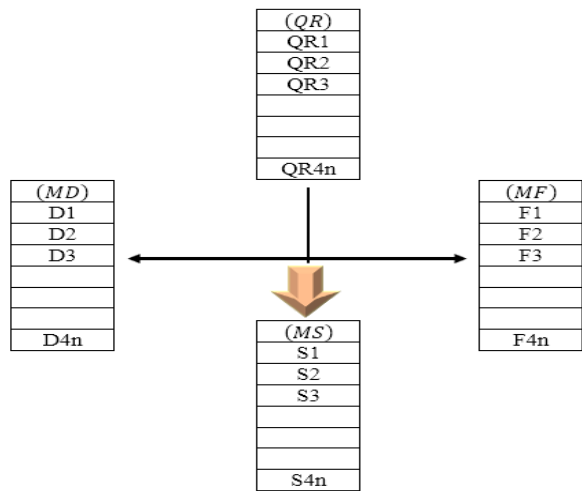
(MC)									
C1	1	0	0	1	1	0	1	0	
C2	1	1	0	0	0	1	0	1	
C3	0	0	1	1	0	1	0	1	
C4	1	1	0	1	1	1	0	0	
C5	0	0	0	1	1	1	0	0	

(MD)									
D1	0	1	0	1	0	0	1	1	
D2	1	0	0	1	0	0	0	1	
D3	0	1	0	0	1	0	1	0	
D4	1	0	1	0	1	1	0	1	
D5	1	1	1	1	1	0	1	1	

b. Permutation on (MD) column

The vector $BT(:1)$ is also converted into a binary matrix (MV) of size $(6m; 4n)$. The column permutation applied to the matrix (MD) in order to obtain the matrix (MS) .

Determine the transposed matrix of the matrix (MD) of dimension $(6m; 4n)$, and we carry out the same treatment on the lines of this new matrix by applying the P-Box (QR) is depicted in the figure below



First line transformation

$$S1 = QR1(L1) \oplus F1$$

subsequent lines Transformation

$$Si = QRi(Di - 1 \oplus Di) \oplus Fi$$

The resulting matrix (MS) is transformed into a binary vector (XC) of dimensions $(1, 24nm)$.

3) Grayscale Transition

The transition to grayscale goes through the following steps:

a. Switching to $\mathbb{Z}/4\mathbb{Z}$

The vector (XC) is divided into two subvectors $(XC1), (XC2)$, each of size $(1; 12nm)$. A cross-section table $(TD1)$ of size $(12nm; 3)$ is generated using the following procedure:

The first column is a permutation $(Hr1)$ obtained by a crossing sort on the $12nm$ value of the sequence $(CT(: 3))$

This column indicates the bit index of the vector $(XC1)$

The second column is a permutation $(Hr2)$ obtained by a crossing sort on the $12nm$ value of the sequence $(CT(: 1))$

This column indicates the bit index of the vector $(XC2)$

The third column is a permutation $(Hr3)$ obtained by a crossing sort on the $12nm$ value of the sequence $(CT(: 5))$

This column indicates de l'emplacement du nombre calculé au sein du vecteur (XV) généré.

This construction technique is given by the following algorithm

Algorithm13:	
For $i = 1$ to $12nm$	$TD(i; 3) = Hr3(i)$
$TD(i; 1) = Hr1(i)$	Next i
$TD(i; 2) = Hr2(i)$	

The generation of the vector (XV) is given by the algorithm below:

Algorithm14:	
For $i = 1$ to $12nm$	$XV(TD(i; 3)) = XC1(TD(i; 1)) * 2 + XC2(TD(i; 2))$
If $BT(i; 2) = 0$ Then	$XV(TD(i; 3)) = XC2(TD(i; 2)) * 2 + XC1(TD(i; 1))$
	Next i

Example:

(XC1)			
0	1	1	0

(XC2)			
0	1	1	0

(TD)

(XV)			
3	2	0	1

Index		Place	BT(:2)
(XC1)	(XC2)	(XV)	
3	3	4	0
1	1	3	1
4	2	2	1
2	4	1	0

b. Switching to $\mathbb{Z}/256\mathbb{Z}$

The vector (XV) is divided into four subvectors (XV1), (XV2), (XV3), (XV4) each of size $(1; 3nm)$. A cross-section table (TE) of size $(3nm; 5)$ is generated using the following procedure:

- ❖ The first column is a permutation (Gr1) obtained by a crossing sort on the $3nm$ value of the sequence $(CT(:1))$
 - This column indicates the bit index of the vector (XV1)
- ❖ The second column is a permutation (Gr2) obtained by a crossing sort on the $3nm$ value of the sequence $(CT(:3))$
 - This column indicates the bit index of the vector (XV2)
- ❖ The third column is a permutation (Gr3) obtained by a crossing sort on the $3nm$ value of the sequence $(CT(:2))$
 - This column indicates the bit index of the vector (XV3)
- ❖ The third column is a permutation (Gr4) obtained by a crossing sort on the $3nm$ value of the sequence $(CT(:5))$
 - This column indicates the bit index of the vector (XV4)
- ❖ This column indicates de l'emplacement du nombre calculé au sein du vecteur (XS) généré

This construction technique is given by the following algorithm

Algorithm15:

For $i = 1$ to $13m$	$TE(i; 4) = Gr3(i)$
$TE(i; 1) = Gr1(i)$	$TE(i; 5) = Gr4(i)$
$TE(i; 2) = Gr2(i)$	Next i
$TE(i; 3) = Gr3(i)$	

The generation of the vector (XS) is given by the algorithm below:

1. Algorithm16:

2. For $i = 1$ to $13m$
3. If $BT(i; 1) = 0$ Then
4. $XS(TE(i; 5)) = XV1(TE(i; 1)) * 64 + XV2(TE(i; 2)) * 16 + XV3(TE(i; 2)) * 4 + XV4(TE(i; 2))$
5. Else
6. $XS(TE(i; 5)) = XV1(TE(i; 1)) * 16 + XV2(TE(i; 2)) * 4 + XV4(TE(i; 2)) * 64 + XV3(TE(i; 2))$
7. $XS(TE(i; 5)) = VG(XS(TE(i; 5)) \oplus CT(i; 3))$
8. Next i

This initialization value is closely related to the control vector $BT(:2)$.

4) The initialization value:

To thwart differential attacks, an additional phase is implemented. This involves obtaining an initialization value from the (XS) vector and applying it to modify solely the seed pixel, which in turn initiates the encryption process:

Algorithm17:

$IK = 0$	Else
	$IK = XS(i) \oplus IK \oplus CT(i; 1)$

For $i = 2$ to $3nm$ If $BT(i; 1) = 0$ Then $IK = XS(i) \oplus IK \oplus CT(i; 5)$	Next i
--	----------

This initialization value is closely related to the control vector $BT(: 1)$.

5) The final round

By applying the new Vigenère function to the initialization value, we can modify the first pixel's value and start the encryption process. The improved encryption process, utilizing only Vigenère, is depicted in the figure below:

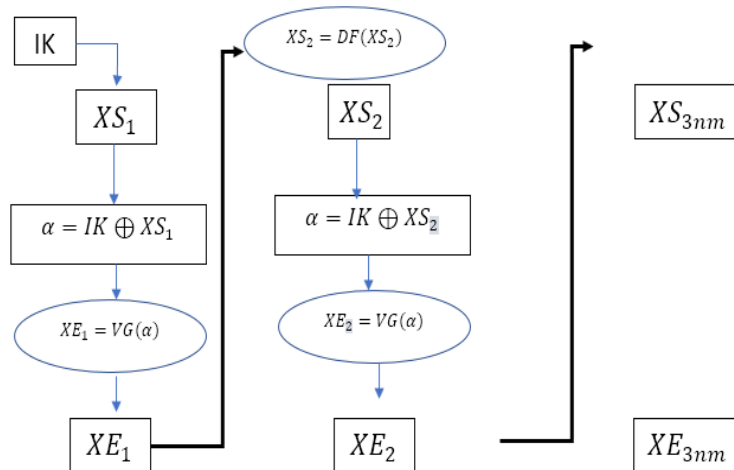


Figure3: Image encryption process

The vector (XE) represents the image encrypted by our new technology. This encryption scheme is given by the following algorithm:

Algorithm18

- | | |
|--|---|
| 1. First pixel encryption
2. $X'(1) = VG(X(1) \oplus IV) \oplus CT(2; 1)$
3. Next Pixel Encryption
4. For $i = 2$ to $3nm$
5. $\Pi(X(i)) = VG(X'(i-1) \oplus KL(i)) \oplus X(i)$ | 6. If $CR(i) = 0$ Then
7. $X'(i) = VG(\Pi(X(i))) \oplus KR(i)$
8. else
9. $X'(i) = VG(\Pi(X(i))) \oplus CL(i)$
10. Next i |
|--|---|

Step 5: Decryption encrypted images

Utilizing a symmetric encryption scheme with a confusion/diffusion framework, our method ensures accurate decryption by applying the inverse functions of the encryption process, beginning with the final encrypted block. The inherent reversibility of all functions within our scheme makes this possible. The decryption process follows

1) Vigenere Reciprocal matrix

Let:

- ✓ $(WB1)$ be the inverse substitution table of $(BW1)$
- ✓ $(WB2)$ be the inverse substitution table of $(BW2)$

These two tables are generated by the following algorithm

Algorithm19

1. for $i = 1$ to 256
2. for $j = 1$ to 256
3. $WB1(i, BW1(i, j)) = j$
4. $WB2(i, BW2(i, j)) = j$
5. Next j, i

Example:

(VG)	1	2	3	4	5	6	7	0
1	3	5	0	6	2	7	1	4
2	2	7	1	4	3	5	0	6
3	4	3	5	0	6	2	7	1
4	2	7	1	4	3	5	0	6
5	0		2	7	1	4	3	5

(GV)	1	2	3	4	5	6	7	0
1	7	5	1	0	2	4	6	3
2	3	1	5	4	6	0	2	7
3	0	6	2	1	3	5	7	4
4	3	1	5	4	6	0	2	7
5	5	3	7	6	0	2	4	1

By following the same logic of Vigenere's traditional technique, we obtain

Algorithm20:

if $z = VG(y, x)$

Then

$x = GV(y, z)$

2) Vigenere's inverse expression

First invers Substitution function

$$(GV1)^{-1}(X(i)) = WB2(CT(i; 2); WB1(CT(i; 4); X(i)) \oplus CT(i; 3))$$

(GV2)⁻¹ second Substitution function

$$(GV2)^{-1}(X(i)) = WB1(CT(i; 3); WB2(CT(i; 2); X(i)) \oplus CT(i; 5))$$

(4)

The confusion function applied is governed by the binary vector $B(: 1)$, as shown in the following formula

$$(GV)^{-1}(X(i)) = \begin{cases} (GV1)^{-1}(X(i)) & \text{if } BT(i, 1) = 0 \\ (GV2)^{-1}(X(i)) & \text{if } BT(i, 1) = 1 \end{cases} \quad (5)$$

3) Inverse of the scattering function

The inverse of the diffusion function used in our system is given by the following formula

First invers diffusion function

$$(DF1)^{-1}(X(i)) = WB1(CT(i; 1); X(i)) \oplus Y(i - 1)$$

Second invers Substitution function

$$(DF2)^{-1}(X(i)) = WB2(CT(i; 2); X(i)) \oplus Y(i - 1)$$

(6)

The diffusion function applied is governed by the binary vector $B(: 2)$, as shown in the following formula.

$$(DF)^{-1}(X(i)) = \begin{cases} (DF1)^{-1}(X(i)) & \text{if } BT(i; 2) = 0 \\ (DF2)^{-1}(X(i)) & \text{if } BT(i; 2) = 1 \end{cases}$$

RESULTS

To analyze the security of our cryptosystem, we will compute various statistical and differential parameters. These metrics will allow us to evaluate the system's strength against statistical and differential attacks and its overall resistance to cryptanalysis. In this section, our new algorithm will be applied to a large set of randomly selected images from a comprehensive database. We will then demonstrate its performance and compare it with that of other existing technologies [28-29], keeping in mind that a truly effective algorithm is one that can withstand all known attack methods.

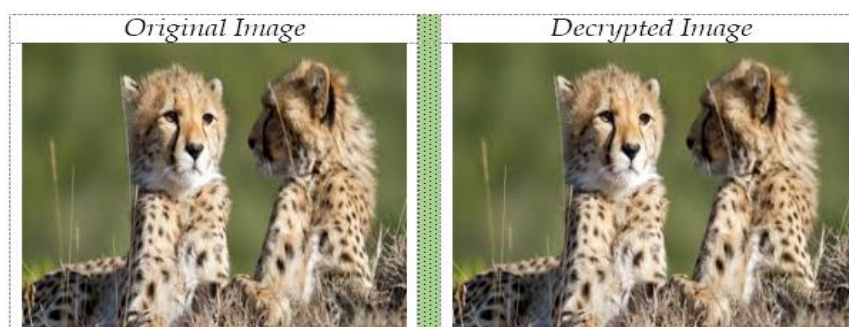
1) Comparison between the original and decrypted image

Assessing the similarity between an original image and its deciphered counterpart involves applying different evaluation methods based on the specific aspects under scrutiny, such as visual resemblance, image quality, and data fidelity. The techniques outlined below will be utilized:

a. Visual Comparison

Manual Inspection: Perform a side-by-side visual examination of the two images to identify any notable differences. While this method provides a qualitative assessment, it lacks precision. However, it is a good first step in the evaluation process.

The table below contains the original image and the decrypted image.



The visual test detects no apparent difference between the two images.

Overlay Method: By superimposing the two images using software like Photoshop, no visible differences were detected between them.

2) Image Quality Metrics

PSNR (Peak Signal-to-Noise Ratio): Measures the quality of the decrypted image relative to the original. A higher PSNR indicates better quality

Where:

MAX: is the maximum possible pixel value (e.g., 255 for an 8-bit image)

MSE: is the mean square error between the two images.

For two images:

I (original image) and K (decrypted or reconstructed image) of size $m \times n$, the MSE is calculated as follows:

$$\text{Equation 7}$$

$$MSE = \frac{1}{nm} \sum_{i=1; j=1}^{mn} (I(i; j) - K(i; j))^2$$

- $I(i, j)$: Pixel value at position in the original image.
- $K(i, j)$: Pixel value at position in the deciphered or reconstructed image.
- $m * n$: Total number of pixels in the image (height* width)

a. MSE Properties

Minimum Value: The MSE is always non-negative. A value of (0) indicates that the two images are identical at 100%.

Sensitivity to Large Errors: Since differences are squared, MSE is highly sensitive to large deviations between pixel values.

Unit: MSE is expressed in squared units of pixel intensity (e.g., for 8-bit pixel values, MSE will be in squared intensity levels).

In our approach, in all simulations, the calculation of the MSE is

$$MSE \approx 0$$

Therefore, the two images are almost identical at 99,99%

i. MSE Limitations

Ignores Human Perception: MSE treats all pixel differences equally, even if some variations are imperceptible to the human eye.

Highly Sensitive to Outliers: A single large pixel difference can disproportionately increase the MSE value.

Does Not Capture Structural Differences: MSE fails to account for image texture and structural variations, which are crucial for perceptual quality assessment.

b. SSIM computation

SSIM (Structural Similarity Index) is a metric that evaluates the similarity between two images by analyzing three key components: luminance, contrast, and structure. It is specifically designed to align with human visual perception of image quality. Below is a detailed breakdown of the SSIM formula.

$$SSIM(I; K) = \frac{(2\mu_I\mu_K + C_1) * (2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1) * (\sigma_I^2 + \sigma_K^2 + C_2)} \quad (8)$$

- ✓ I: Original Image
- ✓ K: Decrypted image
- ✓ μ_I and μ_K : are the averages of the pixel intensities of the images I and K
- ✓ σ_I^2 and σ_K^2 : are the variances of the pixel intensities of the images I and K
- ✓ σ_{IK} : Est la covariance entre les intensités des pixels des images I and K
- ✓ C_1 and C_2 : are stabilization constants to avoid divisions by zero.

i. Stabilization constants

$$\begin{cases} C_1 = (k_1 L)^2 \\ C_2 = (k_2 L)^2 \end{cases}$$

where L is the dynamic range of pixel values (e.g., 255 for an 8-bit image), and

$$\begin{cases} k_1 = 0,01 \\ k_2 = 0,03 \end{cases}$$



In all our simulation the value of the statistical constant SSIM is

$$SSIM = 0,9993 \approx 1$$

In practice, an SSIM value close to 1 indicates strong visual similarity. Therefore, the two images are similar.

3) Image Texture Analysis

To evaluate the differences in texture; the spatial variations in pixel intensity that create visual patterns like smoothness, roughness, or graininess; between two images, a variety of methods are available, spanning simple statistical analysis to sophisticated deep learning. Some commonly employed statistical approaches include:

a. Statistical Methods

These techniques evaluate the statistical characteristics of textures to facilitate comparison.

b. Gray Level Histogram:

- ✓ Compute the histogram of pixel intensities for each image.
- ✓ Compare the histograms using metrics like Bhattacharyya distance, correlation, or Kullback-Leibler divergence.
- ✓ While straightforward, this method does not account for spatial relationships within the texture.

c. Grayscale Co-occurrence Matrix (GL)

(GL) quantifies how frequently pairs of pixels with specific intensity values occur at a defined spatial distance and angle. From the (GL), various texture features can be extracted, including:

i. Calculation of the Co-Occurrence Matrix (GL)

The Gray Level Co-occurrence matrix (GL) of size (256; 256); is crucial for analyzing the texture of a digital image. It captures the spatial relationships between pixels based on their gray levels and their spatial configuration. The co-occurrence matrix characterizes the texture of an image by examining how pixels of a particular grey level are positioned next to each other in particular directions. It captures the spatial structure of the image by analyzing the relationships between adjacent pixels.

Example:

Let's take a grayscale image represented by the following matrix:

$$K = \begin{pmatrix} 0 & 1 & 2 & 2 & 3 & 1 \\ 1 & 0 & 1 & 3 & 2 & 2 \\ 2 & 1 & 0 & 1 & 3 & 1 \\ 0 & 3 & 1 & 0 & 2 & 2 \\ 2 & 1 & 3 & 2 & 0 & 1 \\ 1 & 2 & 1 & 3 & 1 & 0 \end{pmatrix}$$

There are four gray levels, therefore the matrix (GL) will be of dimension (4; 4)

(GL)	0	1	2	3
0	0	4	1	1
1	3	2	1	3
2	1	1	3	1
3	0	2	1	0

Therefore, the co-occurrence matrix (GL) is

$$GL = \begin{pmatrix} 0 & 4 & 1 & 1 \\ 3 & 2 & 1 & 3 \\ 1 & 1 & 3 & 1 \\ 0 & 2 & 1 & 0 \end{pmatrix}$$

The weight of the matrix (GL) of co-occurrence is the sum of all its components

$$PT = \sum_{i=1; j=1}^{16} GL(i; j) = 24$$

The normalized matrix is

$$GL = \frac{1}{24} \begin{pmatrix} 0 & 4 & 1 & 1 \\ 3 & 2 & 1 & 3 \\ 1 & 1 & 3 & 1 \\ 0 & 2 & 1 & 0 \end{pmatrix}$$

Below are the key roles and steps of the co-occurrence matrix in image analysis:

d. Gray Level Co-Occurrence Matrix (GL) Properties

(GL) measures how often pair of pixels with specific intensities appear at a given distance and angle.

From the (GL), you can extract texture features such as:

Contrast: Measures the local variation in intensity.

Energy: Measures the uniformity of the texture.

Homogeneity: Measures how closely the distribution of *GLCM* elements is to its diagonal.

Correlation: Measures the linear dependence of pixel intensities.

Compare these features between the two images.

i. The contrast

Contrast is one of the most commonly extracted texture features from Gray-Level Co-occurrence Matrix (*GL*). It measures the local variation in pixel intensity in an image, which helps quantify the "roughness" or "graininess" of the texture. Here's a detailed explanation of how contrast is calculated from (*GL*), The Contrast is calculated from the normalized *GLCM* using the following formula

$$C(GL) = \sum_{i=1; j=1}^{256} (i - j)^2 GL(i, j) \quad (9)$$

In our simulation



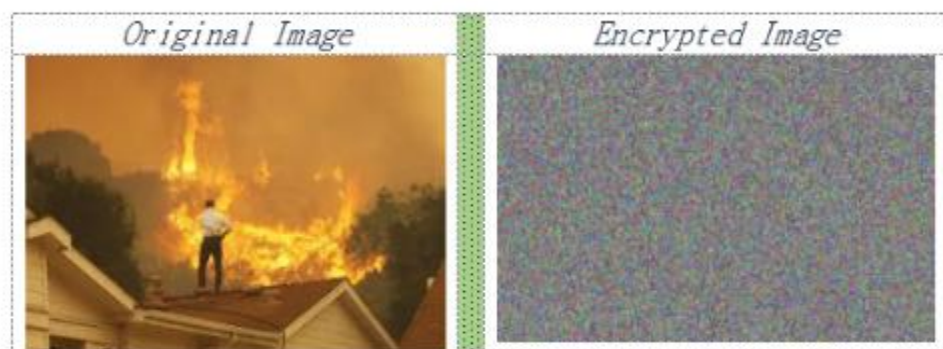
$$C(GL) = 0,0131 \approx 0$$

Low value: Indicates low local variation in intensity (smooth or uniform texture).

j. The Energy:

Energy is a texture feature derived from the Gray-Level Co-occurrence Matrix (*GL*) that quantifies the uniformity or consistency of an image's texture. A high energy value indicates a highly uniform texture, whereas a low energy value signifies greater variation and complexity. Below is a detailed explanation of how energy is computed from *GLCM*. The Energy is calculated from the normalized *GLCM* using the following formula

$$E(GL) = \sum_{i=1; j=1}^{256} (GL(i; j))^2 \quad (10)$$



$$E(GL) = 0,0210 \approx 0$$

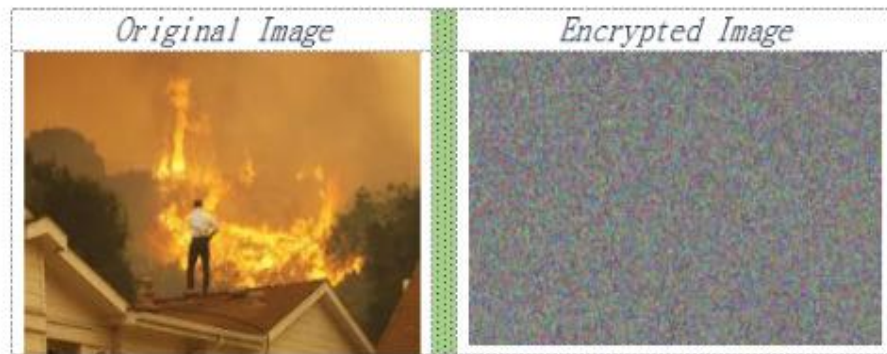
Low Value: Suggests a diverse or irregular texture, where (*GL*) values are more evenly spread out.

k. Homogeneity:

Homogeneity, also known as the inverse difference moment, is a texture feature derived from the Gray-Level Co-occurrence Matrix (*GL*). It quantifies the closeness of (*GL*) values to its diagonal, reflecting texture uniformity. A

high homogeneity value indicates a highly uniform texture, while a low value suggests greater texture variability. Below is a detailed explanation of how homogeneity is computed from the (GL). The Homogeneity is calculated from the normalized *GLCM* using the following formula:

$$H(GL) = \sum_{i=1; j=1}^{256} \frac{GL(i; j)}{1 + |i - j|} \quad (11)$$



$$H(GL) = 0.843 > 0.8$$

- ✓ Smooth texture
- ✓ Few local variations
- ✓ Pixel pairs have partially similar gray levels.
- ✓ The GLCM is partially focused on the diagonal.

4) The contours

An edge in an image represents an abrupt transition in intensity levels between adjacent regions. It generally corresponds to the boundaries of objects present in the image.

a. Characteristics of an Edge:

Intensity Variation: Edges occur where pixel intensity changes rapidly.

Direction and Orientation: Edges can be horizontal, vertical, or diagonal, depending on the direction of intensity change.

Thickness and Sharpness:

- ✓ A sharp, thin edge indicates a sudden transition (e.g., printed text).
- ✓ A wide, blurred edge suggests a gradual transition (e.g., shading or soft lighting).

Example:

$$K = \begin{pmatrix} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 50 & 50 & 10 \\ 10 & 10 & 50 & 200 & 200 \\ 10 & 10 & 50 & 10 & 50 \\ 10 & 10 & 10 & 50 & 50 \end{pmatrix}$$

Moving from 10 to 50 to 200 creates a visible outline between these regions, which ensures the presence of a contour

b. Edge detection of the encrypted image

Detecting edges in an encrypted image is difficult because encryption transforms pixel values, making the image visually unrecognizable. However, if the decryption key is available, standard edge detection techniques can be applied to the decrypted image.

If the image remains encrypted, direct edge detection is generally infeasible, as encryption obscures structural details. Below is a two-step approach for detecting edges in an encrypted image:

Local Processing:

Apply local filters, such as a mean or median filter, to attempt noise reduction caused by encryption. This approach is only effective if the encryption retains some local structural properties. Detecting edges in an encrypted image using a hand-held filter provides a practical way to understand convolution operations in image processing. A randomly selected sub-matrix from the encrypted image is chosen, and Sobel filters are applied manually using a convolution matrix

i. Sobel Filter

The Sobel filter is commonly used to detect edges in an image. It uses two kernels (masks) to detect horizontal and vertical gradients:

Horizontal Mask (Vertical Edge Detection):

$$G_x = \begin{pmatrix} -1 & 0 & 2 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Vertical mask (horizontal contour detection):

$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

The total gradient is calculated as follows:

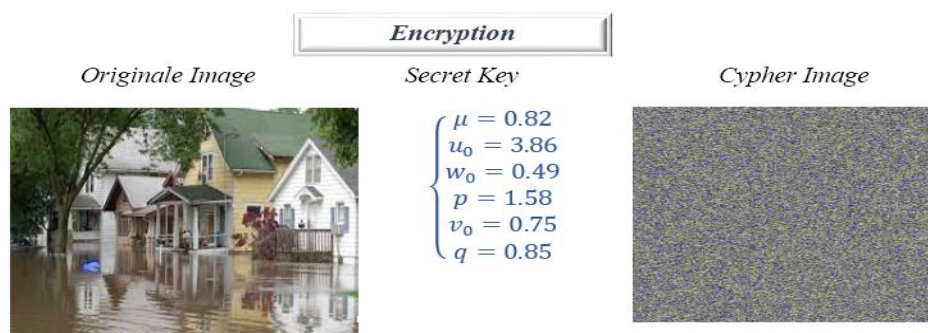
$$G = \sqrt{G_x^2 + G_y^2}$$

Interpretation:

1. $G_x = 0$ then Horizontal contours missing
2. $G_y = 0$ then Vertical contours missing
3. $G = 0$ then contours missing

c. Uniform encrypted image distribution using Lorenz Diagram

To assess the uniform distribution of pixels in an image encrypted by our algorithm, we will generate the Lorenz curve and compute the Gini index for a reference encrypted image using this novel approach. In image cryptography, a properly encrypted image should exhibit a low Gini index (*close to 0*), signifying a uniform pixel distribution, and a Lorenz curve that closely follows the diagonal. This guarantees the absence of any discernible structure in the encrypted image, significantly complicating attacks based on statistical analysis.



We divide the pixel values into intervals of size 25, resulting in the first following statistical table

Intensity interval	Pixel count	Center	Accumulats	Frequency	Accumulats
	(<i>ni</i>)	(<i>xi</i>)	(<i>Xi</i>)	(<i>fi</i>)	(<i>Fi</i>)
[0 25[5000	12.5	5000	0.0763	0.0763
[25 50[7000	37.5	12000	0.1831	0.2594
[50 75[8000	62.5	20000	0.3052	0.5645
[75 100[9000	87.5	29000	0.4425	1.0071
[100 125[8500	114.5	37500	0.5722	1.5793
[125 150[7500	137.5	45000	0.6866	2.2659
[150 175[6000	162.5	51000	0.7782	3.0441
[175 200[5000	187.5	56000	0.8545	3.8986
[200 225[4500	212.5	60500	0.9232	4.8218
[225 256[5036	237.5	65536	1	5.8218

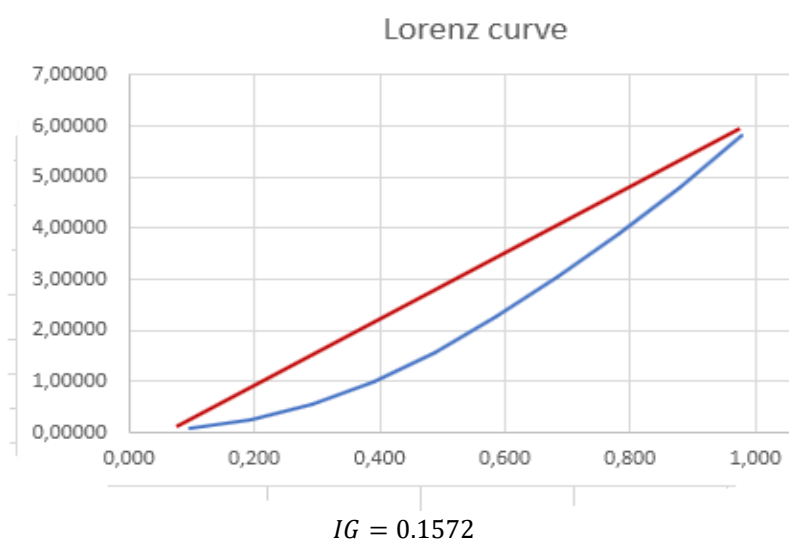
The second statistical table

Pixel count	Center	Accumulats	Frequency	Accumulats
(<i>ni</i>)	(<i>xi</i>)	(<i>nixi</i>)	(<i>qi</i>)	(<i>Qi</i>)

5000	12.5	62500	0,00814	0,00814
7000	37.5	262500	0,04231	0,05044
8000	62.5	500000	0,10740	0,15784
9000	87.5	787500	0,20991	0,36775
8500	114.5	973250	0,33661	0,70436
7500	137.5	1031250	0,47085	1,17521
6000	162.5	975000	0,59778	1,77299
5000	187.5	937500	0,71982	2,49281
4500	212.5	956250	0,84430	3,33711
5036	237.5	1196050	1,00000	4,33711

i. Comparison between (Fi) and (Qi)

La courbe de Lorenz représente les variations de (Qi) et de (Fi). L'indice de Gini est représenté par la surface entre la première bissectrice et la courbe.



Below is the Lorenz curve corresponding to the given data, along with the computed Gini index: 0.1572. A Gini index near 0 signifies a more uniform pixel distribution, whereas a higher value indicates greater disparity in their distribution. The calculated value ensures a uniform distribution of pixels within the image encrypted by our approach.

5) Brutal assaults

Brute force attack is a cryptanalysis method that systematically tests all possible keys until the correct one is found. It relies on brute force, i.e. the computing power and time available to try each key. Brute-force attacks are effective against systems with short keys. However, they become impractical with sufficiently long keys. This is why modern cryptography uses keys of 128 bits or more, making brute-force attacks theoretically impossible with current technologies.

a. Brute-force attack principle

Encryption systems protect data with a secret key used for both encoding and decoding. A brute-force attack aims to discover this key by generating and applying every possible key to the ciphertext until the original message becomes understandable. The table below estimates the time required to reconstruct a key of given size in (s):

Complexity example:

Private key length (bit)	Number of options	Estimated time
8	256	Instant
16	65536	Less than a second
32	$43 * 10^7$	A few seconds
64	$43 * 10^{29}$	Several years
128	$43 * 10^{33}$	Unfeasible over billions of years

6) Key-space analysis

The brute force attack is a technique used in cryptanalysis to find the encryption key. It consists in reconstructing the encryption key using all possible combinations. As a result, a large encryption key makes brute force attacks impossible. In our algorithm, the size of the secret key is much larger than (2^{128}) [18 – 19 – 20], which guarantees a better protection against brute force attacks. The secret key of our simulation is generated from the three most used chaotic maps in cryptography. As a result, the size of our key consists of six parameters written in 32 bits each, and thus, the global size is $(2^{6 \times 32}) = (2^{192}) \gg (2^{100})$.

7) Secret key's sensitivity Analysis

Understanding how even small variations in the encryption key influence image encryption and decryption is vital for key sensitivity studies. This analysis is important for assessing an encryption algorithm's resilience and for comprehending the potential security breaches resulting from key errors or malicious attacks.

a. Changing a single subkey

This high sensitivity of the key of our new technology, can be seen in the following figure

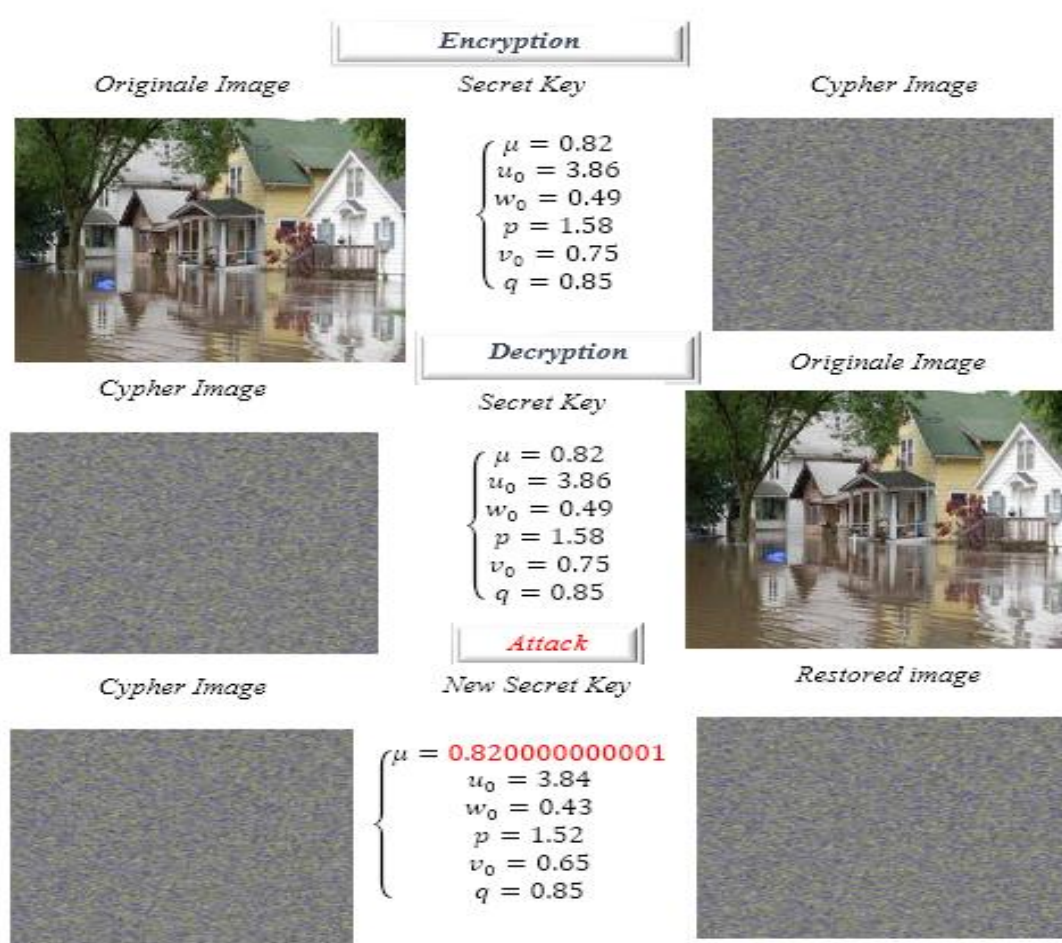


Figure4: Encryption key sensitivity

We note that a perturbation of the order of (10^{-7}) on a single parameter, is not able to reconstruct the original image. In summary, studying the sensitivity of the image encryption key helps to understand how the algorithm reacts to small changes in the private encryption key, which is essential to ensure the security and robustness of the encryption system.

b. Hamming distance

It is a measure of the number of bits that differ between two encrypted versions of the same plaintext data with slightly different keys.

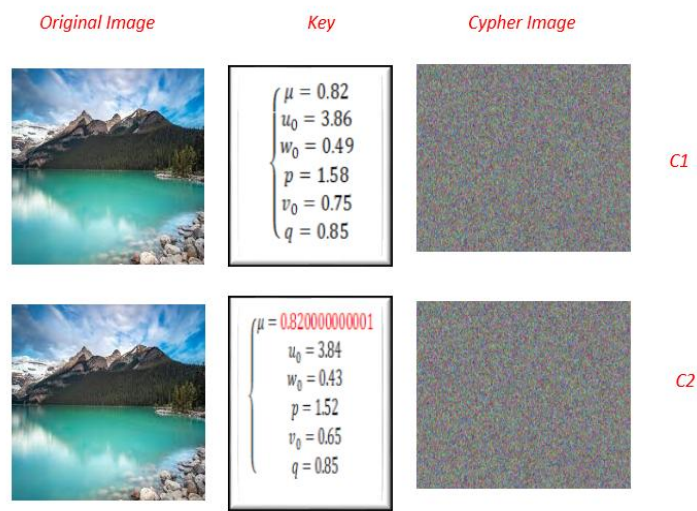


Figure5: Hamming distance

After conversion of the two images into binary vector (IR) and (IT), the Hamming distance is the number of different bits between ($C1$) and ($C2$). The Hamming distance between the two images ($C1$) and ($C2$) is given by the following formula

$$HM(C1; C2) = \sum_{i=0}^{3nm-1} |IR(i) - IT(i)| \quad (12)$$

We have raised more than 90% of different bits, which ensures that our encryption key is very sensitive to the slightest disturbance

8) Statistics Attack Security

Histogram attack is a statistical analysis technique used to evaluate the security of image encryption. It is based on the study of the distribution of pixel intensities before and after encryption.

9) Histogram analysis

Statistical attacks exploit the statistical properties of encrypted data to attempt to recover information about the original message or encryption key. They are particularly used against poorly designed encryption systems or those that do not produce a random distribution of encrypted data. The histogram of an image represents the distribution of gray levels (or colors for an RGB image). A good encryption algorithm must standardize this histogram to avoid any exploitable structure

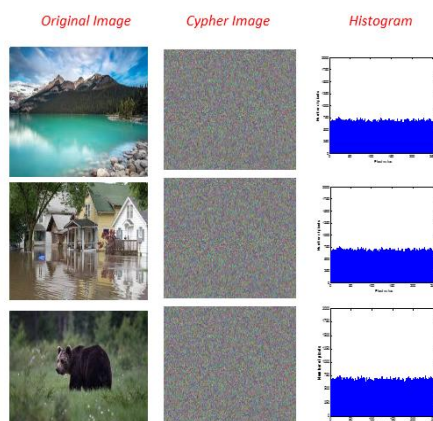


Figure6: Histogram analysis

a. Histogram comparison

To assess how evenly gray levels are distributed in an encrypted image, we will compare its histogram to a theoretically ideal flat histogram by calculating the difference between them. A perfectly flat histogram is

characterized by each gray level having the same frequency, a value dependent on the image dimensions. This uniform value is:

$$Hg(i) = \frac{3nm}{256}$$

The difference (Dh) is given by the algorithm below

Algorithm 21

1. $Dh = 0$
 2. For $i = 0$ to 255
 3. $Dh = Dh + |Hg(i) - Np(i)|$
 4. Next i
 5. $Dh = \frac{1}{3nm}(Dh)$
-

In all our simulations, the (Dh) value remains close to 0, indicating that the histograms of the encrypted images closely resemble perfectly flattened and uniformly distributed histograms. The table below shows that the histograms of the encrypted images are uniformly distributed with an error of 0.01%.

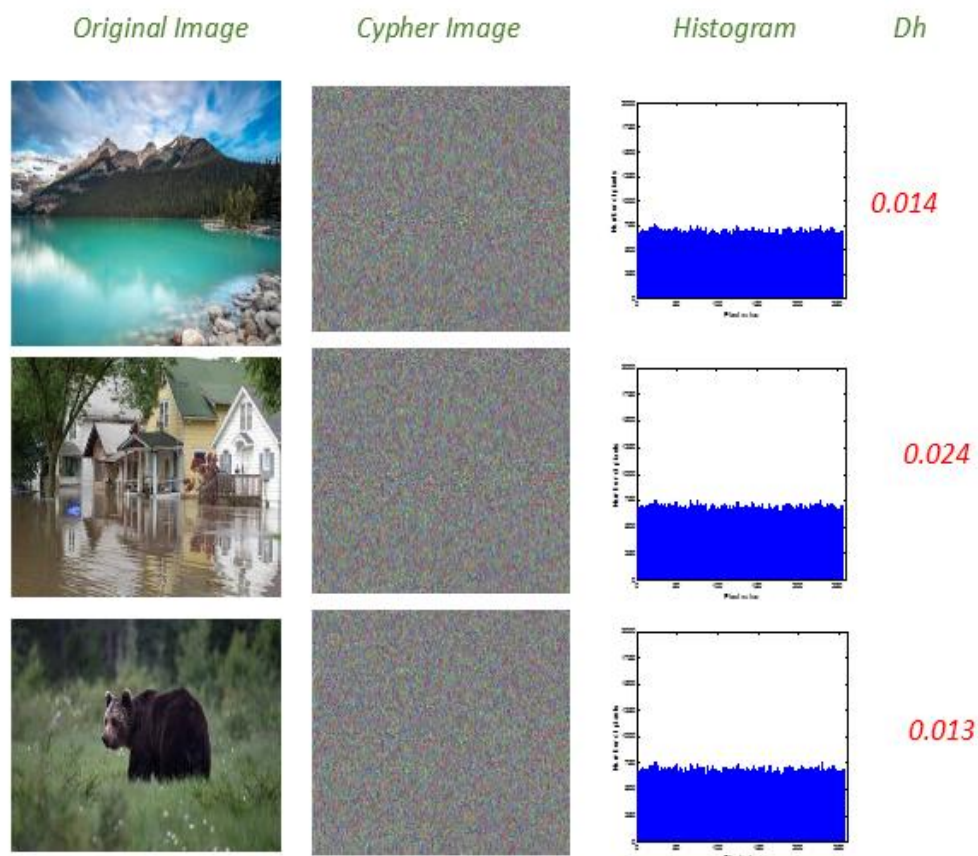


Figure7: Histogram Comparaison

b. Entropy analisys

Entropy is a fundamental concept in cryptography that measures the randomness or unpredictability of a system. It is crucial for assessing the security of cryptographic algorithms, especially in encryption, key generation and secure communications.

Entropy, in the context of cryptography, is derived from information theory (developed by Claude Shannon). It quantifies the uncertainty or randomness in a system. The entropy $H(X)$ of a random variable (X) with possible values (x_i) and corresponding probabilities $P(x_i)$ is given by:

Equation13

$$P(X) = - \sum_{i=0}^{255} P(x_i) * \log(P(x_i))$$

For encrypted images, the entropy is expected to be close to 8 bits per pixel (for 8-bit grayscale images). This means every pixel value should be uniformly distributed across all possible intensity levels (0–255), making it indistinguishable from pure noise.

- ✓ If entropy is significantly lower than 8, the encryption is weak, and patterns may exist in the ciphertext.
- ✓ If entropy is close to 8, it indicates that the encryption is strong, and the encrypted image appears random.

In all our simulations, the entropy of the encrypted image is very close to the maximum value (8), which ensures that our algorithm is immune to any entropy attack.

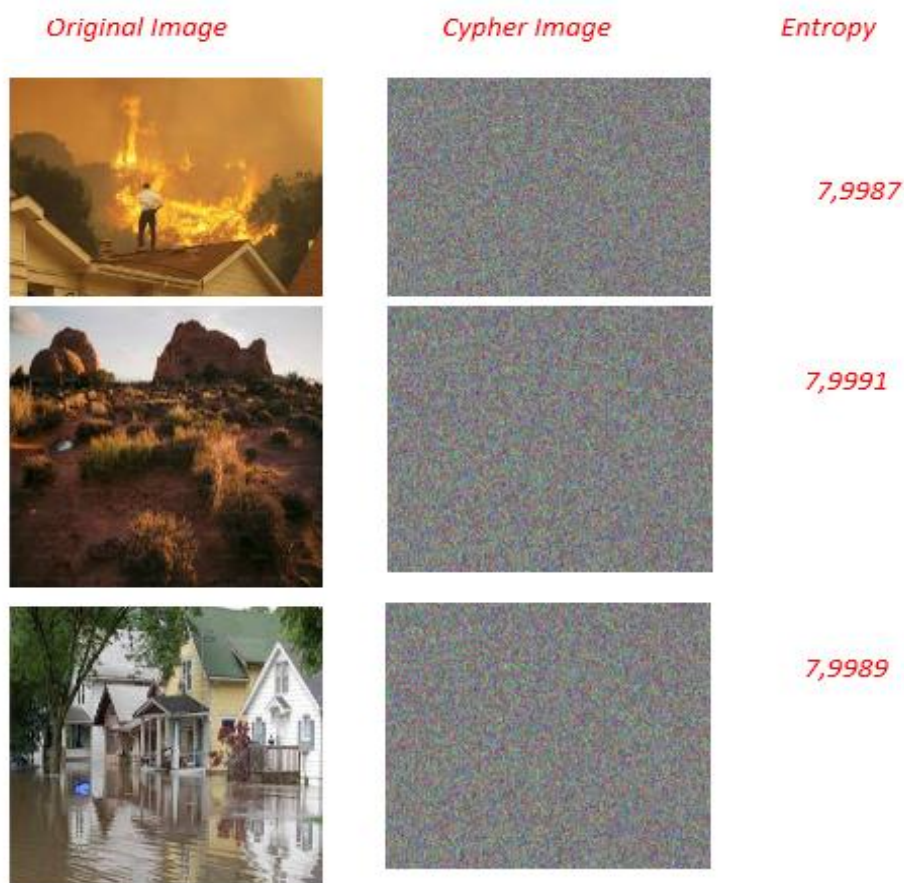


Figure8: Entropy analysis

c. Correlations analysis

Correlation in cryptography is a key metric for assessing the security of encryption systems. It is primarily used to analyze the relationship between plaintext and encrypted data in order to identify potential weaknesses.

Correlation is a statistical measure that assesses the relationship between two data sets. In cryptography, it is used to measure the relationship between:

- ✓ The image or plaintext and the ciphertext
- ✓ Neighboring pixels in an encrypted image
- ✓ The bits of a cryptographic key and their impact on the encryption

Correlation is often calculated using the Pearson correlation coefficient, defined by the formula:

Equation 14

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

$$\text{Cov}(X, Y) = \frac{1}{3nm} \sum_{i=1}^{3nm} (X_i - \bar{X})(Y_i - \bar{Y})$$

✓ \bar{X} : The vector means (X)

✓ \bar{Y} : The vector means (Y)

Interpretation:

$r \approx 0$ then Absence of any linear relationship between adjacent pixels

$r \approx 1$ then existence of a linear relationship between neighboring pixels

The table below presents the correlation values between adjacent pixels in the encrypted image. These values are close to zero, confirming the absence of linear relationships between neighboring pixels.







Original Image	Cypher Image	Correlation
		0.0014
		0.00201
		0.00211

Figure9: Correlation analisys

All images tested with our new technique exhibit a correlation near 0, guaranteeing the absence of any affine relationship between adjacent pixels.

d. Autocorrelation analysis:

Autocorrelation in images quantifies the statistical relationship between pixels based on their separation. In unencrypted, structured images, adjacent pixels are highly correlated, meaning their intensity values are similar. However, secure image encryption aims to eliminate this correlation, resulting in randomly distributed pixel values and near-zero autocorrelation. Analyzing the correlation of an image demonstrates the absence of a linear (affine) dependency between neighboring pixels. By measuring a signal's similarity to its shifted versions, autocorrelation is a key method in image cryptography for determining the security and randomness of the encrypted output.

For a given $(k \in \mathbb{N})$ we analyze the similarity between a signal, the encrypted image, and its shifted version by $(k \in \mathbb{N})$ positions. This involves measuring the rank $(k \in \mathbb{N})$ autocorrelation of the encrypted image to evaluate its security and randomness.

Let

$$\begin{cases} \tau_k: G_{3nm} \rightarrow G_{3nm} \\ i \mapsto \text{mod}(i - k; 3nm) \end{cases}$$

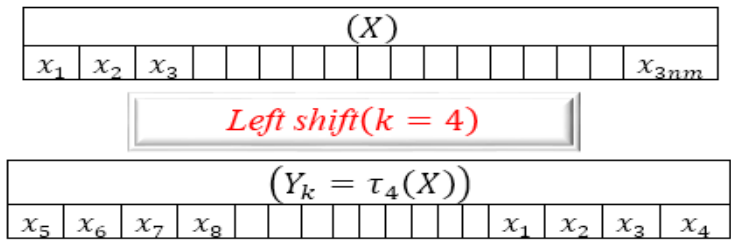
The application (τ_k) is a bijection.

An autocorrelation is a correlation of order between the vector (X) and its image (X_k) by the function.

Note:

- ✓ For $k = 1$, this represents the natural correlation.
- ✓ For $k=2$, it refers to the correlation between (X) and $(\tau_2(X))$.

Example ($k = 4$)



Note:

$$\begin{aligned} \bar{X} &= \bar{Y}_k \\ V(\bar{X}) &= V(\bar{Y}_k) \\ Cov(X, Y_k) &= \frac{1}{3nm} \sum_{i=1}^{3nm} (X_i - \bar{X})(Y_{k,i} - \bar{Y}_k) \\ \bar{X} &: \text{the mean} \quad \text{and} \quad V(\bar{X}): \text{the variance} \end{aligned}$$

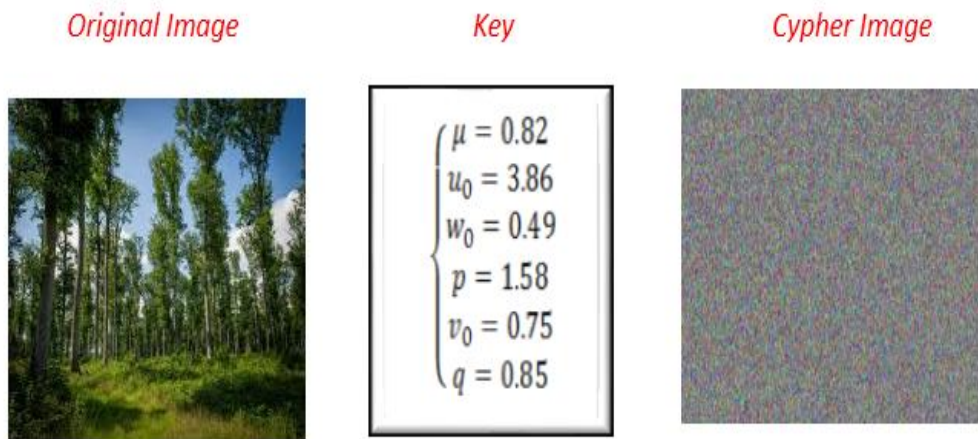
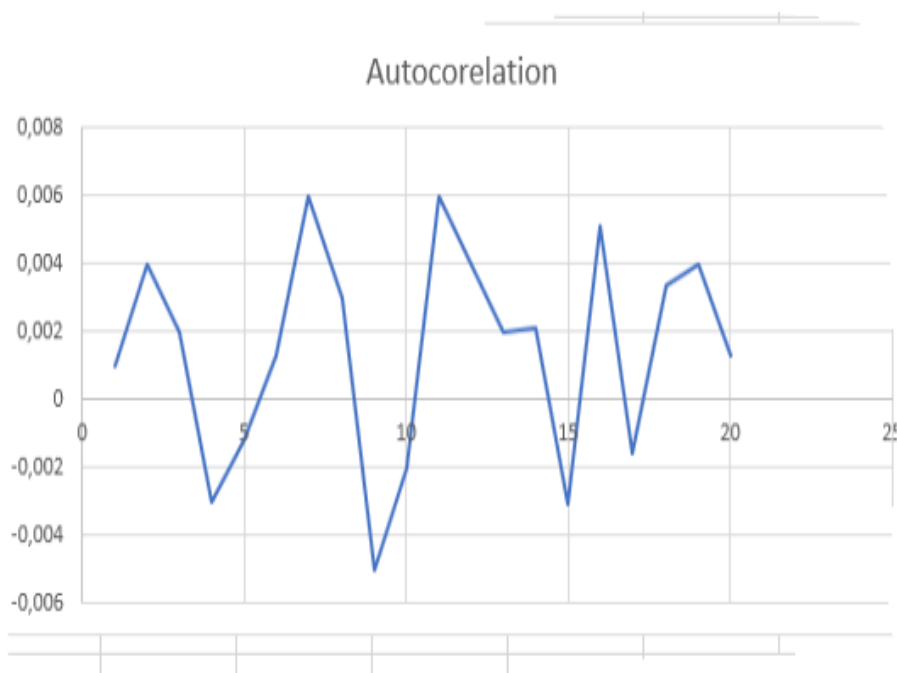


Figure10: Autocorrelation computed

We will calculate the autocorrelation of an image encrypted by our system for a value of k varying between 1 and 20. Here is the representation of the values raised.



All the values obtained are close to zero. This confirms that our encrypted image is almost random.

10) Differential Attack

Differential attack is a cryptanalytic technique that leverages the correlation between variations in plaintext and the corresponding alterations in ciphertext to infer the secret key. Within the domain of image cryptosystems, this form of attack primarily targets block cipher algorithms (such as AES and DES) or chaotic encryption schemes that are commonly utilized for securing image data.

a. Differential Attack Principle

The idea is to introduce small variations (such as a single pixel change or a single bit modification) into the original image, and then analyze the difference between the corresponding encrypted versions. A good cryptosystem should maximize the avalanche effect, that is, a small change in the original image should lead to a radical transformation in the image encrypted with the same private key.

i. Evaluation Criteria

Two key metrics are used to evaluate the resistance of an image cryptosystem to differential attacks:

▪ NPCR - Number of Pixels Change Rate

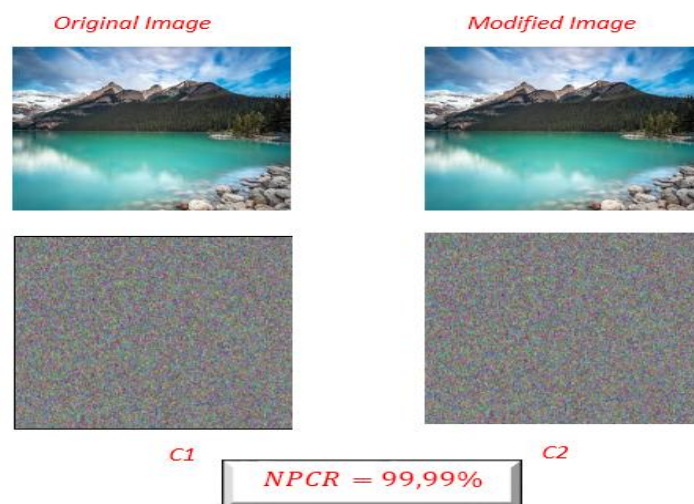
NPCR is a key metric in image cryptanalysis that evaluates a cipher's sensitivity to minor alterations in the original image. It quantifies the percentage of pixels that differ between two encrypted images when a single pixel in the original image is slightly modified. For a cryptosystem to be considered secure, its NPCR should be high (close to 100%), ensuring that even a minimal change in the input image leads to a widespread and unpredictable transformation in the encrypted output.

This constant (NPCR) is given by the following formula

$$NPCR = \frac{1}{3nm} \sum_{i=1}^{3nm} D(i; j) \quad D(i; j) = \begin{cases} 1 & \text{if } C_1(i; j) = C_2(i; j) \\ 0 & \text{if } C_1(i; j) \neq C_2(i; j) \end{cases} \quad (15)$$

- ✓ C1 encrypted original image
- ✓ C2 encrypted modified original image

Example:



▪ **UACI (Unified Average Changing Intensity)**

The Unified Average Changing Intensity (*UACI*) is a cryptanalysis metric that quantifies the impact of a minor alteration in the original image on pixel intensity variations in the encrypted image. It serves as a complement to the Number of Pixels Change Rate (NPCR) by evaluating both the number of modified pixels and the magnitude of their intensity changes.

$$NPCR = \frac{1}{3nm} \sum_{i=1}^{3nm} |C_1(i; j) - C_2(i; j)| \quad (16)$$

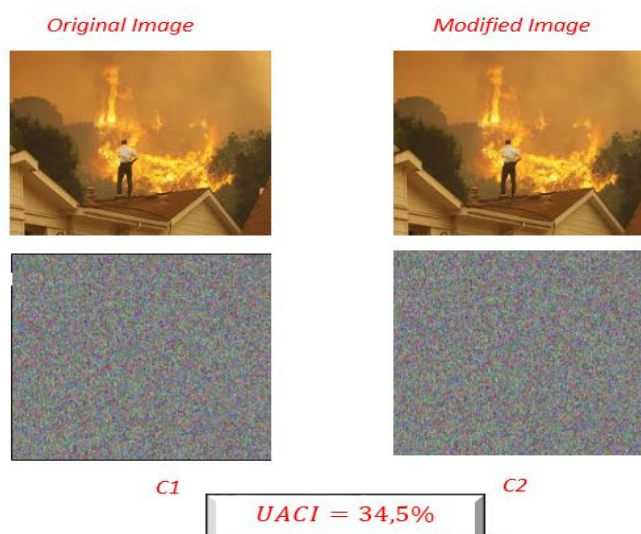
The *UACI* (Unified Average Changing Intensity) is a crucial metric for evaluating the strength of image encryption algorithms. Alongside the *NPCR*, it quantifies the avalanche effect guaranteeing that even a minor modification in the original image leads to a substantial and random alteration in the encrypted version. Together, these metrics ensure the cryptographic system's sensitivity to input changes, a vital property for secure encryption.

Interpretation:

A high *UACI* (ideally around 33.34% for an 8-bit grayscale image) indicates high sensitivity to small changes, which is crucial for a good cryptosystem.

A low *UACI* value means that the encryption algorithm is not sufficiently diffusing changes, which can make the encrypted image vulnerable to differential attacks.

Example:

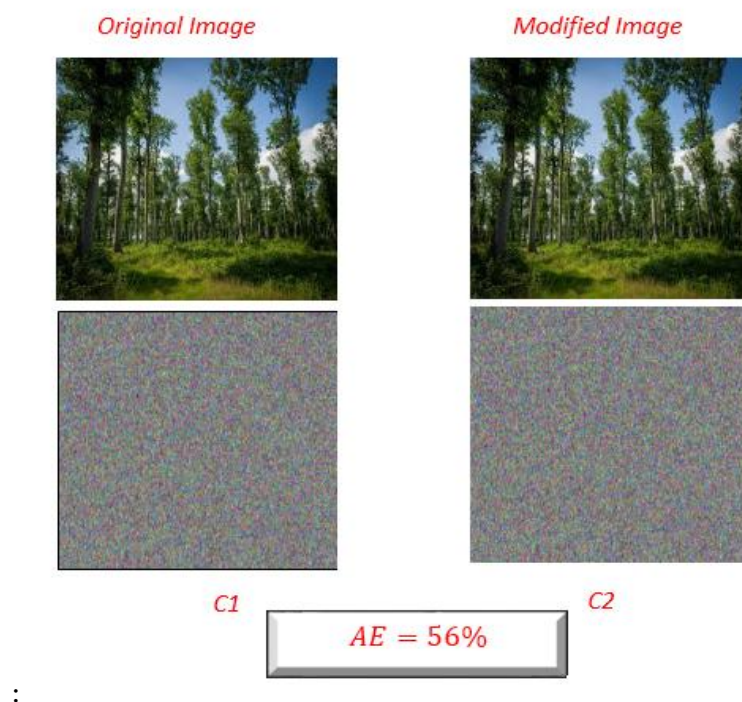


The computed NPCR and UACI values meet international standards, confirming that our system is resistant to differential attacks.

▪ *The Avalanche Effect (AE)*

is a critical concept in cryptography, requiring that even a minimal modification in the input—such as flipping a single bit in the plaintext—should produce a drastic and seemingly random alteration in the output (ciphertext). This property is vital for secure encryption, as it ensures that attackers cannot deduce meaningful patterns or relationships between plaintext and ciphertext, even when analyzing minor input variations. A strong avalanche effect enhances cryptographic robustness by making the ciphertext appear statistically independent of the plaintext, thereby thwarting differential cryptanalysis and other inference-based attacks.

Example:



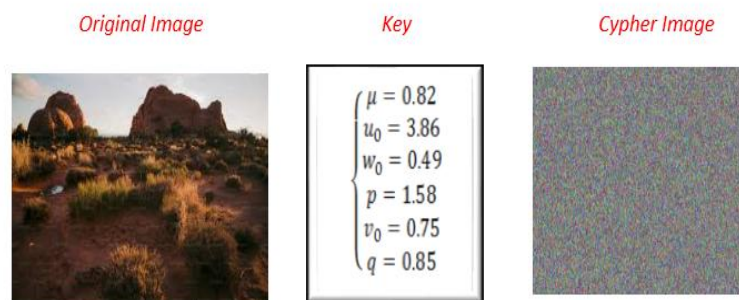
▪ *The Chi-square (χ^2) test is*

The chi-square (chi-square) test is an essential statistical tool for assessing the pixel distribution in an encrypted image. A good encryption system must produce uniform histograms, ensuring a random distribution of pixel values and rendering any statistical attack ineffective. If the chi-square value is close to the theoretical threshold, this indicates that the pixel distribution of the encrypted image is close to a uniform distribution, thus confirming good resistance to statistical analyses. Conversely, a significantly different value could reveal an exploitable weakness in the encryption algorithm. Thus, a strong encryption system must present chi-square test results that comply with international standards, ensuring effective protection against attacks based on frequency analysis and pixel structure.

The Chi-square (χ^2) test is a statistical method used to evaluate whether the grayscale distribution of an encrypted image adheres to a uniform distribution. This uniformity is critical for ensuring resistance against statistical attacks, as deviations could reveal exploitable patterns in the ciphertext. We apply the test formula

$$\chi^2 = \sum_{i=1}^{256} \frac{(O_i - E_i)^2}{E_i} \quad (17)$$

(O_i) Observed gray level frequency (i) (E_i) Expected theoretical frequency (if uniform distribution).



Division of gray levels into 10 classes. We obtain the table below:

Observed and theoretical frequency

- ✓ Assume an image size of 256×256 pixels, or 65,536 pixels.
- ✓ If the distribution is uniform, each class should contain on average:

Expected theoretical frequency

$$E_i = \frac{65536}{10} = 6553.6$$

	Frequencies		Report
Classes	Expected	Theoretical	$\frac{(O_i - E_i)^2}{E_i}$
I_1	6700	6553.6	2.4
I_2	6450	6553.6	1.6
I_3	6600	6553.6	1.51
I_4	6048	6553.6	1.89
I_5	6555	6553.6	0.3
I_6	6610	6553.6	1.4
I_7	6400	6553.6	3.79
I_8	6580	6553.6	0.42
I_9	6520	6553.6	0.16
I_{10}	6700	6553.	1.23

Chi-square calculation

$$X^2 = 14.23$$

Interpreting the results

Degrees of liberty = $10 - 1 = 9$

Critical threshold for $\alpha = 0.05$ (Chi-square table): 16.92

Comparison

$$X^2 = 14.23 < 16.92$$

the hypothesis of a uniform distribution is accepted

MATH SECURITY

All chaotic maps employed in this work demonstrate extreme sensitivity to initial conditions. Additionally, the substantial size of our secret key ensures strong resistance against brute-force attacks. The randomness of the P-Boxes operating at the bit level, combined with the implemented chaining mechanism linking encrypted and plaintext lines, significantly enhances the complexity of timing attacks and safeguards the system from differential attacks. Lastly, the application of genetic crossover at the bit level further reinforces security, giving the encrypted image a highly randomized appearance.

CONCLUSION

This algorithm starts with a deeply improved implementation of Vigenere technology. This method is provided by two chaotic replacement matrices constructed from random displacement controlled by a binary decision vector. These two matrices generate a replacement function controlled by the binary vector and a diffusion function is implemented for protection against differential attacks. At the end of this first round, a quick passage in binary notation is established on the integrity of the encrypted vector, followed by a passage in pure matrix to execute chaotic permutations on the columns followed by permutations on the rows. Finally, the proposed new technique was analyzed according to key space, histogram analysis information entropy analysis, correlation analysis and differential attack analysis. All the findings in the simulations tuned to multiple images are within the universal standards, which provides great protection against the known attacks, and that the proposed algorithm has desirable encryption effect, high encryption efficiency, large key space and high key sensitivity, and can effectively resist the brute force attack, statistical attack and differential attack.

CONFLICT OF INTEREST

All authors of this article, and there are no private or public organizations or laboratories to fund my research, thus avoiding any expected conflicts.

This document does not contain any research or experiments conducted on animals.

ETHICS APPROVAL

No applicable

CONSENT TO PARTICIPATE

No applicable

CONSENT TO PUBLISH

No applicable

DATA AVAILABILITY STATEMENT

The authors used publicly available image databases on the internet to perform the simulations and provide results.

AUTHORS' CONTRIBUTIONS

Mariem Jarjar: Theoretical Foundation

Hassan Tabti, Hicham Rrghout: Manuscript Organization and Simulations

El Bourkadi Hamid: Algorithm Verification

Abdellatif Jarjar and Abdelhamid Benazzi: Supervision and Monitoring of Manuscript Completion.

FUNDING

No private or public organization supports our research through direct or indirect funding.

REFERENCES

- [1] Lone, Parveiz Nazir, et al. "Cryptanalysis and improved image encryption scheme using elliptic curve and affine Hill cipher." *Mathematics* 10.20 (2022): 3878.
- [2] Bansal, Ritesh, Shailender Gupta, and Gaurav Sharma. "An innovative image encryption scheme based on chaotic map and Vigenère scheme." *Multimedia Tools and Applications* 76 (2017): 16529-16562.
- [3] El Kaddouhi, S., et al. "A new image encryption approach that uses an improved Hill-Vigenère method and chaotic maps." *MultiMedia Tools and Applications* (2024): 1-29.
- [4] Jarjar, Mohamed, et al. "New technology of color image encryption based on chaos and two improved Vigenère steps." *Multimedia Tools and Applications* 81.17 (2022): 24665-24689.

- [5] Chemlal, Abdelhakim, et al. "Enhanced Vigenere encryption technique for color images acting at the pixel level." *International Journal of Electrical & Computer Engineering* (2088-8708) 14.6 (2024).
- [6] Paul, Amit, Rajwinder Singh, and Deep Singh. "A Multilayer Encryption Scheme Using Vigenère Cipher and Chaotic Maps." *SN Computer Science* 6.3 (2025): 262.
- [7] A.S alkhaliid « cryptanalyze of Hill cipher using genetic algorithm » dalam IEEE hanmument 2015.
- [8] A. Jarjar« Improvement of hill's classical method in image cryptography » *International Journal of Statistics and Applied Mathematics* 2017, Volume 2 Issue 3, Part A.
- [9] Rani, Narbda, Vinod Mishra, and Suvita Rani Sharma. "Image encryption model based on novel magic square with differential encoding and chaotic map." *Nonlinear Dynamics* 111.3 (2023): 2869-2893.
- [10] Kattass, Mourad, et al. "Chaotic image encryption using an improved Vigenere cipher and a crossover operator." *International Conference on Computing, Intelligence and Data Analytics*. Cham: Springer Nature Switzerland, 2023.
- [11] Abughali, Heba AM, and Utku Kose. "Image Encryption Techniques: A Survey." 2024 *International Jordanian Cybersecurity Conference (IJCC)*. IEEE, 2024.
- [12] Abdul-Kareem, Ali Akram, and Waleed Ameen Mahmoud Al-Jawher. "Image encryption algorithm based on Arnold transform and chaos theory in the multi-wavelet domain." *International Journal of Computers and Applications* 45.4 (2023): 306-322.
- [13] Zheng, Yuxiao, et al. "Image encryption based on novel Hill Cipher variant and 2D-IGSCM hyper-chaotic map." *Nonlinear Dynamics* 113.3 (2025): 2811-2829.
- [14] Jarjar, Abdellatif, and Abdelhamid Benazzi. "Image Encryption Using Hill Cipher Under a Chaotic Vector's Control." *Digital Technologies and Applications: Proceedings of ICDTA'24, Benguerir, Morocco, Volume 1* 1 (2024): 298.
- [15] Mfungo, Dani Elias, et al. "A novel image encryption scheme using chaotic maps and fuzzy numbers for secure transmission of information." *Applied Sciences* 13.12 (2023): 7113.
- [16] Vamsi, D. E. S. A. M., and R. Ch. "Color Image Encryption Based on Arnold Cat Map-Elliptic Curve Key and A Hill Cipher." *J Theor Appl Inf Technol* 15.9 (2024).
- [17] Xi, Yuzhou, et al. "A Dynamic Hill Cipher with Arnold Scrambling Technique for Medical Images Encryption." *Mathematics* 12.24 (2024): 3948.
- [18] Jan Sher Khan all, "Chaos based efficient selective image encryption" *Multidimensional Systems and Signal Processing* volume 30, pages943–961(2019) “.
- [19] Ahmed, Saja Theab, et al. "Medical image encryption: a comprehensive review." *Computers* 12.8 (2023): 160.
- [20] Mahboob, Abid, Muhammad Nadeem, and Muhammad Waheed Rasheed. "A study of text-theoretical approach to S-box construction with image encryption applications." *Scientific Reports* 13.1 (2023): 21081.
- [21] Youssef, Mohammed, et al. "Enhancing satellite image security through multiple image encryption via hyperchaos, SVD, RC5, and dynamic S-Box generation." *IEEE Access* (2024).
- [22] Shafique, Arslan, et al. "Lightweight image encryption scheme for IoT environment and machine learning-driven robust S-box selection." *Telecommunication Systems* 88.1 (2025): 1-23.
- [23] Singh, Laiphrakpam Dolendro, et al. "Image encryption using dynamic s-boxes generated using elliptic curve points and chaotic system." *Journal of Information Security and Applications* 83 (2024): 103793.
- [24] Ali, Rashad, et al. "A novel S-box generator using Frobenius automorphism and its applications in image encryption." *Nonlinear Dynamics* 112.21 (2024): 19463-19486.
- [25] Etem, Taha, and Turgay Kaya. "Modified Bernoulli map-based scr
- [26] Akraam, Muhammad, Tabasam Rashid, and Sohail Zafar. "A Chaos-Based Image Encryption Scheme Is Proposed Using Multiple Chaotic Maps." *Mathematical Problems in Engineering* 2023.1 (2023): 2003724.
- [27] Sani, R. Hoseini, Sohrab Behnia, and J. Ziaei. "Construction of S-box based on chaotic piecewise map: Watermark application." *Multimedia Tools and Applications* 82.1 (2023): 1131-1148.
- [28] Zhang, Bowen, and Lingfeng Liu. "Chaos-based image encryption: Review, application, and challenges." *Mathematics* 11.11 (2023): 2585.
- [29] Qobbi, Younes, et al. "Image encryption algorithm based on genetic operations and chaotic DNA encoding." *Soft Computing* 26.12 (2022): 5823-5832.