Computers and Electrical Engineering 120 (2024) 109722

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

# Medical image encryption algorithm based on Fresnel zone formula, differential neural networks, and pixel-guided perturbation techniques

Muhammed Jassem Al-Muhammed

*Faculty of Information Technology, American University of Madaba, Madaba, Jordan*

ARTICLE INFO

*Keywords:*
Image encryption
Dynamic substitution
Differential neural network
Pixel-based perturbation
Key-based codes

ABSTRACT

This paper proposes an image encryption technique that integrates deep pixel substitution, data-dependent and chaotic pixel perturbation, and differential neural networks. The pixels of the image are manipulated using a deep pixel substitution operation that is based on the Fresnel Zone equation to eliminate the correlations to the input plain image. Additionally, a perturbation process based on pixel values and chaotic noise is applied to further scramble the image. The resulting image is then subjected to a second round of deep substitution. The differential neural network generates blurring codes by incorporating plain pixel blocks and an encryption key, which are subsequently added to the processed image to produce the final ciphered image. The proposed technique's effectiveness was evaluated on a large dataset that included both medical and non-medical images. Simulation results indicated that the proposed technique was not only efficient but also effective for both medical and non-medical images, and it outperformed state-of-the-art encryption methods in both security properties and computational efficiency.

## 1. Introduction

Since the inception of encryption, researchers have proposed different encryption techniques based on distinct computational models [1–6]. The computational models include neural networks [7–9], chaos theory [10–14], mathematical algorithms [15–19], genetic algorithms [20,21], and DNA encoding [22–25]. These models are utilized to enhance the confusion of ciphered images through static symbol manipulation and boost diffusion through chaotic or zigzag operations. A common characteristic of these techniques is that the impact of bit changes is localized; when processing image blocks, the influence of bit changes is restricted to the specific block where the change occurred. However, encryption methods that detect bit changes and propagate these changes across the entire ciphered image are intuitively more secure.

This paper proposes an encryption technique based on dynamic pixel substitution operation. The computation of the substitution operation is defined based on a Fresnel zone system that employs two control parameters. These parameters are linked to both the secret key and the input image pixels, allowing the operation to dynamically adjust pixel processing in response to changes in the input image and the secret key. Unlike other image encryption techniques that rely on static pixel processing models [26,27], the proposed substitution significantly changes the processing of all subsequent pixels, not just the pixel that was changed. This capability increases confusion, thereby improving the technique's resistance to cryptanalysis tools.

In addition to dynamic substitution, the proposed technique uses a differential neural network that utilizes the secret key to generate highly complex codes and obscure the image pixels. By processing the secret key and leveraging the differences between

*E-mail address:* m.almuhammed@aum.edu.jo.

https://doi.org/10.1016/j.compeleceng.2024.109722
Received 26 July 2024; Received in revised form 6 September 2024; Accepted 19 September 2024
Available online 2 October 2024

**Algorithm 1** Initialization of logistic map parameters $x$ and $r$

1: $h \leftarrow$ SHA–512(Key)
2: $r^* \leftarrow \sum_{i=1}^{32} h_i \times \log_e(256^{32-i})$
3: $x^* \leftarrow \sum_{i=33}^{64} h_i \times \log_e(256^{64-i})$
4: $x \leftarrow \text{FRAC}(x^*)$
5: $r \leftarrow 3.57 + \text{FRAC}(r^*) \times 0.43$

the neural network's output and the image pixels, the differential neural network biases the computations and maintains a feedback loop that influences the neural network's input. This interplay between the computation bias and the feedback ensures that the neural network's output depends on both the encryption key and the image data, making the technique resilient against chosen-plaintext attacks—the most effective form of attack. Unlike other key processing methods, such as those in [28,29], which have been cracked in [30,31], and [32], which require the key generation process to remain confidential and shared only between sender and receiver, the proposed technique protects the key without the need for sharing anything beyond the key itself.

The proposed technique has an additional feature that makes it distinct and more powerful than recent techniques. Current encryption techniques often rely on simple pixel scrambling designs that do not provide robust pixel diffusion. Specifically, recent methods employ either Zigzag pixel perturbation operations [33–36] or chaotic means [37–40]. While these techniques offer reasonable diffusion, the proposed technique achieves much more effective diffusion. It incorporates a pixel scrambling operation that is both chaotic and pixel-dependent. This reliance on chaos and pixel values enables the technique to produce entirely different pixel perturbations even with a single pixel change, resulting in a significantly higher level of security.

The proposed technique distinguishes itself from current methods in several key ways, enhancing the security of the ciphered images compared to existing approaches.

1. Unlike current techniques that rely on static substitution operations, the proposed technique employs dynamic substitution operations. This method maintains a global memory that accumulates and amplifies bit changes, propagating them to influence all subsequent pixels in the image.
2. The proposed technique uses a pixel diffusion method that incorporates chaotic signals and is dependent on pixel values to perturb the image pixels. This approach significantly enhances encryption security by ensuring that changes in the plaintext image pixels have a substantial impact on the ciphered images.
3. The proposed technique incorporates a differential neural network to ensure conservative use of the key. This method protects the key's identity and generates highly complex image-hiding codes that depend on both the image pixels and the key itself, making it robust against cryptanalysis attempts.

The paper is organized as follows. Section 2 introduces the chaotic system and presents a specific procedure to initialize its parameters. Section 3 presents the details of the proposed substitution technique based on Fresnel zone. Section 4 introduces the differential neural network and its use in generating blurring codes. Section 5 presents the details of the pixel scrambling technique. Section 6 discusses the proposed encryption technique. Section 7 presents the security evaluation for the proposed technique. We conclude the paper by giving final remarks in Section 8.

## 2. Logistic map

Chaotic maps, which operate in discrete time, align well with the discrete nature of digital images, allowing for seamless integration into encryption algorithms. Their efficiency in generating complex, non-repetitive sequences makes them particularly effective for digital image encryption. These generated sequences can thoroughly scramble image pixels, improving resistance to common cryptanalytic attacks.

In this paper, we used the logistic map due to its mathematical simplicity, efficiency, and strong chaotic properties. The logistic map is defined by Eq. (1), with the parameters $r \in (0, 4]$ and $x \in (0, 1)$. This map enters a state of chaos only when the value of $r$ falls within the range of $(3.57, 4]$. Additionally, its extreme sensitivity to the initial values of $r$ and $x$ allows for the generation of a diverse array of chaotic sequences by varying these parameters within their effective ranges.

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

To securely couple the generated chaotic sequences with the encryption key, we follow the initialization process outlined in Algorithm 1. First, we hash the encryption key using the SHA-512 algorithm to obtain a 512-bit digest (64 hexadecimal values). These values, denoted as $h_1, h_2, \ldots, h_{64}$, are then used to compute the initial values $r^*$ and $x^*$. These initial values are subsequently used to generate the map parameters $r$ and $x$ (steps 4 and 5) within their respective ranges. Since $x \in (0, 1)$, we assign $x$ the fractional part of the generated value $x^*$. For $r$, which must fall within the interval (3.57, 4], we add the product of the fractional part of $r^*$ and 0.43 to 3.57. The function FRAC($z$) is used to extract the fractional part of $z$.

## 3. Fresnel zone-based substitution operation

This section introduces a powerful substitution technique that leverages nonlinear computations and logical operators for deep symbol manipulation. This operation significantly increases the output's entropy, irrespective of the input plaintext's initial entropy. We begin by defining the Fresnel zone concept, followed by a detailed discussion of the substitution operation based on this concept.

**Algorithm 2** Update-df($R$, $c$)

```
1: f ← Long(R) {Extract long integer from the double value R}
2: d ← c << (Mod(d, z)) XOR d {c is a value of a pixel}
3: d ← d XOR d << 21
4: d ← d XOR d >> 35
5: d ← d XOR d << 4
6: return f, d
```

### 3.1. Fresnel zone

The Fresnel zone, named after the physicist Augustin-Jean Fresnel, is an area surrounding the line-of-sight through which radio waves emanating from an antenna spread out. The radius, $R$, of the widest section of the Fresnel zone can be computed using the mathematical model (2).

$$R = 17.32 \times \sqrt{\frac{d}{4f}} \tag{2}$$

where $d$ is the distance between the transmitter and receiver and $f$ is the frequency of the wave signal.

The radius $R$ is determined by a nonlinear relationship between the distance $d$ and the frequency $f$. While this equation exhibits monotonic behavior, it does not follow a consistent pattern of increase or decrease; rather, the values of $R$ fluctuate with changes in $d$ and $f$. This inherent nonlinearity and unpredictability make the Fresnel zone particularly suitable for encryption. By properly linking $d$ and $f$ to the pixels of the input image, we can use model (2) to generate highly complex and pixel-based values for $R$, which can then be used in the encryption process.

This paper proposes a specific method for binding $d$ and $f$ to image pixels, as outlined in Algorithm 2. In step 1, the frequency $f$ is set to the value of the radius $R$ after converting $R$ to a long integer. In step 2, the distance $d$ is updated through a memory-like process, where the cumulative effects of image pixels influence $d$. The location at which a pixel $c$ impacts $d$ is dynamic and can vary across different bits within the long integer $d$. This dynamic impact is determined by left-shifting the pixel value $c$ by Mod$(c, z)$, where $z$ is derived from the rightmost 6 bits of $c$ (i.e., $z$ = Mod$(c, 32)$), as described in step 2. To further amplify the influence of the pixels, the bits of $d$ are deeply manipulated using steps 3–5 after incorporating the impact of pixel $c$.

### 3.2. The substitution operation

The substitution operation is dual-directional processing, where the pixels of the block are substituted using the Fresnel zone model (2). Algorithm 3 illustrates the substitution steps. The forward pass handles the block from left to right. In this pass, for each symbol $b_i$ in the input block $B$, a new Fresnel zone radius $R$ is calculated using model (2). Then, a value $k$ is derived from $R$ and XORed with the pixel value $b_i$ to produce the substitute pixel $s_i$, which is added to the intermediate output, $out_1$. The values of $d$ and $f$ are updated using the pixel $b_i$ as described by Algorithm 2.

To amplify the effect of bit variations in the input and enhance the substitution's efficiency, the backward pass reassigns new long integers to $d$ and $f$. In this pass, the pixels of the block are processed in reverse order, moving from right to left, while maintaining the same calculations as used in the forward pass. This bidirectional processing ensures that change to any single pixel propagates both forwards and backwards, affecting all pixels in the block. As a result, each change in the input significantly impacts the entire output block, enhancing the substitution operation's robustness and sensitivity.

The substitution inverse operation, detailed in Algorithm 4, employs a two-pass decryption process — first performing a backward pass followed by a forward pass — to recover the original data from the encrypted output sequence. Both passes involve calculations similar to those used in the substitution operation. In each pass, the algorithm XORs each input pixel with a value $k$, which is obtained from the parameter $R$ computed using the Fresnel formula and then reduced modulo 256. An important aspect of ensuring the success of the substitution inverse is the correct initialization and assignment of the Fresnel model parameters $d$ and $f$ at the start of both passes. These parameters must be assigned in the same order and with the same values as they were during the initial substitution operation.

## 4. Blurring code generation

Blurring codes are a powerful technique for enhancing the security of ciphered images by significantly increasing their confusion and complexity. In this section, we introduce a specialized neural network called the differential neural network and describe the algorithmic steps for generating blurring codes using this network.

**Algorithm 3** Substitute (B<$b_1 b_2 ... b_n$>)

```
**********Forward pass**********
Assign f and d with long integers (64 bits each)
for i = 1 to n do
    b_i ← B[i]
    Compute R from Fresnel model (2)
    k ← Long(R) mod 256
    s_i ← k ⊕ b_i
    out_1<s_1 s_2 ...> ← Append (s_i)
    Update-df(R, b_i)
end for
**********Backward Pass**********
Assign d and f with long integers
for i = n down to 1 do
    s_i ← out_1[i]
    Compute R from Fresnel model (2)
    k ← Long(R) mod 256
    c_i ← k ⊕ s_i
    out<c_1 c_2 ...> ← Append (c_i)
    Update-df(R, s_i)
end for
Return out
```

**Algorithm 4** Substitute-Inv ($out$<$c_1 c_2 ... c_n$>)

```
**********Backward Pass**********
Assign d and f with long integers (64 bits each)
for i = n down to 1 do
    c_i ← out[i]
    Compute R from Fresnel model (2)
    k ← Long(R) mod 256
    s_i ← k ⊕ c_i
    out_1<s_1 s_2 ...> ← Append (s_i)
    Update-df(R, s_i)
end for
**********Forward Pass**********
Assign f and d with long integers
for i = 1 to n do
    s_i ← out_1[i]
    Compute R from Fresnel model (2)
    k ← Long(R) mod 256
    b_i ← k ⊕ s_i
    B<b_1 b_2 ...> ← Append (b_i)
    Update-df(R, b_i)
end for
Return B
```

### 4.1. Differential neural network

A neural network comprises interconnected neurons that communicate through weighted links. These links transmit information between neurons in adjacent layers, where each link's weight indicates the importance of the information computed by neuron $N_{i-1}$ in the previous layer for neuron $N_i$ in the next layer. While neural networks are commonly used for classification, learning, and image steganography tasks [41,42], their potential extends to cryptographic applications, albeit less frequently explored.

We introduce a specialized neural network, called a differential neural network, designed to meet the specific and rigorous requirements of cryptographic applications (see Fig. 1). Although each neuron functions similarly to those in conventional neural networks, we introduce two fundamental modifications to the input feedback and bias value computations.

The first modification involves using input blocks (image pixels) and the neural network outputs to calculate bias values. This is achieved by the Differential Bias Module (Fig. 1), which intercepts the neural network output and applies Eq. (3) to determine the bias values. In this equation, $c_i$ ($i = 1 \dots n$) represents the pixel values in the block, and $o_i$ represents the neural network's output

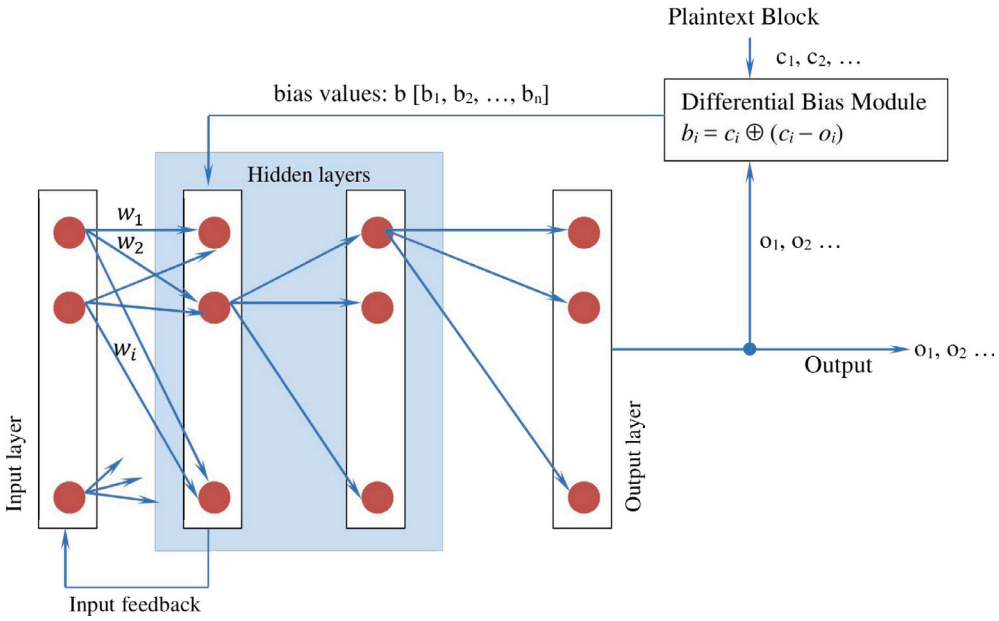



**Fig. 1.** The differential neural network structure.

symbols. The differences between the block's pixels and the output layer's results are calculated, and an XOR operation is performed between the pixels of the previous block and these differences to generate new bias values. This process establishes a direct link between the neural network state and the image pixels, ensuring that changes in image pixels will affect the network state.

$$b_i = c_i \ XOR \ (c_i - o_i), \ \ for \ i = 1 \ to \ w \tag{3}$$

The second modification involves feeding back internal values from the first hidden layer's neurons to the input layer's corresponding neurons. These two modifications introduce three significant effects: they allow differential impacts to influence the neural network's inputs and subsequent computations, prevent correlations between input feedback and network output, and introduce variability into the network's computations, which depend on changes in both image pixels and internal processes.

### 4.2. Blurring code generator

Algorithm 5 shows the steps for generating blurring codes using the differential neural network. During the initialization steps (1–3), the neural network's weights $w^{i,j}_{i-1,k}$ are assigned chaotic values using the logistic map chaotic system. Here, $w^{i,j}_{i-1,k}$ represents the weights from the neurons $k = 1 \ldots n$ of the layer $i-1$ to the neuron $j$ in the layer $i$. The input layer neurons are initialized with values from the encryption key, where each neuron $i$ receives the value of the corresponding key symbol $k_i$. The bias vector $b[]$ is initialized to zero. These initialization steps are performed once for each input image to be encrypted and are used to generate blurring codes for all the input image blocks.

The computation of the blurring codes is performed iteratively using steps 4–16 in Algorithm 5. The internal values for the neurons of the hidden layers are calculated using the corresponding equations in steps (8) or (10), depending on the hidden layer involved. In both equations, $P_{i-1,k}$ represents the internal state value of neuron $k$ in layer $i-1$, and $o_{i,j}$ denotes the internal state of neuron $j$ in layer $i$. For neurons in the first hidden layer, the bias vector values are included in the calculations, whereas subsequent hidden layers use the same equation without the bias vector $b[]$. The outputs $o_1, o_2, \ldots$ of the neural network are used as blurring codes and are also used in step (14) to update the bias vector using Eq. (3). Eq. (3) considers the pixels $c_1, c_2, \ldots$ of the input block and the output values $o_1, o_2, \ldots$ of the neural network, ensuring that the next neural network computation is influenced by the encryption key, the input image block's pixels, and the previous network output. Finally, in step (15), the internal values of the neurons in the first hidden layer are fed back into the corresponding neurons of the neural network's input layer.

## 5. Pixel perturbation operation

The pixel perturbation operation induces structural changes to the original pixel positions by randomly scrambling the pixels' positions. Algorithm 6 shows the main steps for the pixel scrambling. Initially, the first pixel at position (1, 1) is moved to a randomly selected position ($r$, $c$), where $r$ and $c$ are chaotic numbers derived from the chaotic system. For each subsequent pixel, the most recently moved pixel, $s_{i-1}$, determines the position of the next pixel, $s_i$, through the *Update*() function, as shown in Algorithm 7.

**Algorithm 5** Neural network based blurring code generation

**Require:** Key {Key is the encryption private key}
Initialize weights $w_i$ using chaotic numbers with values from the range [0...255]
Initialize the neurons in the input layer with values from the encryption key symbols
Bias vector $b[] = 0$
**repeat**
  **for** each hidden layer $i$ **do**
    **for** each neuron $j$ in the hidden layer $i$ **do**
      **if** first hidden layer **then**
        $o_{i,j} = \sum_{k=1}^{n} w_{i-1,k}^{i,j} \times P_{i-1,k} + b[j]$
      **else**
        $o_{i,j} = \sum_{k=1}^{n} w_{i-1,k}^{i,j} \times P_{i-1,k}$
      **end if**
    **end for**
  **end for**
  Update the bias vector $b[]$ using equation (3)
  Send the output of the first hidden layer as input feedback for more episodes
**until** desired blurring code length reached

**Algorithm 6** Image perturbation

**Require:** Image$[N \times N]$ {original image}
**Ensure:** Image_p$[N \times N]$ {perturbed image}
$r \leftarrow$ random from logistic map
$c \leftarrow$ random from logistic map
**for** $i = 1$ **to** $N$ **do**
  **for** $j = 1$ **to** $N$ **do**
    **if** Image_p$[r, c]$ is empty **then**
      Image_p$[r, c] \leftarrow$ Image$[i, j]$
      Mark Image$[i, j]$ visited
    **else**
      Search for an empty cell in the column $c$
      **if** found **then**
        Place Image$[i, j]$ in this cell
      **else**
        Search in each of the next columns until one is found, place Image$[i, j]$ in this cell
      **end if**
    **end if**
    Update $(r, c,$ Image$[i, j])$
  **end for**
**end for**

The *Update*() function recalculates the values of $r$ and $c$ by incorporating the impact of the most recently moved pixel, $s_{i-1}$. This is achieved using two global variables, $seed_r$ and $seed_c$, both initially set to zero. These variables are updated in steps 2 and 3 to include the impact of the previous pixel. The variables are then further randomized in steps 4 and 5 using the *Randomize*() function (steps 9–13), which applies logical shifts and XOR operations. New values for $r$ and $c$ are generated by XORing their previous values with $seed_r \bmod N$ and $seed_c \bmod N$, respectively. The updated indices $r$ and $c$ are then used to relocate the pixel Image$[i, j]$ to the new position Image_p$[r, c]$.

Due to the chaotic scrambling process, swapping collisions may occur, where the target position Image_p$[r, c]$ is already occupied. In such cases, the algorithm searches for the next available position within the same column of the conflicting cell. If no free position is found in that column, the search extends to the subsequent columns until a free position is located. Since the target matrix (the matrix to which pixels are moved) is the same size as the original image, this search always succeeds.

To reverse the perturbation operation and restore the original structure of the image, the chaotic scrambling process is reversed as described in Algorithm 8. This reverse process mirrors the steps in Algorithm 6, except that instead of randomly relocating each pixel to a new position $(r, c)$, pixels at Image_p$[r, c]$ are moved back to their original locations Image$[i, j]$.

The pixel perturbation operation relies on both the encryption key and the plain image, with pixel positions scrambled based on chaotic numbers generated by the logistic map chaotic system. The *Update*() function uses global variables and logical operations to refine the chaotic positions, ensuring that each pixel's new location is influenced by the previous pixel's position. This iterative and interdependent update mechanism enhances the randomness and structural changes in the scrambling process.

**Algorithm 7** Update chaotic position

Update($r, c, s$)
  $Seed\_r \leftarrow Seed\_r \oplus (s/256)$
  $Seed\_c \leftarrow Seed\_c \oplus (s \bmod 256)$
  Randomize($Seed\_r$)
  Randomize($Seed\_c$)
  $r \leftarrow r \oplus (Seed\_r \bmod N)$
  $c \leftarrow c \oplus (Seed\_c \bmod N)$
**Return** $r, c$
Randomize($seed$)
  $seed \leftarrow seed \oplus (seed \ll 21)$
  $seed \leftarrow seed \oplus (seed \gg 35)$
  $seed \leftarrow seed \oplus (seed \ll 4)$
End Randomize

**Algorithm 8** Image perturbation inverse

**Require:** Image_p[$N \times N$] {perturbed image}
**Ensure:** Image[$N \times N$] {original image}
$r \leftarrow$ random
$c \leftarrow$ random
**for** $i = 1$ to $N$ **do**
  **for** $j = 1$ to $N$ **do**
    **if** Image_p[$r, c$] has a value **then**
      Image[$i, j$] $\leftarrow$ Image_p[$r, c$]
      Image_p[$r, c$] $\leftarrow$ null
    **else**
      Search for non-empty cell in the column $c$
      **if** found **then**
        Image[$i, j$] $\leftarrow$ Image_p[$r, c$]
      **else**
        Search in each of the next columns until finding one
      **end if**
    **end if**
    Update($r, c$, Image[$i, j$])
  **end for**
**end for**

## 6. Encryption technique

The steps for the encryption technique are presented in Algorithm 9, while Fig. 2 illustrates the overall control flow. The encryption begins with initializations, where the logistic map parameters $r$ and $x$ are initialized using the key, $K$. Subsequently, the weights of the differential neural network links are initialized with values obtained from the logistic map.

The input image, represented as an $M \times N$ matrix of pixels, is divided row-wise into $M$ blocks $B_i$, each containing $N$ pixels. Each block $B_i$ is then encrypted using multiple operations to maximize the security of the input image. In particular, the blocks $B_i$'s are processed using the block substitution operation, producing a two-dimensional matrix $T[T_1, T_2, \ldots, T_M]$. This matrix, $T$, is further processed through a pixel perturbation operation to increase pixel diffusion. The perturbed matrix, $T$, is then subjected to another block substitution operation, resulting in the image matrix $V[V_1, V_2, \ldots, V_M]$.

Next, the differential neural network generates sequences of codes $X_i|_{i=1}^{M}$, each corresponding in length to the blocks $V_i$. The first sequence $X_1[x_1, x_2, \ldots, x_N]$ is created using the encryption key without considering the image pixels. The subsequent sequences $X_i|_{i=2}^{M}$ incorporate the impact of the image blocks $V_i$, for $i = 2 \ldots M$. Each pixel $v_i$ of the block $V_i[v_1, v_2, \ldots, v_N]$ is then XORed with the corresponding blurring code $x_i$, resulting in the ciphered pixels $c_i = v_i \oplus x_i$. This integration of image blocks $V_i$ with the differential neural network's state makes the network's output (blurring codes) dependent on both the key and the image pixels, thereby ensuring high confusion in the final encrypted image.

The steps of the decryption process are illustrated in Fig. 3. This process closely follows the steps of the encryption process shown in Fig. 2, with two major differences: (1) the encryption operations are reversed, beginning with the removal of the effects of the blurring codes and proceeding back to the substitution step, and (2) the substitution and perturbation operations are replaced with their respective inverse operations.

**Algorithm 9** Algorithmic steps for the proposed encryption technique

**Input:** Image matrix $P$ of size $M \times N$, Key $K$
**Output:** Encrypted image $C$
**Step 1: Initialization**
  Use the key $K$ to generate initial values for the parameters $r$ and $x$ of the logistic map using Algorithm 1
  Generate chaotic values using the initialized logistic map to assign weights for the differential neural network
**Image encryption: Steps 2–5**
**Step 2: Substitute the blocks of the image $P$**
**for** i =1 to M **do**
    $T_i$ = Substitute pixels in $B_i$ using Algorithm 3 (Block Substitution)
**end for**
The resulting image after the substitution is represented by the matrix $T[T_1, T_2, \dots T_M]$
**Step 3: Permute pixels of the matrix $T$**
  initialize $(c, r)$ with chaotic values from the logistic map
  Permute the pixels of block $T$ using Algorithm 6
**Step 4: Substitute the blocks of the image $T$**
**for** i =1 to M **do**
    $V_i$ = Substitute pixels in $T_i$ using Algorithm 3 (Block Substitution)
**end for**
The resulting image after the substitution is represented by the matrix $V[V_1, V_2, \dots V_M]$
**Step 5: Generate Blurring Codes using the Differential Neural Network and XOR them with the blocks of the image $V$**
**for** each $V_i$ in V **do**
  Generate blurring codes $X_i[x_1, x_2, \dots, x_N]$ using Algorithm 5
  **for** each value $v_i$ in $V_i$ **do**
      $c_i = v_i \oplus x_i$
  **end for**
**end for**
**Step 6: Form Encrypted Image**
Combine all encrypted blocks, $C_i|_{i=1}^{M}$, row-wise to form the encrypted image $C[C_1, C_2 \dots C_M]$.
**Step 7: Output Encrypted Image $C$**

The decryption process starts with the initialization of the logistic map parameters, $r$ and $x$, using the key. This initialized logistic map is used to initialize the weights of the differential neural network. Once the neural network is initialized, the decryption process handles the blocks of the encrypted image $\{C_1, C_2, \dots, C_M\}$. The impact of the blurring codes is removed from each block $C_i$ as follows: the pixels of the first block, $C_1$, are XORed with the blurring codes $X_1[x_1, x_2, \dots, x_N]$, which are generated by the neural network using the key only. For subsequent blocks $C_i$ where $i > 1$, the pixels are XORed with blurring codes $X_i$ generated by the neural network using both the key and the preceding blocks $V_{i-1}$. (Blocks $V_{i-1}$ are derived from the blocks $C_{i-1}$ after the removal of the blurring codes.)

The resulting blocks $\{V_1, V_2, \dots, V_M\}$ are then processed using the inverse substitution operation to counteract the impact of the corresponding substitution operation from the encryption process. The output blocks $\{T_1, T_2, \dots, T_M\}$ are subsequently processed by the inverse perturbation operation to restore the correct pixel organization. Finally, the restored blocks are further subjected to the inverse substitution operation to fully recover the original image.

The proposed encryption technique provides robust security through the highly sensitive generation of blurring codes and the application of substitution and perturbation operations. The blurring code generation process is nonlinearly dependent on both the encryption key and the plain image, such that even a single bit change in either the key or the input image results in substantial variations in the output. This sensitivity effectively resists hacking attempts, making it infeasible for tools to detect exploitable patterns. Moreover, the substitution and perturbation operations further enhance this sensitivity, ensuring that minor alterations in the key or image can greatly alter the encrypted output. This high responsiveness to micro changes in the input adds additional complexity, making it exceedingly difficult for unauthorized entities to decipher the image through pattern recognition or brute force methods. By leveraging these features, the encryption technique offers robust protection against unauthorized access.

## 7. Security analysis

This section presents various fundamental security tests to validate the performance of the technique. We also compare the security metrics of the proposed technique with those of recent image encryption techniques. The images used were obtained from the benchmarks: the SIPI database (https://sipi.usc.edu/database/), https://ccia.ugr.es/cvg/dbimagenes/, and Google Image Search. Table 1 introduces notations that help readers understand the significance of the findings, while Table 2 shows the experimental configurations. Fig. 4 shows the encryption and decryption results for a sample of medical and non-medical images.

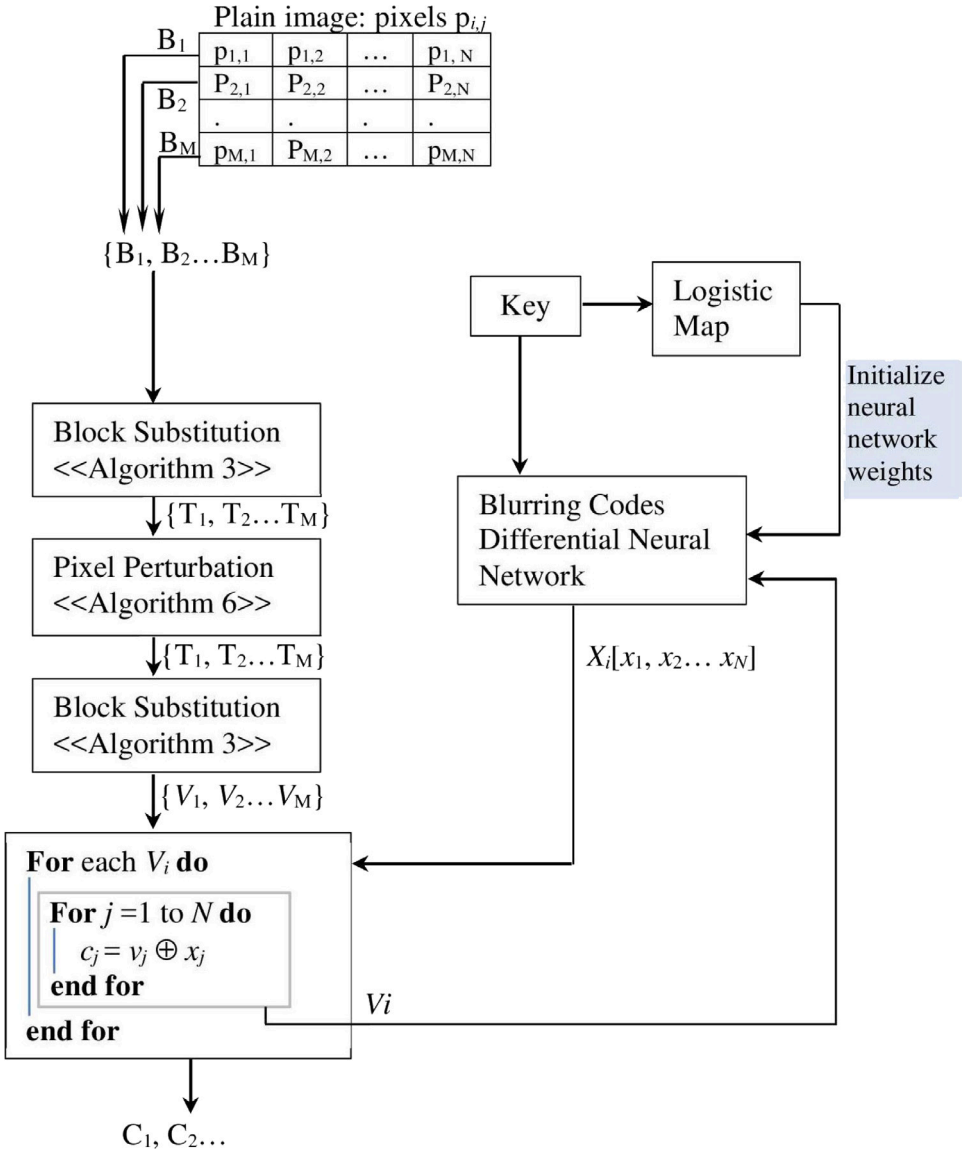



**Fig. 2.** The encryption technique.

**Table 1**
Notation for improved readability of results.

| Notation | Description |
|---|---|
| ↑ and ↓ | Indicate that higher or lower values are preferred, respectively |
| $\downarrow^m$ | Indicates that significant values should be less than $m$ |
| $\uparrow^m$ | Indicates that values closer to $m$ are preferred |
| $\downarrow_m$ | Indicates that significant values should be small and close to $m$ |

### *7.1. Key space*

This technique employs four subkeys to ensure robust encryption. Two subkeys, each 64 bits long, are used for the Fresnel zone, while the other two subkeys, each with double precision of $10^{16}$ and greater than $2^{48}$, are used for initializing the logistic map. The resulting key space is calculated as $(2^{64})^2 \times (2^{48})^2$, which equals $2^{224}$. This key space is significantly larger than the secure key space of $2^{100}$ [18]. The vast key space provides the proposed technique with a high level of security against brute-force attacks or other attacks that exploit the limited key space.

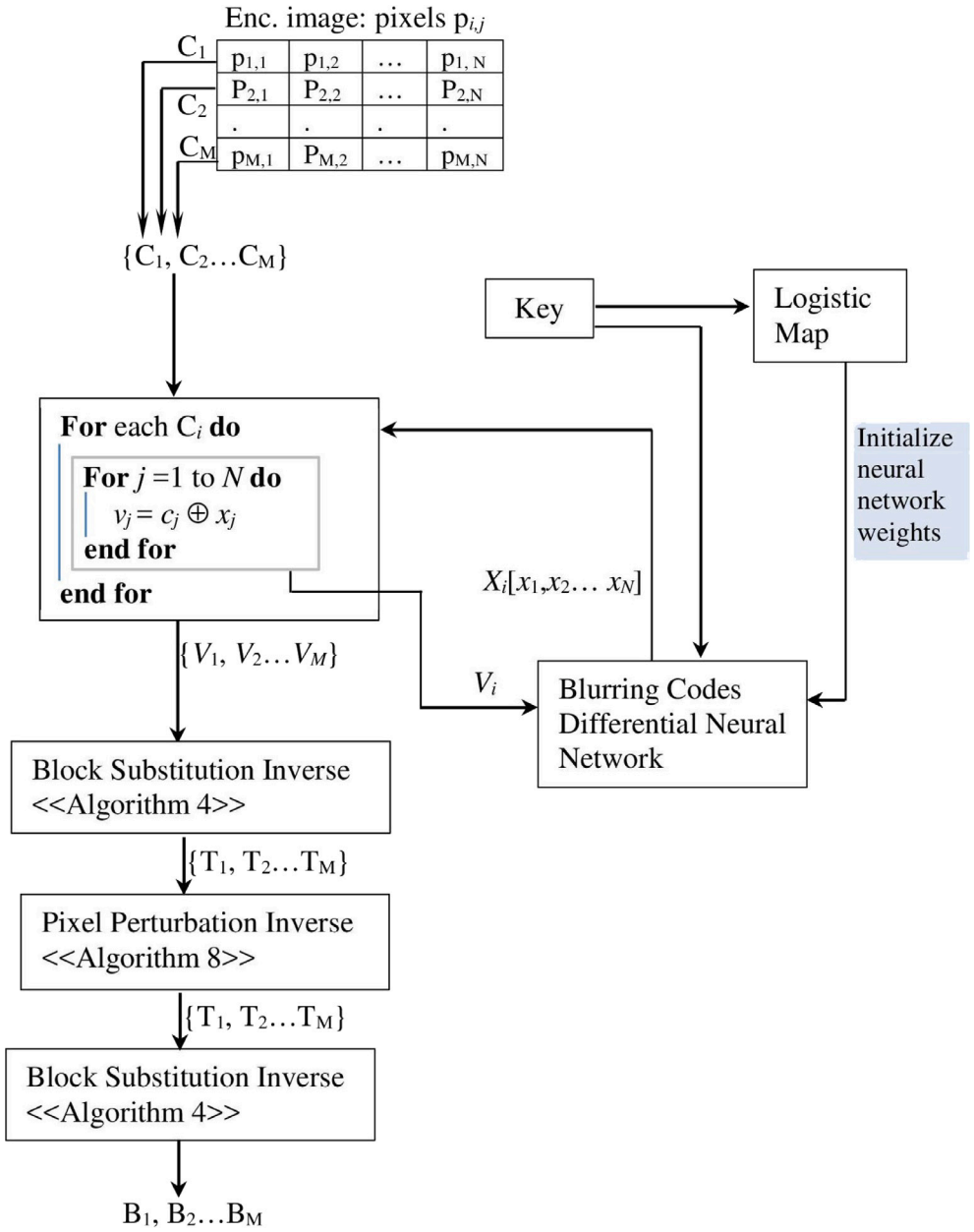



**Fig. 3.** The decryption technique.

**Table 2**
Experimental configurations.

| Parameter | Description and configuration |
|---|---|
| Neural network settings | 1 input layer<br>3 hidden layers<br>1 output layer |
| Number of neurons in each layer | Depends on the key, which is 16 in our experiments |
| Image block size | The block size corresponds to the row size in the image's matrix |
| Sample size | 100 images |

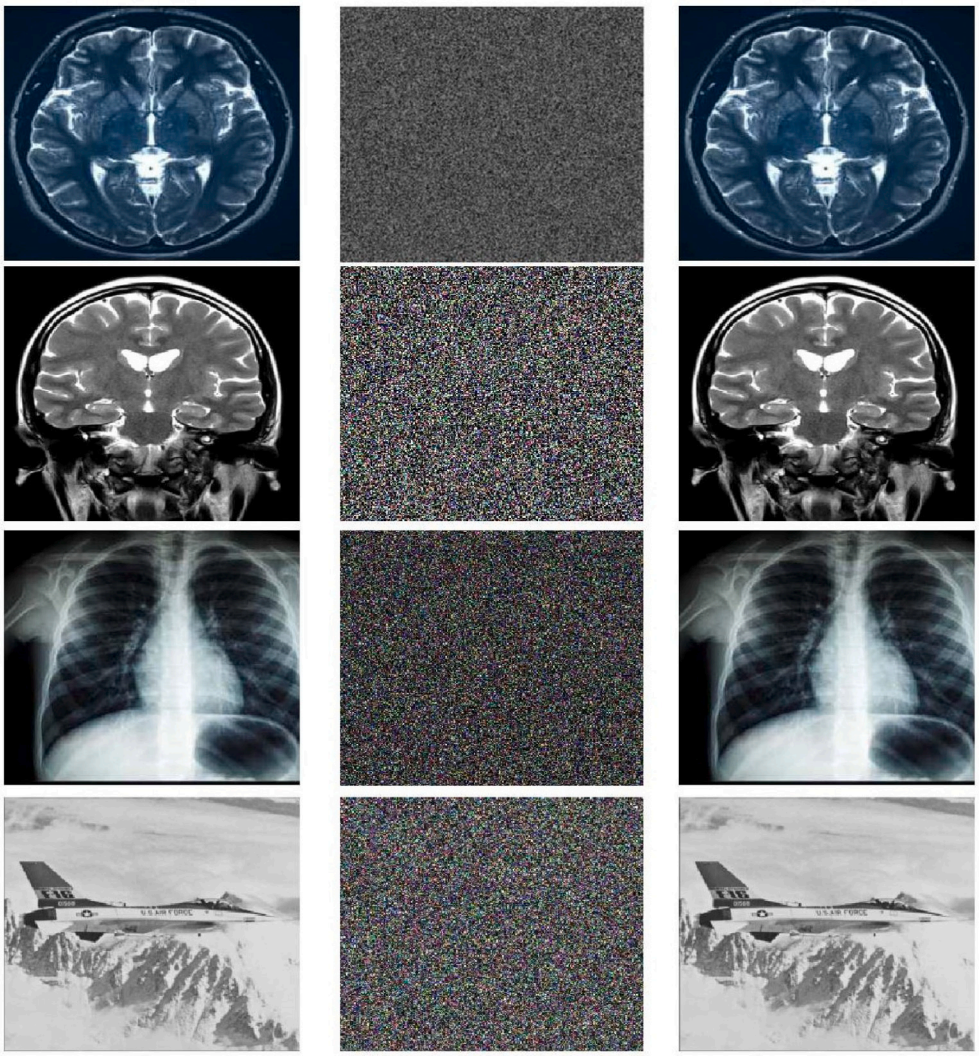


**Fig. 4.** Sample of images, their encryption, and decryption.

### 7.2. Information entropy

The entropy indicates the level of randomness of the pixel distribution. Since the pixel values are the ASCII's in the range [0..255], an entropy of 8 is optimal. The global entropy for a set of pixels $Z$ can be calculated using the formula (4).

$$H(Z) = -\sum_{i=0}^{w} p(z_i) log_2(p(z_i)) \tag{4}$$

where $w$ is the number of pixels, $z_i$ is the number of occurrences of the pixel $i$ in the image, and $p(z_i)$ is the relative frequency of $z_i$.

To evaluate the entropy, we used 40 medical and nonmedical images obtained from our benchmark resources. Table 3 shows the average, minimum, and maximum values for the entropy over 17 medical images and 23 non-medical images. Low entropy values indicate a non-random distribution of pixels, while large values approaching 8 indicate increasing randomness. As shown in the table, the average entropy values are significantly high (indicated by **bold**), nearing the optimal value of 8 (denoted by $\uparrow^8$). Table 4 compares the entropy values of images encrypted with the proposed technique to those encrypted with recent techniques. For all plain images, the entropy is 7 or lower. However, all techniques, including the proposed one, significantly increased the entropy values, bringing them closer to the ideal. Remarkably, the images encrypted by the proposed technique achieve entropy values that are closer to the ideal than most other image encryption techniques. This indicates high performance of the proposed technique.

**Table 3**
Average, minimum, and maximum entropy values for a sample of 40 images.

| Image type | Entropy ($\uparrow^{8}$) | Min | Max |
|---|---|---|---|
| Medical images: 17 images | **7.9990** | **7.9991** | **7.9999** |
| Nonmedical images: 23 images | **7.9995** | **7.9989** | **7.9998** |

**Table 4**
Entropy: proposed technique compared to other related techniques.

| Technique | Lena | Barbara | Baboon | Peppers | Brain MRI | Chest Xray | Backbone Xray |
|---|---|---|---|---|---|---|---|
| Proposed | 7.9999 | 7.9998 | 7.9999 | 7.9995 | 7.9997 | 7.9999 | 7.9991 |
| [43] | 7.9967 | – | 7.9979 | 7.9971 | – | – | – |
| [44] | – | 7.9987 | 7.9987 | 7.9987 | – | – | – |
| [45] | 7.9975 | – | 7.9975 | – | – | – | – |
| [46] | 7.9994 | 7.9995 | 7.9995 | 7.9994 | – | – | – |
| [47] | – | – | – | – | 7.9993 | 7.9993 | 7.9993 |
| [13] | – | – | – | – | 7.9987 | 7.9987 | 7.9987 |
| [48] | – | – | – | – | 7.9983 | 7.9983 | 7.9983 |
| [49] | – | – | – | – | 7.9992 | 7.9992 | 7.9992 |
| [50] | – | – | – | – | 7.9982 | 7.9932 | 7.9982 |

**Table 5**
$\chi^2$ values for a sample of the testing images.

| Images | $\chi^2$ values ($\downarrow^{300}$) | |
|---|---|---|
| | Plain | Ciphered |
| Chest X-ray | 101 387 | **277.89** |
| Lena | 172 009 | **131.77** |
| Peppers | 192 063 | **217.66** |
| Airplane | 497 888 | **152.12** |
| Brain MRI | 98 765 | **243.35** |
| Backbone X-ray | 90 881 | **117.83** |

### 7.3. Histogram study

The histogram is a useful tool for analyzing the distribution of pixels in an image. In an ideal random distribution, each pixel would have an equal chance of occurrence, which is accurately represented by a histogram with uniformly distributed bars. Fig. 5 shows the histograms. In Fig. 5(a), the histograms of the images are shown before encryption, while in Fig. 5(b), the histograms for the images are shown after encryption. The correlation between pixels in the original image is reduced after encryption, as seen by the evenly distributed bars in the encrypted image's histogram. This is in contrast to the original image's histogram, where the bars vary significantly.

The chi-square ($\chi^2$) test measures if the observed distribution of the image's pixels is different from the expected distribution under the randomness assumption. For this test, $\chi^2$ is computed using the formula (5).

$$\chi^2 = \sum_{i=0}^{n-1} \frac{(E(i) - E)^2}{E} \tag{5}$$

where $E(i)$ is the observed frequency of the pixel $i$, and $E = (N \times M)/n$ is the expected number of each pixel under the random distribution assumption ($M$ and $N$ are the dimensions of the image).

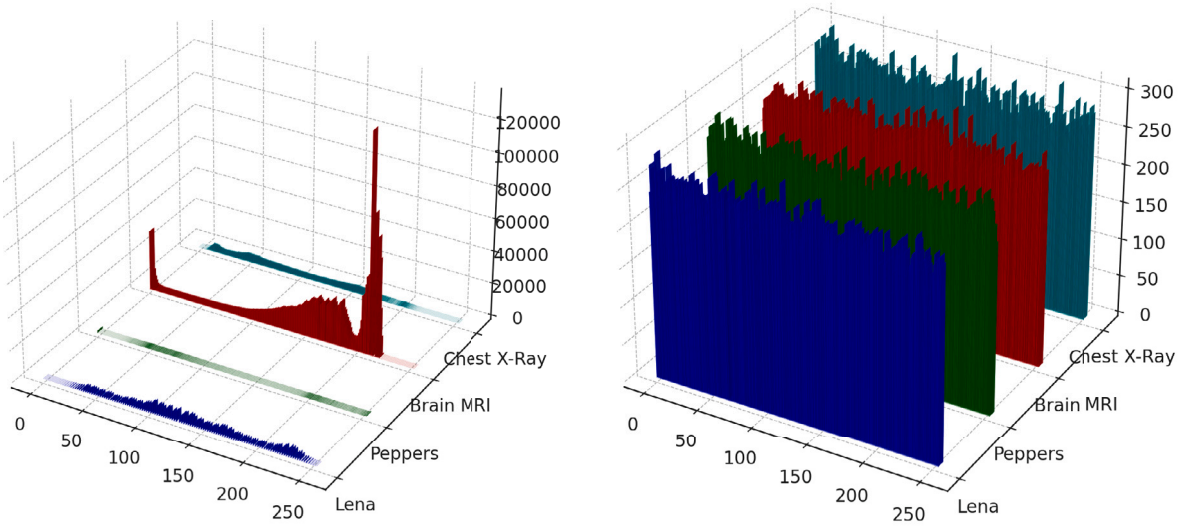
Table 5 shows $\chi^2$ values for a sample of images, with significant values highlighted in **bold**. Small values, where $\chi^2 < 300$ (visually denoted in the table by $\downarrow^{300}$), indicate consistency between the expected and the observed distribution of pixels, while large values indicate inconsistency. As shown in the table, the $\chi^2$ values for all plain images are significantly greater than 300, indicating inconsistency between the expected and observed pixel distributions. In contrast, all $\chi^2$ values for the encrypted images are below 300, demonstrating consistency and indicating that the encryption technique is secure.

### 7.4. Pixel correlation analysis

Adjacent pixels of images often have similar or identical intensities, leading to high correlation. An effective encryption technique must substantially reduce this high correlation; otherwise, the security of the encrypted image can be easily compromised. The correlation between adjacent pixel pairs $(x, y)$ can be measured by formula (6) from [51].

$$\rho_{xy} = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{6}$$

where $cov(x, y) = \frac{1}{M}\sum_{i=1}^{M}(x_i - Exp(x))(y_i - Exp(y))$, $D(z) = \frac{1}{M}(z_i - Exp(z))$, and $Exp(w) = \frac{1}{M}\sum_{i=1}^{M} w_i$.




(a) Histograms for plain images (b) Histograms for ciphered images

**Fig. 5.** The histograms for a sample of images.

**Table 6**
Correlation between adjacent pixels along Horizontal (H), Vertical (V), and Diagonal (D) axes.

| Image | Plain images | | | Encrypted images ($\downarrow_0$) | | |
|---|---|---|---|---|---|---|
| | H | V | D | H | V | D |
| Lena | 0.9703 | 0.9681 | 0.9445 | **0.0001** | **0.0028** | **−0.0007** |
| Brain MRI | 0.9865 | .0.9811 | 0.9877 | **−0.0005** | **0.0005** | **−0.0007** |
| Chest X-ray | 0.9708 | 0.9712 | 0.9783 | **−0.0033** | **−0.0009** | **−0.0009** |
| Pepper | 0.9666 | 0.9804 | 0.9689 | **−0.0018** | **0.0006** | **−0.0034** |
| Barbara | 0.8911 | 0.9633 | 0.8777 | **−0.0077** | **−0.00007** | **0.00317** |
| Baboon | 0.9534 | 0.91004 | 0.8956 | **−0.0002** | **0.00102** | **−0.00133** |
| Boat | 0.9377 | 0.9645 | 0.9217 | **0.0005** | **0.0004** | **−0.0007** |
| Airplane | 0.9633 | 0.9701 | 0.96655 | **0.0003** | **0.0008** | **−0.0003** |

**Table 7**
Comparing the correlation for different techniques.

| Lena | | | | Baboon | | | |
|---|---|---|---|---|---|---|---|
| Tech. | H | V | D | Tech. | H | V | D |
| prop. | **0.0001** | **0.0028** | −0.0007 | prop. | **−0.0002** | **0.00102** | −0.00133 |
| [52] | 0.0033 | 0.0070 | 0.0027 | [53] | 0.0058 | 0.0051 | −0.0009 |
| [43] | 0.0069 | 0.0069 | 0.0069 | [44] | −0.0075 | −0.0071 | 0.0041 |
| [54] | 0.0070 | 0.0070 | −0.0080 | [55] | 0.0029 | 0.0014 | −0.0006 |
| [44] | −0.0076 | −0.0071 | 0.0042 | [52] | 0.0025 | 0.0064 | 0.0035 |
| [56] | 0.011 | −0.0032 | −0.0001 | [57] | −0.0005 | 0.0032 | −0.0026 |
| [58] | −0.0036 | −0.0045 | −0.0041 | [58] | −0.0036 | −0.0014 | −0.0065 |

Table 6 compares the pixel correlation of images before encryption (plain images) and after encryption (encrypted images) along the vertical, diagonal, and horizontal axes. The results show a substantial decrease in correlation after encryption, with values approaching 0 (denoted by $\downarrow_0$). Table 7 compares the proposed technique with recent image encryption techniques. While all techniques effectively reduce pixel correlation in encrypted images, in some instances (highlighted in **bold**), the proposed method achieves even lower correlation values than other approaches. Fig. 6 provides the visual correlation between pixels of the encrypted Lena and Chest X-ray images across the three dimensions. As the figure shows, the scattering of the encrypted image pixels is random.

### *7.5. Differential attacks*

Encryption techniques should be highly sensitive to bit variations, detecting even slight alterations in plain images and generating extremely different ciphered images in response. Literature offers two metrics, namely NPCR (number of pixels change rate) and

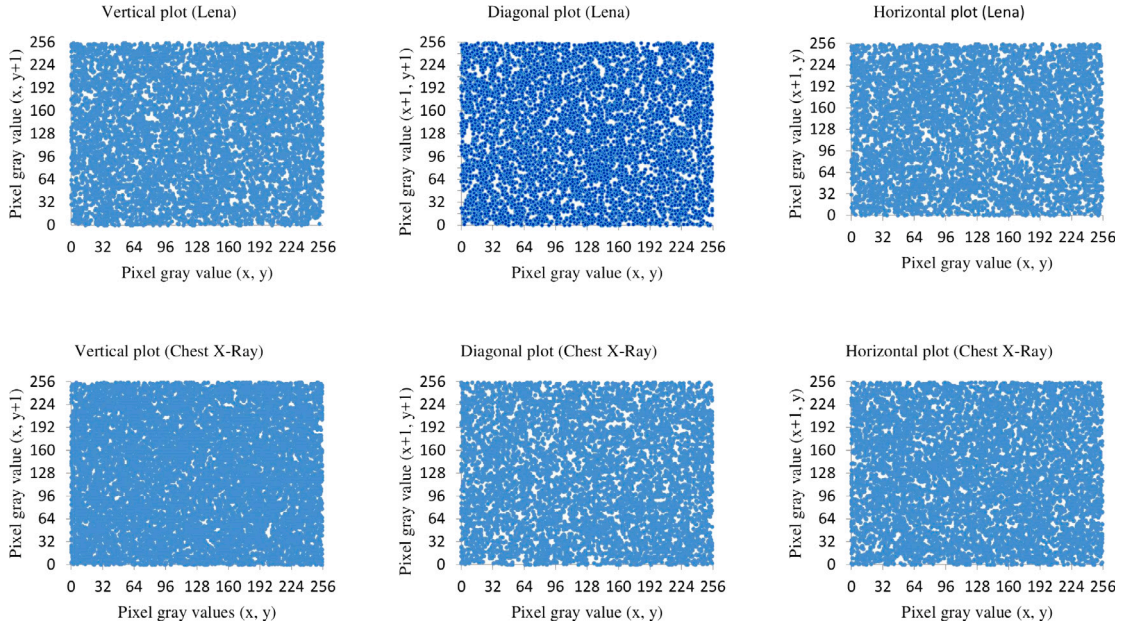



**Fig. 6.** Scatter plot for pairs of pixels along horizontal, vertical and diagonal.

**Table 8**
NPCR and UACI for a sample of images.

| Images | NPCR | UACI | Images (Medical) | NPCR | UACI |
|---|---|---|---|---|---|
| Lena | **99.6070** | **33.4700** | Brain MRI | **99.5983** | **33.4619** |
| Airplane | **99.6113** | **33.4576** | Chest X-ray | **99.6100** | **33.4496** |
| House | **99.6217** | **33.4566** | Backbone X-ray | **99.6104** | **33.4704** |
| Baboon | **99.6233** | **33.4588** | Hand X-ray | **99.5966** | **33.4444** |

**Table 9**
NPCR values for the proposed technique and a sample of state-of-the-art techniques.

| Images | Prop. | [61] | [44] | [62] | [52] | [48] | [50] | [49] | [63] |
|---|---|---|---|---|---|---|---|---|---|
| Lena | 99.6070 | 99.5729 | 99.5700 | 99.5947 | 99.6193 | – | – | – | – |
| Peppers | 99.6103 | 99.5881 | 99.5600 | 99.6185 | 99.6093 | – | – | – | – |
| Barbara | 99.5999 | – | 99.5800 | 99.6185 | 99.6219 | – | – | – | – |
| Cameraman | 99.6173 | 99.5956 | 99.5800 | 99.5978 | – | – | – | – | – |
| Brain MRI | 99.5983 | – | – | – | – | 99.6134 | 99.5911 | 99.6901 | 99.6000 |
| Chest-XRay | 99.6100 | – | – | – | | – | 99.6256 | 99.6901 | 99.5900 |
| Backbone-XRay | 99.6104 | – | – | – | – | – | 99.5990 | 99.6901 | 99.6600 |

UACI (unified average changing intensity), for determining the degree of sensitivity of a technique to bit variation. These metrics are defined by Eqs. (7) [59].

$$\begin{aligned} NPCR &= \frac{\sum_{i,j} R(i,j)}{N \times M} \times 100 \\ UACI &= \frac{1}{N \times M}\left[\sum_{i,j} \frac{|f_1(i,j) - f_2(i,j)|}{255}\right] \end{aligned} \tag{7}$$

where $M$ and $N$ are the dimensions of the images, $f_1(i,j)$ and $f_2(i,j)$ are two pixels at the same position of two encrypted images, $|x|$ is the absolute value, and $R(i,j) = 1$ if $f_1(i,j) \neq f_2(i,j)$; $R(i,j) = 0$ otherwise. The values of NPCR and UACI should be close to 99.6093% and 33.4653%, respectively for the technique to be effective [60]. Table 8 presents the NPCR and UACI values for different images. These obtained values of NPCR and UACI indicate the technique's effectiveness, as these values are close to their ideal limits. Tables 9 and 10 compare the proposed technique to other techniques based on the NPCR and UACI values. Despite all the techniques producing NPCR and UACI values that are relatively close to the ideal, the proposed technique yields NPCR/UACI values that are even closer to the ideal values.

### *7.6. Resilience against noise and information loss*

Encrypted images may encounter noise or information loss. Effective encryption techniques must robustly handle this situation and decrypt the ciphered image (even partially). The impact of adding 1% pepper and salt (PSN) and 0.1% Gaussian noise (GN) to

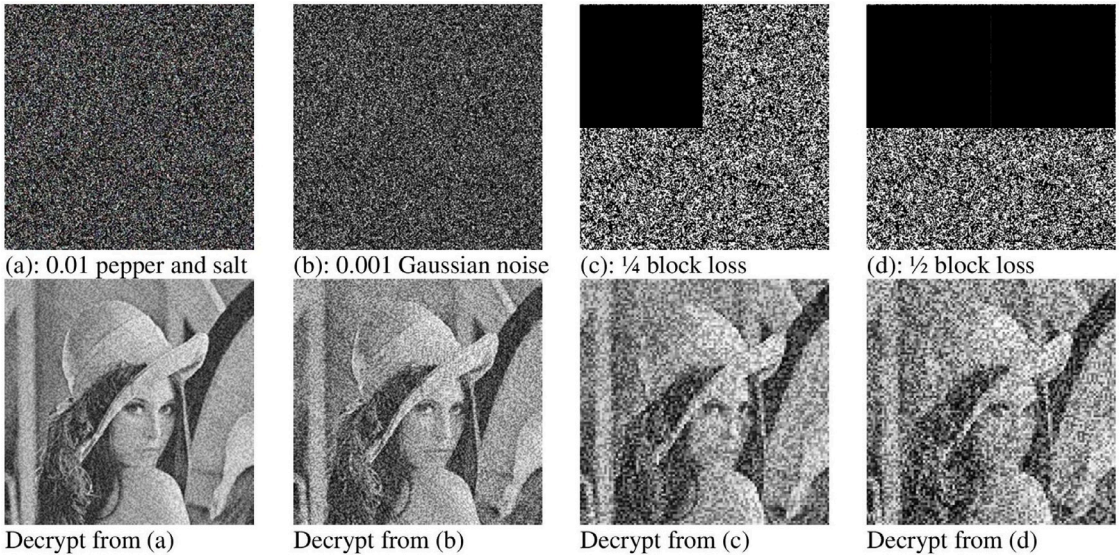



**Fig. 7.** Impact of noise and clip attacks on the encrypted images.

**Table 10**
UACI values for the proposed technique and a sample of state-of-the-art techniques.

| Images | Prop. | [61] | [44] | [62] | [52] | [48] | [50] | [49] | [63] |
|---|---|---|---|---|---|---|---|---|---|
| Lena | 33.4700 | 33.2749 | 33.3600 | 33.6414 | 33.4415 | – | – | – | – |
| Peppers | 33.4583 | 33.5134 | 3.3600 | 33.5488 | 33.8349 | – | – | – | – |
| Barbara | 33.4691 | – | 33.6500 | 33.4204 | 33.4635 | – | – | – | – |
| Cameraman | 33.4711 | 33.4440 | 33.4000 | 33.2515 | – | – | – | – | – |
| Brain MRI | 33.4619 | – | – | – | – | 33.4683 | 33.5260 | 33.5260 | 33.5300 |
| Chest-XRay | 33.4496 | – | – | – | – | – | 33.5658 | 33.5260 | 33.5800 |
| Backbone-XRay | 33.4704 | – | – | – | – | – | 33.4931 | 33.5260 | 33.4400 |

**Table 11**
PSNR (desired $\uparrow_{20}$) and MSE (desired $\downarrow_0$) values for a sample of images.

| | Brain MRI | | | | Lena | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.01% PSN | 0.001% GN | 1/4 | 1/2 | 0.01% PSN | 0.001% GN | 1/4 | 1/2 |
| MSE | 264.168 | 102.986 | 77.639 | 629.912 | 393.895 | 244.219 | 272.321 | 480.264 |
| PSNR | 23.912 | 28.003 | 29.230 | 20.138 | 22.177 | 24.253 | 23.780 | 21.316 |

the encrypted images, as well as losing part of a block information, is shown in Fig. 7. While these attacks did impact the quality of the decrypted images, they remained reasonably recognizable. However, the decrypted images were less recognizable for clip attacks.

The Peak Signal-to-Noise Ratio (PSNR) is an indicator used to measure the similarity between two images. We evaluated the robustness of the proposed technique against noise and information loss attacks by determining how closely the decrypted defected image resembles the original image, using the PSNR equation (8).

$$\begin{aligned} MSE &= \frac{1}{N\times M}\sum_{i=1}^{N}\sum_{j=1}^{M}(D(i,j)-O(i,j))^2 \\ PSNR &= 10.log_{10}\frac{255^2}{MSE} \end{aligned} \tag{8}$$

where $D(i,j)$ and $O(i,j)$ represent the pixels at the same locations $i$ and $j$ in the decrypted and original images, respectively. PSNR values above 20 (with decreasing MSE values, ideally zero) indicate increasing similarity between two images, while values below 20 (with higher MSE values) suggest greater differences.

Table 11 shows the MSE and PSNR values for 0.01% PSN (Pepper and Salt Noise), 0.001% GN (Gaussian Noise), and for 1/4 and 1/2 block information loss. (In the table title, PSNR $\uparrow_{20}$ and MSE $\downarrow_0$ indicate that PSNR values greater than 20 and decreasing MSE values are desired.) The MSE values are relatively small, and the PSNR values are all above 20, indicating a high similarity between the decrypted defected and original images. This demonstrates the technique's strong resilience against noise and information loss attacks.

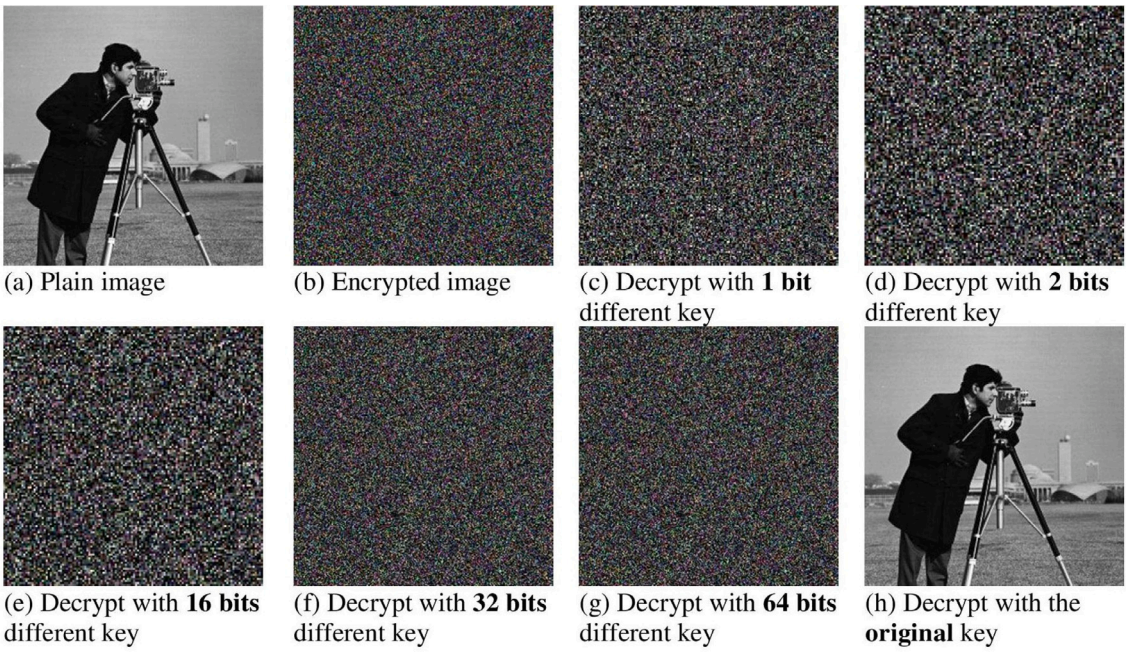



**Fig. 8.** Technique robustness under different wrong key attempts.

**Table 12**
SSIM values (desired values $\downarrow_0$).

| Image | Average SSIM | | | | | | |
|---|---|---|---|---|---|---|---|
| | $K$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K_6$ |
| Brain MRI | 1.0 | 0.000769 | 0.000998 | 0.001052 | 0.000479 | 0.000703 | 0.002008 |
| Chest X-ray | 1.0 | 0.000777 | 0.000313 | 0.000991 | 0.000451 | 0.001007 | 0.003237 |
| Backbone X-ray | 1.0 | 0.000333 | 0.0000808 | 0.000223 | 0.0000893 | 0.000335 | 0.000709 |
| Cameraman | 1.0 | 0.000088 | 0.000987 | 0.000155 | 0.000309 | 0.000166 | 0.000235 |

### *7.7. Response to key modification*

The robustness of the encryption method relies on the encryption key, which must generate incorrect decryption results for altered keys. The decryption attempts, shown in Fig. 8, demonstrated that only the appropriate key could correctly decrypt the image. In contrast, all keys that differed by several bits failed to decrypt the image.

Additionally, we quantitatively evaluated the sensitivity of the proposed technique to encryption key changes using the Structural Similarity Index Measure (SSIM), a widely used technique for comparing two images, as defined by the formula in Eq. (9) [64].

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{9}$$

Here, $\mu_x$ and $\mu_y$ are the average values of the images $x$ and $y$, respectively. $\sigma_x^2$ and $\sigma_y^2$ represent the variances of $x$ and $y$, while $\sigma_{xy}$ denotes their covariance. $C_1$ and $C_2$ are small constants to stabilize the division when the denominators are close to zero.

To compute SSIM, we encrypted four images (Cameraman, Brain MRI, Chest X-ray, and Backbone X-ray) using the original key $K$. We then decrypted them with six modified keys $[K_i]_{i=1}^{6}$, each created by flipping $i$ randomly selected bits of $K$. For each $K_i$, we generated 10 versions with different flipped bit positions. Table 12 shows the average SSIM values. The SSIM values range from 0 to 1, with 1 indicating identical images and 0 indicating no similarity. When decrypted with the original key $K$, the SSIM was 1, indicating identical images. However, when decrypted with perturbed keys $[K_i]$, the values of SSIM were close to zero, indicating no similarity. These numbers demonstrate that the technique is highly sensitive to key changes since minor modifications to the key result in completely incorrect decryption.

### *7.8. CPU time complexity*

CPU time is critical indicator for the practical use of encryption algorithms. The proposed algorithm was implemented using JAVA, and the experiments were conducted on a Core i5 processor with 4 GB memory and Windows 10 operating system. The

**Table 13**
CPU time (seconds) for the proposed algorithm and recent algorithms.

| Image size | Proposed | [54] | [65] | [44] | [46] | [53] | [66] | [52] |
|---|---|---|---|---|---|---|---|---|
| 256*256 | 0.0954 | 0.0779 | 0.0620 | – | – | 1.7715 | 0.2791 | 0.3493 |
| 512*512 | 0.3011 | 0.3261 | 0.6728 | 0.8572 | 0.8775 | – | – | – |

**Table 14**
CPU time (seconds) for medical images.

| Image size | Proposed | [47] | [50] | [67] | [48] | [68] |
|---|---|---|---|---|---|---|
| 256*256 | 0.2791 | 0.2788 | 0.3289 | 0.9400 | 0.5080 | – |
| 512*512 | 0.4333 | 1.712 | 0.4646 | – | – | 0.4437 |

**Table 15**
NIST randomness test results for a set of 100 images.

| NIST randomness test | Ciphered (passed/total) | Max Fail |
|---|---|---|
| Runs | 100/100 | 11 |
| Monobit | 100/100 | 11 |
| Spectral | 95/100 | 11 |
| Serial | 98/100 | 11 |
| Cumulative sums | 98/100 | 11 |
| Non-overlapping template matching | 99/100 | 11 |
| Overlapping template matching | 94/100 | 11 |
| Linear complexity | 96/100 | 11 |
| Binary matrix rank | 96/100 | 11 |
| Maurer's "Universal Statistical" | 97/100 | 11 |
| Approximate entropy | 99/100 | 11 |
| Longest runs of ones | 98/100 | 11 |

time study was performed using a sample of medical and non-medical images with different sizes. Tables 13 and 14 summarize the CPU time for the proposed algorithm as well as other recent algorithms. The proposed algorithm stands out in comparison to other existing algorithms due to its remarkably short execution time, making it highly suitable for real-world encryption tasks.

### 7.9. Randomness test: NIST

The performance of the proposed technique was evaluated using a benchmark set of 100 images (from our benchmark). These images were encrypted using the proposed method, resulting in 100 encrypted sequences, each consisting of at least 1,048,576 bits (depending on the image). We used randomness tests provided by the National Institute for Standards and Technology (NIST) [69]. We used the default test parameters specified by the NIST implementation without any modifications. Table 15 presents the results of randomness tests for each ciphered image. The results are shown in terms of the number of ciphered images that passed a test ("passed/total") and the maximum number of ciphered images that may fail due to random variations ("Max Fail"). The maximum number of ciphered images that may fail for a significant level $\alpha$ is computed using NIST formula (10), with $\alpha = 0.05$ in our testing.

$$Max\ Failure = S.(\alpha + 3.\sqrt{\frac{\alpha(1-\alpha)}{S}}) \tag{10}$$

Although a few encrypted images failed one or more tests, the number of failures remains below the maximum threshold expected due to random variations. This indicates that the output generated by the proposed technique is, overall, both random and secure.

### 7.10. Expanded benchmark evaluation: Average value analysis

In this subsection, we present performance findings based on a comprehensive benchmark of images. The images, referenced in a previous subsection, include 40 samples: 17 medical and 23 non-medical. Table 16 shows the average values for an array of various security test indicators. All the average values are within their ideal ranges. For instance, the correlation values are close to zero (ideal value), entropy values are close to 8, and $\chi^2$ values are less than the threshold of 300.

Table 17 displays the PSNR and MSE averages after simulating noise and data loss attacks on the 40 images. Despite the attacks, the average PSNR values for decrypted images — affected by 0.01% PSN (salt and pepper noise), 0.001% GN (Gaussian noise), 1/4 block loss, and 1/2 block loss — remain above 20, indicating that the quality of decrypted images, while impacted, is still distinguishable.

**Table 16**
Average values for a set of standard security indicators.

| Image type | Entropy | NPCR | UACI | Correlation | $\chi^2$ | SSIM | Time (s) |
|---|---|---|---|---|---|---|---|
| Medical | 7.9990 | 99.5991 | 33.4567 | 0.00092 | 199.904 | 0.000889 | 0.4428 |
| Nonmedical | 7.9995 | 99.6001 | 33.4487 | 0.00088 | 201.003 | 0.000723 | 0.3055 |

**Table 17**
Average PSNR and MSE values for benchmark images.

| Image type | Test | 0.01% PSN | 0.001% GN | 1/4 block loss | 1/2 block loss |
|---|---|---|---|---|---|
| Medical | PSNR | 24.294 | 27.828 | 29.057 | 20.166 |
| | MSE | 241.981 | 107.233 | 80.455 | 625.998 |
| Nonmedical | PSNR | 23.549 | 24.200 | 23.822 | 23.183 |
| | MSE | 287.203 | 247.219 | 269.674 | 495.111 |

**Table 18**
Security metrics comparison before and after removing the substitution operation.

| | Before ablation | | After ablation | |
|---|---|---|---|---|
| | Brain MRI | Cameraman | Brain MRI | Cameraman |
| Entropy | 7.9997 | 7.9993 | 7.8016 | 7.9111 |
| Correlation | 0.0007 | 0.00201 | 0.00963 | 0.08777 |
| NPCR | 99.5983 | 99.6173 | 98.1700 | 95.6667 |
| UACI | 33.4619 | 33.4711 | 31.3297 | 31.0377 |
| $\chi^2$ | 243.350 | 199.708 | 308.961 | 407.002 |

### *7.11. Ablation analysis*

The encryption technique integrates three orthogonal operations: deep substitution (Section 3), data-dependent chaotic pixel perturbation (Section 5), and pixel confusion through differential-network blurring code generation. Collectively, these operations create three crucial layers of security, enabling the technique to resist various forms of attacks.

- **Deep Substitution**: This operation transforms the plain image pixels into a completely different and independent space, erasing any correlation with the original image. By leveraging the Fresnel formula, the substitution operation uses input pixels and randomization to generate substitution elements, ensuring that the substituted image occupies an entirely new space.
- **Chaotic Pixel Perturbation**: This step disrupts the inherent structural relationships among pixels, enhancing pixel diffusion. The scrambling of pixels results in a uniform pixel distribution in the encrypted image, making it highly resistant to statistical attacks that exploit pixel distribution patterns.
- **Differential-Network Blurring Code Generation**: This operation adds substantial complexity to the ciphered image by incorporating both pixel variation and the encryption key's impact. This dual influence strengthens the technique against advanced attacks, such as differential attacks.

As a result, from an analytical perspective, each of these three operations plays an integral role in ensuring the security of the proposed technique. They collectively enhance the security of the proposed technique.

To validate our analytical findings, we conducted an ablation study by removing the substitution operation while retaining the pixel perturbation and blurring code generation operations. We then encrypted the Brain MRI and Cameraman images with this modified technique. Table 18 compares several security indicators computed for the two images encrypted using the full encryption technique versus the partial encryption technique (which lacks the substitution component). These indicators were chosen based on their sensitivity to the absence of one component, particularly the substitution operation. It is important to note that the correlation values reported in Table 18 were chosen as follows: for the partial encryption technique, the minimum correlation across the three dimensions (H, V, D) was selected, while for the full encryption technique, the maximum correlation across the three dimensions was used.

The results show that removing the substitution component negatively impacts the security indicators, causing their values to deviate from their ideal targets to varying degrees—with the underlined values being the most significantly impacted. For instance, the entropy values for the images encrypted with the reduced technique deviate significantly more from the ideal value of 8 compared to those encrypted with the full technique. Similar conclusions can be drawn for other metrics, where their values dropped to varying degrees from their ideal values.

Table 19 presents the security indicator values when the perturbation operation is turned off. The results clearly show that some of the security indicators deviated from their ideal values to varying degrees, with the underlined values indicating the most significant deviations. Table 20 compares the randomness test results for 100 images (the same images used in Section 7.9) encrypted using the full encryption technique versus the reduced technique. The comparison is based on the number of ciphered images that failed each of the NIST tests. Overall, removing one of the encryption modules increases the number of failed randomness tests, with a notably higher failure rate when the perturbation operation is turned off. This outcome is expected, as the perturbation operation plays a crucial role in chaotically scattering the pixels of the image.

**Table 19**
Security metrics comparison before and after removing the perturbation operation.

| | Before ablation | | After ablation | |
|---|---|---|---|---|
| | Brain MRI | Cameraman | Brain MRI | Cameraman |
| Entropy | 7.9997 | 7.9993 | 7.9837 | 7.9666 |
| Correlation | 0.0007 | 0.00201 | 0.0033 | 0.0409 |
| NPCR | 99.5983 | 99.6173 | 99.01 | 96.9898 |
| UACI | 33.4619 | 33.4711 | 31.5554 | 32.8733 |
| $\chi^2$ | 243.350 | 199.708 | 302.788 | 356.223 |

**Table 20**
NIST randomness test results for 100 images: pre- and post-ablation.

| NIST randomness test | Failed (full) | Failed (lacks substitution) | Failed (lacks perturbation) |
|---|---|---|---|
| Runs | 0 | 4 | 15 |
| Monobit | 0 | 4 | 16 |
| Spectral | 5 | 13 | 33 |
| Serial | 2 | 7 | 18 |
| Cumulative sums | 2 | 7 | 16 |
| Non-overlapping template matching | 1 | 7 | 19 |
| Overlapping template matching | 6 | 4 | 13 |
| Linear complexity | 4 | 12 | 22 |
| Binary matrix rank | 4 | 11 | 22 |
| Maurer's "Universal Statistical" | 3 | 11 | 17 |
| Approximate entropy | 1 | 10 | 21 |
| Longest runs of ones | 2 | 9 | 17 |

## 8. Conclusions

In this paper, we proposed a novel image encryption technique that combines a nonlinear Fresnel Zone-based substitution operation, a data-dependent diffusion operation, and a differential neural network to address flaws in existing methods. This approach maximizes the impact of bit variations and enhances resistance against various security attacks.

The experimental results show that the proposed encryption technique significantly outperforms current state-of-the-art methods in terms of both security and efficiency. The chaotic substitution and differential neural network enhance the technique's sensitivity to bit variations and strengthen its resistance to security attacks, as indicated by high entropy, low correlations, and optimal NPCR and UACI values. Furthermore, the pixel-dependent and chaotic perturbation operation substantially increases the randomness of the encrypted images, as indicated by the NIST results in Table 15. Unlike recent techniques that rely primarily on chaotic means for pixel scrambling, the proposed technique applies a dual impact on pixel ordering, enhancing resistance to hacking tools that exploit pattern recognition and low correlations.

The reduced execution time of the proposed method indicates its practicality for real-world applications, especially in scenarios demanding high-speed and secure data transmission, such as digital communications, cloud storage, and online media sharing.

While the proposed technique demonstrates competitive security performance, we believe that incorporating additional chaotic inputs into the differential neural network could further improve the generated hiding codes. Future research will explore the use of noise chaotic signals and evaluate their impact on the overall performance of the neural network, which is fundamental component of the encryption technique.

## CRediT authorship contribution statement

**Muhammed Jassem Al-Muhammed:** Conceptualization, Testing, Drafting.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


[1] İnce C, İnce K, Hanbay D. Novel image pixel scrambling technique for efficient color image encryption in resource-constrained IoT devices. Multimedia Tools Appl 2024.
[2] Erkan U, Toktas A, Lai Q. Design of two dimensional hyperchaotic system through optimization benchmark function. Chaos Solitons Fractals 2023;167:113032.
[3] Zhou W, Wang X, Wang M, Li D. A new combination chaotic system and its application in a new Bit-level image encryption scheme. Opt Lasers Eng 2022;149:106782.
[4] Mansouri A, Wang X. A novel one-dimensional sine powered chaotic map and its application in a new image encryption scheme. Inform Sci 2020;520:46–62.
[5] Hajjaji MA, Albouchi A, Mtibaa A. Combining DWT/KLT for secure transfer of color images. In: 2019 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems. DTS, 2019, p. 1–6.
[6] Ajili S, Hajjaji MA, Mtibaa A. Hybrid SVD-DWT watermarking technique using AES algorithm for medical image safe transfer. In: 2015 16th international conference on sciences and techniques of automatic control and computer engineering. STA, 2015, p. 69–74.
[7] Man Z, Li J, Di X, Sheng Y, Liu Z. Double image encryption algorithm based on neural network and chaos. Chaos Solitons Fractals 2021;152:111318.
[8] Yao W, Liu J, Sun Y, Zhang J, Yu F, Cui L, Lin H. Dynamics analysis and image encryption application of Hopfield neural network with a novel multistable and highly tunable memristor. Nonlinear Dynam 2024;112(1):693–708.
[9] Wang S, Hong L, Jiang J. An image encryption scheme using a chaotic neural network and a network with multistable hyperchaos. Optik 2022;268:169758.
[10] Lai Q, Hu G, Erkan U, Toktas A. A novel pixel-split image encryption scheme based on 2D Salomon map. Expert Syst Appl 2023;213:118845.
[11] Lai Q, Liu Y. A cross-channel color image encryption algorithm using two-dimensional hyperchaotic map. Expert Syst Appl 2023;223:119923.
[12] Zhu L, Jiang D, Ni J, Wang X, Rong X, Ahmad M. A visually secure image encryption scheme using adaptive-thresholding sparsification compression sensing model and newly-designed memristive chaotic map. Inform Sci 2022;607:1001–22.
[13] Wang X, Wang Y. Multiple medical image encryption algorithm based on scrambling of region of interest and diffusion of odd-even interleaved points. Expert Syst Appl 2023;213:118924.
[14] Ajili S, Hajjaji MA, Mtibaa A. Crypto-watermarking algorithm using Weber's law and AES: A view to transfer safe medical image. Sci Program 2021;2021(1):5559191.
[15] Zhang Z, Tang J, Zhang F, Huang T, Lu M. Medical image encryption based on Josephus scrambling and dynamic cross-diffusion for patient privacy security. IEEE Trans Circuits Syst Video Technol 2024;1.
[16] Wang Y, Shang Y, Shao Z, Zhang Y, Coatrieux G, Ding H, Liu T. Multiple color image encryption based on cascaded quaternion gyrator transforms. Signal Process, Image Commun 2022;107:116793.
[17] Inam S, Kanwal S, Firdous R, Hajjej F. Blockchain based medical image encryption using Arnold's cat map in a cloud environment. Sci Rep 2024;14(1):5678.
[18] Zhang H, Wang X-Q, Sun Y-J, Wang X-Y. A novel method for lossless image compression and encryption based on LWT, SPIHT and cellular automata. Signal Process, Image Commun 2020;84:115829.
[19] Hajjaji MA, Bourennane E-B, Mtibaa A, Ochoa-Ruiz G. A digital watermarking algorithm based on quantization of the DCT: Application on medical imaging. In: 2013 international conference on control, decision and information technologies. CoDIT, 2013, p. 372–7.
[20] Bhowmik S, Acharyya S. Image encryption approach using improved chaotic system incorporated with differential evolution and genetic algorithm. J Inf Secur Appl 2023;72:103391.
[21] Kumar S, Sharma D. Image scrambling encryption using chaotic map and genetic algorithm: a hybrid approach for enhanced security. Nonlinear Dynam 2024;112(14):12537–64.
[22] Liang Q, Zhu C. A new one-dimensional chaotic map for image encryption scheme based on random DNA coding. Opt Laser Technol 2023;160:109033.
[23] Chai X, Fu X, Gan Z, Lu Y, Chen Y. A color image cryptosystem based on dynamic DNA encryption and chaos. Signal Process 2019;155:44–62.
[24] Zhao J, Wang S, Zhang L. Block image encryption algorithm based on novel chaos and DNA encoding. Information 2023;14(3).
[25] Amdouni R, Gafsi M, Guesmi R, Hajjaji MA, Mtibaa A, Bourennane E-B. High-performance hardware architecture of a robust block-cipher algorithm based on different chaotic maps and DNA sequence encoding. Integration 2022;87:346–63.
[26] Gao S, Wu R, Wang X, Wang J, Li Q, Wang C, Tang X. A 3D model encryption scheme based on a cascaded chaotic system. Signal Process 2023;202:108745.
[27] Yan S, Li L, Gu B, Cui Y, Wang J, Song J. Design of hyperchaotic system based on multi-scroll and its encryption algorithm in color image. Integration 2023;88:203–21.
[28] Zhang Q, Guo L, Wei X. A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Opt-Int J Light Electron Opt 2013;124(18):3596–600.
[29] Zhu Z-L, Zhang W, Wong K-W, Yu H. A chaos-based symmetric image encryption scheme using a bit-level permutation. Inform Sci 2011;181(6):1171–86.
[30] Zhang Y-Q, Wang X-Y. Analysis and improvement of a chaos-based symmetric image encryption scheme using a bit-level permutation. Nonlinear Dynam 2014;77(3):687–98.
[31] Zhang Y-Q, Wang X-Y. Spatiotemporal chaos in mixed linear–nonlinear coupled logistic map lattice. Phys A 2014;402:104–18.
[32] Wen H, Lin Y, Feng Z. Cryptanalyzing a bit-level image encryption algorithm based on chaotic maps. Eng Sci Technol Int J 2024;51:101634.
[33] Xingyuan W, Suo G, Xiaolin Y, Shuang Z, Mingxu W. A new image encryption algorithm with cantor diagonal scrambling based on the PUMCML system. Int J Bifurcation Chaos 2021;31(1):2150003–2151.
[34] Zhang T, Wang S. Image encryption scheme based on a controlled zigzag transform and bit-level encryption under the quantum walk. Front Phys 2023;10.
[35] Xingyuan W, Junjian Z, Guanghui C. An image encryption algorithm based on ZigZag transform and LL compound chaotic system. Opt Laser Technol 2019;119:105581.
[36] Wang X, Chen X. An image encryption algorithm based on dynamic row scrambling and Zigzag transformation. Chaos Solitons Fractals 2021;147(C).
[37] Li C, Tan K, Feng B, Lu J. The graph structure of the generalized discrete arnold's cat map. IEEE Trans Comput 2022;71(2):364–77.
[38] Wei D, Jiang M, Deng Y. A secure image encryption algorithm based on hyper-chaotic and bit-level permutation. Expert Syst Appl 2023;213:119074.
[39] Erkan U, Toktas A, Lai Q. 2D hyperchaotic system based on Schaffer function for image encryption. Expert Syst Appl 2023;213:119076.
[40] Wang L, Cao Y, Jahanshahi H, Wang Z, Mou J. Color image encryption algorithm based on Double layer Josephus scramble and laser chaotic system. Optik 2023;275:170590.
[41] Tan J, Liao X, Liu J, Cao Y, Jiang H. Channel attention image steganography with generative adversarial networks. IEEE Trans Netw Sci Eng 2022;9(2):888–903.
[42] Chen J, Liao X, Wang W, Qian Z, Qin Z, Wang Y. SNIS: A signal noise separation-based network for post-processed image forgery detection. IEEE Trans Circuits Syst Video Technol 2023;33(2):935–51.
[43] Kumari P, Mondal B. Lightweight image encryption algorithm using NLFSR and CBC mode. J Supercomput 2023;79(17):19452–72.
[44] Kumar S, Sharma D. A chaotic based image encryption scheme using elliptic curve cryptography and genetic algorithm. Artif Intell Rev 2024;57(4):87.
[45] Lone PN, Singh D, Mir UH. Image encryption using DNA coding and three-dimensional chaotic systems. Multimedia Tools Appl 2022;81(4):5669–93.
[46] Kocak O, Erkan U, Toktas A, Gao S. PSO-based image encryption scheme using modular integrated logistic exponential map. Expert Syst Appl 2024;237:121452.
[47] Yasser I, Khalil AT, Mohamed MA, Samra AS, Khalifa F. A robust chaos-based technique for medical image encryption. IEEE Access 2022;10:244–57.

[48] Man Z, Li J, Di X, Zhang R, Li X, Sun X. Research on cloud data encryption algorithm based on bidirectional activation neural network. Inform Sci 2023;622:629–51.
[49] Ningthoukhongjam TR, Devi Heisnam S, Singh Khumanthem M. Medical image encryption through chaotic asymmetric cryptosystem. IEEE Access 2024;12:73879–88.
[50] Razaq A, Maghrabi LA, Ahmad M, Aslam F, Feng W. Fuzzy logic-based substitution-box for robust medical image encryption in telemedicine. IEEE Access 2024;12:7584–608.
[51] Chen G, Mao Y, Chui CK. A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitons Fractals 2004;21(3):749–61.
[52] Abduljabbar ZA, Abduljaleel IQ, Ma J, Sibahee MAA, Nyangaresi VO, Honi DG, Abdulsada AI, Jiao X. Provably secure and fast color image encryption algorithm based on S-boxes and hyperchaotic map. IEEE Access 2022;10:26257–70.
[53] Chen X, Gong M, Gan Z, Lu Y, Chai X, He X. CIE-LSCP: color image encryption scheme based on the lifting scheme and cross-component permutation. Complex Intell Syst 2023;9(1):927–50.
[54] Xian Y, Wang X. Fractal sorting matrix and its application on chaotic image encryption. Inform Sci 2021;547:1154–69.
[55] Zhang T, Zhu B, Ma Y, Zhou X. A novel image encryption algorithm based on multiple random DNA coding and annealing. Electronics 2023;12(3).
[56] Haiping C, Enjian B, Xueqing J, Yun W. A fast image encryption algorithm based on improved 6-D hyper-chaotic system. IEEE Access 2022;10:116031–44.
[57] Yao W, Gao K, Zhang Z, Cui L, Zhang J. An image encryption algorithm based on a 3D chaotic Hopfield neural network and random row–column permutation. Front Phys 2023;11.
[58] Benaissi S, Chikouche N, Hamza R. A novel image encryption algorithm based on hybrid chaotic maps using a key image. Optik 2023;272:170316.
[59] Zhang Y. The unified image encryption algorithm based on chaos and cubic S-box. Inform Sci 2018;450:361–77.
[60] Ben Farah MA, Guesmi R, Kachouri A, Samet M. A novel chaos based optical image encryption using fractional Fourier transform and DNA sequence operation. Opt Laser Technol 2020;121:105777.
[61] Chen H, Bai E, Jiang X, Wu Y. A fast image encryption algorithm based on improved 6-D hyper-chaotic system. IEEE Access 2022;10:116031–44.
[62] Lu Y, Gong M, Cao L, Gan Z, Chai X, Li A. Exploiting 3D fractal cube and chaos for effective multi-image compression and encryption. J King Saud Univ - Comput Inf Sci 2023;35(3):37–58.
[63] Zhuang Z, Zhuang Z, Wang T. Medical image encryption algorithm based on a new five-dimensional multi-band multi-wing chaotic system and QR decomposition. Sci Rep 2024;14(1):402.
[64] Panigrahy AK, Maniyath SR, Sathiyanarayanan M, Dholvan M, Ramaswamy T, Hanumanthakari S, Vignesh NA, Kanithan S, Swain R. A faster and robust artificial neural network based image encryption technique with improved SSIM. IEEE Access 2024;12:10818–33.
[65] Zhou Z, Xu X, Yao Y, Jiang Z, Sun K. Novel multiple-image encryption algorithm based on a two-dimensional hyperchaotic modular model. Chaos Solitons Fractals 2023;173:113630.
[66] Nassef M, Alkinani MH, Shafik AM. A novel image cryptosystem inspired by the generation of biological protein sequences. IEEE Access 2023;11:29101–15.
[67] Man Z, Li J, Di X. Medical image encryption scheme based on self-verification matrix. IET Image Process 2021;15(12):2787–98.
[68] Zhang L, Song X, El-Latif AA, Zhao Y, Abd-El-Atty B. Reversibly selective encryption for medical images based on coupled chaotic maps and steganography. Complex Intell Syst 2024;10(2):2187–213.
[69] Juan Soto J. Randomness testing of the AES candidate algorithms. 1999.