Sudoku-Based Image Encryption with Iterative Thresholding and Strong Diffusion

Chang Chia Wei *

Posted Date: 14 April 2026

doi: 10.20944/preprints202604.0861.v1

Keywords: cryptography; Sudoku; diffusion

 doi:10.20944/preprints202604.0861.v1

Article

Sudoku-Based Image Encryption with Iterative Thresholding and Strong Diffusion

Chang Chia Wei

Chienkuo Technology University; kaojj222@gmail.com

Abstract

In the current internet era, the number of information security vulnerabilities has increased dramatically. Image and text encryption have become critical preprocessing steps in secure information transmission. Sensitive information can be transmitted through encrypted images, facilitating the implementation of various secure communication systems. This paper proposes an image encryption scheme that employs Sudoku as a cryptographic key matrix, combined with a strong diffusion mechanism to enhance pixel confusion and diffusion effects. The proposed method achieves high-level pixel scrambling through multiple rounds of iterative threshold encryption, pixel padding with random shuffling, and Sudoku-based permutation. Additionally, rotation operations are applied to further increase the irreversibility of the encrypted image. The core keys include the iterative threshold sequence, row-column diffusion keys, and random premutation parameters, ensuring that the encryption is fully reproducible. Experimental results demonstrate that, while preserving reversibility, the proposed method achieves significant confusion and diffusion performance. For the Lena image, the method attains NPCR ≈ 99.22% and UACI ≈ 33.30%, indicating its effectiveness as a robust image encryption approach.

Keywords: cryptography; Sudoku; diffusion

Introduction

Data and communication have become central to the 21st-century world, and sensitive information contained within data must be adequately protected. To safeguard data privacy, encryption plays a crucial role, as it is directly related to privacy and information security. Malicious actors often attempt to illegally obtain private information, which has led to increasing concern regarding data security and protective measures [1].

Such attackers typically aim to disclose confidential data, potentially causing significant losses to individuals or organizations. In other cases, stolen private information may be exploited for extortion or other malicious purposes [2].

Several strategies have been proposed to protect image and data privacy, including encryption and data hiding techniques such as steganography. Many researchers have developed various image encryption algorithms, each based on distinct principles and effectiveness levels. In 2020, Saidi et al. [3] investigated the sensitivity of interferometric phase images derived from interferometric synthetic aperture radar (InSAR) systems, particularly focusing on encrypted images under pixel permutation and key variation conditions.

The authors analyzed pixel sensitivity and key sensitivity using two evaluation metrics, namely the Number of Pixel Change Rate (NPCR) and the Unified Average Changing Intensity (UACI), to compare the performance of AES-256-OFB and AES-256-CTR encryption modes. These metrics are useful for assessing the robustness and reliability of encryption schemes [4].

Considering differences in underlying principles, effectiveness, operational scope, and application domains, several alternative image encryption techniques have been proposed by researchers. Currently, various standard algorithms have been developed for image and video encryption. For example, Chakraborty et al. studied encryption methods based on chaos theory and

 doi:10.20944/preprints202604.0861.v1

DNA substitution techniques [5]. Naim, M. and Ali Pacha investigated a seven-dimensional hyper-chaotic system applied to satellite image transmission security [6].

Furthermore, Bavishi et al. proposed a multimedia encryption method based on Sudoku permutations, which can be applied to the secure transmission of images, audio, and video data [7]. Mehdi and Kadhim proposed a novel five-dimensional chaos-based encryption algorithm utilizing a Sudoku matrix structure for color image encryption [8]. In their system, chaotic sequences are first generated and used to scramble the image, followed by an XOR operation with the Sudoku matrix, subsequent image permutation, and a final XOR operation between the chaotic key and the original image. In 2023, Deshpande et al. proposed a Sudoku-based image encryption algorithm that generates cryptographic keys using Sudoku structural properties and encrypts images through multiple rounds of iterative processing [9].

This paper proposes an image encryption method that utilizes Sudoku as a cryptographic key matrix, combined with a strong diffusion mechanism to enhance pixel confusion effects. Sudoku puzzles became highly popular in the United Kingdom around the end of 2004. The term 'Sudoku' originates from Japanese and means 'number position.' The concept of Sudoku is relatively simple. The solver is presented with a 9x9 grid, which is further divided into 3x3 subgrids.

In a partially filled grid, some cells are pre-filled with numbers from 1 to 9. The objective is to complete the grid such that each row, each column, and each 3x3 subgrid contains the digits 1 to 9 exactly once [10].

System Architecture

Several cryptographic algorithms, such as SHA, RSA, ECC, and AES, have been widely applied in multimedia transmission security. The proposed method in this study utilizes Sudoku-based key matrices for image encryption, combined with a strong diffusion mechanism to enhance pixel confusion effects. In this approach, Sudoku is not used as a constraint-solving problem but rather as a pre-generated valid Sudoku permutation structure to construct pixel rearrangement rules.

Therefore, the security of the system does not rely on the NP-completeness of Sudoku solving problems. Instead, it depends on the key space size and the confusion-diffusion effects introduced by iterative threshold diffusion operations. Even if an attacker is aware that a Sudoku structure is used, successful decryption still requires knowledge of the specific permutation parameters, including row-column ordering, permutation mapping, and random seed values, Otherwise, the correct pixel rearrangement sequence cannot be reconstructed.

Furthermore, Sudoku-based permutation is only applied during the confusion stage, while the overall security of the system is jointly guaranteed by multiple rounds of threshold diffusion. At present, the algorithm is experimentally validated only on image data.

The proposed algorithm adopts a multi-parameter composite key structure, including iterative threshold diffusion keys, random shuffling seeds, and Sudoku-based pixel permutation parameters. Different keys jointly determine the pixel diffusion and permutation processes, thereby constructing a large key space and improving system security. This section provides an overview of the key encryption and decryption algorithms used in image processing.

Encryption: Step 1 – Threshold Perturbation and Strong Diffusion

In the first step, to rapidly disrupt the statistical characteristics of the original image pixel values at the initial stage, this study designs a hybrid mechanism that combines threshold perturbation with multi-round strong diffusion. Let the input original image be defined as:

$$I \in \mathbb{Z}_{256}^{H\times W\times C} \tag{1}$$

Given a random threshold sequence of length T:

$$\{r_1, r_2, \ldots, r_T\}, \quad r_t \in [0,255], \tag{2}$$

 doi:10.20944/preprints202604.0861.v1

Pixel-wise additive perturbation is performed iteratively on the image. The threshold perturbation at the t-th iteration is defined as:

$$I^t(i,j,c) = (I^{t-1}(i,j,c) + r_t) \bmod 256,\ I^0 = I, \tag{3}$$

This operation applies the same random offset to all pixels in each iteration, thereby rapidly disrupting the low-frequency structures in the original image that arise from intensity distribution patterns.

Subsequently, a one-dimensional forward-backward strong diffusion (1D Forward-Backward Diffusion) process is applied. The image I^T is first flattened into a one-dimensional vector in row-major order:

$$x = \{x_0, x_1, \dots, x_{N-1}\}, N = H \times W \times C, \tag{4}$$

In the r-th diffusion round, a random key sequence of length N is introduced:

$$k^r = \{k_0^r, k_1^r, \dots, k_{N-1}^r\}, \tag{5}$$

The forward diffusion is defined as:

$$x_i' = \begin{pmatrix} x_i \oplus x_{i-1} \oplus k_i^r + \\ (x_i \cdot k_i^r \bmod 256) \end{pmatrix} \bmod 256, \tag{6}$$

Subsequently, the backward diffusion is performed:

$$x_i'' = \begin{pmatrix} x_i' \oplus x_{i+1}' \oplus k_i^r + \\ (x_i' \cdot k_i^r \bmod 256) \end{pmatrix} \bmod 256, \tag{7}$$

This forward-backward structure ensures that any slight variation in a single pixel propagates throughout the entire sequence in both directions, thereby significantly enhancing the Avalanche Effect [11].

Next, a two-dimensional row-column cross diffusion (2D Row-Column Diffusion) is performed. The diffused one-dimensional sequence is remapped into a two-dimensional image:

$$I_r \in \mathbb{Z}_{256}^{H \times W \times C}, \tag{8}$$

First, an independent random row key sequence K_{row} is introduced for each row, and intra-row diffusion is performed as follows:

$$\begin{aligned} I_r(i,j,c) = I_r(i,j,c) \oplus I_r(i,j \\ - 1,c) \oplus K_{row}(i,j,c), j \\ = 1, \dots, W-1 \end{aligned}, \tag{9}$$

Subsequently, a random column key sequence K_{col} is introduced for each column, and intra-column diffusion is carried out as:

$$\begin{aligned} I_r(i,j,c) = I_r(i,j,c) \oplus I_r(i \\ - 1,j,c) \oplus K_{col}(i,j,c), i \\ = 1, \dots, H-1 \end{aligned}, \tag{10}$$

Through the bidirectional cross diffusion along rows and columns (a modulo 256 operation is applied at the final stage in the algorithm to preserve 8-bit pixel value wrapping), the spatial dependency among pixels is further strengthened. This process extends the diffusion effect from a one-dimensional structure to a two-dimensional configuration.

The above one-dimensional and two-dimensional diffusion procedures are repeated for R rounds (in this study, R=3). In each round, independent random key sequences are employed, thereby significantly enlarging the key space and enhancing resistance against differential attacks and statistical attacks. The final diffusion result is denoted as:

$$I^r = D(I^{r-1}, K^r), \quad r = 1, \dots, R, \tag{11}$$

 doi:10.20944/preprints202604.0861.v1

$$D(\cdot,\cdot)\ is\ strong\ diffusion\ function, \tag{12}$$

$$K^r\ is\ the\ key\ sequence\ used\ in\ round\ r, \tag{13}$$

$$D(I,K) = ColDiff(RowDiff \downarrow (BackwardXOR(ForwardXOR(I,K),K),K),K), \tag{14}$$

$$I_{diffused} = I^R, \tag{15}$$

Step2: Padding and Random Permutation

In the second step, image padding and random permutation are performed to further disrupt the spatial correlation of the image and ensure that the image dimensions are compatible with subsequent Sudoku-based block permutation operations.

Specifically, the image width and height are first padded according to the Sudoku order N, such that the image dimensions become divisible by N. If the image height or width is smaller than N, it is extended to at least N. If the dimensions are not divisible by N, they are padded to the nearest multiple of N.

The padding pixels are generated using a default expansion method, which does not alter the statistical distribution characteristics of the original image.

After padding is completed, to further enhance pixel confusion, the image undergoes multiple rounds of random permutation operations. The procedure consists oftwo main steps:

· Row permutation: Each row of the image is shuffled according to an order generated by a random seed.

· Column permutation: Each column of the image is rearranged based on a permutation sequence generated from a random seed.

To ensure reproducibility of the encryption process, the permutation seeds are deterministically derived from the image dimensions and padding length, and are progressively updated across multiple iterations.

By repeating the permutation process for multiple rounds, the spatial correlation between adjacent pixels is significantly reduced, thereby increasing the unpredictability of the encrypted image.

After this step, the resulting image not only satisfies the dimensional requirements for subsequent Sudoku-based block partitioning, but also achieves a globally shuffled pixel distribution.

$$H_{pad} = \begin{cases} N, & H < N \\ \left\lceil \frac{H}{N} \right\rceil \cdot N, & H\ mod\ N \neq 0 \\ H, & HmodN = 0 \end{cases}, \tag{16}$$

$$W_{pad} = \begin{cases} N, & W < N \\ \left\lceil \frac{W}{N} \right\rceil \cdot N, & W\ mod\ N \neq 0 \\ W, & WmodN = 0 \end{cases}, \tag{17}$$

Let the original image be denoted as $I \in \mathbb{R}^{H\times W\times C}$, where H, W, and C represent the height, width, and number of channels, respectively. Let the block size be N (the Sudoku order). After padding, the image becomes $I_{pad} \in \mathbb{R}^{H_{pad}\times W_{pad}\times C}$.

For multi-round random row–column permutation, let the number of iterations be K (in this experiment, K=3). In each round, different random seeds are used to generate permutation matrices.

 doi:10.20944/preprints202604.0861.v1

Denote the random row permutation matrix in the k-th round as P_{row}^{k}, and the column permutation matrix as P_{col}^{k}.

The image after the k-th permutation round is expressed as:

$$I^{k} = P_{row}^{k} \cdot I^{k-1} \cdot P_{col}^{k}, k = 1,2, \dots, K, \quad (18)$$

$$I^{0} = I_{pad}, \quad (19)$$

$$I_{shuffled} = I^{K}, \quad (20)$$

Step3: Sudoku-Based Pixel Permutation

After completing the iterative threshold diffusion and image padding procedures, the proposed method further introduces a Sudoku-structure-based pixel permutation mechanism to disrupt the spatial correlation of the image.

Specifically, the encrypted image is first divided into multiple non-overlapping blocks of size N×N, where N corresponds to the Sudoku order (in this study, N=9). Subsequently, a pre-generated valid Sudoku permutation is used as the permutation mapping to rearrange the column indices of pixels within each block.

Let the encrypted image be denoted as III, with dimensions H×W (for grayscale images) or H×W×3 (for color images). The block size is N×N, and the Sudoku permutation mapping is defined as $perm = [p_1, p_2, \dots, p_N]$, where $p_i \in \{1,2, \dots, N\}$.

1. Block partitioning: The image is divided into multiple non-overlapping blocks.

$$\begin{matrix} B_{i,j} = I[i: i+N, j: j+N, :] \\ i = 0, N, 2N, \dots H-N \\ j = 0, N, 2N, \dots W-N \end{matrix}, \quad (21)$$

2. Intra-block column permutation: For each block Bi,jB_{i,j}Bi,j, the column indices of every row rrr are permuted according to the predefined Sudoku permutation mapping.

$$B_{i,j}(r,c) \longmapsto B_{i,j}(r,p_c), \quad r,c = 0,1, \dots, N, \quad (22)$$

3. Color image processing: For color images, the permutation is applied simultaneously across all channels to maintain inter-channel consistency.

$$B_{i,j}(r,c,k) \longmapsto B_{i,j}(r,p_c,k), k = 0,1,2, \quad (23)$$

4. Reconstruction: After completing the internal permutation within each block, all blocks are reassembled to reconstruct the image.

$$I_{shuffled}[i: i+N, j: j+N, :] = B_{i,j}, \quad (24)$$

As a simple illustrative example, consider a 6×6 image matrix (grayscale image, where pixel values are simplified as numerical representations).

$$I = [\begin{bmatrix} 1 & \cdots & 6 \\ \vdots & \ddots & \vdots \\ 31 & \cdots & 36 \end{bmatrix}, \quad (25)$$

 doi:10.20944/preprints202604.0861.v1

$$B_{0,0} = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 13 & 14 & 15 \end{bmatrix} B_{0,1} = \begin{bmatrix} 4 & 5 & 6 \\ 10 & 11 & 12 \\ 16 & 17 & 18 \end{bmatrix}$$
$$B_{1,0} = \begin{bmatrix} 19 & 20 & 21 \\ 25 & 26 & 27 \\ 31 & 32 & 33 \end{bmatrix} B_{1,1} = \begin{bmatrix} 22 & 23 & 24 \\ 28 & 29 & 30 \\ 34 & 35 & 36 \end{bmatrix}, \quad (26)$$

$$through\ perm\ = [2,0,1], \quad (27)$$

$$B_{0,0} = \begin{bmatrix} 3 & 1 & 2 \\ 9 & 7 & 8 \\ 15 & 13 & 14 \end{bmatrix} B_{0,1} = \begin{bmatrix} 6 & 4 & 5 \\ 12 & 10 & 11 \\ 18 & 16 & 17 \end{bmatrix}$$
$$B_{1,0} = \begin{bmatrix} 21 & 19 & 20 \\ 27 & 25 & 26 \\ 33 & 31 & 32 \end{bmatrix} B_{1,1} = \begin{bmatrix} 24 & 22 & 23 \\ 30 & 28 & 29 \\ 36 & 34 & 35 \end{bmatrix}, \quad (28)$$

For each block, the permutation operation is independently performed within each row, meaning that the row indices remain unchanged, while the column indices are rearranged according to the Sudoku permutation mapping. This process does not modify the pixel intensity values themselves; it only alters their spatial distribution. Therefore, it constitutes a typical permutation operation.

For color images, the permutation is synchronously applied across the RGB channels while maintaining inter-channel consistency, thereby preventing information distortion between color channels.

By introducing Sudoku-based block-level pixel permutation, this stage effectively reduces the spatial correlation between adjacent pixels and enhances the confusion property of the encrypted image. Moreover, since the permutation mapping is reversible, the original image can be accurately reconstructed during the decryption stage using the same permutation.

Step4: Image Rotation Transformation

In the final stage, a global geometric transformation is introduced. Specifically, the image processed by the previous steps is rotated clockwise by 90 degrees to alter the spatial distribution of pixels within the global coordinate system.

Let the output image from Step 3 be denoted as $I_{perm} \in \mathbb{R}^{H\times W\times C}$, where C=1 for grayscale images and C=3 for color images.

The rotation operation can be expressed by the following mapping relationship:

$$I_{rot}(x, y, c) = I_{perm}(H - 1 - y, x, c), c = 1,2, \dots, C, \quad (29)$$

Where I_{rot} denotes the encrypted image after rotation. The rotation operation does not modify the pixel intensity values; instead, it further disrupts the inherent orientation and structural characteristics of the image by altering the spatial arrangement of pixels. This enhances the overall unpredictability of the encryption result.

Decryption: Step 1 – Inverse Rotation

In the first stage of the decryption process, an inverse rotation operation is performed on the encrypted image to restore its original spatial orientation prior to encryption.

During the encryption phase, the original image was rotated clockwise by 90 degrees as part of the confusion mechanism. Therefore, in the decryption stage, an inverse operation with the same magnitude is applied, namely a 90-degree counterclockwise rotation, to ensure accurate recovery of the image structure.

$$I_1(x, y, c) = I_{rot}(y, W - 1 - x, c), c = 1,2, \dots, C, \quad (30)$$

Let I_1 denote the output image of the first decryption stage.

 doi:10.20944/preprints202604.0861.v1

Step2: Inverse Pixel Permutation Based on Sudoku Structure

In the second stage, an inverse pixel permutation operation based on the Sudoku structure is performed. This stage corresponds to the Sudoku-based pixel permutation applied during the encryption process. Its primary objective is to strictly compute the inverse of the previously applied column permutation mapping.

Let the Sudoku permutation adopted in the encryption stage be defined as:

$$p = \{p_0, p_1, \cdots, p_{N-1}\}, \tag{31}$$

where N represents the Sudoku order (in this experiment, N=9). The inverse permutation is then constructed based on the original permutation mapping to ensure that pixels can be accurately mapped back to their original index positions during the decryption process.

$$p^{-1} = argsort(p), \tag{32}$$

Subsequently, the image is partitioned into multiple non-overlapping blocks of size N×N. For each block $B_{i,j}$, the inverse permutation operation is independently performed within each row, keeping the row indices unchanged, and restoring the original pixel positions according to the inverse permutation p^{-1}:

$$B_{i,j}^{rec}(r,c) = B_{i,j}^{enc}(r, p_c^{-1}), \qquad r,c = 0,1,\ldots,N-1, \tag{33}$$

For color images, the inverse permutation process is simultaneously applied across the three RGB channels.

Step3: Inverse Padding and Inverse Row–Column Permutation

During the second encryption stage, image padding was performed to ensure that the image dimensions are divisible by the Sudoku block size, followed by multiple rounds of random row–column permutation. Therefore, during the decryption phase, it is necessary to restore the original image dimensions and reverse the row–column permutation operations.

First, for the multi-round random permutation, the same random seed used in the encryption stage is employed to regenerate the identical row and column permutation matrices. For each iteration round, the inverse row permutation matrix is applied first to restore the row order of the image, followed by the inverse column permutation matrix, thereby gradually recovering the original pixel arrangement:

$$I^{k-1} = P_{row}^{k^{-1}} \cdot I^k \cdot P_{col}^{k^{-1}}, \qquad k = K, K-1, \ldots, 1, \tag{34}$$

Let I^k denote the image after the k-th permutation round, where $P_{row}^{k^{-1}}$ and $P_{col}^{k^{-1}}$ represent the inverse row and inverse column permutation matrices, respectively, and KKK denotes the number of iterations.

After completing the inverse permutation process, the padded regions introduced during the encryption stage are removed by cropping the image, restoring the image dimensions to their original size H×W. Through this operation, the image is restored to the state prior to the second encryption step, thereby providing an accurate input for subsequent threshold-based decryption.

Step4 : Iterative Threshold and Strong Diffusion Decryption

In the final decryption stage, iterative threshold-based encryption applied during the encryption process is reversed to recover the original grayscale or RGB pixel values. The decryption procedure maintains strict symmetry with the encryption process and primarily consists of two components: strong inverse diffusion and iterative random threshold inverse operations.

First, the strong diffusion inverse function is applied to process the image. This function performs round-by-round reverse operations using the key sequence K^r, the forward and backward

 doi:10.20944/preprints202604.0861.v1

nonlinear product terms $\mathrm{F}^{\mathrm{r}}, B^{r}$, and the row–column diffusion keys $\mathrm{K}_{\mathrm{row}}^{\mathrm{r}}, K_{col}^{r}$, all of which are stored from the encryption stage.

Let the image be denoted as:

$$I_{diffused} \in \mathbb{Z}_{256}^{H\times W\times C}, \tag{35}$$

The strong diffusion operation in the encryption stage is performed over R iterative rounds. In each round, a random key sequence K^{r}, forward and backward nonlinear product terms $\mathrm{F}^{\mathrm{r}}, B^{r}$, and row–column diffusion keys $\mathrm{K}_{\mathrm{row}}^{\mathrm{r}}, K_{col}^{r}$, are employed.

During the decryption stage, the inverse operations are executed in the reverse order of the encryption procedure. First, inverse column diffusion is performed:

$$I_r(i,j,c) = I_r(i,j,c)\oplus K_{col}^{r}(i,j,c)\oplus I_r(i-1,j,c), i = H-1,\dots,1, \tag{36}$$

Next, inverse row diffusion is applied:

$$I_r(i,j,c) = I_r(i,j,c)\oplus K_{row}^{r}(i,j,c)\oplus I_r(i,j-1,c), j = W-1,\dots,1, \tag{37}$$

Then, the inverse backward nonlinear XOR operation is carried out. Using the backward nonlinear product term B^{r}, and the key sequence K^{r}, stored during the encryption stage, the nonlinear XOR operation is reversed element-wise to restore the original pixel values:

$$x_i' = \left((x_i'' - B_i^{r})\ mod\ 256\right)\oplus x_{i+1}'\oplus k_i^{r}, \quad i = 0,1,\dots,N-2, \tag{38}$$

Subsequently, inverse forward nonlinear XOR operation is performed by combining the forward nonlinear product term F^{r} and the key sequence K^{r}:

$$x_i = \left((x_i' - F_i^{r})\ mod\ 256\right)\oplus x_{i-1}\oplus k_i^{r}, \quad i = N-1,\dots,1, \tag{39}$$

Finally, the inverse threshold operation is applied by sequentially subtracting the random threshold values $\mathbf{r_t}$ used in each iteration:

$$I^{t-1}(i,j,c) = (I^{t}(i,j,c) + r_t)\ mod\ 256, \quad t = T,\dots,1, \tag{40}$$

The entire decryption process can be abstracted as an inverse strong diffusion function:

$$I^{r-1} = D^{-1}(I^{r}, K^{r}), \qquad r = R, R-1,\dots,1, \tag{41}$$

Where $D^{-1}(\cdot,\cdot)$, denotes the inverse operation of the strong diffusion function. The internal processing order follows: inverse column diffusion → inverse row diffusion → inverse backward XOR → inverse forward XOR. After completing the decryption procedure, the original image is successfully reconstructed:

$$I_{restored} = I^{0}, \tag{42}$$

Analysis

To evaluate the computational efficiency of the proposed Sudoku-based iterative threshold image encryption method, experiments were conducted on a 512×512 color image (Lena). All experiments were performed on a desktop computer equipped with an Intel Core i7-13700 processor, 16 GB RAM, and Intel UHD Graphics 770 integrated graphics. No dedicated GPU acceleration was used.

The experimental results show the time consumption distribution of each processing step. In the encryption stage, iterative threshold encryption (Step 1) constitutes the primary computational bottleneck, requiring approximately 7.03 seconds. The subsequent image padding and shuffling (Step

 doi:10.20944/preprints202604.0861.v1

2), Sudoku-based pixel permutation (Step 3), and image rotation (Step 4) require approximately 0.048 s, 0.063 s, and 0.026 s, respectively. The total encryption time is approximately 7.19 seconds.

In the decryption stage, iterative threshold inverse decryption (Step 4) also requires the longest processing time, approximately 6.47 seconds, while inverse rotation (Step 1), pixel unshuffling (Step 2), and inverse padding removal (Step 3) require approximately 0.027 s, 0.097 s, and 0.045 s, respectively. The total decryption time is approximately 6.65 seconds.

The NPCR and UACI metrics used in this study are consistent with the implementation code. When input image dimensions are inconsistent, the images are first flattened and truncated to a common length before computation to ensure a consistent comparison basis.

First, the difference matrix $D(i,j,c)$ is defined as:

$$D(i,j,c) = \begin{cases} 1, & if\ I_i(i,j,c) \neq I_2(i,j,c) \\ 0, & otherwise \end{cases} \quad (43)$$

The Number of Pixel Change Rate (NPCR) is given by:

$$NPCR = \frac{\sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \sum_{c=0}^{C} D(i,j,c)}{H \times W \times C} \times 100\% \quad (44)$$

NPCR is used to measure the proportion of changed pixels in the encrypted image when a slight modification is applied to the original image.

The Unified Average Changing Intensity (UACI) is given by:

$$UACI = \frac{1}{H \times W \times C} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \sum_{c=0}^{C} \frac{|I_1(i,j,c) - I_2(i,j,c)|}{255} \times 100\% \quad (45)$$

The value 255 represents the maximum pixel intensity value for an 8-bit image. UACI is used to evaluate the average intensity difference between the original image and the encrypted image.

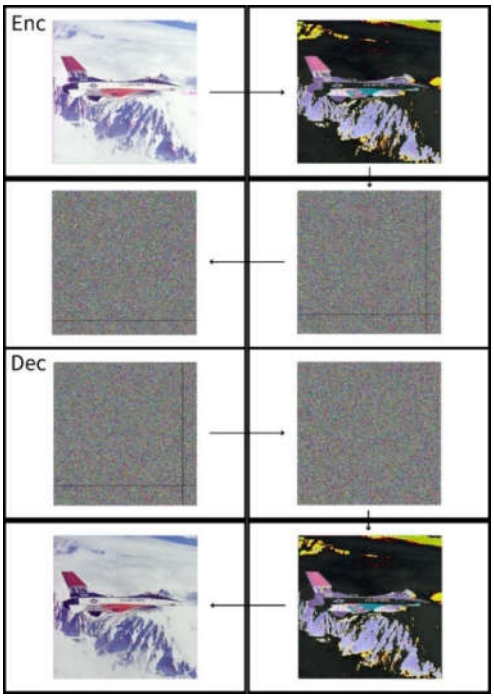
Table 1 presents the NPCR and UACI evaluation results for five encrypted images processed by the proposed Sudoku-based pixel permutation and strong diffusion algorithm. The table lists the image name, image type (color or grayscale), Sudoku block size used, and the corresponding NPCR and UACI values.

Table 1. NPCR and UACI Analysis Results under the 9×9 Sudoku Structure.

Image Name	Image Type	Sudoku block size	NPCR (%)	UACI (%)
Lena	Color	9 × 9	99.2239	33.2896
Airplane (F-16)	Grayscale	9 × 9	99.2333	33.3401
Sailboat	Color	9 × 9	99.2269	33.3458
Peppers	Color	9 × 9	99.2175	33.3203
Tank	Grayscale	9 × 9	99.2313	33.3008

As shown in Figure 1, the encryption and decryption processes were demonstrated using an Airplane F-16 image. The upper half of the figure illustrates the encryption procedure, while the lower half shows the decryption procedure. The image size is 512×512.

 doi:10.20944/preprints202604.0861.v1



Discussion

The experimental results demonstrate that the proposed method consists of several sequential steps, including iterative threshold encryption, image padding and shuffling, Sudoku-based permutation, and rotation transformation.

Through multi-round threshold encryption and strong diffusion operations, the algorithm effectively disrupts the spatial distribution of image pixels. Consequently, small variations in a single pixel can propagate globally across the encrypted image, ensuring high NPCR and UACI values and providing strong resistance against differential attacks.

In terms of parameter configuration, this study adopts a 9×9 Sudoku structure and three rounds of iterative threshold encryption, achieving relatively high NPCR (≈ 99.22%) and UACI (≈ 33.3%) values, which satisfy common cryptographic security requirements.

While maintaining image security, the overall computational efficiency remains acceptable. Iterative threshold encryption and decryption constitute the primary performance bottleneck, and further optimization can be considered to improve processing speed. The performance analysis provides a reference for the feasibility of applying the proposed method to practical image encryption scenarios.

References

1. Mhatre, M., Kashid, H., Jain, T., & Chavan, P. (2022). BCPIS: Blockchain-based counterfeit product identification system. Journal of Applied Security Research, 1–26.
2. Mehta, K., Dhingra, G., & Mangrulkar, R. (2022). Enhancing multimedia security using shortest weight first algorithm and symmetric cryptography. Journal of Applied Security Research, 1–24.
3. Saidi, R., Cherrid, N., Bentahar, T., Mayache, H., & Bentahar, A. (2020). Number of pixel change rate and unified average changing intensity for sensitivity analysis of encrypted inSAR interferogram. Ingénierie Des Systèmes D'information/Ingénierie Des Systèmes D'Information, 25(5), 601–607.
4. Y. Wu, J. Noonan and S. Agaian, "Npcr and Uaci Randomness Tests for Image Encryption," Cyber Journals: Multidisciplinary, Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), 2011, pp. 31-38.

 doi:10.20944/preprints202604.0861.v1

5. Chakraborty, S., Seal, A., Roy, M., & Mali, K. (2016). A novel lossless image encryption method using DNA substitution and chaotic logistic map. International Journal of Security and Its Applications, 10(2), 205–216.
6. Naim, M., & Ali Pacha, A. (2021). New chaotic satellite image encryption by using some or all the rounds of the AES algorithm. Information Security Journal: A Global Perspective, 32(3), 187–211.
7. Mohammad, O. F., Rahim, M. S. M., Zeebaree, S. R. M., & Ahmed, F. (2017). A survey and analysis of the image encryption methods. International Journal of Applied Engineering Research, 12(23), 13265–13280.
8. Mehdi, S. A., & Kadhim, A. A. (2019). Image encryption algorithm based on a new five dimensional Hyperchaotic system and Sudoku matrix. *2019 International Engineering Conference (IEC), IEEE*, 188–193.
9. Deshpande, K., Girkar, J., & Mangrulkar, R. (2023). Security enhancement and analysis of images using a novel Sudoku-based encryption algorithm. Journal of Information and Telecommunication, 7(3), 270–303
10. Felgenhauer, B., & Jarvis, F. (2006). Mathematics of Sudoku I. Mathematical Spectrum, 39(1), 15–22.
11. Menezes, van Oorschot, Vanstone (1996): Handbook of Applied Cryptography, Chapter 7.