**SN Computer Science manuscript No.**
(will be inserted by the editor)

# An Image Encryption Algorithm Based on Trivium Cipher and Random Substitution

**Yousef Alghamdi**[a,1], **Arslan Munir**[b,1]

[1]Department of Computer Science, Kansas State University, Manhattan, Kansas, 66506 USA

Received: date / Accepted: date

**Abstract** Traditional encryption algorithms are not suitable and computationally efficient for encrypting multimedia data due to the large size and high redundancy inherent in multimedia data. In this paper, a new image encryption algorithm based on nonlinear-feedback shift registers is proposed. The proposed algorithm is based on the Trivium cipher and has multiple encryption rounds. A key schedule produces the round keys from the initial secret key, and the Trivium cipher generates the key-streams for the bit-level substitution for each round utilizing the round key and an initialization vector (IV). Each round of the proposed algorithm consists of three steps, pixel-based row permutation, pixel-based column permutation, and bit-level substitution. Experimental results show that the proposed algorithm is reliably secure and outperforms the contemporary image encryption algorithms in terms of quality, efficiency, and security on most of the image encryption metrics. Furthermore, the low complexity of the proposed Trivium-based image encryption algorithm demonstrates high potential for deployment in real-time applications.

**Keywords** Image Encryption, Trivium Cipher, Permutation, Substitution

## 1 Introduction

Over the past few decades, image encryption algorithms have been the focus of researchers in the information security field. With the current increase in multimedia data sharing, such as images and videos, over the communication networks, a need for protecting the security and privacy of such data with real-time encryption arises. Multimedia data usually contains private and personal information, which needs to be protected from unauthorized access, and transmitted securely. While the well-established and secure encryption methods, such as Advanced Encryption Standard (AES) [1] and Data Encryption Standard (DES) [2] are recommended and widely used, they are not suitable for real-time multimedia data encryption because of their high computational requirements and slow encryption speed. Thus, specialized algorithms for multimedia data encryption are needed. Digital images usually have high redundancy due to the correlation between neighboring pixels. The high redundancy refers to the fact that adjacent pixels in an image tend to have the same values, that is, the images which contain repeating colors and pattern will have the same pixel value repeated multiple times within an image. To solve the correlation and encryption speed problems associated with digital images, researchers have proposed various encryption algorithms specifically designed for image encryption. The proposed algorithms are usually based on chaos theory [3–8], DNA encoding [9–13], neural networks [14, 15], compressive sensing [16–18], etc.

In this work, we propose a multi-round image encryption algorithm based on the Trivium cipher [19]. In our proposed image encryption algorithm, the Trivium cipher is used to produce pseudorandom key-streams for pixel permutation and bit-level substitution in each round. The algorithm utilizes an initial key and a key schedule to generate the round keys. The Trivium cipher utilizes the round keys and an initialization vector (IV) to generate the round key-streams. Our main contributions in this article are as follows:

- Developing a new Trivium based image encryption algorithm that has multiple encryption rounds and generates the key-streams for pixel-based row permutation, pixel-based column permutation, and bit-level substitution.

[a]e-mail: usef@ksu.edu
[b]e-mail: amunir@ksu.edu

- Security analysis of the proposed image encryption algorithm.
- Evaluating the proposed image encryption algorithm with a comprehensive set of evaluation metrics, and comparing the evaluation results of the proposed algorithm with the contemporary image encryption algorithms.

The rest of the paper is organized as follows. Section 2 discusses image encryption algorithms in the literature. Section 3 presents preliminary information related to Trivium cipher. Section 4 details the proposed image encryption algorithm. Security analysis of the proposed image encryption algorithm is presented in Section 5. Section 6 discusses the experimental evaluation results. Finally, Section 7 concludes this work.

## 2 Related Work

Image encryption has gained a significant interest in recent years due to the increase in multimedia data sharing in recent times. A variety of image encryption algorithms have been presented in the literature. Talhaoui et al. [3] proposed a novel real-time image encryption algorithm that uses a new simple one-dimensional chaotic map, and combines the substitution and permutation of an image's pixels to modify the pixels' positions and values simultaneously. In [4], Gao introduces an image encryption algorithm based on a 2D hyperchaotic map by using two 1D-chaotic maps, a linear function, and a multiplier. The proposed algorithm encrypts images by performing row and then column shifts. Next, the pixel values are obscured through forward and backward diffusion to produce the cipher image.

Wu et al. [9] have proposed an image encryption algorithm based on a hyperchaotic map on top of DNA coding. In the proposed algorithm, a key-stream is generated using a hyperchaotic map to scramble the plain image. Then, the scrambled image and the key-stream are encoded by the DNA. The proposed algorithm then diffuses the resulting DNA-encoded scrambled image with the DNA-encoded key-stream, and proceeds to DNA decode the resulting diffused image which outputs the encrypted cipher image. In Zhang et al. [10], the authors have proposed an image encryption integrating hyperchaotic map, DNA encoding, Arnold transform, and phase-truncated fractional Fourier transform (ptFrFT). The proposed algorithm uses the plaintext image to generate a SHA-256 hash which is used to produce the control parameters and initial values for the hyperchaotic map, DNA encoding, Arnold transform, and ptFrFT. The plaintext image is scrambled by Arnold's transform and then encoded into a noise-like image using ptFrFT with two random phase masks using keys generated by the hyperchaotic map. Finally, the cipher image is produced by DNA-level diffusion and scrambling on the masked noise-like image. Algorithms based on DNA coding yield secure encrypted images, but require the transmission of additional tables to decrypt the image [11].

Wang and Li [14] have proposed an image encryption algorithm based on Hopfield chaotic neural network. The proposed algorithm utilizes a staged composite chaotic map based on a logistic map and a tent map to generate initial parameters for Arnold map and Hopfield neural network. The plaintext image is first scrambled through Arnold map, then a self-diffusion chaotic key-stream matrix is generated using Hopfield neural network, and the scrambled image is XORed with the key-stream to generate the cipher image. In [15], Man et al. have proposed a double image encryption algorithm based on a convolutional neural network and dynamic adaptive diffusion. The proposed algorithm uses a logistic map to control the initial values of a 5D conservative chaotic system.

Recently, compressive sensing-based image encryption algorithms have attracted a lot of attention. Compressive sensing can help reduce the size of encrypted images. Dou and Li [16] have proposed a new image encryption algorithm based on compressive sensing, M sequence, and an improved 1D chaotic map. The proposed algorithm uses the plaintext image to generate a SHA-512 hash which is used to produce the control parameters of the improved 1D chaotic map and the linear-feedback shift registers (LFSR). The plain image is then transformed into a discrete wavelet transform (DWT) and scrambled using the LFSR. Then, the improved 1D chaotic system is used to generate the measurement matrix, which the compressive sensing uses to compress the DWT image and produce the cipher image. In [17] Zhang et al. have proposed an image compression and encryption algorithm based on compressive sensing and Fourier transform. In the proposed algorithm, tent-sine chaotic map is used to generate the measurement matrix using a SHA-256 hash of the plain image. Arnold transform is used to scramble the compressed image, and the 2D Fourier transform produced by Chen hyperchaotic map is used as a mask to produce the cipher image. Ping et al. [18] have proposed an algorithm combining compressive sensing and steganography to generate a visually meaningful encrypted image. In the proposed algorithm, a secret key is used to generate the initial parameters for 2D logistic-adjusted-Sine map (2D-LASM), and 3D cat map. The plain image is then sparsified into DWT and permuted using the 2D-LASM chaotic sequence. The measurement matrix for compressive sensing is constructed by the 3D cat map and is used to compress and encrypt the permuted image. Lastly, the cipher image is embedded into a carrier image, and the order of the embedding position is encrypted by the 2D-LASM chaotic sequence.

## 3 Preliminaries

This section provides a brief overview of nonlinear-feedback shift registers (NLFSRs) and Trivium cipher.

### 3.1 Nonlinear-Feedback Shift Registers (NLFSR)

NLFSRs are feedback shift registers whose input bits are a non-linear function of their previous states. Feedback shift registers consists of $n$ binary clocked storage elements, which are called stages or bits. Each stage has a state value of 0 or 1, and a feedback function determines how the stage state value is updated. At each cycle of the feedback shift register, the register's stage values are shifted by 1-bit to the right, and the value of stage 1 is updated via the feedback function. A feedback function is a mathematical function that takes the values of two or more specific stages and performs bitwise operation(s) to produce a new bit that is then fed back to the shift register. Appropriately selected stages and bitwise operations can ensure better randomness and longer cycle of generated key-streams.

### 3.2 Trivium Cipher

Trivium is a synchronous stream cipher that is based on an NLFSR which generates a key-stream of up to $2^{64}$ bits using an 80-bit secret key and an 80-bit IV [19]. Flowchart of the Trivium cipher is depicted in Figure 1. Trivium cipher has three phases: initialization phase, warm-up phase, and encryption phase. In the *initialization* phase, registers of the Trivium cipher are loaded with an 80-bit key and an 80-bit IV to the 288-bit initial state denoted as ($S_1$, ..., $S_{287}$, $S_{288}$) and setting all remaining bits to 0, except for $S_{286}$, $S_{287}$, and $S_{288}$, which are set to 1. In the *warm-up* phase, the cipher is clocked $4 \times 288 = 1152$ times, and no cipher output (key-stream) is generated/used. The bits produced hereafter, that is, starting with the output bit of cycle 1153, form the key-stream, and this phase can be referred to as the *encryption* phase. The warm-up phase is required for randomizing the cipher sufficiently, and to make sure that the key-stream depends on both the secret key and the IV . Algorithm 1 depicts the initialization and warm-up phase of the Trivium cipher.

In the key-stream generation or *encryption* phase, the Trivium cipher's state iteratively extracts values of 15 specific state bits which are used to update 3-bits of the state and produce 1-bit of the key-stream $z_i$. The Trivium cipher is then continuously clocked and at each clock cycle, one key-stream bit is generated. Algorithm 2 depicts the key-stream generation process of the Trivium cipher.

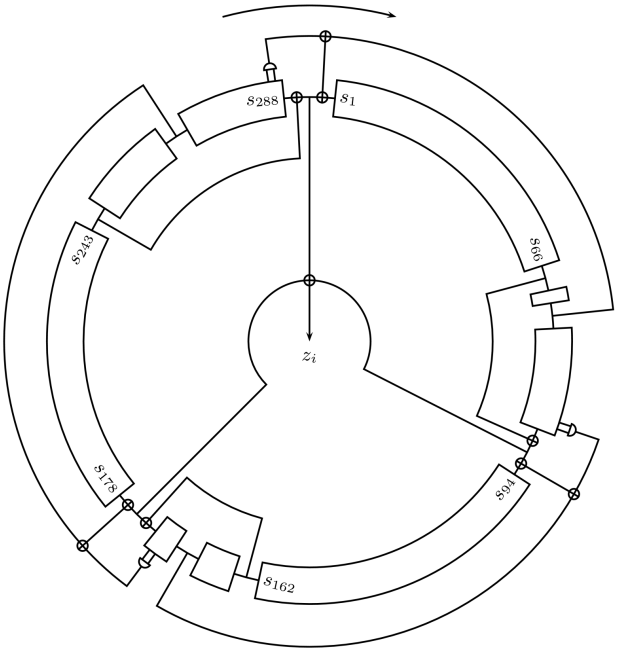



Fig. 1: Trivium cipher flowchart [19].

**Algorithm 1:** Trivium Initialization and Warm-up Phase [19]

$(S_1, S_1, ..., S_{93}) \leftarrow (K_1, ..., K_{80}, 0, ..., 0)$;
$(S_{94}, S_{95}, ..., S_{177}) \leftarrow (IV_1, ..., IV_{80}, 0, ..., 0)$;
$(S_{178}, S_{179}, ..., S_{288}) \leftarrow (0, ..., 0, 1, 1, 1)$;
**for** $i = 1$ ***to*** $4 \times 288$ **do**
    $t_1 \leftarrow S_{66} + S_{91} \cdot S_{92} + S_{93} + S_{171}$;
    $t_2 \leftarrow S_{162} + S_{175} \cdot S_{176} + S_{177} + S_{264}$;
    $t_3 \leftarrow S_{243} + S_{286} \cdot S_{287} + S_{288} + S_{69}$;
    $(S_1, S_1, ..., S_{93}) \leftarrow (t_1, S_1 ..., S_{92})$;
    $(S_{94}, S_{95}, ..., S_{177}) \leftarrow (t_2, S_{94} ..., S_{176})$;
    $(S_{178}, S_{179}, ..., S_{288}) \leftarrow (t_3, S_{178} ..., S_{287})$;
**end**

## 4 Proposed Image Encryption Algorithm

The flowchart of the proposed Trivium-based image encryption algorithm is illustrated in Figure 2. The proposed algorithm utilizes a key schedule to generate the 80-bit round keys from the initial secret key $k$. The 80-bit round keys are then used with the 80-bit IV as the initial parameters for the Trivium cipher to generate the rounds key-streams. Each round of the proposed algorithm consists of three steps: pixel-based row permutation, pixel-based column permutation, and bit-level substitution based on bitwise XOR operation to obscure the plaintext image values and reduce the correlation between the image pixels. The rounds key-streams are used to generate Matrix$_i$ ($M_i$) which is used for the pixel-based row and column random permutations. Each 16-bits of the binary round key-stream are converted to a decimal value, and then the `mod` of the converted decimal

**Algorithm 2:** Trivium Key-Stream Generation Phase [19]

1 **for** $i = 1$ *to* $N$ **do**
2 $t_1 \leftarrow S_{66} + S_{93}$;
3 $t_2 \leftarrow S_{162} + S_{177}$;
4 $t_3 \leftarrow S_{243} + S_{288}$;
5 $z_i \leftarrow t_1 + t_2 + t_3$;
6 $t_1 \leftarrow t_1 + S_{91} \cdot S_{92} + S_{171}$;
7 $t_2 \leftarrow t_2 + S_{175} \cdot S_{176} + S_{264}$;
8 $t_3 \leftarrow t_2 + S_{286} \cdot S_{287} + S_{69}$;
9 $(S_1, S_1, ..., S_{93}) \leftarrow (t_1, S_1 ..., S_{92})$;
10 $(S_{94}, S_{95}, ..., S_{177}) \leftarrow (t_2, S_{94} ..., S_{176})$;
11 $(S_{178}, S_{179}, ..., S_{288}) \leftarrow (t_3, S_{178} ..., S_{287})$;
12 **end**

value is taken with the height of the image $H$ to be suitable for use in performing the random permutations.

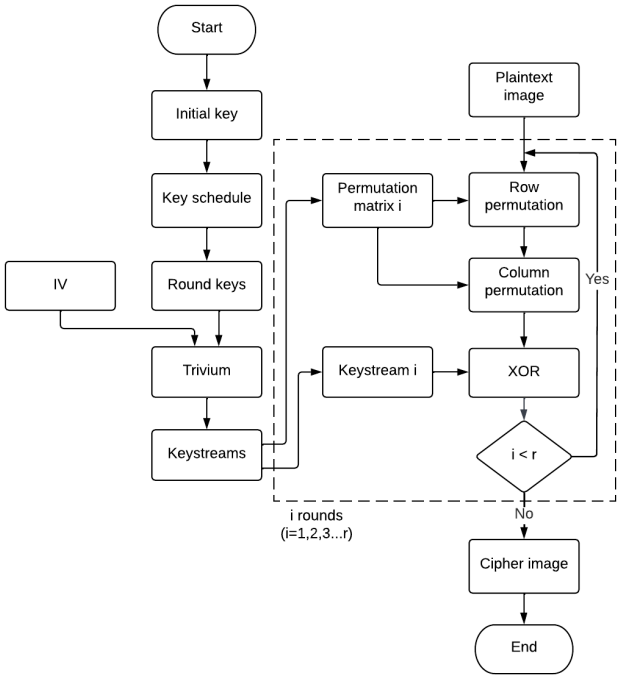



Fig. 2: Proposed image encryption algorithm's flowchart.

To illustrate the generation of the matrices $M_i$, let us assume that the image height $H$ is 256, and the key-stream value is {01011010001000000001010001...} (line 3 in Algorithm 3). The first iteration of the matrix generation uses the first 16-bits of the key-stream to calculate the first value of $M_i$ (line 5 in Algorithm 3) as uint16(0101101000100000) mod 256 = 23072 mod 256 = 32. In the second iteration, the next 16-bits of the key-stream is used to calculate the second value of $M_i$ as uint16(0010100011101001) mod 256 = 10473 mod 256 = 233. Similarly, all the other values in $M_i$ are calculated following the same steps (Table 1).

Table 1: An example of the generation of Matrix$_i$ ($M_i$) from the round key-stream.

| | | | | |
|---|---|---|---|---|
| 32 | 233 | 19 | 99 | 84 |
| 176 | 107 | 98 | 82 | 38 |
| 186 | 53 | 66 | 135 | 200 |
| 117 | 209 | 6 | 142 | 122 |
| 239 | 100 | 54 | 30 | 160 |

The steps of the proposed algorithm to encrypt an image are as follows:

1. Read initial secret key $k$.
2. Generate round keys via the key schedule using secret key $k$ (line 2 in Algorithm 3).
3. Generate the IV random data block.
4. Generate key-stream$_i$ from Trivium using the round key$_i$ and the IV (line 3 in Algorithm 3).
5. Use key-stream$_i$ to generate permutation matrix$_i$ ($M_i$) (lines 4-6 in Algorithm 3).
6. Read plaintext image $P$.
7. Use the matrix $M_i$ to perform random row permutation on the plain image (line 9 in Algorithm 3).
8. Use the matrix $M_i$ to perform random column permutation on the resulting image from Step 7 (line 10 in Algorithm 3).
9. Perform bitwise XOR operation between key-stream$_i$ and the permuted image resulting from Step 8 (line 11 in Algorithm 3).
10. Repeat Steps 7, 8, and 9 until $i = r$ ($r$ is the total number of rounds).
11. Produce the cipher image $C$ (line 13 in Algorithm 3).

The steps of the decryption process for the proposed algorithm are outlined below:

1. Read the cipher image $C$ along with the initial key $k$ and the IV.
2. Perform Steps 2 to Steps 5 of the encryption process.
3. Perform bitwise XOR operation between the key-stream$_i$ and the cipher image $C$.
4. Use matrix $M_i$ to perform random column permutation on the resulting image from Step 3.
5. Use matrix $M_i$ to perform random row permutation on the resulting image from Step 4.
6. Repeat Steps 3, 4, and 5 until $i = r$.
7. Produce the decrypted image $P$.

**Algorithm 3:** Proposed Trivium-based Image Encryption

**Input:** Plaintext image $P$, initial key $k$, initialization vector $IV$, number of rounds $r$
**Output:** Cipher image $C$

$[H,W,nc] \leftarrow size(P)$ ; //Image height $H$, width $W$, and number of channels $nc$
$roundKey_{1,2,\ldots,r} \leftarrow keySchedule(k)$ ; //Generate round keys via the key schedule using initial key $k$
$keyStream_{1,2,\ldots,r} \leftarrow Trivium(roundKey_{1,2,\ldots,r}, IV, r)$ ; //Generate key-streams via Trivium using $roundKey_{1,2,\ldots,r}$, $IV$, $r$
**for** $j \leftarrow 1$ ***to*** $H$, $k \leftarrow 1$ ***to*** $W$ **do**
  $M_{1,2\ldots r}(j,k) \leftarrow keyStream_{1,2,\ldots,r}(j : j+15) \bmod H$ ; //$M_i$ denotes Matrix i; $H$ denotes image height
**end**
$Q \leftarrow P$ ; //$Q$ denotes the current state of the image
**for** $i \leftarrow 1$ ***to*** $r$ **do**
  $Q \leftarrow rowPermutation(Q, M_i)$;
  $Q \leftarrow colPermutation(Q, M_i)$;
  $Q \leftarrow Q \oplus keyStream_i$;
**end**
$C \leftarrow Q$;
**return:** $C$

## 5 Security of The Proposed Algorithm

This section analyzes security of the proposed algorithm. The section discusses the security of each of the key aspects of the proposed algorithm, viz., Key space, the proposed key schedule and rounds, the row and column permutations, and the bitwise XOR of the image. After analyzing the security of each of the key aspects of the proposed algorithm, the overall security of the proposed algorithm is estimated.

### 5.1 Key Space

An encryption algorithm should have a large key space to be able to provide sufficient security against brute-force attacks. A key space is the set of all possible keys that could be used by an encryption algorithm. Since in the proposed algorithm, round keys are generated using an initial key $k$ of $n = 80$ bits, the key space for the proposed algorithm is $2^n = 2^{80}$, which is large enough to withstand brute-force attacks. The initial key $k$ of $n$ bits impart a security level of $n$ bits, where $n = 80$ for the Trivium-based image encryption algorithm.

### 5.2 Rounds and Key Schedule

The proposed algorithm has multiple rounds of encryption, where in each round a new key-stream is generated using the Trivium cipher with an 80-bit round key produced by the key schedule. By using multiple rounds $r$, the algorithms security increases by $r$ times. The key schedule is designed based on a modified AES key schedule. The key schedule uses the initial key $k$ of length 80-bit to produce the round keys used in the Trivium cipher to generate the key-streams for each round. The number of round keys generated by the key schedule is equal to the number of rounds $r$ in the algorithm. The key schedule leads each round to act differently and prevent slide attacks.

### 5.3 Pixel-Based Permutations

The row and column permutations add a level of security to the encryption by reducing the correlation between the pixels and obscuring the original image by adding randomness. The permutation is made by changing the location of the pixels based on the permutation matrix which is randomly generated by the key-stream produced by the Trivium cipher. In the proposed algorithm, there are $(H \times W)!$ possible permutations for a given image with height $H$ and width $W$ [20]. A security level of $m$ bits is imparted by the row and column permutations of the proposed algorithm if $2^m = (H \times w)!$, which implies that $m = log_2[(H \times W)!]$. Thus, the row and column permutations of the proposed algorithm impart a security level of $m = log_2[(H \times W)!]$ bits.

### 5.4 Bitwise XOR of Image Bits

The last security step of the proposed algorithm is to take a bitwise XOR of the permuted image with the key-stream bits generated by the Trivium cipher. Bitwise XORing an image of size $H \times W$ assuming each pixel is $l$ bits adds a security level of $(H \times W \times l)$ to the proposed image encryption.

### 5.5 Overall Security Level of the Proposed Image Encryption Algorithm

Overall security level of the proposed image encryption algorithm is estimated by summing the security level imparted by each key aspect of the proposed algorithm.

Thus, the overall security level $\mathscr{L}$ of the proposed image encryption algorithm is given as:

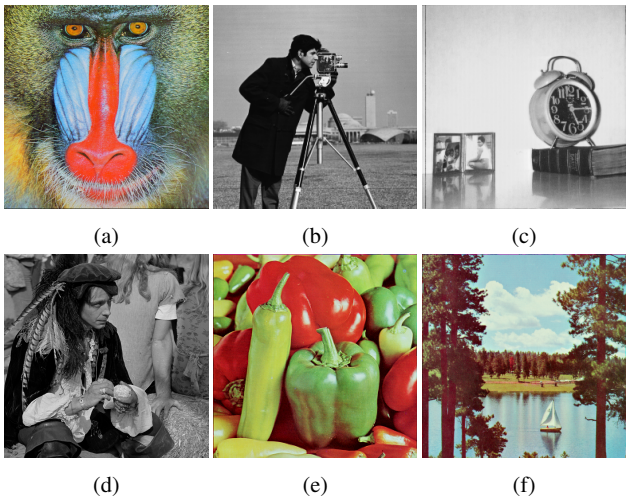
$$\mathscr{L} = r \times \left[ n + log_2 \{ (H \times W)! \} + (H \times W \times l) \right] \text{ bits}, \quad (1)$$

where $r$ is the number of rounds, $n = 80$ is the initial key length for Trivium cipher, $l$ denotes the number of bits used to represent a pixel, and $H$ and $W$ are the height and width of the image, respectively.

## 6 Results and Analysis

We have evaluated the proposed algorithm using images from the USC-SIPI image dataset [21]. Fig. 3 depicts the images used, viz., the images of Baboon, Cameraman, Clock, Male Pepper, and Sailboat. The proposed algorithm has been assessed using both 8-bit grayscale images and 24-bit color images with variable resolutions ranging from $128 \times 128$, $256 \times 256$, $512 \times 512$, to $1024 \times 1024$ pixels. Fig. 4 shows a sample set of plaintext images and their corresponding cipher images.

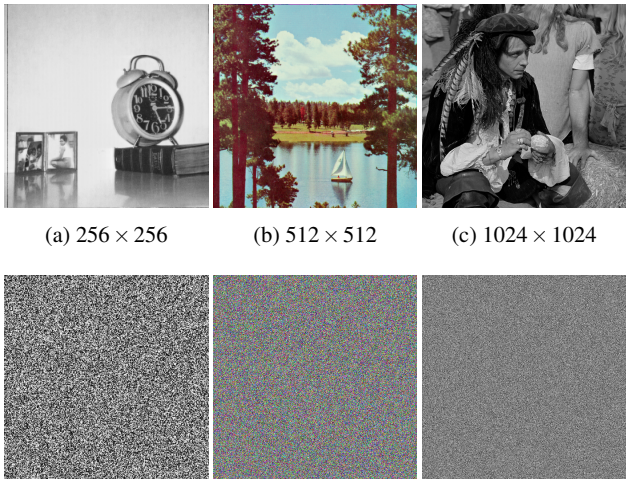
The proposed algorithm has multiple encryption rounds and can produce cipher images using 1 round up to 15 rounds; however, all the test results in this paper are produced using 5 rounds of encryption. The proposed algorithm is analyzed and compared to related algorithms from the literature for a variety of metrics, such as correlation coefficient, histogram analysis tests (chi-square, maximum deviation, irregular deviation, and deviation from uniform histogram), information entropy (global and local), gray-level co-occurrence matrix (GLCM) analysis (contrast, energy, and homogeneity), encryption quality analysis (mean square error (MSE), mean absolute error (MAE), and peak signal-to-noise ratio (PSNR)), resistance to differential attacks, number of pixels change rate (NPCR), unified average changing intensity (UACI), resistance to noise and data loss attacks, and the algorithm's key sensitivity. Each of these tests and their results are discussed in detail below.



(a) (b) (c)

(d) (e) (f)

Fig. 3: Images used to test the proposed algorithm: **(a)** Baboon, **(b)** Cameraman, **(c)** Clock, **(d)** Male, **(e)** Peppers, **(f)** Sailboat.



(a) $256 \times 256$ (b) $512 \times 512$ (c) $1024 \times 1024$

(d) (e) (f)

Fig. 4: Sample images and their respective encrypted ciphers: **(a)** Clock, **(b)** Sailboat, **(c)** Male, **(d)** Clock cipher, **(e)** Sailboat cipher, **(f)** Male cipher.

### 6.1 Correlation Coefficient Analysis

Digital images have strong correlation between adjacent pixels, and an image encryption algorithm should eliminate this correlation to be secure and resilient against statistical attacks. The correlation coefficient metric is used to quantify this correlation, and it has a range of [1, −1], where 1 and −1 represent the maximum positive and maximum negative correlation values, respectively, and 0 means no correlation between the compared pixels. The proposed algorithm is evaluated for the horizontal, vertical, and diagonal correlation coefficients between adjacent pixels of a cipher image (Section 6.1.1), and the horizontal, vertical, and diagonal correlation coefficients between the plaintext image and its cipher (Section 6.1.2).

#### *6.1.1 Correlation Coefficient of Adjacent Pixels*

The vertical correlation coefficient of an image for adjacent pixels can be calculated as:

$$CC_v = \frac{\sum_{i=1}^{H-1} \sum_{j=1}^{W} (C_{(i,j)} - \overline{C})(C_{(i+1,j)} - \overline{C})}{\sqrt{\sum_{i=1}^{H-1} \sum_{j=1}^{W} (C_{(i,j)} - \overline{C})^2 \sum_{i=1}^{H-1} \sum_{j=1}^{W} (C_{(i+1,j)} - \overline{C})^2}}, \quad (2)$$

where $H$ and $W$ are the height and width of the image, respectively. $C_{(i,j)}$ and $C_{(i+1,j)}$ denote the values of two adjacent pixels in the cipher image at positions $(i, j)$ and $(i+1, j)$, respectively. $\overline{C}$ denotes the mean of cipher image's pixel values. In order to calculate the horizontal and the diagonal correlation coefficient, Equation (2) is adjusted such that the value of the pixel at $C_{(i,j+1)}$ is used to calculate the horizontal correlation coefficient, and the value of the pixel at $C_{(i+1,j+1)}$ is used to calculate diagonal correlation coefficient. Table 2 presents the results for the vertical, horizontal, and diagonal correlation coefficients of adjacent pixels for the cipher images. The average of the algorithm's correlation coefficient of adjacent pixels is calculated as an absolute average, that is, any deviation from zero means the presence of correlation (positive or negative) between the two compared pixels. Table 3 depicts a comparison of the vertical, horizontal, and diagonal correlation coefficients of the encrypted Baboon image for the proposed algorithm and other related encryption algorithms in the literature. Comparison results show that the proposed algorithm has smaller values of correlation coefficients than the contemporary image encryption algorithms. Even though the vertical correlation coefficient of the encryption algorithm proposed in [5] is better than the proposed algorithm, the vertical correlation coefficient of the proposed algorithm is extremely close to 0 which means that the correlation between the adjacent vertical pixels is negligible, and thus the proposed algorithm is resilient to statistical attacks.

Table 2: Vertical, horizontal, and diagonal correlation coefficients of adjacent pixels of encrypted images.

| | | Correlation Coefficient | | |
|---|---|---|---|---|
| **Image** | **Size and Color** | **Vertical** | **Horizontal** | **Diagonal** |
| Baboon | 512 × 512 Color | -0.0005 | 0.0004 | 0.0010 |
| Cameraman | 256 × 256 Gray | 0.0003 | -0.0004 | -0.0004 |
| Clock | 256 × 256 Gray | 0.0003 | 0.0003 | -0.0020 |
| Male | 1024 × 1024 Gray | 0.0003 | 0.0004 | 0.0004 |
| Peppers | 256 × 256 Color | 0.0020 | 0.0005 | 0.0006 |
| Sailboat | 512 × 512 Color | 0.0001 | -0.0009 | 0.0001 |
| Average$_{abs}$ | | 0.0006 | 0.0005 | 0.0008 |

Table 3: Vertical, horizontal, and diagonal correlation coefficients of the encrypted Baboon image for the proposed algorithm compared to different encryption methods.

| | Correlation Coefficient | | |
|---|---|---|---|
| | **Vertical** | **Horizontal** | **Diagonal** |
| Proposed | 0.0006 | **0.0005** | **0.0007** |
| [5] | **−0.0001** | 0.0006 | −0.0021 |
| [6] | −0.0086 | 0.0023 | 0.0402 |
| [7] | −0.0036 | −0.0019 | −0.0033 |
| [8] | −0.0004 | −0.0007 | 0.0029 |

### *6.1.2 Correlation Coefficient Between Plaintext and Cipher Images*

The correlation coefficient between a plaintext image and its cipher image can be calculated as:

$$CC_{P,C} = \frac{\sum_{i=1}^{H}\sum_{j=1}^{W}(P_{(i,j)}-\overline{P})(C_{(i,j)}-\overline{C})}{\sqrt{\sum_{i=1}^{H}\sum_{j=1}^{W}(P_{(i,j)}-\overline{P})^2\sum_{i=1}^{H}\sum_{j=1}^{W}(C_{(i,j)}-\overline{C})^2}}, \tag{3}$$

where $P_{(i,j)}$ and $C_{(i,j)}$ are the values of the pixel at index $i, j$ of the plaintext image and cipher image, respectively. $\overline{P}$ and $\overline{C}$ represent the mean of the pixels' values of the plaintext image and the cipher image, respectively. The vertical, horizontal, and diagonal correlation coefficients between the plaintext images and their cipher images for the proposed algorithm are presented in Table 4. The average of the algorithm's correlation coefficient between a plaintext and its cipher is calculated as an absolute average, that is, any deviation from zero is means the presence of correlation between the two compared pixels. The results show that the proposed algorithm's correlation coefficient values between the plaintext and cipher images are close to 0, which means that the cipher images have negligible to no correlation with their corresponding plaintext images.

## 6.2 Histogram Analysis

In the image encryption field, histogram analysis is used to evaluate the uniformity of the cipher image's histogram. The histogram of an image is an important index that reflects the distribution of the values of an image's pixels. A cipher image with a uniform histogram prevents attackers from extracting useful information from the cipher image. Figure 5 shows histograms of sample plaintext images and their respective cipher images. It can be seen from Figure 5 that the histograms of the cipher images are almost uniformly distributed. Hence, the proposed algorithm is effective in hiding the plaintext image's information.

Table 4: Vertical, horizontal, and diagonal correlation coefficients of plaintext and cipher images.

| | | Correlation Coefficient | | |
|---|---|---|---|---|
| **Image** | **Size and Color** | **Vertical** | **Horizontal** | **Diagonal** |
| Baboon | 512 × 512 Color | -0.0141 | -0.0123 | 0.0003 |
| Cameraman | 256 × 256 Gray | -0.0349 | -0.0251 | 0.0012 |
| Clock | 256 × 256 Gray | -0.0607 | 0.0119 | 0.0018 |
| Male | 1024 × 1024 Gray | 0.0011 | -0.0060 | 0.0004 |
| Peppers | 256 × 256 Color | -0.0102 | -0.0052 | 0.0004 |
| Sailboat | 512 × 512 Color | 0.0000 | 0.0000 | 0.0003 |
| $\text{Average}_{abs}$ | | 0.0202 | 0.0101 | 0.0007 |

Thus, is considered to be secure against histogram analysis attacks. To quantify histogram analysis, a set of metrics are used, namely chi-square ($\chi^2$), maximum deviation, irregular deviation, and deviation from the uniform histogram.

### 6.2.1 Chi-Square ($\chi^2$)

Chi-Square ($\chi^2$) is mathematically represented as:

$$\chi^2 = \sum_{i=0}^{255} \frac{(f_i - \mathscr{E})^2}{\mathscr{E}}, \tag{4}$$

$$\mathscr{E} = \frac{H \times W}{256},$$

where $f$ denotes the cipher image's histogram, $f_i$ represents the value of the cipher image's histogram at index $i$, $\mathscr{E}$ denotes the expected value of the histogram of the cipher image, and $H$ and $W$ are the height and width of the image, respectively. When an image has a histogram distribution closer to a uniform distribution, then the value of $\chi^2$ is lower. Furthermore, a uniformly distributed histogram has a $\chi^2$ value of 0. The $\chi^2$ test results listed in Table 5 indicate that the images encrypted by the proposed algorithm have extremely low $\chi^2$ values, and are nearly uniformly distributed. Table 6 compares $\chi^2$ test results of the proposed algorithm to other contemporary image encryption algorithms. Results in Table 6 show that the proposed algorithm is better than the contemporary algorithms with respect to $\chi^2$ metric.

### 6.2.2 Maximum Deviation

Maximum deviation ($D_{max}$) metric measures the deviation between the histograms of a plaintext image and its cipher. This metric is useful in determining the quality of the image encryption algorithm [24]. The encryption quality is better when an algorithm produces a cipher image that is highly deviated from the plaintext image. The mathematical expression of maximum deviation can be given as:

$$d = histogram(|P - C|) \tag{5}$$

$$D_{max} = \frac{d_0 + d_{255}}{2} + \sum_{i=1}^{254} d_i, \tag{6}$$

where $d$ is the histogram of the absolute values of the difference between the plaintext image and its cipher, $d_i$ represents the histogram $d$ value at index $i$, and $d_0$ and $d_{255}$ are the histogram values at index 0 and 255, respectively. Table 7 compares the maximum deviation for cipher images encrypted by the proposed algorithm with contemporary encryption algorithms. From the results, the cipher images produced by the proposed algorithm are highly deviated from the original images. Thus have a better encryption quality as compared to the cipher images produced by other encryption algorithms in the literature.

Table 5: Chi-Square test values for cipher images produced by our proposed algorithm.

| **Image** | **Size and Color** | **Chi-square** |
|---|---|---|
| Baboon | 512 × 512 Color | 241.091 |
| Cameraman | 256 × 256 Gray | 218.547 |
| Clock | 256 × 256 Gray | 213.797 |
| Male | 1024 × 1024 Gray | 215.956 |
| Peppers | 256 × 256 Color | 241.974 |
| Sailboat | 512 × 512 Color | 239.318 |
| Average | | 228.447 |

Table 6: Chi-Square test values of the proposed algorithm compared to different encryption algorithms.

| | **Proposed** | [5] | [12] | [22] | [23] |
|---|---|---|---|---|---|
| Chi-square | **228.447** | 273.271 | 252.000 | 265.000 | 243.969 |

### 6.2.3 Irregular Deviation

Irregular deviation ($D_{irregular}$) metric measures the deviation of individual pixels of the plaintext and the cipher image. The lower the irregular deviation value of a cipher image, the better the encryption quality of the algorithm. The irregular deviation can be mathematically expressed as follows:

$$D_{irregular} = \sum_{i=0}^{255} \Big[ \big| d_i - D_{avg} \big| \Big], \tag{7}$$

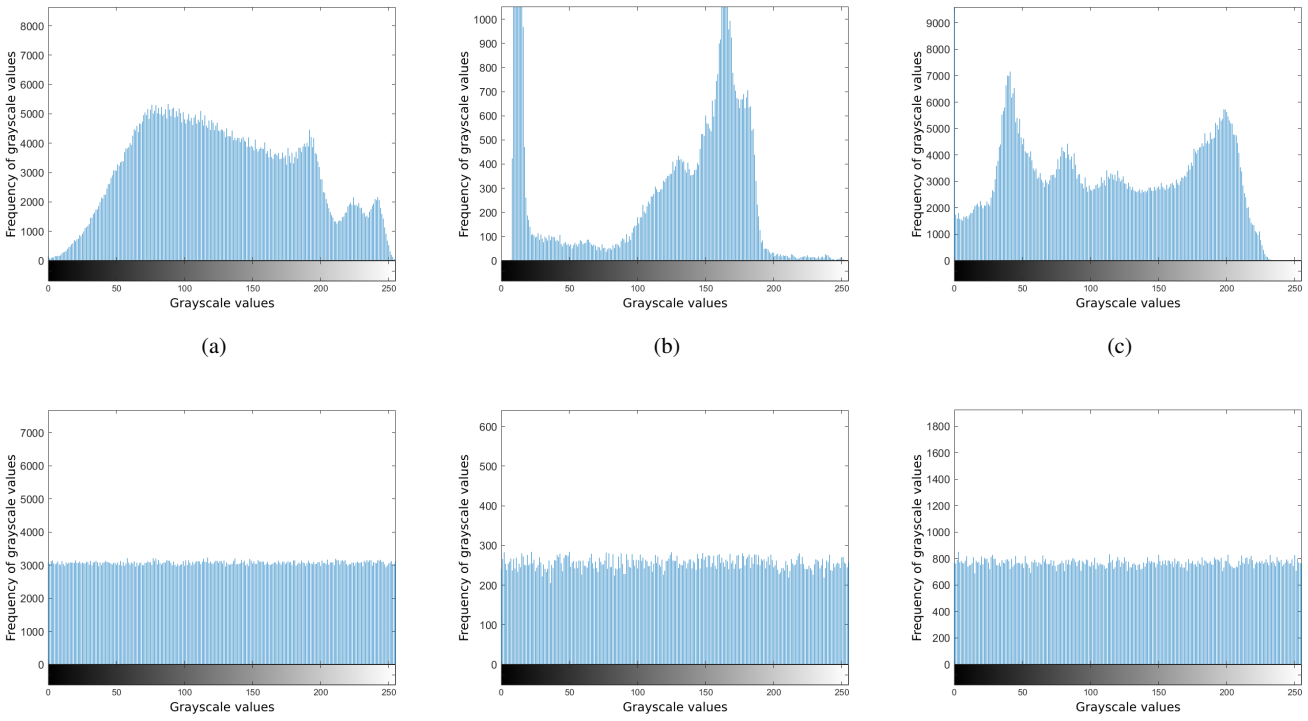



Fig. 5: Histograms of images and their respective ciphers: (**a**) Baboon histogram, (**b**) Cameraman histogram, (**c**) Peppers histogram, (**d**) Baboon cipher histogram, (**e**) Cameraman cipher histogram, (**f**) Peppers cipher histogram.

Table 7: Maximum deviation results of the proposed algorithm as compared to the other algorithms.

| **Image** | **Proposed** | **[5]** | **[7]** | **[25]** |
|---|---|---|---|---|
| Baboon | **438546** | 363121 | 199158 | - |
| Cameraman | **65235** | 64382 | 64998 | 18007 |
| Clock | **62822** | - | - | 24823 |
| Peppers | **256837** | 209618 | 146408 | 22935 |
| Sailboat | **581918** | - | - | 25442 |
| Average | **319498** | 167482 | 102002 | 20109 |

where

$$D_{avg} = \frac{1}{256}\sum_{i=0}^{255} d_i \tag{8}$$

where $d_i$ represents the histogram value at index $i$ (from Equation (5)), and $D_{avg}$ denotes the average value of the pixels that are deviated at every deviation value. Table 8 presents the irregular deviation of cipher images encrypted by the proposed algorithm and compares it with related encryption algorithms. From the results in Table 8, it can be observed that the irregular deviation values of the cipher images produced by the proposed algorithm are small. Even though the performance of Belazi et al. [25] is slightly better than the proposed algorithm for some cipher images, the proposed algorithm has a better average value than all the compared algorithms.

Table 8: Irregular deviation results of the proposed algorithm as compared to the other encryption algorithms.

| **Image** | **Proposed** | **[5]** | **[7]** | **[25]** |
|---|---|---|---|---|
| Baboon | **59876** | 59921 | 80203 | - |
| Cameraman | **32111** | 32165 | 32706 | 39224 |
| Clock | 47060 | - | - | **27442** |
| Peppers | **23901** | 76075 | 84465 | 35088 |
| Sailboat | 72916 | - | - | **36226** |
| Average | **39311** | 91460 | 129253 | 40739 |

### *6.2.4 Deviation from Uniform Histogram*

Deviation from uniform histogram ($D_{uniform}$) metric measures the deviation of a cipher image from a uniform histogram distribution. The closer the histogram value of

a cipher image to the uniform histogram, the better the encryption quality of the algorithm. The deviation from the uniform histogram metric can be mathematically expressed as follows:

$$D_{uniform} = \frac{\sum_{i=0}^{255} |H_{C_i} - H_{U_i}|}{H \times W}, \quad (9)$$

$$H_{U_i} = \begin{cases} \frac{H \times W}{256} & \text{if } 0 \leq i \leq 255 \\ 0 & \text{if } 0 > i > 255 \end{cases}$$

where $H_{C_i}$ denotes the cipher image histogram value at index $i$, and $H_{U_i}$ represents the uniform histogram value at index $i$. It is to be noted that $H_{U_i}$ values are only considered if the pixel value is between [0 – 255], otherwise $H_{U_i}$ values are taken as zero. Table 9 presents the deviation from uniform histogram values for cipher images encrypted by the proposed algorithm and compares it with related encryption algorithms. Results in Table 9 show that the values of deviation from the uniform histogram of the cipher images produced by the proposed algorithm are small. Even though the performance of the encryption algorithm in [5] is slightly better in terms of deviation from the uniform histogram metric for some cipher images, the proposed algorithm has a better average value than all the compared algorithms.

Table 9: Deviation from uniform histogram of the proposed algorithm as compared to other encryption algorithms.

| Image | Proposed | [5] | [7] | [25] |
|---|---|---|---|---|
| Baboon | **0.0242** | 0.0256 | 0.9990 | - |
| Cameraman | 0.0477 | **0.0305** | 0.9961 | 0.0942 |
| Clock | **0.0445** | - | - | 0.0949 |
| Peppers | 0.0479 | **0.0315** | 0.9990 | 0.0917 |
| Sailboat | **0.0241** | - | - | 0.0958 |
| Average | **0.0333** | 0.0382 | .9984 | 0.0940 |

## 6.3 Entropy Analysis

Information entropy is an important metric to estimate the uncertainty of image information. This metric is based on Shannon principle that was introduced by Claude Shannon in 1948 [26]. The entropy value of a cipher image should be as close as possible to the ideal entropy value of 8 [5]. The higher the entropy value of a cipher image, the better the algorithm is at encrypting the plaintext image. To evaluate the unpredictability and randomness of an encryption algorithm, two types of entropy tests are used: global entropy, and local entropy. Global entropy, also known as Shannon entropy, computes the pixel information for the full image, while local entropy measures the mean entropy of randomly selected non-overlapping blocks. Global entropy can be calculated as:

$$H(m) = -\sum_{i=0}^{2^n-1} p(m_i) \log_2[p(m_i)], \quad (10)$$

where $n$ denotes the number of bits used to represent the symbol $p(m_i)$, and $p(m_i)$ signifies the probability of symbol $m_i$, that is, the occurrence probability of intensity $i$ for a pixel in the image. Entropy for an image is calculated using Equation (10), where $p(m_i)$ can be calculated using normalized histogram counts for each intensity value in the image. Local entropy can be calculated as:

$$H_{k,T_B}(S) = \sum_{i=1}^{K} \frac{H(S_i)}{K}, \quad (11)$$

where $S$ represents a set of randomly selected non-overlapping blocks containing $T_B$ pixels, and $K$ is the number of random blocks. $H(S_i)$ is the Shannon entropy (from Equation (10)) of the $i$th block.

Table 10 presents the entropy analysis of the proposed algorithm for different cipher images. Results in Table 10 show that the entropy values of the cipher images produced by our proposed algorithm are extremely close to the ideal entropy. Table 11 presents a comparison of the global entropy of the cipher images produced by our proposed encryption algorithm to that of other encryption algorithms in literature. Results show that the global entropy of the cipher images encrypted by the encryption algorithm in [5] is better than our proposed encryption algorithm but the difference is negligibly small (i.e., 0.00125%).

Table 10: Global and local entropy values of images encrypted by the proposed algorithm.

| | | Entropy | |
|---|---|---|---|
| **Image** | **Size and Color** | **Global** | **Local** |
| Baboon | 512 × 512 Color | 7.9998 | 7.8949 |
| Cameraman | 256 × 256 Gray | 7.9996 | 7.8948 |
| Clock | 256 × 256 Gray | 7.9995 | 7.9026 |
| Male | 1024 × 1024 Gray | 7.9999 | 7.8883 |
| Peppers | 256 × 256 Color | 7.9992 | 7.8880 |
| Sailboat | 512 × 512 Color | 7.9998 | 7.8924 |
| Average | | 7.9996 | 7.8935 |

Table 11: Global entropy of cipher image Baboon produced by the proposed algorithm compared to other encryption algorithms.

| Entropy | Proposed | [5] | [6] | [7] | [8] |
|---|---|---|---|---|---|
| Global | 7.9996 | **7.9997** | 7.9024 | 7.9992 | 7.9971 |

## 6.4 Gray-Level Co-Occurrence Matrix (GLCM) Analysis

The GLCM is a statistical test that is used to inspect the image texture by considering the spatial relationship of pixels. The GLCM is a two-dimensional matrix of joint probabilities between pairs of pixels with specific values [27]. Specifically, the GLCM calculates the frequency of a pixel with gray-level value $i$ occurring horizontally adjacent to a pixel with the gray-level value $j$. The GLCM is helpful in feature extraction and texture analysis. From the GLCM, statistical measures are extracted that provide information about the image's texture features. These measures include homogeneity, contrast, and energy. In the following, we discuss these measurements for the proposed image encryption algorithm along with their comparison with other image encryption algorithms.

### *6.4.1 Homogeneity*

Homogeneity analysis reflects the closeness of the elements in the GLCM distribution to the GLCM diagonal. Homogeneity range is [0, 1], where smaller values of homogeneity reflect better encryption. Higher values of homogeneity means similar pixels are close to each other, which could reveal information about the plaintext image. The homogeneity formula is as follows:

$$Homogeneity = \sum_{i,j} \frac{p(i,j)}{1+|i-j|}, \tag{12}$$

where $i$ and $j$ are two gray-level values in the GLCM, and $p(i,j)$ is the probability of $i$ and $j$ occurring horizontally adjacent to each other in the image, and is given by the value of the element at position $(i,j)$ in the normalized GLCM. The homogeneity analysis results for plaintext images and their cipher images produced by our proposed encryption algorithm are listed in Table 12. Results in Table 12 indicate that the proposed algorithm produces low values of homogeneity.

### *6.4.2 Contrast*

Contrast analysis reflects the intensity difference between a pixel and its neighbor over the whole image. A higher value of contrast between neighboring pixels indicates better encryption. Furthermore, a higher level of contrast among neighboring pixels signifies better randomness in the cipher image. The contrast can be calculated as follows:

$$Contrast = \sum_{i,j} |i-j|^2 p(i,j), \tag{13}$$

where $i$ and $j$ are two gray-level values in the GLCM. The contrast analysis results for plaintext images and their cipher images are presented in Table 13. The results show that cipher images produced by the proposed algorithm have high levels of contrast.

Table 12: Homogeneity analysis for plaintext and cipher images produced by the proposed encryption algorithm.

| Image | Size and Color | Plaintext Image | Cipher Image |
|---|---|---|---|
| Baboon | 512 × 512 Color | 0.7653 | 0.3899 |
| Cameraman | 256 × 256 Gray | 0.8953 | 0.3889 |
| Clock | 256 × 256 Gray | 0.9131 | 0.3897 |
| Male | 1024 × 1024 Gray | 0.8979 | 0.3893 |
| Peppers | 256 × 256 Color | 0.8837 | 0.3894 |
| Sailboat | 512 × 512 Color | 0.8475 | 0.3898 |
| Average | | | 0.3895 |

Table 13: Contrast analysis for plaintext and cipher images produced by the proposed encryption algorithm.

| Image | Size and Color | Plaintext Image | Cipher Image |
|---|---|---|---|
| Baboon | 512 × 512 Color | 0.74257 | 10.4995 |
| Cameraman | 256 × 256 Gray | 0.58716 | 10.5647 |
| Clock | 256 × 256 Gray | 0.36036 | 10.5433 |
| Male | 1024 × 1024 Gray | 0.25280 | 10.5170 |
| Peppers | 256 × 256 Color | 0.42068 | 10.5549 |
| Sailboat | 512 × 512 Color | 0.41400 | 10.5347 |
| Average | | | 10.5357 |

### *6.4.3 Energy*

Energy is calculated by the sum of squared elements in the GLCM. The range of energy values is [0, 1], and an encryption algorithm is considered more secure if the cipher images produced by the algorithm have low energy values. The energy can be calculated from the following expression:

$$Energy = \sum_{i,j} p(i,j)^2, \tag{14}$$

where $i$ and $j$ are two gray-level values occurring horizontally to each other in the GLCM. The energy analysis results for plaintext images and their cipher images produced by the proposed algorithm are presented in Table 14. The results show that the cipher images produced by the proposed algorithm have low values of energy.

Table 15 compares the average values of homogeneity, contrast, and energy of the cipher images produced by the proposed algorithm against other encryption algorithms. The results reveal that the proposed algorithm produces excellent values for these metrics when compared to other encryption algorithms; however, the energy value in [28] is slightly lower (i.e., by 0.128%) than the proposed algorithm. Furthermore, the homogeneity value in [5] is also marginally lower (i.e., by 1.49%) than the proposed algorithm.

Table 14: Energy analysis for plaintext and cipher images produced by the proposed algorithm.

| Image | Size and Color | Plaintext Image | Cipher Image |
|---|---|---|---|
| Baboon | 512 × 512 Color | 0.06399 | 0.01563 |
| Cameraman | 256 × 256 Gray | 0.18054 | 0.01564 |
| Clock | 256 × 256 Gray | 0.20371 | 0.01564 |
| Male | 1024 × 1024 Gray | 0.11994 | 0.01563 |
| Peppers | 256 × 256 Color | 0.14053 | 0.01564 |
| Sailboat | 512 × 512 Color | 0.11551 | 0.01563 |
| Average | | | 0.01563 |

Table 15: Comparison of homogeneity, contrast, and energy values of cipher images produced by the proposed algorithm against other encryption algorithms.

| | Proposed | [5] | [13] | [28] | [29] |
|---|---|---|---|---|---|
| Contrast | **10.5357** | 10.5081 | 10.5079 | 8.6448 | 8.3301 |
| Energy | 0.01563 | 0.01563 | 0.01562 | **0.01561** | 0.01760 |
| Homogeneity | 0.3895 | **0.3837** | 0.3895 | 0.4110 | 0.4208 |

## 6.5 Encryption Quality

The quality of the image encryption is evaluated by various metrics, such as mean square error (MSE), mean absolute error (MAE), and peak signal-to-noise ratio (PSNR). We have evaluated our proposed image encryption algorithm for these metrics. The results of these tests for our proposed image encryption algorithm as well as comparison with contemporary encryption algorithms are detailed below.

### *6.5.1 Mean Square Error*

The MSE is a statistical test that calculates the average squared difference between pixel values of two images. In image encryption, it is used to measure the average squared difference between pixel values of a plaintext image and its cipher image. The MSE between a plaintext image and its cipher image can be calculated as follows:

$$MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} [P(i,j) - C(i,j)]^2, \tag{15}$$

where $P(i,j)$ and $C(i,j)$ are the plaintext image and the cipher image pixel values at position $(i,j)$, respectively. High values of MSE imply that an encryption algorithm performs high quality image encryption.

Table 16 lists the MSE values between the plaintext images and their cipher images for the proposed algorithm. Results indicate that the proposed algorithm yields high MSE values (i.e., $\geq 30$), and thus performs high quality encryption. Table 17 presents a comparison of the proposed image encryption algorithm to other image encryption algorithms in literature in terms of the MSE metric. The comparison results show that cipher images produced by the proposed algorithm have better encryption quality than other encryption algorithms; however, encryption algorithm in [5] performs slightly better (i.e., by 4.75%) than the proposed algorithm in terms of the MSE metric.

Table 16: MSE values of images encrypted by the proposed algorithm.

| Image | Color and size | MSE |
|---|---|---|
| Baboon | 512 × 512 Color | 46.0810 |
| Cameraman | 256 × 256 Gray | 45.3302 |
| Clock | 256 × 256 Gray | 32.4363 |
| Male | 1024 × 1024 Gray | 48.1911 |
| Peppers | 256 × 256 Color | 26.8998 |
| Sailboat | 512 × 512 Color | 43.8024 |
| Average | | 40.4568 |

Table 17: MSE values for images encrypted by the proposed algorithm compared to related methods.

| | Proposed | [5] | [7] | [30] | [31] |
|---|---|---|---|---|---|
| MSE | 40.4568 | **42.3794** | 39.6794 | 33.4275 | 40.3295 |

### 6.5.2 Mean Absolute Error

The MAE is a statistical test that calculates the difference between the pixel values of two images. In image encryption, it is used to measure the difference between the pixel values of a plaintext image and its cipher image. High values of MAE imply high quality encryption by an encryption algorithm. The MAE can be calculated as follows:

$$MAE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} |P(i,j) - C(i,j)|, \quad (16)$$

where $P(i,j)$ and $C(i,j)$ are the plaintext image and its cipher image pixel values at position $(i,j)$, respectively.

Table 18 lists the MAE values between the plaintext images and their cipher images for the proposed algorithm. Results indicate that the proposed algorithm yields high MAE values, and thus performs high quality encryption. Table 19 compares the MAE values of the proposed algorithm with other image encryption algorithms. The comparison results show that cipher images produced by the proposed algorithm have better encryption quality than other encryption algorithms in terms of MAE metric.

Table 18: MAE values of images encrypted by the proposed algorithm.

| Image | Color and size | MAE |
|---|---|---|
| Baboon | 512 × 512 Color | 89.8921 |
| Cameraman | 256 × 256 Gray | 94.1693 |
| Clock | 256 × 256 Gray | 97.7385 |
| Male | 1024 × 1024 Gray | 97.6980 |
| Peppers | 256 × 256 Color | 96.9876 |
| Sailboat | 512 × 512 Color | 96.7950 |
| Average | | 95.5468 |

Table 19: MAE values for images encrypted by the proposed algorithm compared to related methods.

| | Proposed | [5] | [13] | [32] | [33] |
|---|---|---|---|---|---|
| MAE | **95.5468** | 91.58 | 79.57 | 78.10 | 90 |

### 6.5.3 Peak Signal-to-Noise Ratio

In the image encryption field, the PSNR statistical test is used to measure the noise ratio between the plaintext image and its cipher image. PSNR can be calculated as follows:

$$PSNR = 10 \log_{10} \frac{MAX_p}{MSE}, \quad (17)$$

where $MSE$ is the mean squared error value which can be calculated by using Equation (15), and $MAX_p$ is the maximum value a pixel can have (i.e., 255 in 8-bit pixels). Low values of PSNR of cipher images imply that an encryption algorithm produces good quality image encryption. Table 20 presents the PSNR values of cipher images encrypted by the proposed algorithm. Results in Table 20 indicate that the proposed image encryption algorithm has a good quality image encryption. Table 21 compares the PSNR values of the proposed algorithm to other contemporary image encryption algorithms. Results in Table 21 reveal that the cipher images produced by the proposed algorithm have better encryption quality than other encryption algorithms, but the algorithm in [5] performs marginally better (i.e., by 0.13%) than the proposed algorithm for the PSNR metric.

Table 20: PSNR results of images encrypted by the proposed algorithm.

| Image | Size and color | PSNR |
|---|---|---|
| Baboon | 512 × 512 Color | 8.7966 |
| Cameraman | 256 × 256 Gray | 8.4021 |
| Clock | 256 × 256 Gray | 8.2915 |
| Male | 1024 × 1024 Gray | 8.2112 |
| Peppers | 256 × 256 Color | 8.2453 |
| Sailboat | 512 × 512 Color | 8.7947 |
| Average | | 8.4569 |

Table 21: PSNR of the images encrypted by the proposed algorithm compared to other encryption algorithms.

| | Proposed | [5] | [7] | [13] | [34] |
|---|---|---|---|---|---|
| PSNR | 8.4569 | **8.4458** | 9.5424 | 9.0996 | 9.7936 |

## 6.6 Resistance Against Differential Attacks

Differential attacks are used by cryptanalysts to determine the relationship between the plaintext image and its cipher image, by making minor changes to the plaintext image and encrypting it using the same key. To resist such

attacks, an encryption algorithm must have good diffusion characteristics, where changing a pixel in a plaintext image should produce a very different cipher image. The ability of an encryption algorithm to resist differential attacks can be determined by analyzing the algorithm's performance on avalanche effect, number of pixels change rate (NPCR), and unified average changing intensity (UACI) tests. To assess the proposed algorithm's resistance against differential attacks, the same plaintext is encrypted twice. The first cipher image is produced by encrypting the original plaintext image. The second cipher image is produced by changing a random pixel's value. For changing the pixel value in the plaintext image and to assess the resistance to differential attacks for different assigned values, we have considered and evaluated both possible extreme values (0 or 255), as well as the change in the intensity of the pixel by only 1-bit. Each of these tests and their results are discussed in detail below.

### *6.6.1 Avalanche Effect*

Avalanche effect occurs when a minor change in the key or the plaintext image results in a significantly different cipher image. The MSE metric can be used to analyze the avalanche effect. Equation (15) can be modified to compute the mean squared difference between two cipher images produced by encrypting (i) a plaintext image, and (ii) the same plaintext image with a change of only one random pixel. The avalanche effect MSE ($MSE_{av}$) can be calculated as follows:

$$MSE_{av} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} [C_1(i,j) - C_2(i,j)]^2, \tag{18}$$

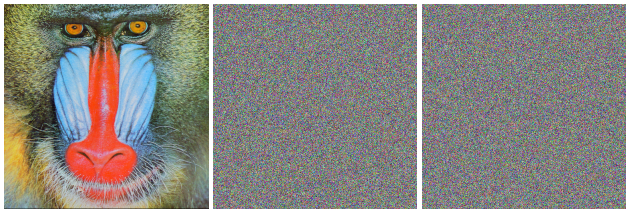
where $C_1$ and $C_2$ are the two cipher images produced by encrypting the two plaintext images, where the two plaintext images differ in only 1 pixel and the rest of the pixels in the two plaintext images are the same. $C_1(i,j)$ and $C_2(i,j)$ denote the pixel values at position $(i,j)$ of the cipher image 1 and the cipher image 2, respectively. Figure 6 shows the plaintext image Baboon and its resulting cipher images. Table 22 lists the $MSE_{av}$ values of the two cipher images that are produced by encrypting two plaintext images that are same but differ in only 1 pixel for the proposed algorithm. Results indicate that the proposed algorithm yields high $MSE_{av}$ values (i.e., $\geq 50$), and thus the cipher images produced by our proposed encryption algorithm are safe against differential attacks. Table 23 lists the $MSE_{av}$ between two cipher images produced by encrypting the image Baboon. The first cipher image is produced by encryption the original plaintext image. The second cipher image is produced by changing a random pixel's value in the plaintext image to: (i) 0, (ii)255, (iii) increasing the intensity of the pixel by 1-bit, and (iv) reducing the intensity of the pixel by 1-bit.

Table 22: Avalanche effect $MSE_{av}$ values of images encrypted by the proposed algorithm.

| **Image** | **Size and Color** | **$MSE_{av}$** |
|---|---|---|
| Baboon | 512 × 512 Color | 57.4265 |
| Cameraman | 256 × 256 Gray | 57.6061 |
| Clock | 256 × 256 Gray | 58.1382 |
| Male | 1024 × 1024 Gray | 57.3568 |
| Peppers | 256 × 256 Color | 57.6522 |
| Sailboat | 512 × 512 Color | 57.5834 |
| Average | | 57.6272 |

Table 23: Avalanche effect $MSE_{av}$ values of the image Baboon encrypted by the proposed algorithm.

| **Random pixel change intensity** | **$MSE_{av}$** |
|---|---|
| Pixel value set to 0 | 57.2865 |
| Pixel value set to 255 | 57.4265 |
| Pixel value increased by 1-bit | 57.2591 |
| Pixel value decreased by 1-bit | 57.3810 |
| Average | 57.3383 |



(a) (b) (c)

Fig. 6: Avalanche effect on two cipher images encrypted from the same plaintext images differing in only a single pixel, (a) Baboon, (b) Baboon cipher, (c) Baboon cipher with 1 pixel changed in the Baboon image.

### *6.6.2 Number of Pixels Change Rate*

NPCR metric sums the differences between two cipher images that are produced by encrypting two plaintext images

that are same but differ in only 1 bit (1 pixel). The NPCR value can be calculated as:

$$NPCR = \sum_{i=1}^{H}\sum_{j=1}^{W}\frac{D(i,j)}{H\times W}, \tag{19}$$

where

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

where $C_1$ and $C_2$ are the two cipher images produced by encrypting the two plaintext images, where the two plaintext images differ in only 1 pixel and the rest of the pixels in the two plaintext images are the same. $C_1(i,j)$ and $C_2(i,j)$ denote the pixel values at position $(i,j)$ of the cipher image 1 and the cipher image 2, respectively. If the pixel values of $C_1$ and $C_2$ at position $(i,j)$ are not different, then $D(i,j)$ has a value of 0, otherwise $D(i,j)$ has a value of 1 if the pixel values are different. The higher the NPCR value, the higher an algorithm's responsiveness to plaintext changes, and thus greater the algorithm's ability to resist differential attacks. In other words, to resist differential attacks, the algorithm should be very sensitive to changes in the plaintext image, where high NPCR values imply higher resistance against differential attacks. The ideal value for NPCR is $\geq 99.6094$ as calculated by Wu et. al. [35]. Table 24 lists the NPCR values for the proposed algorithm. Results indicate that the proposed algorithm yields high NPCR values, and thus the cipher images produced by the proposed encryption algorithm are safe against differential attacks. Table 25 lists the NPCR of the two cipher images produced by encrypting the image Cameraman. The first cipher image is produced by encryption the original plaintext image. The second cipher image is produced by changing a random pixel's value in the plaintext image to: (i) 0, (ii) 255, (iii) increasing the intensity of the pixel by 1-bit, and (iv) reducing the intensity of the pixel by 1-bit. Table 26 compares the NPCR values of the proposed algorithm with other image encryption algorithms. Results in Table 26 indicate that the NPCR values of the proposed algorithm are higher than other encryption algorithms.

Table 24: NPCR values of images encrypted by the proposed algorithm.

| **Image** | **Size and Color** | **NPCR** |
|---|---|---|
| Baboon | 512 × 512 Color | 99.6222 |
| Cameraman | 256 × 256 Gray | 99.6109 |
| Clock | 256 × 256 Gray | 99.6292 |
| Male | 1024 × 1024 Gray | 99.6156 |
| Peppers | 256 × 256 Color | 99.6206 |
| Sailboat | 512 × 512 Color | 99.6199 |
| Average | | 99.6197 |

Table 25: NPCR values of the image Cameraman encrypted by the proposed algorithm.

| **Random pixel change intensity** | **NPCR** |
|---|---|
| Pixel value set to 0 | 99.5727 |
| Pixel value set to 255 | 99.6109 |
| Pixel value increased by 1-bit | 99.5666 |
| Pixel value decreased by 1-bit | 99.5819 |
| Average | 99.5830 |

Table 26: NPCR of the proposed algorithm compared with other encryption methods.

| | **Proposed** | **[5]** | **[6]** | **[7]** | **[8]** |
|---|---|---|---|---|---|
| NPCR | **99.6197** | 99.6153 | 99.5893 | 99.6059 | 99.5743 |

### *6.6.3 Unified Average Changing Intensity (UACI)*

UACI is another quantitative test utilized to assess an algorithm's ability to resist differential attacks. It is used to compute the change in the average intensity between two cipher images that were produced by encrypting the same plaintext image with a change of 1 pixel. The UACI value can be calculated as:

$$UACI = \frac{1}{H\times W}\left[\frac{\sum_{i=1}^{H}\sum_{j=1}^{W}|C_1(i,j)-C_2(i,j)|}{255}\right]\times 100\%, \tag{20}$$

where $C_1$ and $C_2$ are two cipher images produced by encrypting the same plaintext image with a change of 1 pixel. $C_1(i,j)$ and $C_2(i,j)$ are the cipher images 1 and 2 pixel values at position$(i,j)$, respectively. An algorithm with high responsiveness to plaintext change has higher UACI values. The ideal value for UACI is $\geq 33.4635$ as calculated by Wu et. al. [35]. Table 27 lists the UACI values for the proposed algorithm. Results indicate that the proposed algorithm yields high UACI values, and thus is resilient against differential attacks. Table 28 lists the UACI of two cipher images produced by encrypting the image Clock. The first cipher image is produced by encryption the original plaintext image. The second cipher image is produced by changing a random pixel's value in the plaintext image to: (i) 0, (ii) 255, (iii) increasing the intensity of the pixel by 1-bit, and (iv) reducing the intensity of the pixel by 1-bit. Table 29 compares the UACI values of the proposed algorithm with other image encryption algorithms. Results

in Table 29 indicate that the UACI values of the proposed algorithm are better than other encryption algorithms.

Table 27: UACI values of images encrypted by the proposed algorithm.

| Image | Size and Color | UACI |
|---|---|---|
| Baboon | 512 × 512 Color | 33.4901 |
| Cameraman | 256 × 256 Gray | 33.5461 |
| Clock | 256 × 256 Gray | 33.7048 |
| Male | 1024 × 1024 Gray | 33.4707 |
| Peppers | 256 × 256 Color | 33.5548 |
| Sailboat | 512 × 512 Color | 33.5361 |
| Average | | 33.5504 |

Table 28: UACI values of the image Clock encrypted by the proposed algorithm.

| Random pixel change intensity | UACI |
|---|---|
| Pixel value set to 0 | 33.4870 |
| Pixel value set to 255 | 33.7048 |
| Pixel value increased by 1-bit | 33.4079 |
| Pixel value decreased by 1-bit | 33.5179 |
| Average | 33.4198 |

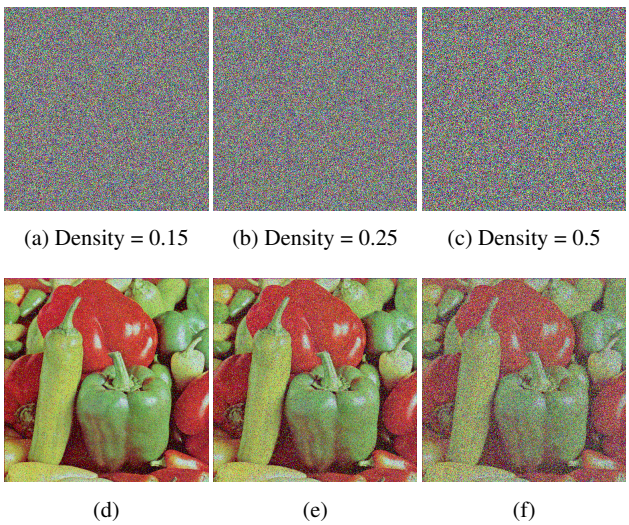
Table 29: UACI of the proposed algorithm compared with other encryption methods.

| | Proposed | [5] | [6] | [7] | [8] |
|---|---|---|---|---|---|
| UACI | **33.5504** | 33.4718 | 33.3730 | 33.4375 | 33.3941 |

### 6.7 Resistance to Noise and Data Loss

In digital communications, images are often affected by noise and loss of data segments. In the case of cipher images, this corruption in the data could prevent the decryption of the image. The noise and data loss could also be introduced by an adversary in an attempt to destroy the cipher images. These attacks are known as *noise attacks* and *occlusion attacks*. An image encryption algorithm should be able to withstand noise and occlusion attacks to a certain limit. Furthermore, the algorithm should be able to retrieve the plaintext image from the altered cipher image.

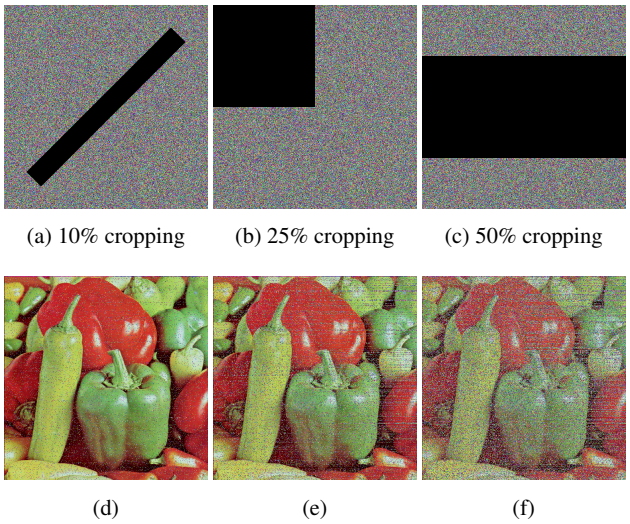
To assess the resilience of the proposed encryption algorithm against noise, we have added salt and pepper noise to the cipher image with densities of 0.15, 0.25, and 0.5. It is to be noted that a noise density of 0.1 affects approximately 10% of pixels, and noise density of 1 affects approximately 100% of pixels. Figure 7 shows cipher images to which salt and pepper noise has been introduced with different densities. Results in Figure 7 reveal that the proposed algorithm is able to decrypt cipher images affected by salt and pepper noise with density up to 0.5. To evaluate the the robustness of the proposed algorithm against occlusion attacks, we have cropped out segments of the cipher image with different severities, that is, 10%, 25%, and 50% cropping. Figure 8 depicts data loss (occlusion) attacks on a cipher image with different severities. Results in Figure 8 reveal that the proposed algorithm is able to successfully decrypt cipher images with cropping up to 50%. It is important to note that the quality of the decrypted images obtained from the encrypted images subjected to noise and/or data loss is expected to be degraded. We have employed these attacks to test the algorithm's ability to decrypt a cipher image affected by noise or cropping during transmission, whether due to noisy channels or adversarial attacks. The tests confirm that the proposed algorithm can withstand such attacks and decrypt the affected cipher image.



(a) Density = 0.15 (b) Density = 0.25 (c) Density = 0.5

(d) (e) (f)

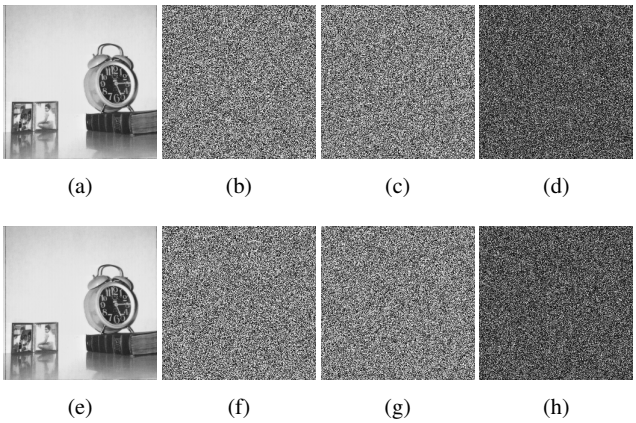
Fig. 7: Results of salt and pepper noise attack on the color image Peppers (512 × 512): (**a**–**c**) different noise densities introduced, (**d**–**f**) corresponding decrypted images.



(a) 10% cropping (b) 25% cropping (c) 50% cropping

(d) (e) (f)

Fig. 8: Results of occlusion attacks (data loss) on the color image Peppers ($512 \times 512$): (**a**–**c**) different severities of occlusion attacks, (**d**–**f**) corresponding decrypted images.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 9: Results of key sensitivity tests on the color image Sailboat ($512 \times 512$): (**a**) plaintext image, (**b**) $C_1$ encrypted using key $k_1$, (**c**) $C_2$ encrypted using key $k_2$, (**d**) difference of images: $|(\mathbf{b}) - (\mathbf{c})|$, (**e**) $C_1$ decrypted using key $k_1$, (**f**) $C_1$ decrypted using key $k_2$, (**g**) $C_2$ decrypted using key $k_1$, (**h**) difference of images: $|(\mathbf{f}) - (\mathbf{g})|$.

### 6.8 Key Sensitivity

The sensitivity of an encryption algorithm to minor changes in the secret key is a principal factor in evaluating an encryption algorithm's security. A key-sensitive encryption algorithm should not be able to decrypt a cipher image and recover the plaintext image with a key that is changed by as little as a single bit. Moreover, a change of 1-bit in the key should cause the algorithm to produce two totally different cipher images. Figure 9 depicts the results of key sensitivity tests for the proposed algorithm. From the test results in Figure 9, it can be seen that the proposed algorithm is highly sensitive to changes in the secret key.

*Key sensitivity for encryption:* To evaluate the sensitivity of the proposed algorithm to slight changes in the key for encryption, we have encrypted the image Clock using two keys $k_1$ and $k_2$ that differ in only 1-bit. Results in Figure 9 (b), (c), and (d) show that the proposed algorithm produces two totally different cipher images for the keys $k_1$ and $k_2$.

*Key sensitivity for decryption:* To assess the key sensitivity to changes in the key for decryption, we have encrypted the image Clock with a key $k_1$ to produce a cipher image $C_1$. Results in Figure 9 (f), (g), and (h) show that decrypting $C_1$ with a key $k_2$ that differs in only 1-bit from key $k_1$ does not recover the plaintext image.

## 7 Conclusions

In this paper, a secure multi-round image encryption algorithm based on nonlinear-feedback shift registers is proposed. The proposed algorithm uses Trivium cipher to produce key-streams from an 80-bit round key and an 80-bit IV. Each key-stream is used once for each round and is used to perform pixel-based row permutation, pixel-based column permutation, and bit-level substitution using bitwise XOR on the plaintext image to produce the cipher image. The proposed algorithm is analyzed and compared with various encryption algorithms in literature utilizing a comprehensive set of quality, efficiency, and security metrics. The experimental results reveal that the proposed algorithm produces better results in terms of correlation coefficient, chi-square, maximum deviation, irregular deviation, and deviation from the uniform histogram when compared to other contemporary image encryption algorithms. Furthermore, the image texture analysis using the GLCM's homogeneity, contrast, and energy show that the proposed algorithm yields better encryption as compared to other image encryption algorithms. When tested for encryption quality, that is, via mean square error, mean absolute error, and peak signal-to-noise ratio metrics, the proposed algorithm demonstrates excellent quality results that are similar to other contemporary image encryption methods. Additionally, experimental results reveal that the proposed algorithm is resilient against differential attacks and noise and occlusion attacks.

One of the limitations for the proposed algorithm that needs to be tested is the algorithm's resilience against quantum computing attacks. Even though the proposed algorithm increases the Trivium key space of $2^{80}$ thus making it more difficult for attackers to break the algorithm using brute-force attacks, its security against attacks by quantum computers needs to be analysed. Moreover, the proposed algorithm needs further optimization to be

compression friendly. At it current state, the algorithm cannot correctly decrypt a cipher image that was compressed after it was encrypted. The image will be decrypted but a large amount of data will be lost. Furthermore, the proposed algorithm uses a software implementation of the Trivium cipher, but the Trivium cipher delivers much faster encryption speed in hardware as compared to software. In our future work, we plan to implement the proposed algorithm in hardware using a field-programmable gate array (FPGA). Furthermore, we plan to explore acceleration or parallelization of the proposed image encryption algorithm.

### Research Data Policy and Data Availability Statements

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

### Compliance with Ethical Standards

The authors declare that they have no conflicts of interest. This article does not contain any studies involving human participants and/or animals performed by any of the authors.

### Competing Interests

The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

### References


1. Dworkin, M., Barker, E., Nechvatal, J., Foti, J., Bassham, L., Roback, E., Dray, J.: Advanced Encryption Standard (AES). Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD (2001). https://doi.org/10.6028/NIST.FIPS.197
2. Pub, F.: Data encryption standard (des). FIPS PUB, 46–3 (1999)
3. Talhaoui, M.Z., Wang, X.: A new fractional one dimensional chaotic map and its application in high-speed image encryption. Information Sciences **550**, 13–26 (2021)
4. Gao, X.: Image encryption algorithm based on 2d hyperchaotic map. Optics & Laser Technology **142**, 107252 (2021)
5. Alghamdi, Y., Munir, A., Ahmad, J.: A lightweight image encryption algorithm based on chaotic map and random substitution. Entropy **24**(10), 1344 (2022). https://doi.org/10.3390/e24101344
6. Hua, Z., Zhou, Y., Pun, C.-M., Chen, C.P.: 2d sine logistic modulation map for image encryption. Information Sciences **297**, 80–94 (2015)
7. Arif, J., Khan, M.A., Ghaleb, B., Ahmad, J., Munir, A., Rashid, U., Al-Dubai, A.: A novel chaotic permutation–substitution image encryption scheme based on logistic map and random substitution. IEEE Access (2022)
8. Lu, Q., Zhu, C., Deng, X.: An efficient image encryption scheme based on the lss chaotic map and single s-box. IEEE Access **8**, 25664–25678 (2020)
9. Wu, T.-Y., Fan, X., Wang, K.-H., Lai, C.-F., Xiong, N., Wu, J.M.-T.: A dna computation-based image encryption scheme for cloud cctv systems. IEEE Access **7**, 181434–181443 (2019)
10. Zhang, Y., Zhang, L., Zhong, Z., Yu, L., Shan, M., Zhao, Y.: Hyperchaotic image encryption using phase-truncated fractional fourier transform and dna-level operation. Optics and Lasers in Engineering **143**, 106626 (2021)
11. Uddin, M., Jahan, F., Islam, M.K., Rakib Hassan, M.: A novel dna-based key scrambling technique for image encryption. Complex & Intelligent Systems **7**(6), 3241–3258 (2021)
12. Xiao, D., Kulsoom, A., Hashmi, M.A., Abbas, S.A., *et al.*: Block mode image encryption technique using two-fold operations based on chaos, md5 and dna rules. Multimedia Tools and Applications **78**(7), 9355–9382 (2019)
13. Samiullah, M., Aslam, W., Nazir, H., Lali, M.I., Shahzad, B., Mufti, M.R., Afzal, H.: An image encryption scheme based on dna computing and multiple chaotic systems. IEEE Access **8**, 25650–25663 (2020)
14. Wang, X.-Y., Li, Z.-M.: A color image encryption algorithm based on hopfield chaotic neural network. Optics and Lasers in Engineering **115**, 107–118 (2019)
15. Man, Z., Li, J., Di, X., Sheng, Y., Liu, Z.: Double image encryption algorithm based on neural network and chaos. Chaos, Solitons & Fractals **152**, 111318 (2021)
16. Dou, Y., Li, M.: An image encryption algorithm based on compressive sensing and m sequence. IEEE Access **8**, 220646–220657 (2020)
17. Zhang, M., Tong, X.-J., Liu, J., Wang, Z., Liu, J., Liu, B., Ma, J.: Image compression and encryption scheme based on compressive sensing and fourier transform. IEEE Access **8**, 40838–40849 (2020)
18. Ping, P., Fu, J., Mao, Y., Xu, F., Gao, J.: Meaningful encryption: generating visually meaningful encrypted images by compressive sensing and reversible color transformation. IEEE access **7**, 170168–170184 (2019)
19. Canniere, C.D., Preneel, B.: Trivium specifications. eSTREAM, ECRYPT Stream Cipher Project **2006** (2006). Accessed: 2022-10-10
20. Qureshi, M.A., Munir, A.: PUF-RAKE: A PUF-based

robust and lightweight authentication and key establishment protocol. IEEE Transactions on Dependable and Secure Computing (TDSC) **19**(4), 2457–2475 (2022)
21. USC-SIPI Image Database. Volume 3: Miscellaneous. Accessed: 2022-10-10
22. Etemadi Borujeni, S., Eshghi, M.: Chaotic image encryption system using phase-magnitude transformation and pixel substitution. Telecommunication Systems **52**(2), 525–537 (2013)
23. Moafimadani, S.S., Chen, Y., Tang, C.: A new algorithm for medical color images encryption using chaotic systems. Entropy **21**(6) (2019). https://doi.org/10.3390/e21060577
24. Faragallah, O.S.: Digital image encryption based on the rc5 block cipher algorithm. Sensing and Imaging: An International Journal **12**(3), 73–94 (2011)
25. Belazi, A., Abd El-Latif, A.A., Belghith, S.: A novel image encryption scheme based on substitution-permutation network and chaos. Signal Processing **128**, 155–170 (2016)
26. Shannon, C.E.: A mathematical theory of communication. The Bell system technical journal **27**(3), 379–423 (1948)
27. Jawad, L.M.: A novel region of interest for selective color image encryption technique based on new combination between glcm texture features. In: 2021 National Computing Colleges Conference (NCCC), pp. 1–6 (2021). IEEE
28. Khan, F.A., Ahmed, J., Khan, J.S., Ahmad, J., Khan, M.A.: A novel image encryption based on lorenz equation, gingerbreadman chaotic map and s 8 permutation. Journal of Intelligent & Fuzzy Systems **33**(6), 3753–3765 (2017)
29. Anees, A., Siddiqui, A.M., Ahmed, F.: Chaotic substitution for highly autocorrelated data in encryption algorithm. Communications in Nonlinear Science and Numerical Simulation **19**(9), 3106–3118 (2014)
30. Ahmad, J., Ahmed, F.: Efficiency analysis and security evaluation of image encryption schemes. computing **23**, 25 (2010)
31. Khan, J., Ahmad, J., Hwang, S.O.: An efficient image encryption scheme based on: Henon map, skew tent map and s-box. In: 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), pp. 1–6 (2015). IEEE
32. Alexan, W., ElBeltagy, M., Aboshousha, A.: Rgb image encryption through cellular automata, s-box and the lorenz system. Symmetry **14**(3), 443 (2022)
33. Khan, M., Khan, L., Hazzazi, M.M., Jamal, S.S., Hussain, I.: Image encryption scheme for multi-focus images for visual sensors network. Multimedia Tools and Applications **81**(12), 16353–16370 (2022)
34. Qayyum, A., Ahmad, J., Boulila, W., Rubaiee, S., Masood, F., Khan, F., Buchanan, W.J., *et al.*: Chaos-based confusion and diffusion of image pixels using dynamic substitution. IEEE Access **8**, 140876–140895 (2020)
35. Wu, Y., Noonan, J.P., Agaian, S., *et al.*: Npcr and uaci randomness tests for image encryption. Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT) **1**(2), 31–38 (2011)