Image Encryption Based on 2-D Zaslavskii Chaotic Map and Pseudo Hadmard Transform

Gururaj Hanchinamani and Linganagouda Kulakarni

Department of Computer Science and Engineering, BVB College of Engineering and Technology, Hubli, Karnataka, India 580031

gs_hanchinamani@bvb.edu, linganagouda@yahoo.co.uk

Abstract

Recently, several chaotic based image encryption schemes have been proposed, each of them has its own strengths and limitations more or less in terms of security level and computational speed. In this paper, a novel approach for image encryption based on a 2-D Zaslavskii map and Pseudo Hadmard transform is proposed. The encryption process is composed of two stages, i.e. permutation and diffusion. The permutation is achieved by scrambling rows and columns using chaotic values. This stage substantially reduces the correlation between neighboring pixels. During diffusion, the avalanche effect is achieved with 2-D Pseudo Hadmard transforms followed by diffusion in two directions (forward and backward) with multiple additions and xor operations. This stage ensures resistance against differential attacks. The security and performance of the proposed method is analyzed thoroughly by using key-sensitivity, key-space, statistical, entropy, differential and performance analysis. The proposed scheme achieves the required level of security with only one round of encryption operation. Hence the proposed method is computationally fast.

Keywords: Differential attacks, Image encryption, Statistical analysis, 2-D Zaslavskii map, 2-D pseudo Hadmard transform

1. Introduction

The intrinsic features of images are bulk volume of data, high correlation among adjacent pixels, high redundancy and human perception of decrypted image with small distortions. Hence images are considerably different from text information. The conventional encryption methods such as AES, IDEA, DES, RSA etc., are computationally intensive hence consume more time and are not suitable for images [3-7]. Several image encryption algorithms are already available in the literature, however, some of these suffer with brute-force attack, statistical attack, and differential attacks. In this paper, the computational performance and security of the encryption process is improved by using 2-D Zaslavskii map and Hadmard transform.

During the last years, several image encryption schemes have been proposed in the spatial domain, among which chaotic-based methods are most popular. The encryption algorithms based on chaos offer the advantages to be very sensitive to the initial conditions, ergodicity, randomness and simplicity [7]. Chaotic encryption systems generally have high speed with low cost, which makes them better candidate than conventional methods for multimedia data encryption.

The remaining of the paper is organized as follows. In Section 2, the literature survey is presented. The 2-D Zaslavskii map is described in Section 3. In Section 4, the proposed encryption scheme is discussed in detail. Simulation results and security analysis are presented.
in Section 5 to show the efficacy and validity of the algorithm. Finally, conclusions are drawn in the last section.

2. Literature Survey

The general architecture of the chaotic based image encryption systems typically consists of iteration of two stages (i) permutation and (ii) diffusion. The permutation is achieved by scrambling all the pixels as a whole using 2D chaotic maps [1, 10, 11]. In diffusion stage, the pixel values are altered sequentially and the change made to a particular pixel depends on the accumulated effect of all the previous pixel values. However, as many rounds of permutation and diffusion or iterations should be taken, the overall encryption speed is slow.

A brief overview of the recently proposed chaotic based encryption schemes is given hereafter. To resist brute-force attacks the key space is increased by using multiple chaotic maps in [1, 10, 7]. Shatheesh Sam [2] proposed image encryption based on intertwining chaotic maps to enhance security and keylength. Authors of [3] proposed symmetric encryption scheme based on cyclic elliptic curve and chaotic system, which encrypts 256-bit of plaintext to 256-bit of ciphertext within eight 32 bit registers. Guodong Ye [4] proposed an image encryption scheme with generalized Arnold map, as the key stream depends on the processed image the method can resist known- and chosen-plain text attacks. Authors of [5] proposed a gray level encryption scheme to eliminate image outlines and to disrupt the distributional characteristics of gray level. In [6], the computational time is reduced by encrypting significant data in spatial domain and insignificant data in wavelet domain. Xiaoling Huang [8] proposed encryption scheme based on chaotic Chebyshev generator with multiple permutations to enhance the decorrelation. The authors of [9] proposed image encryption with circle map and is resistant against differential attacks. In [11], a hierarchy of 2-D piecewise nonlinear chaotic maps with an invariant measure is introduced. In [12], an encryption scheme based on large pseudorandom permutation is proposed, which is combinatorially generated from small permutation matrices based on chaotic maps. The authors of [13] combined the chaos-based image encryption with pixel bit. This uses a single chaotic system applied directly to the position scrambling operation.

However, some of the chaotic based encryption schemes have been cryptanalyzed successfully [14, 22]. Liang Zhao et. al., [22] presented differential attack on [13], and proposed an improved scheme using self-correlations. Rhouma et. al., [14] presented attack on [23] with only one pair of plaintext and ciphertext.

Based on the above discussions, though there exist several encryption schemes, each of them has its own strength and limitations more or less in terms of security level and computational speed. To resist statistical, differential, brute-force attacks and to improve the computational performance, this paper proposes a novel chaotic image encryption scheme based on 2-D Zaslavskii map and Pseudo Hadmard transform. The proposed method is resistant to brute-force attacks, statistical attacks and differential attacks with high computational speed. The proposed approach achieves the required level of security with only one round of encryption operation. It can be easily implemented and is computationally simple.

3. Chaotic Maps

Chaotic maps are nonlinear maps that exhibit chaotic behavior. The chaotic maps generate pseudo-random sequences, which are used during encryption process. Chaotic maps are sensitive to initial conditions and parameters, non-convergent, non-periodic and topologically mixing. The proposed scheme uses 2-D Zaslavskii map and is discussed hereafter. The 2-D Zaslavskii map is a discrete-time dynamical system, and is defined as
\[ X_{n+1} = (X_n + \nu (1 + \mu Y_n) + \varepsilon \nu \mu \cos (2\pi X_n)) \mod 1 \]  
\[ Y_{n+1} = e^{-\tau} (Y_n + \varepsilon \cos (2\pi X_n)) \]  
and \( \mu = \frac{1 - e^{-\tau}}{\tau} \)

Where \( X_n, Y_n \) are current chaotic values and \( X_{n+1}, Y_{n+1} \) are next chaotic values and \( \nu, \varepsilon, \tau \) are control parameters and \( \varepsilon \) is exponentiation. The key set for Zaslavskii map is \( \{X_0, Y_0, \nu, \varepsilon, \tau \} \). Commonly used values for the parameters are \( \nu = 12.6695, \varepsilon = 9.1, \tau = 3.0 \). The propositions of chaotic maps [12] are given in Eq. (3-5). The Zaslavskii chaotic output sequence is analyzed by computing mean and self-correlations according the propositions given in Eq. (3-5). It is observed that the mean value is close to 0.5 and the self correlations within the sequence and across two sequences are very close to 0.

**Proposition 1.** The mean value of the chaotic sequence is

\[ x_{mean} = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} x_k = 0.5 \]  

**Proposition 2.** Self-correlation of a chaotic sequence is

\[ S1(\beta) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} (x_k - x_{mean})(x_{k+\beta} - x_{mean}) = 0 \]  

**Proposition 3.** Self-correlation function between two chaotic sequences is

\[ S2(\beta) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} (x_k - x_{mean})(y_{k+\beta} - y_{mean}) = 0 \]  

\[ \]  

**4. Proposed Encryption Scheme**

The algorithm consists of two stages, i.e., permutation and diffusion.

**4.1. Permutation**

The purpose of permutation is to reduce the high correlation between adjacent pixels in the plain image. Let \( I \) be a gray original image of size \( M \times N \), it is a matrix containing \( M \) rows and \( N \) columns, and the gray values ranges from 0 to 255. In the process of permutation, initially \( M + N \) chaotic values \( \{(X_1, ..., X_M), (Y_1, ..., Y_N)\} \) are generated by using Eq. (1, 2), after doing iterations in chaos maps. Let \( PM = \{X_1, ..., X_M\} \) and \( PN = \{Y_1, ..., Y_N\} \). Then \( PM \) and \( PN \) are sorted, and the positions of sorted chaotic values in the original chaotic sequence are found and stored in \( PM' \) and \( PN' \). The next step is to scramble row position of all values from first column to last column according to \( PM'_1, ..., PM'_M \). Similarly scramble column position of all values from first row to last row according to \( PN'_1, ..., PN'_N \). This stage shuffles all pixels and decorrelates the neighboring pixels.

**4.2. Diffusion**

The diffusion function is employed to modify the gray values of the image pixels to confuse the relationship between the plain image and the encrypted image. The diffusion function is
used to ensure the plain image sensitivity \textit{i.e.}, a very little change in any one pixel of plain image should spread out to almost all pixels in the whole image. The diffusion process contains two steps. In the first step, the avalanche effect is introduced by using 2-D Pseudo Hadmard transform and the second step performs diffusion in two directions (forward and backward) with chaotic values, modulo addition and xor operations.

The 2-D Pseudo Hadmard transform (\textit{2-PHT}) is defined as,

\begin{align}
  y_1 &= (2x_1 + x_2) \mod 256 \\
  y_2 &= (x_1 + x_2) \mod 256
\end{align}  \tag{6, 7}

Where \(x_1\) and \(x_2\) are inputs to \textit{2-PHT} and \(y_1\) and \(y_2\) are outputs.

and its inverse is defined as,

\begin{align}
  x_1 &= (y_1 - y_2) \mod 256 \\
  x_2 &= (-y_1 + 2y_2) \mod 256
\end{align}  \tag{8, 9}

The avalanche effect is achieved by applying \textit{2-PHT} according to the Figure 1 on a block of eight pixels at a time and for the entire image.

![Figure 1. Application of 2-PHT for the Avalanche Effect](image)

In the second step, initially \(M \times N\) chaotic values \{\((X_1, ..., X_{M\times N}), (Y_1, ..., Y_{M\times N})\)\} are generated by using Eq. (1, 2) after doing iterations in chaos maps. The real chaotic sequences are transformed to integer form by using equation (10).

\[ z'_i = (z_i \times 10^8) \mod m \]  \tag{10}

Where \(z_i\) is real chaotic value, \(z'_i\) is transformed integer value, and \(m\) is 256 for 8-bit gray image.

The 2-D permuted image is transformed to 1-D array \(P_{1\times MN}\) by scanning the image left to right and top to bottom. Diffusion of the processed image is performed by using \(X\) and \(Y\) chaotic sequence of 2-D Zaslavskii map and also previously diffused pixels. The computed encrypted pixel values depends on the previously encrypted pixels and chaotic sequences, hence
algorithm shows resistance to the differential attacks such as known plain-text attack and known cipher-text attack.

The forward diffusion is performed by using following equation,

$$E_i = (((P_i + E_{i-2}) \mod 256) + E_{i-1}) \mod 256 \oplus X_i) \mod 256, i = 1, 2, ..., MN \quad (11)$$

Where + indicates modulo addition, $\oplus$ is bitwise XOR, $E_i$ is current pixel, $E_{i-1}$ and $E_{i-2}$ are previously encrypted pixels, $P_i$ is permuted pixels, $X_i$ and $Y_i$ are the 2-D Zaslavskii chaotic values. $E_{-1}$ and $E_0$ can be considered as constants.

The backward diffusion is performed using following equation to make the influence of every pixel equal.

$$F_i = (((E_i + F_{i+2}) \mod 256) + F_{i+1}) \mod 256 \oplus X_i) \mod 256,$$

$$i = MN, MN - 1, ..., 1 \quad (12)$$

Where $F_i$ is current pixel, $F_{i+1}$ and $F_{i+2}$ are previously encrypted pixels, $E_i$ is forward diffused image pixels, $X_i$ and $Y_i$ are the 2-D Zaslavskii chaotic values and $E_{MN+1}$ and $E_{MN+2}$ can be considered as constants. Finally, the encrypted image is obtained after the diffusions using Eq. (11) and Eq. (12) in two directions.

### 4.3. Encryption Algorithm

The encryption algorithm is composed of thirteen steps.

**Step 1.** Read the original image and store the pixel values in the matrix $I_{M \times N}$.

**Step 2.** Generate $M$ chaotic values of $X_i$ sequence $(X_1, ..., X_M)$, and $N$ chaotic values of $Y_i$ sequence $(Y_1, ..., Y_N)$ using Eq. (1, 2).

**Step 3.** Copy $X_i$ chaotic values to $P_M$ and $Y_i$ chaotic values to $P_N$.

**Step 4.** Sort $P_M$ and $P_N$, find the position of sorted chaotic values in the original chaotic sequence and store in $P_M'$ and $P_N'$.

**Step 5.** Scramble all the rows by using $P_M'$.

**Step 6.** Scramble all the columns by using $P_N'$.

**Step 7.** Apply 2-PHT to the permuted image using Eq. (6, 7) according to Figure 1.

**Step 8.** Transform 2-D processed image to 1-D array i.e. dimension transform from $M \times N$ to $1 \times MN$.

**Step 9.** Generate $M \times N$ chaotic values $\{(X_1, ..., X_{MN}) \}, (Y_1, ..., Y_{MN})\}$ using Eq. (1, 2).

**Step 10.** Transform real chaotic values to integers using Eq.(10).

**Step 11.** Perform forward diffusion using Eq. (11).

**Step 12.** Perform backward diffusion using Eq. (12).

**Step 13.** Transform the 1-D encrypted array to 2-D array i.e., dimension transform from $1 \times MN$ to $M \times N$.

### 4.4. Decryption

Decryption involves reconstructing gray levels of the original image from the encrypted image. It is a simple inverse process of the proposed encryption algorithm.
5. Experiments and Security Analysis

The proposed scheme is implemented on Linux platform using C language using a personal computer with an intel (R) Core(TM) i3-2120 CPU at 3.30 GHz with 2.91 GB of RAM. The initial parameters of the 2-D Zaslavskii map are randomly set to \( \{ X_0 = 0.65, Y_0 = 0.79, \nu = 12.6995, \varepsilon = 9.1, \tau = 3.0 \} \). The Test images are gray-scale images of size 256 × 256 chosen from USC-SIPI image database (sipi.usc.edu/database/). An image encryption scheme should resist the attacks such as brute-force attacks, statistical attacks, differential attacks and so on. This section analyzes the properties of the proposed encryption scheme to show its effectiveness in resisting these attacks.

The proposed encryption algorithm has been tested with several test images of differing content. Figure 2 shows the visual inspection of the original, encrypted and decrypted images of different images after applied only one round of encryption algorithm. The first row shows the original images, second row shows the encrypted images and the last row shows the decrypted images. The encrypted images are non-recognizable in appearance, unintelligible, incomprehensible, random and noise-like images without any leakage of the original information. This demonstrates that the proposed algorithm can be used to protect various images for diverse protection. The decrypted images are exactly same as the original images.

5.1. Histogram Analysis

The histograms present the statistical characteristics of images. An image histogram plots the frequency of occurrences of each gray level. An encrypted image is expected to have no statistical similarity with the original image to prevent the leakage of information. The histogram of original image consists of spikes with some shape. These spikes correspond to gray values that appear more often in the image. The histogram of encrypted image is expected to be sufficiently uniform to resist statistical attack. The histogram of several plain images are computed and analyzed. The histogram for Lena image is shown in Figure 3. The histogram of the encrypted image is uniformly distributed and is completely different from that of the original image, and bear no statistical resemblance to the original image. Hence the proposed algorithm is resistant to statistical attacks.

5.2. Key-space Analysis

The key-space of an encryption system should be sufficiently large enough to resist brute-force attacks. Brute-force attack is an attack where an opponent tries to break the cryptosystem by exhaustive search with all possible keys. The proposed encryption scheme has two initial values and three parameters, hence the key consists of totally five real values \( \{ X_0, Y_0, \nu , \varepsilon , \tau \} \).
Figure 2. Original Images, Encrypted Images and Decrypted Images with Proposed Algorithm (a) Lena (b) Mandrill (c) Aerial (d) Earth from Space (e) Pepper

Figure 3. Histograms of Original Image and Encrypted Image. (a) Original Image (b) Histogram of Original Image (c) Encrypted Image (d) Histogram of Encrypted Image

With 64 bits for each parameter and there are five real values, so the key-length is 320 bits and the key-space is $2^{320}$. Hence the proposed algorithm has adequate key-space and resistant to brute-force attacks. Table 1 shows the key-space size of the proposed algorithm and other algorithms.
Table 1. Key-space of the Proposed Method and Some of the other Methods in the Literature

<table>
<thead>
<tr>
<th>Encryption scheme</th>
<th>Proposed approach</th>
<th>Ref.[13]</th>
<th>Ref.[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-space size</td>
<td>$2^{220}$</td>
<td>$2^{128}$</td>
<td>$2^{256}$</td>
</tr>
</tbody>
</table>

5.3. Information Entropy Analysis

Information entropy is used to measure the amount of unpredictability in information content. Entropy tests whether an image is random kind of image with random distribution of pixel values. Entropy is defined as,

$$H(K) = \sum_{i=0}^{r-1} P(K_i) \log_2 \frac{1}{P(K_i)}$$

(13)

Where $K_i$ represents the pixel values, $P(K_i)$ is the probability of the symbol $K_i$ and $r$ is the number of symbols and is 256 for gray level image. Suppose the gray level image has $2^8$ gray values with equal probabilities, $K = (K_0, K_1, K_2, ..., K_{255})$, according to Eq. (13), we obtain its entropy value $H(K) = 8$. In the original plain image there exist correlation and the pixel values are seldom random, hence the entropy value is generally smaller than the ideal value 8. The entropy reaches the maximum ideal value of 8 when all pixel values are randomly distributed. Table 2 lists the entropy values for the original plain images and the encrypted images. From the results it is observed that the entropy of encrypted images are very close to the ideal value of 8. The information leakage in the proposed encryption scheme is negligible and is secure against the entropy based attacks. The comparison of entropy values with other approaches are listed in Table 3.

Table 2. Entropy Values for Original and Encrypted Images for Different Images

<table>
<thead>
<tr>
<th>Image</th>
<th>Entropy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>original image</td>
<td>encrypted image</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>7.426985</td>
<td>7.997620</td>
<td></td>
</tr>
<tr>
<td>Mandrill</td>
<td>7.242483</td>
<td>7.997126</td>
<td></td>
</tr>
<tr>
<td>Aerial</td>
<td>7.313656</td>
<td>7.997519</td>
<td></td>
</tr>
<tr>
<td>Earth</td>
<td>7.044457</td>
<td>7.997273</td>
<td></td>
</tr>
<tr>
<td>Pepper</td>
<td>7.577819</td>
<td>7.997282</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. The Entropy Analysis of the Proposed Scheme with other Methods for Lena Image

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy values</td>
<td>7.997620</td>
<td>7.9884</td>
<td>7.9812</td>
<td>7.9829</td>
<td>7.9923</td>
</tr>
</tbody>
</table>

5.4. Correlation Analysis

Generally, for any plain-image having visual content, each pixel is highly correlated with its adjacent pixels in all the three directions: horizontal, vertical and diagonal. A good encryption
scheme should produce encrypted images with no such correlations in the neighboring pixels. The correlation coefficient of adjacent pixels is calculated according to Eq. (14-17).

\[
E(x) = \frac{1}{N} \sum_{i=1}^{N} x_i
\]  

(14)

\[
D(x) = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(x))^2
\]  

(15)

\[
cov(x,y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(x))(y_i - E(y))
\]  

(16)

\[
r_{xy} = \frac{cov(x,y)}{\sqrt{D(x)D(y)}}
\]  

(17)

Where \(x\) and \(y\) are adjacent pixels of original or encrypted images, \(E(x)\) is the mean value, \(D(x)\) is the deviation with respect to mean, \(cov(x,y)\) is the covariance between adjacent pixels, and \(r_{xy}\) is the correlation coefficient. To test the correlation in the original and encrypted images, 2048 pairs of adjacent pixels are randomly chosen in horizontal, vertical and diagonal directions, and their correlation coefficients are calculated using Eq. (17). The Table 4 lists the computed correlation coefficients of original and encrypted images for different images. From Table 4 it can be seen that the two adjacent pixels in the original image are highly correlated to each other, whereas the correlation coefficients for encrypted images are very close to zero. Hence the proposed approach is resistant to statistical attacks. The comparison of correlation results with other methods are given in Table 5.

<table>
<thead>
<tr>
<th>Image</th>
<th>Correlation coefficients for original image</th>
<th>Correlation coefficients for encrypted images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal</td>
<td>Vertical</td>
</tr>
<tr>
<td>Lena</td>
<td>0.966883</td>
<td>0.943269</td>
</tr>
<tr>
<td>Mandrill</td>
<td>0.772042</td>
<td>0.845529</td>
</tr>
<tr>
<td>Aerial</td>
<td>0.842869</td>
<td>0.901867</td>
</tr>
<tr>
<td>Earth</td>
<td>0.944232</td>
<td>0.936486</td>
</tr>
<tr>
<td>Pepper</td>
<td>0.966785</td>
<td>0.960691</td>
</tr>
</tbody>
</table>

Table 5. The Correlation Analysis of the Proposed Scheme with other Methods for Lena Image

<table>
<thead>
<tr>
<th>Direction</th>
<th>Method</th>
<th>Plain-image</th>
<th>Proposed scheme</th>
<th>AES</th>
<th>Chen's</th>
<th>Arnold's</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>0.977352</td>
<td>-0.002689</td>
<td>-0.0160</td>
<td>0.0442</td>
<td>0.0787</td>
<td></td>
</tr>
<tr>
<td>Vertical</td>
<td>0.851794</td>
<td>-0.011126</td>
<td>0.8018</td>
<td>0.9728</td>
<td>-0.0793</td>
<td></td>
</tr>
<tr>
<td>Diagonal</td>
<td>0.761644</td>
<td>0.001347</td>
<td>-0.0140</td>
<td>0.0469</td>
<td>-0.0633</td>
<td></td>
</tr>
</tbody>
</table>

5.5. Gray Value Degree (GVD) Analysis

The gray difference of a pixel with its four neighbors in an image can be calculated as follows,

\[
\text{GVD}(x) = \sum_{i=1}^{N} (x_i - y_i)^2
\]
\[ G = \sum \frac{[I(m, n) - I(m', n')]^2}{4}, \text{here } (m', n') = \begin{cases} (m - 1, n) \\ (m + 1, n) \\ (m, n - 1) \\ (m, n + 1) \end{cases} \] (18)

Where \( I(m, n) \) represents the pixel value at location \((m, n)\), and \( I(m', n') \) denotes the pixel values of four neighborhood pixels. The average neighborhood gray difference for the complete image can be calculated by Eq. (19).

\[ W(G(m, n)) = \frac{\sum_{m=2}^{M-1} \sum_{n=2}^{N-1} G(m, n)}{(M - 2) \times (N - 2)} \] (19)

Where \( M \) and \( N \) are the number of rows and columns of the image. By using (18) and (19) the gray value degree is defined as,

\[ GVD = \frac{W(G(m, n)) - W(G(m, n))}{W(G(m, n)) + W(G(m, n))} \] (20)

Where \( W' \) and \( W \) denote the average neighborhood gray difference of original plain image and encrypted image. Table 6 lists computed gray value degree values for different images by the proposed approach. From Table 6 it is observed that the gray value degrees computed by the proposed method are close to the ideal value of 1. Table 7 shows the comparison of GVD with other approaches.

<table>
<thead>
<tr>
<th>Image</th>
<th>GVD value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>0.962362</td>
</tr>
<tr>
<td>Mandrill</td>
<td>0.903770</td>
</tr>
<tr>
<td>Aerial</td>
<td>0.917014</td>
</tr>
<tr>
<td>Earth</td>
<td>0.958430</td>
</tr>
<tr>
<td>Pepper</td>
<td>0.963746</td>
</tr>
</tbody>
</table>

Table 7. The GVD Analysis of the Proposed Scheme with other Approaches

<table>
<thead>
<tr>
<th>Image</th>
<th>GVD value</th>
<th>Proposed approach</th>
<th>Ref.[13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>0.962362</td>
<td>0.89</td>
<td>0.954</td>
</tr>
</tbody>
</table>

5.6. Peak Signal to Noise Ratio (PSNR) Analysis

The PSNR can be used to perform objective evaluation of the encryption methods. The PSNR is computed by considering the original plain image as signal and the encrypted image as a noise. The PSNR can be calculated by using the following formula,

\[ PSNR = 20 \times \log_{10} \left( \frac{255}{\sqrt{MSE}} \right) \text{dB} \] (21)

Where \( MSE \) is mean square error and is computed according to Eq. (22)

\[ MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (|I(i,j) - I'(i,j)|)^2 \] (22)
Where $I(i, j)$ is pixel value of original plain image and $I'(i, j)$ is pixel values of encrypted image at location $(i, j)$. The calculated PSNR values for different test images are listed in Table 8. From Table 8 it is observed that the PSNR values are low, which indicates the difficulty in getting original plain image from the encrypted image for attackers.

### Table 8. The PSNR Values for Different Test Images

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>9.227133</td>
</tr>
<tr>
<td>Mandrill</td>
<td>9.707395</td>
</tr>
<tr>
<td>Aerial</td>
<td>9.282019</td>
</tr>
<tr>
<td>Earth</td>
<td>9.402777</td>
</tr>
<tr>
<td>Pepper</td>
<td>8.890136</td>
</tr>
</tbody>
</table>

#### 5.7. Key Sensitivity Analysis

Key sensitivity implies that the small change in the secret key should produce entirely different encrypted image. The key sensitivity test is conducted by using following steps.

**Step 1.** The original plain image is encrypted by using a test key $K_1$ to produce cipher image $C_1$.

**Step 2.** The original plain image is encrypted again with a small change in the test key $K_1$, i.e., $K_2$ to produce cipher image $C_2$.

**Step 3.** The two cipher images $C_1$ and $C_2$ with slightly different keys are compared pixel by pixel to observe the number of differing pixels.

The two parameters $NPCR$ and $UACI$ are used to assess the key sensitivity and are discussed below.

$NPCR$ (Number of Pixels Change Rate) is used to compute the total number of different pixels in two images and is calculated by using Eq. (23).

$$NPCR = \frac{\sum_{i,j} D(i, j)}{M \times N} \times 100\%$$  \hspace{1cm} (23)

$$D(i, j) = \begin{cases} 1, & \text{if } C_1(i, j) \neq C_2(i, j) \\ 0, & \text{Otherwise} \end{cases}$$  \hspace{1cm} (24)

Where $C_1$ and $C_2$ are two encrypted images with slightly different keys $K_1$ and $K_2$. $C_1(i, j)$ and $C_2(i, j)$ are the pixel values of $C_1$ and $C_2$ at location $(i, j)$. $D$ is a bipolar array with the same size as $C_1$ and $C_2$ and its contents are either 0 or 1 based on Eq. (24).

$UACI$ (Unified Average Changing Intensity) is used to compute the average intensity difference between two encrypted images and is given by,

$$UACI = \frac{1}{M \times N} \left[ \sum_{ij} \frac{C_1(i, j) - C_2(i, j)}{255} \right] \times 100\%$$  \hspace{1cm} (25)

The key sensitivity is analyzed by testing one parameter at a time with a very small change in the key. The proposed scheme has five parameters $\{X_0, Y_0, \nu, \varepsilon, \tau\}$. Table 9 lists the $NPCR$ and $UACI$ values for five different parameters. From Table 9 it can be seen that the $NPCR$ and $UACI$ values are close to their ideal values of 99.6% and 33.4%. Hence the proposed scheme has high key sensitivity.
Table 9. Key Sensitivity Results for Lena Image with Different Parameters of the Chaotic Map

<table>
<thead>
<tr>
<th>Parameter changed</th>
<th>NPCR (%)</th>
<th>UACI (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>99.645996</td>
<td>33.375423</td>
</tr>
<tr>
<td>Y</td>
<td>99.615479</td>
<td>33.457809</td>
</tr>
<tr>
<td>ν</td>
<td>99.603271</td>
<td>33.551277</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>99.630737</td>
<td>33.457211</td>
</tr>
<tr>
<td>$\tau$</td>
<td>99.617004</td>
<td>33.406235</td>
</tr>
</tbody>
</table>

Key sensitivity is also tested pictorially with the following approach. The original key is altered with a small change and a different key is generated. The keys can be expressed as, original key $K_1 = (0.65, 0.79, 12.6695, 9.1, 3.0)$, and the slightly modified key $K_2 = (0.65000000000001, 0.79, 12.6695, 9.1, 3.0)$. The two encrypted images ($C_1$ and $C_2$) with slightly different keys ($K_1$ and $K_2$) are shown in figure 4b-c respectively. Even though both look similar, they are almost different from each other. This can be observed by finding the difference image between $C_1$ and $C_2$. Figure 4d shows the difference image between $C_1$ and $C_2$. It is observed that most of the pixels in difference image are nonzero, hence the difference is big enough.

Figure 4. Key Sensitivity Analysis for Encryption Process for Lena Image (a) Original Image (b) Encrypted Image with Correct Key $K_1$ (c) Encrypted Image with Slightly Different Key $K_2$ (d) Difference Image between $C_1$ and $C_2$

The key sensitivity test is also analyzed for decryption process. The decryption is performed with correct key and slightly different keys. Figure 5a shows the decrypted image with correct key $K_1 = (0.65, 0.79, 12.6695, 9.1, 3.0)$ and Figure 5b-c are decrypted images with slightly altered keys $K_2 = (0.65000000000001, 0.79, 12.6695, 9.1, 3.0)$ and $K_3 = (0.65, 0.79, 12.6695, 9.1000000000001, 3.0)$. Hence, if there is a small change in the key, the correct decryption cannot be achieved.

5.8. Plain-image Sensitivity Analysis

To resist differential attacks, the encryption algorithms should satisfy the plain image sensitivity condition, i.e., a small change in the original plain-image should cause a significant change in the encrypted image. The NPCR and UACI parameters are used to test the plain-image sensitivity as given in Eq. (23-25). The NPCR and UACI values are computed for different randomly chosen locations by changing one pixel at a time and the results are listed in Table 10. From Table 10, it is observed that the NPCR and UACI values are close to their ideal values of 99.6% and 33.4% irrespective of the pixel position. Hence the proposed approach has good sensitivity to plain-images.
Figure 5. Key Sensitivity Analysis for Decryption Process for Lena Image. (a) Decryption with Correct Key (b-c) Decryption with Slightly Changed Keys

Table 10. Plain-image Sensitivity Test at Different Positions for Lena Image

<table>
<thead>
<tr>
<th>Position</th>
<th>NPCR (%)</th>
<th>UACI (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>99.752808</td>
<td>33.499668</td>
</tr>
<tr>
<td>(35,80)</td>
<td>99.673462</td>
<td>33.456978</td>
</tr>
<tr>
<td>(100,150)</td>
<td>99.633789</td>
<td>33.421177</td>
</tr>
<tr>
<td>(128,128)</td>
<td>99.591064</td>
<td>33.421783</td>
</tr>
<tr>
<td>(255,255)</td>
<td>99.586296</td>
<td>33.377789</td>
</tr>
</tbody>
</table>

5.9. Computational Speed Analysis

The proposed approach makes use of simple operations such as addition, subtraction and xor, and these are not computationally intensive. Hence the approach can offer a fast and efficient way for image encryption. The complexity of the proposed method is $O(M \times N)$, where $M$ and $N$ are the number of rows and columns of the image. The time needed to encrypt 256×256 gray scale image is 3.43 micro-seconds and for decryption it is the same. So the proposed scheme can encrypt 19106 Mb data per second. The comparisons of the encryption time with other approaches are listed in Table 11.

Table 11. Execution Time Analysis using a 256x256 Image

<table>
<thead>
<tr>
<th>Methods</th>
<th>Encryption time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guodong Ye</td>
<td>0.150 seconds</td>
</tr>
<tr>
<td>Gao. T.G</td>
<td>0.633 seconds</td>
</tr>
<tr>
<td>Ye.R.S</td>
<td>&gt;10 seconds</td>
</tr>
<tr>
<td>Huang X.L</td>
<td>0.547 seconds</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>0.000000343 seconds</td>
</tr>
</tbody>
</table>

6. Conclusions

In this paper, a novel approach for image encryption based on 2-D Zaslavskii map and pseudo Hadmard transform is proposed. The proposed method offer high security and high speed. It is implemented under Linux platform with C language, and the achieved speed is 3.43 μs, hence it is computationally efficient. The proposed approach has key space of $2^{210}$, which is large enough to prevent brute-force attacks. The average entropy achieved is 7.997364, which is close to the ideal value of 8, and hence the information leakage is negligible. The NPCR and UACI values are close to their ideal values of 99.6% and 33.4% for key sensitivity and plain image sensitivity tests and hence resistant to differential attacks. The correlations are close to zero and the histogram is almost uniformly distributed, so statistical attacks are resisted. The GVD is near to 1 and the PSNR is lower. The results shown in section 5 are the obtained results.
after one round of encryption/decryption. So it is observed that even in the first round all the security parameters are already high. The proposed approach can be extended for color images.

References


Authors

**Linganagouda Kulakarni**, he received PhD degree in pattern recognition from Mysore university, India. His research interests are image processing, computer networks and information security. He is currently working as professor at computer science department, BVB college of engineering and technology, Hubli, India.

**Gururaj Hanchinamani**, he received ME degree in computer science and engineering from Walchand college of engineering Sangli, India. He is pursuing PhD degree at Visvesvaraiyah technological university Belgaum. His research interests are information security and computer architectures. He is currently working as associate professor at computer science department, BVB college of engineering and technology, Hubli, India.