Algorithms for Encrypting Images using SEE Transformation

1Mohamed Yahay, 2Najlae Falah Hameed Al Saffar
1Department of Mathematics, 
2Faculty of Computer Science & Mathematics, University of Kufa. Iraq

Abstract—Nowadays, images play a big part in many applications, thus it’s important to safeguard sensitive data from unwanted access or even manipulation. One of the finest ways to accomplish this goal is through encryption. Researchers have developed numerous methods for image encryption in recent years. To boost security, they employ different concepts to be tools for image encryption. In this work SEE transformation will be a tool to encrypt images for the 1st proposed algorithm. RSA public key cryptosystem will be involved together with SEE transformation for the 2nd proposed algorithms in two cases to make the transform of image is more efficient. Mean square error, peak signal to noise ratio and unified average changing intensity were used to test the proposed algorithms, demonstrating their speed and high level of security.

Index Terms—Image Encryption, SEE Transformation, RSA Public Key Cryptography, MSE, PSNR and UACI.

I. INTRODUCTION (HEADING I)

Cryptography is a tool to store and transmit information in a validated structure, in order for a client to read and process it in secret ways. It includes encryption and decryption of messages. The most common way of changing over plain information into cipher is known as encryption, while the method involved with recuperating the plain information from encoded one is known as decryption [1]. Cryptography has a wide range of applications, protecting the images is one of them. In fact, transmission of images across open networks is a necessary activity. Consequently, it is crucial to share confidential information in a secure manner. Different tools were used to increase the level of security such like [2] with two proposed algorithms and [3].

In this work See transformation [4] will be involved to be a tool to increasing the level of security for transform in image. For the 1st proposed algorithm. In the rest algorithms, RSA [5] public key cryptosystem will be used to make the transform of image is more efficient.

SEE transform [4] was created in 2021, which takes its name from Sadiq, Emad and Eman, is an integral transform that solved linear ordinary and partial differential equations with constant coefficients. In the same year, the same authors [6] derived the formula for the complex SEE and applied it in solving Laplace transformation problem. Also they trying to solve linear system of ordinary differential equations in the time domain [7]. In 2021 and 2022 the same authors [8] and [9] used SEE and complex SEE transformation as a tool for public key cryptosystem and applied them in text encryption and image encryption respectively. This why this work will focus on using SEE transform to be a tool to encrypt and decrypt images.

In order to offer confidentiality for transforming images, this work will introduce image encryption algorithms using the properties of SEE transform together with public key cryptosystem. Mean square error (MSE) Peak Signal to Noise Ratio (PSNR) and Unified Average Changing Intensity (UACI) security measures will be used to gauge how effectively these algorithms perform.

II. SEE TRANSFORM

Sadiq, Emad and Eman (SEE) [4] are derived from the Laplace transformation. SEE integral was investigated to work differential conditions in the time area. It is change the characterized for capacity of outstanding request, where the capacities in the set A characterized by the following:

\[ A = \{ F(T) : \text{there exist } M, l_1, l_2 > 0, |f(t)| < M e^{lt}, \text{if } t \in (-1)^n \times [0, \infty) \} \]  

For a given capacity in the arrangement of A, the constant M have to be bounded, while \( l_1 \) and \( l_2 \) need not to be might be limited. SEE transformation has been defined by the following:

\[ S(f(t)) = T(v) = \frac{1}{\pi v} \int_0^{\infty} f(t) e^{-vt} dt = T(s) t \geq 0, n \in Z^+, l_1 \leq v \leq l_2 \]  

The variable \( v \) in this definition is used to aspect the variable \( t \). Invers of SEE transformation is defined to be \( S^{-1}(T(v)) = f(t) \).

III. RSA PUBLIC KEY CRYPTOSYSTEM

In 1976 Diffie and Hellman [10] have been introduced any new type of cryptography which opened the door for the more significant invention, it was the RSA public key cryptosystem by Rivest, Shamir, and Adleman [5] in 1978. An explanation of this system will be brief within three levels: Key Generation, Encryption and Decryption level for any two users A and B.

First Level: Key Creation (User B)
- Select secret primes \( p \) and \( q \).
- Select public key \( e \) with \( \gcd(e, (p - 1)(q - 1)) = 1 \).
- Publish \( N = p \cdot q \) and \( e \).

Second Level: Encryption (User A)
- Select plaintext \( M \).
- Calculate \( C \equiv M^e \mod N \).
- Send cipher text \( C \) to B.

Third level: Decryption (User B)
- Calculate \( d \) where \( ed \equiv 1 \mod (p - 1)(q - 1) \).
- Calculate the plain text $M \equiv C^d \mod N$.

## IV. PROPOSED ALGORITHMS

As mentioned before a wider range of areas, including mathematics, physics, and electrical engineering, use the SEE transform. In this work, two SEE transform-based encryption and decryption images are proposed. The further usage of these algorithms is carried forward in MATLAB R2020a (8.3.0.532) 64-bit software on a work station Intel® Core™ i7- 10510U with CPU 1.80 GHz and RAM 8 GB with Microsoft Windows 11.

### First Algorithm

Suppose that two users A and B went use this algorithm to a message, they have pass those levels:

#### First Level: Encryption Steps: (A)

1. Step 1: choose the image that need to encrypt.
2. Step 2: let the $G_i$ represent pixels image, where $i$ is the length of image.
3. Step 3: use the trigonometric function Taylor series of $\cos x$,
   \[
   \cos x = 1 + \frac{x^2}{2} - \frac{x^4}{4!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{2n!}.
   \] (1)
4. Step 4: multiplication $G_i$ by Taylor trigonometric function as:
   \[
   G_i \cos x = G_i 1 + \frac{G_i x^2}{2} - \frac{G_i x^4}{4!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n G_i x^{2n}}{2n!}.
   \] (2)
5. Step 5: taking SEE transform for (2) as:
   \[
   S(G_i \cos x) = S \left( G_i 1 + \frac{G_i x^2}{2} - \frac{G_i x^4}{4!} + \cdots \right) = S \left( \sum_{n=0}^{\infty} \frac{(-1)^n G_i x^{2n}}{2n!} \right) = G_i + \frac{G_i x^2}{2!} - \frac{G_i x^4}{4!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n G_i x^{2n}}{2n!}.
   \] (3)
6. Step 6: divide all numbers in (3) by 256 and store the reminder as matrix.
7. Step 7: the output matrix will be the encryption image.

#### Second Level: Key generation: (User A)

Consider
\[
 x_i = 256 k_i + r_i
\] (4)
where $x_i$ are the number of the result of (3), $r_i$ are the reminder when divided $x_i$ by 256 , $k_i$ are the number of key that need to generate and $i$ is the length of image. So, there are $k_0, k_1, ..., k_i$ of keys according to solving the following system:
\[
\begin{align*}
  x_0 &= 256 k_0 + r_0 \\
  x_1 &= 256 k_1 + r_1 \\
  &\vdots \\
  x_i &= 256 k_i + r_i
\end{align*}
\]

#### Third Level: Decryption Steps: (B)

1. Step 1: converting the encrypted image to the numbers.
2. Step 2: let the $G'_i$ represent these numbers image and $i$ the length of image.
3. Step 3: solve the following system:
   \[
   \begin{align*}
   y_0 &= 256 k_0 + G'_0 \\
   y_1 &= 256 k_1 + G'_1 \\
    &\vdots \\
   y_i &= 256 k_i + G'_i
   \end{align*}
   \]

To get the values of $y_0, y_1, ..., y_i$
4. Step 4: calculating the invers of SEE of $y_i$ multiply by trigonometric function Taylor series of $\cos x$:
   \[
   S^{-1} \left\{ y_0 + \frac{y_1 x^2}{2!} - \frac{y_2 x^4}{4!} + \cdots \right\} = S^{-1} \left\{ \sum_{n=0}^{\infty} \frac{(-1)^n y_i x^{2n}}{2n!} \right\} = y_0 + \frac{y_1 x^2}{2!} - \frac{y_2 x^4}{4!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n y_i x^{2n}}{2n!}.
   \]
5. Step 5: store the numerator integer value as matrix.
6. Step 6: the output matrix will be the plain image

### Second and Third Algorithm

Making a combination of the RSA algorithm and the SEE transform is an efficient approach to encrypt the image which is a scheme that has been introduced in this section with two cases, where the RSA will be involved to encrypt the ciphered image for the 1st case while the 2nd case, the RSA will be implemented for the plain image.

#### V. IMPLEMENTATIONS

In this section, the results that have been gotten appeared the significance of the proposed algorithms. Lena image of size 256 × 256 is considered a plain image.

First Algorithm: A wants to send a Lena image to B
First Level: Encryption Steps: (A)

Choose “Lena” as plain image:
She separates the pixel values of “Lena 256 × 256” into blocks of size four as:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

Converting a block to a single row from i = 1 to 65536 for example we choose the vertex of i from 1 to 8:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Taylor for \(\cos x = 1 + \frac{x^2}{2!} - \frac{x^4}{4!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{2n!} = G_0 + \frac{G_1 x^2}{2!} - \frac{G_2 x^4}{4!} + \cdots = \sum_{n=0}^{\infty} (-1)^n G_n x^{2n},\)
\(G_0 = 250, G_1 = 250, G_2 = 251, G_3 = 252, G_4 = 253, G_5 = 254, G_6 = 255, G_7 = 255.\)

Taking SEE transform from the series
\(S[G_i \cos x] = S\{250 + \frac{250x^2}{2!} - \frac{252x^4}{4!} + \frac{253x^6}{6!} + \cdots\} = \left(250 + \frac{1}{v^2} \frac{1000}{\mu^6} + \frac{144576}{\mu^8} + \frac{130636800}{\mu^{10}} + \frac{411302707200}{\mu^{12}} + \cdots\right).\)

Taking mod 256 for the numbers in the numerators

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

The output is the Encryption image. Reset the block to 256*256

Converting the encrypted image to matrix.

Second Level: Key generation: (A)

A will do key generation steps for i = 1 to 65536

250 = 256 k_4 + 250, k_4 = 0,
1000 = 256 k_3 + 232, k_3 = 3,
144576 = 256 k_2 + 192, k_2 = 564,
130636800 = 256 k_1 + 0, k_1 = 510300,
411302707200 = 256 k_4 + 0, k_4 = 1606651200,

So the keys are \(\{0, 3, 564, 510300, 1606651200, \ldots\}.\)

Third Level: Decryption Steps: (B)

Converting the encrypted image to matrix.
Let the \(G'_i\) represent these numbers image and i the length of image.

Solve the following system:
\(y_1 = 256 k_4 + G'_1,\)
\(y_0 = 256 \times 0 + 250 = 250,\)
\(y_1 = 256 \times 3 + 232 = 1000,\)
\(y_2 = 256 \times 564 + 192 = 144576,\)
\(y_3 = 256 \times 510300 + 0 = 130636800,\)
\(y_4 = 256 \times 1606651200 + 0 = 411302707200,\)

So "Lena" as plain image.
Calculating the invers of SEE of \( y_i \) multiply by trigonometric function Taylor series of \( \cos x \):

\[
S^{-1}\{250 + \frac{1}{2!} + \frac{1000}{4!} + \frac{144576}{6!} + \frac{41130270200}{10!} + \cdots\} = \{250 + \frac{250x^2}{2!} - \frac{251x^4}{4!} + \frac{252x^6}{6!} - \frac{253x^8}{8!} + \frac{254x^{10}}{10!} - \frac{255x^{12}}{12!} + \frac{255x^{14}}{14!} - \cdots\},
\]

\( = \{G_0 = 250, G_1 = 250, G_2 = 251, G_3 = 252, G_4 = 253, G_5 = 254, G_6 = 255, G_7 = 255 \ldots\}. \)

The block of output:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 65536 \\
1 & 250 & 250 & 251 & 252 & 253 & 254 & 255 & 255 & \ldots
\end{array}
\]

Reset the block to 256*256:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 250 & 250 & 251 & 249 & 255 & 255 & 25 & \ldots
\end{array}
\]

Fig 1. will summarize the output of execution of this implementation.

![Image](original_image.png)

**Figure 1: Implementation the Lena (256x256) image for the 1st proposed algorithm**

**Second Algorithm: A wants to send a Lena image to B**

**First Case: Encryption by SEE and RSA**

A separates the pixel values of “Lena 256x256” into blocks of size four as:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 250 & 255 & 255 & 251 & 249 & 255 & 255 & 25 & \ldots
\end{array}
\]

Encode images as in the 1st proposed algorithm using SEE:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 250 & 0 & 0 & 0 & 0 & 0 & 0 & 220 & 0 & \ldots
\end{array}
\]

Then she will encrypt the output with RSA, she will choose prime numbers whose product must be greater than 256, for example 17 and 19 whose product is 323:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots
\end{array}
\]
A will generate keys as:

\[
250 = 256 k_0 + 250, k_0 = 0,
1000 = 256 k_1 + 232, k_1 = 3,
144576 = 256 k_2 + 192, k_2 = 564,
130636800 = 256 k_3 + 0, k_3 = 510300,
411302707200 = 256 k_4 + 0, k_4 = 1606651200.
\]

Now, B will receive the output, then decrypt it according to RSA algorithm:

\[
1 2 3 4 5 6 7 8 \ldots 256
1 250 0 0 0 0 0 220 0 \ldots
2 232 0 0 0 0 0 192 0 \ldots
3 192 0 0 0 0 0 0 0 \ldots
4 0 0 0 0 0 0 0 0 \ldots
\]

As in the 1st algorithm:

\[
1 2 3 4 5 6 7 8 \ldots 256
1 250 255 255 251 249 255 255 255 \ldots
2 250 255 255 251 250 255 255 254 \ldots
3 251 255 255 251 215 255 255 249 \ldots
4 252 255 255 251 253 255 253 242 \ldots
\]

Fig 2. will summarize the output of execution of this implementation.

**Second Algorithm: A wants to send a Lena image to B**

**Second Case: Encryption by RSA and SEE**

Firstly, A will encrypt image using RSA and then encryption using SEE:

\[
1 2 3 4 5 6 7 8 \ldots 256
1 250 255 255 251 249 255 255 255 \ldots
2 250 255 255 251 250 255 255 254 \ldots
3 251 255 255 251 251 255 255 249 \ldots
4 252 255 255 251 253 255 253 242 \ldots
\]
She chooses two prime 17 and 19 for RSA, the result will be as follows:

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 300 & 221 & 221 & 55 & 260 & 221 & 221 & 221 & \ldots \\
2 & 300 & 221 & 221 & 55 & 300 & 221 & 221 & 220 & \ldots \\
3 & 55 & 221 & 221 & 55 & 55 & 221 & 221 & 260 & \ldots \\
4 & 313 & 221 & 221 & 55 & 43 & 221 & 43 & 200 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ldots \\
256 & & & & & & & & & \\
\end{array}
\]

Then encrypt the output of RSA using SEE. According to the 1st proposed algorithm:

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 300 & 283 & 97 & 39 & 36 & 31 & 204 & 296 & \ldots \\
2 & 231 & 198 & 66 & 273 & 230 & 244 & 204 & 133 & \ldots \\
3 & 26 & 170 & 0 & 6 & 197 & 97 & 34 & 174 & \ldots \\
4 & 150 & 62 & 111 & 236 & 28 & 147 & 138 & 230 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ldots \\
256 & & & & & & & & & \\
\end{array}
\]

**Key generation: (A)**

As a generation key steps: as the 1st proposed algorithm, but here the difference is that the mod is the value of n=323.

**Decryption: (B)**

As in the 1st proposed algorithm from SEE:

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 300 & 221 & 221 & 55 & 260 & 221 & 221 & 221 & \ldots \\
2 & 300 & 221 & 221 & 55 & 300 & 221 & 221 & 220 & \ldots \\
3 & 55 & 221 & 221 & 55 & 55 & 221 & 221 & 260 & \ldots \\
4 & 313 & 221 & 221 & 55 & 43 & 221 & 43 & 200 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ldots \\
256 & & & & & & & & & \\
\end{array}
\]

Applying RSA algorithm for decryption as:

\[
\begin{array}{ccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \ldots & 256 \\
1 & 250 & 255 & 255 & 251 & 249 & 255 & 255 & 255 & \ldots \\
2 & 250 & 255 & 255 & 251 & 250 & 255 & 255 & 254 & \ldots \\
3 & 251 & 255 & 255 & 251 & 251 & 255 & 255 & 249 & \ldots \\
4 & 252 & 255 & 255 & 251 & 253 & 255 & 253 & 242 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ldots \\
256 & & & & & & & & & \\
\end{array}
\]

Fig 3. will summarize the output of execution of this implementation.

**Figure 3. Implementation the Lena (256×256) Image for the 2nd Proposed Algorithm/ 2nd Case**

**SECURITY EXAMINATION**

There are three measures used to assess encrypted images and compare them with plain images to calculate the performance of the proposed algorithms.
Mean Square Error and Peak Signal-to-Noise Ratio: Mean square error (MSE) and peak signal-to-noise ratio (PSNR) are efficient tools to measure the efficiency of image encryption algorithms. So, employing MSE and PSNR is a good way to evaluate the quality of plain images with respect to the encrypted images. The equation of the PSNR is as:  
$$ \text{PSNR} = 10 \log_{10} \frac{255^2 \times 255}{\text{MSE}} $$  
where \( \text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (X(i,j) - Y(i,j))^2 \). \( X(i,j) \) and \( Y(i,j) \) are the pixel value of plain image and encrypted image respectively both have size \( m \times n \). Obviously, that calculating the MSE will affect the value of PSNR, if PSNR is decreasing means that the MSE is increasing, in other words the high value of MSE and low value of PSNR indicates that the two images are totally different, and this leads to the efficiency of the proposed algorithms.

In this work, the upsides of the MSE were 242.0946, 242.0946 and 82.7409 for 1st proposed algorithm, 2nd proposed algorithm/ 1st Case and 2nd proposed algorithm/ 2nd Case respectively, while PSNR were 24.2910, 24.2910 and 28.9536 for 1st proposed algorithm, 2nd proposed algorithm/ 1st Case and 2nd proposed algorithm/ 2nd Case respectively for the Lena as a plain image, it is so difficult for an aggressor to recuperate the plain image.

Unified Average Changing Intensity (UACI): It is one of differential investigations used to assess the strength of image encryption, where it is assessed the differentiation between the encrypted image and plain image [3], [9]. The highest value of the UACI implies that the proposed algorithms are safe against differential assaults. The condition of the UACI is as:  
$$ \text{UACI} = \frac{1}{256^2} \sum_{i=1}^{256} \sum_{j=1}^{256} \frac{|X(i,j) - Y(i,j)|}{255} \times 100\% $$  
\( X(i,j) \) and \( Y(i,j) \) are the pixel value of plain image and encrypted image respectively.

In this work, the worth of the UACI were 42.2457, 42.2457 and 9.8888 for 1st proposed algorithm, 2nd proposed algorithm/ 1st Case and 2nd proposed algorithm/ 2nd Case respectively of the encrypted image with respect to Lena as a plain image, it is so hard for an attacker to recover the plain image.

Table 1 The performance of the proposed algorithms using various security measures for different image with size 256 × 256

<table>
<thead>
<tr>
<th>Plain image Name</th>
<th>Measures</th>
<th>1st Algorithm</th>
<th>2nd Algorithm/ 1st case</th>
<th>2nd Algorithm/ 2nd case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>MSE</td>
<td>242.0946</td>
<td>242.0946</td>
<td>82.7409</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>24.2910</td>
<td>24.2910</td>
<td>28.9536</td>
</tr>
<tr>
<td></td>
<td>UACI</td>
<td>42.2457</td>
<td>42.2457</td>
<td>9.8888</td>
</tr>
<tr>
<td>Peppers</td>
<td>MSE</td>
<td>244.3767</td>
<td>244.3767</td>
<td>110.0144</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>24.2502</td>
<td>24.2502</td>
<td>27.7163</td>
</tr>
<tr>
<td></td>
<td>UACI</td>
<td>55.8598</td>
<td>55.8598</td>
<td>14.8896</td>
</tr>
<tr>
<td>Cameraman</td>
<td>MSE</td>
<td>226.9854</td>
<td>226.9854</td>
<td>87.0020</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>24.5708</td>
<td>24.5708</td>
<td>28.7355</td>
</tr>
<tr>
<td></td>
<td>UACI</td>
<td>44.0755</td>
<td>44.0755</td>
<td>10.9918</td>
</tr>
<tr>
<td>Baboon</td>
<td>MSE</td>
<td>243.4494</td>
<td>243.4494</td>
<td>94.0968</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>24.2667</td>
<td>24.2667</td>
<td>28.3951</td>
</tr>
<tr>
<td></td>
<td>UACI</td>
<td>48.2092</td>
<td>48.2092</td>
<td>11.1703</td>
</tr>
</tbody>
</table>

Table 2 Comparison of the proposed algorithm with other methods over security measures for different image with size 256 × 256

<table>
<thead>
<tr>
<th>Lena</th>
<th>peppers</th>
<th>Cameraman</th>
<th>Baboon</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>PSNR</td>
<td>UACI</td>
<td>MSE</td>
</tr>
<tr>
<td>242.0946</td>
<td>24.2910</td>
<td>42.2457</td>
<td>244.3767</td>
</tr>
</tbody>
</table>
Table 3 Encryption and Decryption time for the proposed scheme for different gray image with size 256 × 256.

<table>
<thead>
<tr>
<th>Case</th>
<th>Image Name</th>
<th>Encryption/ and Decryption time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Najlae [1]</td>
<td>Lena</td>
<td>7.697</td>
</tr>
<tr>
<td>Zahrue [12]</td>
<td>Cameraman</td>
<td>10.939s</td>
</tr>
<tr>
<td></td>
<td>Mandrill</td>
<td>20.391s</td>
</tr>
</tbody>
</table>

According to Table 1, the values of PSNR were very low while the values of MSE were very high, which means that the proposed scheme is an efficient algorithm. The values of UACI for the selected images were almost close to the expected value of 33.46, and this demonstrates the strength of the proposed algorithms against the attackers. These results actually showed that the encrypted images were barely perceptible, which serves as the foundation for a strong encryption technique.

In Table 2, a comparison of the proposed method and a few other recent algorithms regarding security precautions for various gray images with size 256 × 256 is presented. Thus, it can be said that the proposed algorithms is more effective than the alternatives.

Finally, table 3 summarized the time consumed in encryption and decryption processes over the implementing images considered in this work. The time it takes for each step of the proposed algorithms to complete is used to gauge how effective they are; table 3 demonstrates how quickly the proposed scheme can be implemented.

REFERENCES