A Simple, Accurate and Highly Secure Method to Encrypt-Decrypt Digital Images

Jamil Al-Azzeh#, Ziad Alqadi#, Qazem Jaber#

# Computer Engineering Department, Al Balqa’ Applied university, Amman, 11134, Jordan
E-mail: azzehjamil@gmail.com, natalia_mav@yahoo.com, qazemjaber@gmail.com

Abstract— The digital image may be important and has a secret character, which requires not understanding it when looking at the naked eye or not understanding the contents. So seeking a method of digital image encryption-decryption is a very important task. In this paper we will introduce a new method of digital image encryption-decryption, which will be very simple, highly secure and accurate and highly efficient.

Keywords— Encryption, decryption, private key, speedup, throughput.

I. INTRODUCTION
Digital image encryption is the process of encoding an image in such a way that only authorized parties can access it and those who are not authorized cannot. The decryption process is to return back the original image without losing any piece of information from the original image.

Digital color images [1-33] are one of the most important types of data currently in the process of messaging through the Internet, which leads us to resort to the use of multiple ways to protect them from parasitism. The digital image may be important and has a secret character, which requires not understanding it when looking at the naked eye or not understanding the contents [3]. In order to do this, we must use a safe and efficient way to encrypt and re-encrypt them so that we can obtain a new image that matches the original image as shown in figure (1).

![Fig 1. Encrypted and decrypted color images](image_url)

Digital images are treatment (and here encryption=decryption) is different from text encryption-decryption due to some valuable features of the digital image, such as bulk data capacity and high correlation among pixels. [4], [5], [6].

In order to solve the problem of image encryption-decryption, we introduced a simple one key which can be used to encrypt-decrypt any image (binary, gray color) with any size.

II. THE MATERIAL AND METHOD / ALGORITHM
The digital image may be important and has a secret character, which requires not understanding it when looking at the naked eye or not understanding the contents, many different digital image encryption-decryption methods and techniques have been investigated tested and proposed for enhancing the security of images. In [7] an encryption technique for encryption-decryption using the Hill cipher method was proposed. In [8] a comparative analysis was introduced and different methods of image encryption-decryption were tested and compared.

In [9] a New Chaotic Algorithm for Image Encryption-decryption was proposed this method was tested and implemented and it gave a 0.5 second encryption time to encrypt an RGB color image with size 256x256x3.

In [10] A Symmetric Image Encryption Scheme based on 3D Chaotic Cat Maps was proposed this method was tested and implemented and it gave a 0.4 second encryption time to encrypt an RGB color image with size 256x256x3.

In [11] An Image Scrambling Encryption using Chaos-controlled Poker Shuffle Operation was proposed this method was tested and implemented and it gave a 0.56 second encryption time to encrypt an RGB color image with size 256x256x3.
Proposed method

The sender and receiver must use the same key for encryption-decryption as shown in figure (2)

![Fig 2. Encryption-decryption](image)

The proposed method can be implemented applying the following phases:

**Phase 1: Private-key generation**

To increase the security of the proposed method and to suit any image size a large 3D matrix with random values will be generated, the generated key must be saved for later use to encrypt or decrypt any image.

The following key was generated and used here in this paper:

\[
\text{key} = \text{uint8}(255 \times \text{rand}(5000, 5000, 3))
\]

Figure (3) shows a sample part of the generated key:

\[
\text{key}(1:10, 1:10, 1) =
\begin{bmatrix}
127 & 19 & 223 & 76 & 1 & 183 & 253 & 127 & 9 & 55 \\
211 & 41 & 197 & 75 & 226 & 209 & 242 & 94 & 250 \\
39 & 64 & 249 & 52 & 67 & 162 & 225 & 20 & 142 & 38 \\
49 & 143 & 106 & 104 & 4 & 223 & 63 & 71 & 69 & 173 \\
155 & 53 & 211 & 61 & 2 & 130 & 91 & 9 & 129 & 127 \\
90 & 157 & 26 & 07 & 07 & 75 & 00 & 225 & 214 & 11 \\
222 & 162 & 200 & 90 & 201 & 49 & 127 & 24 & 129 & 149 \\
87 & 209 & 159 & 06 & 35 & 126 & 202 & 42 & 191 & 213 \\
107 & 224 & 237 & 111 & 166 & 117 & 189 & 124 & 137 & 105 \\
60 & 42 & 159 & 99 & 64 & 192 & 240 & 223 & 123 & 11 \\
\end{bmatrix}
\]

Fig 3. Sample of the generated key

**Phase 2: Image encryption**

This phase can be implemented applying the following steps:

- Get the original input image.
- Find the input image dimensions as follows:
  \([\text{rows}, \text{columns}, \text{colors}] = \text{size(\text{original image})}\)
- Load the key.
- Adjust the key to suit the input image size by extracting a used_key as follows:
  \(\text{Used key} = \text{key}(1: \text{rows}, 1: \text{columns}, 1: \text{colors})\)
- Find the encrypted image by applying the following formula:
  \(\text{Encrypted image} = \text{Original image} \oplus \text{Used key}\)
- Save the encrypted image.

**Phase 3: Image decryption**

This phase can be implemented applying the following steps:

- Get the encrypted image.
- Find the encrypted image dimensions as follows:
  \([\text{rows}, \text{columns}, \text{colors}] = \text{size(\text{Encrypted image})}\)
- Load the key.
- Adjust the key to suit the encrypted image size by extracting a used_key as follows:
  \(\text{Used key} = \text{key}(1: \text{rows}, 1: \text{columns}, 1: \text{colors})\)
- Find the decrypted image by applying the following formula:
  \(\text{Decrypted image} = \text{Encrypted image} \oplus \text{Used key}\)
- Save the decrypted image.

The proposed method was implemented and the decrypted image was always the same as the original input image, some experimental samples are shown in figures (4) through (8):
III. RESULTS AND DISCUSSION

The proposed method was implemented using various images (binary, gray and color images with different types), one key for all the experiments was selected and table (I) shows some results samples of the performed experiments:

<table>
<thead>
<tr>
<th>Image number</th>
<th>Image size (pixels)</th>
<th>Size in pixels</th>
<th>Encryption time (seconds)</th>
<th>Decryption time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>177x2843 x</td>
<td>150804</td>
<td>0.323000</td>
<td>0.312000</td>
</tr>
<tr>
<td>2</td>
<td>222x2283 x</td>
<td>151848</td>
<td>0.327000</td>
<td>0.327000</td>
</tr>
<tr>
<td>3</td>
<td>186x2713 x</td>
<td>151218</td>
<td>0.327000</td>
<td>0.311000</td>
</tr>
<tr>
<td>4</td>
<td>196x2583 x</td>
<td>151704</td>
<td>0.323000</td>
<td>0.308000</td>
</tr>
<tr>
<td>5</td>
<td>177x2843 x</td>
<td>150804</td>
<td>0.323000</td>
<td>0.310000</td>
</tr>
<tr>
<td>6</td>
<td>225x2253 x</td>
<td>151875</td>
<td>0.325000</td>
<td>0.310000</td>
</tr>
<tr>
<td>7</td>
<td>177x2843 x</td>
<td>150804</td>
<td>0.320000</td>
<td>0.307000</td>
</tr>
<tr>
<td>8</td>
<td>177x2843 x</td>
<td>150804</td>
<td>0.321000</td>
<td>0.304000</td>
</tr>
</tbody>
</table>

Table I

SIMPLICITY ISSUES

It is very simple to generate the encryption-decryption key, this key can be generated once and it can be used for any image type with any size by adjusting the key size to suit the image size. Also an XORing operation used is very simple and fast to implement.

SECURITY ISSUES

The generated encryption-decryption key is very huge and contains 750000 values each of them within the range 0 to 255, thus making the process of guessing the key very difficult; this key must be known only by the image sender and the receiver. In bad cases (if the key was hacked) it is very easy to generate a new one.

EFFICIENCY ISSUES

From table (I) we can see that the average encryption time is around 0.3276 seconds which give us a high throughput which is in average around 2 Mbyte per second. The throughput was calculated using the following formula:

\[
\text{Throughput} = \frac{\text{Image size in bytes}}{\text{Encryption time in seconds}}
\]

The experimental results were compared with other methods result and the results of comparisons gave a good speedup as show in table (II):

<table>
<thead>
<tr>
<th>Method</th>
<th>Encryption time (seconds)</th>
<th>Decryption time (seconds)</th>
<th>Total time</th>
<th>Speedup of the proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>0.3276</td>
<td>0.3147</td>
<td>0.6423</td>
<td>1.0000</td>
</tr>
<tr>
<td>Ref[9]</td>
<td>0.5</td>
<td>0.5</td>
<td>1.0000</td>
<td>1.5569</td>
</tr>
<tr>
<td>Ref[10]</td>
<td>0.4</td>
<td>0.4</td>
<td>0.8000</td>
<td>1.2455</td>
</tr>
<tr>
<td>Ref[11]</td>
<td>0.56</td>
<td>0.56</td>
<td>1.1200</td>
<td>1.7437</td>
</tr>
</tbody>
</table>

The speedup was calculated using the following formula:

\[
\text{Speedup} = \frac{\text{Other method time}}{\text{proposed method time}}
\]

ACCURACY ISSUES

The obtained decrypted image was always the same as the original image for all experiments and the value of the mean square error (MSE) [12] was always zero and the value of peak signal to noise ratio (PSNR)[12] was always infinite which means the 100% of encryption-decryption process.

IV. CONCLUSION

A method of image encryption-decryption process was produced, the experimental results showed that the proposed method has the following important features:

Very simple to use.
High secure making hacking impossible.
Very accurate by minimizing MSE to zero.
Very efficient by increasing the speedup and increasing the method throughput.
REFERENCES


