Image Encryption System using Rubik’s Cube Algorithm

Srushi Gandhi\textsuperscript{1}, Taral Butani\textsuperscript{2}, Ravi Gor\textsuperscript{2}

\textsuperscript{1}Research Scholar, Department of Mathematics, Gujarat University, Ahmedabad, Gujarat, India
\textsuperscript{2}P.G. Student, Department of Applied Mathematical Science, Actuarial Science and Analysis, Gujarat University, Ahmedabad, Gujarat, India
\textsuperscript{3}Department of Mathematics, Gujarat University, Ahmedabad, Gujarat, India

Abstract: An amazing digital multimedia revolution marked the end of the 20th century. In this revolution, images are extremely important to communicate in the present era of multimedia. When a person transfers images over an unsecured communication network, maintaining image confidentiality is a difficult task.\textsuperscript{[1]}In this paper, the Rubik's cube algorithm's principle is applied to the image pixel value of Color, greyscale and text picture images. The XNOR operation is then performed on the scrambled image using two secret keys. Secret keys are generated using a random number and can also be manually selected. Decrypting images requires both keys and the number of iterations performed by the algorithm. Finally, the experimental results and security analysis show how important keys are in encryption or decryption. Whenever it comes to attacks, images are highly secure.

Keywords: Rubik’s Cube, key Generation, Scrambled image, Encryption, Decryption.

Date of Submission: 15-08-2022
Date of Acceptance: 31-08-2022

I. Introduction

Technology and security are important facts of living in the twenty-first century. The base of technology is digital multimedia. The advantages of the digital revolution, however, did not come without drawbacks, such as illegal copying and sharing of digital multimedia documents. To meet this challenge, documents were transformed using cryptography to make them unreadable by anyone except the legal person. The digital image is now one of the most popular multimedia formats. It is widely used in a variety of fields, such as medicine, research, industry, economy, and national defense as images contain a lot of data.\textsuperscript{[3]}

Cryptography is the art and science of securing data from unauthorized users via converting it into an unknown format which is unrecognizable while being stored and delivered. In cryptography, image encryption is an extremely effective method of protecting images by transforming them into unrecognizable formats. Moreover, image data has its own set of characteristics, including a large amount of information, significant redundancy, and a strong correlation between nearby pixels.

Image encryption is dependent on the size and format of the image. The most common image formats are black and white, grayscale, and RGB. A binary image is one that contains only pure black and white pixels is called a black and white image. A grayscale image contains only shades of grey, varying from black to white. It’s a collection of $M \times N$ measures. An RGB image is a sophisticated image in which each pixel has three tone components. Red (R), green (G), and blue (B) are the three-part. The image shows a range of measurements. $M \times N \times 3$, where n denotes the number of lines and m denotes the number of sections, and 3 denotes the layer or shading segment quantity. Each pixel's layer has its power value. The shading power is a whole integer ranging from 0 to 255, with 0 being dark (the haziest) and 255 representing white (the lightest).\textsuperscript{[4]}

The goal of this study is to secure images.\textsuperscript{[5]}Traditional image encryption can be done by symmetric key algorithms such as DES and AES, asymmetric key algorithms such as Rivest Shamir Adleman (RSA), and the family of elliptic-curve-based encryption techniques (ECC). Also, some algorithms such as the Linear Feedback Shift Register (LFSR), Triple DES, and Chaotic Map like Gauss map, logistic map, tent map etc. and many others are used for image encryption and decryption. Rubik’s cube algorithm is one of them.

In this paper, we show a color image encryption algorithm based on the Rubik's cube principle. First, the pixel of the original image is scrambled using the Rubik's cube technique, which simply changes the position of the pixels. The bitwise XNOR is applied to the even rows and columns using two secret keys. Then, also for odd rows and columns, the bitwise XOR is applied. Color images should be scrambled in three separate formats: red (R), green (G), and blue (B).\textsuperscript{[1]}\textsuperscript{[2]}\textsuperscript{[6]}

The rest of the article is organised as follows: The first section introduced information security and image encryption techniques. Section II covers literature review Section III Covers key generation and the Rubik’s cube algorithm. Section IV details the proposed image encryption and decryption technique, Section V
describes the results and discusses various parameters, and Section V summarises the research outcome and suggests future directions.

II. Literature Review

Khaled et al. [1] (2011) proposed a method for scrambling the pixels of a grayscale original image by using Rubik’s cube principle, which simply changes the position of the pixels. They used two secret keys that have been generated randomly, and the bitwise XOR was applied to the odd rows and columns. The flipped secret keys were used to apply the bitwise XOR to even rows and columns. These steps were repeated until the required number of iterations were not reached. It was also shown that decrypting with the incorrect key gives a completely different image and some different types of attack against the Rubik’s cube algorithm.

Gomathi [6] (2016) projected a secure image encryption algorithm based on ANN (Artificial Neural Network) and Rubik’s cube principle. In this paper, they used a method called ANN-based multistage image encryption using Rubik’s cube method was that proposed to secure data transmission. A better image hiding method for secured data communication has been proposed. Experiments demonstrated that the proposed approach had a good encryption effect.

Vidhya et al. [3] (2020) proposed a new chaos-based image encryption technique that used the Rubik’s cube and a prime factorization process based on initial random value creation to achieve high plain picture sensitivity and defeat plain image-related assaults. The CIERPF approach was shown to be effective against various attacks. The system’s randomness was proved using an entropy analysis that was close to 8. For the encryption approach, a correlation and histogram analysis were performed, revealing that there was no statistical similarity between the original and encrypted images.

III. Terminologies

Key Generation

In the encryption and decryption of images, the keys are very important. The most common way to produce a key with a random number or a specific method, such as RSA key generation, LFSR, Pseudorandom Number Generator (PRNG), and so on. The length of keys in the Rubik’s Cube algorithm is determined by the image's width and height. [12]

Assume that the presented algorithm will use both the Kc and Kr keys. The length of the generated key and the image width must both be the same for row operation. Similarly, the image height and key length must be the same for column operation. In other words, the length of the keys is the same as the image size. Each keys take value of the set \( \{0, 1, 2, \ldots, 2^\alpha - 1\} \) where \( \alpha \) is bit value for key. It plays an important role in encryption. Keys generate with random number otherwise select manually. [1]

Rubik’s Cube

The Rubik’s Cube is a three-dimensional combination puzzle created by Ern Rubik, a Hungarian professor of architecture, in 1974. The Magic Cube was its original name. Rubik patented the three-plane Rubik’s cube puzzle, which was sold by Ideal Toy Corp in 1980. Each of the six faces of the Rubik’s Cube was covered by three by three by three nine labels each of which was one of six solid colors: white, red, blue, orange, green, and yellow. Each face of the three-dimensional Rubik’s cube puzzle must be returned to having only one color to solve it. Original Image compares with the Rubik’s cube, when the cube is rotated, and the position of box will change. All boxes will be rearranged. They return all the colored boxes to their original positions after using the algorithms for solving the Rubik’s Cube. [5][6]

Figure 1: Encryption - Decryption Rubik’s Cube
IV. METHODOLOGY

- The Rubik's cube algorithm is based on the image's Pixel value. The Rubik's cube principle is used to scramble the original image by changing a pixel value and shifting a row or column using modulo 2 (mod 2).
- The bit wise XNOR operation apply on even column with $K_C$ and even column of $K_R$ then after bit wise XNOR operation is applied with flipping of key $K_C$ and $K_R$. Here, $K_R$ is used for row operation and $K_C$ is used for column operation.
- These steps can be repeated as many times as required number of iterations is reached.
- Encrypted image, both keys, and the number of iterations executed are required for decryption.

The Rubik's cube algorithm is based on a single component. Each pixel in grayscale or black-and-white image has a single tone component. When an image pixel has more than one tone component, such as in an RGB image, the Rubik's cube algorithm faces a challenge because it has three tone components. Each component of the RGB image is subjected to the Rubik's cube method one by one. The output of each component is then combined to create an encrypted image.$^5$

![Image](image.png)

**Figure 2:** Pixel of RGB image is formed from the corresponding pixel of the three-component image

V. PROPOSED ALGORITHM

**Image Encryption Algorithm**$^{18,26}$

Rubik’s cube image encryption algorithm mainly works on pixels values matrix of image and below is a way for encrypting an image.

**Step-1:** Let I represent image of the size $M \times N$ length. The pixels values matrix of image is represented by $I_0$. Take two keys $K_R$ and $K_C$ of length $M$ and $N$, $M$ & $N \in \{0, 1, 2, \ldots, 2^\alpha - 1\}$. Where $\alpha$ is bit value.

**Step-2:** For each Row $i$ of image $I_0$,

$$\alpha(i) = \sum_{j=1}^{N} I_0(i,j); \ i = 1, 2, \ldots, M \ & j = 1, 2, \ldots, N$$

Compute modulo 2 of $\alpha(i)$, denoted by $M\alpha(i)$, if $M\alpha(i) = 0 \rightarrow$ left circular shift. else → right circular shift.

In the $R(i)$ position, an image pixel moves left or right. The first pixel moves at the last pixel position, or the last pixel moves at the first position

**Step-3:** For each Column $j$ of image $I_0$,

$$\beta(j) = \sum_{i=1}^{M} I_0(i,j); \ i = 1, 2, \ldots, M \ & j = 1, 2, \ldots, N$$

Compute modulo 2 of $\beta(j)$, denoted by $M\beta(j)$, if $M\beta(j) = 0 \rightarrow$ down circular shift. else → up circular shift.

In the $C(j)$ position, an image pixel moves up or down. The first pixel moves down at the last pixel position, or the last pixel moves up at the first position.

Steps 2 and 3 will give a scrambled image, which is denoted by $I_{SCR}$. Determine the number of iterations, $ITER_{MAX}$, and initialize $ITER$ at 0.

**Step-4:** Using vector $K_C$ bitwise XNOR operator is applied to each row of scrambled image.

$$I_1(2i,j) = I_{SCR}(2i,j) \bigoplus K_C(j),$$

$$I_1(2i-1,j) = I_{SCR}(2i-1,j) \bigoplus rot_{180}(K_C(j))$$

The bitwise XNOR operator and the shifting of vector $K_C$ from left to right are represented by $rot_{180}(K_C)$.

**Step-5:** Using vector $K_R$ bitwise XNOR operator is applied to each column of scrambled image.

$$I_{ENC}(i, 2j) = I_1(i, 2j) \bigoplus K_R(j),$$

$$I_{ENC}(i, 2j-1) = I_1(i, 2j-1) \bigoplus rot_{180}(K_R(j))$$

The bitwise XNOR operator and the shifting of vector $K_R$ from left to right are represented by $rot_{180}(K_R)$. If $ITER \leq ITER_{MAX}$, the algorithm generates an encrypted image $I_{ENC}$ and completes the encryption process;
otherwise repeat from step. However, it is better to set \( \text{ITER}_\text{MAX} = 1 \) to obtain a fast encryption algorithm (single iteration). If \( \text{ITER}_\text{MAX} > 1 \), on the other hand, the algorithm is more secure since the key space is greater than it is when \( \text{ITER}_\text{MAX} = 1 \).

**IMAGE DECRYPTION ALGORITHM**\(^{[1][2][6]}\)

**Step-1:** The decrypted image is recovered from the encrypted image \( I_{\text{ENC}} \) by the secret keys \( K_R \) and \( K_C \), and \( \text{ITER}_\text{MAX} \). Initialize \( \text{ITER} \) at 0.

**Step-2:** The bitwise XNOR operation is applied on each column of the encrypted image \( I_{\text{ENC}} \) as follows:

\[
I_1(i, 2j) = l_{\text{ENC}}(i, 2j) \oplus K_R(j), \\
I_1(i, 2j - 1) = l_{\text{ENC}}(i, 2j - 1) \oplus \text{rot180}(K_R(j))
\]

**Step-3:** Using the \( K_C \) vector, the bitwise XNOR operator is applied to each row of image \( I_1 \):

\[
l_{\text{SCR}}(2i, j) = l_1(2i, j) \oplus Kc(j), \\
l_{\text{SCR}}(2i - 1, j) = l_1(2i - 1, j) \oplus \text{rot180}(Kc(j))
\]

**Step-4:** For each column \( j \) of the scrambled image \( I_{\text{SCR}} \), compute the sum of all elements in that column \( j \), denoted as

\[
\beta_{\text{SCR}}(j) = \sum I_{\text{SCR}}(i, j), \quad i = 1, 2, \ldots, M & j = 1, 2, \ldots, N
\]

Compute \( \text{mod} \)ulo 2 of \( M(\beta_{\text{SCR}}(j)) \), denoted by \( M(\beta_{\text{SCR}}(j)) \), if \( M(\beta_{\text{SCR}}(j)) = 0 \) → down circular shift.

else → up circular shift.

**Step-5:** Compute the sum of all elements in row \( i \), denoted as

\[
\alpha_{\text{SCR}}(i) = \sum I_{\text{SCR}}(i, j), i = 1, 2, \ldots, M & j = 1, 2, \ldots, N
\]

Compute \( \text{mod} \)ulo 2 of \( \alpha_{\text{SCR}}(i) \), denoted by \( M(\alpha_{\text{SCR}}(j)) \).

If \( M(\alpha_{\text{SCR}}(j)) = 0 \) → left circular shift.

else → right circular shift.

If \( \text{ITER}_n = \text{ITER}_\text{MAX} \), completes the decryption process, otherwise repeat from step 2.

**VI. EXAMPLE**

For the experimental results analysis, we took different two types of images: Color image and grayscale image.

**For Color Image**

![Figure 3: Color Image](image)

**Color Image Encryption**

Using python,

**Step-1:** Enter the value of ALPHA: 8

High, low: 255 0

Width, Height of image: 1674, 1046

\( K_R \) = randomly generate of length 1674

\( K_C \) = randomly generate of length 1046

It processes color image encryption using three different parts for red, green, and blue color components.

![Step-2: Encrypted red tone components](image)

![Step-3: Encrypted green tone components](image)

![Step-4: Encrypted blue tone components](image)

**Figure 4:** Encrypted images of RGB Components
**Step -5:** Final encrypted image with red, green and blue ton component.

![Figure 5: Color Encrypted Image](image1.jpg)

The encrypted image is 99.56% different compared to the original image. Here White dotes show the similar pixel values of original image and encrypted image.

![Figure 6: Similarity between Original & Encrypted Image](image2.jpg)

**Color Image Decryption**

**Step-1:** Decryption works in the same way as encryption, but in the opposite way. Take an image that has been encrypted processed using both the $K_C$ and $K_R$ keys that are used in encryption.

**Step-2**

![Figure 7A: Decrypted red tone component](image3.jpg)

**Step-3**

![Figure 7B: Decrypted green tone component](image4.jpg)

**Step-4**

![Figure 7C: Decrypted red tone component](image5.jpg)

**Figure 7: Decryption of RGB tone component**

**Step -5:** Final decrypted image with red, green and blue ton component. The decrypted image is 99.98% similar compared to the original image.

![Figure 8: Decrypted color image](image6.jpg)
For Grayscale Image
It processes gray scale image encryption using one component. With using alpha: 8, High, low: 255 0, Width, Height of an image: 275, 183. Randomly generated $K_C$ of length 275, $K_R$ of length 183.

On comparing original and encrypted images, similarity between encrypted and original grayscale image is 1%. White dots show the similar pixel values of original image and encrypted image.

For Text Image
It processes image which also contain encryption using one component. With using alpha: 8, High, low: 255 0. Width, Height of image: 735 764. Randomly generated $K_C$ of length 275, $K_R$ of length 183.

On decrypting the text image, the curve of the letters become slightly blurred.
Comparing original and encrypted images, Similarity between encrypted and original image is 8%. White dots show the similar pixel values of original image and encrypted image.
VII. RESULTS AND DISCUSSION

Visual Testing

The encrypted image should look very different than the original. To measure this requirement, two different measurements are used in general. The number of pixels change rate (NPCR) is the first metric, which indicates the percentage of different pixels between two images. The unified average changing intensity (UACI) is the second, and it measures the average intensity of pixel differences between two images. Let $I_0(i,j)$ and $I_{ENC}(i,j)$ be the pixels values of original and encrypted images, $I_0$ and $I_{ENC}$, at the $i^{th}$ pixel row and $j^{th}$ pixel column, respectively.

\[
NPCR = \frac{\sum \sum D(i,j)}{MN} \times 100\%
\]

Where, $D(I,j) = \begin{cases} 0 & I_0(i,j) = I_{ENC}(i,j) \\ 1 & \text{Otherwise} \end{cases}$

\[
UACI = \frac{\sum \sum |I_0(i,j)-I_{ENC}(i,j)|}{255MN} \times 100\%
\]

<table>
<thead>
<tr>
<th>Images</th>
<th>Number of Pixels Change Rate (NPCR)</th>
<th>Unified Average Changing Intensity (UACI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color image – Red Ton component</td>
<td>99.60%</td>
<td>35.55%</td>
</tr>
<tr>
<td>Color image – Green Ton component</td>
<td>99.61%</td>
<td>31.98%</td>
</tr>
<tr>
<td>Color image – Blue Ton component</td>
<td>99.60%</td>
<td>34.39%</td>
</tr>
<tr>
<td>Color image -1</td>
<td>99.99%</td>
<td>33.97%</td>
</tr>
<tr>
<td>Grayscale image -2</td>
<td>99.98%</td>
<td>39.79%</td>
</tr>
<tr>
<td>Text image -3</td>
<td>99.99%</td>
<td>48.22%</td>
</tr>
</tbody>
</table>

The NPCR values must be as large as possible, and UACI values must be around 33%, to accurately measure the performance of an ideal image encryption algorithm. The NPCR and UACI values for the original and encrypted images are listed in the above table. For each image, the number of pixels change rate is close to unity (100%).

The Unified average changing intensity (UACI) values show that almost all the encrypted image’s pixel values have been changed from their original values. UACI value is higher for Text image -3 because of the pixels values in text images are at the limits of the pixels value range, i.e., pixel values for text images are either 0 for black or 255 for white, resulting in a large absolute difference between the original and encrypted images.

Key Space Analysis

To make brute-force attacks computationally impossible, a secure image encryption algorithm must have a large key space. The proposed algorithm, in theory, can accommodate an infinite key space. The encryption key in our approach, on the other hand, is made up of the $(K_R, K_C, ITER_{MAX})$ triplet. For an $a$-bit image $I_0$ of size $M \times N$ pixels, the vector $K_R$ and $K_C$ can take $2^{Ma}$ and $2^{Na}$ possible values respectively. The total possible keys are $2^{a(M \times N)} \times ITER_{MAX}$. Suppose values of $a$ is 8 of image size 256 x 256 and $ITER_{MAX} = 1$ then total possible key is $2^{4096}$. This key space is large enough to resist exhaustive attack.

Key Sensibility

Encryption algorithms should be sensitive to the encryption key, any little change in the key should result in a large change in the encrypted (or decrypted) image. On a grayscale image, we tested key sensibility by encrypting with key set-1 ($K_{C_1}, K_{R_1}$) and then encrypting with alternative values of key, say key set-2($K_{C_2}, K_{R_2}$), and getting a different encrypted image.
VIII. LIMITATIONS

The encrypted image must be in.png format for the decryption process to work; otherwise, the decrypted image will not be accurate. In the.png format, accuracy is over 99%, but in other formats, it’s around 20%.

When the same process is done on digital image, accuracy for .png, .jpg, .jpeg is 99.71%, 99.49%, 100% respectively. For that, it is not necessary that the encrypted image should be in.png format for the decryption process to work.

IX. COMPARISON[5]

On comparing Rubik’s Cube Algorithm for color image, Text image, greyscale image and digital image:

<table>
<thead>
<tr>
<th>Type of image</th>
<th>Color Image</th>
<th>Text Image</th>
<th>Greyscale Image</th>
<th>Digital image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image form</td>
<td>Input image:.jpeg Encrypted image:.png Decrypted image:.jpeg</td>
<td>Input image:.jpeg Encrypted image:.png Decrypted image:.jpeg</td>
<td>Input image:.jpeg Encrypted image:.png Decrypted image:.jpeg</td>
<td>Input image:.jpeg Encrypted image:.png Decrypted image:.jpeg</td>
</tr>
<tr>
<td>Generation of key</td>
<td>Randomly generated Key through PYTHON</td>
<td>99.99% 99.99% 99.99% 99.71%–99.99%</td>
<td>99% 99% 99% 100%</td>
<td></td>
</tr>
<tr>
<td>UACI</td>
<td>33.97%</td>
<td>48.22%</td>
<td>39.79%</td>
<td>39.21%</td>
</tr>
</tbody>
</table>

The above compression shows that Rubik’s Cube Algorithm works well for the color image and digital image.

X. CONCLUSION

On color or grayscale images, a new image encryption algorithm is proposed in this study. This algorithm permutes image pixels using the Rubik’s cube principle. The XNOR operator is applied to odd rows and columns of animage using a key to obfuscate the relationship between original and encrypted images. The same key is flipped and applied to even rows and columns of image.

Experiments with comprehensive numerical analysis have been conducted, showing the proposed algorithm’s stability against a variety of attacks, including statistical and differential attacks (visual testing). Furthermore, for the performance evaluation, experimental results show Rubik’s cube image encryption technique is also accurate and secure for color images.

REFERENCE


