

Digital Image Encryption using Rubik's Cube Algorithm

Gandhi Srushti¹, Ravi Gor²

¹Research Scholar, Department of Mathematics, Gujarat University, Gujarat, India

²Department of Mathematics, Gujarat University, Gujarat, India

Abstract:

An amazing digital multimedia revolution was marked at the end of the 20th century. In this revolution, images are extremely important to communicate in the present era of multimedia. In this world of hasty evolution of exchanging digital data as well as images, data protection is essential to protect data from the unauthorized parties. With the broad use of digital images of various fields, it is important to preserve the confidentiality of image's data from any party without authorization access. In this paper, encryption and decryption process is based on key which is generated randomly which is applied on Rubik's Cube Algorithm.

Key Words: Rubik cub, Scrambled image, image encryption-decryption, key generation.

Date of Submission: 22-06-2022

Date of Acceptance: 04-07-2022

I. INTRODUCTION

Since digital images are very important in multimedia technologies, user privacy becomes even more important. Encrypting the image to prevent against unauthorised access is critical to ensure security and privacy of the user. Encryption of images and videos are used in several fields, together with Internet communications, multimedia systems, medical imaging, telemedicine and military communications. Massive numbers of images are sent and stored via Internet and wireless networks, which takes advantage of the increasing development of multimedia and network technology. Cryptography has been a battleground for mathematicians and scientists since 1949, when it became a critical component of security. AES, DES, RSA, IDEA, and some cryptographic methods are now implemented.

The image is most commonly used by the means of communication in a variety of fields, including medicine, research, industry, and the military. The crucial image transfer will take place across an unauthorized Internet network. As a result, robust security is essential to ensure that the image prevents unauthorised access to sensitive data. The image has the advantage of covering more multimedia information and hence requires security. Cryptography is a form of image security method that allows to send and save images securely over the Internet. The integrity, privacy, and authenticity of the image are the top priorities for any system. When dealing with data containing grey levels, cryptography is the most efficient solution, but it also has security difficulties. In this scheme, a Rubik cube is included with permutation-boxes to the higher security of the scheme. The main objectives of this paper are as follows:

- Presenting a novel dynamic method which is simple and easy to implement. It is difficult to guess the key as well as an image by taking advantage of both the traditional and modern encryption algorithms.
- Rather than focusing just on developing a powerful algorithm, it is focused on developing a strong algorithm that can execute in an acceptable timeframe on currently available infrastructure.
- Proper mathematical methods that help our design to perform at a higher speed rate.
- Increasing the diffusion by using powerful Rubik cube algorithm.

II. LITERATURE REVIEW

Samson^[2](2017) projected RSA algorithm to encrypt images. To operate with RGB images, RSA algorithm has been modified. The trials revealed that they were able to encrypt and decrypt a variety of images successfully, and that the technology has a good encryption effect. The cipher image developed by their technology was totally different from the original image. This approach was suitable for securing image transfer over the Internet. It provides the great security.

Santhiya et.al.^[3](2018) proposed a new medical image encryption algorithm. They employed a Henon map for visual confusion and a Linear Feedback Shift Register (LFSR) for diffusion. Based on matrix values, the researchers claimed that their system could withstand differential attacks. They demonstrated that their method can assure the security of DICOM images.

Mousavi et.al.^[4] (2021) presented a novel chaos-based dynamic encryption scheme with a permutation substitution structure. The scheme's S-boxes and P-boxes were built using chaotic transformation and Rubik

cube-based permutation. It improved the security, sensitivity, and robustness of their scheme. To create block cipher, they combine a chaotic map and a Feistel network. They employed the Feistel network to encrypt and decrypt data using the same structure. Due to the use of chaotic systems, the Feistel structure only required seven rounds. It was less than the number of rounds required by encryption techniques. They improve their proposed algorithm efficiency. To improve security and efficiency, they made this block cipher totally dynamic. Their architecture comprises of a 192-bit block cypher and eight 8-bit S-boxes, which were simple and secure due to chaotic systems.

Zhu^[6] (2021) introduced a new three-dimensional bit-level image encryption scheme with Rubik's cube method (3D-BERC). The Rubik's cube method and the bit-level encryption theory were coupled in this permutation process to achieve image scrambling in three-dimensional space. The ascending and decreasing dimension operations were successfully achieved using the contraction mapping method. They created a better two-dimensional diffusion structure during the diffusion phase. They propagated minor changes in the plain image to the entire cipher image. The suggested scheme's security and validity were demonstrated by experimental findings and simulation analyses.

Chen^[1] (2021) proposed a double colour image encryption method based on DNA computation and chaos theory. Unlike traditional techniques, double colour images were encrypted simultaneously to save information about each other. It made encryption more secure and reliable. As a pseudo-random number generator, a novel chaotic fractional order (FO) was discrete and improved by Henon map (FODIHM). In addition, the pixels in each colour component of the two images were scrambled using a Rubik's cube transform. Then, using distinct DNA coding algorithms, each pixel in each colour component of the two images were diluted. Finally, the CAT transform was applied to improve the security performance, which was based on the FO discrete Logistic map and the conventional XNOR. The algorithm's numerical findings and security analysis showed that it is resistant to a number of common threats

Zhao^[5] (2022) used an alternate quantum walk and Rubik's cube transform for a novel colour image encryption scheme. This scheme's main approach was to use a quantum random walk to generate a random sequence. It extracts image pixels to form a third-order Rubik's Cube. Then he used random sequences to achieve visual scrambling by controlling the rotation of the Rubik's Cube. Experiments demonstrated that the proposed approach had a good encryption effect.

III. TERMINOLOGIES

Rubik's Cube^[6]

The Rubik's cube is a three-dimensional puzzle created by Ern Rubik in 1974. Ern Rubik was a well-known sculptor and architectural professor in Hungary. Rubik patented the three-plane Rubik's cube puzzle, which was sold by Ideal Toy Corp.

Each side of the three-dimensional Rubik's cubes is covered with nine labels. Each of the six colours (yellow, green, orange, blue, red, and white) is represented by a label. The red is opposite to the orange in the currently available versions, the green is opposite to the blue and the yellow is opposite to the white. The blue, white, and red are arranged in a clockwise direction.

The Rubik's cube can twist independently, thanks to an inbuilt rotating mechanism. The colours become disorganised as a result. To solve the three-dimensional Rubik's cube puzzle, each of the cube's six faces must display one colour. Since the introduction of the Rubik's cube in the mid-1970s, three plane puzzles of the same sort have been made, and not all of them are Rubik's cubes.

The encryption and decryption of the Rubik's cube is shown in the following figure.

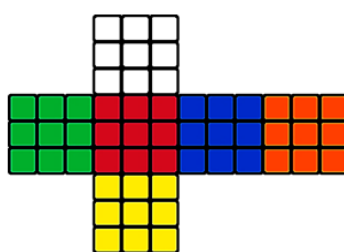


Figure 1(a): Rubik Cube

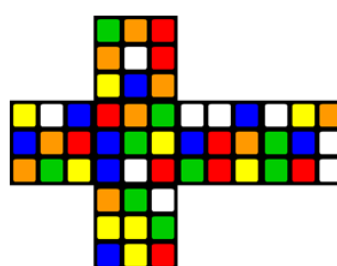


Figure 1(b): Encrypted Rubik Cube

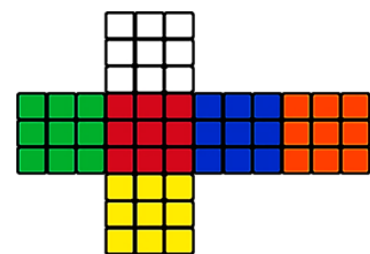


Figure 1(c): Decrypted Rubik Cube

Figure 1: Rubik Cube Algorithm

IV. METHODOLOGY

The Rubik's cube algorithm is based on the Pixel value of an image. The Rubik's cube principle is used to scramble the original image by 180°. Thus, there is a change in the pixel values and shifting row or column using modulo 2. Calculate the pixel values row by row, then column by column, and take modulo 2. If the value is 0, the pixel values shift left for the row, up for the column, and opposite for the row and column when value is 1.

Then, using two secret keys K_1 and K_2 generated randomly, XNOR operation on row and column are performed. Repeat the steps until the iteration is the same. The encrypted image is obtained after the maximum iteration. These steps can be repeated as many times as required number of iterations are reached. Here K_1 key is used for column operation and K_2 key is used for row operation.

Encrypted image, both the keys, and the number of iterations executed are required for decryption. To decrypt images, perform the XNOR operation by row and column wise and modulo 2 image scrambles to obtain the decrypted image.

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. The similar process is done for digital image.

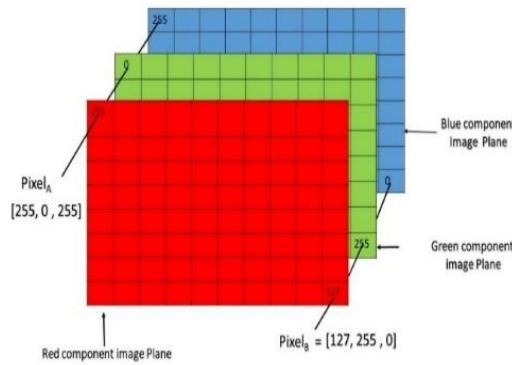


Figure 2: RGB plane

V. PROPOSED ALGORITHM

Encryption

Step1: Let I denote an α -bit of image of the size $m \times n$ length. Generate two random sequences of length m and n . Let them be denoted

as K_1 and K_2 . Each take a random value of the set $A = \{0, 1, 2, \dots, 2\alpha - 1\}$.

Step2: For each row i of image I , $\alpha(i) = \sum I(i, j)$; $i = 1, 2, \dots, m$ & $j = 1, 2, \dots, n$. Calculate modulo 2 of $\alpha(i)$, denoted by $M\alpha(i)$,

$$\text{If } M\alpha(i) = \begin{cases} 0; & \text{right circular shift} \\ \text{else;} & \text{left circular shift} \end{cases}$$

Step3: For each column i of image I , $\beta(i) = \sum I(i, j)$; $i = 1, 2, \dots, m$ & $j = 1, 2, \dots, n$. Calculate modulo 2 of $\beta(i)$, denoted by

$$\text{If } M\beta(i) = \begin{cases} 0; & \text{up circular shift} \\ \text{else;} & \text{down circular shift} \end{cases}$$

Determine the number of iterations, $Iter_{max}$, and initialize the $iter$ at 0.

Step4: By using key K_2 , bitwise XNOR is applied to each row of the scrambled image.

$$\begin{aligned} I_1(2i - 1, j) &= I_{scr}(2i - 1, j) \oplus K_2(j) \\ I_1(2i, j) &= I_{scr}(2i, j) \oplus K_2(j) \end{aligned}$$

Step5: By using key K_1 , bitwise XNOR is applied to each column of the scrambled image.

$$\begin{aligned} I_{enc}(2i - 1, j) &= I_i(2i - 1, j) \oplus K_1(j) \\ I_{enc}(2i, j) &= I_i(2i, j) \oplus K_1(j) \end{aligned}$$

Now, the image obtained is encrypted.

Decryption

Step1: The decrypted image, I , is recovered from the encrypted image, I_{enc} , and the secret keys: K_1 and K_2 , and $Iter_{max}$. Initialize $Iter = 0$.

Step2: By using key K_1 , bitwise XNOR is applied to each column of scrambled image.

$$I_{enc}(2i-1, j) = I_i(2i-1, j) \oplus K_1(j)$$

$$I_{enc}(2i, j) = I_i(2i, j) \oplus K_1(j)$$

Step3: By using key K_2 , bitwise XNOR is applied to each row of image I_1 .

$$I_1(2i-1, j) = I_{scr}(2i-1, j) \oplus K_2(j)$$

$$I_1(2i, j) = I_{scr}(2i, j) \oplus K_2(j)$$

Step4: For each column j of the scrambled image I_{scr} , calculate the sum of all elements in that column j , denoted as

$$\beta(i) = \sum I(i, j); i = 1, 2, \dots, m \ \& \ j = 1, 2 \dots n.$$

Calculate modulo 2 of $\beta(i)$, denoted by $M\beta(i)$,

$$\text{If } M\beta(i) = \begin{cases} 0; & \text{up circular shift} \\ \text{else;} & \text{down circular shift} \end{cases}$$

Step5: For each Row i of image I , $\alpha(i) = \sum I(i, j); i = 1, 2, \dots, m \ \& \ j = 1, 2 \dots n$. Calculate modulo 2 of $\alpha(i)$, denoted by $M\alpha(i)$,

$$\text{If } M\alpha(i) = \begin{cases} 0; & \text{right circular shift} \\ \text{else;} & \text{left circular shift} \end{cases}$$

Now, the image is decrypted and so we get our original image back.

Here, generally, we get encrypted image at $I_{ter}_n = I_{ter}_n + 1$; we get decrypted image at $I_{ter}_n = I_{ter}_{max}$.

VI. EXAMPLE



Figure 1: Original image

Using python,

Image array: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1599x1091 at 0x20DCA2AF730>

Numpy array :<class 'numpy.ndarray'>

Image shape: (1091, 1599, 3)

Pillow image: <class 'PIL.Image.Image'>

Image mode: RGB

Image size: (1599, 1091)

Image array/in matrix form:

30	38	17	...	11	47	21
30	38	17	...	38	73	49
31	39	18	...	38	71	50
29	34	12	...	0	32	7
23	28	6	...	10	42	19
20	25	3	...	3	35	12
26	30	7	...	0	22	0
15	19	0	...	0	24	1
7	1	0	...	0	15	0
...
255	219	177	...	112	72	36
255	213	171	...	121	81	45
251	212	169	...	154	114	78

255	214	172	169	219	93
250	208	166				151	111	75
247	208	165				144	104	68
254	212	170	191	151	115
249	207	165				178	138	102
246	207	164				163	123	87

Dtype=Unit 8

Image Encryption

Step-1: insert the image and select the value of alpha.

Enter The value of ALPHA: 8

255, 0

(30, 38, 17)

<PixelAccess object at 0x7fd511a6c7d0>

Width and height of an image: 1599, 1091

3

False

Step-2: Randomly generate the both keys K_C and K_R otherwise select manually. Here, key is selected randomly.

$K_1 = [232, 219, 1, 67, 29, 53, 231, 219, 249, 47, 121, 194, 228, 12, 155, 201, 186, 108, 15, 107, 161, 99, 6, 204, 193, 93, 180, 11, 159, 210, 87, 243, 118, 146, 181, 131, 236, 117, 65, 139, 227, 204, 251, 60, 6, 192, 51, 229, 106, 142, 156, 15, 46, 132, 177, 134, 246, 199, 41, 11, 193, 112, 161, 111, 14, 50, 208, \dots, 25, 232, 182, 84, 198, 248, 161, 113, 63, 55, 101, 183, 2, 46, 53, 70, 89, 1, 90, 80, 207, 123, 159, 105, 57, 6, 233, 73, 93, 129, 29, 242, 223, 232, 0, 62, 83, 9]$

Length of $K_1 = 1599$

$K_2 = [18, 140, 133, 16, 117, 67, 249, 200, 168, 107, 32, 148, 174, 177, 144, 3, 66, 185, 4, 118, 161, 111, 223, 154, 235, 159, 129, 174, 55, 96, 231, 97, 251, 202, 165, 191, 171, 77, 202, 213, 43, 77, 162, 238, 27, 100, 18, 192, 7, 186, 116, 162, 57, 26, 225, 129, 162, 138, 106, 151, 44, 233, 31, 113, 119, 26, 12, 43, 218, \dots, 60, 41, 235, 11, 43, 234, 176, 232, 96, 76, 232, 68, 215, 109, 137, 26, 52, 52, 180, 154, 74, 62, 133, 159, 234, 62, 170, 53, 235, 54, 44, 37, 147, 84, 169, 208, 82, 110, 63, 171, 37, 241, 19, 255, 233, 125, 249, 7, 19]$

Length of $K_2 = 1091$

Step-3, 4, 5 are done on red, green and blue ton component of an image simultaneously. On combining RGB ton component of an encrypted image, final encrypted image is obtained.

Encryption of red ton component of image:



Figure 2: Encrypted red ton component

Encryption of green ton component of image:



Figure 3: Encrypted green ton component

Encryption of blue ton component of image:



Figure 4: Encrypted blue ton component

Final encrypted image with RGB ton component:



Figure 5: Final encrypted image

Image Decryption

All the steps of decryption are done on red, green and blue ton component of an image simultaneously. On combining RGB ton component of decrypted image, final decrypted image is obtained.

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_1 and K_2 keys that are used in encryption. The same key is flipped and applied to even rows and columns of image.

Decryption of red ton component of image:



Figure 6: Decrypted red ton component

Decryption of green ton component of image:



Figure 7: Decrypted green ton component

Decryption of blue ton component of image:



Figure 8: Decrypted blue ton component

Final decrypted image with RGB ton component:





Figure 9: Decrypted image

VII. RESULTS AND DISCUSSION

By this process, encrypted and decrypted image of an original .png image are obtained in 3 formats: .jpg, .jpeg and .png.

Table 1: Encryption and decryption of an image in .jpg, .png and .jpeg form.

	Encrypted Image	Decrypted Image
.jpg		





.jpeg		
.png		

Image quality and vision outcomes were generated as a result of the experiment. All the tests are done on .png images.


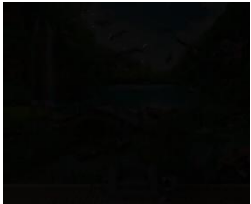

The following parameters are used to evaluate image quality:

Visual Testing

The visual testing is done online on <https://www.textcompare.org/image/>. Visual testing is done on .png images. Comparing original image with encrypted red ton component, encrypted green ton component, encrypted blue ton component and final encrypted image, similarity between them is shown in the below table. White dots show the similar pixel values of original image and encrypted image.

By comparing original image with encrypted red ton component, encrypted green ton component, encrypted blue ton component and final encrypted image the difference is shown in below table. The difference is found with full transparency when nothing is ignored along with original size and movement with different intensity.

Table 2: Comparison of an image when nothing is ignored.

Image	Comparison	Difference
Encrypted Red ton component		99.99%
Encrypted Green ton component		99.99%
Encrypted Blue ton component		98.52%

Final encrypted image		99.99%
-----------------------	---	--------

It gives same result when scale to size in same.

By comparing original image with final encrypted image, the difference is shown in below. The difference is found with full transparency when alpha is ignored along with original size and movement with different intensity. The difference is 99.71%.

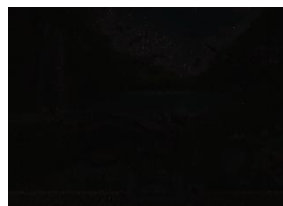


Figure 10: comparison of original and final encrypted image

Sensitivity Analysis:

Image quality and vision outcomes were generated as a result of the experiment. The following parameters are used to evaluate image quality:

i. Number of Pixels Change Rate (NPCR)

When the difference between two encrypted images is negligible, NPCRs are used to verify the number of changing pixels between them. The NPCR can be mathematically defined as follows:

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

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

m, n is the weight and height of the encrypted interferogram,

$C_1(i, j)$ is the interferogram encrypted before pixel change,

$C_2(i, j)$ is the interferogram encrypted after pixel change,

$D(i, j)$ is the bipolar network.

The optimal NPCR value is:

Table 3: NPCR of images.

Image	NPCR
Encrypted Red ton component image	100%
Encrypted Blue ton component image	100%
Encrypted Green ton component image	100%
Final encrypted image	100%

All the images have same pixel value. But in this algorithm, pixels rotate by 180° for all the RGB ton component. So, the pixel change ratio is almost 100%. Almost no pixels i.e., 0% pixels remains at their original place.

ii. Mean Squared Error (MSE) and Peak Signal to Noise Ratio (PSNR)

The PSNR block analyses the peak signal-to-noise ratio between two images in decibels. The PSNR ratio is used to compare the quality of the original and encrypted images. The better the quality of the compressed or reconstructed image, the higher the PSNR.

To compare image compression quality, the mean square error (MSE) and peak signal-to-noise ratio (PSNR) are evaluated. The MSE is a measure of the peak error between the encrypted and original image, whereas the PSNR is a measure of the cumulative squared error.

The smaller the MSE value, the smaller the error.

The PSNR is calculated by first calculating the mean-squared error (MSE) using the equation:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - k(i, j)]^2$$

$$= \sum_{i,j} \frac{[I(i,j) - k(i,j)]^2}{mn}$$

PSNR can be calculated as:

$$\begin{aligned} \text{PSNR} &= 20 \log_{10} \frac{256}{\sqrt{\text{MSE}}} \\ &= 20 \log_{10} I(i,j) - 10 \log_{10} \text{MSE} \end{aligned}$$

This result is calculated in PYTHON.

The optimal MSE and PSNE values are:

Table 4: MSE and PSNR of images.

Image	MSE	PSNR
Encrypted Red ton component image	0	27.897785825555733 dB
Encrypted Blue ton component image	0	27.899472422663468 dB
Encrypted Green ton component image	0	27.89787809645044 dB
Final encrypted image	0	27.89817638583279 dB

All the images have same pixel value. So MSE is 0.

This results are calculated in PYTHON.

iii. Unified Average Changing Intensity (UACI)

UACI is used to calculate the average intensity of the difference between the two encrypted images (C_1 and C_2). It is used to determine the strength of an encryption scheme. Its quality is determined by the format and size of the image. The average intensity variation between the ciphered and original images is measured using UACI. The highest UACI suggests that the recommended technique is resistant enough to a variety of attacks.

For an image of size $m \times n$, UACI is calculated as follows:

$$\text{UACI} = \frac{1}{mn} \sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{256} \cdot 100\%$$

The optimal UACI values are:

Table 5: UACI of images.

Image	UACI
Encrypted Red ton component image	39.21%
Encrypted Blue ton component image	39.21%
Encrypted Green ton component image	39.21%
Final encrypted image	39.21%

Entropy Analysis

Information entropy is the degree of the uncertainty associated with a random event. It tells us the amount of information present in the event. It increases with uncertainty or randomness. It finds its application in various fields such as statistical inference, lossless data compression and cryptography. The entropy $H(m)$ of m can be calculated as:

$$H(m) = - \sum_{i=0}^{255} P(x_i) \log_2 P(x_i)$$

where L is the total number of symbols;

$m_i \in m$ and $p(x_i)$ is the probability of symbol x_i .

In case of the original digital image, $H(m)$ should theoretically be equal to 8 as there are 256 values of the information source in red, green, blue, and green colours of the image with the same probability.

Entropy values are:

Table 6: Entropy of images.

Image	Entropy
Encrypted Red ton component image	5.7
Encrypted Blue ton component image	5.7
Encrypted Green ton component image	5.7
Final encrypted image	5.7

Time taken for encryption and decryption of an image

The time taken to encrypt the image is 3.154599999999341e-05 sec and time taken to decrypt the image is 1.495236727sec.

VIII. CONCLUSION

For digital image, 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 the rows and columns of an image using a key to complicate 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. This shows that the proposed algorithm is stable against a variety of attacks, including statistical and differential attacks (visual testing). The computations and coding were done using the capabilities of PYTHON. Furthermore, performance evaluation experimental results shows that Rubik's cube image encryption technique is accurate and secure for digital images.

REFERENCES

- [1]. Chen, L., Yin, H., Yuan, L., Machado, J. T., Wu, R., & Alam, Z. (2021). Double color image encryption based on fractional order discrete improved Henon map and Rubik's cube transform. *Signal Processing: Image Communication*, 97, 116363.
- [2]. Chepuri, S. (2017). An RGB image encryption using RSA algorithm. *International Journal of Current Trends in Engineering & Research (IJCTER)*, 3(3), 1-7.
- [3]. Devi. S. R., Rajarajeswari.V, Thenmozhi. K, RengarajanAmirharajan and PraveenkumarP.(2018). Henon and LFSR assisted key based encryption. *International Journal of Pure and Applied Mathematics*, 119(16), 455-460.
- [4]. Mousavi, M., & Sadeghiyan, B. (2021). A new image encryption scheme with Feistel like structure using chaotic S-box and Rubik cube based P-box. *Multimedia Tools and Applications*, 80(9), 13157-13177.
- [5]. Zhao, J., Zhang, T., Jiang, J., Fang, T., & Ma, H. (2022). Color Image Encryption Scheme Based On Alternate Quantum Walk and Controlled Rubik's Cube.
- [6]. Zhu, H., Dai, L., Liu, Y., & Wu, L. (2021). A three-dimensional bit-level image encryption algorithm with Rubik's cube method. *Mathematics and Computers in Simulation*, 185, 754-770.
- [7]. <https://www.textcompare.org/image/>

Gandhi Srushti, et. al. "Digital Image Encryption using Rubik's Cube Algorithm." *IOSR Journal of Mathematics (IOSR-JM)*, 18(4), (2022): pp. 16-25.