Image Encryption And Decryption Using Rubick's Cube Algorithm

Anitej Srivastava
19BCE0835
anitej.srivastava2019@vitstudent.ac.in

Rahul Agarwal
19BCE0720
rahul.agarwal2019@vitstudent.ac.in

Harsh Londhekar
19BCE0496
harshvivek.londhekar2019@vitstudent.ac.in

Anirudh A
19BCB0039
anirudh.a2019@vitstudent.ac.in

Rishabh Jain
19BCE0240
rishabh.jain2019a@vitstudent.ac.in

***Abstract*—In today's world, almost 80% of our lives have been digitized and in the near future everyone and everything will be connected to a digital system of the internet and its subsidiaries. This paves the way for a huge security concern in the form of privacy of information and data. To evade such privacy concerns various countermeasures, need to be incorporated. Image encryption and decryption is just an example of one such countermeasure. In recent times several algorithms have been put forth based on chaotic systems for encryption and decryption, however they offer very limited security. In this project a new encryption technique is discussed which will use the Rubik's Cube Algorithm. The algorithm will be applied on coloured images and the pixel values of red, blue and green will be in action. The pixel values will be used in the algorithm to provide encryption by scrambling them using the XOR operation. The decryption of the image will be the reverse of the encryption process with the help of the keys produced in the previous phase. On successful completion of the project, one will be able to perform full encryption and decryption.**

***Index Terms*—Image processing, cyber security, rubick's cube, rgb images, encryption decryption**

I. Introduction

Since the dawn of the computer age, security has been a primary goal of man. The advancements in the field of security have been duly noted and has been seen in the form of various security mechanisms and algorithms being developed. Cryptography has always played a major role and the combination of numbers and operations has proved to be an amazing tool to encapsulate user data. There are various cryptographic paradigms that involve ciphers based on mathematical tools. Since it all comes down to playing with numbers, the same tools have been used by researchers to provide encryption of images by manipulating pixel values. The pixel values of images are just numbers which can be scrambled, jumbled to make a new sequence and in turn a new distorted image. This project has a particular interest in providing image security using image processing techniques mixed with a security algorithm. We have derived inspiration from various research papers on Visual Cryptography and related techniques to create an improved and efficient code. The project uses the basic concept of visual cryptography. The simplest version of the visual secret sharing problem assumes that the message consists of a collection of black and white pixels and each pixel is handled separately. Our main motive is that we will be using a better and much simpler algorithm, that is, Rubik's Cube Principle based image encryption which includes an RBG color scheme. This will enable us to encrypt images in a colored pixels format instead of using outdated black and white scheme. The objective of project to produce the encrypted and decrypted version of the image that is taken as the input from the user. The encrypted image would be having a color scheme of red, blue and green. The code will decrypt the encrypted image and return the same image as the input by the user. Through the project we will be addressing the problem of encrypting and decrypting an image file by using Rubik's Cube Principle based image encryption which includes a RBG color scheme(red, blue, green). Through the project we will be implementing the Rubik's Cube Algorithm which is nothing but the various possible combinations of a Rubik's Cube and technique's related to solve the randomly jumbled cube. Encryption: In a similar way, we need to encrypt the image file by randomly arranging the bits and add a color scheme of red, blue and green to display the encrypted form of the image file. Decryption: For decrypting the image we need to solve the randomly ordered bits to the original order which will be possible by using the Rubik's Cube algorithm

II. Related Work

There are various techniques, methods and schemes revolving around image encryption and decryption. Waness et al. [8] has presented three particular methods of image encryption that are - symmetric, asymmetric and chaotic systems. Out of the three symmetric systems are rarely used, hence a deep study of chaotic system techniques and asymmetric algorithms have been made before selecting the choice of algorithm to be used in this project. Some valuable and resourceful information on chaotic systems is given below. [4][5] One of the chaotic system techniques dealt with use of a generalized logistic map for real time image processing [5]. Shah et al. [5] presents a scheme that makes use of encryption with an efficient permutation based technique based on a modular logistic map to bring down the size of the chaotic value

vector, required to permute a real-time image. The chaotic key sequence used in this technique changes the pixel values of a grayscale image. This optimises the chaotic key generation but is only extended on a grayscale image. Another chaotic system based encryption deals with double spiral scans and chaotic maps [4]. Tang et al. [4] presents the Double Spiral Scans which take a random scan of the image matrix and jumble those particular locations it scans twice. This results in randomization of the image and hence makes it distorted and encrypted. However if the entire image is based on the same pixel value then there will be no encryption despite the double scans. Another technique is by using basic image processing concepts [7][9][10]. They make use of XOR operations along with image matrix arithmetic operations. Ahmad et al. [9] makes use of XOR and the formation of the orthogonal matrix and skew tent map using image pixels for providing a secure cipher image. Astuti et al. [10] deals with steganography and gives a vivid description on steganography. It also uses bit plane slicing along with XOR operation to work on the image and provide security. Another technique discussed is really interesting as it revolves around using destructive techniques on images to provide for security [7]. Mivule et al. [7] presents a scheme that will add noise to the image matrix and perform certain arithmetic operations on the pixels to change the original image. This technique however would not be beneficial for the retrieval process as the computation would be very difficult. Another approach to encryption is one which is advanced and allows cloud features to come into play. Fu et al. [2] deals with the outsourcing of computation to a third party which provides image processing resources and thereby giving security. It deals with the application of an extended code base which helps other applications to function at the same computational complexity level without having to spend more on the computation of the image encryption algorithm. Another extension to this uses traditional image processing techniques combined with a homomorphic secret key to work on floating point numbers and provide an encrypted image [3]. Also, multiple images can be encrypted at once using ghost imaging. This deals with a multiple image encryption technique in which a group of plain images are encrypted. Each plain image is encrypted into an intensity vector by using the computational ghost vectoring scheme. Then all the vectors are superimposed on each other to result in a single ciphertext image [6]. Finally after a thorough examination and survey of all possible techniques, a new algorithm became the inspiration for this project's encryption and decryption mechanism. It deals with the Rubik's Cube algorithm [1]. This is a novel algorithm which can encrypt an image without any chaotic means and uses the asymmetric encryption-decryption ideology combined with XOR operations and certain arithmetic operations. The algorithm suggested only applies to gray scale images but with the help from multiple image encryption techniques as seen above my project aims to extend it to coloured images. The coloured images can be divided into three separate planes of red, blue and green pixel values. These pixel values will form three different image matrices which will be encrypted separately and then superimposed together to get the final single encrypted image as the output. For decryption, the reverse of the encryption process will be followed which means dividing the encrypted image into red, blue and green pixel matrices, performing decryption and then superimposing to get the original image back.

III. Methodology

Any image when provided as an input to the project should result in the generation of a new image which is distorted, unrecognisable and has minimal resemblance to the input image. Also, a file containing keys should be generated. When the distorted image is input in the project with the set of keys, the original image should be generated.

A. Image Pre-processing

In this module, a colored image is taken as an input. The image is loaded into the pre-processing module and the following changes happen. The colored image is divided into the primary color matrices ie the red, blue and green matrices. The pixel values of all the 3 matrices are obtained. Next, the encryption keys are set and stored according to the Rubik's Cube algorithm.

B. Image Encryption Module

In this module, the input image would be encrypted after pre-processing using the techniques mentioned in the research framework. The sum of the rows of all the red, blue and green values is calculated one by one. The sum is then stored separately for the three values after performing mod 2 operation. After performing that, a check is made if the modulus value is 0 or not. If the value is 0 then a right circular shift is made a certain number of times according to the keys generated in the pre-processing module else left circular shift. The same process is repeated for the green and blue values also. After processing the rows, we process the columns. The sum operation is applied and the modulus 2 values are stored for the red, blue and green matrix. After performing that, a check is made if the modulus value is 0 or not. If the value is 0 then a upshift is made a certain number of times according to the keys generated in the pre-processing module else downshift. After upshift and downshift and circular shifting, XOR operation is performed on the resultant red, blue and green matrix according to the key values.

C. Image Decryption Module

After following the image pre-processing module another time, the reverse of the encryption process is followed. The XOR operation is performed on the red, blue and green matrix obtained from the pre-processed stage. The XOR is performed according to the keys entered by the user during the decryption input phase. Now we process the columns of the modified red, blue and green matrices obtained. The sum operation is applied and the modulus 2 values are stored for the red, blue and green matrix. After performing that, a check is made if the modulus

value is 0 or not. If the value is 0 then a upshift is made a certain number of times according to the keys generated in the pre-processing module else downshift. After processing the columns we process the rows. The sum of the rows of all the red, blue and green values is calculated one by one. The sum is then stored separately for the three values after performing mod 2 operation. After performing that, a check is made if the modulus value is 0 or not. If the value is 0 then a right circular shift is made a certain number of times according to the keys generated in the pre- processing module else left circular shift. The same process is repeated for the green and blue values also. Finally the three matrices obtained undergo post processing to retrieve the original form of the image.

D. Image post-processing

After generating the modified pixel values for the red, blue and green image matrix, these are superimposed and combined together to form an encrypted image. The encrypted image is then stored and is free for use. Apart from this, keys are generated for decrypting the same in a txt file.

E. Key Generation

The image to be encrypted is taken as input from the user. This image is then processed as pixels. These pixels are stored into a multi-dimensional array. The number of rows of this array is stored into a variable say m and the number of columns is stored into another variable n. A variable alpha is set to 8 and then two vectors Kr and Kc are declared to represent the keys of rows and columns respectively. Kr vector is filled with m random integers in the Integer range and Kc is filled with n random integers in the Integer range using the randint(a,b) function in python.

F. Diagrams

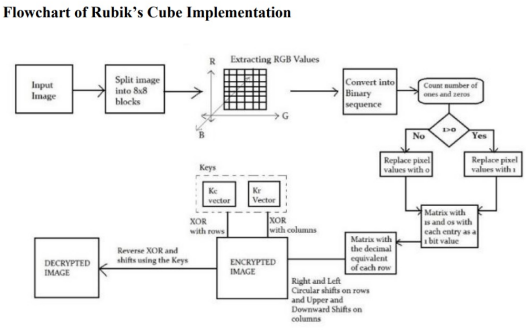

Fig. 1. System Architecture

G. Algorithm

Encryption
Decryption

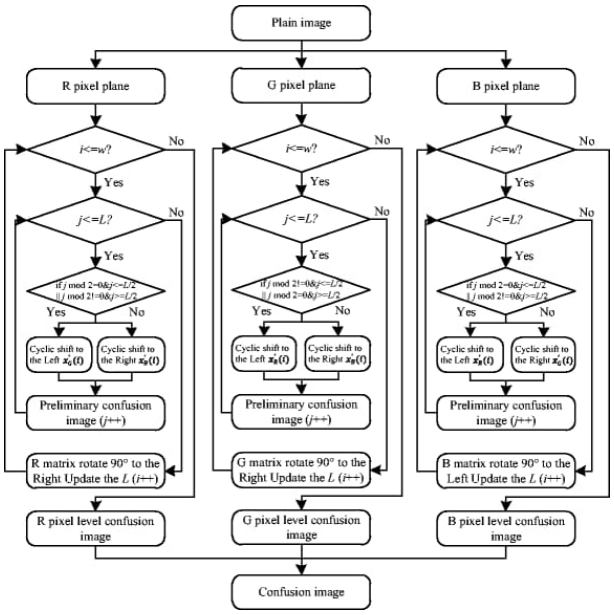

Fig. 2. Encryption and Decryption

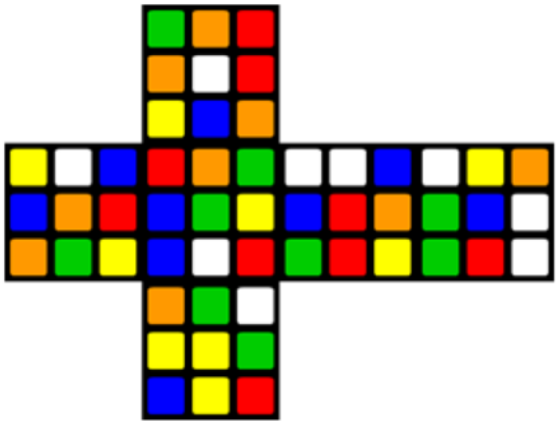

Fig. 3. Encrypted RGB Rubick's Cube

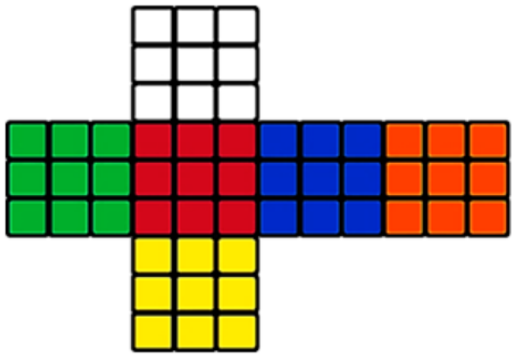

Fig. 4. Decrypted RGB Rubick's Cube

Let I_o represent an α-bit gray scale image of the size $M \times N$. Here, I_o represent the pixels values matrix of image I_o. The steps of encryption algorithm are as follows:

(1) Generate randomly two vectors K_R and K_C of length M and N, respectively. Element $K_R(i)$ and $K_C(j)$ Each take a random value of the set $\mathscr{A} = \{0, 1, 2, \ldots, 2^{\alpha} - 1\}$. Note that both K_R and K_C must not have constant values.
(2) Determine the number of iterations, $\mathrm{ITER}_{\max}$, and initialize the counter ITER at 0.
(3) Increment the counter by one: ITER = ITER + 1.
(4) For each row i of image I_o,
   (a) compute the sum of all elements in the row i, this sum is denoted by $\alpha(i)$

$$\alpha(i) = \sum_{j=1}^{N} \mathrm{I}_o(i, j), \quad i = 1, 2, \ldots, M. \tag{1}$$

   (b) compute modulo 2 of $\alpha(i)$, denoted by $M_{\alpha(i)}$,
   (c) row i is left, or right, circular-shifted by $K_R(i)$ positions (image pixels are moved $K_R(i)$ positions to the left or right direction, and the first pixel moves in last pixel.), according to the following:

$$\begin{aligned} &\text{if } \; M_{\alpha(i)} = 0 \longrightarrow \text{right circular shift} \\ &\text{else} \longrightarrow \text{left circular shift.} \end{aligned} \tag{2}$$

(5) For each column j of image I_o,
   (a) compute the sum of all elements in the column j, this sum is denoted by $\beta(j)$,

$$\beta(j) = \sum_{i=1}^{M} \mathrm{I}_o(i, j), \quad j = 1, 2, \ldots, N, \tag{3}$$

   (b) compute modulo 2 of $\beta(j)$, denoted by $M_{\beta(j)}$.
   (c) column j is down, or up, circular-shifted by $K_C(i)$ positions, according to the following:

$$\begin{aligned} &\text{if } \; M_{\beta(j)} = 0 \longrightarrow \text{up circular shift} \\ &\text{else} \longrightarrow \text{down circular shift.} \end{aligned} \tag{4}$$

Steps 4 and 5 above will create a scrambled image, denoted by I_{SCR}.

(6) Using vector K_C, the bitwise XOR operator is applied to each row of scrambled image I_{SCR} using the following expressions:

$$\begin{aligned} \mathrm{I}_1(2i-1, j) &= \mathrm{I}_{\mathrm{SCR}}(2i-1, j) \oplus K_C(j), \\ \mathrm{I}_1(2i, j) &= \mathrm{I}_{\mathrm{SCR}}(2i, j) \oplus \text{rot } 180\left(K_C(j)\right), \end{aligned} \tag{5}$$

where $\oplus$ and rot $180(K_C)$ represent the bitwise XOR operator and the flipping of vector K_C from left to right, respectively.

(7) Using vector K_R, the bitwise XOR operator is applied to each column of image I_1 using the following formulas:

$$\begin{aligned} \mathrm{I}_{\mathrm{ENC}}(i, 2j-1) &= \mathrm{I}_1(i, 2j-1) \oplus K_R(j), \\ \mathrm{I}_{\mathrm{ENC}}(i, 2j) &= \mathrm{I}_1(i, 2j) \oplus \text{rot } 180\left(K_R(j)\right). \end{aligned} \tag{6}$$

with rot $180(K_R)$ indicating the left to right flip of vector K_R.

(8) If $\mathrm{ITER} = \mathrm{ITER}_{\max}$, then encrypted image I_{ENC} is created and encryption process is done; otherwise, the algorithm branches to step 3.

Vectors K_R, K_C, and the max iteration number $\mathrm{ITER}_{\max}$ are considered as secret keys in the proposed encryption algorithm. However, to obtain a fast encryption algorithm it is preferable to set $\mathrm{ITER}_{\max} = 1$ (single iteration). Conversely, if $\mathrm{ITER}_{\mathrm{MAX}} > 1$, then the algorithm is more secure because the key space is larger than for $\mathrm{ITER}_{\mathrm{MAX}} = 1$. Nevertheless, in the simulations presented in Section 3, the number of iterations $\mathrm{ITER}_{\max}$ was set to one.

The decrypted image, I_o, is recovered from the encrypted image, I_{ENC}, and the secret keys, K_R, K_C, and $\mathrm{ITER}_{\max}$ as follows in the following.

(1) Initialize ITER = 0.
(2) Increment the counter by one: ITER = ITER + 1.
(3) The bitwise XOR operation is applied on vector K_R and each column of the encrypted image I_{ENC} as follows:

$$\begin{aligned} \mathrm{I}_1(i, 2j-1) &= \mathrm{I}_{\mathrm{ENC}}(i, 2j-1) \oplus K_R(j), \\ \mathrm{I}_1(i, 2j) &= \mathrm{I}_{\mathrm{ENC}}(i, 2j) \oplus \text{rot } 180\left(K_R(j)\right), \end{aligned} \tag{7}$$

(4) Then, using the K_C vector, the bitwise XOR operator is applied to each row of image I_1:

$$\begin{aligned} \mathrm{I}_{\mathrm{SCR}}(2i-1, j) &= \mathrm{I}_1(2i-1, j) \oplus K_C(j), \\ \mathrm{I}_{\mathrm{SCR}}(2i, j) &= \mathrm{I}_1(2i, j) \oplus \text{rot } 180\left(K_C(j)\right). \end{aligned} \tag{8}$$

(5) For each column j of the scrambled image I_{SCR},
   (a) compute the sum of all elements in that column j, denoted as $\beta_{\mathrm{SCR}}(j)$:

$$\beta_{\mathrm{SCR}}(j) = \sum_{i=1}^{M} \mathrm{I}_{\mathrm{SCR}}(i, j), \quad j = 1, 2, \ldots, N, \tag{9}$$

   (b) compute modulo 2 of $\beta_{\mathrm{SCR}}(j)$, denoted by $M_{\beta_{\mathrm{SCR}}(j)}$,
   (c) column j is down, or up, circular-shifted by $K_C(i)$ positions according to the following:

$$\begin{aligned} &\text{if } \; M_{\beta_{\mathrm{SCR}}(j)} = 0 \longrightarrow \text{up circular shift} \\ &\text{else} \longrightarrow \text{down circular shift.} \end{aligned} \tag{10}$$

(6) For each row i of scrambled image I_{SCR},
   (a) compute the sum of all elements in row i, this sum is denoted by $\alpha_{\mathrm{SCR}}(i)$:

$$\alpha_{\mathrm{SCR}}(i) = \sum_{j=1}^{N} \mathrm{I}_{\mathrm{SCR}}(i, j), \quad i = 1, 2, \ldots, M, \tag{11}$$

   (b) compute modulo 2 of $\alpha_{\mathrm{SCR}}(j)$, denoted by $M_{\alpha_{\mathrm{SCR}}(j)}$,
   (c) row i is then left, or right, circular-shifted by $K_R(i)$ according to the following:

IV. Results

A. Tools Used

- Python
- Flask
- Visual Studio Code
- PIL Library
- numpy

B. Techniques Used

1) Up Circular Shift
   This is a user defined function that performs up circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically upwards in the column.
2) Down Circular Shift
   This is a user defined function that performs down circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically downwards in the column.
3) Bit Rotation
   This function is used to reverse the bits that is entered as parameter to this function. It uses basic string reversal technique which is starting from the end of the string and reaching to first bit there by getting the bits of the string in reverse order.

C. Outputs and Discussion

1) Encryption
   First, we need to generate two vectors Kr and Kc of length M and N and they are initialized with random values. Now determine the total number of iterations and then implement the algorithm and the code as given.
2) Decryption
   Load the image and convert it into matrix containing the RGB values of the pixels using im.load(). Extract the RBG Values into 3 separate lists using nested for loop : outer for traversing pixels and inner for accessing the

R,G,B value. Declare m and n as the number of pixels in row and columns respectively. INPUT the values of Kr and Kc. Follow the algorithm as given.

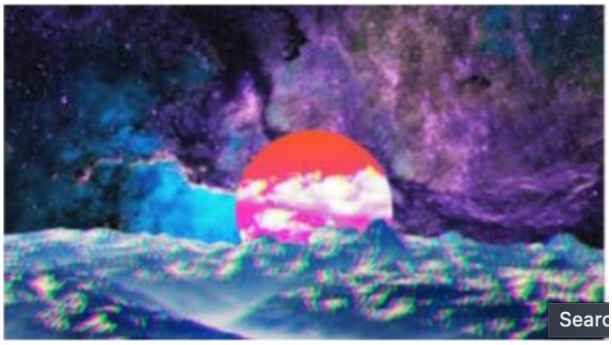

Fig. 5. Original Image

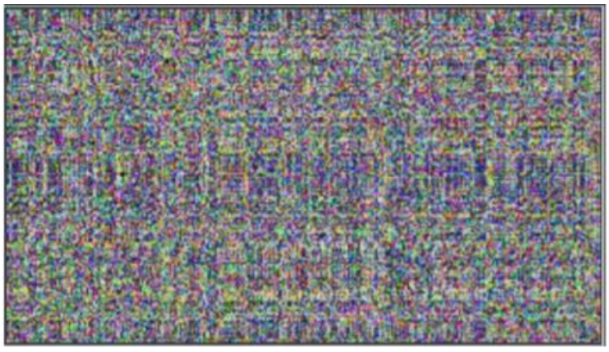

Fig. 6. Encrypted Image

D. Running the Project

On running the encrypt.py file in the terminal, the keys are displayed and the encrypted image along with the keys.txt file is created.

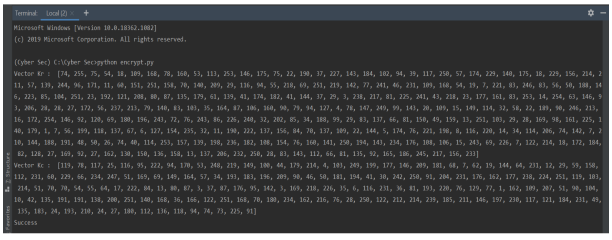

Fig. 7. Generated Keys

The keys.txt file containing the keys required for decryption.

On running the decrypt.py file in terminal to decrypt the encrypted image, we first need to input the keys.txt file and enter it into the interface and we obtain the decrypted image.

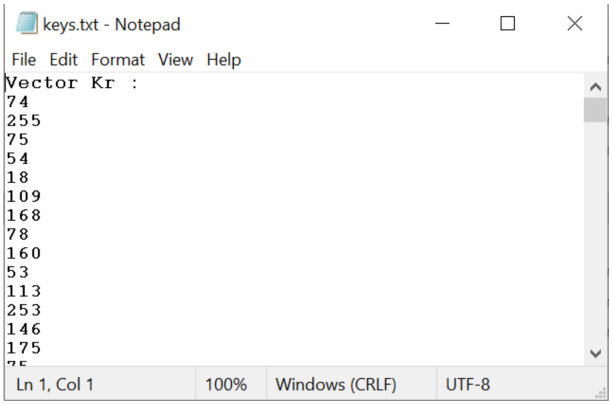

Fig. 8. File keys.txt

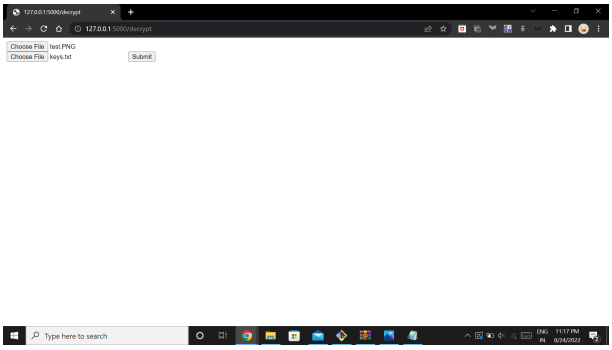

Fig. 9. Decrypting the image

V. Conclusion

Our project successfully encrypts an image, generates the keys which are further used for decryption. On decryption the user can see the initial image as it is. The image on encryption or decryption does not get corrupted and hence there is no loss of data. This project can be extended to protect files of other formats such word, excel, ppt etc. by converting the file to image format and then using the project to encrypt and decrypt the same. In future we can write code to convert a non-image file into an image and protect it using our project.

The proposed system and research framework shows that using the Rubik's Cube algorithm one can encrypt any coloured image using the red, blue and green pixel values of the image. By following the reverse procedure, the image can be decrypted. The proposed methodology consisted of simple operations based on arithmetic and image processing. This can enable the system to process in an optimized environment as the encryption and decryption process does not take much time and space complexity. The files and outputs generated do not have a significant space requirement as compared to the input image. Also, the encrypted image is very difficult to decrypt with Brute-Force mechanism hence a high security parameter is ensured.

The proposed system can be developed on various platforms and the implemented across various applications. The process can be used for protecting files and information on servers. The project can be extended to protect data hosted on online platforms. The system can be scaled onto different compiling languages which are faster than python, to increase the speed

of computing.

REFERENCES

[1] Cheng, Shuli, Liejun Wang, Naixiang Ao, and Qingqing Han, "A Selective Video Encryption Scheme Based on Coding Characteristics", 2020.

[2] B. Guan, D. Xu and Q. Li, 'An Efficient Commutative Encryption and Data Hiding Scheme for HEVC Video', 2020.

[3] Shah, Asif Parah, Shabir Rashid, Mamoon Elhoseny, Mohamed, "Efficient image encryption scheme based on generalized logistic map for real time image processing", 2020.

[4] Zhenjun Tang,1 Ye Yang,1 Shijie Xu,1 Chunqiang Yu,2 and Xianquan Zhang Guangxi, "Image Encryption with Double Spiral Scans and Chaotic Maps" , 2019.

[5] William Fu, Raymond Lin, Daniel Inge, "Fully Homomorphic Image Processing" , 2018.

[6] Yang, Pan Gui, Xiaolin An, Jian Tian, Feng, "An efficient secret key homomorphic encryption used in image processing service" , 2017.

[7] Jawad Ahmad Muazzam A. Khan Khattak Jan Sher Khan Fawad Ahmed, "A novel image encryption scheme based on orthogonal matrix, skew tent map, and XOR operation", 2018.

[8] Manju Kumari, Shailender Gupta Pranshul Sardana, "A Survey on the Image Encryption Methods" , 2017.

[9] Zhenjun Tang, Ye Yang, "Image Encryption with Double Spiral Scans and Chaotic Maps" , 2019.