International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

# IMAGE ENCRYPTION BASED ON RUBIK'S CUBE PRINCIPLE

[1]MR. K. RAGHAVENDAR, [2]K. SATHISH KUMAR, [3]G. SANTOSH KUMAR, [4]SRIGADHA KRISHNA SAI

[1](Assistant Professor), CSE. Teegala Krishna Reddy Engineering College Hyderabad.

[2,3,4]B, tech , scholar , CSE. Teegala Krishna Reddy Engineering College Hyderabad.

## ABSTRACT

Today almost all digital services like internet communication, medical and military imaging systems, multimedia system requires reliable security in storage and transmission of digital images. Due to faster growth in multimedia technology, internet and cell phones, there is a need for security in digital images. Therefore, there is a need for image encryption techniques in order to hide images from such attacks. The proposed system is based on Rubik's Cube Principle in order to hide images. Such encryption technique helps to avoid intrusion(third-party) attacks.In the past few years, several encryption algorithms based on chaotic systems have been proposed as means to protect digital images against cryptographic attacks. These encryption algorithms typically use relatively small key spaces and thus offer limited security, especially if they are one-dimensional. Rubik's Cube Principle is used to secure data from unauthorized user. An image is given as input to encryption algorithm which gives encrypted output. This encrypted output is given as input to decryption algorithm and original image is regained as output. This encryption principle not only can achieve good and prefect hiding ability but also can resist exhaustive attack, statistical attack, and differential attack.

## 1. INTRODUCTION

The end of the 20th century was marked by an extraordinary technical revolution from analog to numerical as documents and equipments became increasingly used in various domains. However, the advantages of the digital revolution were not achieved without drawbacks such as illegal copying and distribution of digital multimedia documents. To meet this challenge, researchers were motivated more than ever to protect multimedia documents with new and efficient document protection techniques. In this context, different techniques have been introduced such as encryption and digital watermarking. The first one consists in transforming multimedia documents using an algorithm to make it unreadable to anyone except for the legitimate users. The second one consists of embedding digital watermarks into multimedia documents to guarantee the ownership and the integrity of the digital multimedia contents. Traditional image encryption algorithms such as private key encryption standards (DES and AES), public key standards such as Rivest Shamir Adleman (RSA), may not be the most desirable candidates for image encryption, especially for fast and real-time communication applications. These encryption schemes can be classified into different categories such as value transformation, pixels position permutation, and chaotic systems. Some encryption schemes based on permutation had already been found insecure against the ciphertext-only, due to the high information redundancy, and it is quite understandable since the secret permutations can be recovered by comparing the plaintexts and the permuted ciphertexts. Generally, chaos-based image encryption algorithms are used more often than others but require high computational cost.

## 2. LITERATURE REVIEW

A Survey of Image Encryption Algorithms Data Encryption Standard (DES):

PSYCHOLOGY AND EDUCATION (2021) 57 (9) :3077-3081ISSN:003330773078

www.psychologyandeducation.net Data Encryption Standard (DES) is one among earliest block ciphers developed. Even though the processes for encryption and decryption include the number of rounds, the DES security mechanism is breakable by many ways. Brute force attack, known-plain text attacks and chosen plain text attacks are the most common approaches. International Data Encryption Algorithm (IDEA):

The encryption and decryption structures are similar and use eight full rounds plus an additional half-round, making a total of 8.5 rounds. IDEA is vulnerable to various attacks like narrow-bicliques attack and man-in the-middle attack. Blowfish Algorithm: Blowfish is a symmetric-key block cipher algorithm. Its main component is a Feistel network, iterating 16 times. Blowfish has small block size, lar4ger files are not recommended to be encrypted.

Advanced Encryption Standard (AES): AES utilizes a substitution and permutation network structure, while the previously widely used DES was based on Feistel network.

Different encryption and decryption processes uses similar byte substitution, shift row, mix column, and add round key steps.

There are three different kinds of AES algorithm. These had an equal block size of 128-bit but had different key sizes of 128, 192, 256-bits signifying increase in security strength.With increase in bits AES gets vulnerable to full brute force attack. Triple Data Encryption Standard (TDES): Triple Data Encryption Standard (TDES) is a symmetric-key block cipher. The algorithm uses the DES algorithm three times in encryption, decryption, and key generation processes. However, Triple DES is more secure than DES, but it is vulnerable to meet-in-the-middle attack and block collision attacks.

## 3. SYSTEM DESIGN

### 3.1 SYSTEM ARCHITECTURE

Operating System An Operating system (OS) is a software which acts as an interface between the end user and computer hardware. Every computer must have at least one OS to run other programs. An application like Chrome, MS Word, Games, etc needs some environment in which it will run and perform its task. The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible for the user to use any computer or mobile device without having an operating system

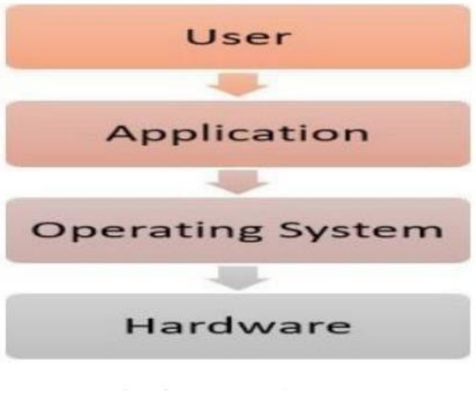


Figure:3.1 Operating System

**Features of operating system**

a) Protected and supervisor mode

b) Allows disk access and file systems Device drivers Networking Security

c) Program Execution

d) Memory management Virtual Memory Multitasking

e) Handling I/O operations Here we used Ubuntu operating system.

## PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by "GUIDO VAN ROSSUM" and first released in 1991. Python is a powerful generalpurpose programming language. It is used in web development, data science, creating software prototypes, and so on. Fortunately for beginners, Python has simple easy-to-use syntax. This makes Python an excellent language to learn to program for beginners.

Figure:3.2 Python

Python is largely used for developing data science and machine learning models. It provides certain libraries for various features like extraction, splitting, Calculation, etc. Python syntax is very easy to use and understandable. The length of the code is reduced by using python programming language.

## Advantages of python

a) Extensive support libraries

b) Open source and community development

c) User-friendly data structures

d) Productivity and speed

## NUMPY

NumPy (Numerical Python) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. NumPy is open-source softwareand has many contributors.

Figure:3.3 Numpy

It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

## 3.2ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control.

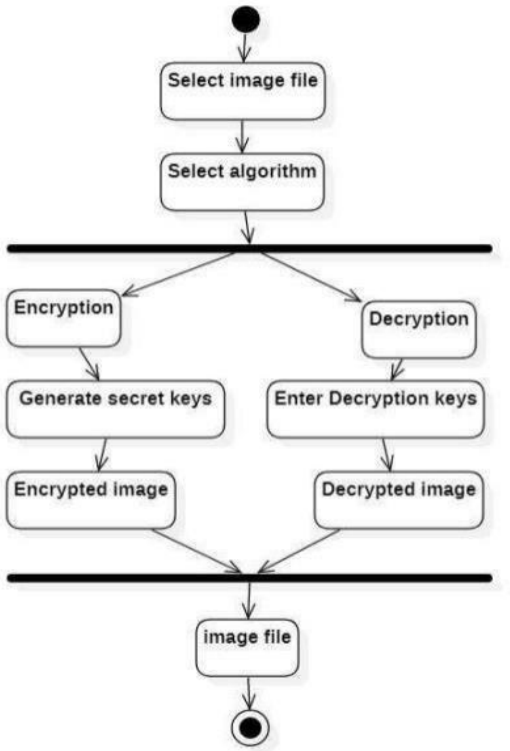
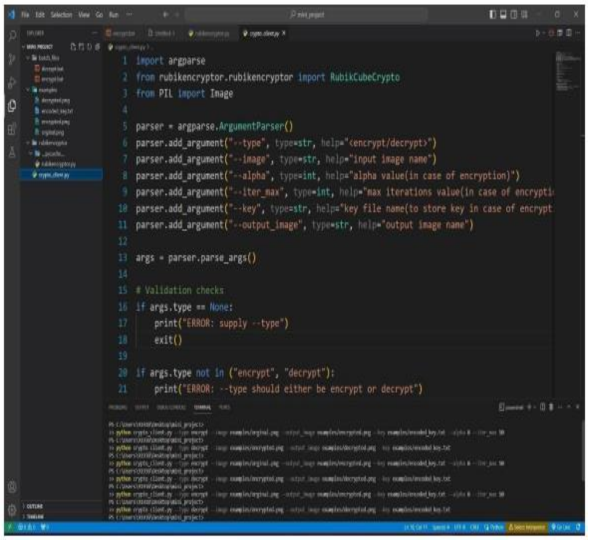


Figure:3.2 Activity Diagram

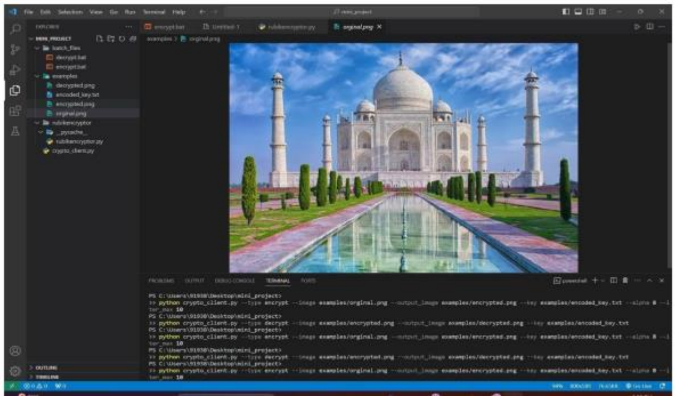
## 4.OUTPUT SCREENS



```
import argparse
from rubikencryptor.rubikencryptor import RubikCubeCrypto
from PIL import Image

parser = argparse.ArgumentParser()
parser.add_argument("--type", type=str, help="<encrypt/decrypt>")
parser.add_argument("--image", type=str, help="input image name")
parser.add_argument("--alpha", type=int, help="alpha value(in case of encryption)")
parser.add_argument("--iter_max", type=int, help="max iterations value(in case of encrypti
parser.add_argument("--key", type=str, help="key file name(to store key in case of encrypt
parser.add_argument("--output_image", type=str, help="output image name")

args = parser.parse_args()

# Validation checks
if args.type == None:
    print("ERROR: supply --type")
    exit()

if args.type not in ("encrypt", "decrypt"):
    print("ERROR: --type should either be encrypt or decrypt")
```

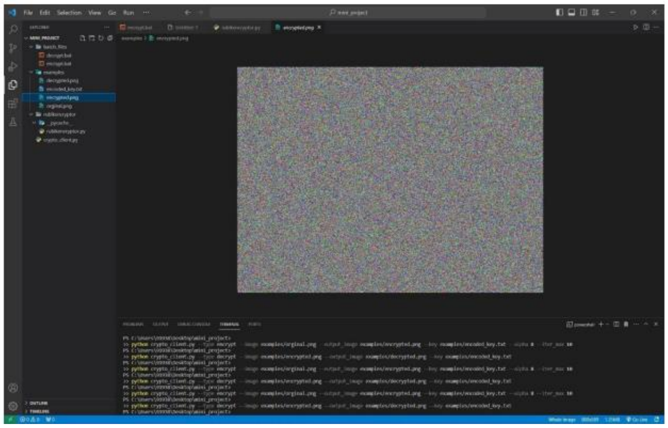
```
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
```

Figure:4.1 Source Code



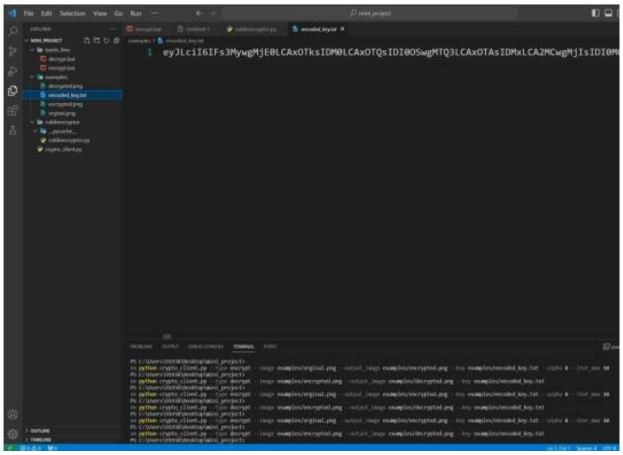
```
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --i
ter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --i
ter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --i
ter_max 10
```

Figure:4.2 Original image



```
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
```

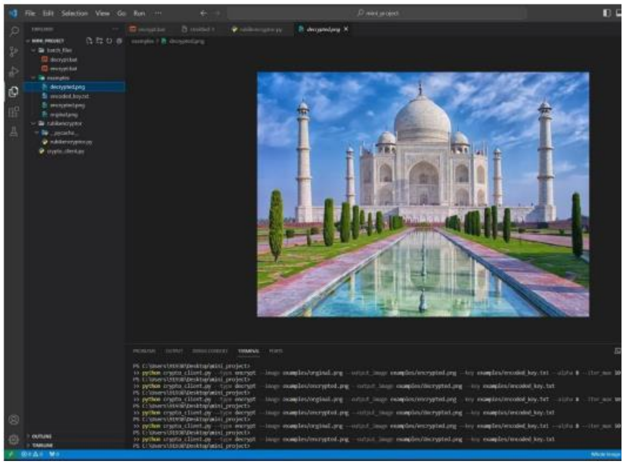
Figure: 4.3 Encrypted image



```
1  eyJLciI6IFs3MywgMjE0LCAxOTksIDM0LCAxOTQsIDI0OSwgMTQ3LCAxOTAsIDMxLCA2MCwgMjIsIDI0M
```

```
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
```

Figure:4.4 Encoded key



```
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type encrypt --image examples/orginal.png --output_image examples/encrypted.png --key examples/encoded_key.txt --alpha 8 --iter_max 10
PS C:\Users\91930\Desktop\mini_project>
>> python crypto_client.py --type decrypt --image examples/encrypted.png --output_image examples/decrypted.png --key examples/encoded_key.txt
PS C:\Users\91930\Desktop\mini_project>
```

Figure:4.5 Decrypted image

## 5. CONCLUSION

In this project, a novel image encryption algorithm is proposed. This algorithm is based on the principle of Rubik's cube to permute image pixels. To confuse the relationship between original and encrypted images, the XOR operator is applied to odd rows and columns of image using a key. The same key is flipped and applied to even rows and columns of image. The proposed image encryption algorithm is highly secure. It is also capable of fast encryption/decryption which is suitable for real-time Internet encryption and transmission applications.

## 6. FUTURE ENHANCEMENTS

We are really enthusiastic about the vast future possibilities that our project has to offer. Possible improvements include encrypting and decrypting the videos both in black & white and color by extracting each frame and encrypting the images simultaneously. We know that all the images have sound. So, it can plan to encrypt frames and sound simultaneously. Finally, we can also create an app which can contain all of the above activities, with two people having the app; one will become the sender and other the receiver at a time, based on the requirement of either of two. By taking this model into consideration we can also develop an android application on it which encrypts the images and videos provides security. With the help of PHP Xammp we can store images in Xampp mysql database and retrieve form the database.

## 7. REFERENCES

1.Cryptology,History(http://www.faqs.org/espionage/Cou-De/Cryptology History)

2. C. K. Huang, H. H. Nien, S. K. Changchien, and H. W. Shieh, "Image encryption with chaotic random codes by grey relational grade and Taguchi method," Optics Communications, vol. 280, no. 2, pp. 300–310, 2004.

3. G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," Chaos, Solitons and Fractals, vol. 21, no. 3, pp. 749–761, 2014.

4. Q. Guo, Z. Liu, and S. Liu, "Color image encryption by using Arnold and discrete fractional random transforms in IHS space," Optics and Lasers in Engineering, vol. 48, no. 12, pp. 1174–1181, 2010

5. Z.-L. Zhu, W. Zhang, K.-W. Wong, and H. Yu, "A chaos-based symmetric image encryption scheme using a bit-level permutation," Information Sciences, vol. 181, no. 6, pp. 1171–1186, 2011

6. Y. Wang, K. W. Wong, X. Liao, and G. Chen, "A new chaos-based fast image encryption algorithm," Applied Soft Computing Journal, vol. 11, no. 1, pp. 514–522, 2011.

7. "An Improved Secure Image Encryption Algorithm Based on Rubik's Cube Principle and Digital Chaotic Cipher", Adrian-Viorel Diaconu and Khaled Loukhaoukha, Mathematical Problems in Engineering, Volume 2013, Article ID 84839.