1. INTRODUCTION

1.1 BACKGROUND

Computer has become an essential device now a days. The main use of computer is to store data and send it from one location to other. The information that is shared must be transferred in a secured manner. To ensure secured transmission of information, data is encrypted to unreadable formats by an unauthorized person. Cryptography is the science of information security which has become a very critical aspect of modern computing systems towards secured data transmission and storage. The exchange of digital data in cryptography results in different algorithms that can be classified into two cryptographic mechanisms: symmetric key in which same key is used for encryption and decryption and asymmetric key in which different keys are used for encryption and decryption.

Images are broadly used in numerous processes. As a result, the safety of image data from unauthorized access is crucial at the hands of user. Image encryption plays a significant role in the field of information hiding. Image hiding or encryption methods and algorithms ranges from simple spatial domain methods to more complicated and reliable frequency domain. Image Encryption Using Rubik's Cube Based Algorithm is the process to transform the image securely so that no unauthorized user can be able to decrypt the image. Image encryption have applications in many fields including the internet communication, transmission, medical imaging etc.

First, in order to scramble the pixels of gray-scale original image, the principle of Rubik's cube is deployed which only changes the position of the pixels. Using two random secret keys, the bitwise XOR is applied into the rows and columns. These steps can be repeated until the number of iterations is not reached. Numerical simulation has been performed to test the validity and the security of the proposed encryption algorithm.

1.2 PROBLEM DEFINITION

Information security is the most common word uttered by any man, any device or any peripheral since past two centuries. Protection from malicious sources has become a part of the invention or the discovery cycle. Myriad methods of protection are used ranging from a simple authentication password to most complex Cryptography. The advancements of digital revolution were not achieved without drawbacks such as illegal copying and distribution of digital multimedia documents. In order to provide data security and protection, different encryption methods must be used.

1.3 PURPOSE

This is the proposal for the design and development of software named as “Secure Image Encryption Using Algorithm Based On Rubik’s Cube Principle”. It applies special mathematical algorithm and keys to transform digital image into chiper code before they are transmitted and decrypts using application of mathematical algorithms and keys to get back the original data from the chiper code. The goal is to provide authentication of users and integrity, accuracy and safety of data resources.

2. OBJECTIVE

The main objective of our project is as follows:

- To provide security of the image-based data with the help of suitable key and protect the image from illegal copying and distribution.

3. LITERATURE REVIEW

Evolution of Image Encryption and Decryption

First, we want to review the history of image encryption and decryption and how it came to be a subject of interest. Here, we have listed the significant encryption and decryption techniques used in each era and how they have changed throughout human history.

Historical Cryptography

The earliest known text containing cryptography came from Egypt in the form of hieroglyphics. These texts were dated to be as old as 4000 years old. Nobody knows why Egyptian used such encryptions but scientists assume it was done so as to show that they can write a language that is of higher level than common people or to make it look formal similar to modern day legal documents. In Greece, in about 500 B.C., Spartans used Scytale which was a device to send or receive secret messages. The device was a cylinder in which a narrow strip of parchment was wound and the message was written length-wise. Once it was unwound the strip was unreadable and could only be read with the help of a similar cylinder.2500 years ago not many people could read or write so this technique was quite effective but nowadays it could easily be deciphered. 2000 years ago, the earliest use of cryptography was seen and used by Julius Caesar. He devised a technique to send secret messages as at that time messengers were captured by enemies who could steal the message. But, Caesar would shift the texts by a specific number so that only the person who new the cipher code could decipher the texts, giving Julius Caesar a huge advantage against his enemies. In the 1400's, Leon Battista Alberti devised an encryption system using Cipher disk. This device allowed for many methods of substitution. In the 1500's, following the work of Alberti, Blaise De Vigenere created a cipher which came to be known as Vigenere Cipher. In the late 1700's, Thomas Jefferson devised a cipher system similar to the Vigenere cipher but with higher security. The encrypted message was present on the wheels and the ciphertext could have been any other plain text present in the wheel. Similar to Alberti, Jefferson never developed his encryption system. "During the early 1900's, the United States Army reinvented Jefferson's Wheel Cipher without any knowledge about Jefferson's invention. Jefferson was over a hundred years before his time. The United States Army used this system from 1923to 1942 (Thinkquest.org 1999)." In

World War II, many encryption methods were developed such as Enigma Encryption Machine which was virtually unbreakable with brute force methods and Purple developed by the Japanese whose encryption technique gave them a huge advantage in war but later on was deciphered [1].

Modern Cryptography

In the modern age of internet and technology data security is a huge field and an equally immense challenge as highly sensitive data are stored in the digital realm. Personal info such as bank information, social media accounts, etc. all are in threat from phishing, pharming and spyware. To counteract such threats many ingenious techniques have been developed including image encryption.

Some of the previous image encryption methods developed include chaotic systems, pixel position permutation and value transformation. Huang, C.K., Nien, H.H proposed a novel pixel shuffling method which used chaotic systems to generate chaotic sequences as encryption codes [2]. Chen, G., Mao, Y., Chui, C.K. generalize a two-dimensional cat map to a 3D one for a real-time secure symmetric scheme [3]. Guo et al. proposed a color image encryption method using discrete fractional random transforms (DFRNT) and Arnold transform (AT) in which DFRNT encrypts the intensity component while AT encrypts hue and saturation components [4]. Zhu et al. proposed a permutation method to confuse and diffuse the gray-scale image at the bit level, which changes the position of the pixel and modifies its value also using the Arnold cat map to permute the bits and the logistic map to further encrypt the permuted image [5]. Wang, Y., Wong, K.-W., Liao, X., Chen, G. in their research paper proposed image encryption algorithm combining permutation and diffusion where the original image is partitioned into blocks and then spatiotemporal chaos is applied to shuffle the block and generate pseudorandom sequence [6].

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Our proposed system of Image Encryption application functional requires

- The application is able to ask user to input keys ‘kr’ and ‘kc’ for decryption process to take place.
- The application is able to send the keys through email provided by users after encryption.
- The process of both encryption and decryption is incredibly fast.
- The image file size is not significantly affected to prevent doubt for unauthorized user.
- The image is transmitted according to user preference safely after being encrypted.

4.2 NON-FUNCTIONAL REQUIREMENTS

Our proposed system of Image Encryption application non-functional requirements are

4.2.1 Accessibility

Accessibility requirements refers to the system implementation to as many people as possible. The application is very simple and easy to use provided the image file is compatible to the system.

4.2.2 Efficiency

Efficiency refers to competency in performance. System performance like time constraints, memory consumption or disk space is important to make sure the efficiency of application. This requirement can be accomplished by having a quick response time and low process power consumption.

4.2.3 Reliability

Reliability requirement refers to as the ability of system to perform and maintain its function in routine situation and vice versa. The application should be able to decrypt the transmitted image into its original image provided the correct key is entered.

4.3 SOFTWARE REQUIREMENTS

Here we specify the types of software that is used within the system.

4.3.1 Photoshop

4.3.2 Python 3.7

4.3.3 SQLite

4.3.4 Atom IDE

4.3.5 Microsoft Windows 10 OS

4.3.1 Photoshop

Adobe photoshop is a raster graphics editor developed and published by Adobe Inc. for Windows and Mac OS. In our project, photoshop will be used to design different user interfaces, profile design and so on. Similarly, for better background image and designing logo of our application, photoshop will be used.

4.3.2 Python 3.7

Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology.

4.3.3 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

4.3.4 Atom IDE

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in Coffee Script and Less.

4.3.5 Microsoft Windows 10 OS

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1, and was released to manufacturing on July 15, 2015.

4.4 FEASIBILITY STUDY

A feasibility study is carried out to select the system that meets the performance requirements. A feasibility study aims to objectively and rationally uncover the strength and weaknesses of an existing system. A feasibility study evaluates the project’s potential for success.

4.4.1 ECONOMIC FEASIBILITY

Economic feasibility is the analysis of the project’s cost and revenues in an effort to determine whether or not it is logical and possible to complete. We have conducted economic feasibility and thus concluded that our system is economically feasible.

4.4.2 TECHNICAL FEASIBILITY

Technical feasibility is to determine whether the system has technical expertise to handle completion of project. It is an evaluation of the hardware and software and how it meets the need of the proposed system. Since our application runs on minimum requirements, our system is technically feasible.

4.4.3 OPERATIONAL FEASIBILITY

Operational feasibility is mainly related to human organization and political aspects. It refers to evaluation which analyses how well a system operates. The system requires windows Os to run the app.

4.4.4 TIME FEASIBILITY

A time feasibility study will take into account the period in which the project is going to take up to its completion. Our project is expected to complete in 2 months’ time.

5. SYSTEM DESIGN AND ARCHITECTURE

5.1 USE CASE DIAGRAM

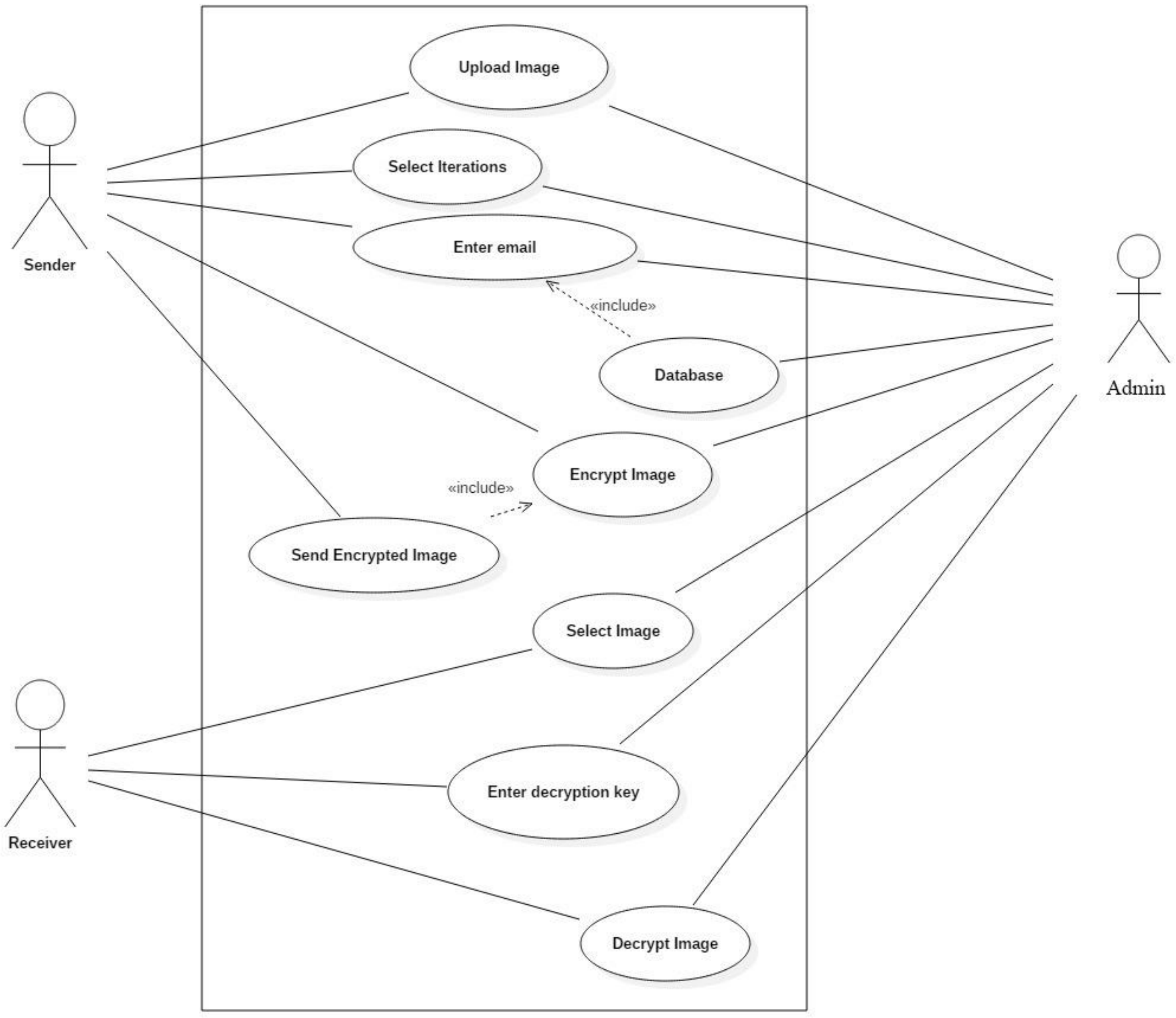


Fig 5.1 Use Case Diagram

5.2 LEVEL 0 DFD

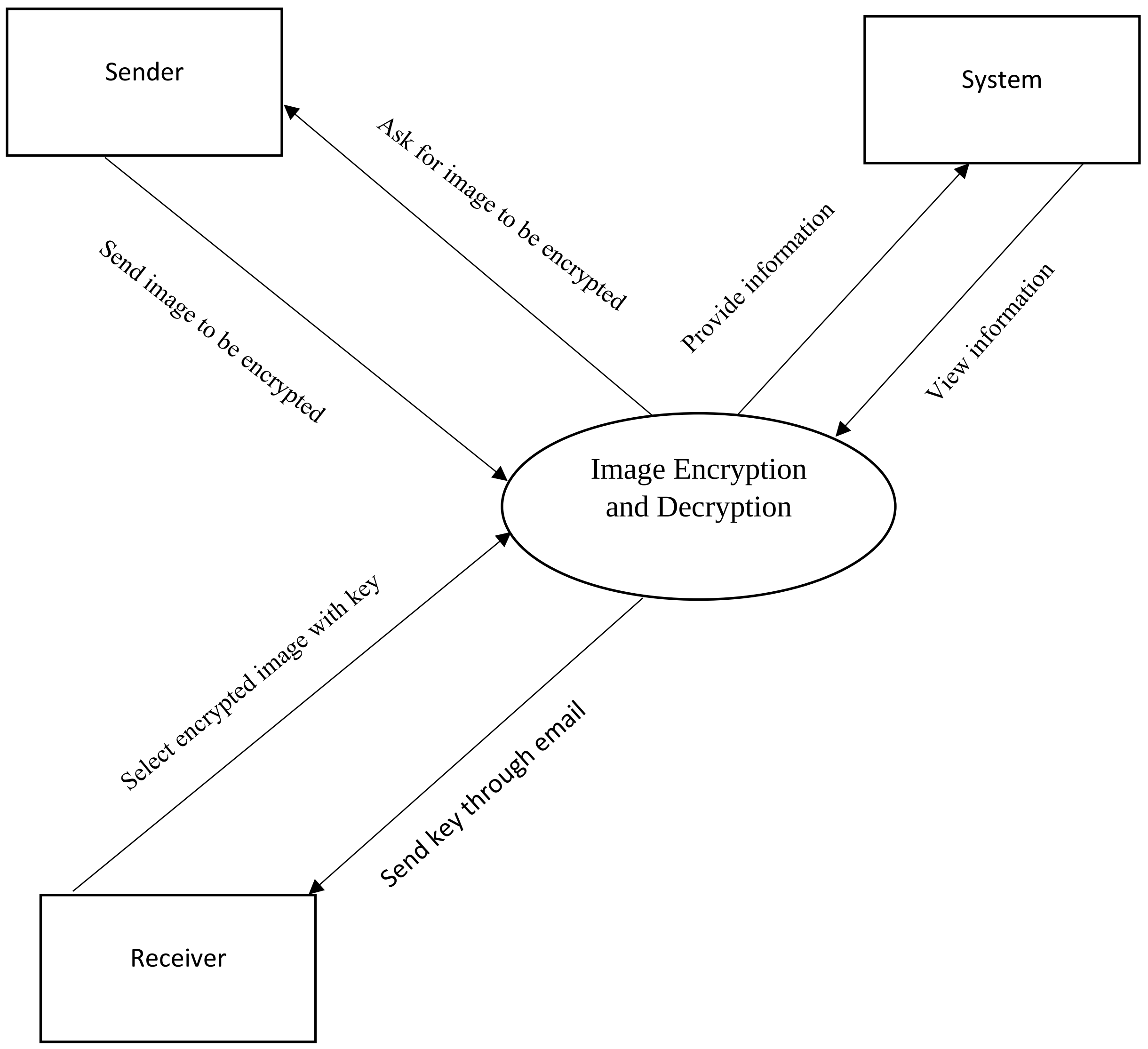

Sender
System
Ask for image to be encrypted
Send image to be encrypted
Provide information
View information
Image Encryption and Decryption
Select encrypted image with key
Send key through email
Receiver

Fig 5.2: Level 0 DFD

5.3 LEVEL 1 DFD

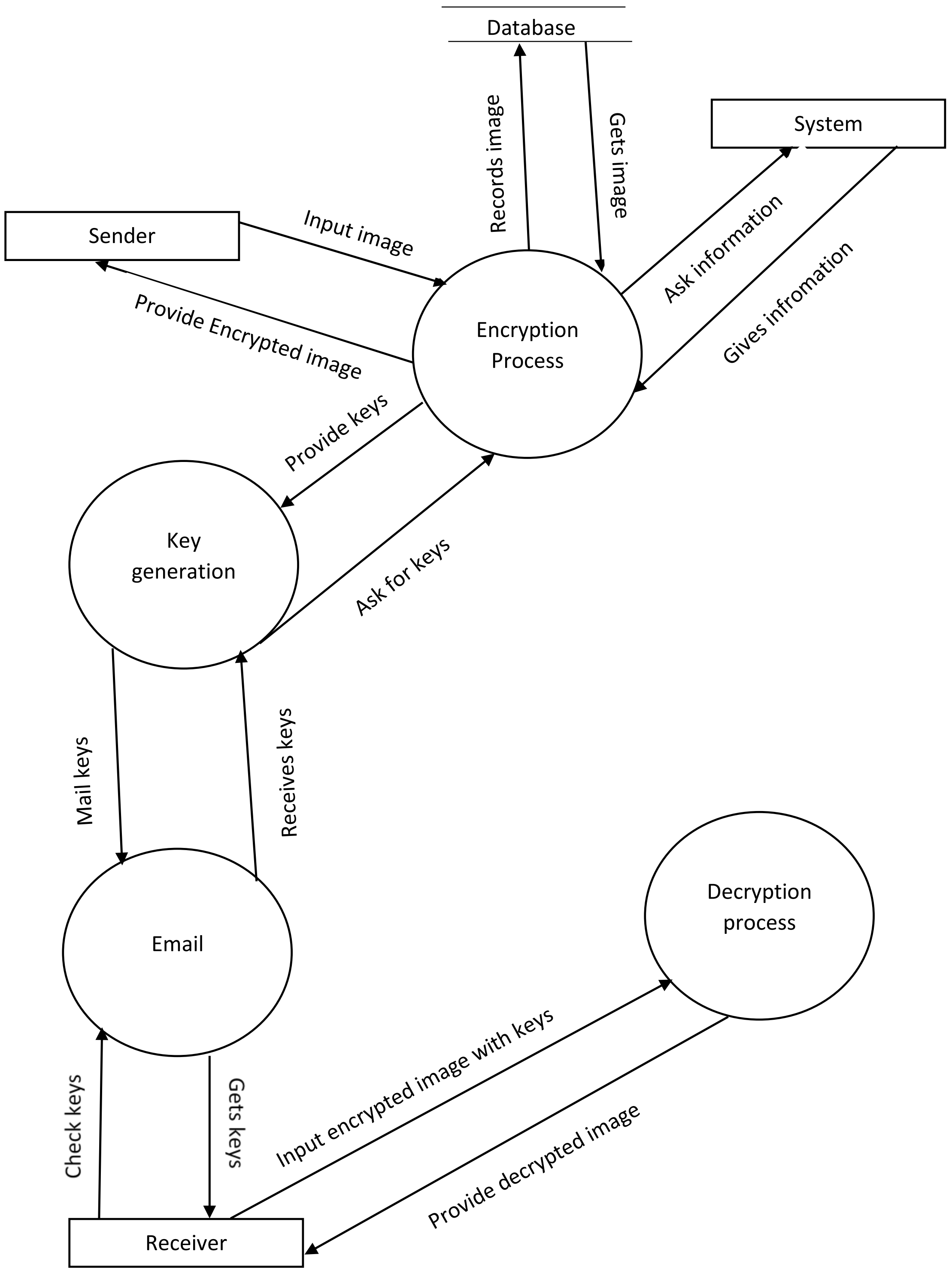


Fig 5.3: Level 1 DFD

5.4 Entity Relationship Diagram (ERD):

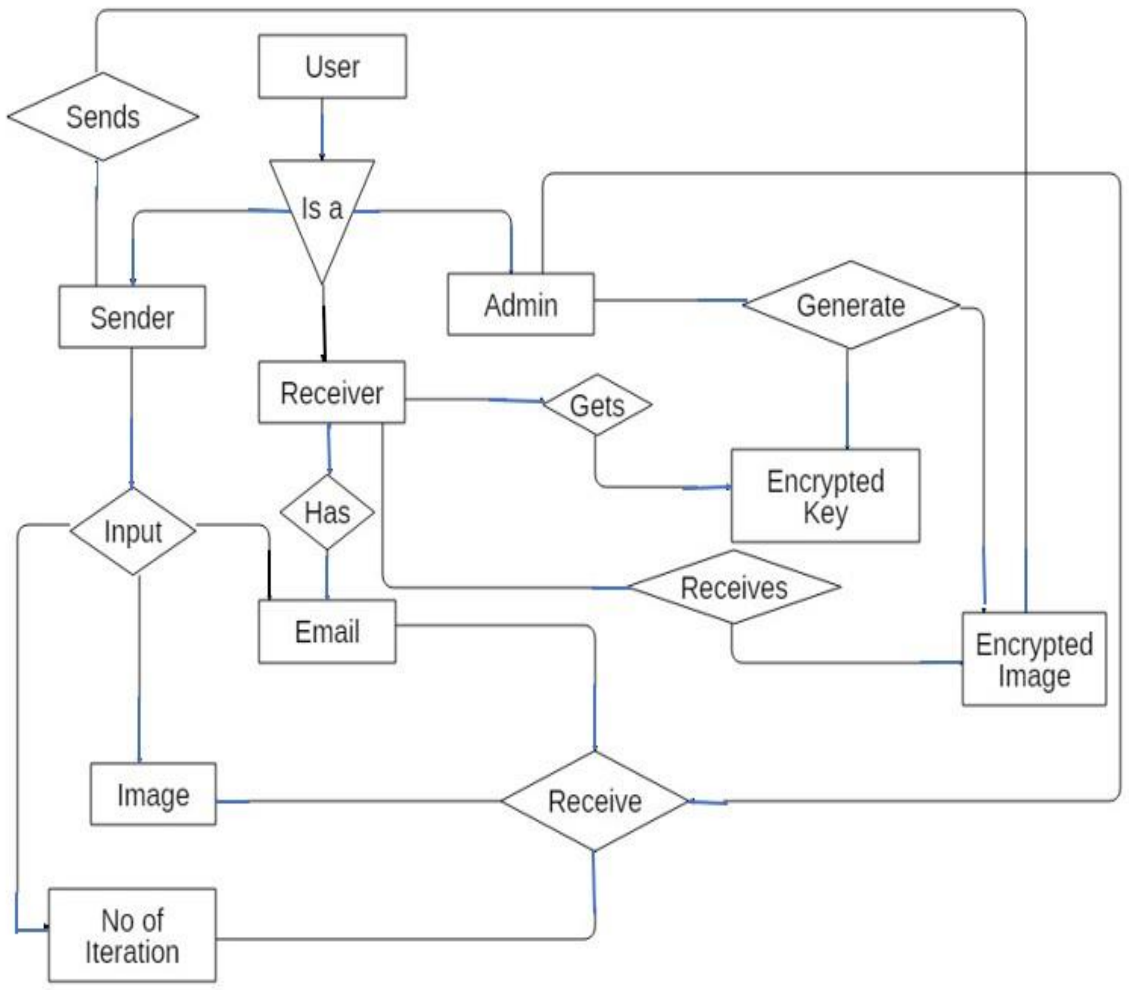


Fig5.4: Entity Relationship Diagram

5.5 SYSTEM DIAGRAM

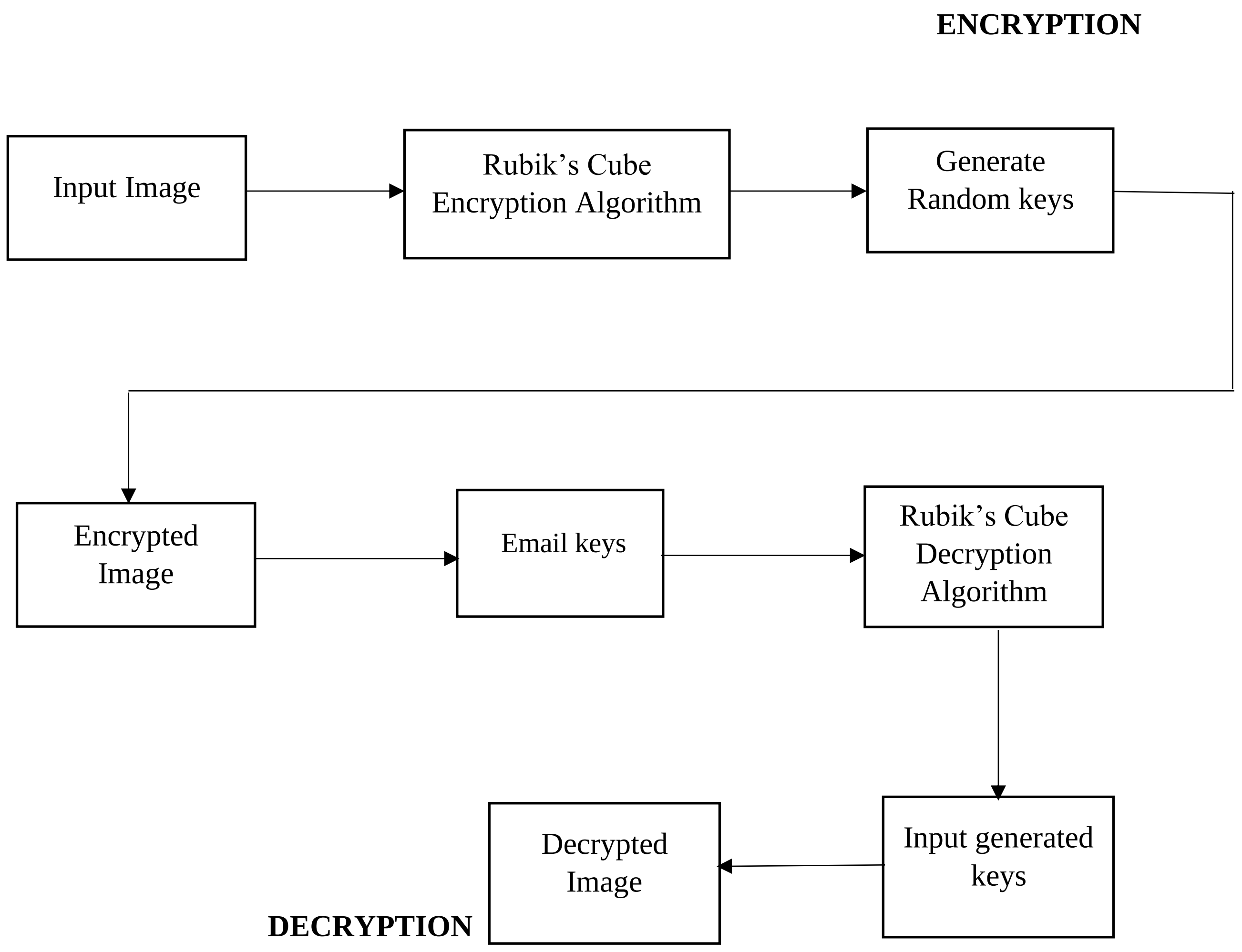


Fig 5.5: System Diagram

6. METHODOLOGY

6.1 SOFTWARE DEVELOPMENT APPROACH

6.1.1 INCREMENTAL MODEL

The incremental model is a method of software development where the product is designed, implemented and tested incrementally until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. The product is decomposed into a number of components, each of which are designed and built separately(termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

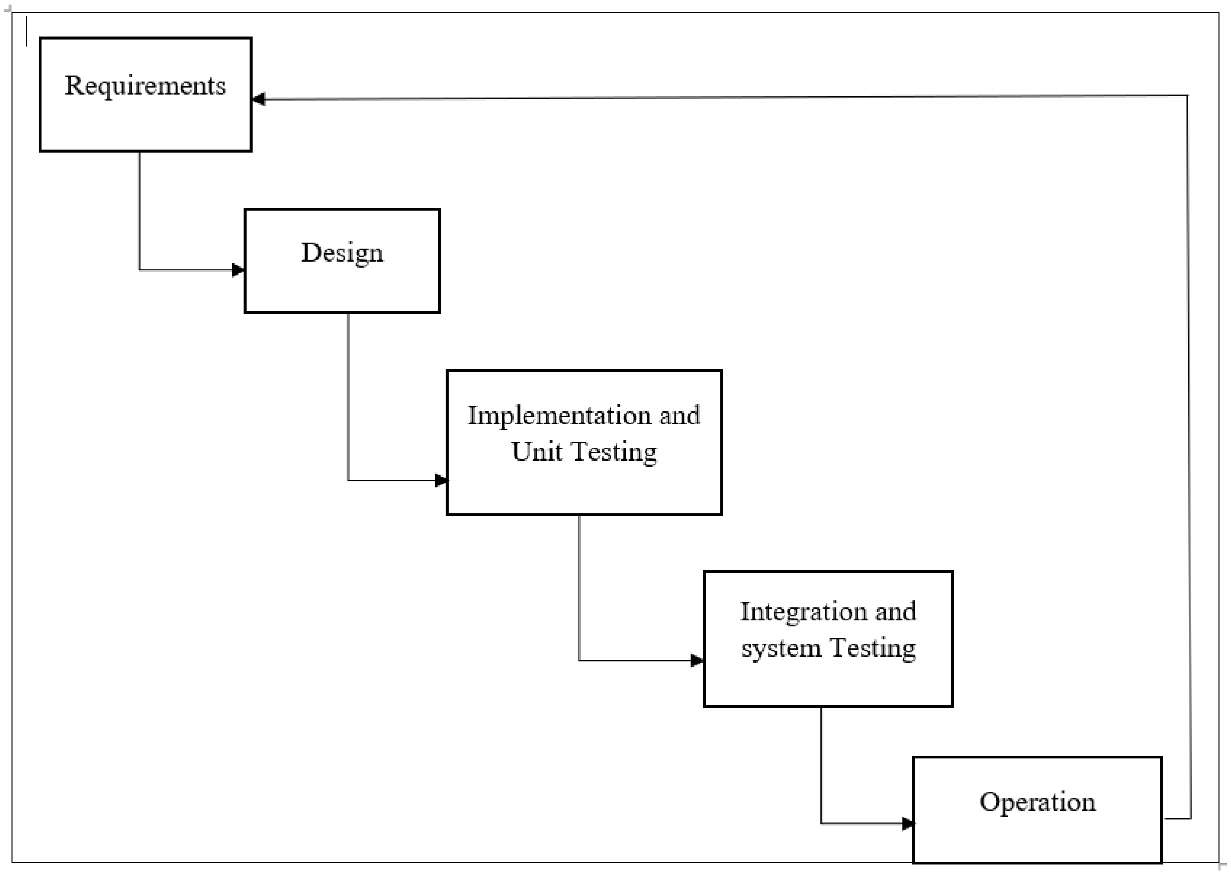

Requirements
Design
Implementation and Unit Testing
Integration and system Testing
Operation

Fig 6.1.1: Incremental Life Model

6.2 ALGORITHMIC STEPS FOR RUBIK'S CUBE IMAGE ENCRYPTION

The proposed encryption algorithm based on Rubik's cube principle is described along with the decryption algorithm as follows:

6.2.1 Rubik's Cube Encryption Algorithm

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 $\mathcal{A}=\{0,1,2,\ldots,2^{\alpha}-1\}$. 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 $\alpha(i)$

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

- (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 $\beta(j)$,

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

- (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:

$$\text{if } M_{\beta(j)} = 0 \longrightarrow \text{up circular shift}$$
$$\text{else} \longrightarrow \text{down circular shift.} \quad (4)$$

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:

$$\mathrm{I}_1(2i-1, j) = \mathrm{I}_{\mathrm{SCR}}(2i-1, j) \oplus K_C(j),$$
$$\mathrm{I}_1(2i, j) = \mathrm{I}_{\mathrm{SCR}}(2i, j) \oplus \text{rot } 180(\,\mathrm{k}_C(j)\,) \quad (5)$$

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:

$$\mathrm{I}_{\mathrm{ENC}}(i, 2j-1) = \mathrm{I}_1(i, 2j-1) \oplus K_R(j),$$
$$\mathrm{I}_{\mathrm{ENC}}(i, 2j) = \mathrm{I}_1(i, 2j) \oplus \text{rot } 180\,(\mathrm{k}_R(j)). \quad (6)$$

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

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

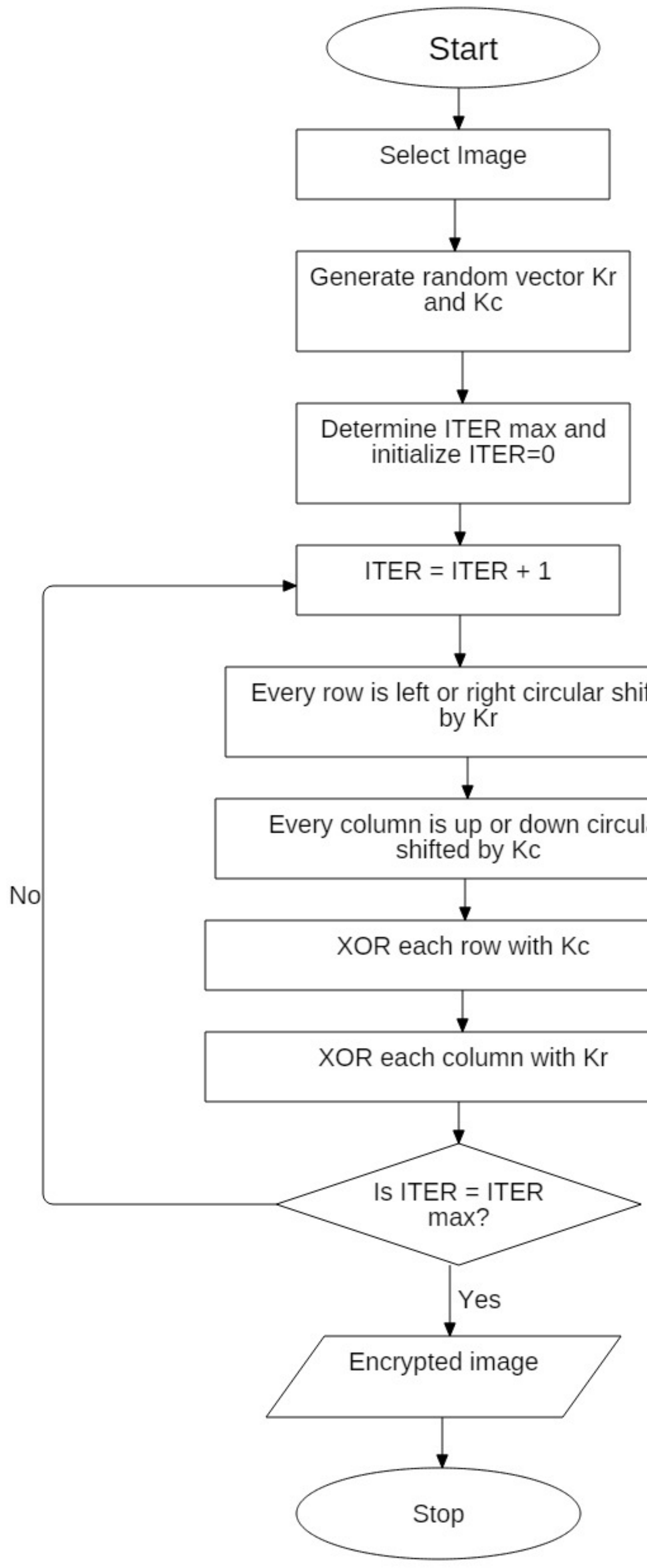

Start
Select Image
Generate random vector Kr and Kc
Determine ITER max and initialize ITER=0
ITER = ITER + 1
Every row is left or right circular shifted by Kr
Every column is up or down circular shifted by Kc
XOR each row with Kc
XOR each column with Kr
Is ITER = ITER max?
No
Yes
Encrypted image
Stop

Fig 6.2.1: Flowchart of Rubik's Cube Encryption Algorithm

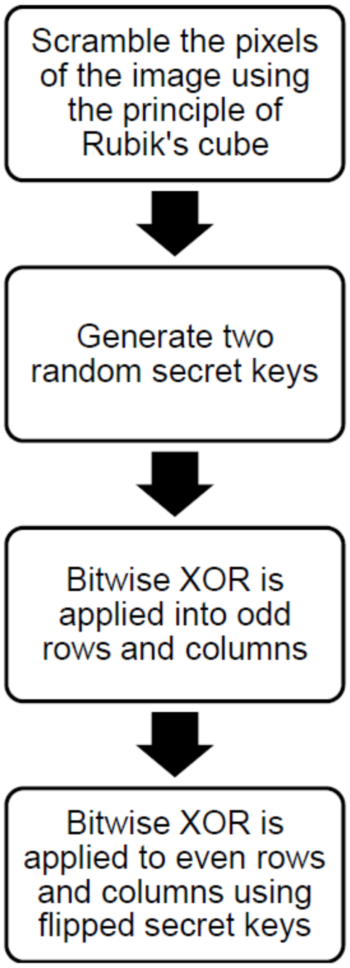

Scramble the pixels of the image using the principle of Rubik's cube
Generate two random secret keys
Bitwise XOR is applied into odd rows and columns
Bitwise XOR is applied to even rows and columns using flipped secret keys

Fig 6.2.2: Flow Diagram of Rubik’s Cube Encryption

6.2.2 Rubik's Cube Decryption Algorithm

The decrypted image, I_o, is recovered from the encrypted image, I_{ENC}, and the secret keys, K_R, K_C, and 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:

$$\text{I}_1(i,2j-1) = \text{I}_{\text{ENC}}(i, 2j-1) \oplus K_R(j),$$
$$\text{I}_1(i,2j) = \text{I}_{\text{ENC}}(i,2j) \oplus \text{rot } 180\ (\text{K}_\text{R}(j)), \quad (7)$$

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

$$\text{I}_{\text{SCR}}(2i-1, j) = \text{I}_1(2i-1, j) \oplus K_C(j),$$
$$\text{I}_{\text{SCR}}(2i, j) = \text{I}_1(2i, j) \oplus \text{rot } 180\ (\text{K}_C(j)). \quad (8)$$

(5) For each column j of the scrambled image I_{SCR},

(a) compute the sum of all elements in that column j, denoted as $\beta_{\text{SCR}}(j)$:

$$\beta_{\text{SCR}}(j)=\sum_{\text{i}=1}^{\text{M}} \text{Iscr}(\text{i}, \text{j}), \qquad j=1,2,\ldots,N, \quad (9)$$

(b) compute modulo 2 of $\beta_{\text{SCR}}(j)$, denoted by $M_{\beta\text{SCR}(j)}$,

(c) column j is down, or up, circular-shifted by $K_C(i)$ positions according to the following:

$$\text{if } M_{\beta\text{SCR}(j)}= 0 \longrightarrow \text{up circular shift}$$
$$\text{else} \longrightarrow \text{down circular shift.} \quad (10)$$

(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 $\alpha_{\text{SCR}}(i)$:

$$\alpha_{\text{SCR}}(i) = \sum_{j=1}^{N} I\,scr(i,j), \qquad i=1,2,\ldots,M, \quad (11)$$

(b) compute modulo 2 of $\alpha_{\text{SCR}}(j)$, denoted by $M_{\alpha\text{SCR}(j)}$,

(c) row i is then left, or right, circular-shifted by $K_R(i)$ according to the following:

$$\text{if } M_{\alpha\text{SCR}(j)} = 0 \longrightarrow \text{right circular shift} \quad (12)$$
$$\text{else} \longrightarrow \text{left circular shift.}$$

(7) If ITER = $\mathrm{ITER}_{\mathrm{max}}$, then image I_{ENC} is decrypted and the decryption process is done; otherwise, the algorithm branches back to step 2.

6.2.3 ADVANTAGES OF RUBIK'S CUBE ALGORITHM

i. No data loss during encryption or decryption of image.
ii. No pixel expansion is made. So, the encrypted image size is same as the original image size.
iii. The resultant image can be encrypted any number of times and decrypted back by the same number of times to get back the original image.
iv. This algorithm has high key sensitivity, i.e. image is encrypted again if wrong key is entered.

6.2.4 APPLICATION OF IMAGE ENCRYPTION

i. Governmental as well as non-governmental organizations.
ii. Places where data needs to be secure from external threats.
iii. Devices where sensitive image is stored.
iv. Hacker targeted areas.

7. RESULT AND ANALYSIS

We have design and implement a system which provide dedicated to highly secure data storage for different governmental and non- governmental sector. To implement this system, we have used encryption algorithm based on Rubik's cube principle that encrypted the image into unexpected rough pixel format and generate the key for future decryption and send the generated key into respected email id of an individual's / organization. After the completion of this project, the obtained output is shown as shown in figure below:

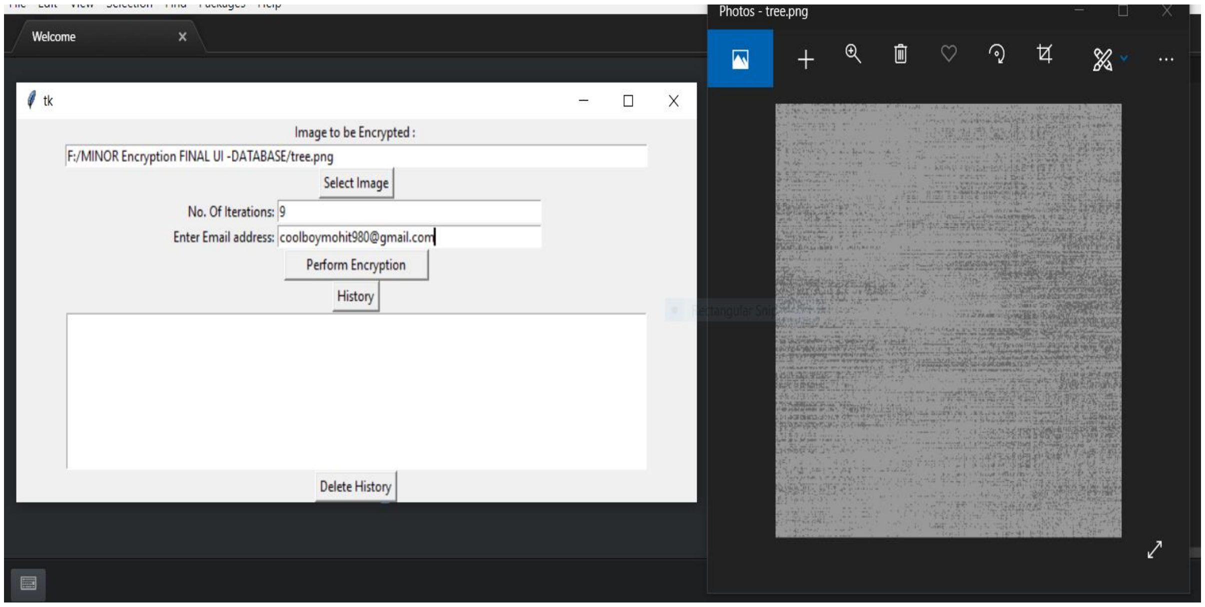


Image to be Encrypted :
F:/MINOR Encryption FINAL UI -DATABASE/tree.png
No. Of Iterations: 9
Enter Email address: coolboymohit980@gmail.com

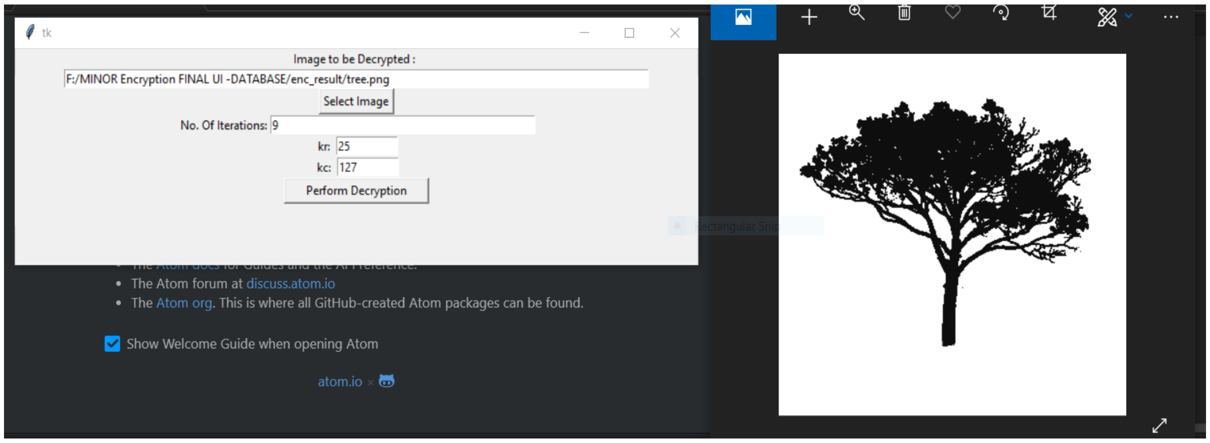


Image to be Decrypted :
F:/MINOR Encryption FINAL UI -DATABASE/enc_result/tree.png
No. Of Iterations: 9
kr: 25
kc: 127

The Atom forum at discuss.atom.io
The Atom org. This is where all GitHub-created Atom packages can be found.

Fig 7: Sample figure of Resultant output

8. CONCLUSION

A secure image encryption and decryption 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. Experimental tests have been carried out with detailed numerical analysis which demonstrates the robustness of the proposed algorithm against several types of attacks such as statistical and differential attacks (visual testing). Moreover, performance assessment tests demonstrate that the proposed image encryption algorithm is highly secure.

9. LIMITATIONS AND FUTURE WORK

Limitations of our system are:

1. Our system requires email to be sent to the receiver. People in the region where there is no internet connection will not be able to use our application.

2. Our system can only encrypt square grey-scale images.

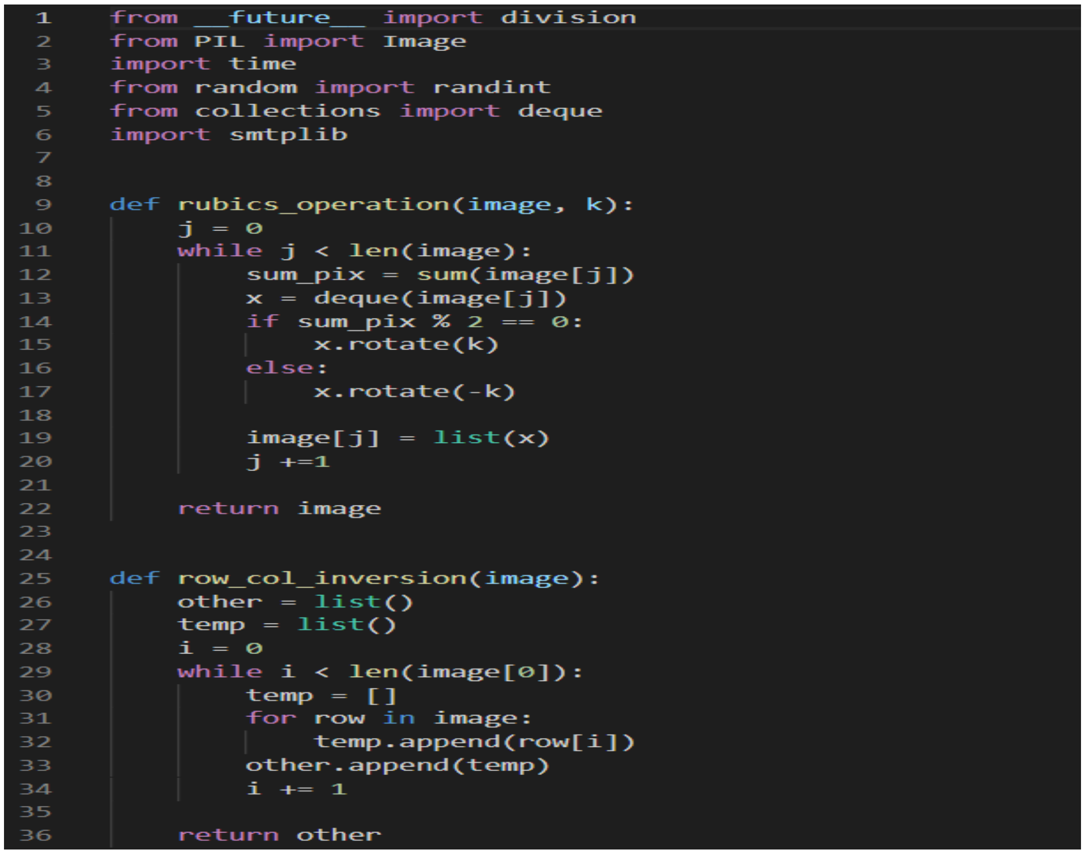
Future enhancements to be made in our system are:

1. This can be extended to an n-digit code and even for a string code.
2. As public key cryptography is more secured than secret key cryptography, this should be extended to public key cryptography.
3. Testing should be done on various equations for determining the constraints to increase the computational speed.
4. It could be implemented on android platform too.

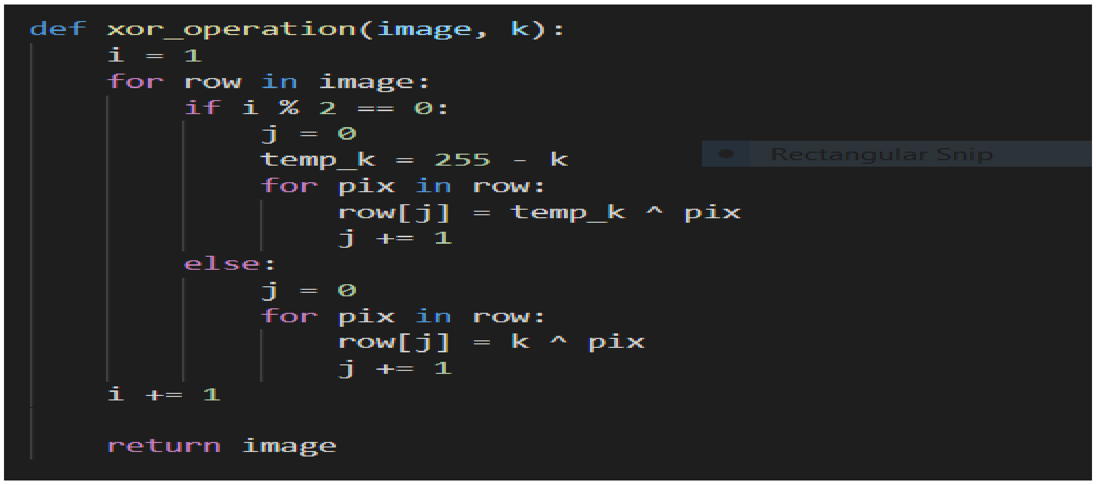
10. REFERENCES

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 848392, 10 pages
(http://dx.doi.org/10.1155/2013/848392)

11. APPENDIX

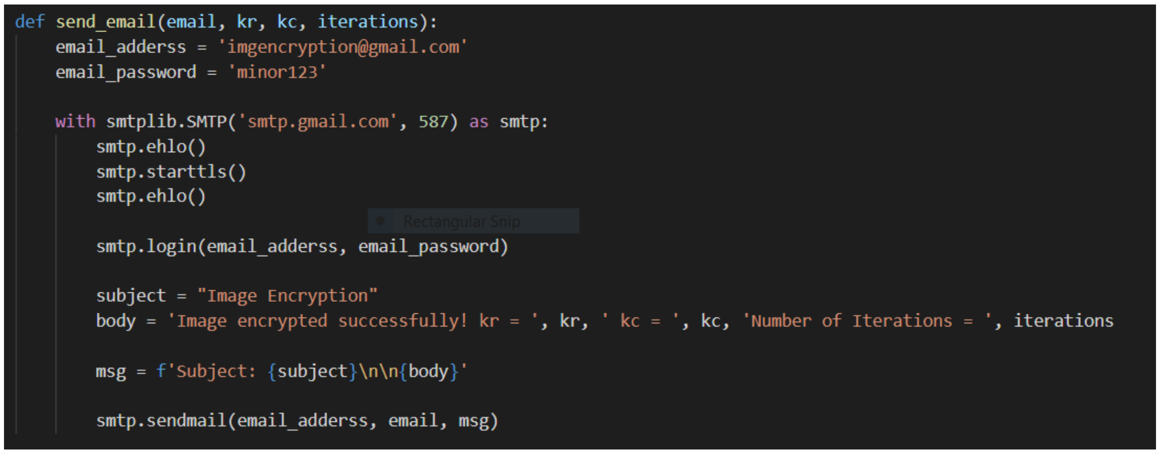


```python
from __future__ import division
from PIL import Image
import time
from random import randint
from collections import deque
import smtplib


def rubics_operation(image, k):
    j = 0
    while j < len(image):
        sum_pix = sum(image[j])
        x = deque(image[j])
        if sum_pix % 2 == 0:
            x.rotate(k)
        else:
            x.rotate(-k)

        image[j] = list(x)
        j +=1

    return image


def row_col_inversion(image):
    other = list()
    temp = list()
    i = 0
    while i < len(image[0]):
        temp = []
        for row in image:
            temp.append(row[i])
        other.append(temp)
        i += 1

    return other
```



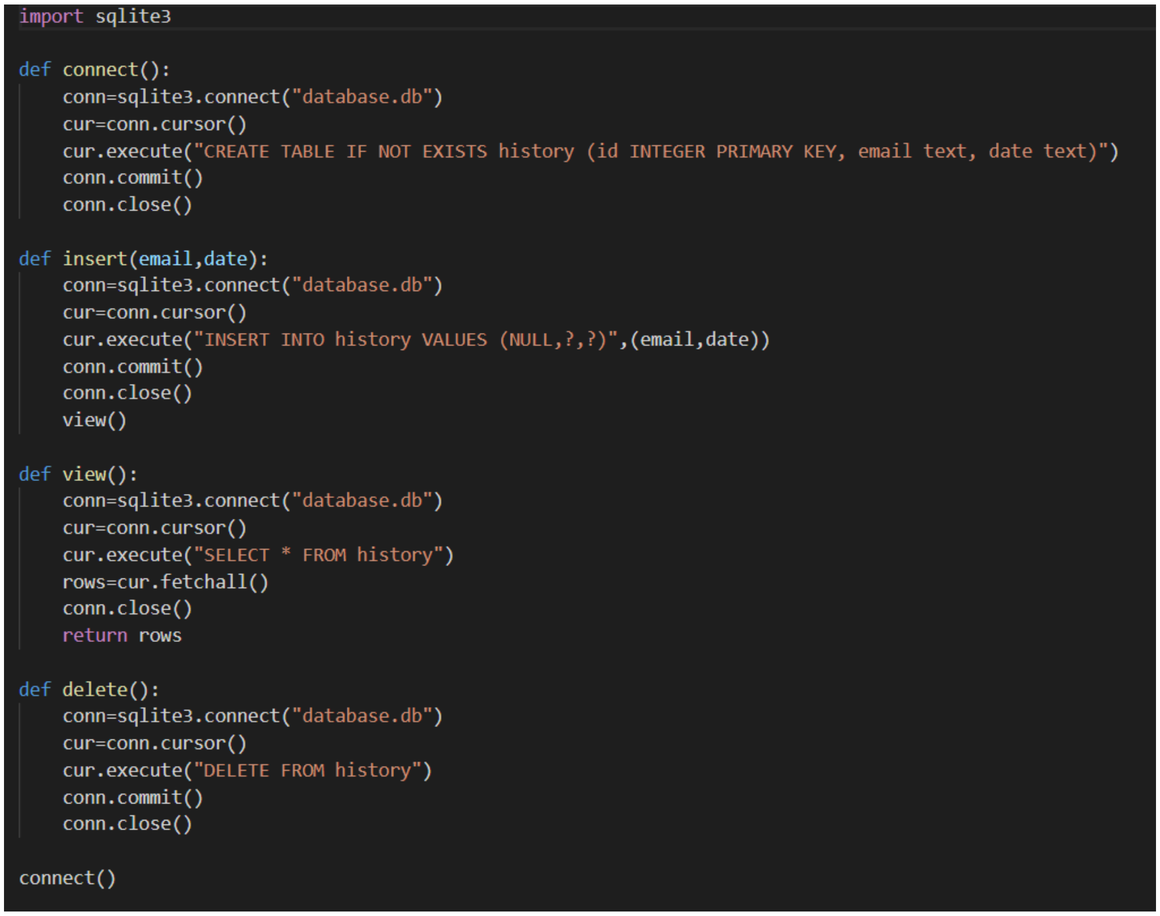
```python
def xor_operation(image, k):
    i = 1
    for row in image:
        if i % 2 == 0:
            j = 0
            temp_k = 255 - k
            for pix in row:
                row[j] = temp_k ^ pix
                j += 1
        else:
            j = 0
            for pix in row:
                row[j] = k ^ pix
                j += 1
    i += 1

    return image
```

Fig 11.1: Code sample for encryption part



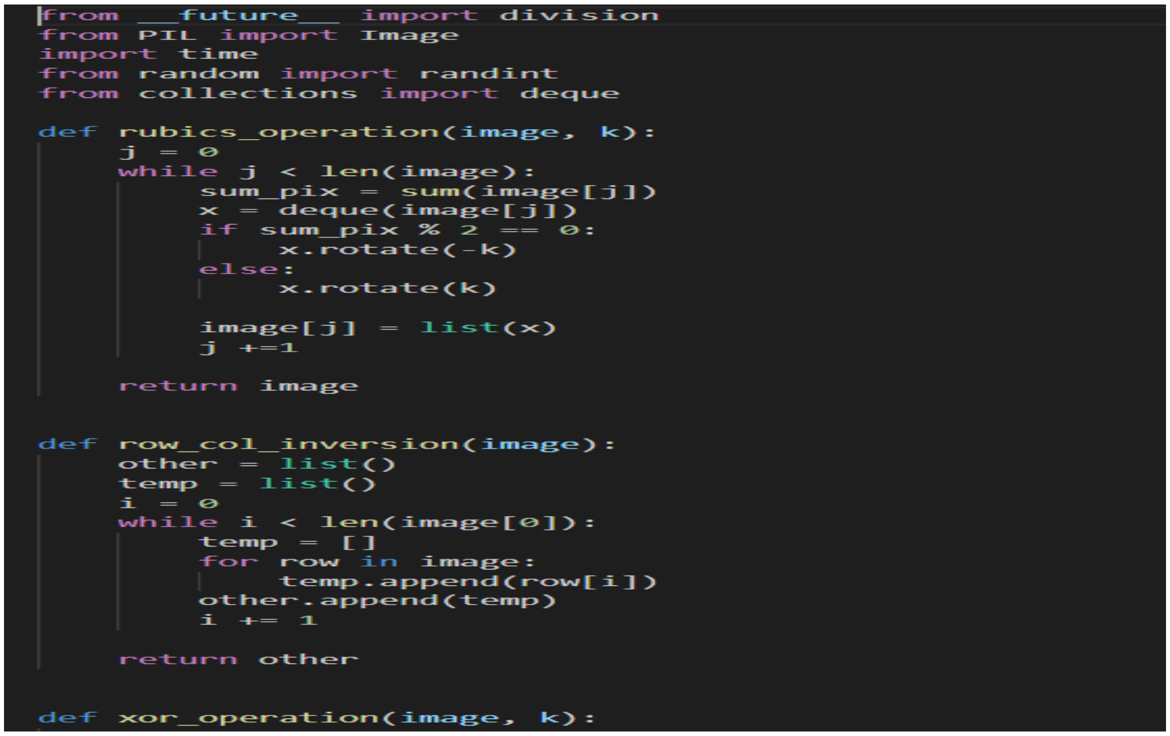
```python
def send_email(email, kr, kc, iterations):
    email_adderss = 'imgencryption@gmail.com'
    email_password = 'minor123'

    with smtplib.SMTP('smtp.gmail.com', 587) as smtp:
        smtp.ehlo()
        smtp.starttls()
        smtp.ehlo()

        smtp.login(email_adderss, email_password)

        subject = "Image Encryption"
        body = 'Image encrypted successfully! kr = ', kr, ' kc = ', kc, 'Number of Iterations = ', iterations

        msg = f'Subject: {subject}\n\n{body}'

        smtp.sendmail(email_adderss, email, msg)
```

Fig 11.2: Code sample for Sending Email

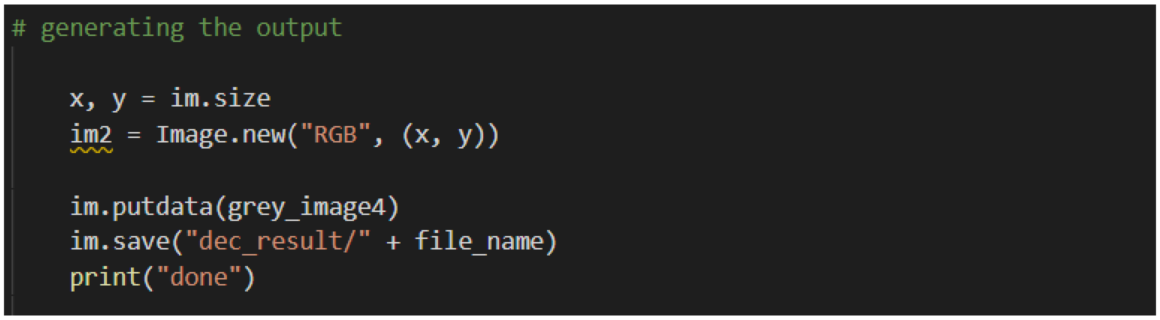


```python
import sqlite3

def connect():
    conn=sqlite3.connect("database.db")
    cur=conn.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS history (id INTEGER PRIMARY KEY, email text, date text)")
    conn.commit()
    conn.close()

def insert(email,date):
    conn=sqlite3.connect("database.db")
    cur=conn.cursor()
    cur.execute("INSERT INTO history VALUES (NULL,?,?)",(email,date))
    conn.commit()
    conn.close()
    view()

def view():
    conn=sqlite3.connect("database.db")
    cur=conn.cursor()
    cur.execute("SELECT * FROM history")
    rows=cur.fetchall()
    conn.close()
    return rows

def delete():
    conn=sqlite3.connect("database.db")
    cur=conn.cursor()
    cur.execute("DELETE FROM history")
    conn.commit()
    conn.close()

connect()
```

Fig 11.3: Code sample for database



```
from __future__ import division
from PIL import Image
import time
from random import randint
from collections import deque

def rubics_operation(image, k):
    j = 0
    while j < len(image):
        sum_pix = sum(image[j])
        x = deque(image[j])
        if sum_pix % 2 == 0:
            x.rotate(-k)
        else:
            x.rotate(k)

        image[j] = list(x)
        j +=1

    return image


def row_col_inversion(image):
    other = list()
    temp = list()
    i = 0
    while i < len(image[0]):
        temp = []
        for row in image:
            temp.append(row[i])
        other.append(temp)
        i += 1

    return other


def xor_operation(image, k):
```



```
# generating the output

    x, y = im.size
    im2 = Image.new("RGB", (x, y))

    im.putdata(grey_image4)
    im.save("dec_result/" + file_name)
    print("done")
```

Fig 11.4: Code sample for decryption