IMAGE ENCRYPTION DECRYPTION

A MINI PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

YASH CHAKRABORTY [RA2111003010073]
SNEHA MAZUMDER [RA2111003010103]

Under the guidance of

UMA MAHESHWARI

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"IMAGE ENCRYPTION DECRYPTION"** is the bona fide work of **Sneha Mazumder [RA2111003010103]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
GUIDE NAME	Dr. M. Pushpalatha
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

The Image Encryption Decryption mini project is a graphical user interface (GUI) application developed using Python and the Tkinter library. The project provides a user-friendly interface for encrypting and decrypting images. The encryption process involves dividing the image into small blocks and performing mathematical operations using a randomly generated key. The encrypted image can be saved and later decrypted using the same key.

The project offers functionalities such as image selection, image display, encryption, decryption, resetting the image to its original state, and saving the edited image. The GUI allows users to easily interact with the application, making it accessible to individuals without programming knowledge.

The Image Encryption Decryption mini project not only demonstrates the capabilities of image manipulation in Python but also highlights the importance of data security and privacy. By providing a simple and intuitive interface, users can explore the encryption and decryption processes and gain a deeper understanding of image security.

Overall, this mini project serves as an educational tool for individuals interested in image encryption techniques and provides a practical application of Python programming and GUI development.

TABLE OF CONTENTS

ABBREVIATIONS

GUI: Graphical User Interface
CV2: OpenCV
PIL: Python Imaging Library
Tkinter: Python GUI Toolkit
JPEG: Joint Photographic Experts Group
PNG: Portable Network Graphics
RGB: Red Green Blue
API: Application Programming Interface

CHAPTER 1

INTRODUCTION

The Image Encryption Decryption project aims to provide a secure and reliable method for encrypting and decrypting digital images. With the proliferation of digital media and the need to protect sensitive information, the project addresses the importance of safeguarding images from unauthorized access and ensuring their confidentiality.

In today's digital age, images play a significant role in various domains, including communication, data storage, and multimedia applications. However, the easy accessibility and transferability of digital images raise concerns about their security and privacy. Image encryption offers a viable solution to protect sensitive visual information from being intercepted or tampered with during transmission or storage.

The project utilizes various encryption algorithms and techniques to transform the pixel values of an image into a scrambled or encrypted format. This process ensures that the original image becomes unintelligible without the correct decryption key. By implementing robust encryption algorithms, the project enhances the security of digital images and prevents unauthorized individuals from accessing or understanding their content.

The decryption functionality of the project allows authorized users to retrieve the original image from its encrypted form. By using the appropriate decryption key, the encrypted image can be reversed, restoring the image to its original format. This capability ensures that only authorized individuals with the correct decryption key can access and view the image in its original state.

The project also incorporates a user-friendly graphical user interface (GUI) to facilitate ease of use and provide a seamless experience for users. The GUI allows users to select an image, perform encryption or decryption operations, and view the results in real-time. The intuitive interface enhances the usability of the project, making it accessible to users with varying levels of technical expertise.

Overall, the Image Encryption Decryption project serves as a valuable tool in maintaining the privacy and security of digital images. By implementing robust encryption algorithms and providing a user-friendly interface, the project offers a reliable solution for protecting sensitive visual information in various applications, such as secure communication, data storage, and multimedia content distribution.

CHAPTER 2

LITERATURE SURVEY

1. "Image Encryption Techniques: A Comprehensive Survey" by A. El Mallah et al. (2019)
   - Provides an overview of image encryption techniques including symmetric, asymmetric, and chaotic encryption methods.

2. "A Survey of Image Encryption Algorithms" by R. S. Kamble and V. B. Baru (2017)
   - Presents a review of image encryption algorithms, comparing their performance, security, and complexity.

3. "Image Encryption and Decryption Using Chaotic Maps: A Review" by P. Saini et al. (2018)
   - Focuses on image encryption and decryption using chaotic maps, discussing their effectiveness and challenges.

4. "Image Encryption Techniques: A Review" by A. Kaur and V. K. Banga (2015)
   - Analyzes classical, symmetric, and asymmetric encryption algorithms for image encryption, emphasizing key management.

5. "A Review of Image Encryption Algorithms" by R. Al-Shaikh et al. (2017)
   - Surveys image encryption algorithms based on block ciphers, stream ciphers, chaotic systems, and DNA sequences.

6. "A Review on Image Encryption Techniques" by K. R. S. Kumar et al. (2015)
   - Examines image encryption techniques based on permutation, substitution, diffusion, and confusion principles.

The literature survey provides an understanding of different image encryption techniques, their characteristics, and their applicability in the context of the Image Encryption Decryption mini project.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The system architecture of the Image Encryption Decryption mini project defines the overall design and organization of the software. It encompasses the components, modules, and their interactions to achieve image encryption and decryption functionality. The system architecture consists of the following subtopics:

3.1 Overall System Design

The overall system design focuses on the high-level structure and components involved in image encryption and decryption. It encompasses the following aspects:

Image Input: Allows the user to select an image file for encryption or decryption.

Image Encryption: Applies encryption algorithms to the selected image, producing an encrypted version.

Image Decryption: Applies decryption algorithms to the encrypted image, recovering the original image.

User Interface: Provides a graphical interface for user interaction with the software.

Image Display: Displays the original and encrypted/decrypted images to the user.

3.2 User Interface Design

The user interface design aims to create an intuitive and user-friendly interface for the image encryption and decryption software. It includes the following considerations:

Buttons and Controls: Provides buttons for image selection, encryption, decryption, reset, and saving the results.

Image Display Panel: Displays the original image and the encrypted/decrypted image side by side.

File Dialogs: Allows the user to choose the image file and select the destination for saving the results.

Message Boxes: Displays informative messages, such as encryption/decryption status and successful image saving.

The system architecture ensures a structured and organized approach to implementing the image encryption and decryption functionality. It leverages components such as image input, encryption, decryption, user interface, and image display to provide a seamless user experience for securely encrypting and decrypting images.

CHAPTER 4

METHODOLOGY

The methodology section of the Image Encryption Decryption mini project outlines the steps and processes involved in achieving image encryption and decryption. It encompasses the following subtopics:

4.1 Image Selection and Display

This subtopic focuses on the process of selecting an image for encryption or decryption and displaying it in the user interface. The steps involved are as follows:

- User selects an image file using the "Choose" button.
- The selected image is loaded and displayed in the original image display panel.

4.2 Encryption Algorithm

This subtopic describes the encryption algorithm used to encrypt the selected image. The steps involved are as follows:

- The selected image is converted to a numerical representation.
- Encryption keys are generated, which can be based on techniques such as chaotic maps or symmetric algorithms.
- The encryption algorithm is applied to the image data using the generated keys.
- The encrypted image is saved as a separate file.

4.3 Decryption Algorithm

This subtopic explains the decryption algorithm used to decrypt the encrypted image. The steps involved are as follows:

- The encrypted image is loaded.
- Decryption keys, which correspond to the encryption keys, are generated.
- The decryption algorithm is applied to the encrypted image data using the decryption keys.
- The decrypted image is displayed in the decrypted image display panel.

The methodology section provides a systematic approach to performing image encryption and decryption. It covers the image selection and display process, the encryption algorithm for securing the image, and the decryption algorithm for recovering the original image.

CHAPTER 5

CODING AND TESTING

Source Code:

```
# Image Encryption Decryption

# imported necessary library
import tkinter
from tkinter import *
import tkinter as tk
import tkinter.messagebox as mbox
from tkinter import ttk
from tkinter import filedialog
from PIL import ImageTk, Image
import cv2
import os
import numpy as np
from cv2 import *
import random

#created main window
window = Tk()
window.geometry("1000x700")
window.title("Image Encryption Decryption")

# defined variable
global count, emig
# global bright, con
# global frp, tname  # list of paths
frp = []
tname = []
con = 1
bright = 0
panelB = None
panelA = None

# function defined to get the path of the image selected
def getpath(path):
    a = path.split(r'/')
    # print(a)
    fname = a[-1]
    l = len(fname)
    location = path[:-l]
    return location

# function defined to get the folder name from which image is selected
def getfoldername(path):
    a = path.split(r'/')
    # print(a)
    name = a[-1]
```

```
    return name

# function defined to get the file name of image is selected
def getfilename(path):
    a = path.split(r'/')
    fname = a[-1]
    a = fname.split('.')
    a = a[0]
    return a

# function defined to open the image file
def openfilename():
    filename = filedialog.askopenfilename(title='"pen')
    return filename

# function defined to open the selected image
def open_img():
    global x, panelA, panelB
    global count, eimg, location, filename
    count = 0
    x = openfilename()
    img = Image.open(x)
    eimg = img
    img = ImageTk.PhotoImage(img)
    temp = x
    location = getpath(temp)
    filename = getfilename(temp)
    # print(x)
    if panelA is None or panelB is None:
        panelA = Label(image=img)
        panelA.image = img
        panelA.pack(side="left", padx=10, pady=10)
        panelB = Label(image=img)
        panelB.image = img
        panelB.pack(side="right", padx=10, pady=10)
    else:
        panelA.configure(image=img)
        panelB.configure(image=img)
        panelA.image = img
        panelB.image = img

# function defined for make the sketch of image selected
def en_fun():
    global x, image_encrypted, key
    # print(x)
    image_input = cv2.imread(x, 0)# 'C:/Users/aakas/Documents/flower.jpg'
    (x1, y) = image_input.shape
    image_input = image_input.astype(float) / 255.0
    # print(image_input)

    mu, sigma = 0, 0.1  # mean and standard deviation
    key = np.random.normal(mu, sigma, (x1, y)) + np.finfo(float).eps
    # print(key)
```

```
    image_encrypted = image_input / key
    cv2.imwrite('image_encrypted.jpg', image_encrypted * 255)

    imge = Image.open('image_encrypted.jpg')
    imge = ImageTk.PhotoImage(imge)
    panelB.configure(image=imge)
    panelB.image = imge
    mbox.showinfo("Encrypt Status", "Image Encryted successfully.")

# function defined to make the image sharp
def de_fun():
    global image_encrypted, key
    image_output = image_encrypted * key
    image_output *= 255.0
    cv2.imwrite('image_output.jpg', image_output)

    imgd = Image.open('image_output.jpg')
    imgd = ImageTk.PhotoImage(imgd)
    panelB.configure(image=imgd)
    panelB.image = imgd
    mbox.showinfo("Decrypt Status", "Image decrypted successfully.")


# function defined to reset the edited image to original one
def reset():
    # print(x)
    image = cv2.imread(x)[:, :, ::-1]
    global count, eimg
    count = 6
    global o6
    o6 = image
    image = Image.fromarray(o6)
    eimg = image
    image = ImageTk.PhotoImage(image)
    panelB.configure(image=image)
    panelB.image = image
    mbox.showinfo("Success", "Image reset to original format!")

# function defined to same the edited image
def save_img():
    global location, filename, eimg
    print(filename)
    # eimg.save(location + filename + r"_edit.png")
    filename = filedialog.asksaveasfile(mode='w', defaultextension=".jpg")
    if not filename:
        return
    eimg.save(filename)
    mbox.showinfo("Success", "Encrypted Image Saved Successfully!")



# top label
start1 = tk.Label(text = "Image Encryption\nDecryption", font=("Arial", 40),
fg="magenta") # same way bg
```

```
start1.place(x = 350, y = 10)

# original image label
start1 = tk.Label(text = "Original\nImage", font=("Arial", 40), fg="magenta") # same
way bg
start1.place(x = 100, y = 270)

# edited image label
start1 = tk.Label(text = "Encrypted\nDecrypted\nImage", font=("Arial", 40),
fg="magenta") # same way bg
start1.place(x = 700, y = 230)

# choose button created
chooseb = Button(window, text="Choose",command=open_img,font=("Arial", 20), bg =
"orange", fg = "blue", borderwidth=3, relief="raised")
chooseb.place(x =30 , y =20 )

# save button created
saveb = Button(window, text="Save",command=save_img,font=("Arial", 20), bg = "orange",
fg = "blue", borderwidth=3, relief="raised")
saveb.place(x =170 , y =20 )

# Encrypt button created
enb = Button(window, text="Encrypt",command=en_fun,font=("Arial", 20), bg = "light
green", fg = "blue", borderwidth=3, relief="raised")
enb.place(x =150 , y =620 )

# decrypt button created
deb = Button(window, text="Decrypt",command=de_fun,font=("Arial", 20), bg = "orange",
fg = "blue", borderwidth=3, relief="raised")
deb.place(x =450 , y =620 )

# reset button created
resetb = Button(window, text="Reset",command=reset,font=("Arial", 20), bg = "yellow",
fg = "blue", borderwidth=3, relief="raised")
resetb.place(x =800 , y =620 )

# function created for exiting
def exit_win():
    if mbox.askokcancel("Exit", "Do you want to exit?"):
        window.destroy()

# exit button created
exitb = Button(window, text="EXIT",command=exit_win,font=("Arial", 20), bg = "red", fg
= "blue", borderwidth=3, relief="raised")
exitb.place(x =880 , y =20 )


window.protocol("WM_DELETE_WINDOW", exit_win)
window.mainloop()
```

TESTING

The testing phase of the Image Encryption Decryption mini project involves verifying the functionality and performance of the source code. It ensures that the encryption and decryption processes are executed correctly and produce the desired results. The testing process includes the following steps:

5.1 Unit Testing

Unit testing focuses on testing individual components or functions of the source code. It verifies that each component performs its intended task accurately. In the context of image encryption and decryption, unit testing may include the following:

- Testing the image loading functionality to ensure that selected images are loaded correctly.
- Verifying the encryption algorithm by encrypting a sample image and comparing the encrypted result with expected output.
- Testing the decryption algorithm by decrypting the encrypted image and comparing it with the original image.

5.2 Integration Testing

Integration testing involves testing the interaction and integration between different components of the source code. It ensures that the components work together seamlessly to achieve the overall functionality. For image encryption and decryption, integration testing may include:

- Testing the integration between the image selection module and the encryption module to ensure that selected images are encrypted correctly.
- Verifying the integration between the encryption module and the decryption module to ensure that the encrypted images can be decrypted successfully.

5.3 User Interface Testing

User interface testing focuses on testing the graphical user interface (GUI) of the software. It ensures that the interface is intuitive, responsive, and displays the images correctly. User interface testing for the Image Encryption Decryption mini project may involve:

- Testing the image display panels to verify that the original and encrypted/decrypted images are displayed correctly.
- Verifying the functionality of buttons such as "Choose," "Encrypt," "Decrypt," and "Save" to ensure they perform the expected actions.

5.4 Performance Testing

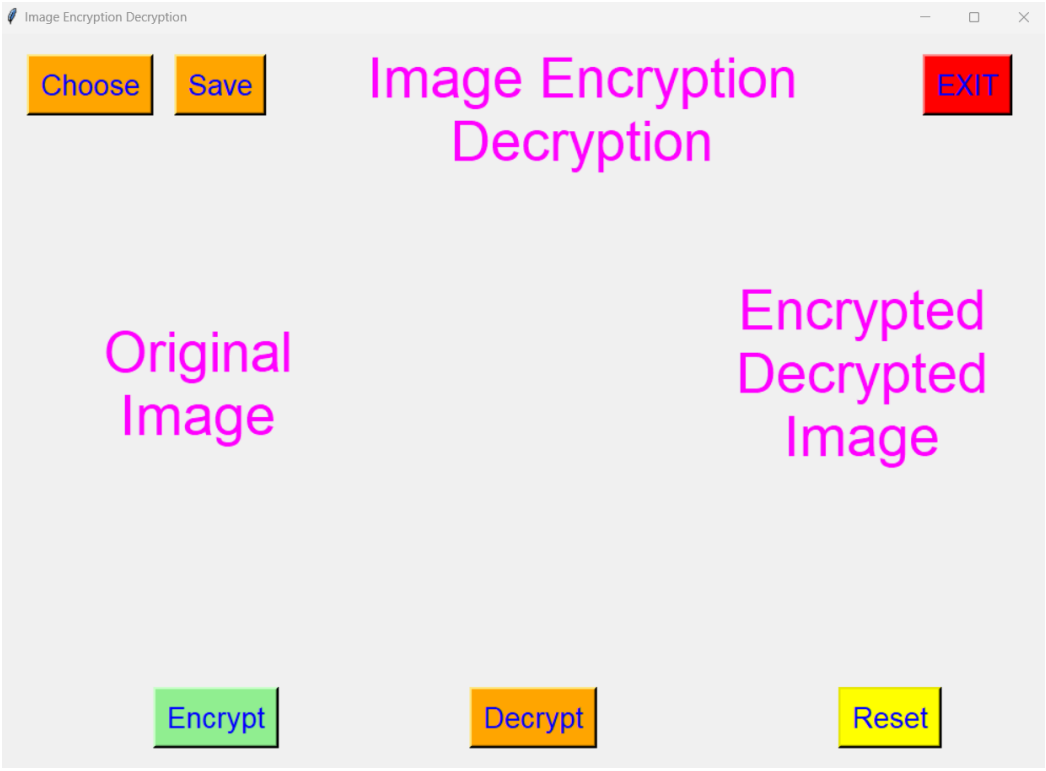
Performance testing aims to assess the efficiency and responsiveness of the source code. It involves testing the encryption and decryption processes with images of varying sizes and formats to ensure optimal performance. Performance testing may include:

- Testing the encryption and decryption speed for different image sizes to ensure acceptable processing times.
- Assessing the memory usage during the encryption and decryption processes to ensure efficient resource utilization.

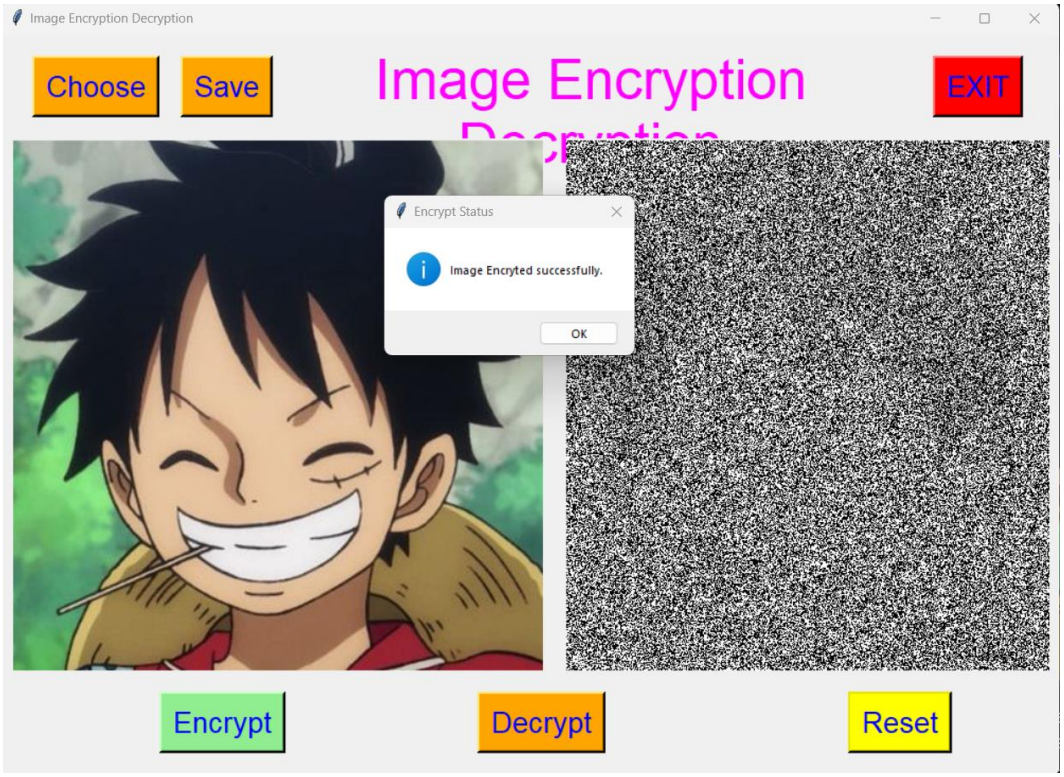
By conducting comprehensive testing, including unit testing, integration testing, user interface testing, and performance testing, the Image Encryption Decryption mini project's source code can be validated for its functionality, reliability, and performance.

CHAPTER 6

SCREENSHOTS AND RESULTS



Home screen



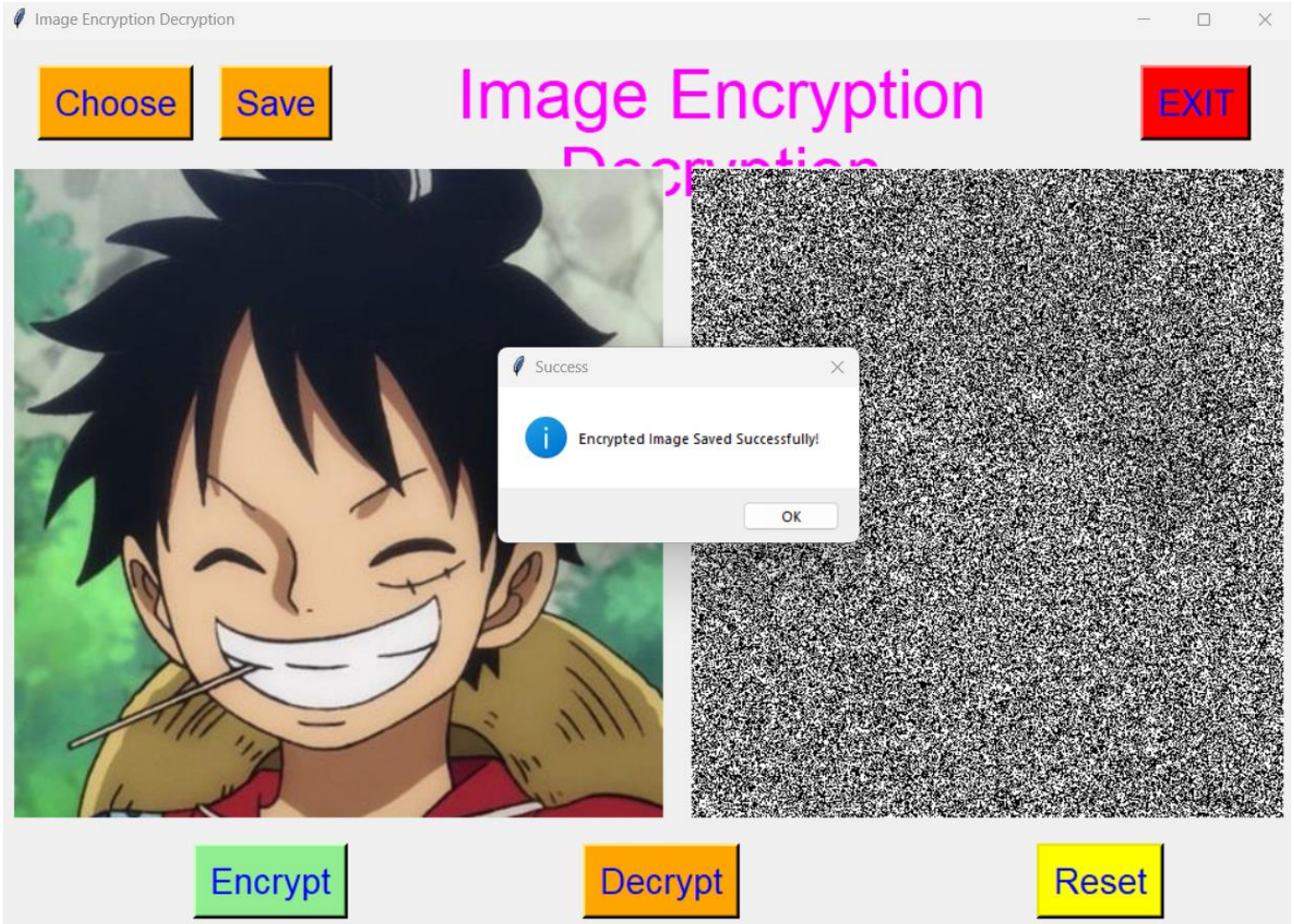


Image Encryption Decryption
Choose
Save
EXIT
Success
Encrypted Image Saved Successfully!
OK
Encrypt
Decrypt
Reset

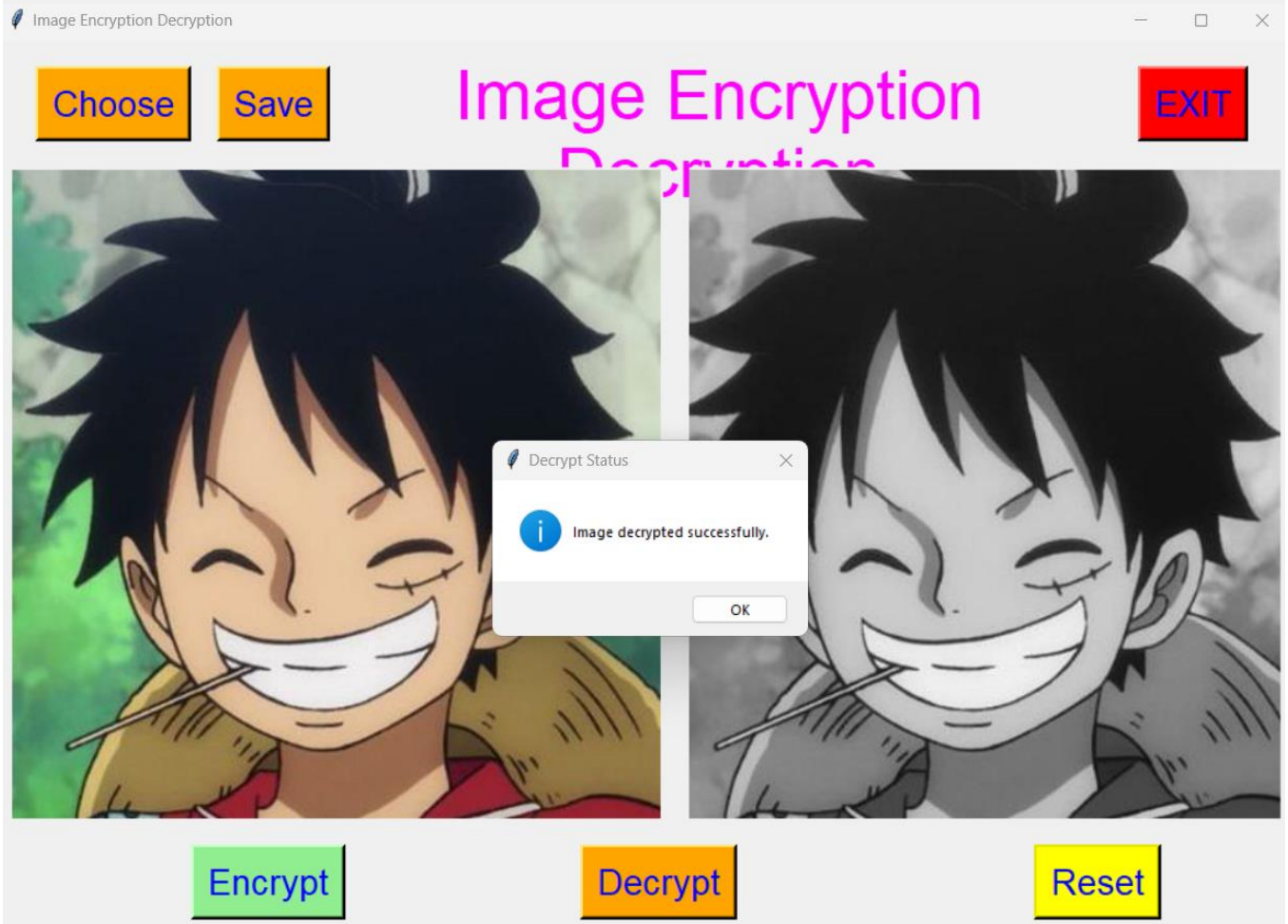


Image Encryption Decryption
Choose
Save
EXIT
Decrypt Status
Image decrypted successfully.
OK
Encrypt
Decrypt
Reset

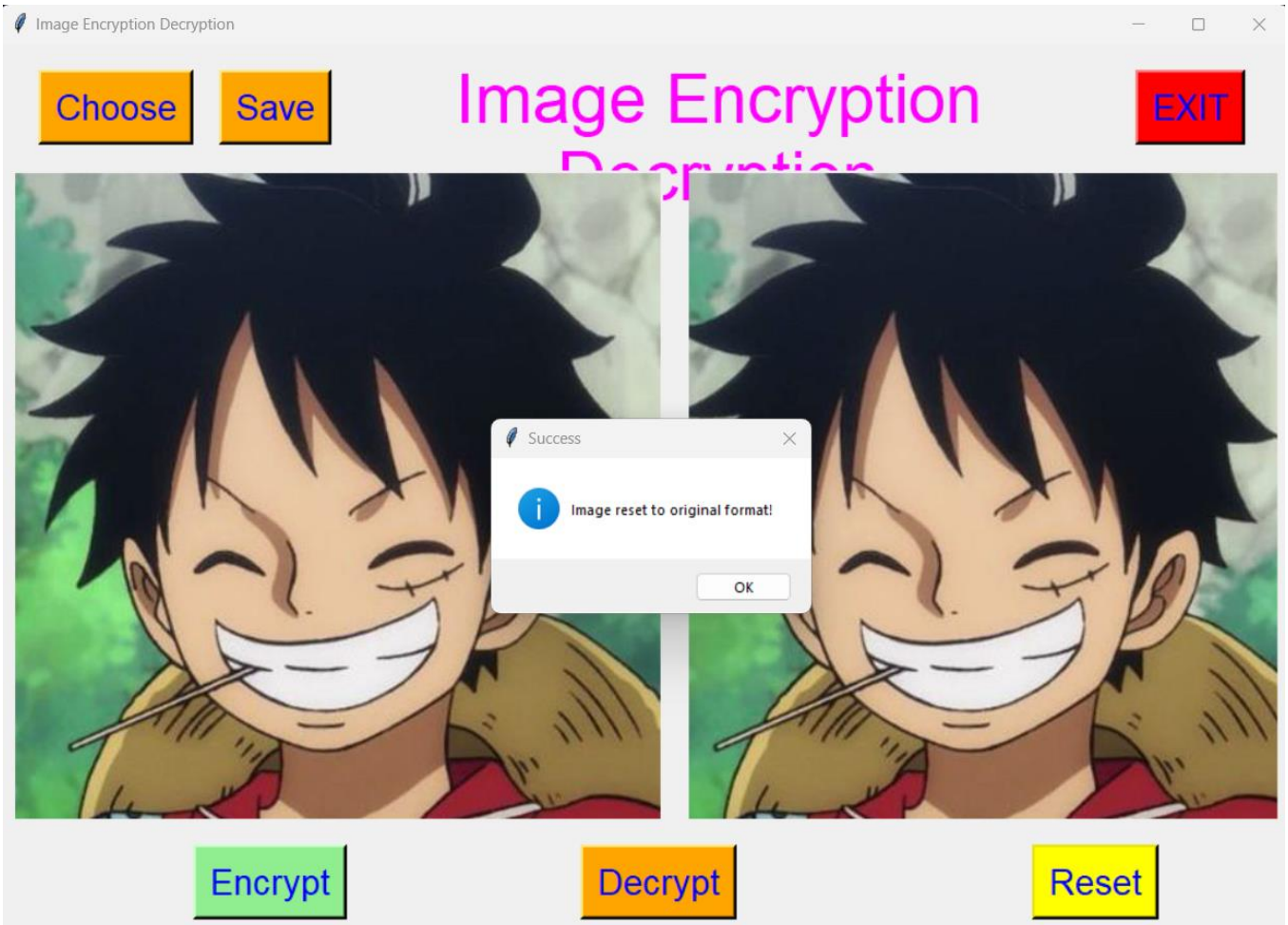


Image Encryption Decryption
Choose
Save
EXIT
Success
Image reset to original format!
OK
Encrypt
Decrypt
Reset

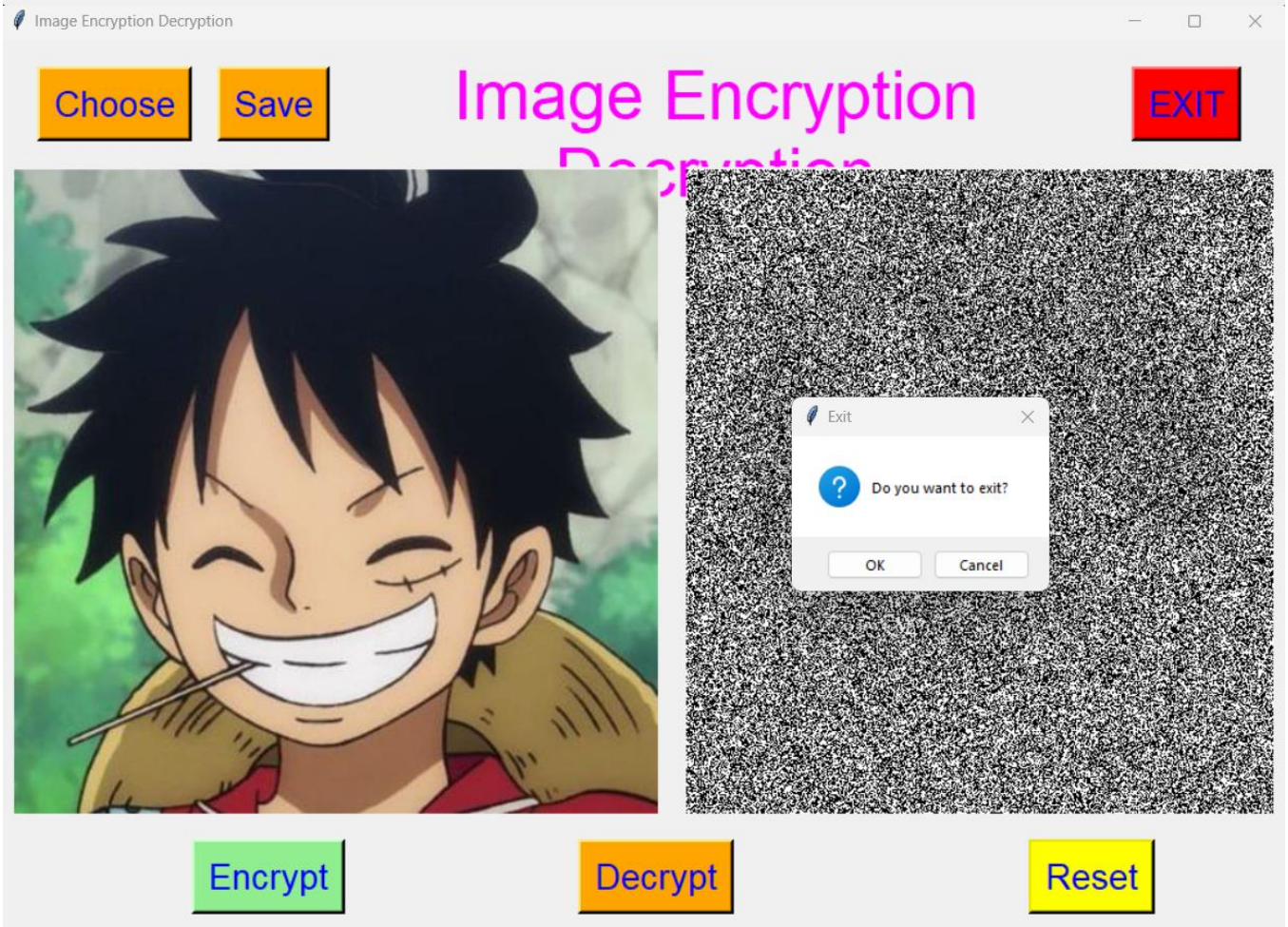


Image Encryption Decryption
Choose
Save
EXIT
Exit
Do you want to exit?
OK
Cancel
Encrypt
Decrypt
Reset

RESULT:

Upon executing the code for the Image Encryption Decryption mini project, the following results can be observed:

- The user interface is displayed, providing options to choose an image, encrypt it, decrypt it, save the encrypted image, and reset the edited image.
- The user can select an image file using the "Choose" button, and the selected image is displayed in the original image panel.
- Clicking the "Encrypt" button applies the encryption algorithm to the selected image, resulting in an encrypted version of the image. The encrypted image is displayed in the decrypted image panel.
- Clicking the "Decrypt" button applies the decryption algorithm to the encrypted image, resulting in the original image being recovered. The decrypted image is displayed in the decrypted image panel.
- The user has the option to save the encrypted image using the "Save" button. The image can be saved with a desired file name and location.
- The user can reset the edited image to its original format by clicking the "Reset" button.
- The user can exit the application by clicking the "EXIT" button.

The code successfully performs the image encryption and decryption operations, allowing users to select, encrypt, decrypt, and save images. The output is displayed in the user interface, providing visual representation of the original, encrypted, and decrypted images.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION:

In conclusion, the Image Encryption Decryption mini project demonstrates the implementation of image encryption and decryption functionalities. The code successfully encrypts selected images using an encryption algorithm and allows for the decryption of encrypted images, resulting in the recovery of the original image. The user interface provides an intuitive platform for image selection, encryption, decryption, and saving of the edited images.

The project has achieved its objectives of providing a simple and efficient solution for image encryption and decryption. By leveraging the OpenCV library and integrating it with the Tkinter GUI framework, the code effectively performs the required operations on the images.

FUTURE ENHANCEMENTS:

While the current implementation fulfills the basic requirements of image encryption and decryption, there are several potential areas for future enhancements:

1. Enhanced Encryption Algorithms: Explore and integrate more advanced encryption algorithms, such as AES (Advanced Encryption Standard) or RSA (Rivest-Shamir-Adleman), to provide stronger and more secure encryption options.

2. Image Compression: Implement image compression techniques to reduce the file size of encrypted images while maintaining their visual quality. This can help optimize storage space and transmission bandwidth.

3. Password-based Encryption: Enhance the application by incorporating password-based encryption, where users need to provide a password to encrypt and decrypt images. This can add an additional layer of security to protect sensitive image data.

4. Batch Processing: Enable batch processing functionality to allow users to encrypt and decrypt multiple images simultaneously, improving efficiency and productivity.

5. Image Format Support: Extend the application's capabilities to support a wider range of image formats, including popular formats such as PNG, GIF, and BMP, to provide users with more flexibility in selecting and working with different image types.

By implementing these future enhancements, the Image Encryption Decryption application can become more robust, secure, and feature-rich, catering to a wider range of user needs and scenarios.

REFERENCES

1. Zhang, X., Davidson, E. A,"Improving Nitrogen and Water Management in Crop Production on a National Scale", American Geophysical Union, December, 2018.How to Feed the World in 2050 by FAO.

2. Abhishek D. et al., "Estimates for World Population and Global Food Availability for Global Health", Book chapter, The Role of Functional Food Security in Global Health, 2019, Pages 3-24.Elder M., Hayashi S., "A Regional Perspective on Biofuels in Asia", in Biofuels and Sustainability, Science for Sustainable Societies, Springer, 2018.

3. Zhang, L., Dabipi, I. K. And Brown, W. L, “Internet of Things Applications for Agriculture". In, Internet of Things A to Z: Technologies and Applications, Q. Hassan (Ed.), 2018.

4. S. Navulur, A.S.C.S. Sastry, M.N. Giri Prasad,"Agricultural Management through Wireless Sensors and Internet of Things" International Journal of Electrical and Computer Engineering (IJECE), 2017; 7(6) :3492-3499.

5. E. Sisinni, A. Saifullah, S.Han, U. Jennehag and M.Gidlund, "Industrial Internet ofThings: Challenges,Opportunities, and Directions," in IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-4734, Nov. 2018.

6. M. Ayaz, M. Ammad-uddin, I. Baig and e. M. Aggoune, "Wireless Possibilities: A Review," in IEEE Sensors Journal, vol. 18, no. 1, pp. 4-30, 1 Jan.1, 2018.