IMAGE ENCRYPT

PROJECT CODE:

```
from tkinter import *
from tkinter import filedialog, messagebox
from cryptography.fernet import Fernet
from io import BytesIO
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
import string
import random
import io
from PIL import Image

# Global variables for original image
original_image = None
original_image_path = None

# Function to generate a random and strong key
def generate_key():
    # Generate a random 16-byte key for AES encryption
    return Fernet.generate_key()

# Function to encrypt image
def encrypt_image(filepath, key):
    if filepath and key:
        try:
            img = Image.open(filepath)

            # Convert image to bytes
            img_bytes = BytesIO()
            img.save(img_bytes, format='PNG')
            img_bytes.seek(0)
            img_data = img_bytes.read()

            # Encrypt image data using AES
            cipher = Fernet(key)
            encrypted_data = cipher.encrypt(img_data)

            # Save encrypted data to file
            with open(filepath, 'wb') as f:
                f.write(encrypted_data)

            messagebox.showinfo("Success", "Image encrypted successfully!")
```

```
        except Exception as e:
            messagebox.showerror("Error", f"Failed to encrypt image: {str(e)}")
    else:
        messagebox.showerror("Error", "Please select an image file and enter a key.")

# Function to decrypt image
def decrypt_image(filepath, key):
    if filepath and key:
        try:
            # Read encrypted data from file
            with open(filepath, 'rb') as f:
                encrypted_data = f.read()

            # Decrypt image data using AES
            cipher = Fernet(key)
            decrypted_data = cipher.decrypt(encrypted_data)

            # Create an image from decrypted data
            img = Image.open(BytesIO(decrypted_data))

            # Save decrypted image back to file
            img.save(filepath)

            messagebox.showinfo("Decryption Successful", "Image decrypted
successfully!")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to decrypt image: {str(e)}")
    else:
        messagebox.showerror("Error", "Please select an image file and enter a key.")

# Function to send decryption key via email
def send_key_email(recipient_email, sender_email, key):
    # SMTP server settings
    smtp_server = "smtp.gmail.com"  # Update with your SMTP server
    smtp_port = 587  # Update with your SMTP port
    smtp_username = "malumhyterakissa@gmail.com"  # Update with your email address
    smtp_password = "qewehcwsnqhcpure"  # Update with your email password

    # Email content
    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = recipient_email
    msg['Subject'] = "Key for Image Decryption"
    body = f"The key for decrypting the image is: {key}"
    msg.attach(MIMEText(body, 'plain'))

    # Connect to SMTP server and send email
    try:
```

```
        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.starttls()
            server.login(smtp_username, smtp_password)
            server.send_message(msg)
        messagebox.showinfo("Success", "Email sent successfully!")
    except Exception as e:
        messagebox.showerror("Error", f"Failed to send email: {str(e)}")

# Function to open a new window for encryption
def open_encrypt_window():
    def browse_image():
        filepath = filedialog.askopenfilename(title="Select Image File")
        if filepath:
            image_path.set(filepath)
            key_entry.delete(0, END)
            key_entry.insert(0, generate_key().decode())  # Convert bytes to string

    def encrypt():
        filepath = image_path.get()
        key = key_entry.get().encode()  # Convert string to bytes
        encrypt_image(filepath, key)
        if send_email_var.get():
            send_key_email(recipient_email_entry.get(), sender_email_entry.get(),
key_entry.get())

    encrypt_window = Toplevel(root)
    encrypt_window.title("Encrypt Image")
    encrypt_window.geometry("400x400")
    encrypt_window.configure(bg="#FFB6C1")  # Pastel pink background

    image_path = StringVar()
    image_path_entry = Entry(encrypt_window, textvariable=image_path, width=50)
    image_path_entry.pack(pady=10)

    browse_button = Button(encrypt_window, text="Browse", command=browse_image,
bg="#87CEEB")  # Pastel blue button
    browse_button.pack(pady=5)

    key_label = Label(encrypt_window, text="Encryption Key:", bg="#FFB6C1")  # Pastel
pink background
    key_label.pack(pady=5)

    key_entry = Entry(encrypt_window)
    key_entry.pack(pady=5)

    send_email_var = BooleanVar()
    send_email_check = Checkbutton(encrypt_window, text="Send decryption key via
email", variable=send_email_var, bg="#FFB6C1")  # Pastel pink background
```

```
    send_email_check.pack(pady=5)

    recipient_email_label = Label(encrypt_window, text="Recipient Email:",
bg="#FFB6C1")  # Pastel pink background
    recipient_email_label.pack(pady=5)

    recipient_email_entry = Entry(encrypt_window)
    recipient_email_entry.pack(pady=5)

    sender_email_label = Label(encrypt_window, text="Your Email:", bg="#FFB6C1")  #
Pastel pink background
    sender_email_label.pack(pady=5)

    sender_email_entry = Entry(encrypt_window)
    sender_email_entry.pack(pady=5)

    encrypt_button = Button(encrypt_window, text="Encrypt", command=encrypt,
bg="#87CEEB")  # pastel blue button
    encrypt_button.pack(pady=5)

# Function to open a new window for decryption
def open_decrypt_window():
    def browse_image():
        filepath = filedialog.askopenfilename(title="Select Encrypted Image File")
        if filepath:
            image_path_entry.delete(0, END)
            image_path_entry.insert(0, filepath)

    def decrypt():
        key = key_entry.get().encode()  # Convert string to bytes
        filepath = image_path_entry.get()
        if filepath:
            decrypt_image(filepath, key)

    decrypt_window = Toplevel(root)
    decrypt_window.title("Decrypt Image")
    decrypt_window.geometry("400x300")
    decrypt_window.configure(bg="#87CEEB")  # Set background color

    image_path_label = Label(decrypt_window, text="Encrypted Image Path:",
bg="#87CEEB")
    image_path_label.pack(pady=5)

    image_path_entry = Entry(decrypt_window, width=50)
    image_path_entry.pack(pady=5)

    browse_button = Button(decrypt_window, text="Browse", command=browse_image,
bg="#32CD38")  # Green button
```

```
    browse_button.pack(pady=5)

    key_label = Label(decrypt_window, text="Decryption Key:", bg="#87CEEB")
    key_label.pack(pady=5)

    key_entry = Entry(decrypt_window)
    key_entry.pack(pady=5)

    decrypt_button = Button(decrypt_window, text="Decrypt", command=decrypt,
bg="#32CD38")  # Green button
    decrypt_button.pack(pady=5)

# Main window
root = Tk()
root.title("Image Encryptor and Decryptor")
root.geometry("400x300")
root.configure(bg="#C8A2C8")  # Pastel lilac background

title_label = Label(root, text="ImageEncrypt", font=("Arial", 20), bg="#C8A2C8")  #
Pastel lilac background
title_label.pack(pady=20)

encrypt_button = Button(root, text="Encrypt", command=open_encrypt_window,
bg="#87CEEB")  # Pastel blue button
encrypt_button.pack(pady=15)

decrypt_button = Button(root, text="Decrypt", command=open_decrypt_window,
bg="#87CEEB")  # Pastel blue button
decrypt_button.pack(pady=15)

root.mainloop()
```

Step-by-step description of how the code works:

Imports: The code imports necessary modules and libraries, such as Tkinter for GUI, filedialog for opening files, PIL for image processing, and smtplib for sending emails.

Global Variables: It declares global variables original_image and original_image_path to store information about the original image file.

Key Generation: The generate_key() function generates a random 16-byte key for AES encryption using the Fernet library.

Image Encryption: The encrypt_image() function takes a file path and encryption key as input. It opens the image file, converts it to bytes, encrypts the image data using AES encryption, and then saves the encrypted data back to the original file.

Image Decryption: The decrypt_image() function takes a file path and decryption key as input. It reads the encrypted image data from the file, decrypts it using AES decryption, creates an image from the decrypted data, and saves the decrypted image back to the original file.

Sending Key via Email: The send_key_email() function sends the decryption key via email to the recipient's email address. It uses the SMTP protocol to connect to an SMTP server, logs in with the sender's email credentials, and sends an email containing the decryption key.

Encryption Window: The open_encrypt_window() function creates a new window for encrypting images. It allows the user to browse for an image file, generate an encryption key, encrypt the image, and optionally send the decryption key via email.

Decryption Window: The open_decrypt_window() function creates a new window for decrypting images. It allows the user to browse for an encrypted image file, enter the decryption key, and decrypt the image.

Main Window: The main window of the application (root) is created using Tkinter. It contains buttons for encrypting and decrypting images. The buttons call the respective functions to open the encryption and decryption windows.

GUI Styling: The GUI elements are styled using background colors to create a pastel-themed interface.

Main Loop: The root.mainloop() function starts the Tkinter event loop, which listens for user interactions and updates the GUI accordingly.