Secure Image Encryption

IITSOC 24_CYBER_PS3

OUR TEAM

- Team head -
- Name - Krishay Rathaure
- GitHub - https://github.com/Quanmat

- Other Member -
- Name - Gajendra Singh Rana
- GitHub - https://github.com/pratapsingh123om

Problem Statement

Design and implement a secure image Encryption system using powerful cryptographic algorithms such as DES, AES, and RSA. The system will allow users to encrypt images, ensuring that only individuals with the correct decryption key can access the original images.

Project Solution

Key Steps

Algorithm Implementation

User Authentication

Key Management

Usability and Interface

Documentation

Further Improvemnet

Tech stack

- Programming Language: Python

- Front End:
Graphical User Interface (GUI) : tkinter

- Back End:
Cryptographic Libraries: PyCryptodome
Database: SQLite
Password Hashing: bcrypt

Algorithm Implementation

Implementing a hybrid system that uses both symmetric(DES, AES) and asymmetric(RSA) algorithms to encrypt and decrypt images. The DES and AES are used for image encryption and the RSA for symmetric keys(for DES and AES)encryption.

- Dividing the images (data) into 64 and 128 bit blocks to implement DES/AES algorithms.
- Generating AES keys(symmetric) through a random number generator and using RSA to generate a pair of public and private key.
- Encryption of data(images)- using the symmetric key(for DES and AES).
- Encryption of the symmetric key- using the public key(of the user), through RSA.

How it works?

01

Key Generation:

- RSA public and private key.
- Random AES Key.

02

Image Encryption:

- Image encrypted using AES key.

03

AES key encryption:

- Extracting public key from the database.
- AES key encrypted using the public key through RSA.

04

Storage:

- Both the encrypted image and the encrypted AES key are stored locally.

User Authentication

- We have used a simple username and password system.
- Storing hashed passwords, while user registration and verifying those at user login.
- ***Unique username system.***
- Library used for hashing - bcrypt.

Key Management

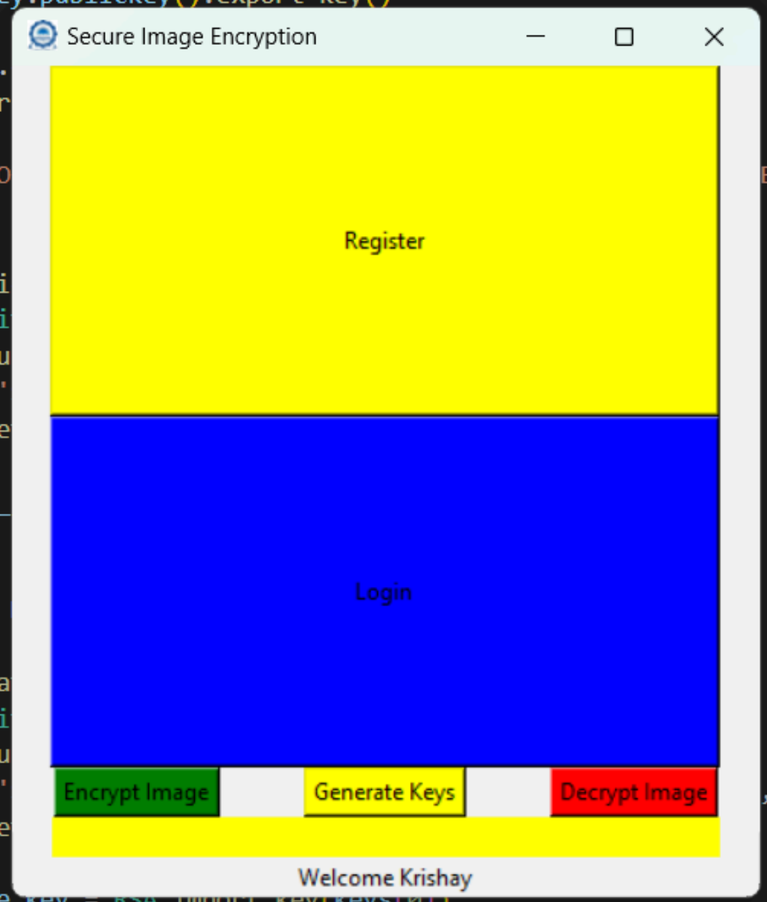
- All the hashed keys and user credentials will be stored in the database(inbuilt SQLite).
- Storing the RSA public key and the RSA private key in the database(using tables) and storing the encrypted symmetric key (of AES) locally.

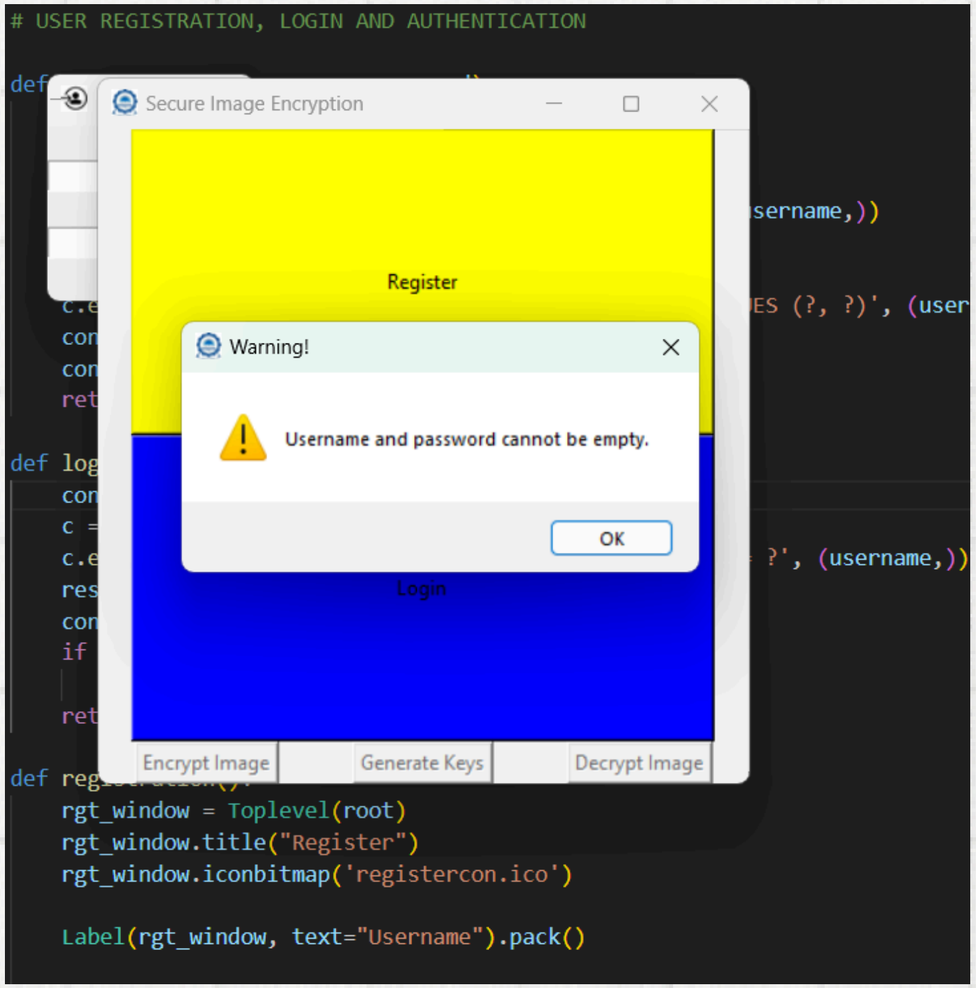
GUI Interface

- A simple way for users to encrypt and decrypt images.
- Built with Tkinter for a user-friendly experience.
- Provides options for selecting images and performing encryption/decryption tasks, including error/success messages.

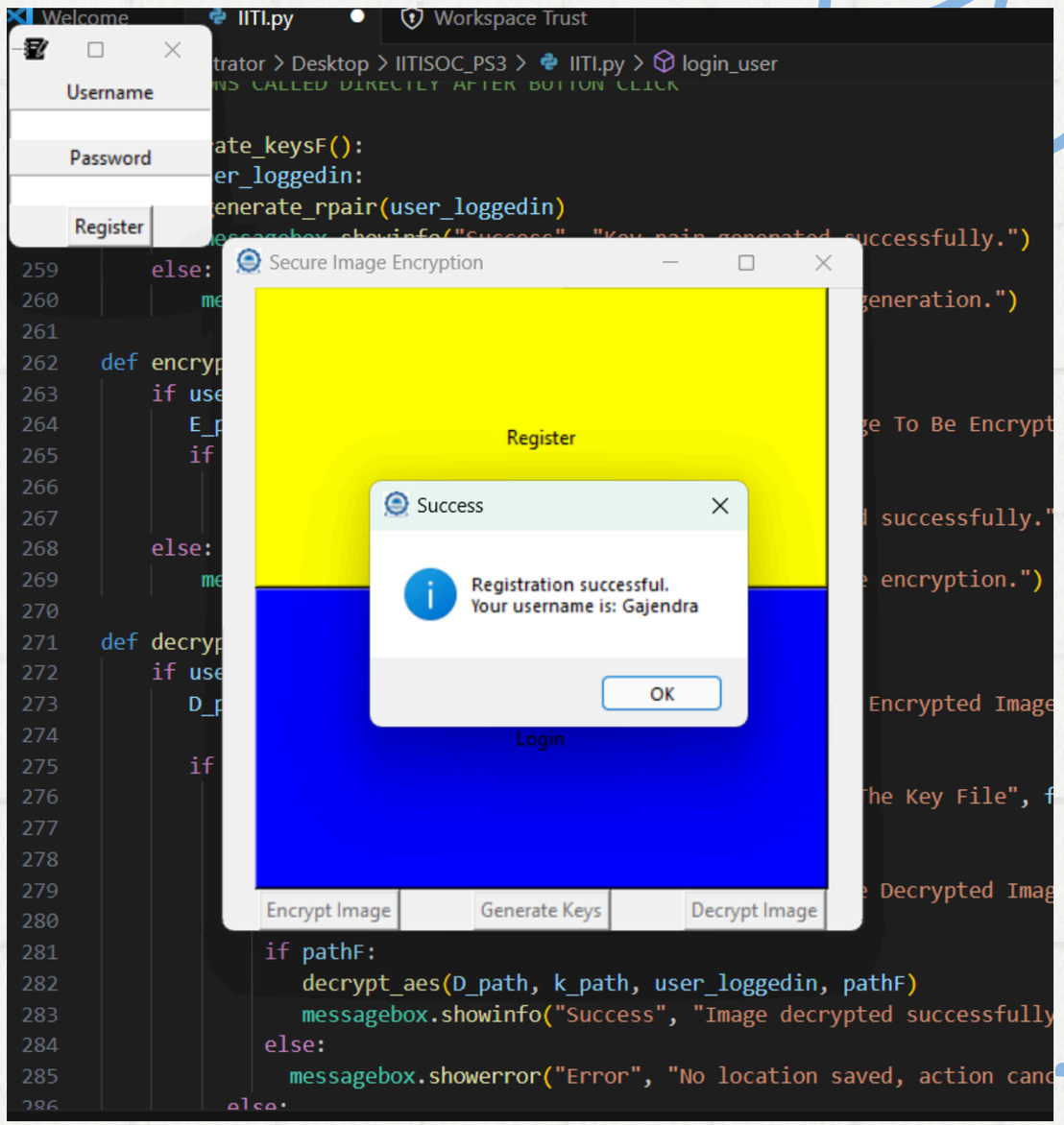
Some Snippets

```
# ALGORITHM IMPLEMENTATION
def generate_rpair(username):
    key = RSA.generate(2048)
    private_key = key.export_key()
    public_key = key.publickey().export_key()

    conn = sqlite3.
    c = conn.cursor
    c.execute('''
        INSERT INTO                                          ES (?, ?, ?)'''
    conn.commit()

def get_rkeys_Publi
        conn = sqli
        c = conn.cu
        c.execute('                                          (username,))
        keys = c.fe
        if keys:
            public_
            return
        else:
            return

def get_rkeys_Priva
        conn = sqli
        c = conn.cu
        c.execute('                                          , (username,))
        keys = c.fe
        if keys:
            private_key = RSA.import_key(keys[0])
            return private_key
        else:
```






```
# USER REGISTRATION, LOGIN AND AUTHENTICATION

def
                                                    sername,))
    c.e
    con
    con
    ret

def log
    con
    c =
    c.e                                                    ?', (username,))
    res
    con
    if

    ret

def registration():
    rgt_window = Toplevel(root)
    rgt_window.title("Register")
    rgt_window.iconbitmap('registercon.ico')

    Label(rgt_window, text="Username").pack()
```




```
ate_keysF():
er_loggedin:
enerate_rpair(user_loggedin)
                                                    successfully.")
        else:
            me                                      generation.")

    def encryp
        if use
            E_p                                     ge To Be Encrypt
            if

        else:                                       l successfully."
            me                                      e encryption.")

    def decryp
        if use
            D_p                                     Encrypted Image

            if                                      The Key File", f

                                                    e Decrypted Imag
            if pathF:
                decrypt_aes(D_path, k_path, user_loggedin, pathF)
                messagebox.showinfo("Success", "Image decrypted successfully
            else:
                messagebox.showerror("Error", "No location saved, action canc
        else:
```

Deliverables

DES, AES and RSA algorithm implementation-

- DES and AES for image encryption.
- RSA for secure key exchange.

User authentication (username and password system, bcrypt)-

- Functions for user registration and user login.

Key management system- Functions to:

- Generate RSA keys
- To store keys (SQLite database)
- For key retrieval.

A GUI developed using tkinter with functions like:

- Register and login buttons
- Key generation button
- Encrypt/decrypt buttons.

Some Questions

Q. What can we do to improve this program?

- Well, a lot can be done to improve it. Firstly, we can create a central database (using for eg - Django). Additionally, we can create a UI, which allows users to share images among themselves, with algorithm implementation.

Some Questions

Q. What is the basic principle of this hybrid system?

This hybrid system merges both DES/AES and RSA algorithms.

- Encryption of data(images) is done using the symmetric key(for DES and AES).
- Encryption of the symmetric key is done using the public key(of the receiver/user), through RSA.

Q. Why use a hybrid system?

To ensure both security (using RSA) and fast processing (using DES/AES).

Some Questions

Q. What makes this code more secure?

- Using bcrypt over hashlib
- Using placeholders to prevent SQL Injections
- By using a GUI, the risk of command-line injection attacks is reduced

Contributions

- Team Leader: (Back End)

- Algorithm Implementation
- User Authentication
- Key Management
- Documentation

- Other member: (Front End)

- GUI Interface
- Algorithm Implementation

Thank you very much!