Secure Photo Gallery Application

Math319 | Project Report

Prepared by:

- Melaf Alobidan
- Hoor Al-ahihi
- Shatha Alzughibi
- Remaz Alrazgan
- Fajr Alnajmi

Content

Introduction:

Our project is called “photoCipher” as the name suggests it's about developing a secure image encryption and decryption application using Python and AES (Advanced Encryption Standard), our app offer users to create accounts to secure the access to their encrypted/decrypted photos ensuring the users confidentiality and integrity of their data, our goal is to provide a user-friendly interface and strong cryptographic protections for the privacy of the users.

Project overview

- Project Description

Our project revolves around three main functions which is uploading and saving the images and image encryption and decryption, once the user upload the image and it will require to enter the key for the AES which is must be 16,24 or 32 character and used a random Initialization Vector (IV) to provide a unique encryption results for each image for better security, the output would be an encrypted file and a scrambled image, the scrambled image acts as additional security layer making it difficult for unauthorized access for the content.
On the decryption side the user will upload the encrypted image, enter the same key and restore the original image from the encrypted date and if the key is not matched an exception will occurred to manage invalid or failed decryption attempts, these function is implemented using Python from many libraries like **tkinter** for a user friendly interface, **SQLite** for user authentication and photo storage, **PIL** for image uploading and processing and **PyCryptomdome** for AES encryption, all of this to ensure an effortless operation for the users.

- **Key features**
    - Image encryption
    - Image decryption
    - User-Friendly Interface
    - Security Controls
    - image Display and Handling

Graphical User Interface (GUI)

We made four interfaces for better user interaction registration page, log in page, uploading page and the gallery page we will dive in and explain each one.

Registration Page:

The purpose of the registration page interface to allows the users to create an account and put their information like username, email, date of birth, password and confirming the password, it also includes fields for input validation and buttons to perform the action of submitting the data or switching to the login page and exceptions for existing username and if the passwords doesn't match, although for a better security we used a hashing algorithm for the password to store it in encrypted form so the data cannot be easily converted back to its original form, even if it is exposed.

the tools and the libraries we used:

tkinter: we used it for creating the graphical user interface

ttk: to Provides styled widgets such as combo boxes for date input.

SQLite: to make serverless database that provides a relational database management system.

Design of the Interface:

The registration page is built in as a frame within the application, there is label to explain
the purpose of each field such as "Username" "Email" and "Password" and for the password and confirming the password field entries we used a masked password entry “*” to
ensure better security by hiding the input characters.
As for date of birth field, we used Combo boxes for day, month, and year to provide a dropdown selection to ensure an accurate input for the date of birth and for the action buttons a
"Register" button that submits the user's data and store it in the database we created, also
an "already have an account?" Button to let the users navigate to the login page easily,
And the layout for all the components we used a grid layout to arrange it efficiently.

key features of the Registration Page:

Error handling: to prevent the program from crashing, we ensured to handle empty fields, unmatched password, and if the username already exists which a descriptive error message will be shown to the user.

Password Hashing: we used the hashing function SHA-256 to protect the user's password in case the database got compromised, avoiding the passwords to be stored as plain text which is a major security risk.

Log in Page:

The log in page is smaller version of the registration, if the user has a registered account the user will need just the username and the password the login function will handle the process of verifying if the user is in our database and provide a message box based on the results, if the user is found a “login successful” message box will appear, else an “Incorrect username or password!” instead will appear.
and because security is important to us, we used the hashing function for the entered user to compare it with the stored password to ensure there is no plain text passwords are ever stored or transmitted.

For the design of the Interface:

The log in includes labels and fields input for the username and the password and action buttons like ‘log in’ to initiates the authentication process or ‘Don't have an account?’ to switch to the registration page

Uploading Page:

after login in it will show the Uploading page which is our main page that has our main functions like uploading images, encryption, decryption and photo gallery page, with uploading function that handle many formats image and if the image is bigger than the canvas it will resize to fit the canvas and a message box will notify the user of the changes, also to prevent the code from crashing if there is error with reading the images an UnidentifiedImageError exception will show.

For the design of the Interface:

There is a (400x300) canvas that will contain the uploaded image and four actions button, the upload button to allow the users to select and upload images from their local file system in any supported format like JPG, PNG and other common type.
For the encrypted and decrypted button when selected the user must enter a key to encrypt and decrypt the image and show the encrypted image in the canvas, the last button is Gallery photo button when selected it will directly show a new window.

Photo Gallery page:

The Purpose of this page is to provide our users with a visual way to view their images and allows filtering between the different types encrypted, decrypted, and both images stored in the database and displays them dynamically.

the tools and the libraries we used:

tkinter: we used it to create the graphical user interface.
SQLite: to Handle the storage and retrieval of image data from the database we used.
Pillow (PIL): to Process and display images, like resizing and error handling.
io: to convert the BLOB data and retrieve it from the database into a format that is usable by Pillow.

Design of the Interface:

The photo gallery page is built in a Top-level window with a title and filter dropdown at the top, which allows the users to filter the displayed images from Encrypted images to Decrypted images or both, then a dynamic photo display frames updates automatically based on the selected filter, to make a clean and professional layout we arranged the Images in a row with padding ensuring a responsive and seamless user experience.

Key components of the Photo Gallery Page:

- Dynamic Image Loading: The Images are dynamically fetched from the database of the users images and its displayed based on the selected option from the filter, also to ensure a real time response we converted each image from BLOB format to a displayable format using the blob_to_image function.
- Error Handling: If an image cannot be identified an error message is shown to the user, to prevent crashes and maintain a smooth experience for the user.

Functions and Implementations

- **User Authentication**

Function Description: this function authenticates the authorized users who can access the application function, it implemented in the registration page and login page where the users provide their information and store it in secure database, and in the login page we validate the existing users by checking their entered information, also passwords are hashed before being stored this prevent SQL injection.

- **File Upload and Image Handling**

Function Description: this function let our users to upload images for encryption and .enc encrypted file for decryption, after the uploading image will be resized to fir the canvas, and there is handled the errors for checking the issues with unsupported format.

- **Encryption Functionality**

Function Description: this function transform the input image into in encrypted form that cannot be understood, we used AES algorithm with CBC (Cipher Block Chaining) mode form the Crypto library to encrypt by a user provided key and random generated Initialization Vector (IV), we added a key length validation function to make the user must enter 16,24,32 character then saves the encrypted file and generate a scrambled image for user verification by encrypting the byte data to a numby array and reshape it to fit the original image.

- **Decryption Functionality**

Function Description: this function is the opposite from encryption function, restoring the original image by uploading the .enc file and using the same encryption key, first we take the key from the user then retrieve first 16 bytes of the encrypted file of the Initialization Vector (IV) the same one we used in the encryption which is very important to save the decrypted data as PNG file and show the original image it in the canvas, also we used the key length validation function and try-except for error handling to catch any decryption error.

Testing and Validation

Our program PhotoCipher was thoroughly tested to ensure the correctness, functionality, and the reliability of the program. Several test cases were defined and executed to validate the functionality of the program:

Test case	Input	Expected output	Result
Register new user	Unique username	Account is created successfully	Pass
Register with duplicate username	Existing username, new password	Error: “username already exists.”	Pass
Login with correct credentials	Valid username, and password	User is logged in successfully	pass
Login with incorrect password	Valid username, incorrect password.	Error: “invalid username or password”	pass
Encrypt an image with valid key	Image files, 16-byte or 28-byte or 32-byte key	encrypted files saved unreadable	Pass
Decrypt an encrypted image	Encrypted file, correct key	Original image file is restored.	pass
Decrypt with an incorrect key	Encrypted file, wrong key.	Error: "Decryption failed: Padding is incorrect"	pass
Insert image into the database	Encrypted and decrypted files	Files stored in the correct database columns	Pass
Fetch and display encrypted images	Filter: Encrypted	Only encrypted images are displayed	Pass
Fetch and display decrypted images	Filter: Decrypted	Only decrypted images are displayed	Pass
Fetch all images	No Filter	Both encrypted and decrypted images are displayed	Pass

Challenges and Solutions

During the development of our project, we faced a lot of errors and challenges that required innovative solutions and teamwork we solved some of them found another way for the rest so here is some challenges that we faced:

First problem we had Limited Python Expertise Out of the five members, only two were familiar with using python, so tasks were divided based on strengths and knowledge, and we shared learning resources to improve the rest of the team, also this project required the use of unfamiliar libraries for tasks like image processing and encryption, so we dedicated our time to study them and improve our skill.

Some of the challenges in the code itself, like the encryption output length sometimes its exceeded or fell short of the size so it’s needed to reshape it to the original image dimensions, this lead to a runtime errors like "ValueError: cannot reshape array" so we solved it by flattened the encrypted data and used NumPy's resizing function to match the dimensions of the image array.
Unfortunately, some problem we couldn’t solve but found another way for it, like decryption directly from the scrambled image, because when image is saved as scrambled version it goes to several transformations (e.g., compression, pixel data adjustments) that makes it different from the original encrypted bytes and the for decryption to work, you must construct the exact sequence of bytes used during encryption that's why it can be difficult to achieve so we relied on the .enc file that contain the raw encrypted data.

Also, adding Filer Function for different users was challenging but we made the database quires retrieve images based on their encryption status (e.g., encrypted, decrypted, or all) we tested it a lot to ensure better accuracy, and we had problem with Gallery Duplication so we added some logic to ensure only one version of an image scrambled or decrypted was displayed, keeping the gallery clean and organized, And to make User-Specific Image Gallery a user id column was introduced in the database to associate images with users, ensuring only the logged-in user’s images were displayed in the gallery

work section

Names	work section
Melaf Alobidan	- Implementation of the uploading and encryption and decryption pages - Report
Hoor Al-ahihi	- Implementation of the gallery and registration pages - Report
Shatha Alzughibi	Implementation of the log in page - Report
Remaz Alrazgan	-Presentation -Editing
Fajr Alnajmi	-Presentation -Editing