Kingdom of Saudi Arabia
Ministry of Education
Qassim University
College of Computer
Information Technology Dept.

المملكة العربية السعودية
وزارة التعليم
جامعة القصيم
كلية الحاسب
قسم تقنية المعلومات

Secure Photo Gallery Application

Students:

xxx

xxx

xxx

xxx

xxx

Supervisor:

xxx

Qassim-Saudi Arabia
1445/1446 (2024/2025)

Table of content

1. Introduction

The objective of this project is to develop practical skills in implementing the AES cryptosystem to enhance data privacy in real-world applications. The Secure Photo Gallery Application serves as a case study, focusing on user authentication, data encryption, and an intuitive user interface. This report outlines the methodology, implementation details, challenges faced, and measures taken to ensure the privacy and usability of the application.

2. Methodology

2.1 Framework Selection

Django was chosen as the web framework for the Secure Photo Gallery Application. Django is an open-source web framework designed for developing web applications with the Python programming language, known for its high-level features and capabilities. Our choice was motivated by several key factors that align with the project's objectives of security, usability, and rapid development. Below are the reasons for selecting Django:

- **Robust Features**: Django comes with a comprehensive set of built-in features that streamline the development process. It includes an ORM (Object-Relational Mapping) system, which simplifies database interactions by allowing developers to work with Python objects instead of SQL queries. This was particularly beneficial for managing user data and photo uploads in a relational database.
- **Security Practices**: Security is a critical concern for any application, especially one that handles sensitive user data like personal photos. Django provides several built-in security features, including protection against SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking. These features help ensure that the application adheres to best practices in security, reducing the risk of vulnerabilities.
- **User Authentication**: Django's built-in authentication system simplifies user management, allowing for secure registration, login, and session handling. This functionality is crucial for the Secure Photo Gallery Application, as it ensures that only authenticated users can access their photos and personal data.

By leveraging Django's robust features and security practices, we were able to develop the Secure Photo Gallery Application with a strong focus on user privacy, data protection, and overall usability, making it a suitable choice for this project.

2.2 Encryption Algorithm

The **AES (Advanced Encryption Standard)** encryption algorithm was selected for several compelling reasons, aligning with the project's requirements and objectives. AES is renowned for its strength, efficiency, and suitability for securing sensitive data, making it an ideal choice for the Secure Photo Gallery Application.

2.2.1 Justification for Choosing AES:

- **Strength and Efficiency**: AES is widely recognized as one of the most secure encryption algorithms available today, extensively analyzed and trusted globally to protect sensitive data. It operates efficiently on both software and hardware platforms, allowing for rapid encryption and decryption processes. This efficiency is crucial for applications that require real-time data access, such as our photo gallery.
- **Support for Varying Key Lengths**: AES supports key lengths of 128, 192, and 256 bits, providing flexibility in balancing security and performance. For this project, a **128-bit key** length was utilized, which is considered secure for most applications while ensuring fast processing speeds.
- **Compliance with Project Requirements**: The project specifications explicitly required the use of the AES encryption algorithm, making it a necessary choice for fulfilling the assignment's objectives.

2.2.2 How AES Works:

AES operates as a symmetric key encryption algorithm, which means the same key is used for both encryption and decryption. Here's a brief overview of how AES functions:

1- **Block Size**: AES processes data in fixed-size blocks of 128 bits (16 bytes). If the input data is larger than this block size, it is divided into multiple blocks for processing.

2- **Key Expansion**: The original encryption key is expanded into a set of round keys using a key schedule. This process generates multiple keys from the original key, which are used in successive rounds of encryption.

3- **Rounds of Encryption**: AES performs a series of transformations on the data block over multiple rounds, depending on the key length:

   a. **10 Rounds** for 128-bit keys
   b. **12 Rounds** for 192-bit keys
   c. **14 Rounds** for 256-bit keys

   Each round consists of several operations:

   d. **SubBytes**: A non-linear substitution step where each byte in the block is replaced with a corresponding byte from a fixed substitution table (S-box).
   e. **ShiftRows**: A transposition step where the rows of the block are shifted cyclically to the left.
   f. **MixColumns**: A mixing operation that combines the bytes of each column, providing diffusion in the cipher.
   g. **AddRoundKey**: The round key is combined with the block using the XOR operation.

4- **Final Round**: The final round of AES excludes the MixColumns step, consisting only of SubBytes, ShiftRows, and AddRoundKey.

5- **Decryption**: The decryption process is similar to encryption but reverses the order of operations and uses the round keys in reverse order. The inverse operations for SubBytes, ShiftRows, and MixColumns are applied to retrieve the original plaintext (or original bytes in our case).

By employing the AES encryption algorithm, the Secure Photo Gallery Application ensures robust protection of user data. The combination of AES's strength, efficiency, and compliance with project requirements makes it an excellent choice for safeguarding sensitive information against unauthorized access. This approach not only enhances the security of the application but also reinforces the importance of encryption in modern web applications.

2.3 System Architecture

The application architecture for the Secure Photo Gallery Application is designed with a clear separation of concerns, ensuring maintainability and scalability. The components of the architecture include:

- **Frontend**: The user interface is built using the Django template engine, which allows for dynamic content rendering within HTML pages. This approach facilitates the integration of Python code directly into HTML, enabling a seamless interaction between the user and the application. The frontend is designed to be user-friendly, providing intuitive navigation for tasks such as registration, login, photo uploads, and viewing uploaded images.

- **Backend**: The backend is powered by Django views that handle the application logic. This includes user authentication, managing photo uploads, and processing encrypted data. Django's robust framework ensures that the application adheres to best practices in security and data management.
- **Database**: User credentials are securely stored in an SQLite database. SQLite was chosen for its lightweight nature and ease of setup, making it ideal for development and small-scale applications. While user data is stored in the database, the actual encrypted photo files are stored directly in the filesystem. This design choice allows for efficient retrieval and management of files while keeping sensitive data encrypted and secure.

This architecture not only emphasizes a modular approach but also ensures that each component can be developed and maintained independently, contributing to the overall robustness of the application.

2.4 Implementation Phases

1- **User Authentication**: Implemented registration and login functionalities using Django's built-in authentication system, ensuring secure password management. A unique encryption private key is generated for each user at account creation, allowing for secure encryption of user data.

2- **User Interface Design**: Designed and built the user interface using HTML and the Django template engine. The interface was created to be intuitive and user-friendly, facilitating easy navigation through registration, login, photo upload, and viewing functionalities. User feedback was incorporated during the design process to enhance usability.

3- **Photo Upload**: Developed the upload feature to accept image files from users. Upon upload, images are validated and encrypted using the AES algorithm before being stored in the filesystem. User feedback is provided to confirm successful uploads, enhancing the user experience.

4- **Photo Retrieval**: Users can view their uploaded photos through an intuitive interface. When a user requests to view an image, the application retrieves the encrypted file and decrypts it using the user's unique AES key, ensuring that only authorized users can access their images.

3. Implementation Details

3.1 User Registration and Authentication

- **Registration**: Users are required to provide a username and password. Upon successful registration, a unique AES key is generated using *os.urandom(16).hex()* and stored in the database.
- **Authentication**: Utilized Django's built-in authentication system to manage user sessions securely.

3.2 AES Encryption and Decryption

- **Encryption:** The *encrypt_image(data: bytes, key: str) -> bytes* function encrypts image data using the AES encryption algorithm in CBC mode.
    - **Input**:
        - *data*: The image data to be encrypted, provided as a byte string.
        - *key*: The AES key as a hexadecimal string, which is converted into bytes for use in encryption.
    - **Process:**
        - A random Initialization Vector (IV) is generated using *os.urandom(16)*. This IV is essential for ensuring that the same plaintext results in different ciphertexts each time it is encrypted.
        - The data is padded using PKCS7 padding to ensure its length is a multiple of the AES block size (128 bits).

        - The data is then encrypted using the AES algorithm, with the IV prepended to the resulting ciphertext to facilitate decryption.
    - **Output:** The function returns the encrypted data, which consists of the IV followed by the encrypted image data.
    -
- **Decryption**: The *decrypt_image(encrypted_data: bytes, key: str) -> bytes* function decrypts the previously encrypted image data.
    - **Input:**
        - *encrypted_data*: The encrypted data, which includes the IV and the encrypted content.
        - *key*: The AES key as a hexadecimal string, which is also converted into bytes.
    - **Process:**
        - The IV is extracted from the first 16 bytes of the *encrypted_data*.
        - The remaining bytes are the encrypted content, which is decrypted using the AES algorithm with the extracted IV.
        - The decrypted data is then unpadded using PKCS7 unpadding to retrieve the original image data.
    - **Output**: The function returns the decrypted image data as a byte string, allowing users to access their original files.

3.3 User Interface

The user interface is designed to be intuitive using HTML, CSS, and Django template engine, allowing users to easily navigate through registration, login, photo upload, and viewing functionalities.

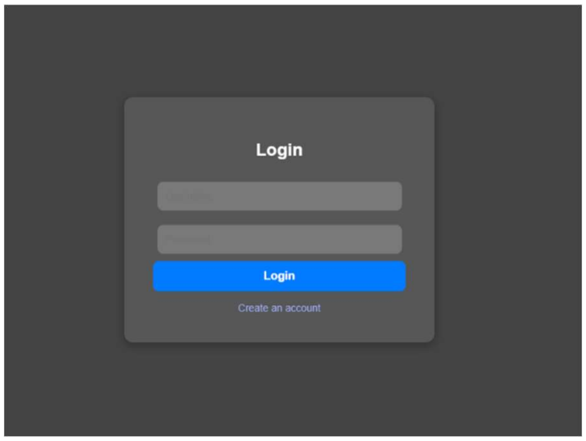

Figure 3.3.1: Login Page of the Secure Photo Gallery Application

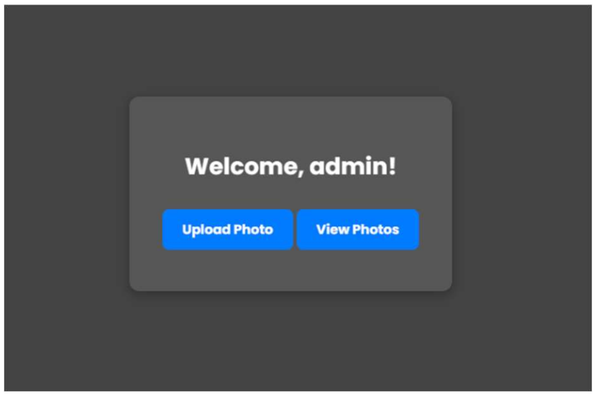

Figure 3.3.2: Home Page of the Secure Photo Gallery Application

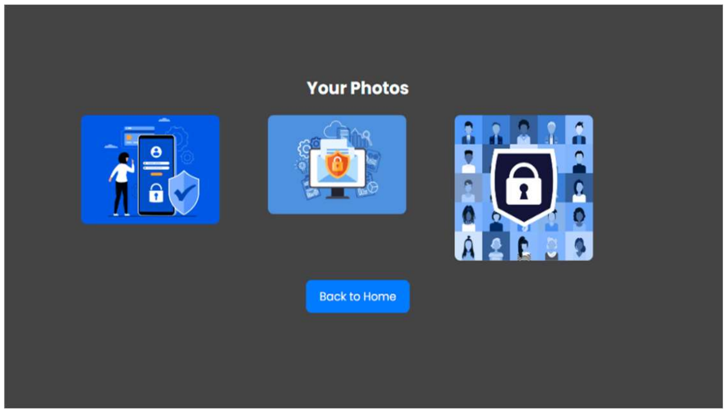

Figure 3.3.3: Encrypted Images Viewing Page of the Secure Photo Gallery Application

3.4 Database Design

The UML diagram illustrates the database schema, including the relationships between users, and their encrypted images. Each table represents an entity with its attributes, and the relationships demonstrate how data is interconnected within the application.

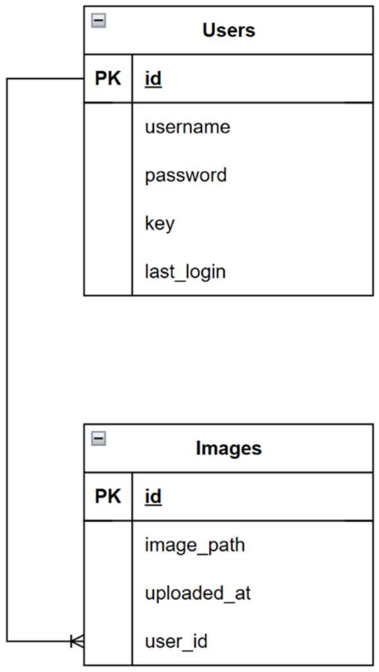

Figure 3.4.1: UML Diagram for the Secure Photo Gallery Application

4. Challenges Faced

4.1 Key Management

Managing AES keys securely was a significant challenge. Ensuring that keys are generated and stored securely, while preventing unauthorized access, required careful design considerations. Currently, the system stores the AES keys as plaintext in the users table in the database. While this approach facilitates easy access for encryption and decryption, it poses a security risk. In the future, we may enhance security by encrypting the keys before storage or utilizing a dedicated key management solution to protect them from unauthorized access.

4.2 Applying the AES encryption in Python

Implementing AES encryption in Python presented its own set of challenges. However, thanks to the cryptography library, the process became significantly easier. This library provides a straightforward API for AES encryption and decryption, allowing for efficient implementation of cryptographic functions without needing to manage low-level details. The documentation and community support further helped in overcoming initial hurdles.

4.3 Image Processing and Relationships:

A key challenge was storing the encrypted images and ensuring they were properly linked to the users who uploaded them. This involved designing a robust data model that associates each encrypted image with its corresponding user in the database. Managing these relationships required careful planning to ensure data integrity and efficient retrieval, particularly when users needed to access their uploaded images.

5. Ensuring Privacy and Usability

5.1 Privacy Measures

- **Strong Encryption**: The application employs AES encryption to protect user data at rest. User passwords are securely hashed, ensuring that even if the database is compromised, the passwords remain inaccessible and cannot be easily retrieved. Each user has their own unique AES key, which ensures that only the respective user can decrypt and view their photos. This means that sensitive information is safeguarded against unauthorized access.
- **Secure Key Storage**: User-specific AES keys are stored securely in the database, minimizing the risk of exposure. This design helps to ensure that keys are not easily accessible to unauthorized parties.
- **Robust Authentication**: The application mandates user authentication for all actions, ensuring that only authorized users can access their data. This includes measures such as secure password management and session handling, which further protect user accounts and data integrity.

5.2 Usability Features

- **Intuitive Design**: The application interface is user-friendly, featuring a clear layout and straightforward navigation. Users are guided with clear instructions and prompts for actions like registration and photo uploads, making it easy to understand and use the application.

- **Feedback Mechanisms**: Users receive immediate feedback on their actions, such as confirmation messages for successful uploads and specific error messages when issues occur. This real-time feedback enhances user confidence and helps resolve problems quickly.
- **Responsive Layout**: The design adapts to various screen sizes, ensuring accessibility on both mobile and desktop devices. This responsiveness allows users to interact with the application comfortably, regardless of their device, promoting a consistent experience.

6. Results and Discussion

The Secure Photo Gallery Application successfully demonstrates the application of AES encryption in safeguarding user data.

6.1 Encryption and Data Integrity

Testing confirmed that the encryption and decryption processes functioned correctly, maintaining data integrity while ensuring privacy. Encrypted images are stored securely in the filesystem, demonstrating that user data is protected against unauthorized access.

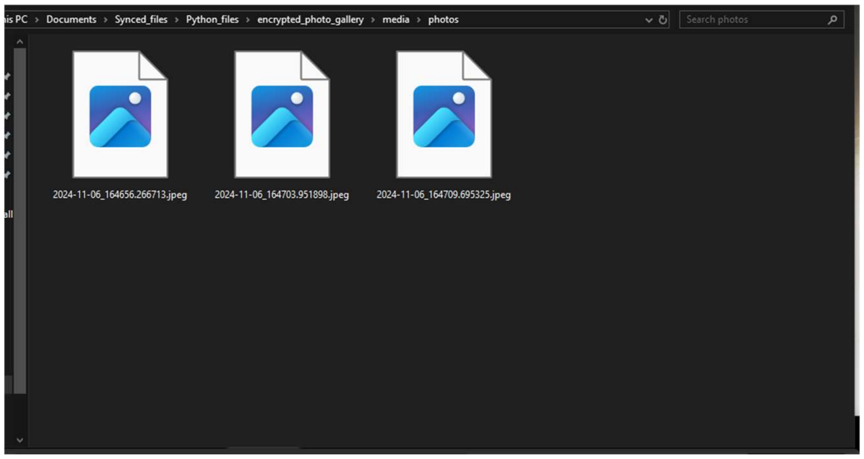
is PC › Documents › Synced_files › Python_files › encrypted_photo_gallery › media › photos
Search photos
2024-11-06_164656.266713.jpeg
2024-11-06_164703.951898.jpeg
2024-11-06_164709.695325.jpeg
all

Figure 6.1.1: Encrypted Images Stored in the Filesystem

6.2 Retrievability of Images

Users can easily retrieve their uploaded images through the application interface. Upon requesting an image to view, the application decrypts the corresponding file, allowing users to access their photos seamlessly.

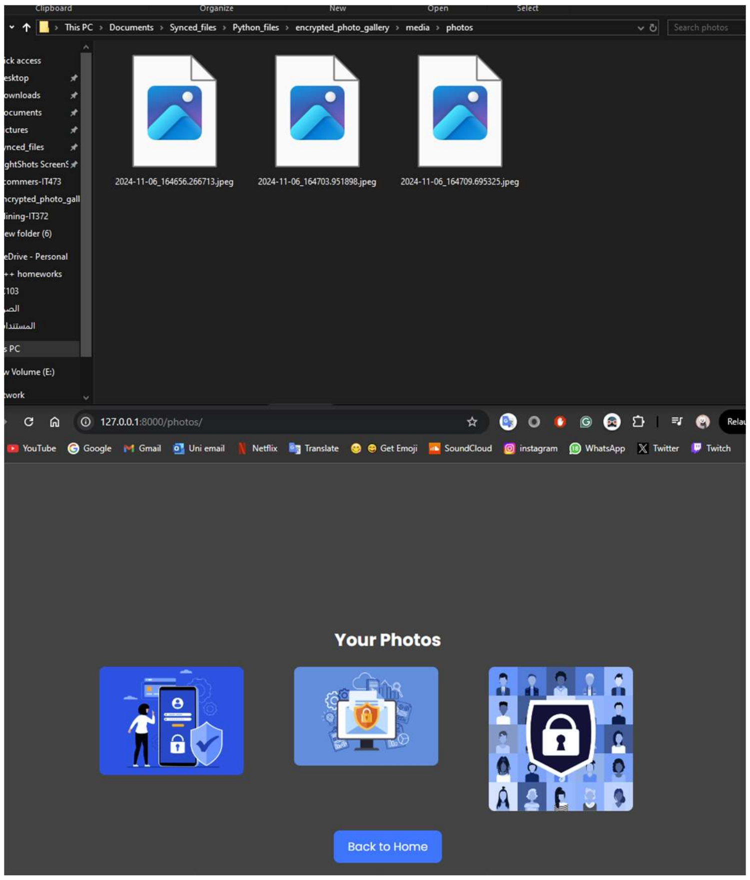
This PC › Documents › Synced_files › Python_files › encrypted_photo_gallery › media › photos
2024-11-06_164656.266713.jpeg
2024-11-06_164703.951898.jpeg
2024-11-06_164709.695325.jpeg
127.0.0.1:8000/photos/
Your Photos
Back to Home

Figure 6.2.1: Encrypted Images in Filesystem and Their Visibility in the Application

6.3 Future Work

For future enhancements, the following features could be implemented:

- **Two-Factor Authentication (2FA)**: Implementing two-factor authentication would add an additional layer of security during the login process. By requiring users to verify their identity through a secondary method, such as a mobile app or SMS code, the application can significantly reduce the risk of unauthorized access.
- **Encrypted Key Storage**: To further enhance security, user-specific AES keys could be encrypted before being stored in the database. This would ensure that even if the database

is compromised, the encryption keys remain protected, making it substantially more difficult for attackers to access user data.

- **Account Recovery for Forgotten Passwords**: Introducing an account recovery mechanism would allow users to regain access to their accounts in the event they forget their passwords. This could involve sending a password reset link to the user's registered email or using security questions to verify their identity before allowing password changes.

7. Conclusion

The development of the Secure Photo Gallery Application has provided valuable insights into the practical implementation of encryption techniques to enhance data privacy. The project not only met the course objectives for MATH319 but also emphasized the importance of security in web applications. Future work will focus on expanding the application's capabilities while maintaining a strong commitment to user privacy and security.

8. References

Django Documentation, https://docs.djangoproject.com/

Django Tutorial, https://www.youtube.com/watch?v=nGIg40xs9e4

How AES works, https://www.youtube.com/watch?v=O4xNJsjtN6E

How AES works, https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption

Cryptography Library Documentation, https://cryptography.io/en/latest/