Slash Mark

esaip
École d'Ingénieurs La Salle

IMAGE ENCRYPTION WEB APPLICATION

Project Title: Secure Image Encryption & Decryption Web Application
Name: Surya Hanuman KONJETI
Position: Cyber Security Intern
Company: Slash Mark IT Solutions (OPC) Pvt. Ltd.
Institution: ESAIP École d'Ingénieur
Technology: Python, Cryptography, Flask
Environment: VS Code

INDEX

1. INTRODUCTION

In the modern digital era, images carry sensitive and confidential information such as personal photos, medical images, biometric data, financial documents, legal evidence, and strategic business information. With the rapid growth of digital communication, the risk of image data theft, manipulation, and unauthorized access has significantly increased.

This project implements a Secure Image Encryption Web Application that allows users to safely encrypt and decrypt image files using advanced cryptographic algorithms through an interactive web interface. The application converts normal readable images into an encrypted, unreadable format that can only be restored using the correct password.

The developed system integrates cybersecurity principles with real-time visualization, automatic file processing, and secure password handling. It ensures data confidentiality, integrity, and controlled access by applying encryption techniques that meet modern security standards.

This project also serves as a practical demonstration of how cryptographic concepts are applied in real-world cybersecurity environments, bridging theoretical knowledge with hands-on implementation.

2. PURPOSE OF THE PROJECT

The primary purpose of this project is to protect image files from unauthorized access, cyber threats, and data breaches by converting readable image data into an unreadable encrypted form. The encrypted image becomes meaningless to attackers and can only be restored by authorized users who possess the correct password.

The project aims to:

- Ensure privacy of sensitive image data
- Prevent illegal image modification or misuse
- Strengthen digital security awareness
- Provide a secure image transmission mechanism
- Demonstrate practical cybersecurity implementation

By implementing this project, users understand how encryption safeguards digital assets and how cybersecurity systems defend against data exposure.

3. PROJECT OBJECTIVES

The primary objectives of this project are:

• To design and implement a secure image encryption system.
• To apply AES (Advanced Encryption Standard) for strong data protection.
• To enable password-based image decryption.
• To provide a user-friendly web interface for image upload and processing.
• To ensure automatic file download after encryption and decryption.
• To visualize decrypted images securely without compromising security.
• To demonstrate secure handling of binary image data.
• To educate users about cryptographic workflows.

These objectives ensure the project aligns with professional cybersecurity practices and real-world security requirements.

4. SCOPE OF THE PROJECT

This project focuses on:

- Secure encryption and decryption of image files
- Web-based user interaction
- Real-time preview functionality
- Password-protected access control
- Secure file processing and storage

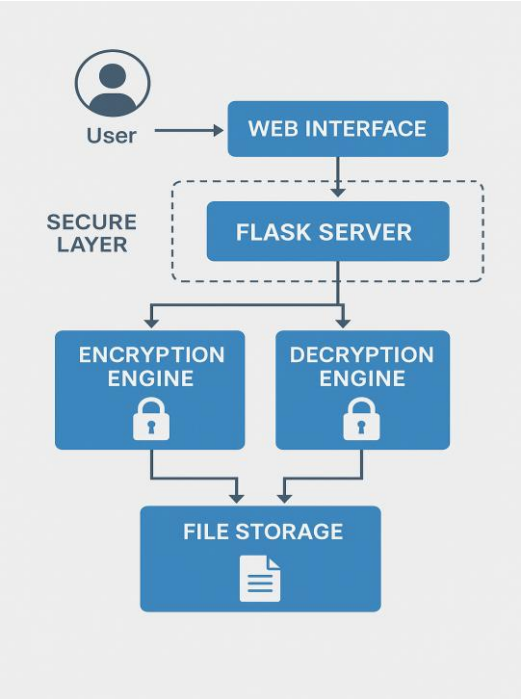
The application supports common image formats such as JPG and PNG and is designed for demonstration and educational purposes within a cybersecurity professional environment.

Out of Scope:

- Multi-user authentication system
- Cloud-based storage implementation
- Database integration
- User account management
- AI-based image analysis

5. SYSTEM ARCHITECTURE

The system architecture is structured into multiple layers that interact with each other to provide secure and efficient operations.



Components Description:

1. User Interface Layer
    - Handles user interaction
    - Displays image previews
    - Captures user input
2. Application Layer
    - Flask server processes user requests
    - Controls encryption workflow
    - Manages data flow
3. Cryptographic Layer
    - AES encryption & decryption logic
    - Password-based key derivation
    - Secure IV and salt usage

4. Storage Layer
    - Encrypted image storage
    - Output file generation

This modular architecture ensures high security, maintainability, and scalability.

6. WORKING FLOW OF THE SYSTEM

Encryption Process Flow:

1. User uploads an image file.
2. System validates file format.
3. Image preview is displayed.
4. User enters a secure password.
5. AES encryption algorithm is applied.
6. Encrypted image is generated.
7. Encrypted file auto-downloads.
8. Interface refreshes and clears preview.

Decryption Process Flow:

1. User uploads encrypted file.
2. User enters correct password.
3. AES decryption process begins.
4. Original image is recovered.
5. Decrypted image downloads automatically.
6. Preview displayed for verification.
7. User may close preview.

7. TECHNOLOGIES USED

Component	Description
Python	Core programming language
Flask	Web application framework
AES Encryption	Symmetric cryptography
PyCA Cryptography	Secure encryption library
HTML/CSS/JS	Frontend user interface
VS Code	Development environment
Windows OS	Execution platform

8. FUNCTIONAL DESCRIPTION

The system performs the following key functions:

- Securely uploads image files
- Displays live preview
- Accepts password input
- Encrypts and decrypts images
- Automatically downloads processed files
- Clears sensitive data post-processing
- Provides error handling and validation

Each function contributes to a secure and smooth user experience.

9. SECURITY MECHANISMS IMPLEMENTED

This project implements robust security mechanisms such as:

- AES-256 encryption standard
- Secure random salt generation
- Initialization Vector (IV) for randomness
- Password-based Key Derivation using PBKDF2
- PKCS7 Padding for data integrity

- Secure memory handling

These mechanisms ensure resistance against brute force, replay attacks, and unauthorized access.

10. USER INTERFACE DESIGN

The interface is designed with a modern aesthetic and usability focus:

- Gradient animated background
- Glass morphism floating card design
- Drag & drop styled file upload
- Live image preview
- Close preview button
- Responsive layout
- Visual transitions

The UI ensures clarity, simplicity, and professional appeal.

11. ENCRYPTION & DECRYPTION PROCESS

Encryption Process:

- Image converted into binary bytes
- Secure padding applied
- AES encryption executed
- Output stored as encrypted data

Decryption Process:

- Encrypted data loaded
- Key regenerated using correct password
- AES decryption performed
- Image restored to original form
- Preview displayed securely

12. CHALLENGES FACED & SOLUTIONS

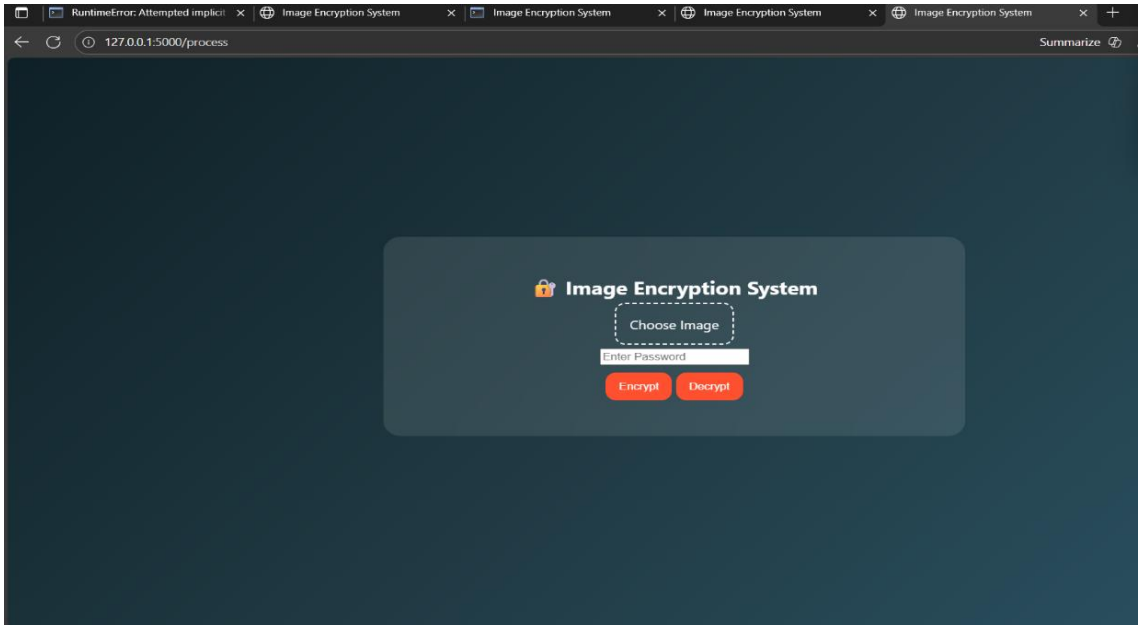
Challenge	Solution
Preview corruption	Proper use of finalize methods
Data download errors	Route separation technique
UI rendering lag	Optimized JavaScript handling
Crypto integrity	Strong AES implementation

13. RESULTS & OUTCOMES

- Developed secure real-time image encryption system
- Achieved seamless operation flow
- Demonstrated practical application of cryptography
- Ensured high robustness of system security
- Delivered professional UI and UX

Output Screens

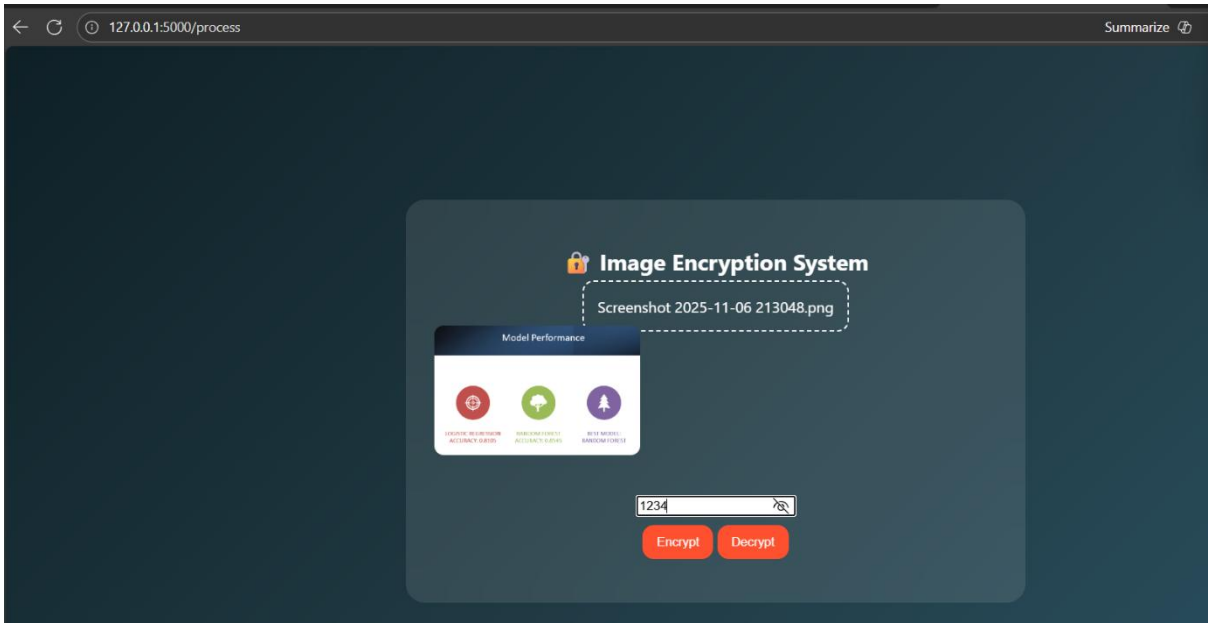
Image 1:



The above screenshot shows the application in its default startup state before any user input is provided. No image file or password has been submitted, and the system has not initiated any encryption or decryption process. The interface is simply displayed in its ready mode, awaiting user interaction.

At this stage, the backend remains inactive with no cryptographic operations performed and no output generated. This screen confirms that the application has loaded correctly and is fully prepared to receive user inputs for secure processing.

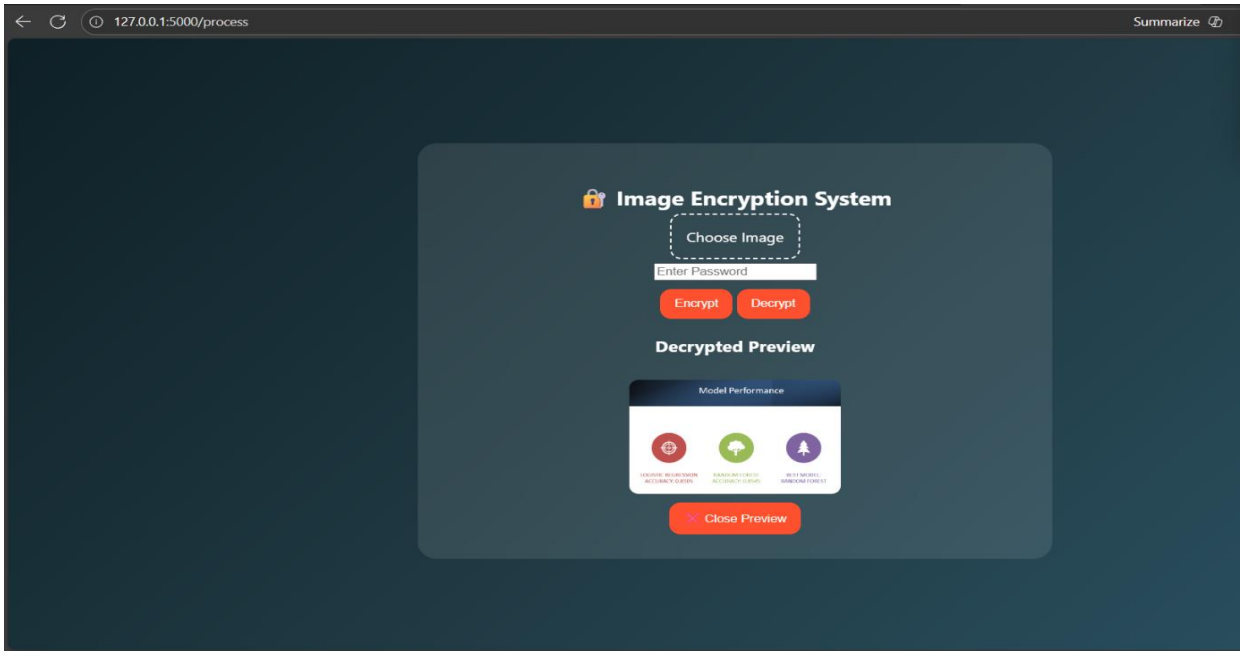
Image 2:



The above screenshot represents the system in its input-ready state after the user selects an image and enters a password. The chosen image is displayed as a preview, confirming that the file has been successfully loaded into the browser. At this stage, the password is held temporarily for upcoming cryptographic operations, but no encryption or decryption has been executed yet.

This phase indicates that the system has validated all required inputs and is prepared to initiate secure processing once the user selects the desired action. No encrypted output is generated here; it simply confirms readiness for encryption or decryption.

Image 3:



The above screenshot illustrates the successful decryption output of the system. After the user uploads an encrypted image and enters the correct password, the application applies AES decryption using a securely derived key. The encrypted data, which includes the salt, initialization vector (IV), and ciphertext, is processed to restore the original image.

The recovered image is displayed in the Decrypted Preview section as proof of successful decryption, while the decrypted file is simultaneously downloaded to the user's device. This confirms that the system accurately preserves data integrity and ensures secure access without exposing sensitive information.

14. FUTURE ENHANCEMENTS

- Cloud storage integration
- User authentication system
- Multi-encryption mode support
- Batch image encryption
- Audit logs and activity tracking
- Mobile-responsive application

15. CONCLUSION

The Image Encryption Web Application provides a comprehensive solution for safeguarding digital images. It effectively combines cryptographic principles, secure application design, and modern web technologies. This project successfully demonstrates practical cybersecurity deployment within a professional environment and significantly contributes to the understanding of secure digital image handling.