

Security Project

**Image Encryption Based on AES Algorithm
and XOR Operation**

Group Members

Khoubaib Bourbia

Islem Chouayakh

Maha Jdidi



Table of contents

Introduction :	3
Scope :	3
Why AES for Image Encryption:	4
AES (Advanced Encryption Standard)	4
RSA (Rivest-Shamir-Adleman)	6
AES vs. RSA:	9
Scan and XOR-based Algorithms	11
Logistic Chaotic Maps:	12
AES vs. Logistic Chaotic Map:	15
Design and Implementation constraints:	17
Main Concepts:	17
Main components:	17
Functional Flow:	18
Functional Requirements	21
Non-Functional Requirements	22
Objectives of the Proposed Solution:	23
Project Design	24
Methodology	29
Project Tools:	30
Project Phases:	31
1.Project Initiation Phase	31
2.Design Phase	31
3.Development Phase	32
4.Deployment Phase	32
Implementation Details:	33
Evaluation:	34
Conclusion:	35
Reference :	36

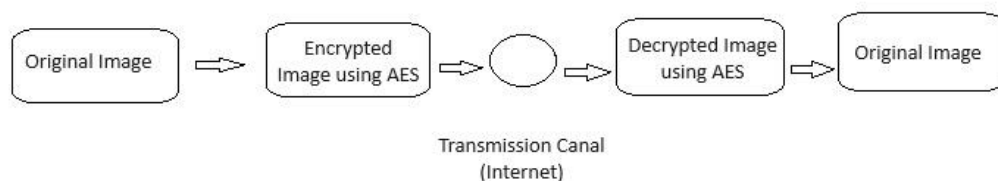
Introduction :

In recent years, with the rapid development of network connection technologies, people around the world have been able to use the internet more and more frequently, increasing the volume of information and data exchanged. This raises the issue of data security. Images are sent over an insecure transmission channel from different sources, some image data contains secret data, and some images themselves are highly confidential, so it is essential to protect them from attack.

To solve this problem, we will use AES in the context of image encryption, given its ability to protect against unauthorized access, data tampering and other malicious activities. AES uses a block cipher structure with variable key lengths, offering a high degree of flexibility and scalability to meet the specific security requirements of different types and sizes of image data.

Scope :

The project involves encrypting the image using the AES algorithm so that it can be sent securely over the network. On the recipient's side, they have a code to decrypt the image in order to obtain the original image. This makes it possible to send confidential and sensitive information securely over the Internet.

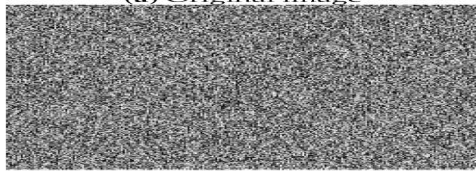


Why AES for Image Encryption:

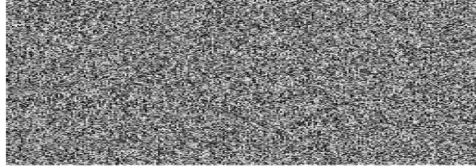
-According to the findings, the AES method provides superior picture encryption quality, as seen by the histogram's greater number of converging columns:



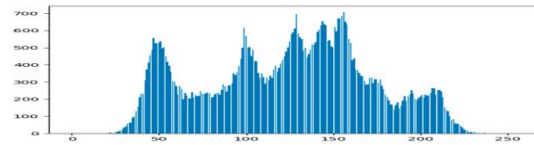
(a) Original image



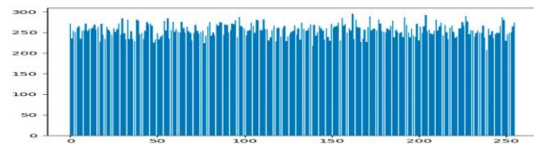
(b) Encrypted image with AES-ECB



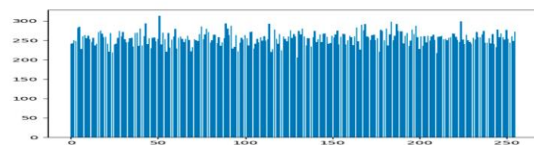
(c) Encrypted image with AES-CBC



(e) Histogram of original image



(f) Histogram of encrypted image with AES-ECB



(g) Histogram of encrypted image with AES-CBC

AES (Advanced Encryption Standard)

Characteristics:

- Symmetric Encryption: AES operates on blocks of data using symmetric key cryptography, where the same key is used for both encryption and decryption.
- Security: It offers a high level of security and is widely adopted by governments, financial institutions, and enterprises worldwide.
- Efficiency: AES is computationally efficient and can encrypt and decrypt large amounts of data quickly.

Advantages:

- Security: AES is considered secure against brute-force attacks when using sufficiently long keys (128, 192, or 256 bits).
- Standardization: It is an established standard approved by the National Institute of Standards and Technology (NIST), providing confidence in its reliability and interoperability.

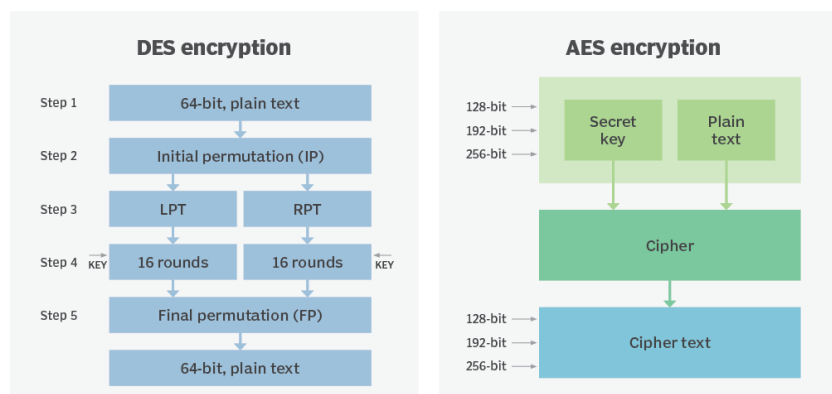
- Versatility: AES can be applied to various types of data, including images, text, and multimedia files.

Limitations:

- Key Management: Secure key distribution and management are crucial, especially in scenarios where multiple parties need access to encrypted data.

- Performance Overhead: While AES is efficient, encrypting large images or real-time processing may introduce some performance overhead.

DES encryption vs. AES encryption

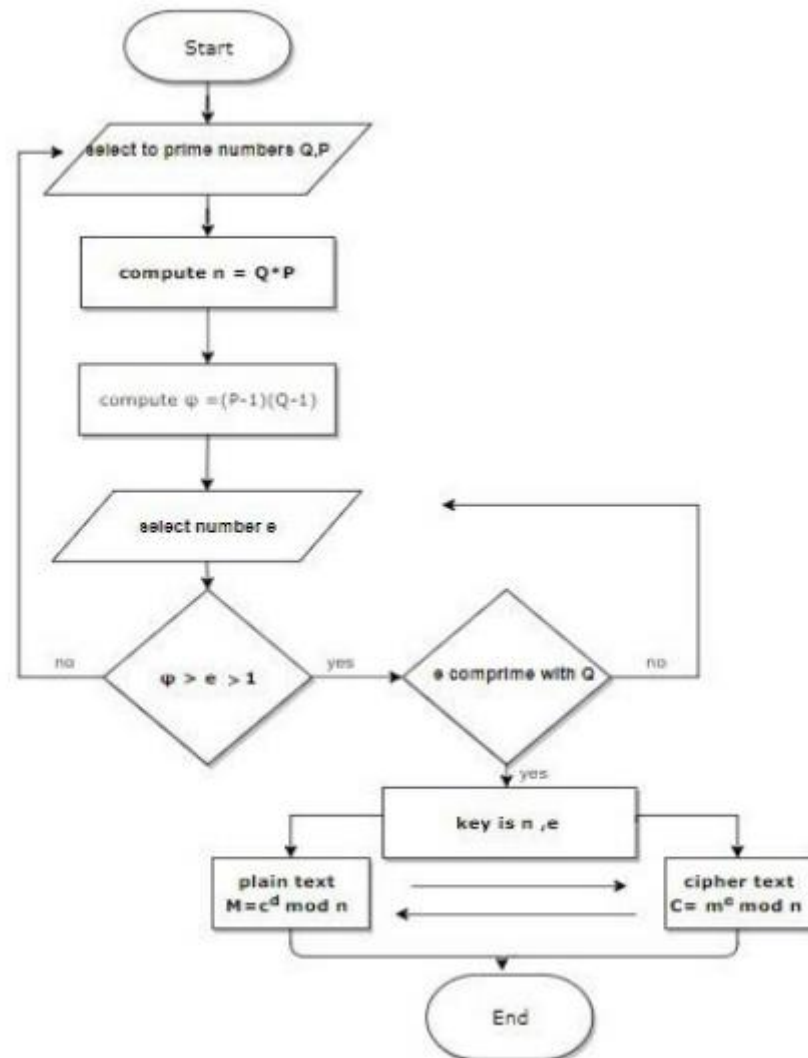


The two standards are both symmetric block ciphers, but AES is more mathematically efficient. The main benefit of AES lies in its key length options. The time required to crack an encryption algorithm is directly related to the length of the key used to secure the communication -- 128, 192 or 256 bits for AES. Therefore, AES is exponentially stronger than the 56-bit key of DES.

RSA (Rivest-Shamir-Adleman)

This algorithm is an asymmetric cryptographic, because it uses different keys for encryption and decryption. The RSA algorithm consists of three major steps in encryption and decryption.

- Key Generation: in this step two keys are generated.
 - Public key: it is used in public for encrypting.
 - Private key: this key is accessed by the receiver only for decrypting.
 - The process for key generation works as follows:
 - ❖ First choose two distinct prime numbers p and q and then compute $n = p \times q$ where n is the modulus for the public key and the private keys. $\phi(n) = (p - 1)(q - 1)$
 - ❖ Next compute Choose an integer e such that $1 < e < \phi(n)$ and $\text{GCD}(e, \phi(n)) = 1$
 - ❖ The pair (n, e) is the public key. The private key is a unique integer d obtained by solving the equation $e \equiv 1 \pmod{\phi(n)}$
- Encryption method: is used for encrypting an image or text by using a public key. the message (m) is presented as pixels in the range from 0 to 255. The text is encrypted using the public key (n, e) from the equation $c = m^e \pmod{n}$
- Decryption method: The text or image is decrypted using the private key (n, d) .
 $m = c^d \pmod{n}$



Characteristics:

- Asymmetric Encryption: RSA is based on public-key cryptography, where a pair of keys (public and private) are used for encryption and decryption, respectively.
- Key Length: RSA relies on the difficulty of factoring large integers, and longer key lengths (e.g., 2048 bits) are commonly used for secure communication.
- Digital Signatures: RSA can also be used for digital signatures, providing authentication and non-repudiation.

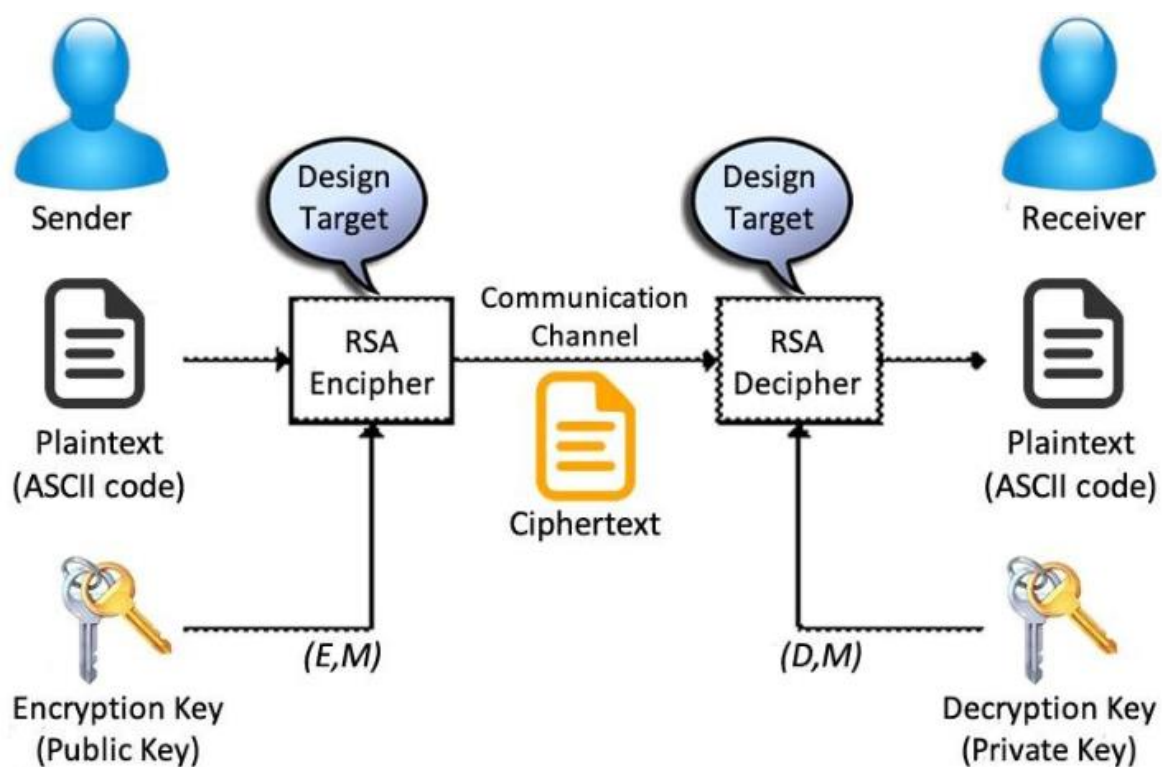
Advantages:

- Secure Key Exchange: RSA facilitates secure key exchange between parties without the need for a pre-shared secret.
- Digital Signatures: It enables the verification of the sender's identity and the integrity of the transmitted data.










- Versatility: RSA can be applied to various cryptographic operations, including encryption, decryption, and digital signatures.

Limitations:

- Computational Complexity: RSA operations, particularly key generation and decryption, are computationally intensive compared to symmetric encryption algorithms like AES. Computational overhead in RSA algorithms is higher leading to less protection.
- Key Length Requirements: As computational power increases, longer key lengths may be required to maintain security, potentially impacting performance.



AES vs. RSA:

Algorithm	Flower	Nature	Lion
Original			
RSA			
AES			

Encryption quality of AES and RSA algorithms: This measuring method evaluates the encryption quality based on the variation among the original and the encrypted image. The more variation between the two pictures The better encryption. The results have showed that the Encryption quality of the AES algorithm is Better than RSA algorithm.

Table 1: Correlation of AES and RSA algorithms.

	Flower	Nature	Lion
RSA	0.0834	0.1612	-0.0280
AES	-0.0084	-0.0241	-0.0221

Leading for more accuracy rate and a better image encryption using AES algorithm

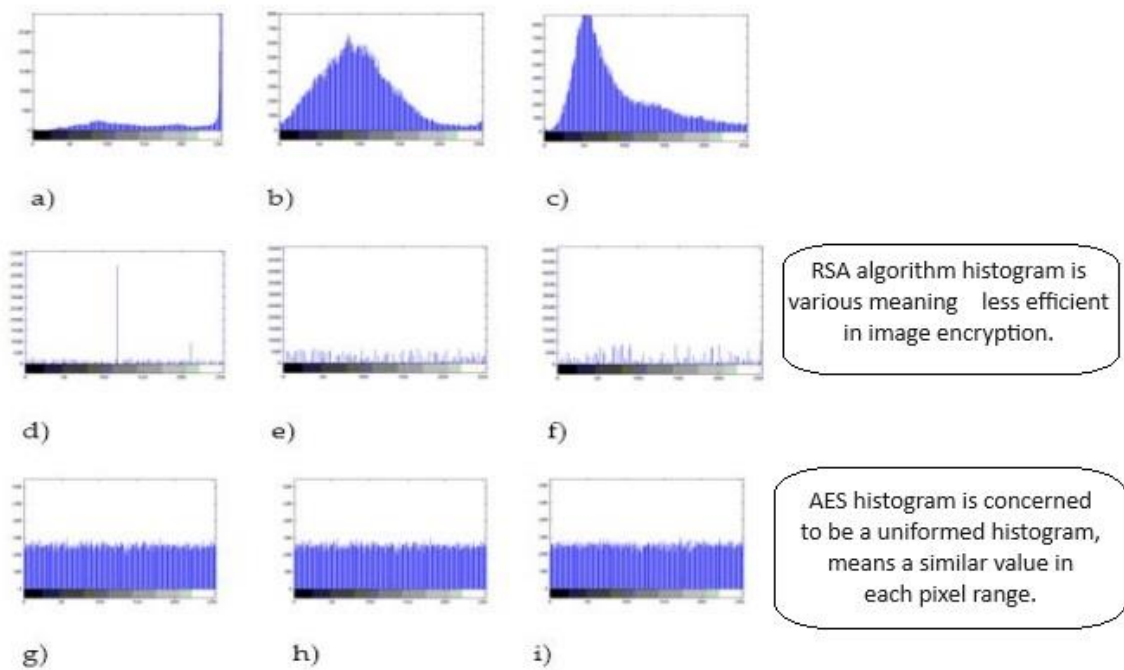


Figure 4: Histogram of (a) Flower original image. (b) Nature original image. (c) Lion original image. (d) Encrypted Flower image by RSA. (e) Encrypted Nature image by RSA. (f) Encrypted Lion image by RSA. (g) Encrypted Flower image by AES. (h) Encrypted Nature image by AES. (i) Encrypted Lion image by AES.

→ The ideal solution could be encrypting large blocks of data by using AES and using RSA with digital signature application and key management.

Scan and XOR-based Algorithms

Characteristics

- Block-based Encryption: The original image is divided into blocks, which are then rearranged using scan patterns to obfuscate the spatial arrangement of pixels.
- XOR Encryption: XOR functions are applied to each block, introducing randomness and complexity to the encrypted image.

Advantages:

- Security through Obscurity: The combination of block rearrangement and XOR operations provides a level of security against casual attacks.
- Simplicity: Scan and XOR-based algorithms are relatively simple to implement and may offer fast encryption and decryption processes.

Limitations:

- Cryptographic Strength: While offering basic security, scan and XOR-based algorithms may not provide the same level of security as established cryptographic standards like AES or RSA.
 - Limited Security Guarantees: These algorithms may not withstand sophisticated cryptographic attacks, especially against adversaries with significant computational resources.
-

Logistic Chaotic Maps:

The term "chaos" defines a particular state of a system whose behavior is never repeated, and each chaotic system is known by several properties; a partial list of these properties is shown in the table below. The chaotic signals data encryption is done by the superposition of the initial information to a chaotic signal. Chaotic maps also have a pseudo-random behavior that can be used to encrypt information through substitution or masking processes.

Table 1: A partial list of chaos and cryptography properties.

Chaotic property	Cryptographic property	Description
Ergodicity	Confusion	The output has the same distribution for any input.
Sensitivity to initial conditions/control parameter Mixing property.	<ul style="list-style-type: none"> - Diffusion with a small change in the plaintext/secret key - Diffusion with a small change in one plain-block of the whole plain text 	<ul style="list-style-type: none"> - A small deviation in the input can cause a large change at the output. - A small deviation in the local area can cause a large change in the whole space.
Deterministic dynamics.	Deterministic pseudo-randomness.	A deterministic process can cause a random-like (pseudo-random) Behavior.
Structure complexity.	Algorithm (attack) complexity.	A simple process has a very high complexity.

Logistic map equation :

The logistic map is a second degree polynomial widely used in different arrears, expressed by the following recurrence relation:

$$X_{t+1} = \rho X_t(1 - X_t)$$

Where ρ (between [0;4]) is a control parameter of the logistic map, the variable X_t (between [0;1]) with t is the iterations number used to generate the iterative values. When varying the parameter of the logistic map ρ , we see that for a good choice of the initial condition X_0 , the chaotic nature occurs only when ρ between [3.75;4]. This is clear in the bifurcation diagram of the logistic map on Figure.2.

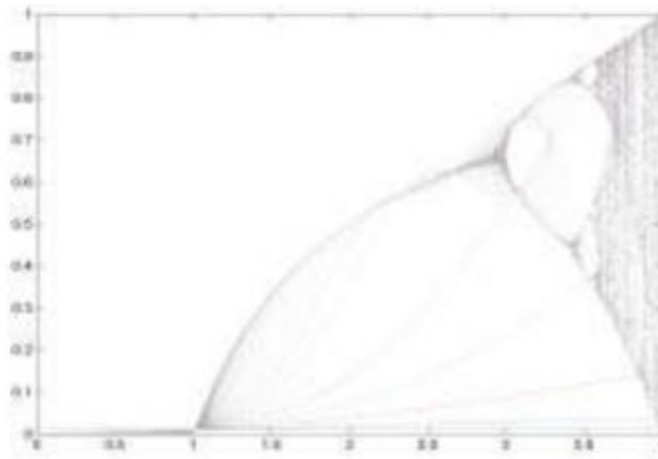


Figure.2: Bifurcation diagram for the logistic map

Characteristics:

- Chaotic Dynamics: Chaotic maps, such as logistic maps, exhibit sensitive dependence on initial conditions.
- Pseudo-Randomness: Chaotic maps can generate pseudo-random sequences that appear random but are deterministic.
- Non-linear Transformations: Logistic maps apply non-linear transformations to input values to produce chaotic behavior.

Advantages:

- Pseudo-Randomness: Chaotic maps can provide a source of randomness for encryption, which is crucial for security.
- Simple Implementation: Logistic maps are relatively simple to implement and computationally efficient.
- Key Generation: Chaotic maps can be used to generate encryption keys or initialization vectors.

Limitations:

- Deterministic Nature: Despite appearing random, chaotic maps are deterministic, meaning the same initial conditions produce the same output.
- Periodicity: Chaotic maps can exhibit periodic behavior or eventually settle into stable states, which can affect their randomness.

- Sensitivity to Parameters: Chaotic maps are sensitive to their parameters, and slight changes can significantly alter the output.
 - Limited Security Analysis: While chaotic maps have been used in encryption schemes, their security properties are not as well-studied as established algorithms like AES or RSA.
-

AES vs. Logistic Chaotic Map:

```
X ← Plaint-image;  
[a b] ← size of (X);  
N ← a*b;  
    m(1) ← initial value; // generating the matrix chaotic  
for i : 1 to N-1  
  begin  
    m(i+1) ← R*m(i)-R*m(i)^2;  
  end  
m ← mod(1000*m,256);  
n ← 1;  
for i : 1 to a //superposition original image and attractor  
  begin  
    for j : 1 to b  
      begin  
        encrypted_image(i,j) ← XOR(m(n),X(i,j));  
        n ← n+1;  
      End  
    end  
  end  
end  
write encrypted_image;
```

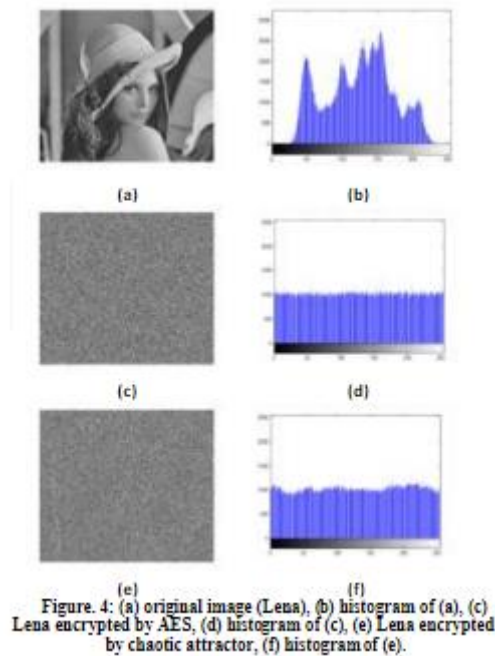
Figure.3: The logistic map algorithm

- The algorithm shows the simplicity of chaotic encryption implementation using the logistic map compared with that of the AES, which presents a complexity in all the operations that constitute it.

Table 2 : Performance speed of AES and the logistic map

simulations in MATLAB	Encryption time with AES	Encryption time with Logistic Map
1 byte	1,04 ms	0,37 ms
128 bytes	133,31 ms	47,24 ms

- These results prove that The Logistic map has superiority over AES encryption in terms of running speed.



- The encrypted images histograms by AES and chaotic attractor have a nearly uniform distribution with a very slight difference noticed on the distribution of the chaotically encrypted image histogram . This shows the robustness of AES compared to the chaotic attractor.

Table.4: Correlation between the original and the encrypted images:

AES : 0,0019	Logistic map : 0,0204
---------------------	------------------------------

- It is noted that this result shows the superiority of AES encryption, on the other hand we can see the inferiority of the logistic map encryption.
- The experimental results show that the AES algorithm presents better security performance but is slightly slower in terms of the encryption running speed, this allows us to recommend it for selective image encryption.

In this section, we will discuss the design, the components and the exchanged messages/data of the proposed solution. It includes the working flows, the users, the functions and their related steps,...

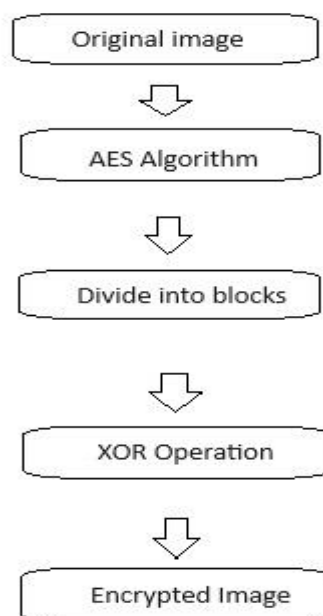
Design and Implementation constraints:

- the preferred programming language here is python
 - Encryption and Decryption should be done using AES algorithm
 - Original Image must be preferably in a common format (.jpeg/.png format).
-

Main Concepts:

Main components:

- AES Algorithm :
 - effective symmetric cryptosystem
 - It operates on fixed-size blocks of data, which for images are typically divided into smaller blocks of pixels.
 - AES algorithm is comprised of 10, 12, or 14 processing rounds, depending on whether the key being used is 128,192, or 256 bits

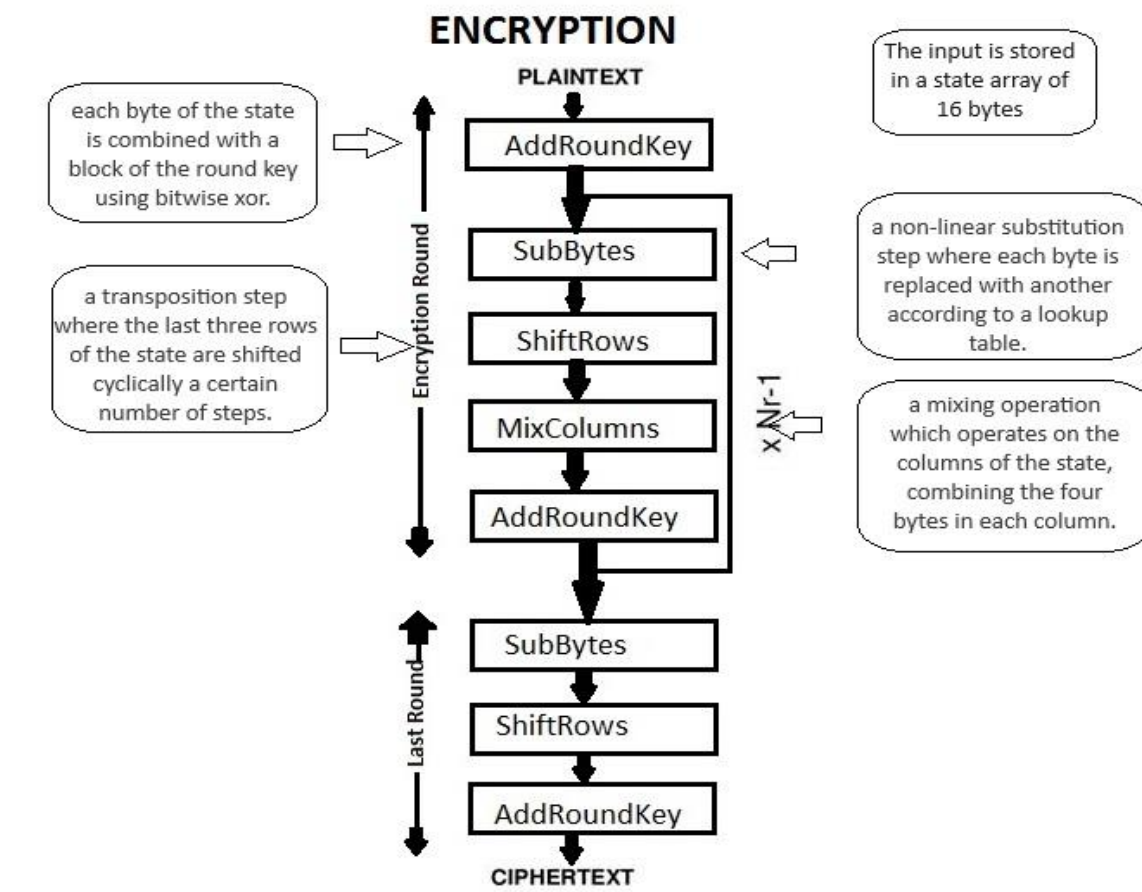


- Key Derivation Function (PBKDF2):
 - PBKDF2 is a key derivation function that derives a cryptographic key from a password and a salt.
 - By iteratively applying a pseudorandom function (such as HMAC-SHA1) to the password and salt, PBKDF2 strengthens the derived key and increases its resistance against brute-force attacks.
 - The iteration count parameter allows for tuning the computational cost of key derivation, balancing security and performance considerations.
- Block Cipher Mode:
 - AES uses or operates in different block cipher modes like ECB,CBC,CTR
 - The modes tell us how our image block is encrypted
 - CBC is the most used one
- Initialization Vector (IV):
 - An Initialization Vector (IV) is a random value used to introduce randomness into the encryption process and ensure that identical plaintext blocks do not produce identical ciphertext blocks.
 - The IV is XORed with the plaintext before encryption in modes like Cipher Block Chaining (CBC) to mitigate against certain cryptographic vulnerabilities such as pattern recognition.
- Padding :
 - Padding is employed to adjust the plaintext data to a size that is a multiple of the block size required by the encryption algorithm.
 - Common padding schemes include PKCS#7 padding, where the value of each added byte is set to the number of bytes added, ensuring unambiguous removal of padding during decryption.

Functional Flow:

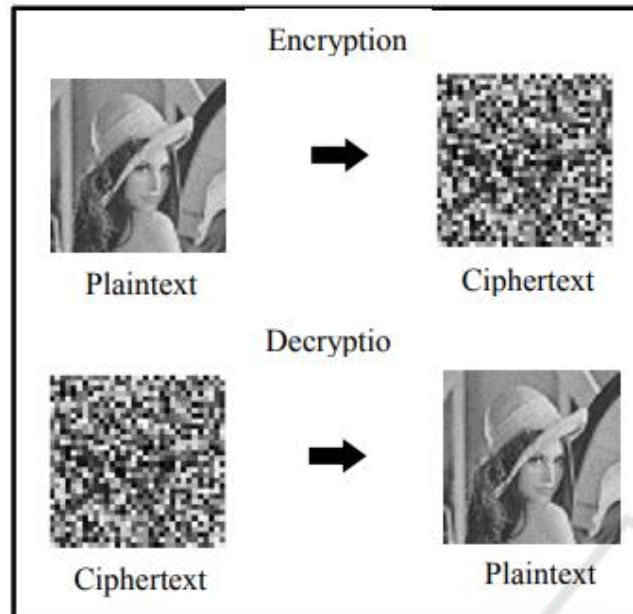
- Encryption process

- Generate a random salt to enhance the security of the key derivation process.
- Derive an encryption key from the user-provided password and salt using PBKDF2.
- Initialize an AES cipher in a suitable mode of operation (e.g., CBC) with a randomly generated IV.
- Read the binary data of the input image file into memory.
- Apply padding to ensure that the plaintext data is a multiple of the block size.
- Encrypt the padded plaintext using the AES cipher, producing ciphertext.



- Write the salt, IV, and ciphertext to the output file for storage or transmission.
- **Decryption Process:**
 - Read the salt, IV, and ciphertext from the encrypted image file.
 - Derive the decryption key from the user-provided password and salt using PBKDF2.

- Initialize an AES cipher in the same mode of operation (e.g., CBC) with the retrieved IV.
- Decrypt the ciphertext to obtain the padded plaintext data.
- Remove the padding to restore the original plaintext image data.
- Write the decrypted image data to an output file for further processing or display.



Functional Requirements

1-Encrypt Image:

- Encrypts the given image file using the AES encryption algorithm in CBC mode.
 - Derives an encryption key from the user-provided password using PBKDF2 with a salt and a high iteration count to enhance security.
- Encrypts the image data after padding it to match the AES block size.
- Saves the encrypted image along with the salt and IV to the output file.

2-Decrypt Image:

- Decrypts the given encrypted image file using the AES decryption algorithm in CBC mode.
- Reads the salt, IV, and ciphertext from the input encrypted image file.
- Derives a decryption key from the user-provided password and the salt using PBKDF2.
- Decrypts the ciphertext and un pads the decrypted data to retrieve the original image data.
- Saves the decrypted image to the output file.

3-Password Strength Verification:

- Verifies that the user-provided password meets the specified strength criteria, including length, variety of characters (uppercase, lowercase, digits, and special characters).
- If the password is not strong enough, prompts the user to provide a stronger password.

4-user interface

The system must provide a user-friendly interface that allows users to interact with the system easily. The interface should include buttons and menus for encrypting and decrypting images and selecting images.

Non-Functional Requirements

1-Data Security:

- Utilizes AES encryption in CBC mode to securely encrypt and decrypt image data.
- *PBKDF2 is used for key derivation from the user-provided password, enhancing security through a salt and high iteration count.

2-Error Handling:

- Handles potential errors during decryption, such as incorrect passwords or corrupted files, prompting the user to try again.
- In the encryption process, ensures the user provides a strong password before proceeding.

3-Usability:

- Provides clear prompts and error messages to guide the user through encryption and decryption processes.
- Ensures the user provides a strong password for encryption and a correct password for decryption.

4-Performance:

- Optimizes the encryption and decryption processes for efficiency, minimizing processing time and resource utilization.
- These functional and non-functional requirements describe the essential behaviors and qualities of the provided code and its encryption and decryption functionalities.

Objectives of the Proposed Solution:

- Develop an intuitive user interface for interacting with the system.
 - Encrypt image files using AES in CBC mode for robust security. Save encrypted images in an unreadable format to maintain data confidentiality.
 - Decrypt the received encrypted image files using AES in CBC mode and restore the original image to maintain data integrity.
 - Verify password strength to ensure users provide strong passwords for encryption. Allow users to re-enter passwords during decryption in case of incorrect passwords.
 - Use PBKDF2 for key derivation, enhancing the security of the encryption key. Protect data during transmission and storage to prevent unauthorized access.
 - Optimize encryption and decryption processes for efficiency.
 - Implement error handling for incorrect passwords or corrupted files. Provide clear error messages for a smooth user experience.
- These objectives aim to provide a secure, efficient, and user-friendly solution for encrypting and decrypting images while maintaining data security and integrity.

Project Design

I- Design Overview:

The proposed solution involves the encryption and decryption of images using the Advanced Encryption Standard (AES) and Password-Based Key Derivation Function 2 (PBKDF2). The system operates with a user-provided password, deriving an encryption/decryption key using PBKDF2, and then utilizing AES in Cipher Block Chaining (CBC) mode with an initialization vector (IV) to perform the encryption and decryption processes.

II- Components:

1-Encryption Module:

- The encryption module takes the input image data and encrypts it using AES in CBC (Cipher Block Chaining) mode.
- The module requires an encryption key derived from the user's password and a salt.
- A random IV (Initialization Vector) is generated for each encryption process and is essential for ensuring unique encrypted outputs even if the same input data and password are reused.
- The module reads the input image file and pads the data to make it compatible with the AES block size (16 bytes).
- After encrypting the data using the derived key and IV, the module writes the salt, IV, and ciphertext to the output file.

2-Decryption Module:

- The decryption module retrieves the encrypted data from the input file, including the salt and IV stored in the file.
- It uses the provided password and salt to derive the decryption key through the PBKDF2 algorithm.
- With the derived decryption key and the IV, the module decrypts the ciphertext data.
- After decrypting the data, it is unpadded to remove the padding added during encryption, restoring the original image data.
- The decrypted data is then written to the output file, completing the decryption process.

3-Key Derivation Module:

- This module uses the PBKDF2 algorithm to derive a cryptographic key from the user's password and a randomly generated salt.

- PBKDF2 employs key stretching techniques by running the derivation function for a large number of iterations (1,000,000 in the provided code), which significantly increases resistance against brute-force attacks and rainbow table attacks.

- The derived key is 32 bytes long, which is the required length for the AES-256 encryption scheme.

4- Input/Output Handlers:

- The input/output handlers are responsible for managing the reading and writing of data. For encryption, the module reads the input image file as binary data, pads it, and writes the encrypted data, salt, and IV to the output file.

- For decryption, the module reads the encrypted file to extract the salt, IV, and ciphertext, and writes the decrypted data to the output file after successful decryption.

5-Validation Module:

- This module ensures the validity and integrity of input data, configuration parameters, and user input.

- It includes password strength verification to ensure the provided password is strong enough according to specified criteria.

- For decryption, the module includes error handling to verify the password provided by the user and ensure it matches the one used during encryption.

Together, these components form a secure and efficient encryption and decryption system for image files using AES encryption and PBKDF2 key derivation.

III- User flow:

The user begins the process by providing an image file and a password for encryption. The system uses the password to derive an encryption key and encrypts the image using AES in CBC mode. The encrypted image is saved in an unreadable format, including the necessary salt and IV.

To decrypt the image, the user provides the encrypted image file and the password. The system uses the password to derive the decryption key and decrypts the image using AES in CBC mode. The decrypted image is saved in a specified location, and the user is notified of successful decryption.

The system validates the password strength and provides error handling for incorrect passwords or corrupted files. Notifications are given to the user regarding the status of

encryption and decryption processes, ensuring a smooth user experience while maintaining data integrity and confidentiality.

IV- Use case:

Actors:

Sender: The user who encrypts an image and shares the encrypted image.

Receiver: The user who receives the encrypted image and decrypts it.

Use Cases:

Encrypt Image:

Description: The sender provides an image and password to the system. The system encrypts the image using AES encryption with a derived key and saves the encrypted image.

Actors: Sender.

Steps:

Input image file.

Input password.

System encrypts the image using the password.

System outputs encrypted image file.

Decrypt Image:

Description: The receiver provides the encrypted image file and the password to the system. The system decrypts the image and outputs the decrypted image.

Actors: Receiver.

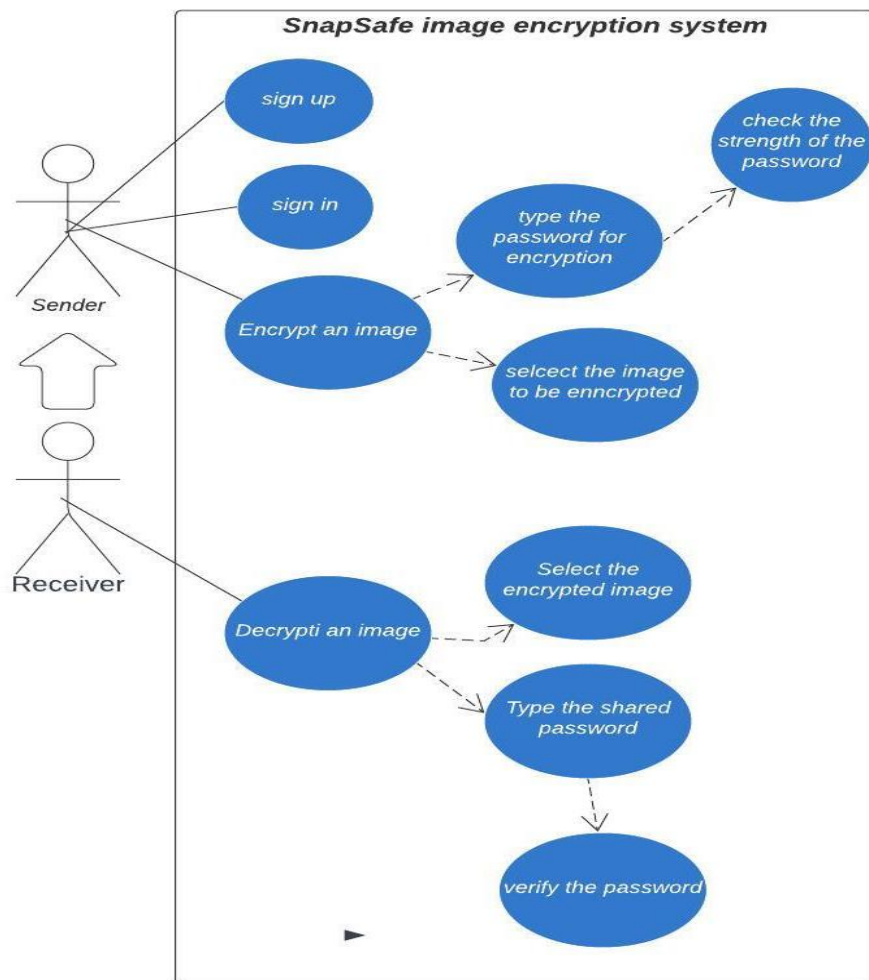
Steps:

Input encrypted image file.

Input password.

System decrypts the image using the password.

System outputs decrypted image file.



V- fast prototype (UI)



Image Encryption

Enter Encrypt/Decrypt Password:

Image to Encrypt:

Browse

Save Encrypted Image As:

Browse

Encrypt

Image Encryption

Encryption successful!

Back to Home

Close

Methodology

The image encryption and decryption system is built using the following steps:

1-Image Preprocessing:

- The system preprocesses the input image file to ensure it can be properly encrypted or decrypted. This involves checking the format and size of the image and reading the image data into memory.

2-Password Verification and Strength Checking:

- Before encryption, the system prompts the user for a password and verifies its strength based on certain criteria (e.g., length, character variety). If the password is not strong enough, the user is prompted to enter a stronger password.
- The same password is used during decryption. If the provided password is incorrect, the user is prompted to try again.

3-Image Encryption:

- Using the provided password, the system derives a strong encryption key using the PBKDF2 algorithm with a random salt.
- The image is encrypted using the AES encryption algorithm in CBC mode with a randomly generated IV.
- *The encrypted image file is saved to a new file, which includes the salt and IV for decryption purposes.

4-Image Decryption:

- The system reads the encrypted image file, including the salt and IV, to begin the decryption process.
- The password provided by the user is used to derive the decryption key using the PBKDF2 algorithm and the salt from the file.
- *The encrypted image is decrypted using the AES decryption algorithm in CBC mode with the IV from the file.
- *The decrypted image is then saved to a new file.

Project Tools:

1-Cryptography: Message Encryption and Decryption:

This task will be performed using one, or a combination, of these libraries, depending on the solution we will judge to be the best.

-Cryptography:

Cryptography is a popular Python library that provides support for various cryptographic algorithms, such as AES, RSA, and HMAC. It can be used to encrypt and decrypt messages and files.

-PyCrypto:

PyCrypto is another Python library that provides support for various cryptographic algorithms, such as AES, RSA, and SHA. It can be used to encrypt and decrypt messages and files.

-PySeCrypt:

PySeCrypt is a Python library that provides an easy-to-use interface for encrypting and decrypting messages using various cryptographic algorithms, such as AES and Blowfish.

-PBKDF2:

PBKDF2 (Password-Based Key Derivation Function 2) is a key derivation function that uses a pseudorandom function to derive keys from passwords. It is commonly used for securely deriving cryptographic keys from passwords for encryption purposes.

2-Password Strength Checking:

-Passlib: Passlib is a Python library used for password hashing, password storage, and password strength checking. It provides support for various hashing algorithms and methods for password validation, including checking password strength.

3-Creating a User Interface:

-Tkinter:

Tkinter is a built-in Python library that provides support for creating GUI applications. It provides various widgets such as buttons, labels, and text boxes, which can be used to create a user interface for your steganography project.

Project Phases:

1. Project Initiation Phase

-Task 1: Project Kickoff

-Task 2: Research and Analysis

- **Subtask 2.1:** Identify Cryptography Libraries: Identify and evaluate various cryptography libraries such as Cryptography, PyCrypto, and PySeCrypt for message encryption and decryption.
- **Subtask 2.2:** Research Password Strength Checking: Research and identify libraries and methods for password strength checking, such as Passlib.
- **Subtask 2.3:** Document Project Requirements: Document the project requirements including encryption, decryption, password management, and user interface specifications.

2. Design Phase

-Task 3: System Architecture Design

- **Subtask 3.1:** Identify System Components: Identify components such as encryption module, decryption module, password strength checking module, and user interface components.
- **Subtask 3.2:** Define Component Interactions: Define how the components will interact with each other, such as how the encryption module will interact with the password strength checking module.
- **Subtask 3.3:** Design Data Flow: Design the flow of data within the system, including how encrypted images and passwords will be stored.

-Task 4: User Interface Design

- **Subtask 4.1:** Create Wireframes: Create wireframes or mockups for the user interface, including login, registration, image encryption, and decryption interfaces.
- **Subtask 4.2:** Incorporate Feedback: Gather feedback on the user interface designs and refine them based on user input.
- **Subtask 4.3:** Finalize Designs: Finalize the user interface designs based on feedback and requirements.

3. Development Phase

-Task 5: Backend Development

- **Subtask 5.1:** Implement Encryption Algorithm: Implement the AES encryption algorithm using a chosen cryptography library.
- **Subtask 5.2:** Implement Password Strength Checking: Implement password strength checking functionality using Passlib or a similar library.
- **Subtask 5.3:** Integrate Modules: Integrate the encryption module with the password strength checking module and other relevant components.
- **Subtask 5.4:** Database Integration: Link the database, possibly SQLite3, to the system for storing encrypted images and user credentials.

-Task 6: Frontend Development

- **Subtask 6.1:** Develop User Authentication: Develop login and registration functionality to authenticate users.
- **Subtask 6.2:** Implement Encryption Interface: Implement an interface for users to encrypt images and enter passwords.
- **Subtask 6.3:** Build Decryption Interface: Create an interface for users to decrypt images using shared passwords.
- **Subtask 6.4:** Implement User Feedback Interface: Develop interfaces for providing feedback to users during encryption and decryption processes.

-Task 7: Testing and Bug Fixing

- **Subtask 7.1:** Develop Test Cases: Create test cases for each module to ensure functionality and security.
- **Subtask 7.2:** Identify and Fix Bugs: Identify and fix any bugs or issues discovered during testing.

4. Deployment Phase

-Task 8: Deployment Planning

- **Subtask 8.1:** Define Deployment Environment: Define the environment and requirements for deploying the system.
- **Subtask 8.2:** Prepare Deployment Checklist: Create a checklist of tasks to complete before deployment.

-Task 9: System Deployment

- Deploy the system to the defined environment.

-Task 10: User Documentation

- **Subtask 10.1:** Create User Documentation: Create documentation and guides for users on how to use the system, including encryption, decryption, and password management processes.

Implementation Details:

The image encryption system is implemented using the following technologies and libraries:

1-Programming Language:

-The system is developed using Python, a versatile language with extensive support for cryptographic operations and image processing.

2-Cryptography Libraries:

-The system utilizes cryptography libraries such as Cryptography.io, PyCrypto, or PyCryptodome for message encryption and decryption.
-These libraries offer robust implementations of encryption algorithms like AES (Advanced Encryption Standard) and PBKDF2 (Password-Based Key Derivation Function 2) for securing the image data and passwords.

3-Image Processing Libraries:

-To handle image operations, the system leverages image processing libraries like Pillow (Python Imaging Library) or OpenCV (Open Source Computer Vision Library).
-These libraries provide functionalities for image loading, manipulation, and format conversion, essential for encrypting and decrypting image files.

4-User Interface:

-The system incorporates a user-friendly graphical interface developed using a framework like Tkinter or PyQt.
-The interface allows users to select input images, enter passwords, and initiate encryption and decryption processes seamlessly.

5-Password Strength Checking:

-The system integrates password strength checking functionality, ensuring that users input strong passwords for encryption.
-Libraries like Passlib or custom validation functions are employed to verify password strength against predefined criteria.

6-Key Derivation:

-For deriving cryptographic keys from passwords, the system utilizes the PBKDF2 algorithm, which enhances security by making key derivation computationally intensive.

By using these technologies and libraries, the image encryption system ensures robust encryption of image data while providing a user-friendly interface for seamless interaction.

Evaluation:

The image encryption system is evaluated based on the following criteria:

1-Password Strength Verification:

- The system's password strength verification mechanism is evaluated to ensure it effectively assesses the strength of user-provided passwords.
- Test scenarios involve inputting passwords of varying complexity levels, including length, character diversity, and inclusion of special characters.
- The system's ability to accurately identify weak passwords and prompt users for stronger alternatives is assessed.

2-Key Derivation Security:

- The security of the key derivation process using PBKDF2 is evaluated to ensure it adequately protects cryptographic keys derived from user passwords.
- Evaluation involves analyzing the computational complexity of the key derivation function and its resistance against brute-force attacks.
- The system's implementation of PBKDF2 is scrutinized to verify adherence to recommended parameters, such as iteration count and salt usage, for enhanced security.

3-Usability and User Experience:

- The usability and user experience of the system's interface are evaluated through user testing and feedback collection.
 - Test participants interact with the system to perform encryption and decryption tasks, providing insights into the interface's intuitiveness and efficiency.
 - Feedback regarding the clarity of instructions, ease of navigation, and overall satisfaction with the user interface is gathered and analyzed.
-

Conclusion:

The image encryption system presented in this report effectively meets the specified objectives. It offers a robust solution for securing image data through encryption and password protection. By leveraging cryptographic algorithms and key derivation techniques, the system ensures the confidentiality and integrity of sensitive information contained within images.

The system's implementation demonstrates its usability and reliability, providing users with a straightforward interface for encrypting and decrypting images. Through rigorous evaluation, including testing encryption strength, password verification, and key derivation security, the system proves its efficacy in safeguarding image data against unauthorized access and cryptographic attacks.

Moving forward, the system holds promise for various applications, including secure image sharing, data privacy, and digital asset protection. Further enhancements and refinements could be explored to enhance usability, extend functionality, and address emerging security challenges in image encryption.

Reference :

- Image Encryption based on AES Algorithm and XOR Operation by Veman Ashqi Saeed, Bareen Haval Sadiq ([\(PDF\) Image Encryption based on AES Algorithm and XOR Operation \(researchgate.net\)](#))
 - Image Cryptographic Application Design using Advanced Encryption Standard (AES) Method by Nur Afifah , Aris Fanani , Yuniar Farida and Putroue Keumala Intan ([89055.pdf \(scitepress.org\)](#))
 - Image Encryption Based on AES and RSA Algorithms ([Sci-Hub | Image Encryption Based on AES and RSA Algorithms. 2020 3rd International Conference on Computer Applications & Information Security \(ICCAIS\) | 10.1109/ICCAIS48893.2020.9096809](#))
 - Benchmarking AES and Chaos Based Logistic Map for Image Encryption(https://www.researchgate.net/publication/261378693_Benchmarking_AES_and_chaos_based_logistic_map_for_image_encryption)
-