TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

IT 360
INFORMATION ASSURA NCE & SECURITY

Security Project
Image Encryption

Authors:
Majd Hamdi
Seif Ben Soltana

Submitted to:
Prof. Manel Abdelkader

Components

The image encryption project we worked on is designed as a Google Colab notebook, and the described components are applicable to a Colab environment.

Image Encryption Component:

Image Selection: The sender chooses an image file (e.g., JPEG, PNG) that they want to encrypt.
Encryption Algorithm Selection: The sender selects a ready-to-use Python script corresponding to the desired encryption algorithm (e.g., AES, chaotic map).
Script Parameters: The sender adjusts parameters within the selected script as needed for the chosen encryption method (e.g., key size, encryption mode).
Encryption Execution: The Python script encrypts the image using the specified parameters, generating the encrypted image file.

Image Decryption Component:

Receiving the Encrypted Image: The receiver obtains the encrypted image file sent by the sender.
Decryption Algorithm Selection: The receiver selects the corresponding decryption Python script that matches the encryption algorithm used by the sender.
Decryption Parameters: If required by the algorithm, the receiver adjusts decryption parameters within the script (e.g., decryption key).
Decryption Execution: The Python script decrypts the encrypted image file using the specified parameters, recovering the original image.

Notes:

- *Script Consistency:* It's crucial that both the sender and receiver use identical versions of the Python scripts with matched parameters to ensure successful encryption and decryption.
- *Algorithm Compatibility:* Scripts should be chosen based on their compatibility with the image file format and security requirements. Ensuring that the chosen algorithm aligns with the image format and security needs is essential for a robust encryption process.

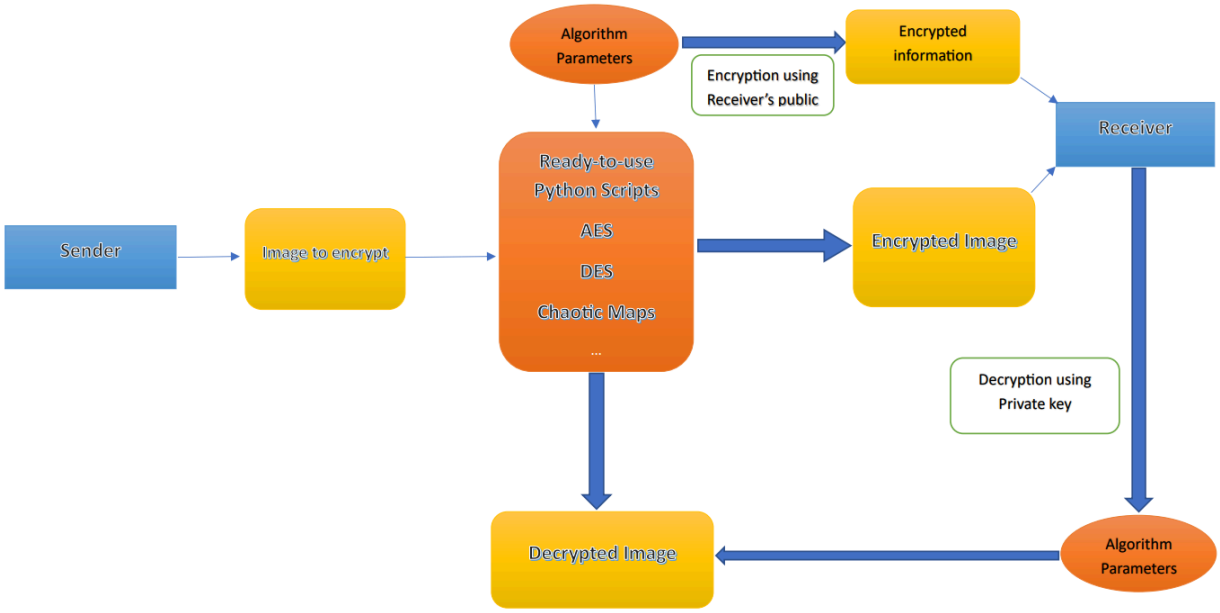
Sending Process:

- One can use any preferred channel such as messenger, email, or other communication tools to share the encrypted image and relevant decryption details with the recipient.

Working Flow

User1 (sender) selects an image to be encrypted and chooses a specific Python script tailored for the desired encryption algorithm. The script encrypts the image according to specified parameters, producing an encrypted image file. This encrypted image file is then transmitted to User2 (receiver) coupled with the encrypted parameters necessary for the decryption.

Upon receiving the encrypted image file and parameters, User2 accesses a matching decryption Python script corresponding to the encryption algorithm used by User1. By configuring the decryption script with necessary parameters, User2 decrypts the image, thereby restoring it to its original form for viewing.



Working flow diagram

The Roles or Users of your Solution

Sender (User1): Selects an image for encryption and configures the encryption parameters within a chosen Python script. Manages the transmission of the encrypted image.

Recipient (User2): Receives the encrypted image and utilizes a corresponding decryption Python script to decrypt the image using specified parameters, thereby accessing the original content of the image.

Exchanged Messages/Data

Sender to Receiver:
Encrypted image: Sent from User1 to User2.
Metadata: Optionally, metadata specifying encryption parameters and script details. To ensure confidentiality and security of the encrypted image, it is preferred to encrypt the metadata with the recipient's public key.

Receiver to Sender:
Decrypted metadata: The metadata is decrypted using the private key.
Decrypted image: Result of successful decryption by User2.
Acknowledgment: Optional message confirming successful decryption.

Use Case

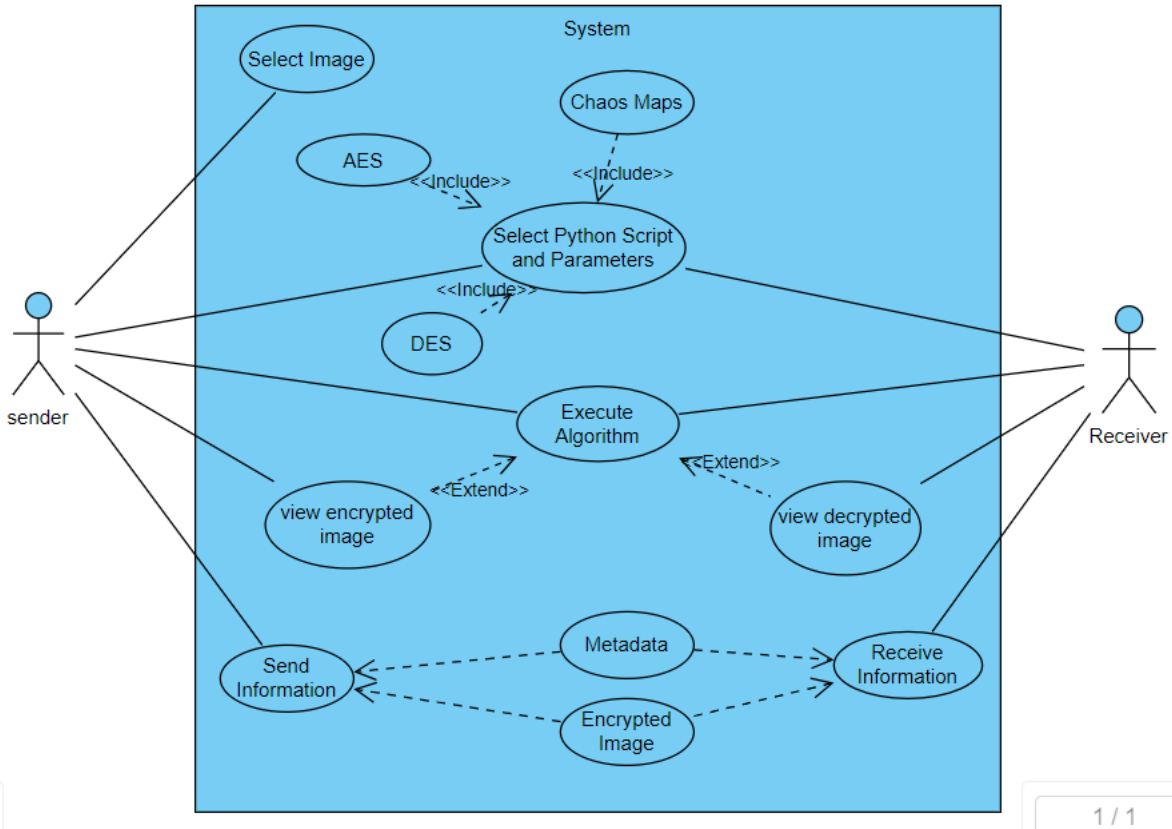
Sender:

- Select an image file to be encrypted.
- Choose an encryption algorithm (Python script) and configure its parameters.
- Execute the encryption process to generate an encrypted image file.
- Send the encrypted image file to User2 securely.

Recipient:

- Receive the encrypted image file from User1.
- Select the corresponding decryption algorithm (Python script) matching the encryption algorithm used by User1.

- Configure decryption parameters within the decryption script, such as the decryption key.
- Execute the decryption process to obtain the original image file.
- View the decrypted image to access the original content.



Use case diagram