# README - Diffie-Hellman Key Exchange with Steganography

#### **Overview**

This project demonstrates a secure communication protocol combining cryptography and steganography. It uses the **Diffie-Hellman (DH)** key exchange to generate a shared secret, and then uses that secret to encrypt messages using **AES encryption**.

All sensitive data — including DH parameters and encrypted messages — is embedded within image files using **Least Significant Bit (LSB)** steganography.

The system supports both:

- Standard LSB embedding
- Local variance-based LSB embedding (improved security through adaptive embedding)

#### **How It Works**

#### **Step 1: Generating and Embedding DH Parameters (Sender – Alice)**

- 1. **Alice** generates Diffie-Hellman parameters:
  - o Prime modulus p
  - Generator g
  - Private key a (random number)
  - Public key A = g^a mod p
- 2. Alice chooses a steganographic embedding method:
  - o Standard LSB
  - Local Variance-Based LSB
- 3. The values p, g, and A are embedded into an image using the selected LSB method.
- 4. The method used is saved alongside the image (.method.txt).
- 5. The image is sent to **Bob**.

#### Step 2: Extracting DH Parameters & Responding with Public Key B (Receiver – Bob)

- 1. **Bob** receives the image and extracts the embedded values p, g, and A, using the same LSB method.
- 2. Bob generates his own private key b and computes his public key: B = g^b mod p
- 3. He calculates the **shared secret**: S = A^b mod p
- 4. Bob embeds his public key B into a new image using standard LSB embedding.
- 5. The new image containing B is sent back to **Alice**.
- 6. Bob temporarily saves the shared secret S in a file (shared\_secret.txt) for message decryption later.

#### Step 3: Computing the Shared Secret & Encrypting the Message (Sender – Alice)

- 1. Alice receives the image containing 'B' and extracts the value.
- 2. She calculates the shared secret: `S = B^a mod p`
- 3. Using the shared secret `S`, Alice encrypts the secret message with AES.
- 4. The encrypted message is embedded into a new image using the selected LSB method.
- 5. The resulting image is sent to Bob.

#### **Step 4: Extracting and Decrypting the Message (Receiver – Bob)**

- 1. Bob loads the previously stored shared secret `S`.
- 2. He reads the LSB method used (from `.method.txt`).
- 3. He extracts the encrypted message from the received image.
- 4. Finally, he decrypts the ciphertext using AES and the shared secret `S`.

## **Usage**

The project is executed via the main.py file, which presents an **interactive menu with** options 1–4:

python main.py

### **Required Inputs:**

The system requires **three image files** ( in .png ) as containers for steganographic embedding:

- 1. Image for embedding initial DH parameters (p, g, and A Alice's public key).
- 2. Image for embedding Bob's public key (B).
- 3. **Image for embedding the encrypted message** (encrypted using AES with a shared DH-based key).

#### Note:

You are *not* required to use three different images.

The system creates a copy of each image before embedding, so you can reuse the same image file across different steps.

However, for clarity and file organization, it is **recommended** to use different images or provide descriptive filenames for each output.

#### **Important:**

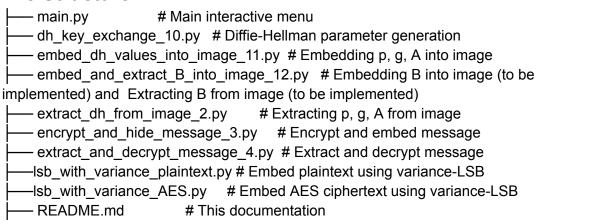
You must follow the steps in order:

Step 1 
$$\rightarrow$$
 Step 2  $\rightarrow$  Step 3  $\rightarrow$  Step 4

Each step depends on the output from the previous one:

- Skipping or reordering steps will result in incorrect behavior or decryption failures.
- Shared secrets and public keys are passed between steps via steganographic images and intermediate files.

## **File Structure**



## **Dependencies**

This project requires the following libraries and tools:

- **Python 3.x** The runtime environment for executing the code.
- Pillow Used for image processing and handling.
  Install via: pip install Pillow
- **NumPy** For numerical computations and array manipulation.
- **SciPy** Required for implementing variance-based LSB steganography methods. Install via: pip install scipy
- PyCryptodome A modern cryptographic library used for AES encryption of messages with the shared secret key.

Install via: pip install pycryptodome

# **Security Notes**

- Diffie-Hellman ensures secure exchange of a symmetric key without exposing private values.
- AES encryption adds another layer of security, making the message unreadable even if extracted.
- Steganography conceals the fact that any message or key exchange is taking place at all.

## **Notes**

- You can choose which **LSB method** to use for embedding/extracting p, g, A, and the encrypted message
- Bob's B key is currently handled using standard LSB
- The shared key S is derived separately by both parties and should match
- Run the steps in **sequential order** to ensure proper operation
- This project demonstrates secure communication that hides both the message and the fact that any secret is being exchanged