|  |  |
| --- | --- |
| **Name** | **Student Number** |
| Mahomed Aadil Ally | ALLMAH002 |
| Tapera Chikumbu | CHKTAP011 |

**CSC4026Z: NIS - Assignment**

**Cryptosystem Design**

The cryptosystem is designed to ensure secure communication between two parties, Alice and Bob, by leveraging various cryptographic techniques. The primary components of the system are:

* Asymmetric Key Encryption: RSA is used for asymmetric key encryption and decryption, employing the "RSA/ECB/PKCS1Padding" algorithm. Each party (Alice and Bob) generates their own RSA key pair (public and private keys).
* Symmetric Key Encryption: AES is used for symmetric key encryption and decryption, utilizing the "AES/CBC/PKCS5Padding" algorithm. The shared secret key for AES is derived from the Diffie-Hellman key exchange.
* Key Exchange: Diffie-Hellman key exchange is implemented to securely generate and exchange a shared secret key between Alice and Bob.
* Key Derivation: HKDF (HMAC-based Extract-and-Expand Key Derivation Function) is used to derive the AES key from the shared secret key obtained from the Diffie-Hellman key exchange. The HKDF uses SHA-256 as the hash function and the optional salt and info parameters are set to None and b'handshake data', respectively.
* Authenticated Encryption: Fernet (from the cryptography library) is used for authenticated symmetric encryption and decryption of messages and image data. Fernet combines AES-CBC encryption with HMAC-based authentication, ensuring both confidentiality and integrity of the encrypted data.

**Communication Connectivity Model**

The communication model follows a client-server architecture, where Alice acts as the client, and Bob acts as the server. The steps involved in the communication process are:

1. Bob starts the server and listens for incoming connections.
2. Alice connects to Bob's server.
3. Alice and Bob exchange their public keys.
4. Alice and Bob perform the Diffie-Hellman key exchange to generate a shared secret key.
5. Alice and Bob derive the AES key from the shared secret key using HKDF.
6. Alice encrypts the message (caption and image data) using the derived AES key and Fernet, and sends the encrypted message to Bob.
7. Bob receives the encrypted message, decrypts it using the derived AES key and Fernet, and saves the image and displays the caption.
8. Alice and Bob exchange and decrypt messages and image data in both directions using the same derived AES key and Fernet.
9. Alice and Bob close the connection.

**Key Management**

Key management is a crucial aspect of the cryptosystem, and the following measures are implemented:

* RSA Key Generation: Alice and Bob each generate their own RSA key pairs (public and private keys) using the cryptography library.
* Diffie-Hellman Key Exchange: Alice and Bob perform the Diffie-Hellman key exchange to securely generate and exchange a shared secret key.
* Key Derivation: The shared secret key from the Diffie-Hellman key exchange is used to derive the AES key for symmetric encryption and decryption using HKDF.
* Key Distribution: Alice and Bob exchange their public keys during the initial communication phase, allowing them to establish a secure communication channel.

**Choice of Cryptographic Algorithms**

The following cryptographic algorithms and techniques were chosen for this implementation:

* RSA: RSA is a widely-used and secure algorithm for asymmetric key encryption and decryption. It is employed for exchanging public keys.
* AES: AES is a highly secure and efficient algorithm for symmetric key encryption and decryption. It is used for encrypting the message and image data using the derived AES key and Fernet.
* SHA-256: SHA-256 is a secure hashing algorithm used by HKDF for deriving the AES key from the shared secret key.
* Diffie-Hellman: The Diffie-Hellman key exchange algorithm is used to securely generate and exchange a shared secret key between Alice and Bob, which is then used to derive the AES key for symmetric encryption and decryption.
* HKDF: HKDF (HMAC-based Extract-and-Expand Key Derivation Function) is used to derive the AES key from the shared secret key obtained from the Diffie-Hellman key exchange.
* Fernet: Fernet is a high-level encryption utility from the cryptography library that combines AES-CBC encryption with HMAC-based authentication, ensuring both confidentiality and integrity of the encrypted data.

These algorithms and techniques were chosen based on their widespread acceptance, security strength, and compatibility with the cryptography library used in the implementation.

**Testing Procedures and Assumptions**

The testing procedures for the implementation involve the following steps:

1. Run the Bob.py script to start the server.
2. Run the Alice.py script to initiate the client connection and secure communication.
3. Verify the console output on both Alice's and Bob's sides, ensuring that the encrypted messages and image data are exchanged successfully.
4. Check that the received image file (received\_image.png) is saved correctly and matches the original image.
5. Verify that the displayed caption matches the original caption.

The following assumptions were made during the development and testing of the implementation:

* The cryptography library is installed and available for use.
* The code is executed in a secure environment, and the private keys are kept confidential.
* The image file (image\_to\_send.png) is present in the same directory as the scripts.

**Execution Instructions**

To execute the submitted program, follow these steps:

1. Ensure that the cryptography library is installed. If not, install it using `pip install cryptography`.
2. Place the image\_to\_send.png file in the same directory as the Alice.py and Bob.py scripts.
3. Run the Bob.py script first to start the server: `python Bob.py`
4. In a separate terminal or console window, run the Alice.py script to initiate the client connection and secure communication: `python Alice.py`
5. Observe the console output on both Alice's and Bob's sides to verify the encryption, decryption, and message exchange processes.
6. Check for the received\_image.png file in the same directory, which should contain the received image.
7. The caption of the received image will be displayed in the console output.