Lab 01: Hybrid Encryption

21120263 - Tong Nguyen Minh Khang

University of Science - HCMUS

CSC15106 – Knowledge Engineering Seminar

1 Introduction

This report describes a simple hybrid encryption system for secure file exchange.

2 Architecture

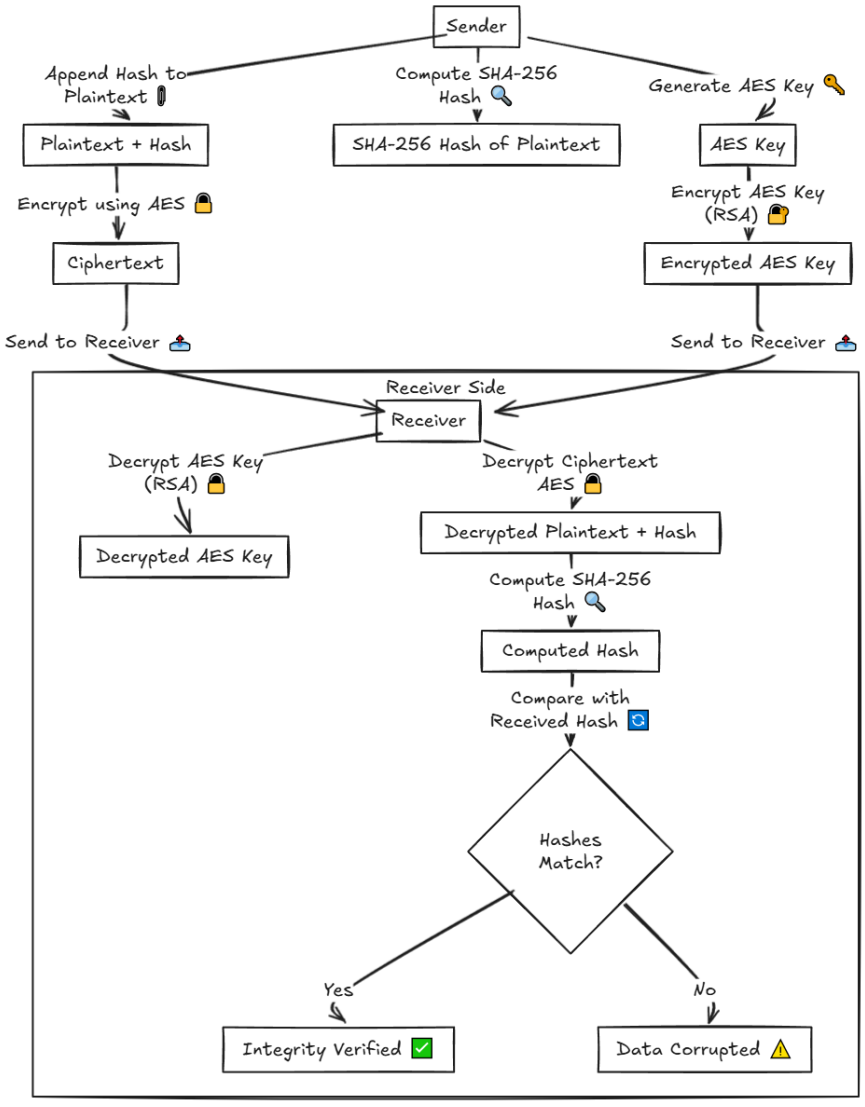

Figure 1: Flow during encryption and decryption.

Sender Side

1. **Compute SHA-256 Hash**: The sender computes the SHA-256 hash of the plaintext message.
2. **Append Hash to Plaintext**: The computed hash is appended to the plaintext, forming a combined message: *Plaintext + Hash*.
3. **Generate AES Key**: An AES key is generated for encrypting the plaintext.
4. **Encrypt Plaintext and Hash**: The combined *Plaintext + Hash* is encrypted using the AES key, resulting in the ciphertext.
5. **Encrypt AES Key**: The AES key is encrypted using RSA encryption to ensure secure key exchange.
6. **Send to Receiver**: The sender transmits the encrypted AES key and the ciphertext to the receiver.

Receiver Side

1. **Receive Data**: The receiver obtains the encrypted AES key and the ciphertext.
2. **Decrypt AES Key**: The receiver uses RSA decryption to recover the AES key.
3. **Decrypt Ciphertext**: The ciphertext is decrypted using the recovered AES key, resulting in the original *Plaintext + Hash*.
4. **Compute SHA-256 Hash**: The receiver extracts the plaintext and computes its SHA-256 hash.
5. **Compare Hashes**: The receiver compares the computed hash with the hash received along with the plaintext.
6. **Verify Integrity**: If the computed hash matches the received hash, data integrity is verified, and the process concludes successfully. If the hashes do not match, the data is identified as corrupted.

Final Result

- If the hashes match, the process confirms that the data has not been tampered with, and the integrity is verified.
- If the hashes do not match, it indicates data corruption during transmission.

3 System Overview

The system consists of the following main components:

- **RSA Encryption:** Used to encrypt the symmetric AES key.
- **AES Encryption:** Used for the encryption of the actual file data.
- **Hashing using SHA-256:** Applied to ensure data integrity by checking the hash of the encrypted file against that of the decrypted file.

4 Special Parameters

The following parameters are used in the program:

- **RSA Key Size:** A standard of 2048 bits is used for generating RSA key pairs.
- **AES Key Size:** An AES block size of 16 bytes (128 bits) is utilized for encrypting the data.
- **Hash Algorithm:** The SHA-256 hash function is employed to verify the integrity of the encrypted data.
- **Supported File Types:** The system is designed to handle various binary files for encryption and decryption, with no explicit limitations on file types within the binary format, as long as you have permissions to read the input file (key and unencrypted file), and writing the output files (encrypted file, decrypted file, encrypted key, public and private keys) to the directory you specified.

5 References

https://pycryptodome.readthedocs.io/en/latest/index.html
https://docs.python.org/3/library/argparse.html