**SECURE FILE TRANSFER SYSTEM**

The aim of this project is to implement a utility to transfer files securely while maintaining their confidentiality and integrity. This utility is essentially required to transfer large files between hosts securely.

A strong cryptographic mechanism is required to encrypt data. Two types of keys are used for encryption known as symmetric and asymmetric key encryption, but both have their draw backs and advantages. Public key encryption is used to prevent key distribution problems, but it has a performance bottleneck when encrypting and decrypting large files. A symmetric key is far more efficient in this case but it’s difficult to maintain the secrecy of the key.

To solve these problems, I’ve used three hybrid encryption techniques:

* Symmetric AES encryption which is used to encrypt the data.
* Asymmetric RSA encryption to encrypt the AES encryption key used to encrypt the data.
* HMAC to validate the authenticity of the received file.

HMACs are almost like digital signatures, the main difference is that digital signatures use asymmetric keys, while HMACs use symmetric keys.

A client and server were implemented on a local machine to demonstrate the above-mentioned technique.

**Client Functionality**:

The client has the capability to encrypt the file to be transferred and send. Following is the process client follows to encrypt and transfer the file.

Calculate the HMAC using SHA1 hash and a randomly generated 16-byte key of the file to be sent. This is transferred to the server and the server used this to validate the integrity of the file.

1. Get the RSA public key from the server.
2. Generate a 16-byte keys using a Random Number Generator which are used as the AES key and HMAC secret key.
3. Encrypt the file using above generated random key with AES Encryption.
4. Encrypt the 128-bit AES key using the Server’s public key.
5. Prepend the calculated HMAC, its secret key and the Encrypted AES key to the Encrypted file.
6. Then the IV is appended to the file.

**Server Functionality:**

The server first generates the key pairs for RSA encryption. Once a file is received by the server, server triggers the decrypt () function to decrypt the received file. Following is the decryption and integrity validation process followed by the server.

1. Server uses its private key to decrypt the encrypted HMAC secret key and AES symmetric key.
2. Server then used the decrypted AES symmetric key to decrypt the Encrypted file.
3. Finally, decrypted file’s HMAC digest is calculated and compared with the HMAC value sent along with the encrypted file. If two hashes are equal, integrity validation passes. Otherwise, server recognizes the file integrity failed and deletes the file.

Server is designed to accept anonymous logins. And, anonymous users are given enough permissions to transfer files to the server using raw FTP operations.

AES Encryption is used with a 16-byte key as the symmetric key. The file to be sent is chunked into fixed sized blocks of the size multiple of 16 since AES Block Cipher standard uses 16 bytes blocks. When chunked, any blocks which does not fit into a size multiple of 16 bytes are padded appropriately with spaces. CBC (Cipher Block Chaining) is used as the AES Mode of Operation. CBC uses a 16 byte IV (Initialization Vector) which is generated randomly using Python’s Random module. IV is also appended to the encrypted file which is used for decryption.

RSA public key encryption is used to encrypt the AES symmetric key. This encrypted symmetric key also appended to the encrypted file to be decrypted by the other end. I have used RSAES-OAEP (Optimal Asymmetric Encryption Padding) to create an encryption cipher for RSA using the RSA public key.

Utility is developed completely using Python. Pyftpdlib and ftplib are used for FTP server and the FTP client functionalities and PyCryptodome library is used for Cryptographic functionality including AES and RSA encryption.

Change the file path in server.py to the folder where server.py script is, I had a different folder for this to see that the file was transferred. Run server.py first

To run server script: python server.py

Keep the client.py and the file to be transferred in the same folder

To run client script: python client.py <filename>