In our cryptography final project, using the SSL protocol to ensure secure message transmission between a client and a bank was a highly practical and effective choice. Establishing a shared key through a public key cryptosystem and then encrypting/decrypting messages with a symmetric encryption scheme is a common practice in modern cryptographic communications. This design in your project not only ensures the confidentiality of the communications but also demonstrates the diversity and flexibility of encryption technologies.

Firstly, allowing the client to choose the public key cryptosystem for establishing the shared key is a very flexible design. The options provided include RSA, RSA_OAEP, and Pailier, each with its own advantages and applicable scenarios:

RSA: This is the most widely known public key encryption algorithm, based on the difficulty of factoring large numbers. It performs robustly in environments where only ciphertext is available since decrypting the ciphertext without the private key is extremely challenging.

RSA_OAEP (RSA Optimal Asymmetric Encryption Padding): This is an enhanced variant of RSA that uses optimal asymmetric encryption padding to improve security and resist chosen ciphertext attacks (CCA).

Pailier: This system supports homomorphic encryption, allowing computations on encrypted data, which is particularly useful in scenarios requiring data privacy preservation during computations.

By choosing among these algorithms, your project not only showcases the practicality of different encryption technologies but also enhances the security of the system. For example, by using RSA_OAEP, your system can withstand more complex attack scenarios, such as chosen ciphertext attacks.

After the shared key is established by the server , using Triple S-DES to encrypt all messages between the client and server is an efficient choice. Triple S-DES is faster than public key cryptosystems and it also provides substantial security, capable of resisting chosen plaintext attacks and meet-in-the-middle attacks.

Moreover, using HMAC (Hash-Based Message Authentication Code) to hash the plaintext messages with the session key will generate a message authentication code (MAC) that ensures not only the integrity of the messages but also prevents tampering during transmission. If tampering is detected, the system would notify the server or user and prevent the execution of the command.

In summary, our project effectively enhances the security and reliability of message transmission by utilizing a variety of encryption technologies and security measures. This design is not only suitable for theoretical study but also meets practical application needs, demonstrating the crucial role of cryptography in securing digital communications.

Operational Process in Secure Communication:

The secure communication process between the client and the bank server in your project can be broken down into the following steps:

1.Client Handshake protocol with the server

Our server and client use the TCP to connect to each other and will send the hello message after getting the connection to verify it is build the connection.

2.Initialization and Key Exchange

Client selects a public key cryptosystem: Based on requirements, the client chooses among RSA, RSA_OAEP, or Pailier for the public key cryptosystem. Factors influencing the choice might include the required level of security, computational efficiency, or the need for homomorphic encryption capabilities.

Generation and exchange of public and private keys: The server generates the public and private keys corresponding to the chosen cryptosystem. The public key is sent to the client, while the private key is securely stored at the server. To prevent man-in-the-middle attacks, digital signatures can be used to verify the authenticity of the public key.

3..Establishing the Session Key:

Encrypting the session key using the public key: The server generates a random session key and encrypts it using the client's public key. The encrypted session key is then sent back to the client. Then the client decrypts the session key using the private key it own: Upon receiving the encrypted session key, the client decrypts it using its private key to retrieve the original session key.

4.Secure Communication

Encrypting messages: Using the established session key, the client encrypts messages intended for the server using Triple S-DES, which provides an additional layer of security suitable for high-security requirements in symmetric encryption. Message authentication: The client uses the session key and HMAC to hash the encrypted messages, creating a MAC that ensures the messages have not been tampered with during transmission. Sending encrypted and authenticated messages: The encrypted messages, along with their MACs, are sent to the server.

5.Receiving and Verifying Messages

Decrypting messages: The server uses the session key to decrypt the received messages through Triple S-DES.

Verifying MAC: The server uses the same HMAC process to verify the integrity of the messages. If the calculated MAC does not match the received MAC, it indicates that the messages were tampered with during transmission, leading the server to refuse to execute the commands in the message and potentially report a security alert to the client.

6.Session Termination

Ending communication: Once a transaction or session is complete, the client and server can choose to discard the current session key and start the key exchange process anew for the next communication to maintain security.

Through this method, our project not only ensures the confidentiality and integrity of messages but also adapts to different security needs and scenarios through the choice of different encryption strategies. This flexible yet rigorous design keeps the communication process efficient while significantly enhancing security.

How to run the code:

First , download our code and make sure to have every file you need.
Second: run the python server.py
Third: run the python client.py
Fourth: After the connect build we can choose the way to exchange the session and do the bank operation.

Contributions:
Wei Jun Li: Public key cryptosystems + Server Client servers+document
Eric Zhong: Symmetric key encryption + help with pkcs and server client+document
XiaoXiao He: SHA1+ hmac