利用OpenSSL库对Socket传输进行安全加密(RSA+AES)   
1. 利用RSA安全传输AES生成密钥所需的Seed(32字节)   
2. 利用AES\_encrypt/AES\_decrypt对Socket上面的业务数据进行AES加密/解密

理论上只需要AES就能保证全部流程，但由于AES加密所需要的AES-KEY是一个结构。 这个一个结构，如果通过网络进行传输，就需要对它进行网络编码，OpenSSL里面没有现成的API   
所以就引入RSA来完成首次安全的传输，保证Seed不会被窃听。同样，只使用RSA也能完成全部流程，但由于RSA的处理效率比AES低，所以在业务数据传输加密上还是使用AES

下面的代码包含了上述传输加密流程所需的所有步骤(OpenSSL部分)   
在实际的Socket应用开发时，需要将这些步骤插入到Client/Server网络通信的特定阶段

所需的OpenSSL主要的API及功能描述   
1. RSA\_generate\_key() 随机生成一个RSA密钥对，供RSA加密/解密使用   
2. i2d\_RSAPublicKey() 将RSA密钥对里面的公钥提出到一个BUF，用于网络传输给对方   
3. d2i\_RSAPublicKey() 将从网络传过来的公钥信息生成一个加密使用的RSA(它里面只有公钥)   
4. RSA\_public\_encrypt() 使用RSA的公钥对数据进行加密   
5. RSA\_private\_decrypt() 使用RSA的私钥对数据进行解密   
6. AES\_set\_encrypt\_key() 根据Seed生成AES密钥对中的加密密钥   
7. AES\_set\_decrypt\_key() 根据Seed生成AES密钥对中的解密密钥   
8. AES\_encrypt() 使用AES加密密钥对数据进行加密   
9. AES\_decrypt() 使用AES解密密钥对数据进行解密

一个典型的安全Socket的建立流程, 其实就是如何将Server随机Seed安全发给Client

C: Client S:Server

C: RSA\_generate\_key() --> RSAKey --> i2d\_RSAPublicKey(RSAKey) --> RSAPublicKey

C: Send(RSAPublicKey) TO Server

S: Recv() --> RSAPublicKey --> d2i\_RSAPublicKey(RSAPublicKey) --> RSAKey

S: Rand() --> Seed --> RSA\_public\_encrypt(RSAKey, Seed) --> EncryptedSeed

S: Send(EncryptedSeed) TO Client

C: Recv() --> EncryptedSeed --> RSA\_private\_decrypt(RSAKey, EncryptedSeed) --> Seed

* 1
* 2
* 3
* 4
* 5
* 6
* 7

— 到此, Client和Server已经完成传输Seed的处理   
— 后面的流程是它们怎样使用这个Seed来进行业务数据的安全传输

C: AES\_set\_encrypt\_key(Seed) --> AESEncryptKey

C: AES\_set\_decrypt\_key(Seed) --> AESDecryptKey

S: AES\_set\_encrypt\_key(Seed) --> AESEncryptKey

S: AES\_set\_decrypt\_key(Seed) --> AESDecryptKey

* 1
* 2
* 3
* 4

— Client传输数据给Server

C: AES\_encrypt(AESEncryptKey, Data) --> EncryptedData --> Send() --> Server

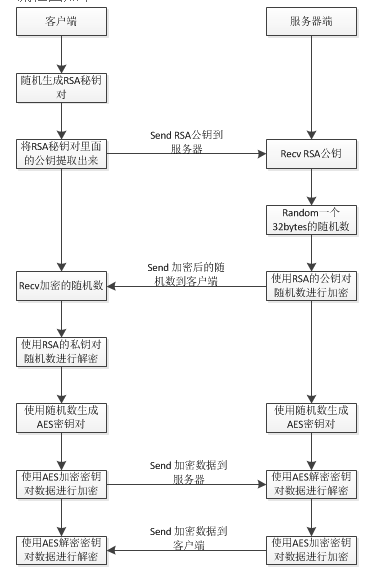
S: Recv() --> EncryptedData --> AES\_decrypt(AESDecryptKey, EncryptedData) --> Data --- Server传输数据给Client

S: AES\_encrypt(AESEncryptKey, Data) --> EncryptedData --> Send() --> Client

C: Recv() --> EncryptedData --> AES\_decrypt(AESDecryptKey, EncryptedData) --> Data

* 1
* 2
* 3
* 4

流程图如下：

   
相关的代码实现如下：

#include <string.h>

#include <openssl/rsa.h>

#include <openssl/aes.h>

int main()

{

// 1. 产生RSA密钥对

// 产生512字节公钥指数为RSA\_F4的密钥对，公钥指数有RSA\_F4和RSA\_3两种

// 我不清楚它们的区别，就随便选定RSA\_F4

// 可以使用RSA\_print\_fp()看看RSA里面的东西

RSA \*ClientRsa = RSA\_generate\_key(512, RSA\_F4, NULL, NULL);

// ---------

// 2. 从RSA结构中提取公钥到BUFF，以便将它传输给对方

// 512位的RSA其公钥提出出来长度是74字节，而私钥提取出来有超过300字节

// 为保险起见，建议给它们预留一个512字节的空间

unsigned char PublicKey[512];

unsigned char \*PKey = PublicKey; // 注意这个指针不是多余，是特意要这样做的，

int PublicKeyLen = i2d\_RSAPublicKey(ClientRsa, &PKey);

// 不能采用下面的方法，因为i2d\_RSAPublicKey()会修改PublicKey的值

// 所以要引入PKey，让它作为替死鬼

// unsigned char \*PublicKey = (unsigned char \*)malloc(512);

// int PublicKeyLen = i2d\_RSAPublicKey(ClientRsa, &PublicKey);

// 逐个字节打印PublicKey信息

printf("PublicKeyBuff, Len=%d\n", PublicKeyLen);

for (int i=0; i<PublicKeyLen; i++)

{

printf("0x%02x, ", \*(PublicKey+i));

}

printf("\n");

// ---------

// 3. 跟据上面提出的公钥信息PublicKey构造一个新RSA密钥(这个密钥结构只有公钥信息)

PKey = PublicKey;

RSA \*EncryptRsa = d2i\_RSAPublicKey(NULL, (const unsigned char\*\*)&PKey, PublicKeyLen);

// ---------

// 4. 使用EncryptRsa加密数据，再使用ClientRsa解密数据

// 注意, RSA加密/解密的数据长度是有限制，例如512位的RSA就只能最多能加密解密64字节的数据

// 如果采用RSA\_NO\_PADDING加密方式，512位的RSA就只能加密长度等于64的数据

// 这个长度可以使用RSA\_size()来获得

unsigned char InBuff[64], OutBuff[64];

strcpy((char \*)InBuff, "1234567890abcdefghiklmnopqrstuvwxyz.");

RSA\_public\_encrypt(64, (const unsigned char\*)InBuff, OutBuff, EncryptRsa, RSA\_NO\_PADDING); // 加密

memset(InBuff, 0, sizeof(InBuff));

RSA\_private\_decrypt(64, (const unsigned char\*)OutBuff, InBuff, ClientRsa, RSA\_NO\_PADDING); // 解密

printf("RSADecrypt OK: %s \n", InBuff);

// ----------

// 5. 利用随机32字节Seed来产生256位的AES密钥对

unsigned char Seed[32]; // 可以采用Rand()等方法来构造随机信息

AES\_KEY AESEncryptKey, AESDecryptKey;

AES\_set\_encrypt\_key(Seed, 256, &AESEncryptKey);

AES\_set\_decrypt\_key(Seed, 256, &AESDecryptKey);

// ----------

// 6. 使用AES密钥对来加密/解密数据

// 注意，256位的AES密钥只能加密/解密16字节长的数据

strcpy((char \*)InBuff, "a1b2c3d4e5f6g7h8?");

AES\_encrypt(InBuff, OutBuff, &AESEncryptKey);

memset(InBuff, 0, sizeof(InBuff));

AES\_decrypt(OutBuff, InBuff, &AESDecryptKey);

printf("AESDecrypt OK: %s \n", InBuff);

// ----------

// 7. 谨记要释放RSA结构

RSA\_free(ClientRsa);

RSA\_free(EncryptRsa);

return(0);

}