2022年秋季密码学大作业

姓名：运开 专业：信息安全 学号：2012619

[2022年秋季密码学大作业 1](#_Toc126774158)

[一、 实验要求 1](#_Toc126774159)

[二、 保密通信协议设计 1](#_Toc126774160)

[2.1 AES密钥分配 2](#_Toc126774161)

[2.2 传输IV初始向量 2](#_Toc126774162)

[2.3 进入会话 3](#_Toc126774163)

[三、 程序功能 4](#_Toc126774164)

[3.1 通过RSA完成保密性和认证性的AES密钥分配 4](#_Toc126774165)

[3.2 AES加密会话 5](#_Toc126774166)

[四、 加密算法介绍 6](#_Toc126774167)

[4.1 RSA加解密传输AES密钥 8](#_Toc126774168)

[4.2 AES加解密传输IV和会话信息 9](#_Toc126774169)

[4.2.1 AES加密： 11](#_Toc126774170)

[4.2.2 AES解密 12](#_Toc126774171)

[五、 程序编译选项（运行环境） 13](#_Toc126774172)

# 实验要求

设计一个保密通信的协议，具体要求为：利用RSA公钥密码算法，为双方分配一个 AES算法的会话密钥，然后利用AES加密算法和分配的会话密钥，加解密传送的信息。

假设条件：假设通讯双方为A和B，并假设发方拥有自己的RSA公钥PKA和私钥SKA ，同时收方拥有自己的RSA公钥PKB和私钥SKB ，同时收发双方已经通过某种方式知道了双方的公钥。

# 保密通信协议设计

由于RSA加密算法安全性高，但加解密运算速度较慢，而AES加密算法运算速度较快，能够满足信息加解密后传输的实时性，因此协议采用RSA加密完成AES密钥分配，分配完AES密钥后使用AES算法进行会话信息的加密传输，为处理较长的会话信息，采用密码分组链接（CBC）模式的AES加密，因此需要一个IV初始向量，这个初始向量采用ECB模式的AES加密。

RSA：512位

AES：128位，ECB模式加密IV初始向量，CBC模式加密会话信息

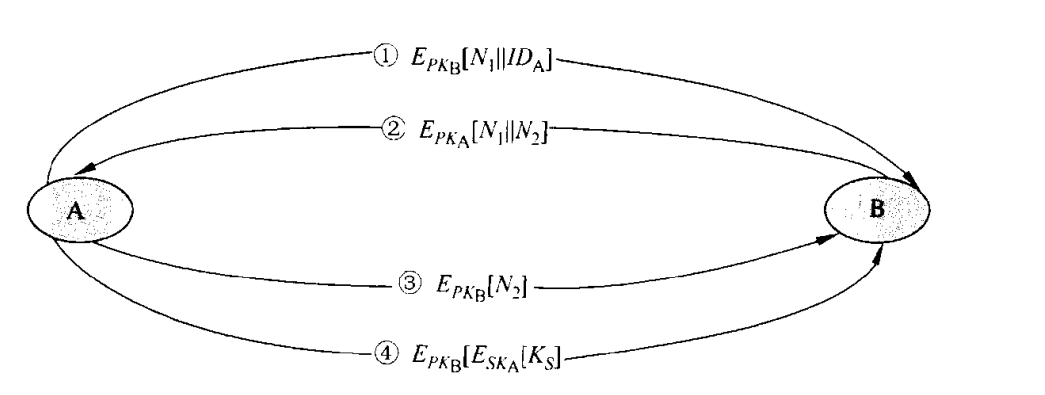
传输：采用TCP协议

## AES密钥分配

保密性和认证性的工作在A、B双方建立TCP连接后、传输AES密钥前完成。

采用如下方式完成密钥分配，具有保密性和认证性，既可以防止被动攻击，又可以防止主动攻击。（参考：现代密码学第4版第143页，清华大学出版社）密钥分配的步骤如下

1. A用B的公开钥加密 A的身份 IDA和一个一次性随机数 N1后发往 B，其中 N1用于惟一地标识这一业务。
2. B用A的公开钥 PKA加密 A 的一次性随机数 N1和B新产生的一次性随机数N2，后发往 A。因为只有B能解读1中的加密,所以B发来的消息中 N1的存在可使A 相信对方的确是 B。
3. A用B的公 PKB对N2加密后返回给 B以使 B相信对方的确是A。
4. A选一会话密钥 Ks，然后将 M=EPKB[ESKA[KS]]发给 B其中用B的公开钥加密是为保证只有 B能解读加密结果,用A的秘密加密是保证该加密结果只有 A能发送
5. B以DPKA[DSKB [M]]恢复会话密钥。

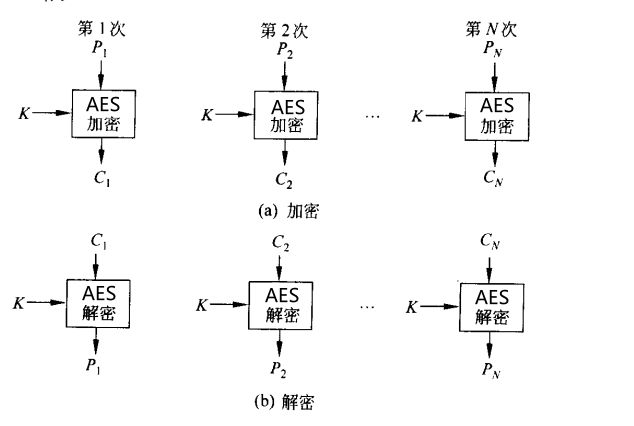


在本程序中将此阶段称为握手阶段，并使用int型变量shake\_state表示不同状态。

## 传输IV初始向量

CBC模式的AES需要一个初始向量IV，这个初始向量与每次AES加密数据的位数相同，在本实验中为128位，它由请求方生成，随后使用ECB模式加密，然后传输给被请求方。

ECB模式的AES加解密方式如下：

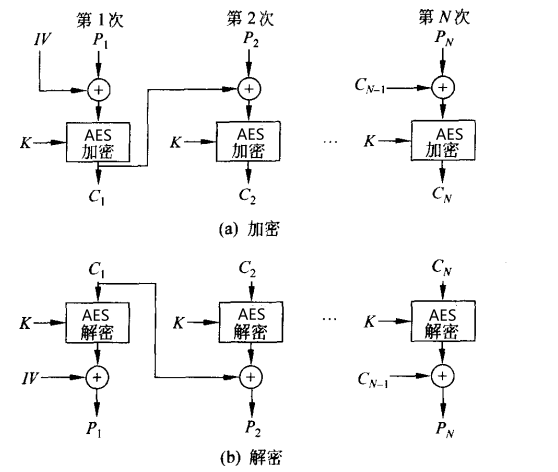


## 进入会话

此时A、B双方都可以向对方发送消息，发送的消息经过CBC模式的AES加密，并设置了单次传输最大字节为1024，接收到消息并解密后输出，格式为：

对方ID：message

CBC模式的AES加解密方式如下：



在CBC模式下，在加密前，每一块明文都与前一个密文块进行异或操作。这有助于确保任何明文的更改都将导致不同的密文，并且可以在解密过程中检测到，保证了认证性。此外，还使用了唯一的初始化向量（IV）以进一步增强加密的安全性。

# 程序功能

本协议基于TCP协议传输加密信息，因此A、B两个用户程序分别作为TCP连接中的客户端和服务端。

服务端启动后等待客户端连接。之后两程序确认是否向对方发送请求。

## 通过RSA完成保密性和认证性的AES密钥分配

建立会话所需的准备：

1. TCP建立连接
2. 一方发起会话请求
3. 完成保密性和认证性工作
4. 分配AES密钥
5. 传输初始向量IV

准备过程如下：

TCP客户端：



TCP服务端：

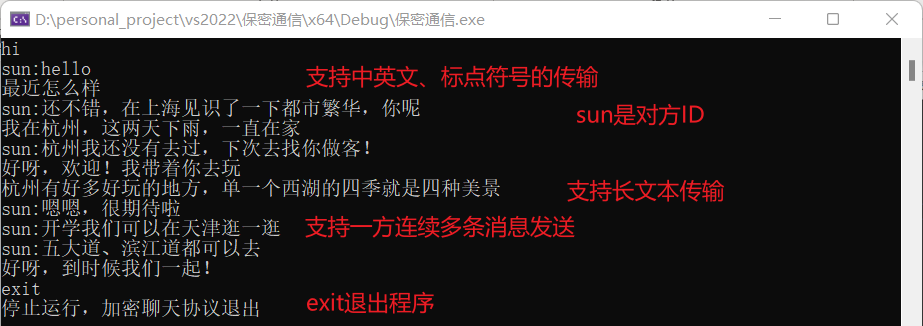


## AES加密会话

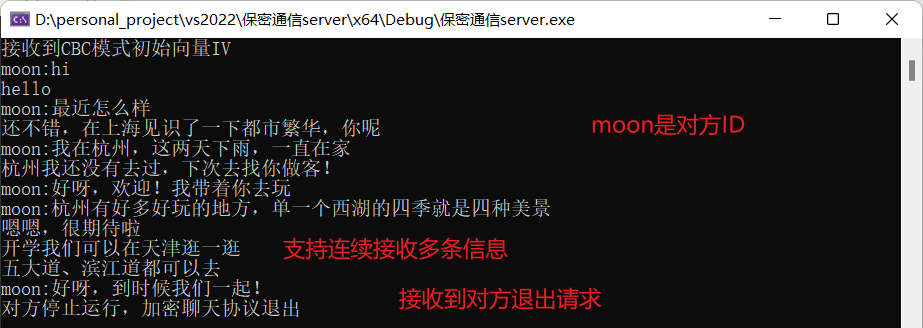
1. 支持中英文以及标点符号的传输
2. 支持长文本传输
   1. 传输的数据每次最长1024字节，通过CBC模式的AES连续加密后传输
3. 支持一方连续发送或接收多条消息
4. 支持双方同时发送、接收消息
   1. 程序使用多线程：收、发两线程分别处理收、发信息，互不干扰

建立连接后双方可以自由向对方发送会话信息：

TCP客户端：



TCP服务端：



# 加密算法介绍

加密完成后通过握手阶段完成保密性和认证性工作：

在处理收发线程时，本来实现的是一个线程专门用来收消息，一个专门用来发消息，在不用处理请求确认机制时这样没问题，但在处理握手消息，也就是请求确认时，收发消息的顺序是固定的，如果还用两个线程分别只处理收发则会导致两个线程间逻辑关系太多，耦合太多，不如将发线程完全设置为发起请求的一方，收线程完全设置为接收请求的一方，这样无论A、B哪个人先发起会话请求都可以分别在一个线程内完成，即使接收请求一方的发线程一直阻塞。

因此更合理的规划是线程越独立越好，最好能独自完成某部分工作，如果线程间耦合过多，会导致一个线程出问题后与之耦合的线程也无法继续运行。之前设置收发两个线程的本质目标也是这样。

故在发线程中也调用了socket的recv函数，这时收、发线程中都调用了recv函数，由此引来的新的问题是对方发送的消息可能随机被两个线程中的任意一个处理，因此发线程中的recv函数应当严格限制条件：只有进入到特定的握手阶段时才调用recv函数。

**发线程的流程图：**



**收线程的流程图：**



## RSA加解密传输AES密钥

RSA的加解密的数学原理基本相同，加密算法是对于明文m计算，得到密文c；解密算法是对于密文c计算，得到明文m；

**大体流程：**

首先将准备好的双方公钥 (n,e) 和私钥 (n,d) 保存，供加密和解密使用。

AES密钥由请求方调用random(128,sKey)随机生成，每次会话生成的密钥都不相同。其中random生成随机数方式如下：

bigInt random(int len, bigInt& ret) {

if (len == 0) return bigInt("0");

if (len == 1) return bigInt("1");

bigInt x("1");

for (int i = 1; i < len; i++)

x = x \* bigInt("2");

string s(len, '0');

for (int i = 0; i < s.length(); i++)

s[i] += rand() % 10;

bigInt res = bigInt(s) % x + x;//结果大于等于2^(len-1)

if (res % bigInt("2") == bigInt("0"))

res = res + bigInt("1");

else

res = res;

ret = res;

return ret;

}

另外保密验证过程中的随机数N1、N2以及CBC模式初始向量IV都通过random()生成，增强安全性。

对于上面生成的信息s要经过RSA加密：调用 string2bigInt 函数将其转化为 bigInt 类型的数字 b，然后调用 power(b,e,n,bigInttemp) 和 power(b,d,n,bigInttemp) 对其进行加密和解密。加解密后的结果会调用 bigInt2string 函数将其转化为字符串并输出。

其中，bigInt 是自定义的数字类型，可以表示任意长度的整数。实现了加减乘除等基本运算，并且可以通过调用 power 函数计算模意义下的幂运算。bigInttemp是一个bigInt型的局部变量，起到传递参数的作用。

string2bigInt 和 bigInt2string函数分别用于将字符串转化为 bigInt 类型的数字和将 bigInt 类型的数字转化为字符串。

void decrypt\_rsa(bigInt\* Cipher, bigInt\* mes, bigInt n, bigInt e)

{

bigInt temp;

\*(mes) = power(\*(Cipher), e, n, temp);

}

void encrypt\_rsa(bigInt\* mes, bigInt\* Cipher, bigInt n, bigInt d)

{

bigInt temp;

\*(Cipher) = power(\*(mes), d, n, temp);

}

## AES加解密传输IV和会话信息

AES的全称是Advanced Encryption Standard，高级加密标准，是一种对称加密算法，用于传输通信数据。AES算法要求分组密码的分组大小为128位，密钥大小可以是128、192或256位。本次实验采用128位密钥。

AES加解密流程图：（分别对应程序中的Encrypt()和Decrypt()函数）



之后，Encrypt()和Decrypt()函数再分别经过encrypt\_aes和decrypt\_aes函数封装，能完成ECB模式和CBC模式的加解密

### AES加密：

void encrypt\_aes(char\* key, char\* mes, char\* cipher, linking\_mode mode) {

int i;

int sel, keysize, cont = 1;

sel = 1;

Nk = 4;

Nr = Nk + 6;

memcpy\_s(AES\_Key, 16, key, 16);

char str[1024];

memcpy\_s(str, 1024, mes, 1024);

KeyExpansion();

// sz is the cursor into the input string

int sz = 0;

int len\_in = 1024;

memset(in, 0, 1024);

memset(out, 0, 1024);

while (sz <= len\_in) {

int j = 0;

if (mode == ECB) {

while (j < Nb \* 4) {

in[j] = (unsigned char)str[sz];//ECB模式

j++;

sz++;

}

}

else {

if (sz == 0) {

while (j < Nb \* 4) {

in[j] = IV[j] ^ (unsigned char)str[sz];

j++;

sz++;

}

}

else {

while (j < Nb \* 4) {

in[j] = out[j] ^ (unsigned char)str[sz];//CBC模式

j++;

sz++;

}

}

}

Encrypt();

for (i = 0; i < Nb \* 4; i++) {

cipher[sz - Nb \* 4 + i] = out[i];

}

}

}

### AES解密

void decrypt\_aes(char\* key, char\* mes, char\* cipher, linking\_mode mode) {

int i;

int sel, keysize, cont = 1;

sel = 2;

Nk = 4;

Nr = Nk + 6;

memcpy\_s(AES\_Key, 16, key, 16);

char str[1024];

memcpy\_s(str, 1024, cipher, 1024);

KeyExpansion();

int sz = 0;

int len\_in = 1024;

memset(in, 0, 1024);

memset(out, 0, 1024);

while (sz <= len\_in) {

sz = fillBlock(sz, str, in);

Decrypt();

if (mode == ECB) {

for (i = 0; i < Nb \* 4; i++)

{

mes[sz - Nb \* 4 + i] = out[i];

}

}

else {

for (i = 0; i < Nb \* 4; i++)

{

mes[sz - Nb \* 4 + i] = out[i];

if (sz <= 16) {

mes[sz - Nb \* 4 + i] ^= IV[i];

}

else {

mes[sz - Nb \* 4 + i] ^= cipher[sz - Nb \* 8 + i];

}

}

}

}

}

# 程序编译选项（运行环境）

本程序使用VS2022编译运行，项目环境设置如下：

