南開大學
Nankai University

网络空间安全学院

密码学期末报告

大作业：基于 RSA 以及 AES 算法的保密通信的协议设计与实现

姓名：魏伯繁

学号：2011395

专业：信息安全

2023 年 1 月 7 日

目录

1 作业要求

1.1 应用场景

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。

1.2 功能实现

作业需要先设计出保密通讯协议的过程和具体步骤；

分别编写 A，B 两个用户端的程序，写清楚两个程序分别要完成的功能，并能够在两个程序间进行通讯；

对 AES 加密的保密信息，要求采用 CBC 模式进行加密；

题外注：本次实验的讲解视频随其他文件一同发送。

2 理论基础

在本实验中具体应用到了包括 AES 算法、RSA 算法、数据传输（socket 编程）、密码分组链接 CBC、无中心情况下的公钥加密分配单钥密码体制的方式等密码学基础知识，在这一部分将对这些理论基础进行简要阐述以便更好理解实验的实现设计。

2.1 AES 算法

2.1.1 算法简介

首先，在对实际的传输内容进行加解密时使用的是 AES 算法，AES 算法因为在之前的课程实验中已经进行了实验，所以在这里只是简单介绍 AES 的特点以及其流程简介。

高级加密标准 (AES) 为最常见的对称加密算法, 对称加密算法也就是加密和解密用相同的密钥

AES 为分组加密法将明文按组划分, 每组长度相等, 每次加密一组数据, 直到加密完整个明文, 在 AES 标准规范中，分组长度只能是 128 位，AES 是按照字节进行加密的，也就是说每个分组为 16 个字节（每个字节 8 位）。密钥的长度可以使用 128 位、192 位或 256 位。这导致密钥长度不同，推荐加密的轮数也不同

2.1.2 算法流程

AES 算法的轮函数包括字节替换、行位移、列混淆 1 以及轮密钥加，具体的轮数视密钥的长度而变化，在加密时第一轮之前要进行轮密钥加，最后一轮没有进行列混淆操作，解密时同样如此，这样就保证了 AES 加密解密的对称性。

正是由于这样的设计，AES 算法才具备设计简单、在多个平台上速度快，编码紧凑、抵抗所有已知攻击、没有采用 Feistel 结构，轮函数由 3 个不同的可逆均匀变换构成：非线性层、线性混合层和密钥加层的特点

下图展示了 AES 算法的加解密示意图

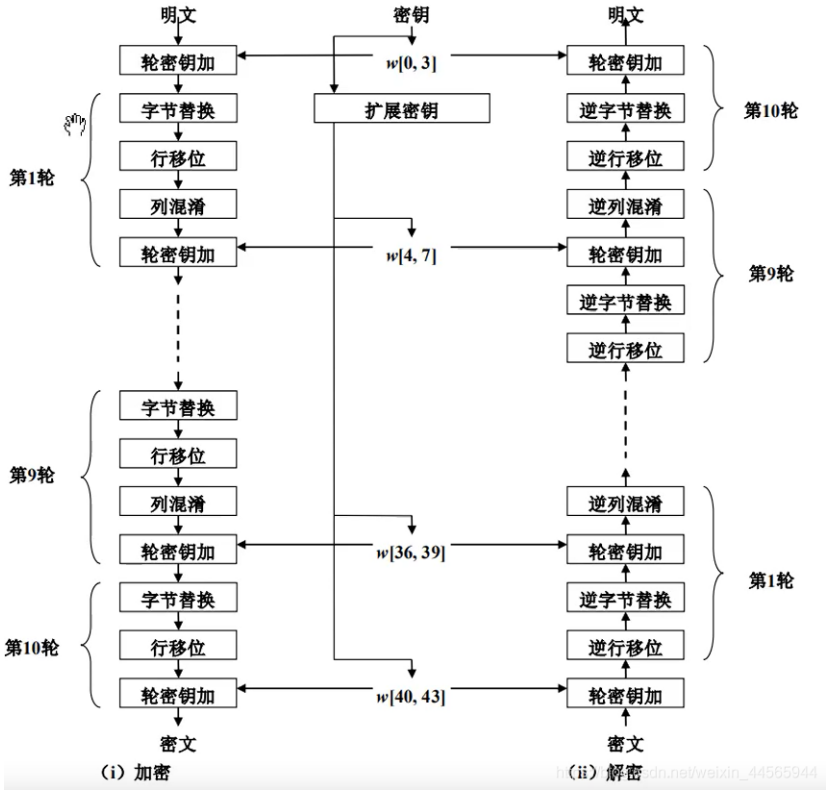

图 2.1: AES 算法流程图

2.2 RSA 算法

2.2.1 RSA 简介

经过了对 AES 的介绍，我们会发现，为了实现 AES 算法，通信双方需要使用共同的 AES 密钥，而为保证 AES 密钥在传递过程中的安全，传递时使用 RSA 加密算法对密钥进行加密。

RSA 算法是一种非对称性加密算法，简单来说 RSA 算法运用了”一个大整数进行因式分解具备一定的难度”这个数学知识来进行加密，对一个极大整数做因式分解越难，那么想要破解加密过后的密码就越难。

实现 RSA 加密算法的难点在于自主设计一个大整数类，这个大整数类可以实现 512 比特的大素数的加减乘除取模，也可以作为参数被调用计算大整数间的互素、最大公因数、乘法逆元等关系。

2.2.2 公钥密码体制简介

RSA 作为公钥密码体制的代码完美展现了公钥密码体制的如下优点：公钥密码体制是不对称密钥，优点是运算速度快，密钥产生容易。

(1) 保密强度高

其理论基础是基于数论中大素数因数分解的难度问题，当 n 大于 2048 位时，目前的算法无法在有效时间内破译 RSA。

(2) 密钥分配及管理简便

在 RSA 体制中，加密密钥和解密密钥互异、分离。加密密钥可以公开，解密密钥则由用户秘密保存，秘密保存的密钥量减少，这就使得密钥分配更加方便，便于密钥管理。

(3) 数字签名易实现

在 RSA 体制中，只有接收方利用自己的解密密钥对明文进行签名，其他任何人可利用公开密钥对签名文进行验证，但无法伪造。

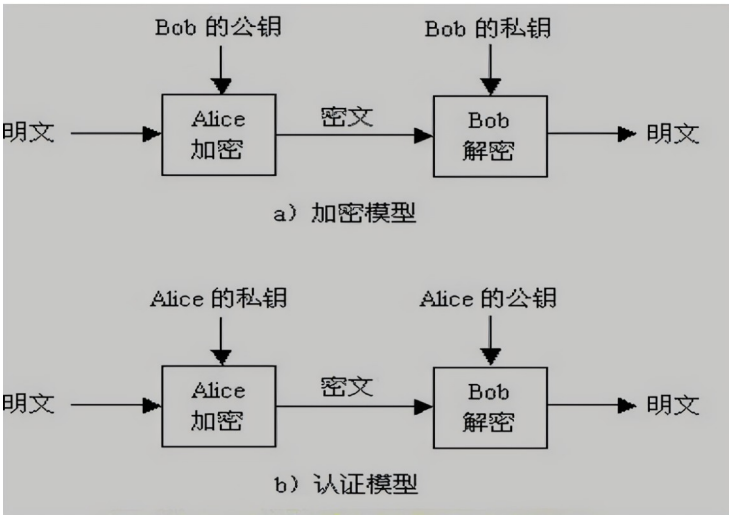

图 2.2: 公钥密码体制示意图

2.2.3 RSA 具体流程

在理解了公钥密码体制的运行流程后，很容易的就可以理解 RSA 密码体制的运作流程，下图展示了 RSA 算法的加密及解密流程

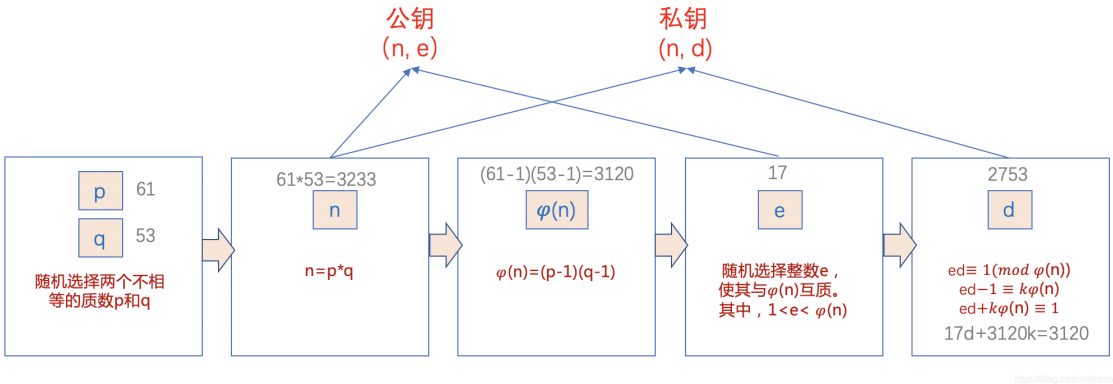

图 2.3: RSA 示例

2.3 CBC 模式

CBC 模式的全称是 Cipher Block Chaining 模式（密文分组链接模式），之所以叫这个名字，是因为密文分组像链条一样相互连接在一起。

在 CBC 模式中，首先将明文分组与前一个密文分组进行 XOR 运算，然后再进行加密。下图展示了 CBC 模式的示意图

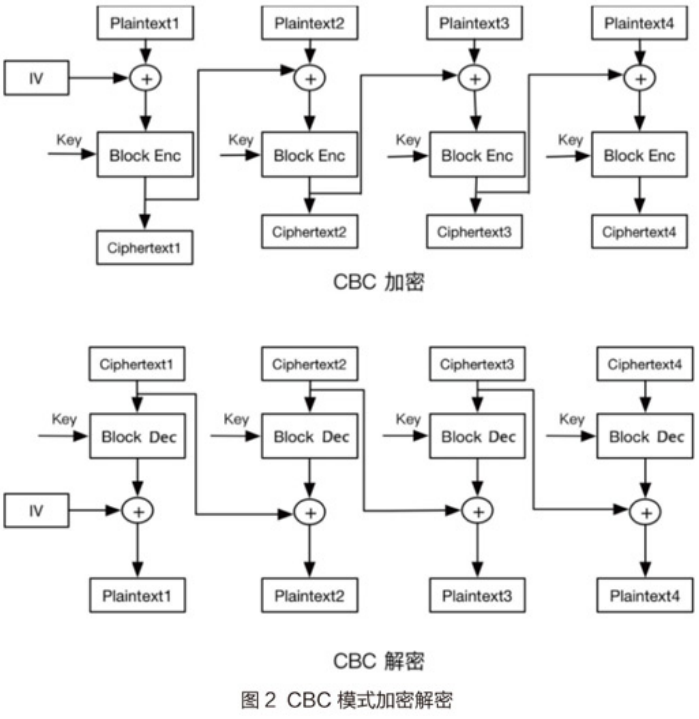

图 2.4: CBC 模式示意图

明文分组在加密之前一定会与“前一个密文分组”进行 XOR 运算，因此即使明文分组 1 和明文分组 2 的值是相等的，密文分组 1 和 2 的值也不一定是相等的。这样一来，ECB 模式的缺陷在 CBC 模式中就不存在了。

2.4 公钥加密分配单钥密码体制

基于公钥的单钥密码分配方式有多种，在此着重介绍无中心化的密钥分配体制，在本次实验中，作者严格按照教材中的流程进行 AES 密钥分配，具体的分配流程示意图如下图所示：

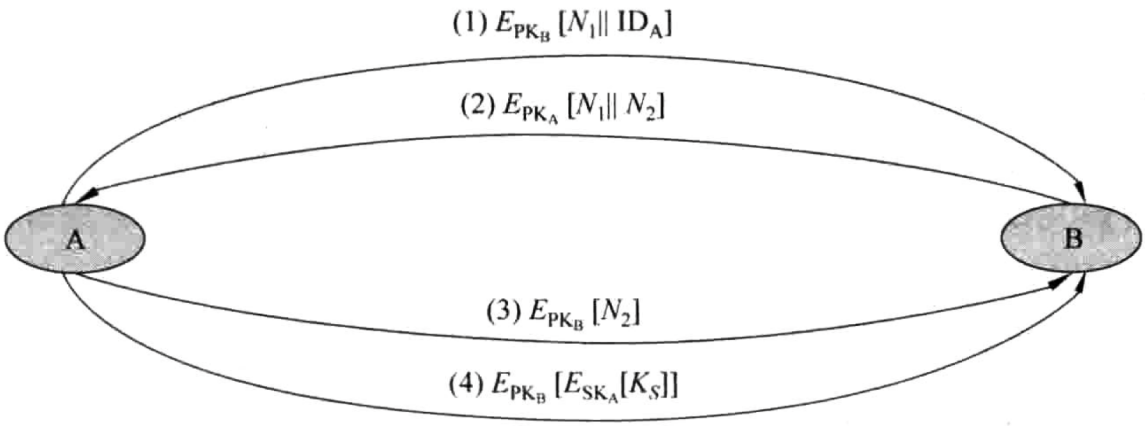

图 2.5: AES 密钥分配流程示意图

具体流程为：假定 A B 双方已完成公钥交换，可按以下步骤建立共享会话密钥:
(1)A 用 B 的公开钥加密 A 的身份 IDA 和一个一次性随机数 N 后发往 B 其中 N 用于唯一地标识这

一业务。
(2)B 用 A 的公开钥 PK、加密 A 的一次性随机数 N 和 B 新产生的一次性随机数 N 后发往 A。因为只有 B 能解读 (1) 中的加密消息, 所以 B 发来的消息中 N 的存在可使 A 相信对方的确是 B。
(3)A 用 B 的公钥 PK，对 N, 加密后返回给 B, 以使 B 相信对方的确是 A。
(4)A 选取一个会话密 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 使用 Dpkb[DsKa,[M]] 恢复会话密
在本次实验中使用的公钥密码体制为 RSA 算法

2.5 数据传输

最后，是本次实验的数据传输实现，在本次实验中使用 socket 编程来实现双方的数据通信。具体的编程流程如下图所示。

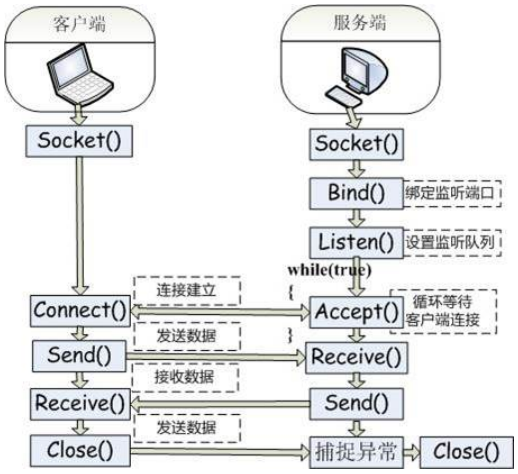

图 2.6: socket 流程示意图

3 保密通信协议设计

3.1 简介

根据实验要求，我设计了一款基于 RSA 和 AES 算法的保密通信协议，其具体流程分为两部分，第一部分为使用 RSA 算法进行身份确认以及 AES 密钥的交换；第二部分为使用分配的 AES 密钥进行文件传输。

3.2 第一部分

首先介绍基于 RSA 的身份确认以及密钥传输算法部分，在本部分主要分为四个阶段。

第一个阶段为 AES 密钥分配者请求建立信道环节，在这个环节，AES 密钥的分配者（clientA）将首先将身份信息 IDA 与一个一次性立即数 N1 通过对方的 RSA 公钥进行加密并发送者 AES 密钥的接收者。

第二个阶段为 AES 密钥接收者（clientB）自证身份环节。AES 密钥的接收者通过使用自身的 RSA 密钥对包含 clientA 身份信息以及一次性立即数的消息进行解密，确认 clientA 的身份并获得一次性立即数 N1，随后接收者将 clientA 发送的一次性立即数加一作为 N2，并拼接在 clientA 选择的一次性

立即数后，将该信息使用 A 的公钥加密并传输给 clientA，如果信息正确，则 clientB 证明了自己拥有 clientB 的私钥，从而完成了身份自证

第三个阶段为 AES 密钥分配者（clientA）自证身份环节，clientA 通过使用自身的 RSA 密钥对收到的包含 N1 以及 N2 的信息进行解密，并将 N2 使用 clientB 的 RSA 密钥发回，如果对方确认 N2 正确，则证明 clientA 掌握了 A 的 RSA 私钥，则 A 完成了身份自证。

最后一个阶段，当双方确认对方身份正确后立即开始进行 AES 密钥分配。接收端进行保存，待发送端将 AES 密钥发送完成后接收端开始对收到的信息进行破译解密，随后获得 AES 密钥。在这里需要特别注意，加密方式是双重加密，也就是会先使用对方的公钥进行加密，在使用自己的私钥加密，这样可以保证只有对方能解密，也保证了该信号只有自己才能加密

3.3 第二部分

保密通信协议的第二部分为文件的传输交换以及对 AES 密钥的使用，具体的流程是：

首先，通讯二人选择自己在本次通讯中扮演的角色，选择发送的一方将选择自己想要传输的文件，并在确认后发送，待文件发送成功后，接收方将对文件进行解密并保存至本地。

4 具体实现

由于在本次大作业之前已经完成了对 RSA 以及 AES 算法的实现，所以本次实验将不再展示 RSA 以及 AES 算法的具体流程，但对会其对外提供的新接口予以说明。

4.1 AES 算法与 CBC 模式

在本次实验中需要实现 AES 算法加密传输内容，根据 AES 算法的加密特点，我们需要对明文进行分组，并对不足分组长度的加密单位进行填充，且填充内容可以在解密后被程序准确识别，不会造成冗余信息的产生。

并且，如果分组的长度大于 1，则还需要实现 CBC 模式，即将上一次的加密结果与本次的明文进行异或运算

下面的代码展示了具体的使用 AES 进行加密并完成 CBC 的功能

AES 加密及 CBC 模式实现

```
string useAESencrypt(string s) {
 if (s.size() <= 32) {
   if (s.size() < 32) {
     s = AppendStringzero(s);
   }
   //到这为止生成了一个128位的数，现在把逗号加上
   //s = AppendStringdh(s);
   string keys = OriginKey;
   cout << "只需要一次加密即可" << endl;
   cout << "明文为" << s << endl;
   cout << "密钥为" << keys << endl;
   cout << keys << endl;
   TestData1 mytd(128, 128, s, keys);
   encrypt(mytd);
```

```
    string ret = GetCstarHex(text, 1, 129);
    cout << "加密结果为" << ret << endl;
    return ret;
  }
  else {
    vector<string>vs;
    string ret="";
    string lastsecret = "";
    string keynow = OriginKey;
    //切割
    for(int i=0;i<s.size();i+=32){
      string temp = s.substr(i, 32);
      vs.push_back(temp);
    }
    //填零
    for (int i = 0; i < vs.size(); i++) {
      vs[i] = AppendStringzero(vs[i]);
    }
    cout << "需要" << vs.size() << "次加密" << endl;
    for (int i = 0; i < vs.size(); i++) {
      if (i == 0) {
        cout << "第一次加密明文为" << vs[i] << endl;
        cout << "第一次加密密钥为" << keynow << endl;
        TestData1 mytd(128, 128, vs[i], keynow);
        encrypt(mytd);
        //加密后的成为了temp
        string temp = GetCstarHex(text, 1, 129);
        cout << "第一次加密密文为" << temp << endl;
        ret.append(temp);
        lastsecret = temp;
      }
      else {
        //CBC
        string a = Hex2Bin(vs[i]);
        string b = Hex2Bin(lastsecret);
        string t = stringADD(a, b);
        t = Bin2Hex(t);
        cout << "第" << i + 1 << "次加密明文为" << t << endl;
        cout << "第" << i + 1 << "次加密密钥为" << keynow << endl;
        TestData1 mytd(128, 128, t, keynow);
        encrypt(mytd);
        //加密后的成为了temp
        string temp = GetCstarHex(text, 1, 129);
        cout << "第" << i + 1 << "次加密结果为" << temp << endl;
        ret.append(temp);
        lastsecret = temp;
      }
    }
    return ret;
```

```
64    }
65  }
```

4.2 RSA 信息传递

在本次实验中 RSA 的功能为分配 AES 密钥，其具体的实现也在密码学实验 4 中进行了实验，在这里仅展示具体使用的 RSA 接口，并以 clientA 使用自己的密钥加密为例进行展示。

RSA 接口访问

```
1   //用A的私钥加密
2  string encrypt_A_private(string s) {
3    // 传进16进制数，所以最多16位
4    if (s.size() > 16) {
5      cout << "错误的加密信息，请检查后重传" << endl;
6      string ss;
7      return ss;
8    }
9    //将输入的内容转换为2进制表示
10   string ss = string2Bin(s);
11   ss = cleanZero(ss);//去掉前导零
12   BigInt b = string2BigInt2(ss);//将输入内容变为BigInt
13
14   BigInt b_d = string2BigInt(private_key_A_d);//密钥中的幂次d
15
16   BigInt b_n = string2BigInt(public_key_A_n);//密钥中的模数n
17
18   BigInt res = MyquickPow(b, b_d, b_n);//完成计算
19   string ret = BigInt2Hex(res);//转换为16进制
20   return ret;
21 }
```

4.3 AES 密钥分配

AES 密钥分配是本次实验中最关键的一部分，也是需要大作业过程中撰写代码最多的一部分，在本部分中，作者严格使用密码学教材中使用的流程介绍进行 AES 密钥的分配与使用。

首先，程序会进行三轮的身份认证，其中包括 A 的身份 IDA，以及 N1、N2，在本次实验中 N1、N2 被置位了 0 和 1，如果双方能够正确的给出对 IDA、N1、N2 的加解密结果则可以证明通信双方确实是 clientA 和 clientB，以防敌手攻击。

在确认双方身份后 clientA 作为 AES 密钥的持有方会在得到允许后主动向 clientB 发送密钥，待所有密钥发送完毕后发送“[OVER]”代表密钥传输结束，随后 clientB 将进行密钥解析

下面的代码站在 clientA 的视角进行密钥分配

AES 密钥分配及身份确认算法

```
1   //用A的私钥加密
2  int allocKey() {
```

```
int check;
cout << "您是AES密钥的分配者，请保证对方成功上线后再进行输入" << endl;
cout << "请在开启客户端B后输入：1，之后将开始交换密钥" << endl;
cin >> check;
if (check != 1) { return -1; }
//发送身份识别
string t = N1 + IDA;
cout << "第一次身份认证明文为" << t << endl;
t = encrypt_B_public(t);
t = "[IDEN]" + t;
cout << "第一次身份认证密文为" << t << endl;
char* sendbuffer = new char[10000];
memcpy(sendbuffer, t.c_str(), t.size());
int ii = sendto(mySocket, sendbuffer, t.size(), 0, (sockaddr*)&oppo_addr, olen);

//接收反馈
char* recvbuffer = new char[10000];
memset(recvbuffer, 0, 10000);
recvfrom(mySocket, recvbuffer, 10000, 0, (sockaddr*)&oppo_addr, &olen);
string s = recvbuffer;
s = cleanString(s);
if (s.substr(0, 6) == "[IDEN]") {
  cout << "第二次身份认证匹配成功" << endl;
  s = s.substr(6, 10000);
}
cout << "第二次身份认证密文为" << s << endl;
s = decrypt_A_private(s);
cout << "第二次身份认证明文为" << s << endl;
if (s == N1 + N2) {
  cout << "身份确认成功，即将进行再确认" << endl;
}
else {
  cout << "身份认证失败" << endl;
  return -1;
}

//再次准备发送
t.clear();
cout << "第三次身份认证明文为" << N2 << endl;
t = encrypt_B_public(N2);
t = "[IDEN]" + t;
cout << "第三次身份认证密文为" << t << endl;
memset(sendbuffer, 0, 10000);
memcpy(sendbuffer, t.c_str(), t.size());
sendto(mySocket, sendbuffer, 10000, 0, (sockaddr*)&oppo_addr, olen);

Sleep(30 * 1000);

//准备发送AES密钥的第一部分
```

```
52    string key1 = AESKey.substr(0, 16);
53    cout << "AES密钥切割第一部分为" << key1 << endl;
54    key1 = encrypt_A_private(key1);
55    cout << "使用A的私钥加密后为" << key1 << endl;
56    int i = 0; vector<string>vs;
57    while (i < key1.size()) {
58      string temp = key1.substr(i, 16); i += 16;
59      vs.push_back(temp);
60    }
61    for (int i = 0; i < vs.size(); i++) {
62      cout << "二次切割后第" << i + 1 << "部分为" << vs[i] << endl;
63      vs[i] = encrypt_B_public(vs[i]);
64      cout << "使用B的公钥加密后为" << vs[i] << endl;
65      memset(sendbuffer, 0, 10000);
66      vs[i] = "[key1]" + vs[i];
67      memcpy(sendbuffer, vs[i].c_str(), vs[i].size());
68      sendto(mySocket, sendbuffer, 10000, 0, (sockaddr*)&oppo_addr, olen);
69    }
70
71    //准备发送AES密钥的第二部分
72    string key2 = AESKey.substr(16, 16);
73    cout << "AES密钥切割第二部分为" << key2 << endl;
74    key2 = encrypt_A_private(key2);
75    cout << "使用A的私钥加密后为" << key2 << endl;
76    i = 0; vs.clear();
77    while (i < key2.size()) {
78      string temp = key2.substr(i, 16); i += 16;
79      vs.push_back(temp);
80    }
81    for (int i = 0; i < vs.size(); i++) {
82      cout << "二次切割后第" << i + 1 << "部分为" << vs[i] << endl;
83      vs[i] = encrypt_B_public(vs[i]);
84      cout << "加密后为" << vs[i] << endl;
85      memset(sendbuffer, 0, 10000);
86      vs[i] = "[key2]" + vs[i];
87      memcpy(sendbuffer, vs[i].c_str(), vs[i].size());
88      sendto(mySocket, sendbuffer, 10000, 0, (sockaddr*)&oppo_addr, olen);
89    }
90
91    string last = "[OVER]";
92    memset(sendbuffer, 0, 10000);
93    memcpy(sendbuffer, last.c_str(), last.size());
94    sendto(mySocket, sendbuffer, 10000, 0, (sockaddr*)&oppo_addr, olen);
95  }
```

4.4 socket 编程

在本次实验中使用 sendto 以及 recvfrom 进行双方的通信

sokcet 初始化

```
void initalNeeded() {
  WSAStartup(MAKEWORD(2, 2), &wsadata);
    //指定clintA的性质
    my_addr.sin_family = AF_INET;//使用IPV4
    my_addr.sin_port = htons(8888);//server的端口号
    my_addr.sin_addr.s_addr = htonl(2130706433);//主机127.0.0.1
    //指定clientB的性质
    oppo_addr.sin_family = AF_INET;//使用IPV4
    oppo_addr.sin_port = htons(8887);//server的端口号
    oppo_addr.sin_addr.s_addr = htonl(2130706433);//主机127.0.0.1

    //绑定服务端
    mySocket = socket(AF_INET, SOCK_DGRAM, 0);
    bind(mySocket, (SOCKADDR*)&my_addr, sizeof(my_addr));
}
```

5 效果演示

、

5.1 RSA 分配 AES 密钥部分

首先，我们需要在 key.txt 中输入自己所掌握的 RSA 密钥，在本次实验中，我已经将双方的密钥写入 key.txt 中，请不要对其进行修改，否则可能会导致程序无法正常运行。

随后，程序会按照之前介绍的顺序进行身份确认，身份确认过程中程序的输出如图所示：

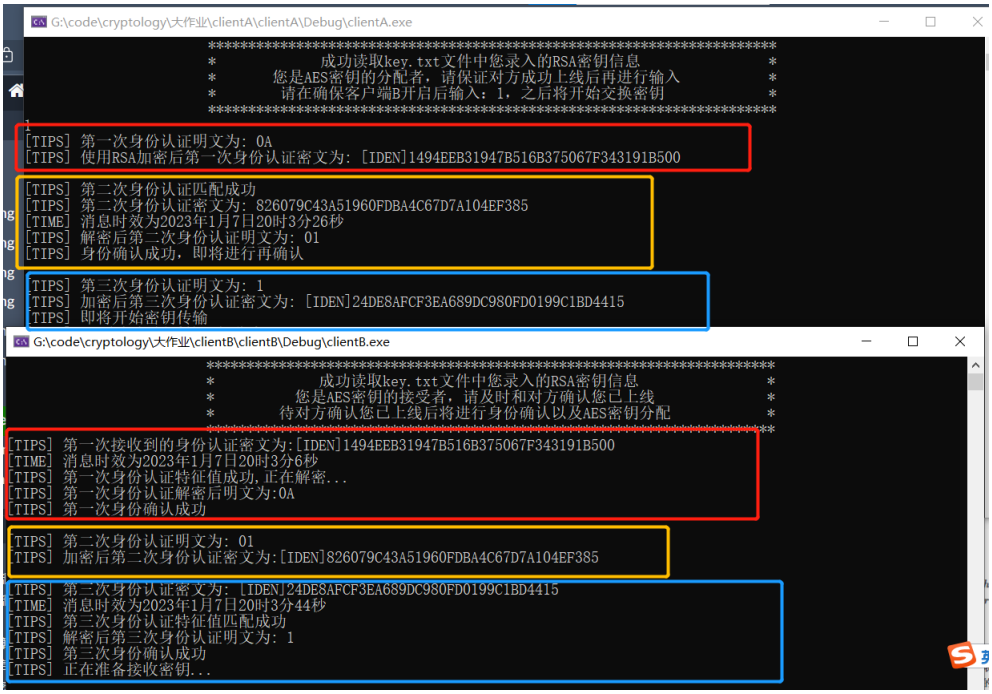

图 5.7: 身份互证部分

程序会展示每一个阶段所收到的密文原文以及解密后的明文，我们可以通过对照来进一步了解详细的加密过程

接下来则会进行 AES 密钥的分配，分配过程中 RSA 算法会对 AES 密钥进行切割，并对切割的部分先使用对方的公钥加密，再用自己的密钥加密。程序会将每一步的具体结果进行展示

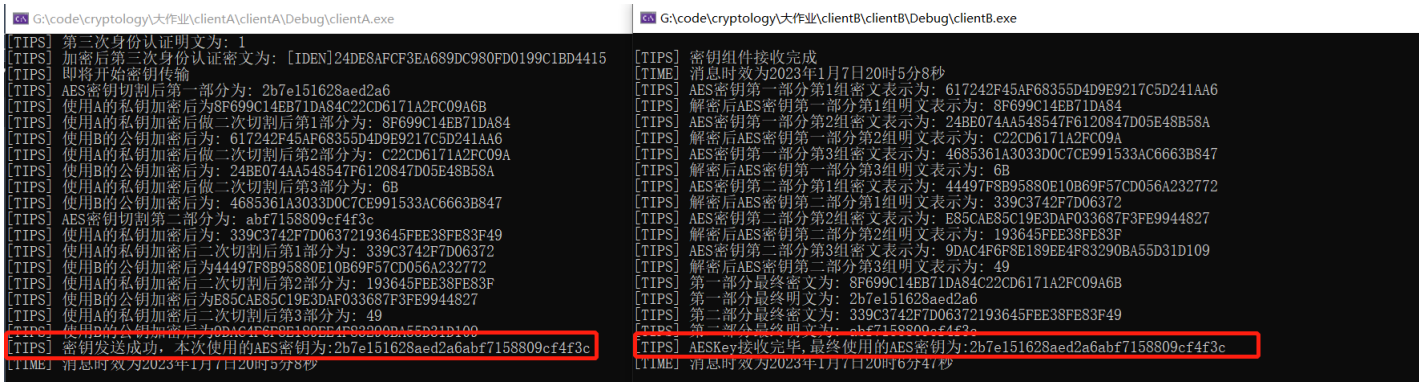

图 5.8: AES 密钥传输过程

5.2 文件传输过程

在文件传输过程中，需要通信双方选择自己是发送端还是接收端，随后接收端进入接收状态，而发送端则需要输入自己想要传输的文件路径，随后程序对文件内容进行读取并进行明文分组，随后加密并发送。

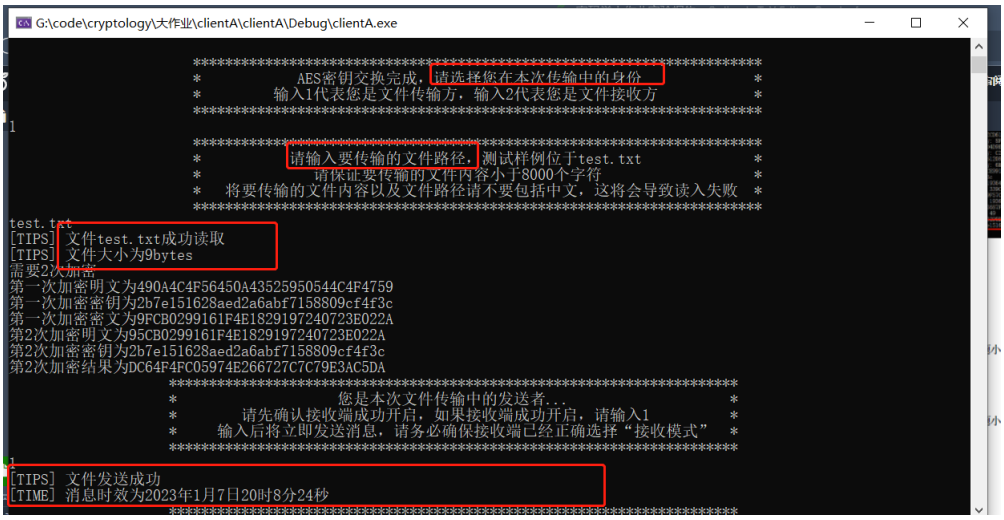

图 5.9: 发送端输出展示

下图是接收端的展示，接收端需要进行的操作就是被动等待，随后解密信息并写入本地。

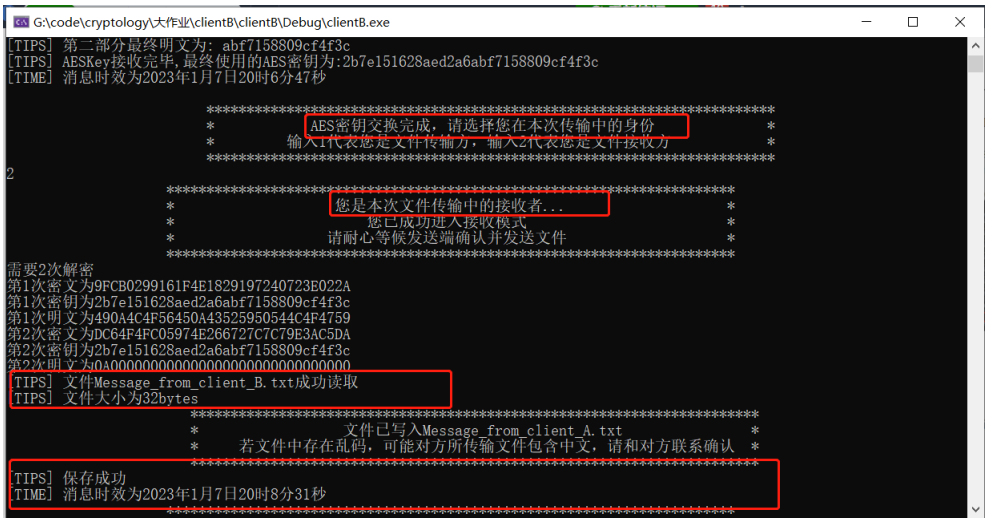

图 5.10: 发送端输出展示

5.3 传输结果展示

在本次实验中，我对 txt 文件的写入工作都采用了追加打开的方式，所以说，txt 中会保存对方传输的每一次消息，并且随接收时间一起保存。

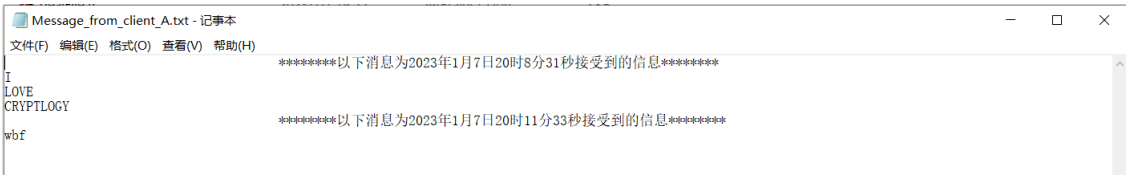

图 5.11: 传输结果保存

在经过了详细的介绍后，下图较为完整的描述了整个程序的运行状况，也较为清晰的演示了保密通讯协议的过程。

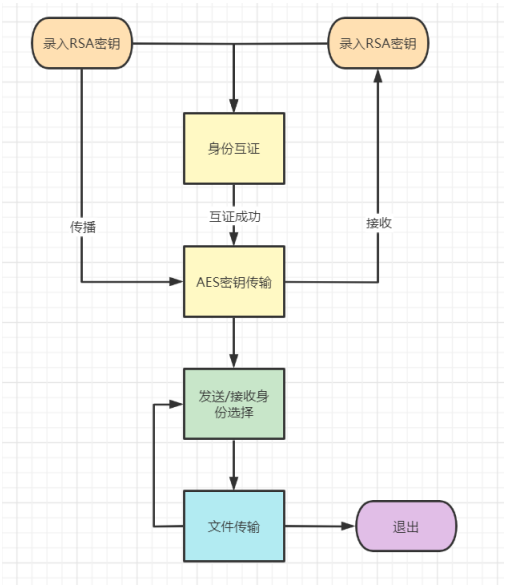

图 5.12: 程序整体流程图

6 附录

在该部分，我罗列出了一些其他的相对比较重要的实验代码，并对其做出解释。(其中我都删除了大量的输出代码)

6.1 交互函数

下图所示代码为程序与外界交互时的函数，其中 File2Key 函数负责从 key.txt 中读取 RSA 密钥，initalneed 函数负责 socket 初始化，allocKey 负责与对方交互分配 AES 密钥

双方交互代码

```
void realmain() {
  File2Key();
  initalNeeded();//初始化socket什么的
  allocKey();
  while (true) {
    int c1; cin >> c1;
    if (c1 == 1) {
      string s = readFile();
      s = useAESencrypt(s);
      char* sendbuffer = new char[10000];
      memcpy(sendbuffer, s.c_str(), s.size());
      int i = 0; while (i != 1) {
        cin >> i;
      }
      int e = sendto(mySocket, sendbuffer, s.size(), 0, (sockaddr*)&oppo_addr, olen);
      //cout << e << endl;
      //cout << WSAGetLastError() << endl;
      cout << "[TIPS] 文件发送成功" << endl;
      outTime();
    }
    if (c1 == 2) {
      char* recvbuffer = new char[10000];
      memset(recvbuffer, 0, 10000);
      while (recvfrom(mySocket, recvbuffer, 10000, 0, (sockaddr*)&oppo_addr, &olen)
          <= 0) {

      }
      string s(recvbuffer);
      cout << s << endl;
      s = cleanString(s);
      cout << s << endl;
      s = useAESdecrypt(s);
      WriteFile(s);
      cout << "[TIPS] 文件内容保存成功" << endl;
      outTime();
    }
    if (c1 != 1 && c1 != 2) { cout << "[TIPS] 错误的输入" << endl; }
```

```
37    }
38 }
```

6.2 文件读取

下面的代码展示了与文件的交互过程

文件交互代码

```
1  string readFile() {
2    string path;
3    cin >> path;
4    ifstream infile(path, ios::in);
5    if (!infile) {//文件没打开
6      cerr<<"[TIPS] 改文件不存在, 请重新检查路径"<<endl;
7    }
8    string ret;
9    string s;//我创建的变量, 存储数据用的
10   while (infile >> s) {//输入文件流
11     //cout << s;
12     ret.append(s);
13     ret.push_back('\n');
14   }
15   cout << "[TIPS] 文件" << path << "成功读取" << endl;
16   cout << "[TIPS] 文件大小为" << s.size() << "bytes" << endl;
17         ret = Text2Hex(ret);
18   return ret;
19 }
20 void WriteFile(string s) {
21   ofstream OutFile("Message_from_client_B.txt",ios::app);
       //利用构造函数创建txt文本, 并且打开该文本
22   s = Hex2Char2(s);
23   OutFile << myGetTime();
24   OutFile <<s; //把字符串内容"This is a Test!", 写入Test.txt文件
25   OutFile.close(); //关闭Test.txt文件
26   cout << "[TIPS] 文件" << "Message_from_client_B.txt" << "成功读取" << endl;
27   cout << "[TIPS] 文件大小为" << s.size() << "bytes" << endl;
28 }
```

6.3 AES 加解密接口

接下来的代码展示了真正在 realmain 函数调用的使用 AES 算法加密的代码。其具体内容也是根据输入的内容进行明文并对长度不足的分组进行填充后分别应用 AES 算法进行加密

逐列访问平凡算法

```
1  //传进来要加密的16进制字符串
2  string useAESencrypt(string s) {
3    if (s.size() <= 32) {
```

```
    if (s.size() < 32) {
      s = AppendStringzero(s);
    }
    //到这为止生成了一个128位的数，现在把逗号加上
    //s = AppendStringdh(s);
    string keys = OriginKey;
    cout << "只需要一次加密即可" << endl;
    cout << "明文为" << s << endl;
    cout << "密钥为" << keys << endl;
    cout << keys << endl;
    TestData1 mytd(128, 128, s, keys);
    encrypt(mytd);
    string ret = GetCstarHex(text, 1, 129);
    cout << "加密结果为" << ret << endl;
    return ret;
  }
  else {
    vector<string>vs;
    string ret="";
    string lastsecret = "";
    string keynow = OriginKey;
    //切割
    for(int i=0;i<s.size();i+=32){
      string temp = s.substr(i, 32);
      vs.push_back(temp);
    }
    //填零
    for (int i = 0; i < vs.size(); i++) {
      vs[i] = AppendStringzero(vs[i]);
    }
    cout << "需要" << vs.size() << "次加密" << endl;
    for (int i = 0; i < vs.size(); i++) {
      if (i == 0) {
        cout << "第一次加密明文为" << vs[i] << endl;
        cout << "第一次加密密钥为" << keynow << endl;
        TestData1 mytd(128, 128, vs[i], keynow);
        encrypt(mytd);
        //加密后的成为了temp
        string temp = GetCstarHex(text, 1, 129);
        cout << "第一次加密密文为" << temp << endl;
        ret.append(temp);
        lastsecret = temp;
      }
      else {
        //CBC
        string a = Hex2Bin(vs[i]);
        string b = Hex2Bin(lastsecret);
        string t = stringADD(a, b);
        t = Bin2Hex(t);
```

```
53          cout << "第" << i + 1 << "次加密明文为" << t << endl;
54          cout << "第" << i + 1 << "次加密密钥为" << keynow << endl;
55          TestData1 mytd(128, 128, t, keynow);
56          encrypt(mytd);
57          //加密后的成为了temp
58          string temp = GetCstarHex(text, 1, 129);
59          cout << "第" << i + 1 << "次加密结果为" << temp << endl;
60          ret.append(temp);
61          lastsecret = temp;
62        }
63      }
64      return ret;
65    }
66  }
```

7 总结

本次的密码学作业让我受益良多，因为之前的实验总是单独做一个实验，虽然编写出一个单独的密码学算法也会很有成就感，但是这种感觉并不强烈，但是编写出本次大作业却给予了我极大的成就感，究其原因，是因为我利用了再课堂上学习到的知识和自己的动手实践相结合，做出了一个有应用场景的实验，这样的动手实验会给予每一位学习的同学极大的鼓舞以及激励，我也会牢记本次实验中的收获，争取在密码学以及信息安全的学习到的道路上越走越远，非常感谢古力老师一学期的悉心指导。

参考文献 [1] 现代密码学第四版 2003 年清华大学出版社杨波