网络安全技术 —— 基于RSA算法自动分配密钥的加密聊天程序

学号：2013921
姓名：周延霖
专业：信息安全

一、实验目的

在讨论了传统的对称加密算法 DES 原理与实现技术的基础上，本章将以典型的非对称密码体系中 RSA 算法为例，以基于 TCP 协议的聊天程序加密为任务，系统地进行非对称密码体系 RSA 算法原理与应用编程技术的讨论和训练。通过练习达到以下的训练目的：

1. 加深对 RSA 算法基本工作原理的理解
2. 掌握基于 RSA 算法的保密通信系统的基本设计方法
3. 掌握在 Linux 操作系统实现 RSA 算法的基本编程方法
4. 了解 Linux 操作系统异步 IO 接口的基本工作原理

本章编程训练的要求如下:

1. 要求在 Linux 操作系统中完成基于 RSA 算法的自动分配密钥加密聊天程序的编写
2. 应用程序保持第三章“基于 DES 加密的 TCP 通信”中示例程序的全部功能，并在此基础上进行扩展，实现密钥自动生成，并基于 RSA 算法进行密钥共享
3. 要求程序实现全双工通信，并且加密过程对用户完全透明

二、实验内容

本章训练要求读者在第三章基于DES加密的TCP通信的基础上进行二次开发，使原有的程序可以实现全自动生成 DES 密钥以及基于 RSA 算法的密钥分配。

1. 要求在 Linux 操作系统中完成基于 RSA 算法的保密通信程序的编写
2. 程序必须包含 DES 密钥自动生成、RSA 密钥分配以及 DES 加密通讯三个部分
3. 要求程序实现全双工通信，并且加密过程对用户完全透明
4. 用能力的同学可以使用select模型或者异步IO模型对基于DES加密的TCP通信一章中`socket`通讯部分代码进行优化

三、实验步骤

注： 本次的des加密部分为复用作业“基于DES加密的TCP聊天”中的代码，加解密过程在上一次报告中已经详述

1、相关概念介绍

> 公钥密码体系的基本概念

传统对称密码体制要求通信双方使用相同的密钥，因此应用系统的安全性完全依赖于密钥的保密。针对对称密码体系的缺陷，Differ 和 Hellman 提出了新的密码体系—公钥密码体系，也称为非对称密码体系。在公钥

加密系统中，加密和解密使用两把不同的密钥。加密的密钥（公钥）可以向公众公开，但是解密的密钥(私钥)必须是保密的，只有解密方知道。公钥密码体系要求算法要能够保证：任何企图获取私钥的人都无法从公钥中推算出来

公钥密码体制中最著名算法是 RSA，以及背包密码、McEliece 密码、Diffe_Hellman、Rabin、零知识证明、椭圆曲线、ElGamal 算法

> 公钥密码体系的特点

公钥密码体制如下部分组成：

1. 明文：作为算法的输入的消息或者数据
2. 加密算法：加密算法对明文进行各种代换和变换
3. 密文：作为算法的输出，看起来完全随机而杂乱的数据，依赖明文和密钥。对于给定的消息，不同的密钥将产生不同的密文，密文是随机的数据流，并且其意义是无法理解的
4. 公钥和私钥：公钥和私钥成对出现，一个用来加密，另一个用来解密
5. 解密算法：该算法用来接收密文，解密还原出明文。

> RSA加密算法的基本工作原理

RSA 加密算法是一种典型的公钥加密算法。RSA 算法的可靠性建立在分解大整数的困难性上。假如找到一种快速分解大整数算法的话，那么用 RSA 算法的安全性会极度下降。但是存在此类算法的可能性很小。目前只有使用短密钥进行加密的 RSA 加密结果才可能被穷举解破。只要其钥匙的长度足够长，用 RSA 加密的信息的安全性就可以保证

2、重要函数分析

大素数生成（512位）

原理：

1. **线性同余算法生成随机数**

   - 该算法产生的是伪随机数，只具有统计意义上的随机性，**易被攻破**，不宜在现实情况中使用
   - 参数：
     - 模数：$m(m>0)$，为使随机数的周期尽可能大，m应尽量大，本次实验取$m=2^{31}-1$
     - 乘数：$a(0\leq a<m)$，是m的原根，$eg.a=7^5=16807$
     - 增量：c（本次实验中取0）
     - 初值种子：X_0，随机选取一32位整数，
   - 随机数序列：$X_{n+1}=(aX_n+c)\bmod n$

2. **Rabin-Miller素数概率检测算法**

- 定理1：如果p为大于2的素数，则方程$x^2 \equiv 1(\mathrm{mod} p)$的解只有$x \equiv 1$ 和 $x \equiv -1$
- 定理2：若n为素数，则$a^{n-1} \equiv 1 \bmod n$

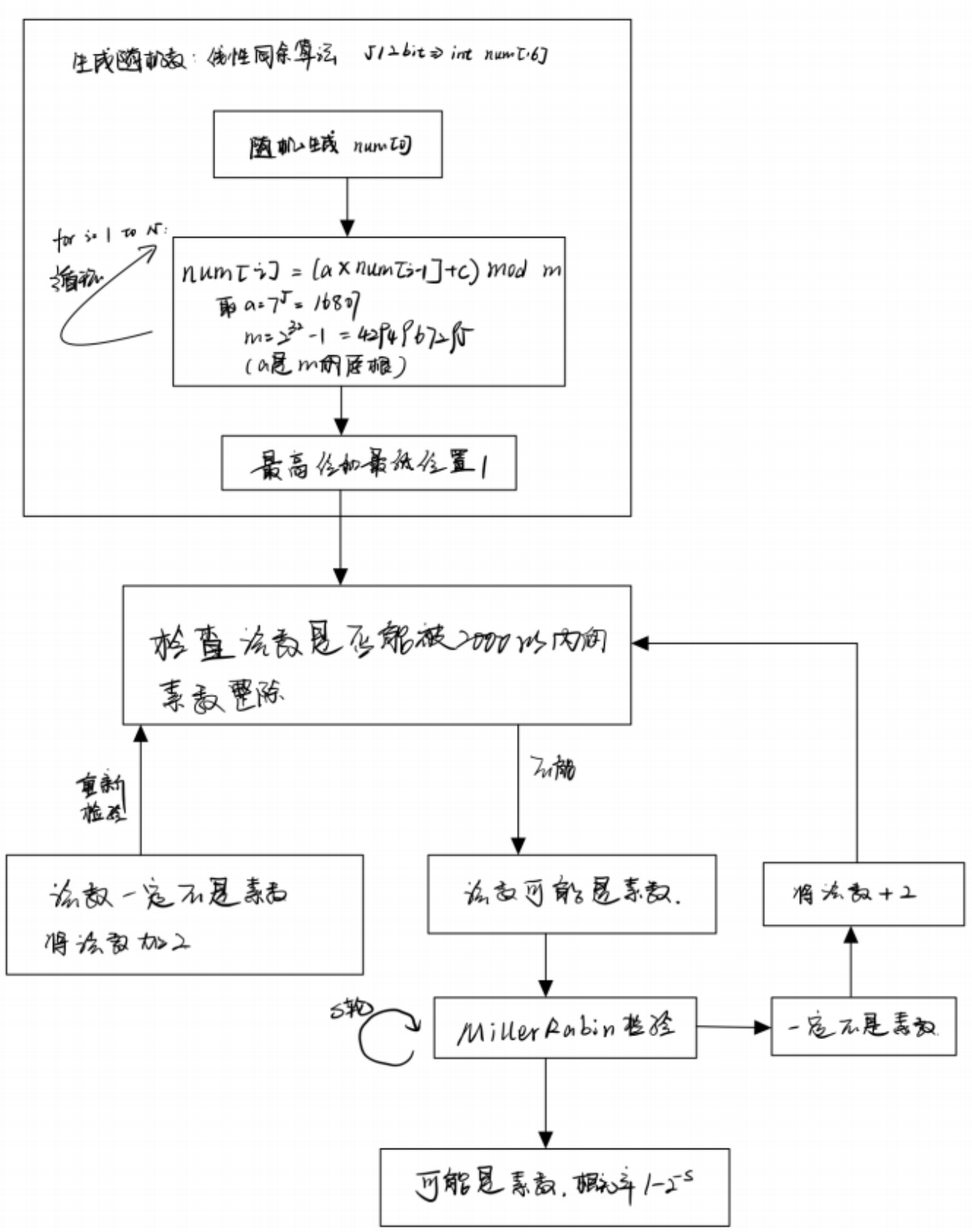
- 每轮检测的伪代码：

```
witness(a,n){
    d=1;//d初值为1
    for i=k downto 0 do{
        x=d;
        d=(d^2) % n;
        if(d==1&&x!=1&&x!=n-1)//若为素数，x不可能不是1或n-1
            return FALSE;
        if (n-1的2^i位为1)//
            d=(d*a) % n;
    }
    if(d!=1)return FALSE;//定理2
    return TRUE;
}
```

- 该算法为概率性检测，若进行s轮检测，则是素数的概率至少为$1-2^{-s}$

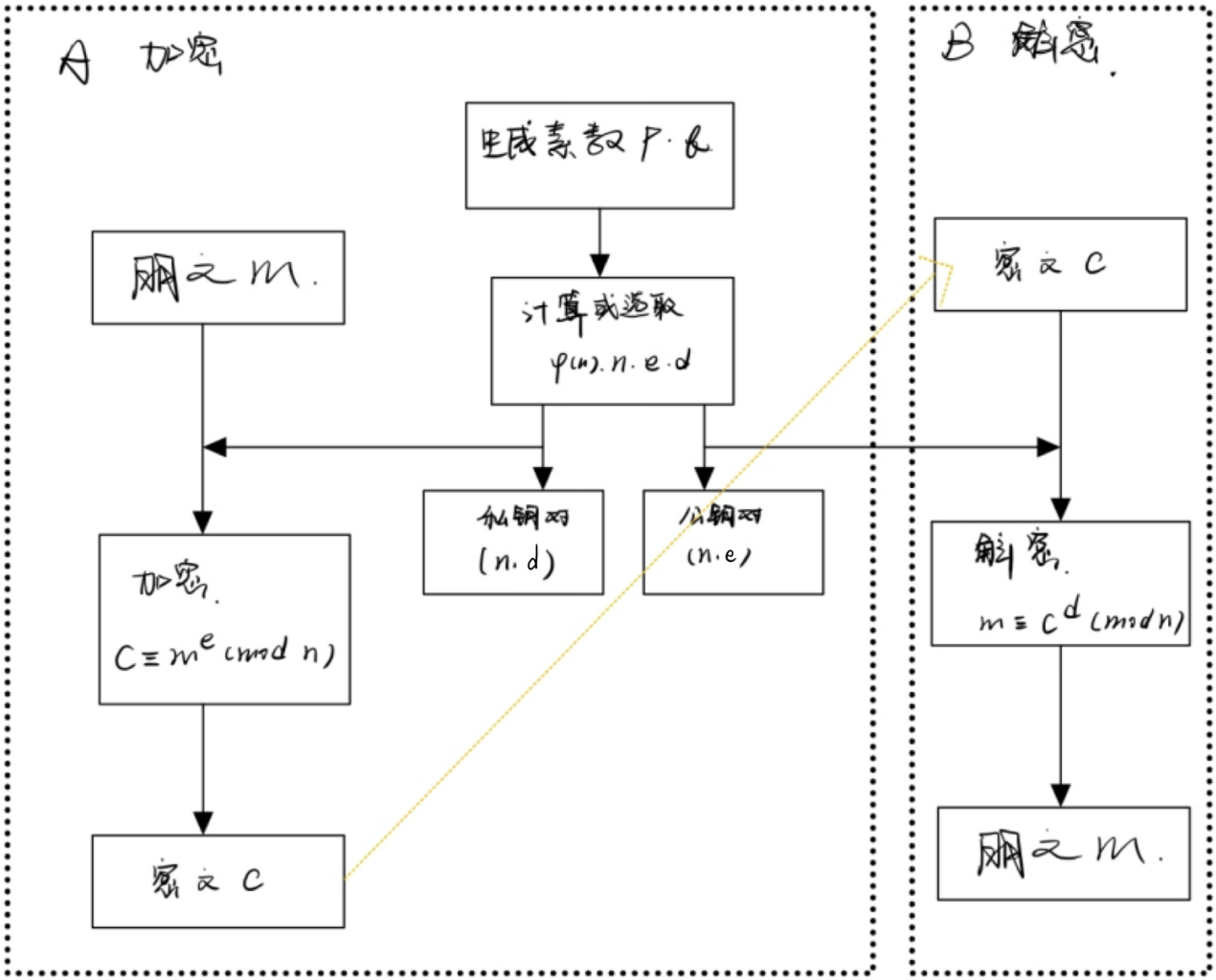
3. **Eratosthenes筛素数**

程序框图：



构建n的长度为1024比特的RSA算法，并利用该算法实现对明文的加密和解密。

程序框图



1. 生成密钥对

```
RSA_::RSA_(big p, big q,big e)
{
    big v1;
    v1.set(1);

    this->p = p;
    this->q = q;
    this->e = e;

    n = mul(p, q);//n=pq
    big p1, q1;
    p1 = sub(p, v1);//p-1
    q1 = sub(q, v1);//q-1

    phi = mul(p1, q1);//phi=(p-1)(q-1)
    d = getinv(phi, e);//逆元
}
```

2. 加密

```
RSAen_::RSAen_(RSA_ a, big m)
{
    this->n = a.n;
    this->e = a.e;
    this->m = m;

    c = pow(m, e, n);//m^e mod n
}
```

3. 解密

```
RSAde_::RSAde_(RSA_ a, big c)
{
    this->n = a.n;
    this->d = a.d;
    this->c = c;

    m = pow(c, d, n);//c^d mod n
}
```

3、通信过程

本次因为是检验rsa的分配密钥算法，所以用的还是单线程通信，并且也是客户端和服务器一人一条的收发消息，将主要通信过程分别封装在一个函数中，客户端封装在`runClient()`中，服务器封装在`runServer()`中，下面分别介绍这两个函数：

客户端`runClient()`

当与客户端与服务器建联成功后，先接收到服务器发过来的公钥，并将其存储起来，用于之后发送信息时可以用这个密钥进行加密：

```
        char publicKey[100] = { 0 };
        recv(ServerSocket, publicKey, 100, 0);
        printf("C: 从服务器接收到公钥:%s\n", publicKey);
        char sN[100], sE[100];
        int pN = 0, pE = 0;
        bool divide = false;
        for (int i = 0; i < strlen(publicKey); i++)
        {
            if (publicKey[i] == ',') { divide = true; continue; }
            if (!divide)
                sN[pN++] = publicKey[i];
            else
                sE[pE++] = publicKey[i];
        }
        PublicKey rsaPublic;
        rsaPublic.nE = atoi(sE);
```

```
rsaPublic.nN = atoll(sN);
char encryKey[300];
```

然后在本地生成自己的des密钥，并将用刚刚的公钥加密的DES密钥发送给服务器

```
char desKey[8] = { 0 }, plaintext[255], ciphtext[500] = { 0 };
GenerateDesKey(desKey);
printf("C: 随机生成DES密钥:");
for (int i = 0; i < 8; i++) {
    printf("%c", desKey[i]);
}
printf("\n");
//scanf("%s", desKey);
char tempK[8];
strcpy(tempK, desKey);
op.MakeKey(tempK);
int p = 0;
for (int i = 0; i < 4; i++) {
    int p1 = i * 2, p2 = i * 2 + 1;
    int num1 = int(desKey[p1]);
    int num2 = int(desKey[p2]);
    UINT64 curNum = (num1 << 8) + num2;
    UINT64 encry = Encry(curNum, rsaPublic);
    char cencry[20];
    sprintf(cencry, "%lu", encry);
    strncpy(encryKey + p, cencry, strlen(cencry));
    p += strlen(cencry);
    char divide = ',';
    encryKey[p] = divide;
    p += 1;
}
encryKey[p] = '\0';
//发送加密的DES密钥给给服务器
printf("C: 向服务器发送加密的DES密钥:%s\n", encryKey);
send(ServerSocket, encryKey, p, 0);
```

通信部分与上一次的代码是差不多的，只不过这次发送数据的时候要将des加密完的数据用rsa的公钥再次进行加密，最后退出用exit来表示

```
while (1)
{  //发送密文信息
    printf("C: 请输入明文:");
    setbuf(stdin, NULL);
    scanf("%[^\n]s", plaintext);//使得空行代表读取完毕而不是空格
    bool exit = false;
    if (strcmp(plaintext, "exit") == 0) { exit = true; }
    op.MakeData(plaintext);
    int count = 0;
    char time[64];
```

```
            strcpy(time, op.getTime());
            printf("C: [%s]向服务器发送密文:", time);
            for (int i = 0; i < op.groupCount; i++)
            {
                for (int j = 0; j < 64; j++)
                    ciphtext[count++] = op.ciphArray[i][j] + 48;//要加上48
            }
            ciphtext[count] = '\0';
            int ciphtexts[32];
            int asc = 0;
            for (int i = 0; i < count; i++) {
                int sub = ciphtext[i] - 48;
                sub = sub * pow(2, (7 - i % 8));
                asc += sub;
                if (i % 8 == 7) {
                    ciphtexts[i / 8] = asc;
                    asc = 0;
                }
            }
            for (int i = 0; i < count / 8; i++)
                printf("%d,", ciphtexts[i]);
            //发送数据给服务器
            send(ServerSocket, ciphtext, strlen(ciphtext), 0);
            if (exit)
                break;
            //利用返回的套接字和服务器通信，接收加密信息
            char s[256] = { 0 };
            recv(ServerSocket, s, 256, 0);
            int counts = strlen(s);
            int asc_recv = 0;
            int s1[32];
            for (int i = 0; i < counts; i++) {
                int sub = s[i] - 48;
                sub = sub * pow(2, (7 - i % 8));
                asc_recv += sub;
                if (i % 8 == 7) {
                    s1[i / 8] = asc_recv;
                    asc_recv = 0;
                }
            }
            printf("\nC: 接收服务器的密文:");
            for(int i=0;i<counts/8;i++)
                printf("%d,", s1[i]);
            printf("\n");
            memset(op.plaintext, 0, sizeof(op.plaintext));//初始化明文
            //收到加密信息后，进行解密
            op.groupCount = 0;
            for (int i = 0; i < strlen(s); i++)//拆解收到的加密信息，转为二进制
数组
            {
                op.ciphArray[op.groupCount][i % 64] = s[i] - 48;
                if ((i + 1) % 64 == 0)
                    op.groupCount++;
            }
```

```
            for (int i = 0; i < op.groupCount; i++)
                op.MakeCiph(op.ciphArray[i], i);
            //输出解密后的明文
            strcpy(time, op.getTime());
            printf("C: [%s]经过解密后的明文:", time);
            for (int i = 0; i < op.groupCount; i++)
                op.Bit2Char(op.textArray[i]);
            printf("%s\n", op.plaintext);
            if (strcmp(op.plaintext, "exit") == 0)
                break;
        }
```

服务器runServer()

首先在服务器端生成相应的公钥和私钥，并与刚连上的客户端通信，告诉客户端生成的公钥：

```
    char desKey[10] = { '\n' };
    RsaParam rsaParam = RsaGetParam();
    m_cParament.d = rsaParam.d;
    m_cParament.e = rsaParam.e;
    m_cParament.n = rsaParam.n;
    PublicKey publicKey = GetPublicKey();
    char cpublicKey[100], sN[100], sE[100];
    sprintf(sN, "%lu", publicKey.nN);
    itoa(publicKey.nE, sE, 10);
    strcpy(cpublicKey, sN);
    int p1 = strlen(sN), p2 = strlen(sE);
    cpublicKey[p1] = ',';
    strncpy(cpublicKey + p1 + 1, sE, p2);
    cpublicKey[p1 + p2 + 1] = '\0';
    printf("S: 向客户端发送公钥和加密密钥:%s\n", cpublicKey);
    send(ClientSocket, cpublicKey, strlen(cpublicKey), 0);
```

接收从客户端发来的des密钥，并用自己的私钥进行解密，这样两边就可以保持通信了：

```
    char encryKey[300] = { '\0' };
    recv(ClientSocket, encryKey, 300, 0);//接收密钥
    printf("S: 从客户端接收加密后的DES密钥:%s\n", encryKey);
    int k = 0;
    for (int i = 0; i < 4; i++) {
        char cencry[20] = { '\0' };
        int p = 0;
        while (encryKey[k++] != ',')
            cencry[p++] = encryKey[k - 1];
        UINT64 encry = atoll(cencry);
        UINT64 decry = Decry(encry);
        desKey[i * 2] = decry >> 8;
        desKey[i * 2 + 1] = decry % 256;
    }
```

```
    printf("S: 密钥解密后:%s", desKey);
    op.MakeKey(desKey);
```

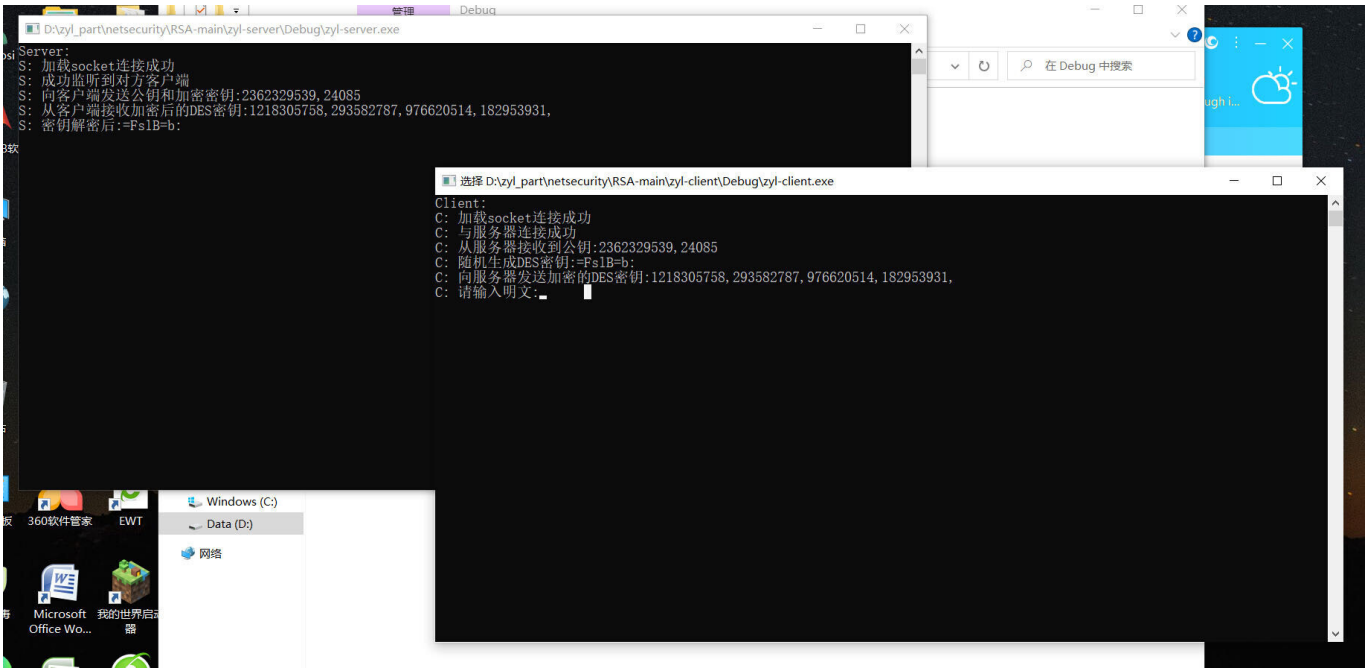
服务器的通信部分也与上一次的代码是类似的，也要将des加密完的数据用rsa的私钥再次进行加密，最后退出用exit来表示

```
    while (1)
    {
        memset(op.plaintext, 0, sizeof(op.plaintext));//初始化明文
        //利用返回的套接字和客户端通信
        char s[256] = { 0 };
        recv(ClientSocket, s, 256, 0);//接收密文
        int counts = strlen(s);
        int asc_recv = 0;
        int s1[32];
        for (int i = 0; i < counts; i++) {
            int sub = s[i] - 48;
            sub = sub * pow(2, (7 - i % 8));
            asc_recv += sub;
            if (i % 8 == 7) {
                s1[i / 8] = asc_recv;
                asc_recv = 0;
            }
        }
        printf("\nS: 接收客户端的密文:");
        for (int i = 0; i < counts / 8; i++)
            printf("%d,", s1[i]);
        printf("\n");
        //拆解收到的加密信息，转为二进制数组
        op.groupCount = 0;
        //printf("%d\n", strlen(s));
        for (int i = 0; i < strlen(s); i++)
        {
            op.ciphArray[op.groupCount][i % 64] = s[i] - 48;
            if ((i + 1) % 64 == 0)
                op.groupCount++;
        }
        //进行密文的解密
        for (int i = 0; i < op.groupCount; i++)
            op.MakeCiph(op.ciphArray[i], i);
        //输出解密后的明文
        char time[64];
        strcpy(time, op.getTime());
        printf("S: [%s]经过解密后的明文:", time);
        for (int i = 0; i < op.groupCount; i++)
            op.Bit2Char(op.textArray[i]);
        printf("%s\n", op.plaintext);
        if (strcmp(op.plaintext, "exit") == 0)
            break;
```

```
        //如果用户需要继续发送信息，则继续发送
        char plaintext[255] = { 0 }, ciphtext[500] = { 0 };
        printf("S: 请输入明文:");
        setbuf(stdin, NULL);
        scanf("%[^\n]s", plaintext);//使得空行代表读取完毕而不是空格
        bool exit = false;
        if (strcmp(plaintext, "exit") == 0)
           exit = true;
        op.MakeData(plaintext);
        int count = 0;
        strcpy(time, op.getTime());
        printf("S: [%s]向客户端发送密文:", time);
        for (int i = 0; i < op.groupCount; i++)
        {
            for (int j = 0; j < 64; j++)
                ciphtext[count++] = op.ciphArray[i][j] + 48;//要加上48
        }
        ciphtext[count] = '\0';
        int ciphtexts[32];
        int asc = 0;
        for (int i = 0; i < count; i++) {
            int sub = ciphtext[i] - 48;
            sub = sub * pow(2, (7 - i % 8));
            asc += sub;
            if (i % 8 == 7) {
                ciphtexts[i / 8] = asc;
                asc = 0;
            }
        }
        for (int i = 0; i < count / 8; i++)
            printf("%d,", ciphtexts[i]);
        //发送数据给服务器
        send(ClientSocket, ciphtext, strlen(ciphtext), 0);
        if (exit)
            break;
    }
```

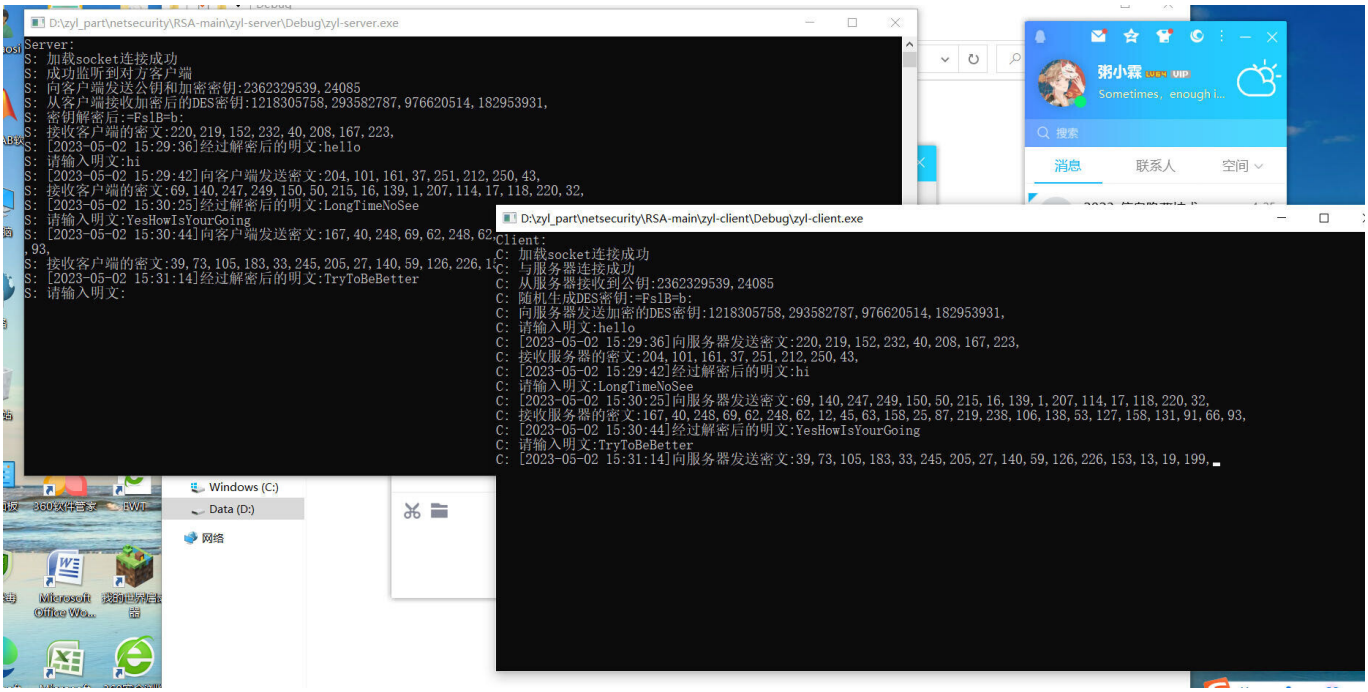
四、实验结果

1、建立连接成功



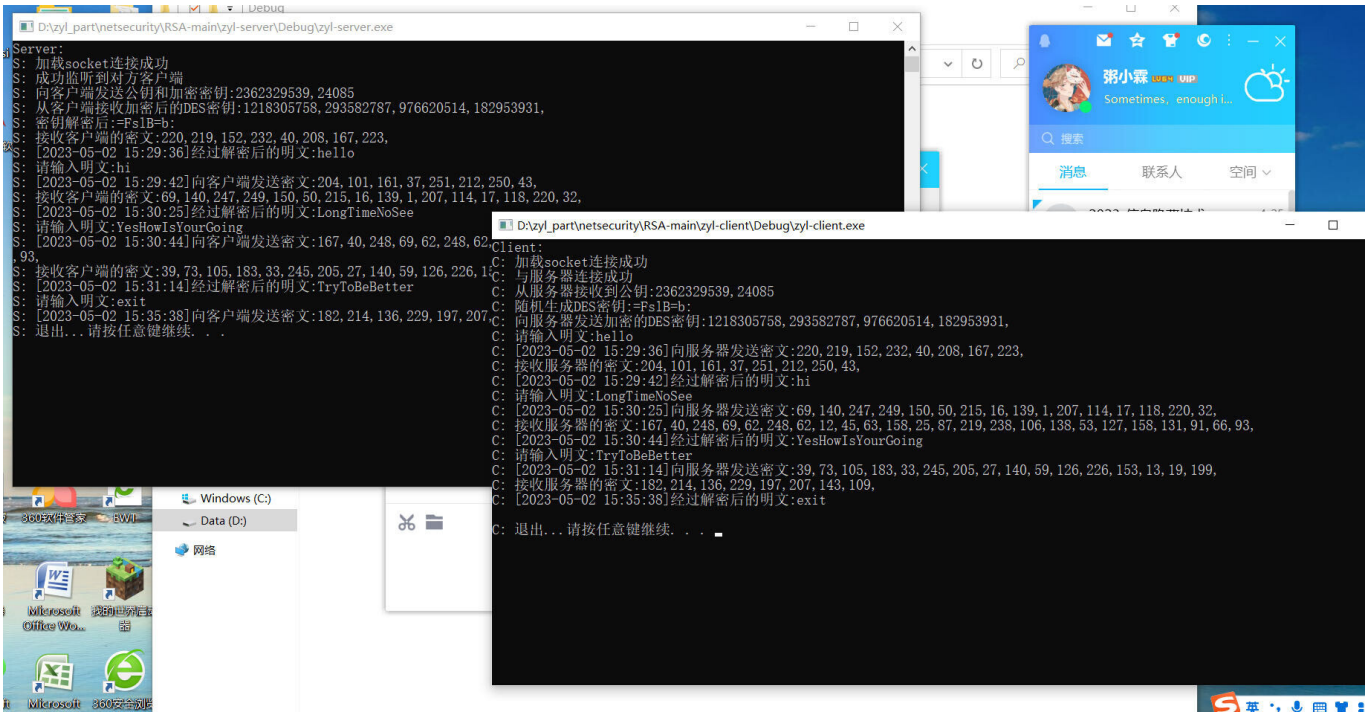
从上图可以看出连接的成功建立

2、进行聊天



从上图可以看出可以成功进行DES加密并进行通信

3、结束聊天



当一方发送提前设定好的结束词时双方结束聊天如上图所示

五、心得体会

在本次实验中，首先复习了当时编写des加解密的过程并应用，也学习了rsa加密如何生成并分配相应的密钥以及其细节，对上学期密码学的知识也算进行了一个回顾

最后通过真正的聊天程序对所学到的理论知识进行相应的应用，对网络编程也更加的熟练，期待自己未来更好的发展，心想事成、万事胜意、未来可期