网络安全技术

实
验
报
告

学　　院　网安学院

年　　级　大三

班　　级　信息安全

学　　号　**2011762**

姓　　名　于文明

2023 年 **5** 月 **4** 日

目录

网络安全技术.....1

一、实验目的.....1

二、实验内容.....1

（一） 保密通信协议设计.....1

（二） DES 加密算法实现.....2

（三） 通信交互.....2

三、 实验步骤及实验结果.....2

（一） RSA 加密实现.....3

（二） DES 加密实现.....13

（三） Linux 通信与密钥分配.....13

（四） 实验结果.....17

四、 实验遇到的问题及其解决方法.....19

五、 实验结论.....19

一、实验目的

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

（一） 保密通信协议设计

•首先，服务端生成 RSA 密码的公钥与私钥，并将私钥通过 socket 传送到客户端。

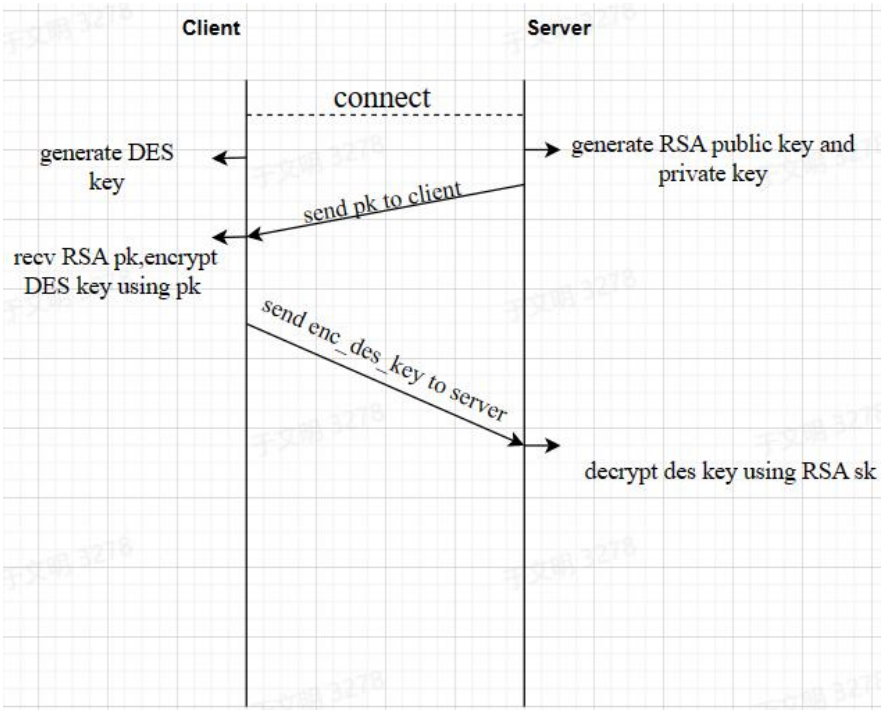

图 1.保密通信协议密钥交互流程图

• 客户端首先调用函数 genDesKey()生成随机 DES 密钥，然后获得服务端提供的 RSA 公钥，并使用该公钥加密 DES 密钥，然后将加密后 DES 密钥发回给服务端，从而实现 DES 密钥可靠共享。

（二） DES 加密算法实现

这一部分内容可参考上一次实验报告

（三） 通信交互

实现 Linux 系统的全双工通信，需要实现多线程。

三、实验步骤及实验结果

文件目录

```
D:.
|   tree.txt
|
├─.vscode
|       settings.json
|
├─client
|   |   CMakeLists.txt ····· cmake 文件
|   |   COPYRIGHT ····· 版权文件
|   |   README.md ····· README
|   |
|   ├─.vscode
|   |       c_cpp_properties.json
|   |       launch.json
|   |       settings.json
|   |
|   ├─build
|   ├─doc
|   |       client_helper.txt ····· 帮助文档
|   |
|   ├─include
|   |       bigint.h ····· 大整数类头文件
|   |       chat.h ····· 聊天实现头文件
|   |       conf.h ····· 基本配置参数
|   |       myDES.h ····· DES 头文件
|   |       myRSA.h····· RSA 头文件
|   |       test.h ····· 测试函数
|   |       tool.h····· 工具函数
```

```
│   │
│   └─src
│           bigint.cpp ····· 大整数类函数实现
│           chat.cpp ····· 聊天程序
│           clientmain.cpp ····· main 函数
│           CMakeLists.txt
│           conf.cpp ····· 全局变量
│           myDES.cpp ····· DES 类成员函数
│           myRSA.cpp····· RSA 类成员函数
│           test.cpp
│           tool.cpp
│
├─doc  相关文档
│       实验二（第 4 章  使用 RSA 算法自动分配密钥的聊天程序 ）.pdf
│       实验报告模板-网络安全 (1).doc
│
└─server
    │  CMakeLists.txt
    │  COPYRIGHT
    │  README.md
```

（一） RSA 加密实现

1. 大整数类

因为 RSA 加密算法的密钥通常为 512bits 或 1024bits，因此需要实现相关的大整数类。

（1）大整数类结构体

大整数的成员有 data[64] 用来存大整数，bool pn 标识正负号，int len 表示数据长度。Clear()用来大数清零，GetPN 判断大数的正负，GetLength() 返回大数的长度，Random() 用来产生大随机数，display() 和 Output() 用来输出 16 进制大整数，IsOdd() 判断奇偶性。此外还重载了一系列运算符

Table_1 . BigInt class

```
class BigInt {
    friend BigInt operator+ (const BigInt&, const BigInt&);
    friend BigInt operator- (const BigInt&, const BigInt&);
    friend BigInt operator- (const BigInt&, const int&);
    friend BigInt operator* (const BigInt&, const BigInt&);
    friend BigInt operator* (const BigInt&, const unsigned int&);
    friend BigInt operator% (const BigInt&, const BigInt&);
    friend BigInt operator/ (const BigInt&, const BigInt&);
    friend bool operator< (const BigInt&, const BigInt&);
    friend bool operator> (const BigInt&, const BigInt&);
    friend bool operator<= (const BigInt&, const int&);
    friend bool operator== (const BigInt&, const BigInt&);
    friend bool operator== (const BigInt&, const int&);
    friend ostream& operator<< (ostream&, const BigInt&);
    friend BigInt Power(const BigInt&, const BigInt&, const
BigInt&);//计算幂
    friend void pTabScr(BigInt& n);

public:
    BigInt();
    BigInt(const int&);
    BigInt(const BigInt&);

    void operator= (const BigInt&);
    void operator= (const int& a) { Clear(); data[0] = a; }
    void operator>> (const int&);

    inline int GetLength() const;     //返回大数的长度
    bool TestSign() { return pn; }    //判断大数的正负
    void Clear();    //大数清 0
    //void Random(); //随机产生一个大数
    void Random(bool small);    //随机产生一个稍小的大数
    void display() const;
    void Output(ostream& out) const;
```

```
    bool IsOdd() const { return (data[0] & 1); }    //判断大数奇偶性

public:
    unsigned int data[MAXSIZE];
    bool pn;
        int len;
};
```

（2）运算符重载

这里以比较复杂的运算符重载作为示例：

A.大数乘法，采用竖式乘法的算法

```
BigInt operator* (const BigInt& a, const BigInt& b)
{
    //last 存放竖式上一行的积,temp 存放当前行的积
    BigInt result, last, tmp;
    //sum 存放当前行带进位的积
    unsigned __int64 sum;
    //存放进位
    unsigned int c;

    //进行竖式乘
    for (int i = 0; i < b.GetLength(); i++)
    {
        c = 0;
        //B 的每一位与 A 相乘
        for (int j = 0; j < a.GetLength() + 1; j++)
        {
            sum = ((unsigned __int64)a.data[j]) * (b.data[i]) + c;
            if ((i + j) < MAXSIZE)
                tmp.data[i + j] = (unsigned int)sum;
            c = (sum >> 32);
```

```
        }
        result = (tmp + last);
        last = result;
        tmp.Clear();
    }

    //判断积的符号
    if (a.pn == b.pn)
        result.pn = true;
    else
        result.pn = false;
    result.len = result.GetLength();
    return result;
}
```

B.大数除法，采用试商除法,采用**二分查找**法优化

```
BigInt operator/ (const BigInt& a, const BigInt& b)
{
    //mul 为当前试商,low,high 为二分查找试商时所用的标志
    unsigned int mul, low, high;
    //sub 为除数与当前试商的积,subsequent 为除数与下一试商的
积
    //dividend 存放临时被除数
    BigInt dividend, quotient, sub, subsequent;
    int lengtha = a.GetLength(), lengthb = b.GetLength();
    //如果被除数小于除数,直接返回 0
    if (a < b)
    {
        if (a.pn == b.pn)
            quotient.pn = true;
        else
            quotient.pn = false;
        return quotient;
    }
```

```
    //把被除数按除数的长度从高位截位
    int i;
    for (i = 0; i < lengthb; i++)
        dividend.data[i] = a.data[lengtha - lengthb + i];
    for (i = lengtha - lengthb; i >= 0; i--)
    {
        //如果被除数小于除数,再往后补位
        if (dividend < b)
        {
            for (int j = lengthb; j > 0; j--)
                dividend.data[j] = dividend.data[j - 1];
            dividend.data[0] = a.data[i - 1];
            continue;
        }
        low = 0;
        high = 0xffffffff;
        //二分查找法查找试商
        while (low < high)
        {
            mul = (((unsigned __int64)high) + low) / 2;
            sub = (b * mul);
            subsequent = (b * (mul + 1));
            if (((sub < dividend) && (subsequent > dividend)) ||
(sub == dividend))
                break;
            if (subsequent == dividend)
            {
                mul++;
                sub = subsequent;
                break;
            }
            if ((sub < dividend) && (subsequent < dividend))
            {
                low = mul;
```

```
                continue;
            }
            if ((sub > dividend) && (subsequent > dividend))
            {
                high = mul;
                continue;
            }
        }
        //试商结果保存到商中去
        quotient.data[i] = mul;
        //临时被除数变为被除数与试商积的差
        dividend = dividend - sub;
        //临时被除数往后补位
        if ((i - 1) >= 0)
        {
            for (int j = lengthb; j > 0; j--)
                dividend.data[j] = dividend.data[j - 1];
            dividend.data[0] = a.data[i - 1];
        }
    }
    //判断商的符号
    if (a.pn == b.pn)
        quotient.pn = true;
    else
        quotient.pn = false;
    quotient.len = quotient.GetLength();
    return quotient;
}
```

（3）快速幂

快速幂运算——计算 n 的 p 次幂模 m,利用 Montgomery 算法

```
BigInt Power(const BigInt& n, const BigInt& p, const BigInt& m)
{
```

```
    BigInt temp = p;
    BigInt base = n % m;
    BigInt result(1);

    //检测指数 p 的二进制形式的每一位
    while (!(temp <= 1))
    {
        //如果该位为 1，则表示该位需要参与模运算
        if (temp.IsOdd())
        {
            result = (result * base) % m;
        }
        base = (base * base) % m;
        temp >> 1;
    }
    return (base * result) % m;
}
```

2.生成大质数

生成大素数主要由四个过程：生成大随机数，奇偶检测，素数表筛，Rabin-Miller 检测，

（1） 生成大随机数

因为库函数 rand() 最大只能产生 0X7FFF 的数, 为了能产生 32 位的随机数, 需要 3 次 rand() 操作, 即 rand()«17 +rand()«2+rand(), 同时根据需要生成的大整数的 size, 在外层添加循环 For i->1:size:

data[i]=rand()«17 +rand()«2

（2） 奇偶检测与素数表筛查

定义 IsPrime() 函数判断是否是素数，然后从小到大生成 1000 个素数填充进素数表 prime[1000]。使用 Random() 生成一个大随机数，进行奇偶性检测，判断是否为奇数。使用 prime[1000] 进行筛查，使之不能被其中的任一素数整除.

```
void pTabScr(BigInt& n)
{
```

```
    int i = 0;
    BigInt divisor;
    const int length = sizeof(prime) / sizeof(int);

    while (i != length)
    {
        n.Random(false);
        while (!n.IsOdd())
            n.Random(false);

        i = 0;
        for (; i < length; i++)
        {
            divisor = prime[i];
            if ((n % divisor) == 0)
                break;
        }
    }
}
```

（3） Rabin-Miller 检测

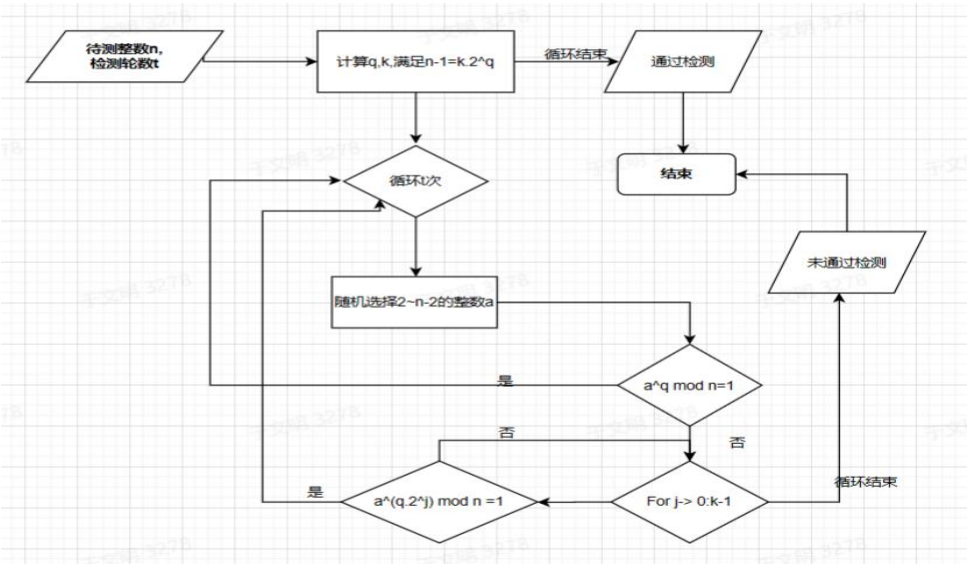

图 2.Rabin-Miller 检测流程图

Miller Rabin 算法的依据是费马小定理:ap−1 ≡ 1(modP)
假设需要判断的数是 p
我们把 p−1 分解为 2k∗ t
当 p 是素数，有 a(2k∗ t) ≡ 1(modp)
然后随机选择一个数 a，计算出 at(modp)
让其不断的自乘，同时结合二次探测定理进行判断
如果我们自乘后的数 (mod p)=1，但是之前的数 (mod p)≠±1
那么这个数就是合数 (违背了二次探测定理)
这样乘 k 次，最后得到的数就是 ap−1
那么如果最后计算出的数不为 1，这个数也是合数 (费马小定理

```
bool RabinMiller(const BigInt& n)
{
    BigInt r, a, y;
    unsigned int s, j;
    r = n - 1;
    s = 0;

    while (!r.IsOdd())
    {
        s++;
        r >> 1;
    }

    //随机产生一个小于 N-1 的检测数 a
    a.Random(true);

    //y = a 的 r 次幂模 n
    y = Power(a, r, n);

    //检测 J=2 至 J<S 轮
    if ((!(y == 1)) && (!(y == (n - 1))))
    {
        j = 1;
```

```
        while ((j <= s - 1) && (!(y == (n - 1))))
        {
            y = (y * y) % n;
            if (y == 1)
                return false;
            j++;
        }
        if (!(y == (n - 1)))
            return false;
    }
    return true;
}
```

4.RSA 加解密

（1）公私钥

选择两个不同的大素数 p 和 q，n 是二者的乘积，即 $\phi(n)=pq$，使为欧拉函数。随机选取正整数 e，使其满足，即 e 和 n 互素，则将（n，e）作为公钥。

求出正数 d，使其满足 $e*d=1 \bmod \phi(n)$，则将（n，d）作为私钥。

（2）加解密

对于明文 m，由 $c \equiv m^e \bmod n$，得到密文 c

对于密文 c，由 $m \equiv c^d \bmod n$，得到明文 m

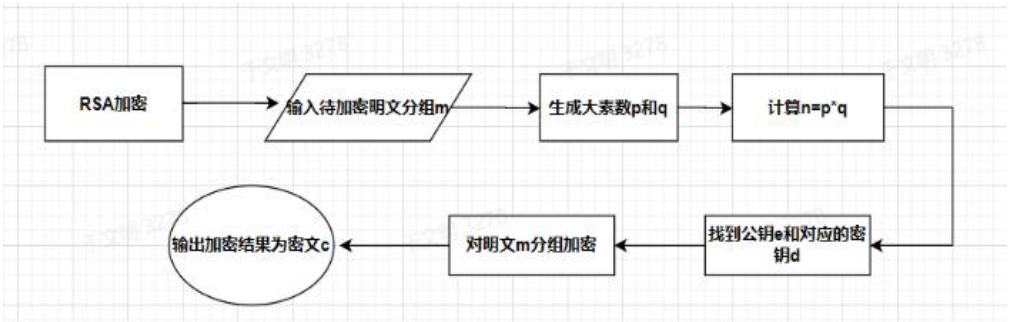

图 3.RSA 加密

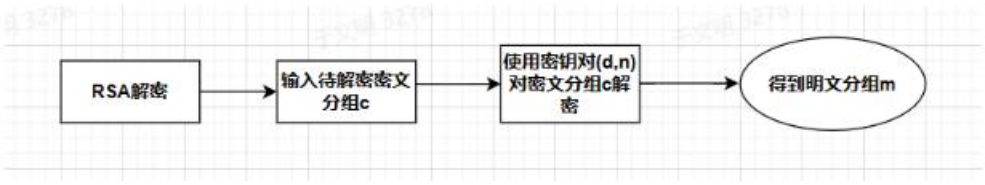

图 4.RSA 解密

（二） DES 加密实现

该部分主体内容请见上一次实验报告，这里只给出加解密的实验流程图

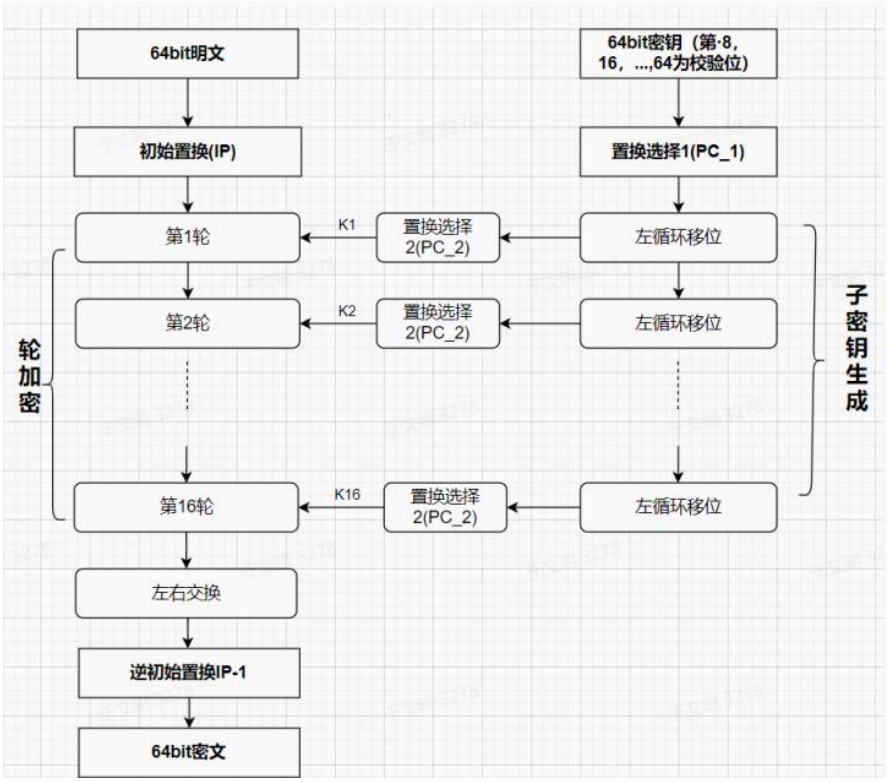

图 4.DES 加密流程图

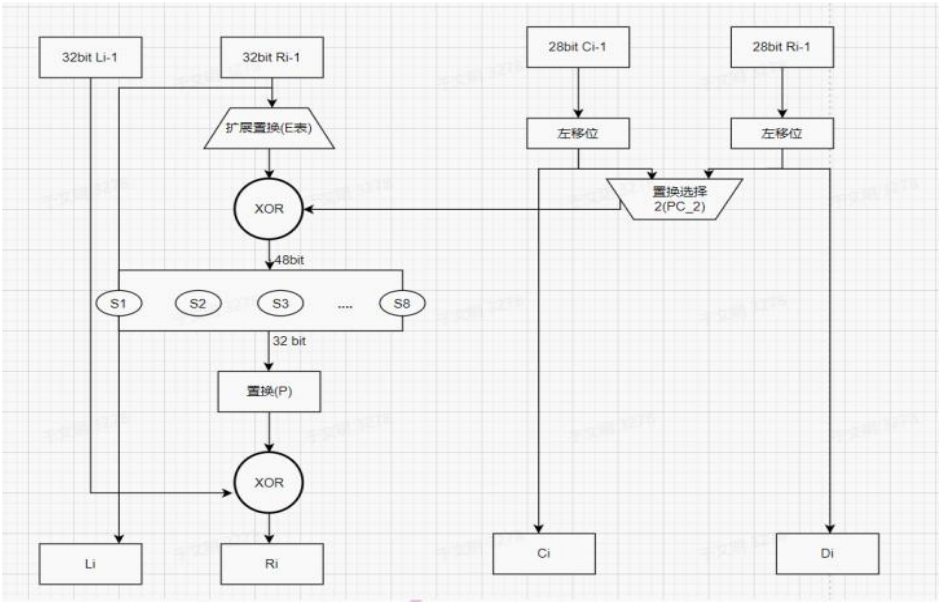

图 5.DES 加密轮结构

（三） Linux 通信与密钥分配

1. 双向通信实现

在 client 和 server 端使用 secretchat()函数实现通信的接收与发送。在 secretchat 函数中使用了 fork 函数创建子线程，父线程用来接收信息，实现密文的 DES 解密;子线程用来发送信息，实现明文的 DES 加密.

```
void SecretChat(int nSock,char *pRemoteName,BigInt des_k)
{
	DES cDes;
	for(int i=0;i<2;i++)
	{
		for(int j=0;j<32;j++)
		{
			key[i * 2 + j] = (des_k.data[i] >> (31 - j)) & 1;
		}
	}
	pid_t nPid;
	nPid=fork();
	if(nPid!=0)
	{
		//recv thread
		extern char recvBuf[BUFFERSIZE];
		while(1)
		{
			bzero(recvBuf,BUFFERSIZE);
			read(nSock,recvBuf,BUFFERSIZE);
			string cipher=recvBuf;
			printf("接收的密文消息为:\n");
			for(auto c:cipher)
			{
				printf("0x%x ",c);
			}
			printf("\n");
			string plain=cDes.decrypt(cipher,key);
			if(plain=="quit")
```

```
                {
                    //退出
                    printf("Quit\n");
                    return;
                }
                std::cout<<"[recv from
<"<<pRemoteName<<">]:"<<plain<<endl;
            }
        }
        else
        {
            //send thread
            while(1){
                //send thread
                string plain;
                //printf("请输入消息:\n");
                while(plain=="")
                {
                    getline(cin,plain);
                }
                string cipher=cDes.encrypt(plain,key);
                printf("发送的消息密文为:\n");
                for(auto c:cipher)
                {
                    printf("0x%x ",c);
                }
                printf("\n");

if(write(nClientSocket,cipher.c_str(),sizeof(cipher))==-1)
                {
                    perror("send msg\n");
                    exit(0);
                }
                else
```

```
                {
                    printf("消息已发送\n");
                }
            }
        }
}
```

2. 多人通信实现

在 server 端，每当 accept 一个新的 client 时，就调用 fork 函数新建一个线程

```
 while(true)
     {
         nClientSocket = accept(nServSocket,NULL,NULL);
         printf("server: got connection from %s,
port %d,socket %d\n",inet_ntoa(sRemoteAddr.sin_addr),ntohs(sRemo
teAddr.sin_port),nClientSocket);
         printf("Key Allocation ...\n");
         BigInt des_k;
         KeyInit(des_k);
         //fork
         pid_t npid;
         npid=fork();
         if(npid<0)
         {
             printf("fork error");
             return;
         }
         else if(npid==0)
         {
...
```

（四） 实验结果

1.Socket 初始化

客户端，输入服务器 IP,等待连接

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab2/client/build/bin$ ./clie
nt
请输入服务器IP: 127.0.0.2
Socket 初始化完成，开始连接...
Connect to Server!
```

服务端，初始化 socket,监听

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab2/server/build/bin$ ./serv
er
Socket 初始化完毕,开始监听...
server: got connection from 0.0.0.0, port 0,socket 4
```

2.密钥分配

服务端，生成随机大质数，Rabin-Miller 检测

```
Key Allocation ...
server启动,分配RSA公私钥
生成RSA公钥密钥
7919
开始生成大素数p
产生待测大奇数:
ptabscr end!
2B739BD3 DED0A1F2 3CB9A1EF 8727385C 9D67F6B2 EE98335D B1B54E0F 20466AF5
D91A143A 8D4676D1 13D108A0 BB0D4EAE F181907D 054AE4CD C9953047 D4DD6D81
RABINMILLER测试失败
```

检测通过，生成公私钥,将公钥 send to client

```
9CC4239F DA69F48B CCD79C6E F164B95F 15497840 6AE24C12 EC6D4C83 5D79EE54
B4B642DA AD5CFADB 876D4912 7CD91584 C1A25E5F 5A7CDFDF 5FC9864C 7FD8BDE1
RSA公钥e为:C76BA516 48778698 748E6BD9 E7CC42BB 892A0F70 E53C6799 DC72F37C F71BC1
60
C61AA4A2 6183FA7E D36E0366 1D97E573 339B0FF8 C18C608A 0D2DC911 39BB8041
RSA公有钥n为:4862B543 ECC4FBDB D179A4B5 5ABBD8AE F196620E 4EAE7A35 B4A7B2DF 98D2
D5B9 74447C28 3931056A 9249F831 23E6582A 81E44143 B1B9A7D5 BE4A5A26 604F7DF7
1F156FFF 84927949 B4F3AC04 8B702228 6756251F F8DC7F91 DE25B6C0 FC45A69A F79C5E3E
 C8147E84 D68FF983 26C6B13D 9D529224 172421A5 8ADA4E6C 8FA760BD
RSA私钥d为:1F73A002 44CD5321 E7E28ADB D7C6EA63 35D16CAB BC87053E 0D34BACD 1FFEF6
B5 350BE9C7 8A594975 222BF1AE 69369410 6EED29C5 A196BF14 1A0902B2 3588941A
06B0B7CC 037A8798 D0EF968F 64898E4C F3E95DF4 C7FEF267 EDDE94DD EA3F5E3D E697728E
 D31E892D 10A4D6F8 8C276FDE EE2D959F 4F7C4AD3 B38384B1 8AA14141
```

客户端，输入随机数种子，生成 DES 密钥，并用接收的 RSA 公钥加密

```
A9755FAE
EAF2F95D
send des_k success
```

服务端，使用私钥解密

```
recv DES key,begin RSA decrypt...
DES KEY:A9755FAE
EAF2F95D
```

3.加密通信过程

（1）C-S 双向通信

client -> server

client(明文加密)

```
你好，我是Metetor.
发送的消息密文为:
0x44 0x17 0x19 0xffffff86 0x2 0xfffffaa 0xffffff92 0x3e 0xffffff8e 0x4b 0x28 0x
47 0xffffffa3 0x8 0x60 0x31 0xffffffcf 0x39 0x29 0xffffffd6 0x3a 0xffffffca 0x55
 0x29
消息已发送
```

server (密文解密)

```
接收的密文消息为:
 0x44 0x17 0x19 0xffffff86 0x2 0xfffffaa 0xffffff92 0x3e 0xffffff8e 0x4b 0x28
 47 0xffffffa3 0x8 0x60 0x31 0xffffffcf 0x39 0x29 0xffffffd6 0x3a 0xffffffca 0
  0x29
[recv from <0.0.0.0>]:你好，我是Metetor.
```

Server->Client

Server**(明文加密)**

```
hello,Metetor.
发送的消息密文为:
0xfffffffdb 0xfffffffaa 0xffffffe7 0x19 0xffffffe0 0x78 0x6d 0x7a 0xffffffa2 0x64
0xffffffffd 0x4a 0x8 0xffffffcf 0xc 0xffffffbe
消息已发送
```

Client **(密文解密)**

```
接收的密文消息为:
0xfffffffdb 0xfffffffaa 0xffffffe7 0x19 0xffffffe0 0x78 0x6d 0x7a 0xffffffa2 0x64
0xffffffffd 0x4a 0x8 0xffffffcf 0xc 0xffffffbe
[recv from <127.0.0.2>]:hello,Metetor.
```

(2) 多人聊天

Client01

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab2/client/build/bin$ ./client
请输入服务器IP: 127.0.0.2
Socket 初始化完成，开始连接...
Connect to Server!
Then allocate the keys...
recv RSA public key from server
generate DES key,please input random seed
2500
711BF76B
B502C028
send des_k success
(-^^-), this is Alice.
发送的消息密文为:
0x67 0x20 0xffffffdd 0xffffff9b 0x19 0xffffff9c 0xffffffff 0xffffffc7 0xffffff86 0x13 0x7 0x3
1 0xfffffffc 0x5f 0xffffffb4 0x17 0xffffffac 0xffffffd2 0xffffffae 0xffffffb8 0x2a 0xffffffb3
 0xffffffbd 0x45
消息已发送
```

Client02

```
metetor@metetor-virtual-machine:/mnt/hgfs/网络安全/Lab2/client/build/bin$ ./clie
nt
请输入服务器IP: 127.0.0.2
Socket 初始化完成，开始连接...
Connect to Server!
Then allocate the keys...
recv RSA public key from server
generate DES key,please input random seed
2140
73AC338D
07824431
send des_k success
你好，我是John.
发送的消息密文为:
0xfffffffc9 0x23 0xffffff8a 0xffffff9b 0xffffff87 0x18 0xfffffc7 0xffffffe0 0x19
 0x60 0xffffffbe 0xfffffffd4 0xffffffa3 0xfffffffc8 0x40 0x5e 0x39 0x55 0x7c 0x40
0xfffffffda 0x4 0x2 0xffffff97
消息已发送
```

Server

```
server: got connection from 0.0.0.0, port 0,socket 4
Key Allocation ...
recv DES key,begin RSA decrypt...
DES KEY:73AC338D
07824431
接收的密文消息为:
0xfffffffc9 0x23 0xffffff8a 0xffffff9b 0xffffff87 0x18 0xfffffc7 0xffffffe0 0x19
 0x60 0xffffffbe 0xfffffffd4 0xffffffa3 0xfffffffc8 0x40 0x5e 0x39 0x55 0x7c 0x40
0xfffffffda 0x4 0x2 0xffffff97
[recv from <0.0.0.0>]:你好，我是John.
server: got connection from 0.0.0.0, port 0,socket 4
Key Allocation ...
recv DES key,begin RSA decrypt...
DES KEY:711BF76B
B502C028
接收的密文消息为:
0x67 0x20 0xffffffdd 0xffffff9b 0x19 0xffffff9c 0xffffffff 0xfffffffc7 0xffffff86
 0x13 0x7 0x31 0xfffffffc 0x5f 0xffffffb4 0x17 0xffffffac 0xffffffd2 0xffffffae
0xffffffb8 0x2a 0xffffffb3 0xffffffbd 0x45
[recv from <0.0.0.0>]:(-^^-this is Alice.
```

四、实验遇到的问题及其解决方法

1. 实现双向通信时，如何处理 server 端的收发。
   解决方法：使用 fork 函数新建线程，专门用来处理发送消息。
2. RSA 加密过程随机数生成
   为了简化实验过程，本实验还是使用了 random()函数，通过移位实现位数要求

五、实验结论

本次实验相关资源已经上传至 https://github.com/Metetor/.

在实现 RSA 加密 DES 会话密钥的 Linux 加密通信程序的过程中，我深

刻认识到了加密通信的重要性。在网络通信中，很多敏感信息需要进行加密保护，以防止被窃听、篡改或伪造。通过本次实验，我了解了 RSA 和 DES 加密算法的基本原理和实现方法，学会了使用数字证书、数字签名、身份验证等安全机制，保障通信的安全性和完整性。

同时，在实现过程中，我也遇到了一些问题，例如密钥生成不足、加解密算法实现问题等，通过查阅文献和资料，我逐步解决了这些问题。在解决问题的过程中，我不断地学习和提高自己的技能和能力，锻炼了自己的实践能力和创新思维能力。

总之，本次实验让我更深入地了解了加密通信的原理和实现方法，也让我认识到了安全技术的重要性和必要性。在今后的工作和学习中，我将继续深入学习和探索相关的安全技术和应用，为保护网络安全和信息安全做出自己的贡献。