



南开大学
Nankai University

南 开 大 学

网 安 学 院

网络安全技术实验报告

第二次作业

沙璇 1911562

年级：2019 级

专业：信息安全

提交日期：2022/4/30

2022 年 4 月 30 日

摘要

基于 RSA 算法自动分配密钥的加密聊天程序

关键字: DES , TCP, RSA

目录

一、 实验目的	1
二、 实验要求	1
三、 实验内容	1
(一) RSA 相关知识	1
(二) 总体流程	3
(三) 具体代码	4
1. 主模块	4
2. DES 信息加密模块	5
3. RSA 密钥分配模块	5
4. TCP 通信模块	9
四、 实验结果	11
五、 总结	12

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：

加深对 RSA 算法基本工作原理的理解。

掌握基于 RSA 算法的保密通信系统的基本设计方法。

掌握在 Linux 操作系统实现 RSA 算法的基本编程方法。

了解 Linux 操作系统异步 IO 接口的基本工作原理。

二、 实验要求

本章编程训练的要求如下。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。

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。

要求程序实现全双工通信，并且加密过程对用户完全透明。

三、 实验内容

本次实验使用 C 语言编写，在 kali 中运行。

聊天程序分为四个模块：主模块、TCP 通信模块、DES 信息加密模块、RSA 密钥分配模块。

主模块：即控制端，用于根据不同的情况调用不同的功能以实现不同的需求。

TCP 通信模块：即通信模块，负责聊天程序的消息传递、接收等通信需求。

DES 信息加密模块：即消息安全模块，负责聊天程序中消息的安全传输。

RSA 密钥分配模块：即密钥安全模块，负责聊天程序中消息加密密钥的安全传输。

本次实验涉及到的源文件有：Chat.c、DESSecurity.c、DESTables.c、TCPCommun.c 及其相应的头文件。完整代码见附件。

1. Chat.c：该文件是程序的主文件，用于完成聊天功能的选择和调用。
2. DESSecurity.c：DES 消息加密模块，负责加密和解密消息字符串。对外提供两个函数，分别是：加密函数 DESEncry()，解密函数 DESDecry()。
3. DESTables.c：DES 表
4. TCPCommun.c：TCP 通信模块，负责聊天程序的连接及消息发送和接收。

(一) RSA 相关知识

1. 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、Diffie-Hellman Rabin

EIGamal

2. 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。
- (6) 公钥密码体系的基本结构如下图1所示

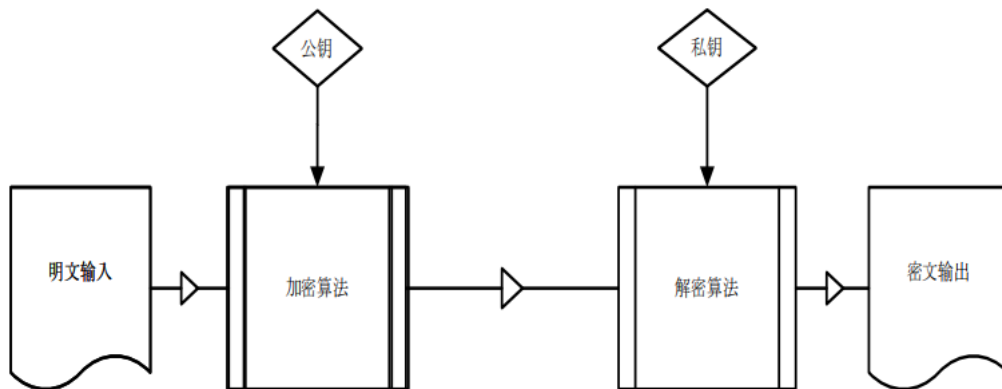


图 4-1 公钥密码体系原理示意图

图 1: caption

3. 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。只要其密钥的长度足够长，用 RSA 加密的信息的安全性就可以保证。RSA 密码体系使用了乘方运算。明文以分组为单位进行加密，每个分组的二进制值均小于 n 。

对于明文分组 M 和密文分组 C ，加密和解密的过程如下图2所示

$$(1) \quad C = M^e \% n$$

$$(2) \quad M = C^d \% n = (M^e)^d \% n = M^{d \times e} \% n = M$$

其中 n 、 d 、 e 为三个整数，且 $d \times e \equiv 1 \% \phi(n)$ 。收发双方共享 n ，接受一方已知 d ，发送一方已知 e ，则此算法的公钥为 $\{e, n\}$ ，私钥是 $\{d, n\}$ 。

理解 RSA 基本工作原理需要数论的基础知识：

(1) 同余：两个整数 a ， b ，若它们除以整数 m 所得的余数相等，则称 a ， b 对于模 m 同余，记作 $a \equiv b \% m$ 。

(2) Euler 函数： $\phi(n)$ 是指所有小于 n 的正整数里，和 n 互质的整数的个数。其中 n 是一个正整数。假设整数 n 可以按照质因数分解写成如下形式： $n = P_1^{a_1} \times P_2^{a_2} \times \dots \times P_m^{a_m}$ ；其中， $P_1, P_2 \dots P_m$ 为质数。则 $\phi(n) = n \times \left(1 - \frac{1}{P_1}\right) \times \left(1 - \frac{1}{P_2}\right) \times \dots \times \left(1 - \frac{1}{P_m}\right)$ 。

图 2: caption

4. RSA 密码体系公钥私钥生成方式

RSA 密码体系公钥私钥生成方式如下。任意选取两个质数， p 和 q ，然后，设 $n = p \times q$ ；函数 $\phi(n)$ 为 Euler 函数，返回小于 n 且与 n 互质的正整数个数；选择一个任意正整数 e ，使其与 $\phi(n)$ 互质且小于 $\phi(n)$ ，公钥 $\{e, n\}$ 已经确定；最后确定 d ，使得 $d \times e \equiv 1 \pmod{\phi(n)}$ ，即 $(d \times e - 1) \pmod{\phi(n)} = 0$ ，至此，私钥 $\{d, n\}$ 也被确定。

图 3: caption

(二) 总体流程

程序执行过程如下：

在客户端与服务器建立连接后，客户端首先生成一个随机的 DES 密钥，在第二章的程序里要求密钥长度为 64 位，所以使用长度为 8 的字符串充当密钥；

同时，服务端生成一个随机的 RSA 公钥/私钥对，并将 RSA 公钥通过刚刚建立起来的 TCP 连接发送到客户端主机；

客户端主机在收到该 RSA 公钥后，使用公钥加密自己生成的 DES 密钥，并将加密后的结果发送给服务器端；

服务器端使用自己保留的私钥解密客户端发过来的 DES 密钥，最后双方使用该密钥进行保密通信。

程序的流程图如下图4所示

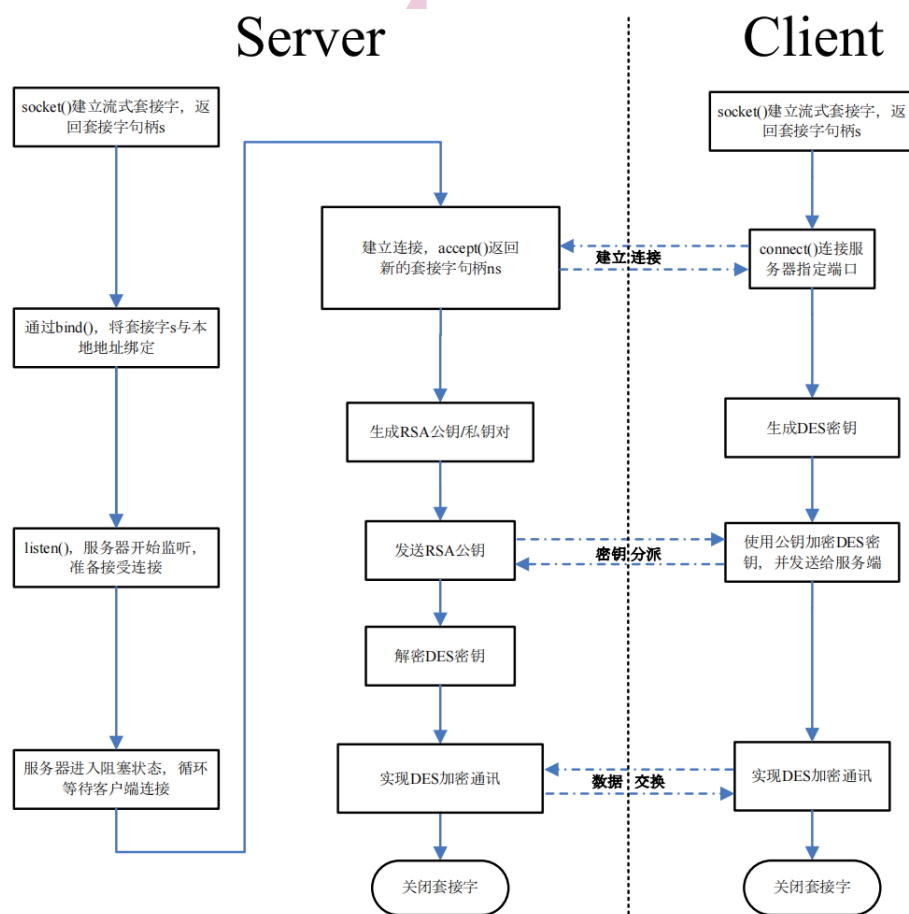


图 4: caption

(三) 具体代码

聊天程序分为四个模块：主模块、DES 信息加密模块、RSA 密钥分配模块、TCP 通信模块

主模块：即控制端，用于根据不同的情况调用不同的功能以实现不同的需求。

DES 信息加密模块：即消息安全模块，负责聊天程序中消息的安全传输。

RSA 密钥分配模块：即密钥安全模块，负责聊天程序中消息加密密钥的安全传输。

TCP 通信模块：即通信模块，负责聊天程序的消息传递、接收等通信需求。

1. 主模块

主模块主要进行根据不同的情况调用不同的功能以实现不同的需求。以下是进行 server 端和 client 端选择的具体代码。

```
des
1 //选择执行的身份: sever or client
2     id = ChooseCorS();
3
4     //启动服务
5     switch(id)
6     {
7         case 'c':
8         {
9             //获取服务器地址
10            printf("Please input the server address:\n");
11            scanf("%s", serveraddr);
12            if(strlen(serveraddr) <= 0 || strlen(serveraddr) >
13                16)
14            {
15                printf("sorry,the server address input error!");
16            }
17            else
18            {
19                //建立连接
20                ClientToServer(serveraddr);
21            }
22            break;
23        }
24        case 's':
25        {
26            //监听连接
27            ServerToClient();
28
29            break;
30        }
31        default:
32        {
33            printf("sorry,the Id has an error!");
```

```
34         break;
35     }
36 }
```

2. DES 信息加密模块

同实验一

3. RSA 密钥分配模块

密钥安全模块，负责聊天程序中消息加解密的安全传输。

des

```
1 void RSAGetParam()
2 {
3     long t;
4
5     //随机生成两个素数
6     rsa.p = RandomPrime(16);
7     rsa.q = RandomPrime(16);
8
9     //计算模数及相应的f
10    rsa.n = rsa.p * rsa.q;
11    rsa.f = (rsa.p - 1)*(rsa.q - 1);
12
13    //生成公钥中的e
14    do
15    {
16        rsa.e = rand()%65536;
17        rsa.e |= 1;
18    } while(Gcd(rsa.e, rsa.f) != 1);
19
20    //生成私钥中的d
21    rsa.d = Enclid(rsa.e, rsa.f);
22
23    //计算n结尾连续的比特1
24    rsa.s = 0;
25    t = rsa.n >> 1;
26    while(t)
27    {
28        rsa.s++;
29        t >>= 1;
30    }
31 }
```

其中 RSA 核心函数如下

des

```
1 //模乘运算
```

```
2 unsigned long MulMod(unsigned long a, unsigned long b, unsigned long n)
3 {
4     return (a*b) %n;
5 }
6
7 // 模幂运算
8 unsigned long PowMod(unsigned long base, unsigned long pow, unsigned long n)
9 {
10     unsigned long a = base, b = pow, c = 1;
11     while(b)
12     {
13         while(!(b & 1))
14         {
15             b >>= 1;
16             a = MulMod(a, a, n);
17         }
18         b--;
19         c = MulMod(a, c, n);
20     }
21
22     return c;
23 }
24
25 // 拉宾——米勒测试, 判别是否为质数
26 long RabinMillerKn1(unsigned long n)
27 {
28     unsigned long a, q, k, v;
29     unsigned int z;
30     int i, w;
31
32     // 计算出q,k
33     q = n - 1;
34     k = 0;
35     while(!(q & 1))
36     {
37         ++k;
38         q >>= 1;
39     }
40
41     // 随机获取一个数
42     a = 2 + rand()%(n - 3);
43     v = PowMod(a, q, n);
44     if(v == 1)
45     {
46         return 1;
47     }
48
49     // 循环检验
```



```
50     for(i = 0; i < k; i++)
51     {
52         z = 1;
53         for(w = 0; w < i; w++)
54         {
55             z *= 2;
56         }
57         if(PowMod(a, z * q, n) == n-1)
58         {
59             return 1;
60         }
61     }
62
63     return 0;
64 }
65
66 //重复拉宾——米勒测试
67 long RabinMiller(unsigned long n, unsigned long loop)
68 {
69     int i;
70
71     for(i = 0; i < loop; i++)
72     {
73         if(!RabinMillerKnl(n))
74         {
75             return 0;
76         }
77     }
78
79     return 1;
80 }
81
82 //质数生成函数
83 unsigned long RandomPrime(char bits)
84 {
85     unsigned long base;
86
87     do
88     {
89         base = (unsigned long)1 << (bits - 1); //保证最高位
90         base += rand()%base; //加上一个随机数
91         base |= 1; //保证最低位为1
92     } while(!RabinMiller(base, 30)); //进
93     行拉宾——米勒测试30次
```

```
94     return base;
95 }
96
97 //求最大公约数
98 unsigned long Gcd(unsigned long p, unsigned long q)
99 {
100     unsigned long a = p>q?p:q;
101     unsigned long b = p<q?p:q;
102     unsigned long t;
103
104     if(p == q)
105     {
106         return p; //若两数相等，最大公约数就是本身
107     }
108     else
109     {
110         //辗转相除法
111         while(b)
112         {
113             a = a % b;
114             t = a;
115             a = b;
116             b = t;
117         }
118         return a;
119     }
120 }
121
122
123 //生成私钥中的d
124 unsigned long Enclid(unsigned long e, unsigned long t_n)
125 {
126     unsigned long max = 0xffffffffffffffff - t_n;
127     unsigned long i = 1, tmp;
128
129     while(1)
130     {
131         if(((i * t_n) + 1)%e == 0)
132         {
133             return ((i * t_n)+1) / e;
134         }
135         i++;
136         tmp = (i + 1) * t_n;
137         if(tmp > max)
138         {
139             return 0;
140         }
141     }
```

```
142  
143     return 0;  
144 }
```

4. TCP 通信模块

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；

客户端首先调用函数 GerenateDesKey() 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

des

```
1 //生成并发送DES密钥  
2 void DESAllocGener(int sock)  
3 {  
4     int i, flag;  
5     char szBuffer[BUFFERSIZE];  
6  
7     //随机生成DES密钥  
8     flag = GenerateDESKey(chatkey);  
9     if(flag)  
10    {  
11        printf("Generate DES key successful!\n");  
12    }  
13    else  
14    {  
15        printf("Generate DES key failed!\n");  
16        exit(0);  
17    }  
18  
19    //接收RSA公钥  
20    flag = recv(sock, (char*)&publickey, BUFFERSIZE, 0);  
21    if(!flag)  
22    {  
23        printf("Receive RSA public key failed!\n");  
24        exit(0);  
25    }  
26  
27    //加密DES密钥  
28    long nEncryDESKey[DESKEYLENGTH/2];  
29    unsigned short* pDesKey = (unsigned short*)chatkey;  
30    for(i = 0; i < DESKEYLENGTH/2; i++)  
31    {  
32        nEncryDESKey[i] = RSAEncry(pDesKey[i], publickey);  
33    }  
34  
35    //将加密后的DES密钥发送给服务端
```

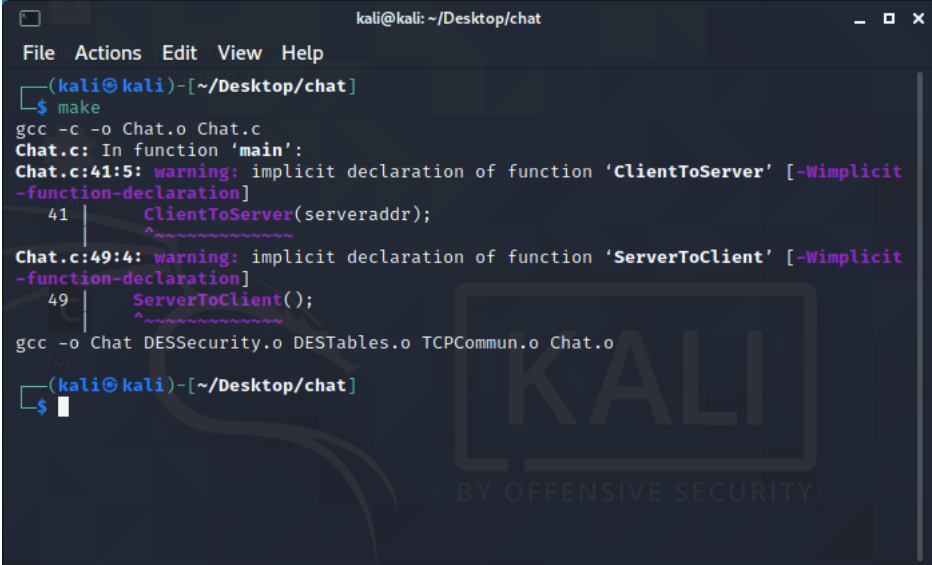
```
36     if(sizeof(long)*DESKEYLENGTH/2 != send(sock, (char*)nEncryDESKey,
37         sizeof(long)*DESKEYLENGTH/2, 0))
38     {
39         printf("Send DES key failed!\n");
40         exit(0);
41     }
42     else
43     {
44         printf("Send DES key successful!\n");
45     }
46 }
47 //生成RSA公私钥对，并解密DES密钥
48 void DESAllocRecv(int sock)
49 {
50     int i;
51
52     //生成RSA公私钥对
53     RSAGetParam();
54     publickey = GetPublicKey();
55
56     //将公钥发送给客户端
57     if(send(sock, (char*)&publickey, sizeof(publickey), 0) != sizeof(
58         publickey))
59     {
60         printf("Send RSA public key failed!\n");
61         exit(0);
62     }
63     else
64     {
65         printf("Send RSA public key successful!\n");
66     }
67
68     //接收加密的DES密钥
69     long nEncryDESKey[DESKEYLENGTH/2];
70     if(recv(sock, (char*)nEncryDESKey, DESKEYLENGTH/2 * sizeof(long), 0)
71         != DESKEYLENGTH/2 * sizeof(long))
72     {
73         printf("Receive DES key failed!\n");
74         exit(0);
75     }
76
77     //解密DES密钥
78     unsigned short* pDesKey = (unsigned short*)chatkey;
79     for(i = 0; i < DESKEYLENGTH/2; i++)
80     {
81         pDesKey[i] = RSADecry(nEncryDESKey[i]);
82     }
```

基于 RAS 算法的密钥分配增加了原有程序的安全性。攻击者只能通过监听截获 RSA 公钥和使用 RSA 公钥加密后的 DES 密钥，却无法获得对应的 RSA 私钥，故无法解密 DES 密钥，进而可以保证 DES 加密通信的安全性。

此外，由于 RSA 算法执行运算量较大，所以只使用 RSA 算法用于密钥共享，而不是直接使用其加密通信内容，以降低系统资源消耗。

四、 实验结果

在 kali 系统中本文件夹下运行 make 编译文件

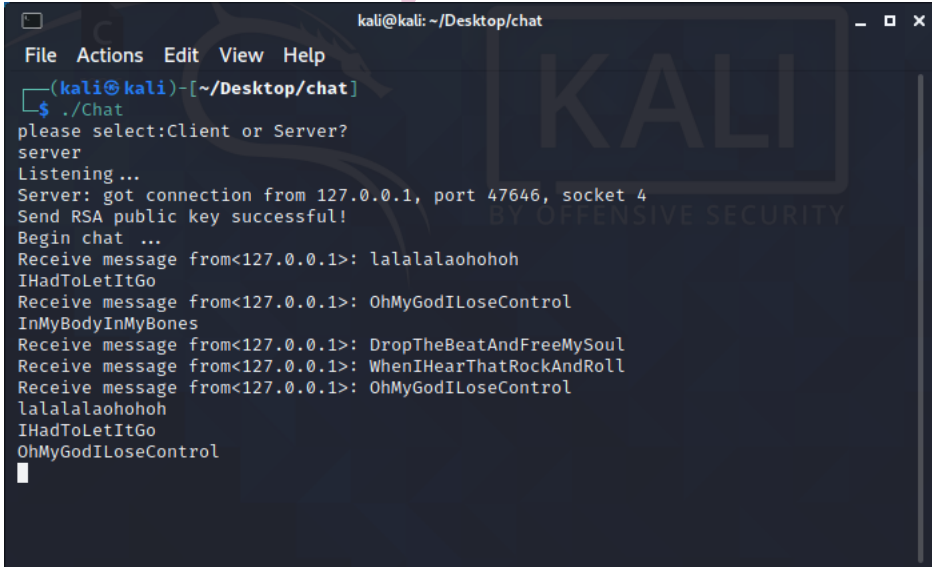


```
kali@kali: ~/Desktop/chat
File Actions Edit View Help
(kali@kali)-[~/Desktop/chat]
$ make
gcc -c -o Chat.o Chat.c
Chat.c: In function 'main':
Chat.c:41:5: warning: implicit declaration of function 'ClientToServer' [-Wimplicit-function-declaration]
    41 |     ClientToServer(serveraddr);
        |     ^~~~~~
Chat.c:49:4: warning: implicit declaration of function 'ServerToClient' [-Wimplicit-function-declaration]
    49 |     ServerToClient();
        |     ^~~~~~
gcc -o Chat DESecurity.o DESTables.o TCPCommun.o Chat.o
(kali@kali)-[~/Desktop/chat]
$
```

图 5: sever

./Chat 执行编译文件。

sever 端结果如图6所示

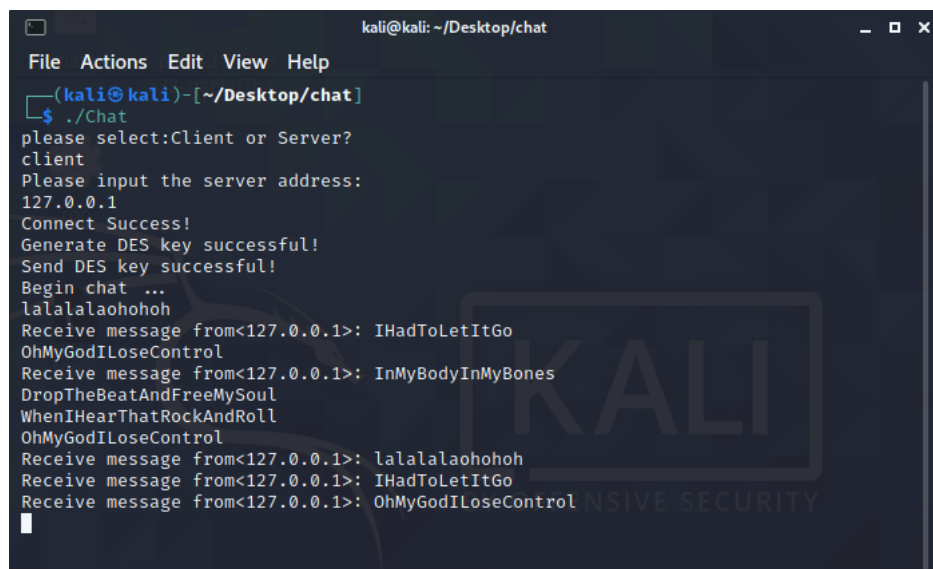


```
kali@kali: ~/Desktop/chat
File Actions Edit View Help
(kali@kali)-[~/Desktop/chat]
$ ./Chat
please select:Client or Server?
server
Listening...
Server: got connection from 127.0.0.1, port 47646, socket 4
Send RSA public key successful!
Begin chat ...
Receive message from<127.0.0.1>: lalalalaohohoh
IHadToLetItGo
Receive message from<127.0.0.1>: OhMyGodILoseControl
InMyBodyInMyBones
Receive message from<127.0.0.1>: DropTheBeatAndFreeMySoul
Receive message from<127.0.0.1>: WhenIHearThatRockAndRoll
Receive message from<127.0.0.1>: OhMyGodILoseControl
lalalalaohohoh
IHadToLetItGo
OhMyGodILoseControl

```

图 6: sever

client 端结果如图7所示



```
kali@kali: ~/Desktop/chat
File Actions Edit View Help
(kali@kali)-[~/Desktop/chat]
$ ./Chat
please select:Client or Server?
client
Please input the server address:
127.0.0.1
Connect Success!
Generate DES key successful!
Send DES key successful!
Begin chat ...
lalalalaohohoh
Receive message from<127.0.0.1>: IHadToLetItGo
OhMyGodILoseControl
Receive message from<127.0.0.1>: InMyBodyInMyBones
DropTheBeatAndFreeMySoul
WhenIHearThatRockAndRoll
OhMyGodILoseControl
Receive message from<127.0.0.1>: lalalalaohohoh
Receive message from<127.0.0.1>: IHadToLetItGo
Receive message from<127.0.0.1>: OhMyGodILoseControl
```

图 7: client

五、 总结

通过本次实验，我加深了对 RSA 算法基本工作原理的理解，掌握了基于 RSA 算法的保密通信系统的基本设计方法，掌握了在 Linux 操作系统实现 RSA 算法的基本编程方法，了解了 Linux 操作系统异步 IO 接口的基本工作。