网络安全技术 —— 基于DES加密的TCP聊天程序

学号：2013921
姓名：周延霖
专业：信息安全

一、实验目的

DES（Data Encryption Standard）算法是一种用 56 位有效密钥来加密 64 位数据的对称 分组加密算法，该算法流程清晰，已经得到了广泛的应用，算是应用密码学中较为基础的加 密算法。TCP（传输控制协议）是一种面向链接的、可靠的传输层协议。TCP 协议在网络层 IP 协议的基础上，向应用层用户进程提供可靠的、全双工的数据流传输。 本次实验目的如下:

1. 理解 DES 加解密原理
2. 理解 TCP 协议的工作原理
3. 掌握 linux 或 Windows下基于 socket 的编程方法

实验要求如下:

1. 利用 socket 编写一个 TCP 聊天程序
2. 通信内容经过 DES 加密与解密

二、实验内容

在 Linux 或 Windows 平台下，实现基于 DES 加密的 TCP 通信，具体要求如下：

1. 能够在了解 DES 算法原理的基础上，编程实现对字符串的 DES 加密解密操作
2. 能够在了解 TCP 和 Linux 平台下的 Socket 运行原理的基础上，编程实现简单的 TCP 通信，为简化编程细节，不要求实现一对多通讯
3. 将上述两部分结合到一起，编程实现通信内容事先通过 DES 加密的 TCP 聊天程序， 要求双方事先互通密钥，在发送方通过该密钥加密，然后由接收方解密，保证在网络上传输的信息的保密性

三、实验步骤

1、DES分析

> 加密过程

- 加密函数首先进行IP初始置换，按照IP表对明文进行置换
- 然后将64位数据分为高低位，然后对低位进行处理，得到的结果与高位进行异或，然后互换新的高低位位置，进行十六次迭代
- 接下来进行合并，注意合并时原本高低位位置发生调换
- 最后进行逆初始置换IP－1，具体函数如下：

```
void DES_Cry(char MesIn[8], char MesOut[8])
{
    //执行DES加密函数
    int i = 0;
```

```
    static bool MesBit[64] = { 0 };                    //信息
    static bool Temp[32] = { 0 };                          //中间变量
    static bool* MiL = &MesBit[0], * MiR = &MesBit[32]; //前后32位
    ByteToBit(MesIn, MesBit, 64);                      //char转bit到MesBit
中
    TablePermute(MesBit, MesBit, IP_Table, 64);        //IP置换，对信息进行错
位
    for (i = 0; i < 16; i++)
    {                                 //16轮迭代
        BitsCopy(MiR, Temp, 32);    //右半边复制到临时变量temp
        DES_1turn(MiR, SubKey[i]); //右半边拓展和子密钥进行异或然后压缩
        Xor(MiR, MiL, 32);          //左右异或放到右边
        BitsCopy(Temp, MiL, 32);    //一开始的右边数据放到左边
    }
    TablePermute(MesBit, MesBit, IPre_Table, 64); //IP逆置换
    BitToHex(MesBit, MesOut, 64);                 //以16进制输出密文
}
```

> 解密过程

- 解密函数与加密过程类似，只是子秘钥的调用顺序与加密相反，具体函数如下：

```
void DES_Dec(char MesIn[8], char MesOut[8])
{
    //DES解密，加密的逆过程 doublesand
    int i = 0;
    static bool MesBit[64] = { 0 };
    static bool Temp[32] = { 0 };
    static bool* MiL = &MesBit[0], * MiR = &MesBit[32];
    HexToBit(MesIn, MesBit, 64);                 //16进制密文转二进制
    TablePermute(MesBit, MesBit, IP_Table, 64); //IP置换
    for (i = 15; i >= 0; i--)
    {                                 //逆循环
        BitsCopy(MiL, Temp, 32);    //R(i-1) = Li, L15是密文的前半段，可以逆推
        DES_1turn(MiL, SubKey[i]); //Ri = L(i-1)^f(R(i-1), K(i-1))   K(i-
1)是子密钥，R15已知，根据 a = b ^ c 得 b = a^c,可以求L(i-1)
        Xor(MiL, MiR, 32);          //左右异或得到右边的原始信息放回左边
        BitsCopy(Temp, MiR, 32);    //中间变量放到右边
    }
    TablePermute(MesBit, MesBit, IPre_Table, 64); //IP逆置换
    BitToByte(MesBit, MesOut, 64);                //二进制转char
}
```

可以看出与上个学期的密码学所用的DES加密技术是一样的

2、TCP通信

当DES加密的功能实现完成后，接下来只剩实现网络通信，因为在上学期的计算机网络课实现过UDP的网络通信，所以对流程也是较为熟悉的，当转换为TCP时其实相对来说变得比较简单了，握手过程以及上锁过程

都可以省略了，依照上学期延续的思路，将收消息和发消息实现为两个线程，其中的recvFun()和sendFun()分别实现，然后赋予给两个线程即可，具体如下所示(接收端和发送端相同)：

首先是发送消息函数，用一个字符数组作为发送消息的缓冲区，首先检测是否是结束的判定消息，在本次实验中将结束的判定消息设置为byebye，当其中一方输入此消息是就会结束双方的聊天，如果不是结束消息的话，用之前写好的DES加密函数对所发送的消息进行加密，然后通过套接字将其发送即可，如下所示：

```
void sendFun()
{
    char zhouyanlin_buffer[128];
    while (1)
    {
        char buf1[64];
        cout << "请输入:";
        cin >> buf1;
        if (strcmp(buf1, "byebye") == 0)
        {
            cout << "chat end" << endl;
            char buf0[64] = "to_the_end";
            strcpy(buf1, buf0); //加密过程
        }                       //判断聊天结束
        for (int i = 0; i < 8; i++)
        {
            char MesHex[16]; //存放密文
            char buf_part[8];
            for (int j = 0; j < 8; j++)
                buf_part[j] = buf1[8 * i + j];
            DES_Cry(buf_part, MesHex); //加密过程
            for (int k = 0; k < 16; k++)
                zhouyanlin_buffer[16 * i + k] = MesHex[k];
        }
        cout << "发送消息-加密后消息如下:";
        for (int i = 0; i < 64; i++)
            cout << zhouyanlin_buffer[i];
        cout << endl;
        //发送数据
        send(sockClient, zhouyanlin_buffer, 128, 0);
    }
}
```

其次是接收消息函数，因为会一直有请输入:的字段出现在窗口，所以不论接收到什么消息都需要先将光标移到本行的开始，使得一直出现在屏幕上的文字不会顶到窗口的顶部造成整体的布局混乱，通过一系列的\b来实现，然后当通过套接字收到密文时，首先会将收到的密文打印在屏幕中，接下来对收到的密文信息进行解密，随后将解密消息打印在屏幕中，在这里需要进行一个判断，如果接收到的是一个结束消息的话，就会结束当前聊天并退出程序，具体过程如下所示：

```
void recvFun()
{
    char buf[128];
```

```
    //接收服务发送的数据
    while (1)
    {
        int n;
        if (recv(sockClient, buf, 128, 0) > 0)
        {
            cout << "\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b";
            cout << "收到密文：";
            for (int i = 0; i < 128; i++)
                cout << buf[i];
            cout << endl;
            char zhouyanlin_xiaoxi[64];
            for (int i = 0; i < 8; i++)
            {
                char buf_part[16];
                char MyMessage_part[8];
                for (int j = 0; j < 16; j++)
                    buf_part[j] = buf[i * 16 + j];
                DES_Dec(buf_part, MyMessage_part); //解密过程
                for (int k = 0; k < 8; k++)
                    zhouyanlin_xiaoxi[8 * i + k] = MyMessage_part[k];
            }
            if (strcmp(zhouyanlin_xiaoxi, "to_the_end") == 0)
            {
                cout << "\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b";
                cout << "对方已end chat" << endl;
                break;
            }
            else
            {
                cout << "\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b";
                cout << "解密后信息如下:";
                for (int i = 0; i < 64; i++)
                {
                    int u = zhouyanlin_xiaoxi[i];
                    if (u != -52)
                        cout << zhouyanlin_xiaoxi[i];
                }
                cout << endl;
                cout << "请输入:";
            }
        }
        else
        {
            break;
        }
    }
}
```

当接受消息和发送消息都完成后，在分析一下客户端和服务端的主函数：

- 客户端

在客户端对一些初始化的数据进行定义，例如就像版本、通信方式、网络地址格式等，然后向服务端的端口进行建连，分别对两个函数进行赋予线程权限，具体代码如下所示：

```
int main()
{
    HANDLE h1, h2;    //线程句柄，其实就是一串数字用来标识线程对象
    SOCKADDR_IN addr; //保存服务器的socket地址信息
    int zhouyanlin_info;          //判断初始化信息
    WSADATA data;     //存储被WSAStartup函数调用后返回的Windows Sockets数据
    WORD version;     //socket版本

    //设定版本，与初始化
    version = MAKEWORD(2, 2); //版本设定
    zhouyanlin_info = WSAStartup(version, &data);

    //1.创建套接字
    sockClient = socket(AF_INET, SOCK_STREAM, 0);
    //要连接的服务器的ip,因为现在服务器端就是本机，所以写本机ip
    //127.0.0.1一个特殊的IP地址，表示是本机的IP地址
    addr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
    //端口要与服务器相同
    addr.sin_port = htons(11111);
    //用IPV4地址
    addr.sin_family = AF_INET;

    //主动连接服务器
    cout << "try to connect with the server" << endl;
    while (1)
    {
        //2.连接指定计算机端口
        if (connect(sockClient, (SOCKADDR*)&addr, sizeof(SOCKADDR)) == 0)
        {
            cout << "successfully connect" << endl;
            //创建线程后立即执行
            //向socket中发送/接受信息
            h1 = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)sendFun,
NULL, 0, NULL);
            h2 = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)recvFun,
NULL, 0, NULL);
            WaitForSingleObject(h1, INFINITE); //会阻塞，直到线程运行结束
            WaitForSingleObject(h2, INFINITE);
        }
        else
            ;
    }
    //关闭
    shutdown(sockClient, 2);
    closesocket(sockClient); //关闭套接字
    return 0;
}
```

- 服务端

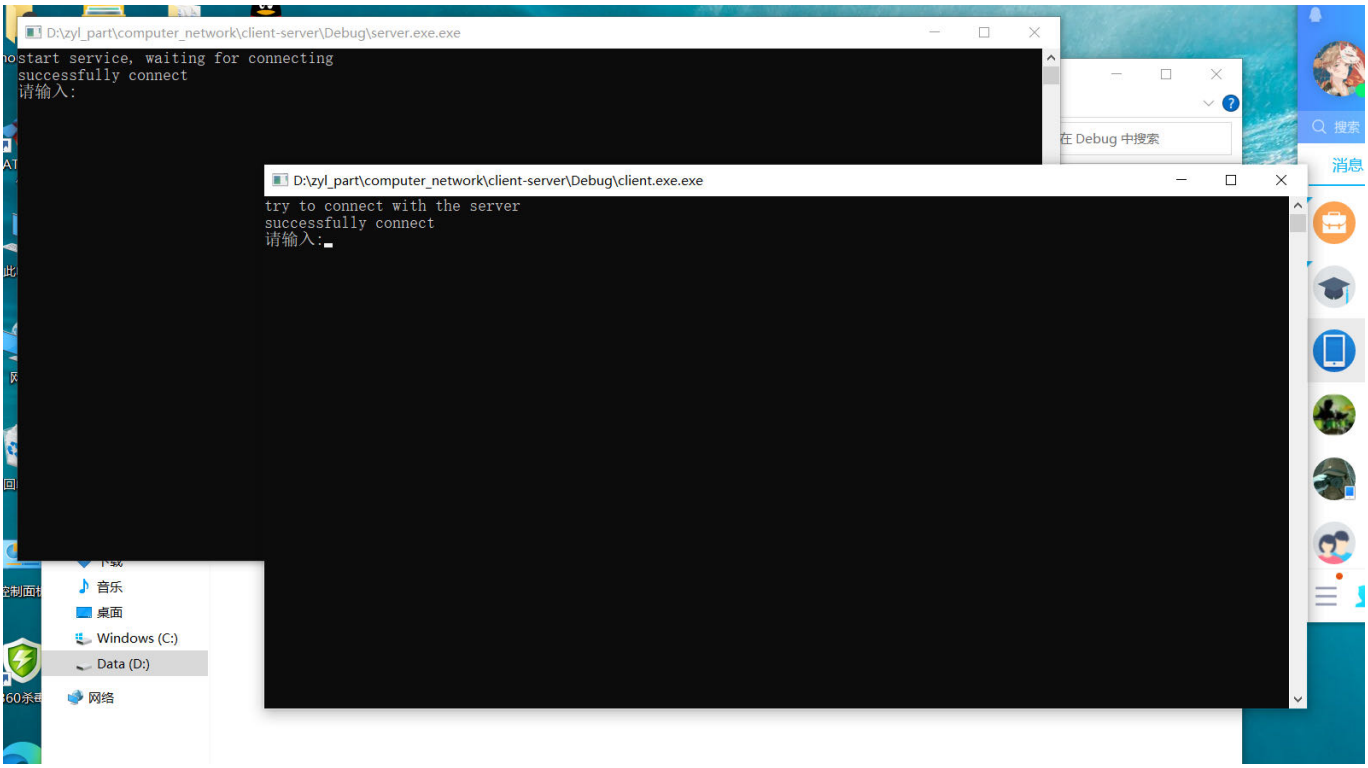
服务端也是一些初始化的定义，然后等待客户端建立连接，在建立连接完成后也是赋予两个线程权限，具体代码如下所示：

```
int main()
{

    SOCKET serverSocket; //监视的套接字
    SOCKADDR_IN newAddr; //保存客户端的socket地址信息
    SOCKADDR_IN addr;    //地址结构体，包括ip port(端口)
    WSADATA data;        //存储被WSAStartup函数调用后返回的Windows Sockets数据
    WORD version;        //socket版本
    int zhouyanlin_info;

    //在使用socket之前要进行版本的设定和初始化
    version = MAKEWORD(2, 2); //设定版本
    zhouyanlin_info = WSAStartup(version, &data);
    //应用程序或DLL只能在一次成功的WSAStartup()调用之后才能调用进一步的Windows
Sockets API函数。
    //有套接字的接口才能进行通信

    //1.创建socket
    serverSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); //AF_INET使用
IPV4地址，SOCK_STREAM使用流传输，IPPROTO_TCP使用TCP协议
    addr.sin_addr.S_un.S_addr = htonl(ADDR_ANY);              //表示任何的ip
过来连接都接受
    addr.sin_family = AF_INET;                                //使用ipv4的地
址
    addr.sin_port = htons(11111);                             //设定应用占用
的端口

    //2.绑定socket端口号
    bind(serverSocket, (SOCKADDR*)&addr, sizeof(SOCKADDR)); //将套接字
serverSocket与端口接收的ip绑定
    //3.开始监听，是否有客服端请求连接,最大连接数为3
    listen(serverSocket, 3);
    cout << "start service, waiting for connecting" << endl;
    int zhouyanlin_length = sizeof(SOCKADDR);
    //accept是一个阻塞函数，如果没有客户端请求，连接会一直等待在这里
    //该函数会返回一个新的套接字，这个新的套接字是用来与客户端通信的套接字，之前那个套接
字是监听的套接字
    while (1)
    {
        //4.接受来自客户端的连接请求
        sockConn = accept(serverSocket, (SOCKADDR*)&newAddr,
&zhouyanlin_length); //接受客户端的请求
        cout << "successfully connect" << endl;
        //创建线程后立即运行
        //第一个参数表示线程内核对象的安全属性；第二个参数表示线程栈空间大小；第三个参数
表示新线程所执行的线程函数地址（函数的名字），多个线程可以使用同一个函数地址
        //第四个参数是传递给线程函数的参数；第五个参数指定什么时候调用线程，为0表示线程
```

```
创建之后就可以进行调用；第六个参数返回线程的ID号，传入NULL表示不需要返回该线程ID号
        //5.向socket中读取/写入信息
        h1 = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)sendFun, NULL,
0, NULL); //用于发送的线程
        h2 = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)recvFun, NULL,
0, NULL); //用于接收的线程
    }
    //6.关闭
    closesocket(sockConn); //关闭套接字
    return 0;
}
```

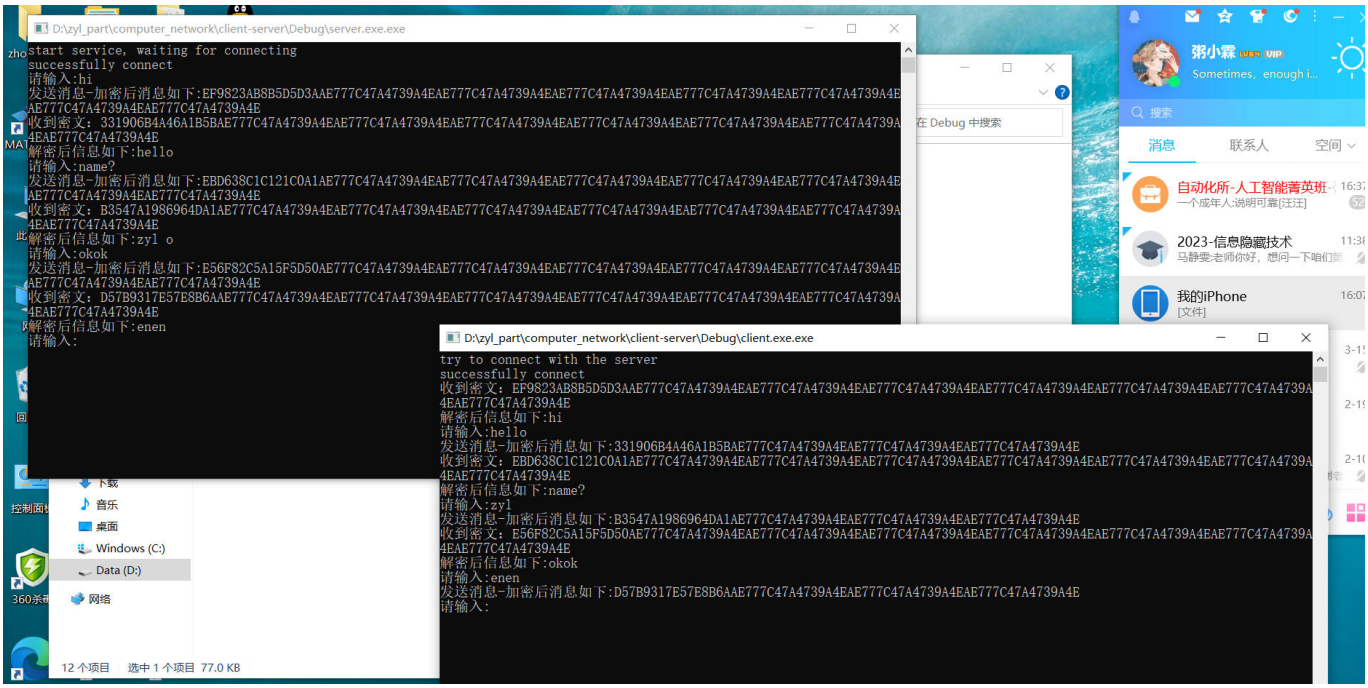
四、实验结果

1、建立连接成功



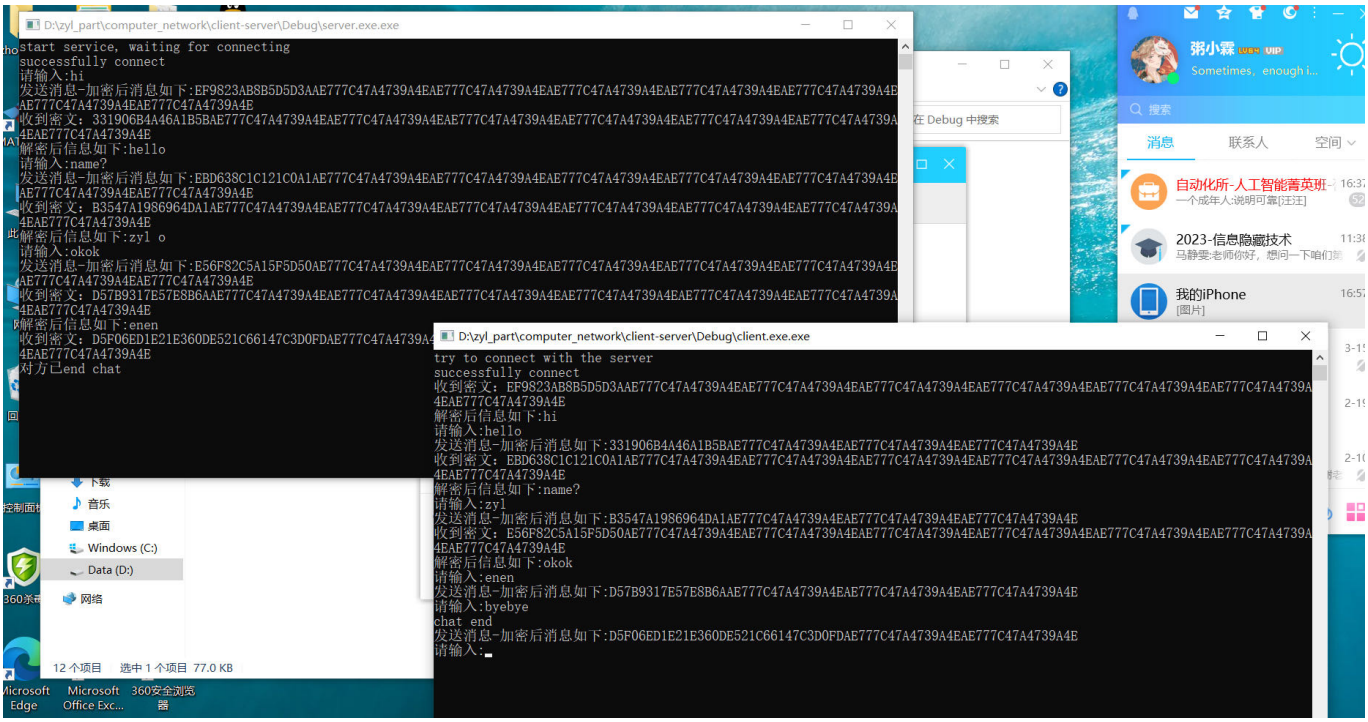
从上图可以看出连接的成功建立

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byebye时双方结束聊天如上图所示

五、心得体会

在本次实验中，首先细致地学习了des加解密的各个细节，也学习了socket通信编程和多线程编程，对上学期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