

网络安全技术课程实验报告

实验一



实验名称：_____ 基于 DES 加密的 TCP 聊天程序 _____

姓名：_____ 卢天鉴 _____

学号：_____ 1911445 _____

专业：_____ 计算机科学与技术 _____

提交日期：_____ 2022/3/18 _____

一、实验目的

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

二、实验要求及要点

- 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。

三、实验内容

1.具体过程：

DES 加密解密设计实现:

首先定义封装 DES 操作的类 CDesOperate，类的私有成员包括生成的 16 轮迭代密钥，初始密钥，以及加密解密流程中用到的四个函数。公有成员包括构造函数，析构函数，以及根据上述的四个函数封装的加密函数与解密函数，方便调用。

```
class CDesOperate{
private:
    ULONG32 m_arrOutKey[16][2];
    ULONG32 m_arrBufKey[2];

    INT32 HandleData(ULONG32 *left, ULONG8 choice);
    INT32 MakeData(ULONG32 *left, ULONG32 *right, ULONG32 number);
    INT32 MakeKey(ULONG32 *keyleft, ULONG32 *keyright, ULONG32 number);
    INT32 MakeFirstKey(ULONG32 *keyP);

public:
    CDesOperate();
    INT32 Encry(char *pPlaintext, int nPlaintextLength,
char *pCipherBuffer, int &nCipherBufferLength, char *pKey, int nKeyLength);
    INT32 Decry(char *pCipher, int nCipherBufferLength,
char *pPlaintextBuffer, int &nPlaintextBufferLength, char *pKey, int nKeyLength);
};

#endif
```

定义使用的静态数组

初始置换 IP，逆初始置换 IP，按位取值或赋值，置换计算，E 盒，压缩 S 盒，等分变换等等。

：

```

static ULONG8 pc_first[64] = {
58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,
62,54,46,38,30,22,14,6,64,56,48,40,32,24,16,8,
57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7
};

static ULONG8 pc_last[64] = {
40,8,48,16,56,24,64,32, 39,7,47,15,55,23,63,31,
38,6,46,14,54,22,62,30, 37,5,45,13,53,21,61,29,
36,4,44,12,52,20,60,28, 35,3,43,11,51,19,59,27,
34,2,42,10,50,18,58,26, 33,1,41,9,49,17,57,25
};

static ULONG32 pc_by_bit[64] = {
0x80000000L,0x40000000L,0x20000000L,0x10000000L, 0x8000000L,
0x4000000L, 0x2000000L, 0x1000000L, 0x800000L, 0x400000L,
0x200000L, 0x100000L, 0x80000L, 0x40000L, 0x20000L,0x10000L,
0x8000L, 0x4000L, 0x2000L, 0x1000L, 0x800L, 0x400L, 0x200L,
0x100L, 0x80L,0x40L,0x20L, 0x10L, 0x8L, 0x4L, 0x2L, 0x1L,
0x80000000L,0x40000000L,0x20000000L,0x10000000L, 0x8000000L,
0x4000000L, 0x2000000L, 0x1000000L, 0x800000L, 0x400000L,
0x200000L, 0x100000L, 0x80000L, 0x40000L, 0x20000L, 0x10000L,
0x8000L, 0x4000L, 0x2000L, 0x1000L, 0x800L, 0x400L, 0x200L,
0x100L, 0x80L, 0x40L,0x20L, 0x10L, 0x8L, 0x4L, 0x2L, 0x1L,
};

static ULONG8 des_P[32] = {
16,7,20,21, 29,12,28,17, 1,15,23,26,
5,18,31,10, 2,8,24,14, 32,27,3,9,
19,13,30,6, 22,11,4,25
};

static ULONG8 des_E[48] = {
32,1,2,3,4,5,4,5,6,7,8,9,8,9,10,11,12,13,
12,13,14,15,16,17,16,17,18,19,20,21,
20,21,22,23,24,25,24,25,26,27,28,29,
28,29,30,31,32,1
};

static ULONG8 des_S[8][64] =
{
    {
        0xe,0x0,0x4,0xf,0xd,0x7,0x1,0x4,0x2,0xe,0xf,0x2,0xb,
        0xd,0x8,0x1,0x3,0xa,0xa,0x6,0x6,0xc,0xc,0xb,0x5,0x9,
        0x9,0x5,0x0,0x3,0x7,0x8,0x4,0xf,0x1,0xc,0xe,0x8,0x8,
        0x2,0xd,0x4,0x6,0x9,0x2,0x1,0xb,0x7,0xf,0x5,0xc,0xb,
    }
}

```

```
static ULONG8 keyleft[28] =
{
    57,49,41,33,25,17,9,1,58,50,42,34,26,18,
    10,2,59,51,43,35,27,19,11,3,60,52,44,36
};
static ULONG8 keyright[28] =
{
    63,55,47,39,31,23,15,7,62,54,46,38,30,22,
    14,6,61,53,45,37,29,21,13,5,28,20,12,4
};
static ULONG8 lefttable[16] = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
static ULONG8 keychoose[48] = {
    14,17,11,24,1,5,3,28,15,6,21,10,
    23,19,12,4,26,8,16,7,27,20,13,2,
    41,52,31,37,47,55,30,40,51,45,33,48,
    44,49,39,56,34,53,46,42,50,36,29,32
};
```

DES 密钥生成:

DES 密钥是一个 64bit 的分组，但是其中 8bit 是用于奇偶校验的，所以密钥的有效位只有 56bit，由这 56bit 生成 16 轮子密钥。

密钥生成，首先将有效的 56bit 进行置换选择，将结果等分为 28bit 的两个部分，再根据所在的迭代轮数进行循环左移，左移后将两个部分合并为 56 位的密钥，从中选取 48 位作为此轮迭代的最终密钥，共生成 16 个 48 位的密钥。每一个密钥，分为两个 24 位的部分放在一个 ULONG32 的二维数组中保存。

```

INT32 CDesOperate::MakeKey(ULONG32 *keyleft, ULONG32 *keyright, ULONG32 number){
    uint32_t tmpkey[2] = {0,0};
    uint32_t *Ptmpkey = (uint32_t*)tmpkey;
    uint32_t *Poutkey = (uint32_t*)&m_arrOutKey[number];
    uint32_t leftandtab[3]={0x0,0x80000000,0xc0000000};
    INT32 j;
    memset((uint8_t*)tmpkey,0,sizeof(tmpkey));
    Ptmpkey[0] = *keyleft&leftandtab[lefttable[number]] ;
    Ptmpkey[1] = *keyright&leftandtab[lefttable[number]] ;
    if ( lefttable[number] == 1)
    {
        Ptmpkey[0] >>= 27;
        Ptmpkey[1] >>= 27;
    }
    else
    {
        Ptmpkey[0] >>= 26;
        Ptmpkey[1] >>= 26;
    }
    Ptmpkey[0] &= 0xffffffff;
    Ptmpkey[1] &= 0xffffffff;
    *keyleft <<= lefttable[number] ;
    *keyright <<= lefttable[number] ;
    *keyleft |= Ptmpkey[0] ;
    *keyright |= Ptmpkey[1] ;
    Ptmpkey[0] = 0;
    Ptmpkey[1] = 0;
    for ( j = 0 ; j < 48 ; j++)
    {
        if ( j < 24 )
        {
            if ( *keyleft&pc_by_bit[keychoose[j]-1])
            {
                Poutkey[0] |= pc_by_bit[j] ;
            }
        }
        else /*j>=24*/
        {
            if ( *keyright&pc_by_bit[(keychoose[j]-28)])
            {
                Poutkey[1] |= pc_by_bit[j-24] ;
            }
        }
    }
    return SUCCESS;
}

```

DES 加密运算:

DES 的加密运算也分为 16 轮迭代。

首先将明文分为 64bit 的数据块，不够 64 位的用 0 补齐。每一轮中，对每一个 64bit 的数据块，首先进行初始换位，并将数据分为 32bit 的两部分：

经过初始置换并且分组之后，将进行 DES 加密算法的核心部分。

首先，保持左部不变，将右部由 32 位扩展成为 48 位，分别存在两个 ULONG32 类型的变量里，每个占 24bit。再将右部扩展成为 48 位之后，与该轮的密钥进行异或操作，由于 48 位分在一个 ULONG32 数组中的两个元素中，故要进行两次异或。在异或操作完成之后，对

新的 48 位进行压缩操作，即 S 盒。

将其每取 6 位，进行一次操作。8 个 6bit 的数据存在 ULONG rexpbuf[8]中，然后进行数据压缩操作，每一个 6 位经过运算之后输出 4 位，故最终输出的是 32 位的压缩后的数据。

再把左右部分进行异或作为右半部分，最原始的右边作为左半部分，即将完成一轮完整的加密操作。最后进行逆初始置换，完成一轮完整的加密操作。

封装 DES 加密函数：

将上述运算整合在一起，可以封装成一个加密函数，以便于调用，其中 pPlaintext 为明文部分，nPlaintextLength 为明文长度，pCipherBuffer 为准备存放密文的缓冲区，nCIPHERBufferLength 为密文长度，pKey 为密钥，nKeyLength 为密钥长度。

首先检查初始密钥长度，若正确，则创建 16 轮迭代的密钥。

```
INT32 CDesOperate::Encrypt(char *pPlaintext, int nPlaintextLength, char *pCipherBuffer, int &nCipherBufferLength, char *pKey, int nKeyLength){
    if(nKeyLength != 8)
    {
        return 0;
    }
    MakeFirstKey((uint32_t *)pKey);
}
```

由于加解密均要以 32bit 为单位进行操作，故需要计算相关参数，以确定加密的循环次数以及密文缓冲区是否够用，确定后将需要加密的明文格式化到新分配的缓冲区内。

```
int nLenthofLong = ((nPlaintextLength+7)/8)*2;
if(nCipherBufferLength<nLenthofLong*4)
{
    nCipherBufferLength=nLenthofLong*4;
}
memset(pCipherBuffer,0,nCipherBufferLength);
uint32_t *pOutPutSpace = (uint32_t *)pCipherBuffer;
uint32_t * pSource;
if(nPlaintextLength != sizeof(uint32_t)*nLenthofLong)
{
    pSource= new uint32_t[nLenthofLong];
    memset(pSource,0,sizeof(uint32_t)*nLenthofLong);
    memcpy(pSource,pPlaintext,nPlaintextLength);
}
else
{
    pSource= (uint32_t *)pPlaintext;
}
```

开始对明文进行加密，加密后将之前分配的缓冲区从内存中删除。

```

uint32_t gp_msg[2] = {0,0};
for (int i=0;i<(nLenthofLong/2);i++)
{
    gp_msg[0] = pSource [2*i];
    gp_msg[1] = pSource [2*i+1];
    HandleData(gp_msg,(uint8_t)0);
    pOutPutSpace[2*i] = gp_msg[0];
    pOutPutSpace[2*i+1] = gp_msg[1];
}
if(pPlaintext!=(char *) pSource)
{
    delete []pSource;
}

```

TCP 聊天程序设计:

服务端:

```

int nListenSocket,nAcceptSocket;
struct sockaddr_in sLocalAddr, sRemoteAddr;
bzero(&sLocalAddr, sizeof(sLocalAddr));
sLocalAddr.sin_family = PF_INET;
sLocalAddr.sin_port = htons(6000);
sLocalAddr.sin_addr.s_addr = INADDR_ANY;
if ((nListenSocket = socket(PF_INET, SOCK_STREAM, 0)) == -1){
    perror("socket");
    exit(1);
}
if(bind(nListenSocket, (struct sockaddr *)&sLocalAddr, sizeof(struct sockaddr)) == -1){
    perror("bind");
    exit(1);
}
if(listen(nListenSocket, 5) == -1){
    perror("listen");
    exit(1);
}
printf("Listening...\n");
socklen_t nLength = 0;
nAcceptSocket = accept(nListenSocket, (struct sockaddr *)&sRemoteAddr, &nLength);
close(nListenSocket);
printf("server: got connection from %s, port %d, socket %d\n",inet_ntoa(sRemoteAddr.sin_addr), ntohs(sRemoteAddr.sin_port), nAcceptSocket);
SecretChat(nAcceptSocket, inet_ntoa(sRemoteAddr.sin_addr), "lutianjian");
close(nAcceptSocket);

```

先设定服务端地址，然后绑定套接字，进行监听，等待来自客户端的请求链接。

客户端:

客户端同服务端类似，先设定地址，然后与正在监听的服务端进行链接。聊天调用加密聊天函数，设置密钥为 lutianjian。

```

else if(choose == 'c'){
    cout << "Please input the server address:" << endl;
    char strIPAddr[16];
    cin >> strIPAddr;
    int nConnectSocket, nLength;
    struct sockaddr_in sDestAddr;
    if((nConnectSocket = socket(AF_INET, SOCK_STREAM, 0)) < 0){
        perror("Socket");
        exit(errno);
    }
    int SEVERPORT = 6000;

    sDestAddr.sin_family = AF_INET;
    sDestAddr.sin_port = htons(SEVERPORT);
    sDestAddr.sin_addr.s_addr = inet_addr(strIPAddr);
    if(connect(nConnectSocket, (struct sockaddr *) &sDestAddr, sizeof(sDestAddr)) != 0){
        perror("Connect");
        exit(errno);
    }else{
        printf("Connect Success! \nBegin to chat...\n");
        SecretChat(nConnectSocket, strIPAddr, "lutianjian");
    }
    close(nConnectSocket);
}else{
    cout << "Error!" << endl;
}
}

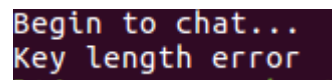
```

加密聊天函数：

该函数在完成必要的错误检查后，调用 fork() 函数创建了一个子进程，如“if(nPid != 0)”满足，则代表当前进程为父进程，否则为子进程。父进程负责接收密文消息，解密并输出到屏幕；同时子进程负责从标准输入读取消息，加密并发送到指定套接字，两个进程完全并行，实现实时聊天的功能。

同时经过了 DES 加密。

3.遇到问题及解决方法：

A terminal window with a dark background. The first line shows 'Begin to chat...' in green text. The second line shows 'Key length error' in red text.

这个问题出现了密钥长度问题，经过排查可以得到：

定义密钥时字符串长度为 4 的倍数 不然在解密的时候会出现报异常 字符串长度问题

四、个人总结

本次实验主要训练了在 LINUX 系统下的网络套接字编程，了解学习了 DES 加密的算法和核心思想，同时对全双工的通信有着更深的理解。现代密码学的特征是算法可以公开。保密的关键是如何保护好自已的密钥，而解密的关键则是如何能破解得到密钥。系统的安全管理者，要根据本系统实际所使用的密钥长度与其所保护的信息的敏感程度、重要程度以及系统实际所处安全环境的恶劣程度，在留有足够的安全系数的条件下来确定其密钥和证书更换周期的长短。同时，将已废弃的密钥和证书放入黑库归档，以备后用。密钥更换周期的正确安全策略是系统能够安全运行的保障，是系统的安全管理者最重要、最核心的日常工作任务。