网络安全技术

实
验
报
告

学　　院 计算机学院

年　　级 16 级

班　　级 3 班

学　　号 1613415

姓　　名 潘巧巧

手机号 13347629779

2020 年 3 月 5 日

目录

一、实验目的

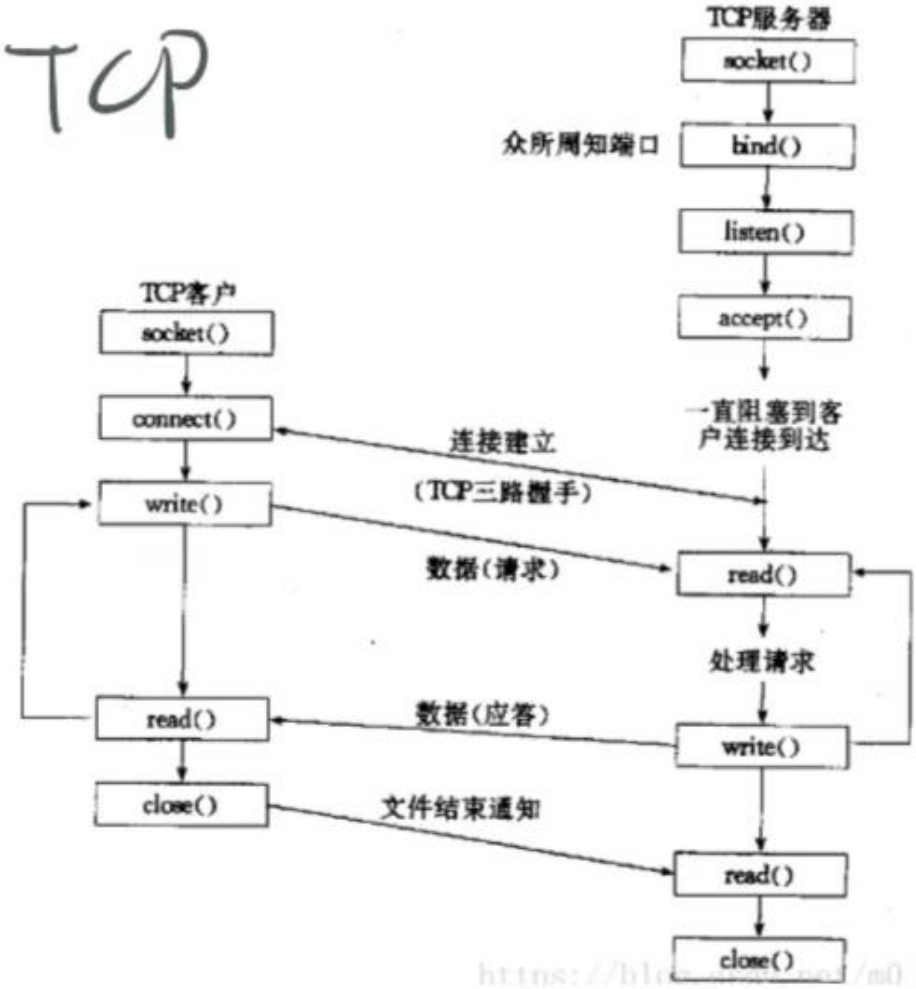
本次实验目的主要包括：
①利用 socket 编写一个 TCP 聊天程序。
②通信内容经过 DES 加密与解密。

二、实验内容

本实验使用 C++语言在 Linux 平台进行编程和运行，代码运行于腾讯云服务器主机。

实验利用 socket 相关接口实现了基于 TCP 的服务器和客户端，可以进行交互，且服务器和客户端的通信内容经过 DES 加密和解密，最终生成一个可执行文件 chat。

TCP 通信流程图如下：



DES 加密基本过程为：明文 → 8 个字母为一组进行分组，最后一组不足 8 个则补充'\0' → 根据 ASCII 码将每一组 8 个字母转为 8*8=64 位二进制 → 每一组进行 DES 加密 → 加密结果以 16 位一组进行分组，转为 4 位 16 进制，得到的结果+65，根据 ASCII 码进行转换 → 密文。

DES 解密流程为加密过程反过来。

实验涉及的程序文件包括：

（1） main.cpp：主函数文件。

（2） cs.h： client 和 server 接口。

（3） server.cpp：服务器代码。

（4） client.cpp：客户端代码。

（5） Des.h：DES 封装类，以及实现过程中需要的常量定义。

（6） Des.cpp：DES 类中的部分函数实现。

（7） DesEncry.cpp：DES 类中加密函数实现。

（8） DesDecry.cpp：DES 类中解密函数实现。

（9） DesFFunc.cpp：DES 类中 F 函数及其内部 4 个步骤实现。

（10） DesMakeKey.cpp：DES 类中生成子秘钥实现。

（11） HelpFunc.h：辅助函数定义。

（12） HelpFunc.cpp：辅助函数实现。

（13） DefineCode.h：返回值常量定义文件。

三、实验步骤及实验结果

本次实验步骤：

（1） main.cpp：主函数文件。

根据作业需求，可执行文件 chat 启动时均进入主函数文件的 main 函数，由用户输入决定当前可执行文件是服务器还是客户端。

（2） cs.h： client 和 server 接口。

（3） server.cpp：服务器代码。

主要流程为：socket() → bind() → listen() → accept() → recv()/read() ↔ send()/write() → close()

创建 socket → 绑定 socket 和端口号 → 监听端口号 → 接收来自客户端的连接请求 → 从 socket 中读取字符 → 关闭 socket

在每一次接收信息后需要进行 DES 解密，每一次发送信息前需要进行

DES 加密。

（4） client.cpp：客户端代码。

主要流程为：socket() → connect() → send()/write() ↔ recv()/read() → close()

创建 socket → 连接指定服务器的 IP/端口号 → 向 socket 中写入信息 → 关闭 socket

在每一次接收信息后需要进行 DES 解密，每一次发送信息前需要进行 DES 加密。

（5） Des.h：DES 封装类，以及实现过程中需要的常量定义。

暴露给外界调用的函数仅有 public 的加密函数 Encry()和解密函数 Decry()。其他函数均为 private，为实现公有函数时需要的细节。其他 DES 相关的常量数组也放在该文件下。

```
class CDesOperate {
public:
    int Encry(string plainText, string key, string& encryResult);  // 加密函数
    int Decry(string cipherText, string key, string& decryResult); // 解密函数

private:
    vector<bool> EncryPro(vector<bool> input, vector<vector<bool> > subKey); // 处理 64bit 的加密函数
    vector<bool> DecryPro(vector<bool> input, vector<vector<bool> > subKey); // 处理 64bit 的解密函数

    vector< vector<bool> > EncryDataProcess(string text); // 明文数据处理：string → bool
    vector< vector<bool> > DecryDataProcess(string text); // 密文数据处理：string → bool
    vector<bool> KeyProcess(string key); // key 处理：string → bool

    vector<bool> InitReplacementIP(vector<bool> input, int type); // 初始/逆初始置换 IP

    vector<bool> fFunc(vector<bool> input, vector<bool> key); // f 函数
    vector<bool> EBox(vector<bool> input); // f1：选择扩展运算 E
    vector<bool> keyAddition(vector<bool> input, vector<bool> key); // f2：密钥加运算
    vector<bool> selectCompressionOperation(vector<bool> input); // f3：选择压缩运算 S
    vector<bool> replacementOp(vector<bool> input); // f4：置换运算 P
```

```
    vector< vector<bool> > MakeKey(vector<bool> initKey); // 生成 16 个密
钥中的每一个子密钥，得到 subKey
};
```

（6） Des.cpp：DES 类中的部分函数实现。

主要包括初始置换 IP/逆初始置换 IP 函数 InitReplacementIP()，明文数据处理函数 EncryDataProcess()，密文数据处理函数 DecryDataProcess()，秘钥处理函数 KeyProcess()。

数据处理包括将长输入分组、补全等。

（7） DesEncry.cpp：DES 类中加密函数实现。

包括暴露给外界的公有函数 Encry()和仅处理 64bit 的私有函数 EncryPro()。

Encry()的主要过程为：获得 16 轮迭代需要的子秘钥 → 数据处理 → 分组调用 EncryPro()进行加密 → 密文编码。

EncryPro()的主要流程即为 DES 加密流程：初始置换 IP → 16 轮迭代 → 逆初始置换 IP。

```
int CDesOperate::Encry(string plainText, string key, string& encryResul
t) {
    if (key.size() != 8) {
        cout << "err: key size isn't 8" << endl;
        return DES_ERR_BIT;
    }

    vector<bool> binKey = KeyProcess(key); // key 预处理，string 转
vector<bool>
    vector< vector<bool> > subKey = MakeKey(binKey); // 生成子秘钥

    vector< vector<bool> > binPlainText = EncryDataProcess(plainText);
    vector< vector<bool> > binCipherText; //二进制结果

    for (int i = 0; i < binPlainText.size(); i++) { // 分组处理
        vector<bool> tempResult = EncryPro(binPlainText[i], subKey);
        binCipherText.push_back(tempResult);
    }

    //将 2 进制的加密结果分组，4 个一组，转为 16 进制，再+65 转为好输出的 ascii 码
    encryResult = "";
    int from10To16[] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
,
```

```
                        0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f
}; // 十进制转十六进制
    for (int i = 0; i < binCipherText.size(); i++) {
        int start = 0;
        while (start < 64) {
            vector<bool> binCipher(binCipherText[i].begin() + start, bi
nCipherText[i].begin() + start + 4);
            int decCipher = from2To10(binCipher);
            encryResult += from10To16[decCipher] + 65;
            start += 4;
        }
    }
    return SUCCESS;
}

vector<bool> CDesOperate:: EncryPro(vector<bool> input, vector<vector<b
ool> > subKey) {
    // 步骤 1: 初始置换 IP
    vector<bool> tempStep1 = InitReplacementIP(input, INIT_REPLACE_IP);
    map<string, vector<bool> > step1;
    vector<bool> tempLeft, tempRight;
    for (int i = 0; i < 32; i++) {
        tempLeft.push_back(tempStep1[i]);
        tempRight.push_back(tempStep1[i + 32]);
    }

    step1["left"] = tempLeft;
    step1["right"] = tempRight;

    // 步骤 2: 16 轮迭代
    // 每一轮: 左 = 右,  右 = f(右, subkey)^左
    for (int i = 0; i < 16; i++) {
        vector<bool> fFuncResult = fFunc(step1["right"], subKey[i]);
        vector<bool> xorResult = XOR(fFuncResult, step1["left"]);

        step1["left"] = step1["right"];
        step1["right"] = xorResult;
    }

    // 步骤 3: 逆初始置换 IP
    vector<bool> step3 = step1["left"];
    step3.insert(step3.end(), step1["right"].begin(), step1["right"].en
d());
    step3 = InitReplacementIP(step3, INVERSE_REPLACE_IP);
```

```
    return step3;
}
```

（8） DesDecry.cpp：DES 类中解密函数实现。

具体与 DesEncry.cpp 类似。

（9） DesFFunc.cpp：DES 类中 F 函数及其内部 4 个步骤实现。

共实现了包括 f 函数在内的 5 个函数。

即 f 函数 = 选择扩展运算 E + 密钥加运算 + 选择压缩运算 S + 置换 P。

```
#include <iostream>
#include <vector>
#include "Des.h"
#include "HelpFunc.h"
using namespace std;

// f 函数 = 选择扩展运算 E + 密钥加运算 + 选择压缩运算 S + 置换 P
vector<bool> CDesOperate::fFunc(vector<bool> input, vector<bool> key) {
    vector<bool> fStep1 = EBox(input);
    vector<bool> fStep2 = keyAddition(fStep1, key);
    vector<bool> fStep3 = selectCompressionOperation(fStep2);
    vector<bool> fStep4 = replacementOp(fStep3);
    return fStep4;
}

//（1）选择扩展运算(E 盒)：将输入的右边 32bit 扩展成为 48bit 输出
vector<bool> CDesOperate::EBox(vector<bool> input) {
    vector<bool> output;
    for (int i = 0; i < 48; i++) {
        output.push_back(input[des_E[i] - 1]);
    }
    return output;
}

//（2）秘钥加运算：将选择扩展运算输出的 48bit 作为输入，与 48bit 的子密钥进行异
或运算^
vector<bool> CDesOperate::keyAddition(vector<bool> input, vector<bool>
key) {
    return XOR(input, key);
}

//（3）选择压缩运算
vector<bool> CDesOperate::selectCompressionOperation(vector<bool> input
) {
    // 1、将密钥加运算的输出作为 48bit 输入，将其分为 8 组，每组 6bit，
```

```
    vector<vector<bool> > inputGroup;
    for (int i = 0; i < 8; i++) {
        vector<bool> tempGroup;
        for (int j = 0; j < 6; j++) {
            tempGroup.push_back(input[i * 6 + j]);
        }
        inputGroup.push_back(tempGroup);
    }

    vector<bool> result;
    // 2、分别进入 8 个 S 盒进行运算，得出 8*4=32bit 的输出结果，拼接后输出
    for (int i = 0; i < 8; i++) {
        int tempInt = des_S[i][from2To10(inputGroup[i])];
        vector<bool> tempBool = from10To2(tempInt, 4);
        result.insert(result.end(), tempBool.begin(), tempBool.end());
    }
    return result;
}

//（4）置换运算 P
vector<bool> CDesOperate::replacementOp(vector<bool> input) {
    vector<bool> result;
    for (int i = 0; i < 32; i++) {
        result.push_back(input[des_P[i] - 1]);
    }
    return result;
}
```

（10） DesMakeKey.cpp：DES 类中生成 16 轮子秘钥的方法实现。

主要过程为：奇偶校验位的补充 → 置换选择 PC-1 → 循环左移 LS → 置换选择 PC2。

```
#include <iostream>
#include <string.h>
#include <vector>
#include "Des.h"
#include "DefineCode.h"
#include "HelpFunc.h"
using namespace std;

// 生成 16 个密钥中的每一个子密钥
vector< vector<bool> > CDesOperate::MakeKey(vector<bool> initKey) {
    // (0) 密钥的奇偶校验位
    for (int i = 0; i < 8; i++) {
```

```
        int count1 = 0;
        for (int j = 0; j < 7; j++) {
            if (initKey[i * 8 + j] == true) {
                count1++;
            }
        }
        if (count1 % 2 == 0) { //当前为偶数个1，本组最后一个数字改为1
            initKey[i * 8 + 7] = true;
        } // 否则改为0
        else {
            initKey[i * 8 + 7] = false;
        }
    }

    // (1) 置换选择PC-1：从64bit初始密钥中选出56bit有效位
    // 64 = 8位奇偶校验 + 左28 + 右28，int = 4字节 = 4*8位
    // 初始密钥的第8、16、24、32、40、48、56、64位是奇偶校验位，其他为有效位
    vector< vector<bool> > keyPC1(2);
    for (int i = 0; i < 28; i++) {
        keyPC1[0].push_back(initKey[keyLeft[i] - 1]);
        keyPC1[1].push_back(initKey[keyRight[i] - 1]);
    }

    //（2）循环左移LS
    // keyPC1[0]和keyPC1[1]各自左移特定位，共16轮，每轮左移位数不同
    vector< vector<bool> > keyLS; // 1维是16轮，2维是每一轮的28*2=56位子
秘钥
    for (int i = 0; i < 16; i++) {
        keyPC1[0] = leftShift(leftTable[i], keyPC1[0]);
        keyPC1[1] = leftShift(leftTable[i], keyPC1[1]);
        vector<bool> tempKeyLS;
        tempKeyLS.insert(tempKeyLS.end(), keyPC1[0].begin(), keyPC1[0].
end());
        tempKeyLS.insert(tempKeyLS.end(), keyPC1[1].begin(), keyPC1[1].
end());
        keyLS.push_back(tempKeyLS);
    }

    //  （3）置换选择PC2
    // 将其余位置按照keyChoose置换位置，输出48bit，作为第N轮的子密钥
    vector< vector<bool> > keyPC2; // 1维是16轮，2维是每一轮的48位子秘钥
    for (int i = 0; i < 16; i++) {
        vector<bool> tempKeyPC2;
        for (int j = 0; j < 48; j++) {
```

```
            tempKeyPC2.push_back(keyLS[i][keyChoose[j] - 1]);
        }
        keyPC2.push_back(tempKeyPC2);
    }

    return keyPC2;
}
```

（11） HelpFunc.h：辅助函数定义。

包括循环左移函数、二进制转十进制函数、十进制转二进制函数、按位异或函数的定义。

（12） HelpFunc.cpp：辅助函数实现。

包括循环左移函数、二进制转十进制函数、十进制转二进制函数、按位异或函数的实现。

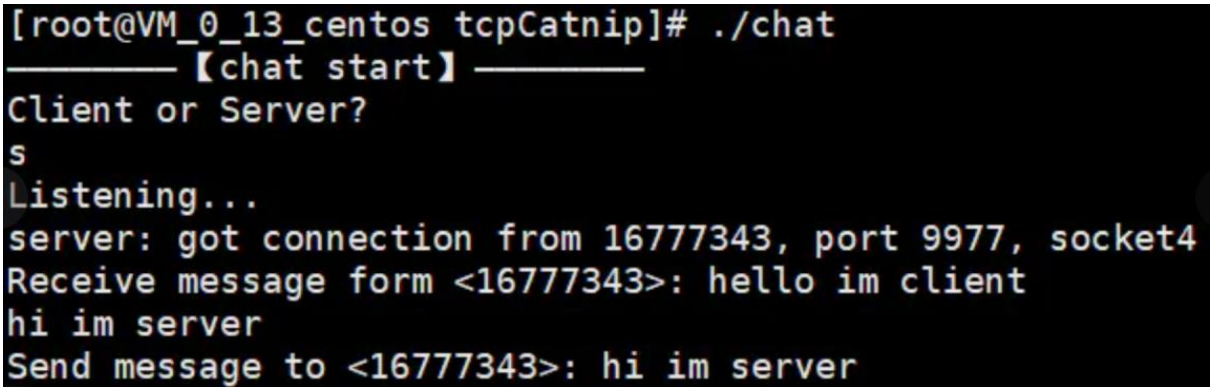
（13） DefineCode.h：返回值常量定义文件。

定义了涉及的返回值，方便查错。

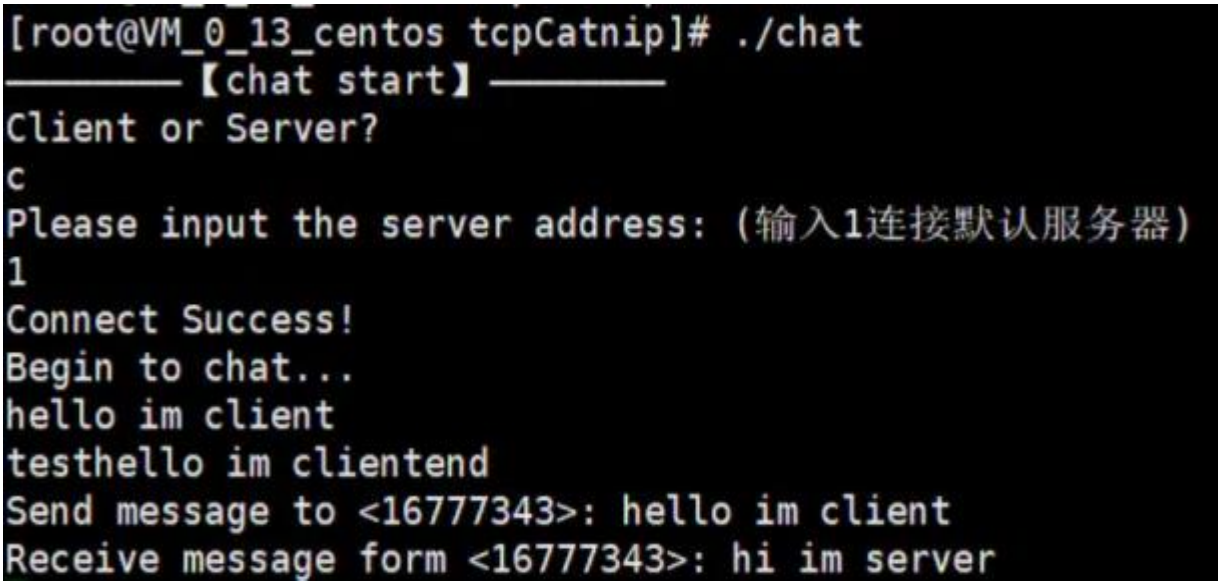
```
#define SUCCESS             0

#define SERVER_SOCKET_ERR   1000
#define SERVER_BIND_ERR     1001
#define SERVER_LISTEN_ERR   1002
#define SERVER_ACCEPT_ERR   1003
#define SERVER_RECV_ERR     1004
#define SERVER_SEND_ERR     1005

#define CLIENT_INETPTON_ERR 1010
#define CLIENT_SOCKET_ERR   1011
#define CLIENT_CONNECT_ERR  1012
#define CLIENT_SEND_ERR     1013
#define CLIENT_RECV_ERR     1014

#define DES_KEY             "BLACKHAT"
#define DES_ERR_BIT         1020
#define INIT_REPLACE_IP     1021 // 初始置换 IP
#define INVERSE_REPLACE_IP  1022 // 逆初始置换 IP
#define DES_ENCRY_ERR       1023 // 加密失败
#define DES_DECRY_ERR       1024 // 解密失败
```

本次实验结果：

【服务器】



【客户端】



四、实验遇到的问题及其解决方法

本次实验中遇到的问题和解决方法列举如下：

（1） 问题：子秘钥错误。

解决方法：忘记进行奇偶校验位的补充。

（2） 问题：加密得到的密文无法打印出来进行结果对比。

解决方法：原因是如果得到的二进制结果继续按 8 位进行编码，可能出现开头是 1 的情况，而 ASCII 码的最左位必然是 0，于是通过转换为 16 进制进行编码，但全都是前几个 ASCII 码，无法展示，于是将得到的编码+65，使其落在大写字母区域。

五、实验结论

复习了 TCP 的通信流程和 socket 编程，通过代码实现学习了 DES 的加密和解密流程。