上海理工大学光电信息与计算机工程学院

**《信息安全》实验报告**

****

**专　　业 计算机科学与技术**

**学生姓名　　 许耀永**

**学　 号 1712480131**

**年　　级 2017级**

**指导教师 刘亚**

**成 绩：**

**教师签字：**

报告格式要求

1. 正文字体中文为宋体，五号，行距为固定值18磅，西文为Times New Rome, 五号，行距为固定值18磅。
2. 章节标题为加粗宋体，小四号，段前段后各0.5行，行距为固定值18磅。
3. 打印时需双面打印。

目录

[实验一 古典加密算法的实现与破解 1](#_Toc493076389)

[实验二 AES的实现 2](#_Toc493076390)

[实验三 RSA的实现 3](#_Toc493076391)

[实验四 网络侦听工具及协议分析 4](#_Toc493076392)

# 实验一 古典密码实验

**一、 实验目的**

(1)实现凯撒密码的加解密

其中加密的明文使用小写字母，解密的密文用大写字母

(2)

**二、实验要求**

**三、程序的数据结构设计和算法流程图或算法描述**

**I.Caser密码**

(1)在数学形式上，将字母映射成数字

(2)选择一个整数密钥 K(K 在 1 到 25 之间)

(3)加密时，依据“加 K 模 26” ，将原字母由 L 转换成(L + K) % 26

(4)解密时，依据“减 K 模 26” ，将密文字母由 L 转换成(L - K) % 26

**II. Polyalphabetic密码**

(1)实现维吉尼亚密码的加解密

(2)实现维吉尼亚密码的破解

**四、源程序**

**I.Caser密码**

**//CaserCipher.h**

#include<stdio.h>

#include<ctype.h>

char caesar(char c,int k){

if(isalpha(c) && c!= toupper(c) ){

c = toupper(c);

c = ( ( (c - 65) + k ) % 26 ) + 65; //encrypt

}else if(isalpha(c) && c == toupper(c) ){

c = ( ( (c - 65) - k + 26) % 26 ) + 65; //decrypt

c = tolower(c);

}

return c;

}

//test.cpp

#include"caesarCipher.h"

#include<stdlib.h>

#include<iostream>

#include<string>

using namespace std;

int main(){

string input,output;

int choice = 0;

while(choice != 2){

cout << endl << "Press 1:Encrypt/Decrypt" << endl << "Press 2:quit "<< endl;

try{

cin >> choice;

if(choice != 1 && choice != 2){

throw "Incorrect Choice";

}

}catch(const char\* chc){

cerr << "Incorrect Choice "<<endl;

return 1;

}

if(choice == 1){

int key;

try{

cout << "Chose key value(a number between 1 - 26):";

cin >> key;

cin.ignore();

if(key < 1 || key > 26){

throw "Incorrect key";

}

}catch(const char\* K){

cerr << "Incorrect key value chosen "<<endl;

return 1;

}

try{

cout << endl << "lowerCase letter for encrypt"<<endl <<"upperCase letter for decrypt" << endl;

cout << "Enter cipherTest: ";

getline(cin,input);

for(int i = 0;i < input.size(); i++){

//if(!(input[i] >= 'a' && input[i] <= 'z') && !(input[i] >= 'A' && input[i] <= 'Z')) throw "Incorrecr String";

}

}catch(const char\* str){

cerr << "may have some digit or special symbols "<<endl;

cerr << "put only alphabets "<<endl;

return 1;

}

for(unsigned int x = 0;x < input.length();x++){

output += caesar(input[x],key);

}

cout << output << endl;

output.clear();

}

}

}

**II. Polyalphabetic密码**

**//NewPolyalphabeticCipher.h**

#include<cstdio>

#include<cctype>

#include<string>

#include<cmath>

using namespace std;

char vigenere\_table[26][26] = {

'A','B','C','D','E','F','G','H','I','J','K',

'L','M','N','O','P','Q','R','S','T','U','V',

'W','X','Y','Z',

'B','C','D','E','F','G','H','I','J','K','L',

'M','N','O','P','Q','R','S','T','U','V','W',

'X','Y','Z','A',

'C','D','E','F','G','H','I','J','K','L','M',

'N','O','P','Q','R','S','T','U','V','W','X',

'Y','Z','A','B',

'D','E','F','G','H','I','J','K','L','M','N',

'O','P','Q','R','S','T','U','V','W','X','Y',

'Z','A','B','C',

'E','F','G','H','I','J','K','L','M','N','O',

'P','Q','R','S','T','U','V','W','X','Y','Z',

'A','B','C','D',

'F','G','H','I','J','K','L','M','N','O','P',

'Q','R','S','T','U','V','W','X','Y','Z','A',

'B','C','D','E',

'G','H','I','J','K','L','M','N','O','P','Q',

'R','S','T','U','V','W','X','Y','Z','A','B',

'C','D','E','F',

'H','I','J','K','L','M','N','O','P','Q','R',

'S','T','U','V','W','X','Y','Z','A','B','C',

'D','E','F','G',

'I','J','K','L','M','N','O','P','Q','R','S',

'T','U','V','W','X','Y','Z','A','B','C','D',

'E','F','G','H',

'J','K','L','M','N','O','P','Q','R','S','T',

'U','V','W','X','Y','Z','A','B','C','D','E',

'F','G','H','I',

'K','L','M','N','O','P','Q','R','S','T','U',

'V','W','X','Y','Z','A','B','C','D','E','F',

'G','H','I','J',

'L','M','N','O','P','Q','R','S','T','U','V',

'W','X','Y','Z','A','B','C','D','E','F','G',

'H','I','J','K',

'M','N','O','P','Q','R','S','T','U','V','W',

'X','Y','Z','A','B','C','D','E','F','G','H',

'I','J','K','L',

'N','O','P','Q','R','S','T','U','V','W','X',

'Y','Z','A','B','C','D','E','F','G','H','I',

'J','K','L','M',

'O','P','Q','R','S','T','U','V','W','X','Y',

'Z','A','B','C','D','E','F','G','H','I','J',

'K','L','M','N',

'P','Q','R','S','T','U','V','W','X','Y','Z',

'A','B','C','D','E','F','G','H','I','J','K',

'L','M','N','O',

'Q','R','S','T','U','V','W','X','Y','Z','A',

'B','C','D','E','F','G','H','I','J','K','L',

'M','N','O','P',

'R','S','T','U','V','W','X','Y','Z','A','B',

'C','D','E','F','G','H','I','J','K','L','M',

'N','O','P','Q',

'S','T','U','V','W','X','Y','Z','A','B','C',

'D','E','F','G','H','I','J','K','L','M','N',

'O','P','Q','R',

'T','U','V','W','X','Y','Z','A','B','C','D',

'E','F','G','H','I','J','K','L','M','N','O',

'P','Q','R','S',

'U','V','W','X','Y','Z','A','B','C','D','E',

'F','G','H','I','J','K','L','M','N','O','P',

'Q','R','S','T',

'V','W','X','Y','Z','A','B','C','D','E','F',

'G','H','I','J','K','L','M','N','O','P','Q',

'R','S','T','U',

'W','X','Y','Z','A','B','C','D','E','F','G',

'H','I','J','K','L','M','N','O','P','Q','R',

'S','T','U','V',

'X','Y','Z','A','B','C','D','E','F','G','H',

'I','J','K','L','M','N','O','P','Q','R','S',

'T','U','V','W',

'Y','Z','A','B','C','D','E','F','G','H','I',

'J','K','L','M','N','O','P','Q','R','S','T',

'U','V','W','X',

'Z','A','B','C','D','E','F','G','H','I','J',

'K','L','M','N','O','P','Q','R','S','T','U',

'V','W','X','Y'

};

void Decrypt(string in,string &out,string k){

int i = 0;

for(string::iterator it = in.begin(); it != in.end();it++){

if((\*it) != ' '){

int column = toupper(k[i % k.length()]) - 'A';

int row;

for(row = 0;row < 26;row++){

if(vigenere\_table[row][column] == \*it) break;

}

out += 'A' + row;

i++;

}else{

out += ' ';

}

}

}

void Encrypt(string in,string &out,string k){

int i = 0;

for(string::iterator it = in.begin(); it != in.end();it++){

if((\*it) != ' '){

int row = toupper(\*it) - 'A';

int column = toupper(k[i % k.length()]) - 'A';

out += vigenere\_table[row][column];

i++;

}else{

out += ' ';

}

}

}

**//test.cpp**

#include "NewPolyalphabeticCipher.h"

#include "EndPolyalphabeticCipher.h"

#include<algorithm>

#include<string>

#include<iostream>

using namespace std;

int main(){

string input,output,key;

int choice = 0;

while(true){

cout << "----------------" << endl;

cout << "1:Encrypt "<<endl

<<"2:Decrypt "<<endl

<<"3:EndDecrypt "<<endl

<<"4:quit"<<endl;

cout << "----------------" << endl;

try{

cin >> choice;

cin.ignore();

if(choice != 1 && choice != 2 && choice != 3 && choice != 4){

throw "Incorrect Choice";

}

}catch(const char\* chc){

cerr << "Incorrecr Choice "<< endl;

return 1;

}

if(choice == 4) break;

try{

cout << endl << "Enter cipher text:";

getline(cin,input);

for(int i = 0;i < input.size();i++){

if( (!(input[i] >= 'a'&& input[i] <= 'z')) && (!(input[i] >= 'A' && input[i] <= 'Z')) && (!(input[i] == ' ')) ){

throw "Incorrect string ";

}

}

}catch(const char\* str){

cerr <<"input string have some digits or special sysbol"

<<endl;

cerr << "please enter only alphabets" << endl;

return 1;

}

if(choice == 1 || choice == 2){

cout << "Enter Key(alphabets/words):";

getline(cin,key);

if(choice == 1){

Encrypt(input,output,key);

cout<<endl<<"Cipher text:"<<output<<endl;

}else{

Decrypt(input,output,key);

cout<<endl<<"Plain text:"<<output<<endl;

}

}else if(choice == 3){

transform(input.begin(),input.end(),input.begin(),::tolower);

getKey(input);

print();

getAns(input);

cout << endl;

}

input.clear();

output.clear();

key.clear();

}

return 0;

}

**//破解代码**

#include<cstring>

#include<string>

#include<vector>

#include<set>

#include<map>

#include<algorithm>

#include<iostream>

using namespace std;

struct Node{

double value;

int length;

};

vector<Node> key;

set<int> keyLen;

double g[] = {

0.08167,0.01492,0.02782,0.04253,0.12702,0.02228,0.02015,

0.06094,0.06966,0.00153,0.00772,0.04025,0.02046,0.06749,

0.07507,0.01929,0.00095,0.05987,0.06327,0.09056,0.02758,

0.00978,0.02360,0.00150,0.01974,0.00074

};

bool cmp(Node a,Node b){

return a.value < b.value;

}

double coincidenceIndex(string cipher,int start,int length){

double index = 0.000;

int sum = 0;

int num[26];

memset(num,0,sizeof(num));

//keyPoint

while(start <= cipher.length()){

num[cipher[start] - 'a']++;

start += length;

sum++;

}

for(int i = 0;i < 26;i++){

if(num[i] <= 1)continue;

index += (double) (num[i] \* (num[i] - 1)) / (double) ((sum) \* (sum -1));

}

return index;

}

void findSame(string cipher){

for(int i = 3;i < 5;i++){

for(int j = 0;j < cipher.length() - 1;j++){

string p = cipher.substr(j,i);

for(int k = j + i;k < cipher.length() - i;k++){

string tmp = cipher.substr(k,i);

if(tmp == p){

Node x;

x.length = k - j;

key.push\_back(x);

}

}

}

}

}

int gcd(int a,int b){

if(b == 0) return a;

else return gcd(b,a % b);

}

void print(){

for(int i = 0;i < key.size();i++){

cout << key[i].length<< " and " << key[i].value << endl;

}

}

void getKey(string cipher){

findSame(cipher);

for(int i = 0;i < key.size();i++){

int x = key[i].length;

for(int j = 0;j < key.size();j++){

if(key[i].length > key[j].length){

keyLen.insert(gcd(key[i].length,key[j].length));

}else{

keyLen.insert(gcd(key[j].length,key[i].length));

}

}

}

key.clear();

set<int>::iterator it = keyLen.begin();

while(it != keyLen.end()){

int length = (\*it);

if(length == 1){

it++;

continue;

}

double sum = 0.000;

cout << length << " ";

for(int i = 0;i < length;i++){

cout << coincidenceIndex(cipher,i,length) << " ";

sum += coincidenceIndex(cipher,i,length);

}

cout << endl;

Node x;

x.length = length;

x.value = (double)fabs(0.065 - (double)(sum / (double)length));

if(x.value <= 0.1) key.push\_back(x);

it++;

}

sort(key.begin(),key.end(),cmp);

}

void getAns(string cipher){

int lss = 0;

while(lss < key.size() && lss < 10){

Node x = key[lss];

int ans[cipher.length()];

memset(ans,0,sizeof(ans));

map<char,int> mp;

for(int i = 0;i < x.length;i++){

double maxPg = 0.000;

for(int k = 0;k < 26;k++){

mp.clear();

double pg = 0.000;

int sum = 0;

for(int j = i;j < cipher.length();j += x.length){

char c = (char)((cipher[j] - 'a' + k) % 26 + 'a');

mp[c]++;

sum++;

}

for(char j = 'a'; j <= 'z';j++){

pg += ( (double)mp[j] / (double)sum ) \* g[j - 'a'];

}

if(pg > maxPg){

ans[i] = k;

maxPg = pg;

}

}

}

cout << endl << "Clear text:"<< endl;

for(int i = 0;i < cipher.length();i++){

cout << (char) ((cipher[i] - 'a' + ans[i % x.length]) % 26 + 'a');

}

cout << endl;

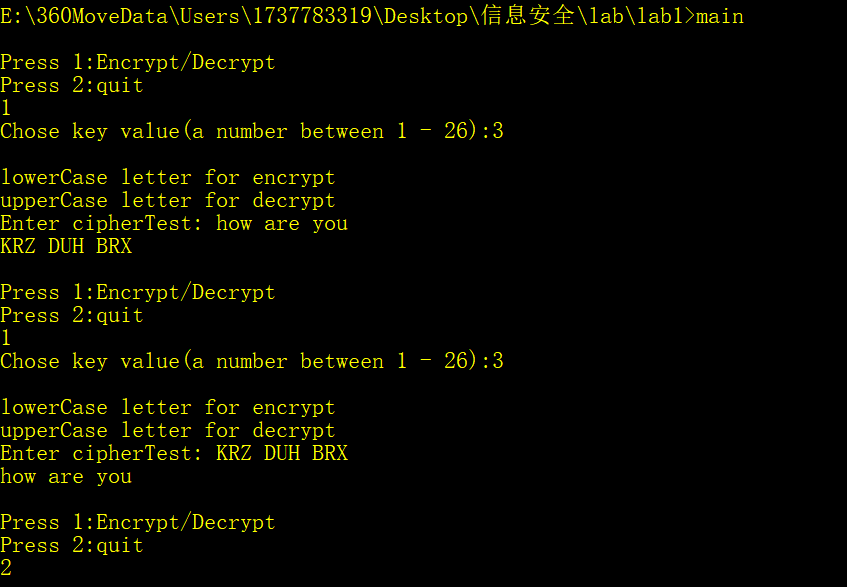
lss++;

}

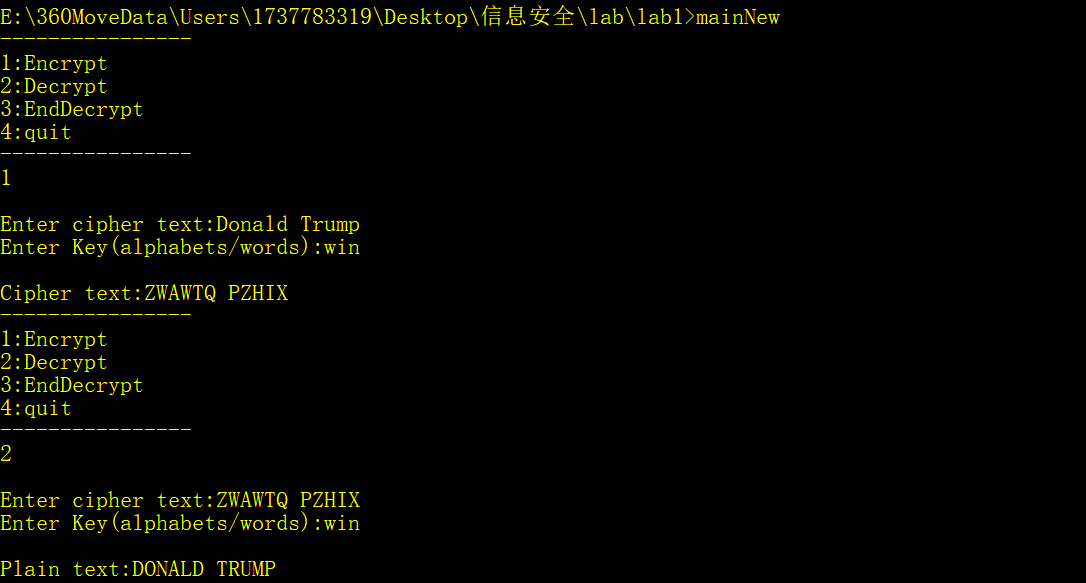
}

**五、程序运行后的结果截图**

**I.Caser密码**



**II. Polyalphabetic密码**



# 实验二 AES实验

**一、 实验目的**

实现AES的加密(Encryption)和解密(Decryption)

**二、 实验要求**

(1)AES的所有操作都是针对于一个二维数组state

一个state数组由 4行Nb列组成

(2)AES基于下列5个基本操作

1.异或(Exclusive Disjunction/XOR):当两个digit不同时输出为真

2.SubByte/InverseSubByte : 一个byte被另一个byte替代

加密时根据S\_Box查表，解密时根据Si\_Box查表

(每个byte都可以表示为两个十六进制的digit，

其中第一个代表S\_Box的行，第二个代表S\_Box的列)

3.Shift Rows(Rotation)/Inverse Shift Rows:

重排bytes，以不同的偏移量循环左移后三行

Row 0不移动

Row 1 移动一个byte

Row 2 移动两个byte

Row3 移动三个byte

解密时循环右移相对应的byte

4.MixColumns/InverseMixColumns:

二维数组state的每一列上的每一个byte映射为一个新的值

a.将state的每一列左乘一个4 \* Nb的矩阵，所得结果逐个异或而产生一个byte

b. a的乘法操作可以用两个查找表L\_Table和E\_Table来实现

step1 :一个byte，其表示为两个十六进制的digit，

而第一个digit用作L\_Table的行数，第二个digit用作L\_Table的列数

step2 : 将两个从L\_Table获得的值相加，得到一个新的byte

step3 : 在E\_Table查看这个相加的结果(按digit来查)

Note:任何数与1相乘的结果都是它本身

5.AddRoundKey:state数组的每个byte都和round key的每个byte相异或(XOR)

(3) 对于128比特的key,执行10 rounds(轮)

对于192比特的key,执行 12 rounds(轮)

对于 256比特的key,执行 14 rounds(轮)

(4) AES的框架

从第一轮到倒数第二轮，均执行:

a.SubBytes/InverseSubBytes

b.ShiftRows/InverseShiftRows

c.MixColumns/InverseMixColumns

d.AddRoundKey

最后一轮 不执行 MixColumns/InverseMixColumns

(6) Key Expansion

将一个长度为Nk的输入key生成一个长度为Nb \* (Nr + 1)的线性key

Nb : state数组的列数

Nr: 轮的个数

Nk: key的长度

记W[i]为expanded key的第i个word

**三、程序的数据结构设计和算法流程图或算法描述**

(1) Key Expansion

Begin

word temp;

for i <- 0 to (Nk - 1)

w[i] <- ((unsigned char) key [ 4 \* i ] << 24)

| ((unsigned char) key [ 4 \* i + 1 ] << 16)

| ((unsigned char) key [ 4 \* i + 2 ] << 8)

| ((unsigned char) key [ 4 \* i + 3 ] );

end for

for i <- (Nk - 1) to Nb \* Nr

temp = w[i - 1];

if (i mod Nk == 0)

temp = SubWord ( RotWord (temp))

XOR (Rcon [ i / Nk ] << 24) ;

else if ( Nk > 6 and ( i mod Nk ) == 4 )

temp = SubWord(temp);

end if

w[i] = w[i - Nk] XOR temp;

end for

End

(2) Encryption

Begin

state = plaintext

1.KeyExpansion

2.AddRoundkey(state,Expandedkey[0])

3.for r <- 1 to (Nr - 1)

a. SubBytes (state , S-Box)

b. ShiftRows (state)

c. MixColumns(state)

d. AddRoundKey(state,ExpandedKey[r])

4. SubBytes (state , S-Box)

ShiftRows (state)

AddRoundKey(state,ExpandedKey[Nr])

Out = cipherText

End

(3) Decryption

Begin

State = cipherText

1.Key Expansion

2.AddRoundKey(state,ExpandedKey[0])

3.for r <- (Nr - 1) to 1

a. InverseShiftRows(state)

b. InverseSubBytes(state,S-Box)

c. AddRoundKey(State,ExpandedKey[r])

4. InverseSubBytes(state,S-Box)

InverseShiftRows(state)

AddRoundKey(state,ExpandedKey[Nr])

Out = plaintext

End

**四、源程序**

**//AES.h**

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cstdlib>

#include<iostream>

#include<cmath>

class AES{

public:

AES(unsigned char\* key);

virtual ~AES();

unsigned char\* Cipher(unsigned char\* input);//16字节为一组

unsigned char\* InvCipher(unsigned char\* input);

void\* Cipher(void\* input, int length=0);//初始化过了

void\* InvCipher(void\* input, int length);

private:

unsigned char Sbox[256];

unsigned char InvSbox[256];//1字节=8位

unsigned char w[11][4][4];

void KeyExpansion(unsigned char\* key, unsigned char w[][4][4]);

unsigned char FFmul(unsigned char a, unsigned char b);

void SubBytes(unsigned char state[][4]);

void ShiftRows(unsigned char state[][4]);

void MixColumns(unsigned char state[][4]);

void AddRoundKey(unsigned char state[][4], unsigned char k[][4]);

void InvSubBytes(unsigned char state[][4]);

void InvShiftRows(unsigned char state[][4]);

void InvMixColumns(unsigned char state[][4]);

};

**//AES.cpp**

#include "string.h"

#include "AES.h"

//unsigned char 8位 1字节

AES::AES(unsigned char\* key){

unsigned char sBox[] ={

0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76,

0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0,

0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15,

0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75,

0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84,

0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf,

0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8,

0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2,

0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73,

0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb,

0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79,

0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08,

0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a,

0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e,

0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf,

0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16

};

unsigned char invsBox[256] = {

0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb,

0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb,

0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0x0b,0x42,0xfa,0xc3,0x4e,

0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,0x6d,0x8b,0xd1,0x25,

0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92,

0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84,

0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,0x05,0xb8,0xb3,0x45,0x06,

0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,0x03,0x01,0x13,0x8a,0x6b,

0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,0xe6,0x73,

0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e,

0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0x0e,0xaa,0x18,0xbe,0x1b,

0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4,

0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,0x59,0x27,0x80,0xec,0x5f,

0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9c,0xef,

0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61,

0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0x0c,0x7d

};

memcpy(Sbox, sBox, 256);

memcpy(InvSbox, invsBox, 256);

KeyExpansion(key, w);

}//构造函数

AES::~AES(){}//析构函数

unsigned char\* AES::Cipher(unsigned char\* input){//只往后16个字节

unsigned char state[4][4];

int i,r,c;

for(r=0; r<4; r++){

for(c=0; c<4 ;c++){

state[r][c] = input[c\*4+r];

}

}

AddRoundKey(state,w[0]);//原秘钥，一共11轮密钥

for(i=1; i<=10; i++){

SubBytes(state);

ShiftRows(state);

if(i!=10)MixColumns(state);

AddRoundKey(state,w[i]);

}

for(r=0; r<4; r++){

for(c=0; c<4 ;c++){

input[c\*4+r] = state[r][c];

}

}

return input;

}

unsigned char\* AES::InvCipher(unsigned char\* input){

unsigned char state[4][4];

int i,r,c;

for(r=0; r<4; r++){

for(c=0; c<4 ;c++){

state[r][c] = input[c\*4+r];

}

}

AddRoundKey(state, w[10]);//注意逆向的顺序

for(i=9; i>=0; i--){

InvShiftRows(state);

InvSubBytes(state);

AddRoundKey(state, w[i]);

if(i){

InvMixColumns(state);

}

}

for(r=0; r<4; r++){

for(c=0; c<4 ;c++){

input[c\*4+r] = state[r][c];

}

}

return input;

}

void\* AES::Cipher(void\* input, int length){

unsigned char\* in = (unsigned char\*) input;

int i;

if(!length){

while(\*(in+length++));

in = (unsigned char\*) input;

}

for(i=0; i<length; i+=16){

Cipher(in+i);

}

return input;

}

void\* AES::InvCipher(void\* input, int length){

unsigned char\* in = (unsigned char\*) input;

int i;

for(i=0; i<length; i+=16){

InvCipher(in+i);

}

return input;

}

void AES::KeyExpansion(unsigned char\* key, unsigned char w[][4][4]){

int i,j,r,c;

unsigned char rc[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36};//轮常量

for(r=0; r<4; r++){

for(c=0; c<4; c++){

w[0][r][c] = key[r+c\*4];

}

}

for(i=1; i<=10; i++){

for(j=0; j<4; j++){

unsigned char t[4];

for(r=0; r<4; r++){

t[r] = j ? w[i][r][j-1] : w[i-1][r][3];

}

if(j == 0){

for(r=0; r<=3; r++){

t[r] = Sbox[t[(r+1)%4]];

}

t[0] ^= rc[i-1];

}

for(r=0; r<4; r++){

w[i][r][j] = w[i-1][r][j] ^ t[r];

}

}

}

}

//有限域乘法

unsigned char AES::FFmul(unsigned char a, unsigned char b){

unsigned char bw[4];

unsigned char res=0;

int i;

bw[0] = b;

for(i=1; i<4; i++){

bw[i] = bw[i-1]<<1;

if(bw[i-1]&0x80){

bw[i]^=0x1b;

}

}

for(i=0; i<4; i++){

if((a>>i)&0x01){

res ^= bw[i];

}

}

return res;

}

void AES::SubBytes(unsigned char state[][4]){//字节替换

int r,c;

for(r=0; r<4; r++){

for(c=0; c<4; c++){

state[r][c] = Sbox[state[r][c]];

}

}

}

//行移位变换

void AES::ShiftRows(unsigned char state[][4]){

unsigned char t[4];

int r,c;

for(r=1; r<4; r++){

for(c=0; c<4; c++){

t[c] = state[r][(c+r)%4];

}

for(c=0; c<4; c++){

state[r][c] = t[c];

}

}

}

void AES::MixColumns(unsigned char state[][4]){ //列混淆变换

unsigned char t[4];

int r,c;

for(c=0; c< 4; c++){

for(r=0; r<4; r++){

t[r] = state[r][c];

}

for(r=0; r<4; r++){

state[r][c] = FFmul(0x02, t[r])

^ FFmul(0x03, t[(r+1)%4])

^ FFmul(0x01, t[(r+2)%4])

^ FFmul(0x01, t[(r+3)%4]);

}

}

}

//轮密钥加变换，正逆是一样的，异或

void AES::AddRoundKey(unsigned char state[][4], unsigned char k[][4]){

int r,c;

for(c=0; c<4; c++){

for(r=0; r<4; r++){

state[r][c] ^= k[r][c];

}

}

}

//逆字节替代

void AES::InvSubBytes(unsigned char state[][4]){

int r,c;

for(r=0; r<4; r++){

for(c=0; c<4; c++){

state[r][c] = InvSbox[state[r][c]];

}

}

}

//逆行移位变换

void AES::InvShiftRows(unsigned char state[][4]){

unsigned char t[4];

int r,c;

for(r=1; r<4; r++){

for(c=0; c<4; c++){

t[c] = state[r][(c-r+4)%4];

}

for(c=0; c<4; c++){

state[r][c] = t[c];

}

}

}

//逆列混淆变换

void AES::InvMixColumns(unsigned char state[][4]){

unsigned char t[4];

int r,c;

for(c=0; c< 4; c++){

for(r=0; r<4; r++){

t[r] = state[r][c];

}

for(r=0; r<4; r++){

state[r][c] = FFmul(0x0e, t[r])

^ FFmul(0x0b, t[(r+1)%4])

^ FFmul(0x0d, t[(r+2)%4])

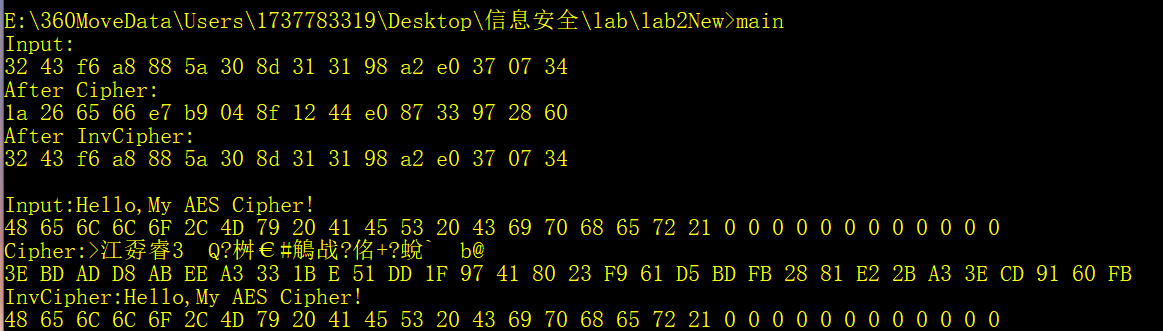
^ FFmul(0x09, t[(r+3)%4]);

}

}

}

**五、程序运行后的结果截图**

****

# 实验三 RSA实验

**一、 实验目的**

实现RSA的加密(Encryption)和解密(Decryption)

**二、实验要求**

**Operation:**

**1.** Key Generation:

**(1)** Select two large prime numbers p and q randomly, such that p is not equal to q

(2) Compute n such that n = p \* q

(3) Compute f(n) = f(p) \* f(q) = (p - 1) \* (q - 1) where f is Euler’s totient function

(4) Select an number **e** such that 1 < **e <** f(n) and gcd (e, f(n)) == 1

where e and f(n) are co-prime

(5) Compute d as the multiplicative inverse of e(mod(f(n))) i.e. de = 1 mod f(n)

(6) Publish the pair PU = (e, n) as participant’s public key

Keep the pair PR = (d, n) as the participant’s private key

Algorithm 1 : findE(phi\_n)

Begin

e <- 0

do

begin

choose an integer number e

(e must be co-prime of phi\_n)

while(!checkCoprime(phi\_n, e))

end do-while

return e

End

Algorithm2 : FindD(phi\_n,e)

Begin

Local variables: a, b , x , y , u , v , m , n , q , r , gcd

a <- phi\_n

b <- e

x <- 0

y <- 1

u <- 1

v <- 0

gcd <- b

while(a != 0)

begin

q <- gcd / a

r <- gcd % a

m <- x – u \* q

n <- y – v \* q

gcd <- a

a <- r

x <- u

y <- v

u <- m

v <- n

end while

if y < 1

begin

y <- phi\_n + y

end if

return y

End

Algorithm 3 : Generate(&n,&e,&d)

Begin

local variables : p , q , phi\_n

Enter two prime numbers and stored them in p and q respective

n <- multiply(p , q)

phi\_n <- multiply(p – 1 , q - 1)

e <- findE(phi\_n)

d <- findD(phi\_n,e)

(e,n) => public key

(d,n) => private key

End

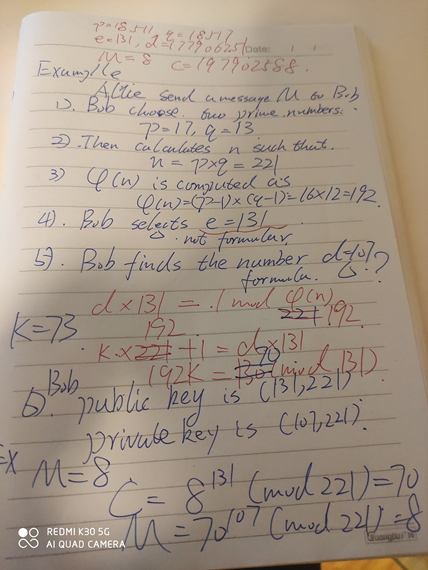
2. Encryption

C = M ^ e ( mod n)

3. Decryption

M = C ^ d ( mod n)

Example

****

**三、源代码**

**//RSA.h**

#include<cstdio>

#include<cstdlib>

#include<ctime>

#include<cmath>

#include<cstring>

typedef long long LL;

const int MAX\_ROW = 50;

//KEY GENERATION

LL getPrime(){

bool ifPrime = false;

LL a = 0;

int arr[MAX\_ROW];

for(int i = 0; i < MAX\_ROW;++i){

arr[i] = i + 3;

}

while(!ifPrime){

srand((unsigned)time(0));

ifPrime = true;

a = (rand() % 1000) \* 2 + 3;

for(int j = 0;j < MAX\_ROW;j++){

if(a % arr[j] == 0){

ifPrime = false;

break;

}

}

}

return a;

}

bool checkIsPrime(LL n){

if(n < 2) return false;

LL i = 2;

while(i <= n /2){

if(!(n % i)) return false;

i++;

}

return true;

}

bool checkCoPrime(LL n1,LL n2){

LL lowest;

if(n1 > n2) lowest = n1;

else lowest = n2;

LL i = 2;

bool coPrime = true;

while(i < lowest){

if(!(n1 % i) && !(n2 % i)) coPrime = false;

i++;

}

return coPrime;

}

LL findE(LL phi\_n){

LL e = 0;

do{

printf("Choose an integer number e\n");

scanf("%lld",&e);

}while(!checkCoPrime(phi\_n,e));

return e;

}

LL findD(LL phi\_n,LL e){

LL a = phi\_n;

LL b = e;

LL x = 0,y = 1;

LL u = 1,v = 0;

LL gcd = b;

LL m,n,q,r;

while(a != 0){

q = gcd / a;

r = gcd % a;

m = x - u \* q;

n = y - v \* q;

gcd = a;

a = r;

x = u;

y = v;

u = m;

v = n;

}

if(y < 1){

y = phi\_n + y;

}

return y;

}

LL multiply(LL n1,LL n2){

return n1 \* n2;

}

int primeMenu(){

int choice = 0;

do{

printf("------pq------\n");

printf("1:auto\n");

printf("2:myself\n");

printf("choose:\n");

scanf("%d",&choice);

printf("--------------\n");

}while(choice != 1 && choice != 2);

return choice;

}

void autoPrime(LL& p,LL& q){

printf("auto\n");

do{

p = getPrime();

q = getPrime();

}while(p == q);

printf("p:%lld q:%lld\n",p,q);

}

void myselfPrime(LL& p,LL& q){

printf("myself\n");

do{

scanf("%lld",&p);

}while(!checkIsPrime(p));

do{

scanf("%lld",&q);

}while(!checkIsPrime(q));

printf("p:%lld q:%lld\n",p,q);

}

void generateKey(LL& n,LL& e,LL& d){

LL p,q,phi\_n;

int choice = primeMenu();

switch(choice){

case 1 :

autoPrime(p,q);

break;

case 2 :

myselfPrime(p,q);

break;

}

n = multiply(p,q);

printf("n:%lld\n",n);

phi\_n = multiply(p-1,q-1);

printf("phi\_n:%lld\n",phi\_n);

e = findE(phi\_n);

printf("e:%lld\n",e);

d = findD(phi\_n,e);

printf("d:%lld\n",d);

}

//ENCRYPT DECRYPT

LL encDec(LL m,LL ed,LL n){

LL rem;

LL x = 1;

while(ed != 0){

rem = ed % 2;

ed = ed / 2;

if(rem == 1) x = (x \* m) % n;

m = (m \* m) % n;

}

return x;

}

void encDecNum(LL ed,LL n){

LL m;

printf("Enter a message number:\n");

scanf("%lld",&m);

LL c = encDec(m,ed,n);

printf("M/C:%lld C/M:%lld\n",m,c);

}

**//main.cpp**

#include "RSA.h"

int main(){

LL n,e,d;

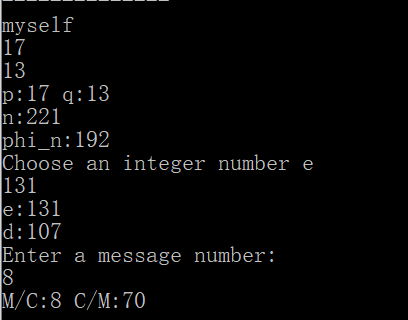
generateKey(n,e,d);

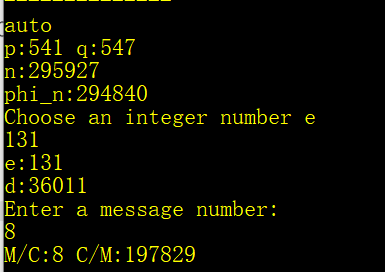
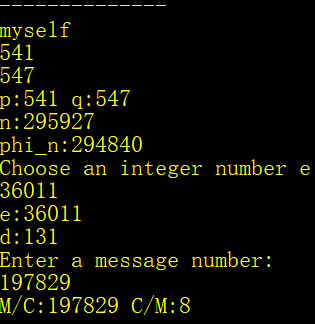
encDecNum(e,n);

return 0;

}

**四、程序运行后的结果截图**

****



# 实验四 Wireshark实验

**一、实验目的**

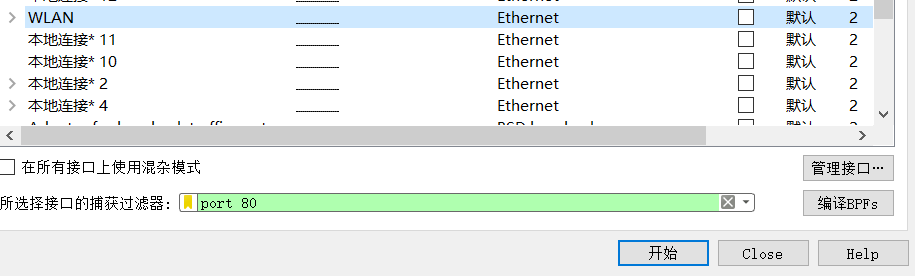
* 了解网络侦听的目的、过程以及基本手段。对协议分析有基本了解。
* 学会使用Wireshark工具。

**二、实验要求**

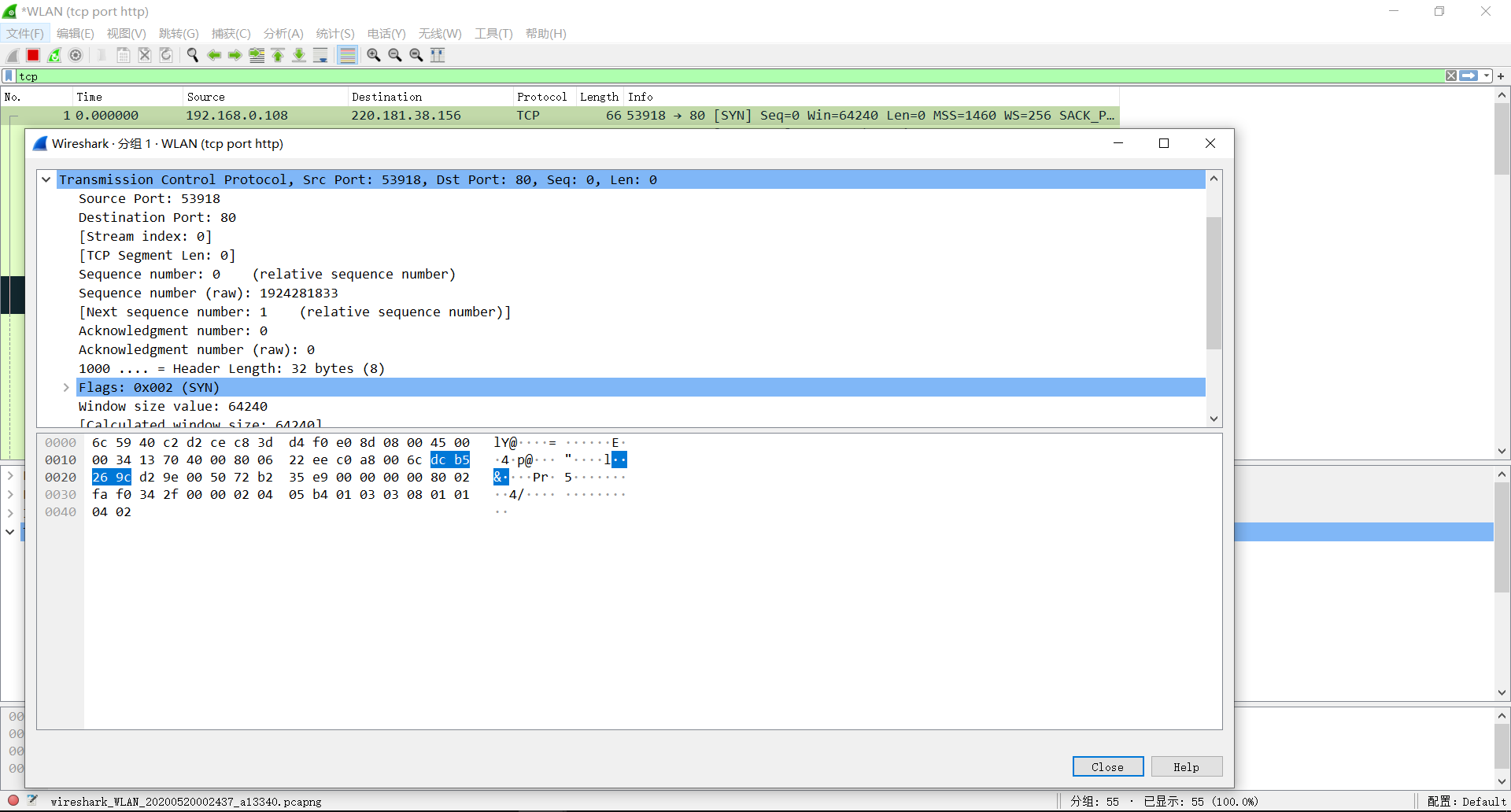
* 1. 正确安装Wireshark软件，并且能够熟练使用Wireshark的主要功能模块；
  + 菜单栏
  + 工具栏
  + 显示过滤器
  + Packet List封包列表
  + Packet Details包详情面板
* 2. 熟悉Wireshark软件的包过滤设置和数据显示功能使用；
  + 捕捉的方法
  + 使用过滤器
* 3. 利用Wireshark抓取TCP协议的数据包，深入分析“TCP三次握手协议”。
  + 确定目标地址
  + 启动抓包
  + 使用FTP服务
  + 通过显示过滤器得到相关数据包
  + 分析数据包

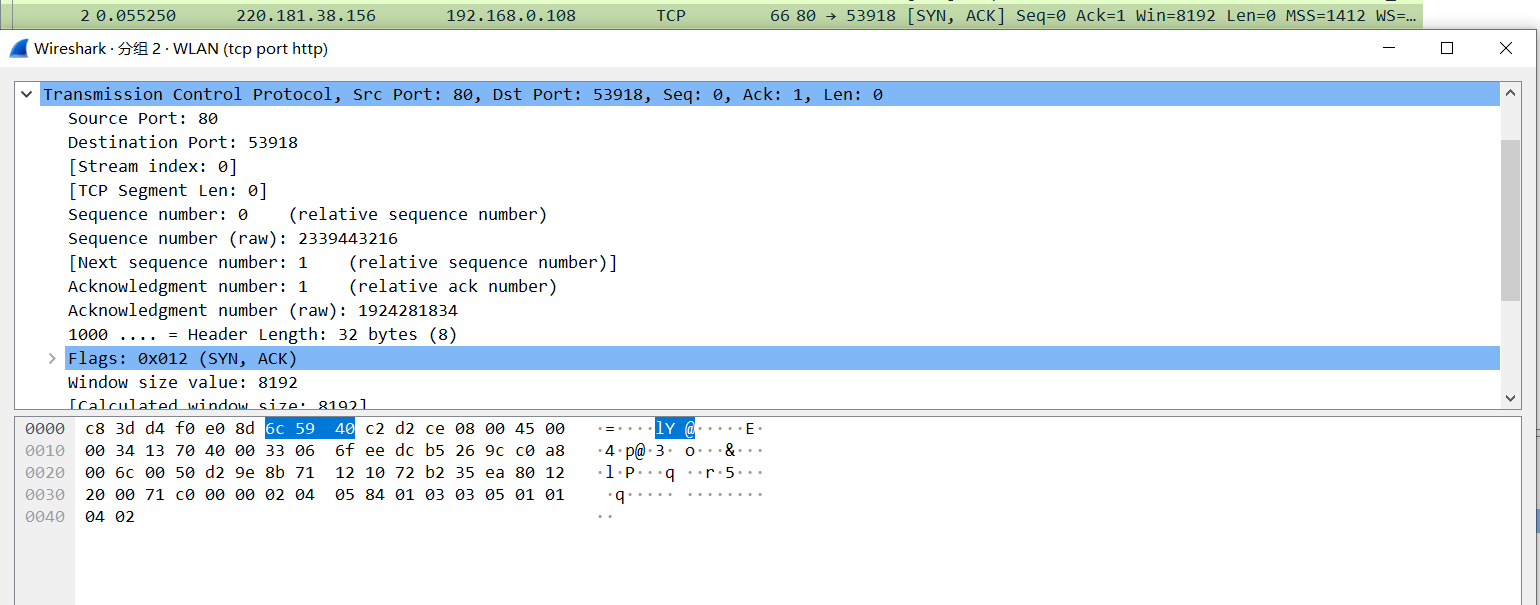
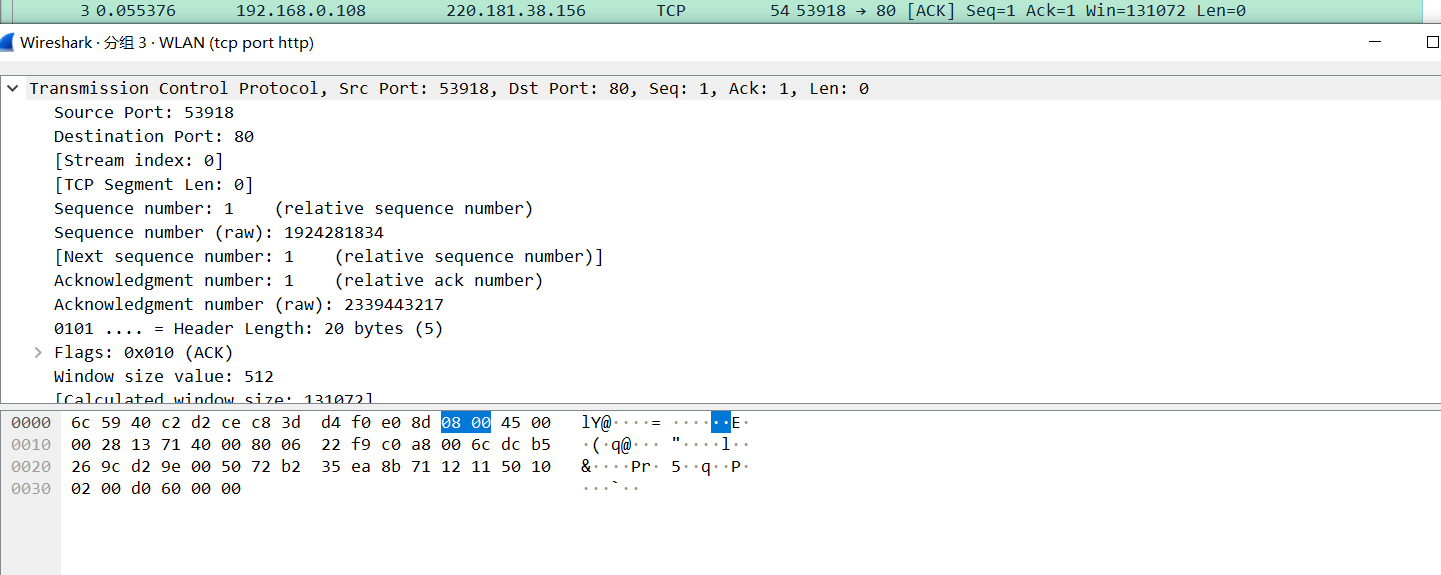
**三、实验内容及结果（写出源程序和运行程序后的结果截图）**

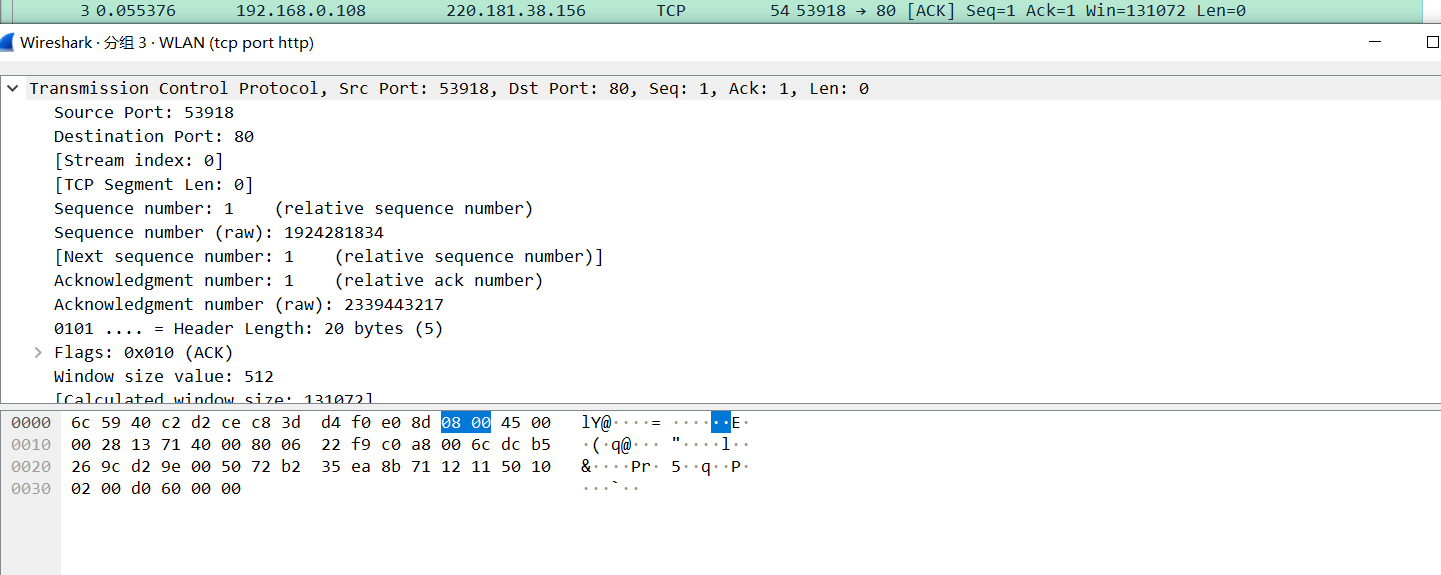
* 1. 实验前Wireshark过滤器设置



* 2. TCP第一次握手



* 3. TCP第二次握手
* 4. TCP第三次握手



* 5. 响应

