南京邮电大学

网络安全课程设计报告

题　　目：_____密钥管理系统_____

专　　业：_____网络工程_____

姓　　名：_____陈慧_____

学　　号：_____B16070404_____

指导教师：_____余金蓉_____

2018 年　　11 月　　29 日

目　　录

1. 前言

本次网络安全课程设计，总共包括两个阶段。第一阶段，体验性实验，即要求我们体验目前网络安全方面的一些软件的使用和功能实现，从而对 ARP 攻击、DOS 攻击、密钥生成、数据加密、水印管理等基本操作有初步的认识。第二阶段，实践性实验，要求我们根据第一阶段对于各种功能的理解，选择出合适自己的题目，进行开发。通过网络安全课程的学习，我和队友胡雪然同学对于密钥生成算法，加解密算法等网络安全知识有了一定的基础，由此选择《密钥管理系统》作为我们本次实验的课题。

本文的设计方案，详细的介绍了密钥生成、密钥保存、密钥加密、密钥管理这四个部分的内容。在系统的开发中，我们采用 C++这一面向对象的编程语言，并综合使用文件存储、调用操作，最终成功完成整个系统的构建。

2. 系统分析

2.1 系统需求重述

设计一个密钥管理系统，可以完成密钥生成和管理。具体功能如下：

（1）可以生成随机数，必要时生成大素数。即需要我们设计一个较好的随机数发生器，并且有素数生成功能。

（2）检测密钥和密码的安全性能。即需要了解密钥安全性能的判定标准。

（3）密钥的加密保存和管理。即需要我们实现一个功能较好的加密算法，完成密钥的加密存储。

（4）生成公钥对。即需要我们联系问题一中的生成大素数功能，运用 RSA 密钥生成算法，生成系统分配给用户的密钥对。

2.2 系统功能分析

对一个密钥管理系统而言，可以分为管理员用户和普通用户两类群体。普通用户可以从系统中获取密钥、知晓该密钥的安全性能、使用[illegible]系统分配的密钥进行加密保存、随时解密密钥、利用密钥加解密消息等。管[illegible]更新密钥库的密钥、查看用户分配密钥情况、用户离开删除相应密钥等。具体用例图如下：

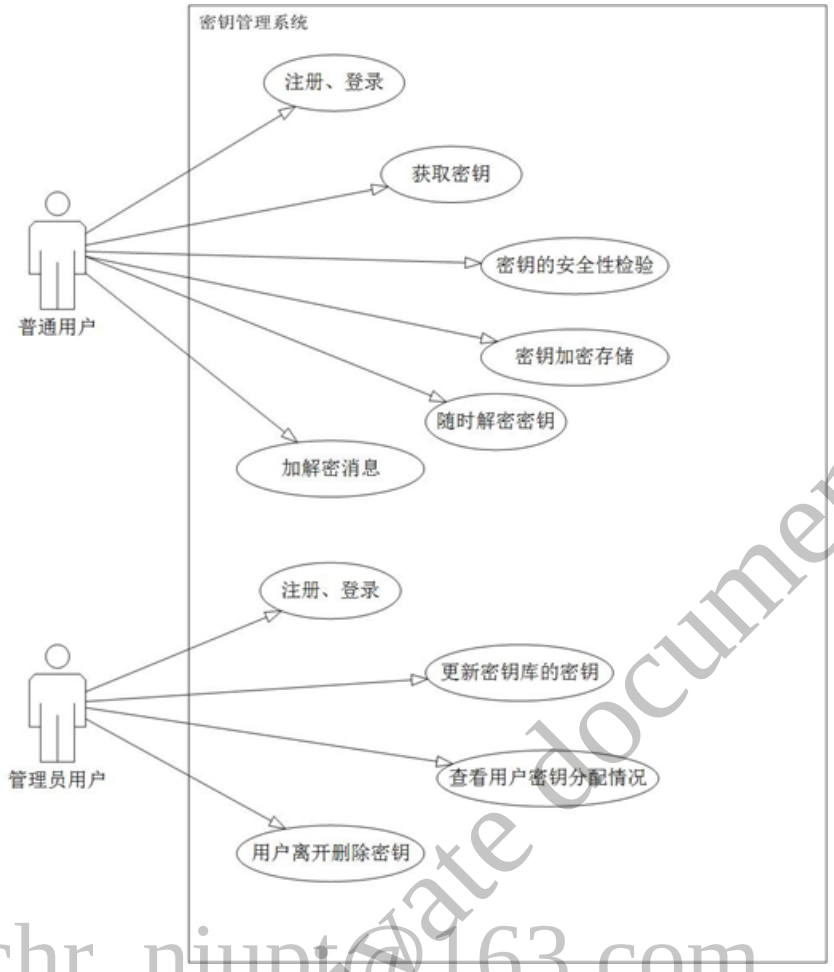

图 2-1 系统分析用例图

对于问题中说明的各种需求，本文进行如下分析：

2.2.1 问题一分析

对于一个密码系统来说，密码随机数常常被用作密钥，补充信息和初始化向量。对于每一个组件来说，使用一个好的随机数发生器是必要的。结合问题四，可以生成一个大素数是使用很多公钥算法重要前提。为了判断一个数字是否为素数，我们可以采用米勒拉宾法，然而由于米勒拉宾法本身算法存在不足，通过测试的数有 25%的概率为非素数，因此多次调用米勒拉宾函数进行测试可以降低非素数存在的概率，运算前使用小素数表可以有效地提高程序效率。因此决定使用米勒拉宾-小素数表复合筛选素数的方法。

2.2.2 问题二分析

一个密码算法的强弱除了与其本身的数学模型有关外，还与其输入密钥的一些特性密切相关。通过上网搜索资料，我们了解了对密钥密码的安全性分析的一些要素，并最终确定了判断密钥强度的具体实现方法。

2.2.3 问题三分析

密钥的加密保存方法有多种，通过我们网络安全课程的学习，我们学到了 DES、3DES、AES、RC4 等对称加密算法和包括 RSA、Diffie-Hellman 算法在内的多种非对称加密算法。

基于前期在网络安全课程中对于密钥加密算法的学习，本方案决定选择 RC4 算法，因为它是一种基于流加密的算法，算法简单，执行速度快。并且密钥长度是可变的，可变范围为 1-256 字节(8-2048 比特)，在现在技术支持的前提下，当密钥长度为 128 比特时，用暴力法搜索密钥也不太可行，所以能够预见 RC4 的密钥范围任然能够在今后相当长的时间里抵御暴力搜索密钥的攻击。并且到现在，也没有找到对于 128bit 密钥长度的 RC4 加密算法的有效攻击方法。所以 RC4 加密算法相较于其他加密算法具有更强的安全性。

2.2.4 问题四分析

题目要求我们生成公钥对，意即我们需要采用某种非对称加密算法来实现此功能。经讨论后我们采用了 RSA 算法。RSA 是目前最有影响力和最常用的公钥加密算法，它能够抵抗到目前为止已知的绝大多数密码攻击，考虑到如今使用 RSA 算法的用户人数众多，因此使用 RSA 算法是具有广泛的应用意义的。虽然 RSA 算法也有其缺点，但我们认为它还是可以很好的满足本系统的需要。

2.3 系统功能分解及合作分工介绍

作为一个密钥管理系统，应该包括密钥生成，密钥分发，密钥加密保存，查看分发给用户的密钥，用户解密密钥，用户使用密钥加解密数据，用户离开删除信息这七大功能。基于 2.2 中对问题的分析，加上一个密钥管理系统所需的以上七大功能，本人将整个系统设计进行模块化的分解，分解为以下 10 个小模块：

（1） 系统架构设计
（2） 随机数及大素数的生成，密钥对生成——RSA 密钥生成算法
（3） 系统密钥库建立
（4） 密钥安全性判断
（5） 密钥加密保存——RC4 加解密算法
（6） 分发密钥
（7） 用户密钥对加密保存，解密读取
（8） 密钥对运用——加解密用户指定内容——RSA 数据加解密
（9） 管理人员查看密钥分发情况
（10）用户离开，删除库中分配给该用户的密钥

本人主要负责的是：（1）系统架构设计；（2）随机数及大素数的生成；（3）系统密钥库建立；（5）密钥加密保存；（6）分发密钥；（7）用户密钥对加密保存，解密读取；（9）管理人员查看密钥分发情况；（10）用户离开，删除库中分配给该用户的密钥。对于模块中的（2）RSA 密钥对生成和（8）用户指定数据加解密，（4）密钥安全性检验，都由队友胡雪然完成，在本文中就不再介绍

3. 功能设计及特色技术实现

3.1 系统结构设计

系统分为管理员用户模块和普通用户模块两种。首先，由系统后台管理员进行“密钥库”的创建。接着，普通用户登录系统后即可进行密钥分发和安全性评判，该用户得到密钥对后，选择密钥加密存储选项，即将密钥加密保存于系统“已分配密钥保存区”。然后等到该用户需要使用密钥时，选择密钥解密选项，此时只有输入自己加密时的指定密钥，才能正确解密出系统分配给该用户的密钥对，从而进行信息的加解密操作。最后如果该用户离开，系统管理员即对“已分配密钥保存区”进行修改，删除该用户的密钥信息。系统结构流程图

如下：

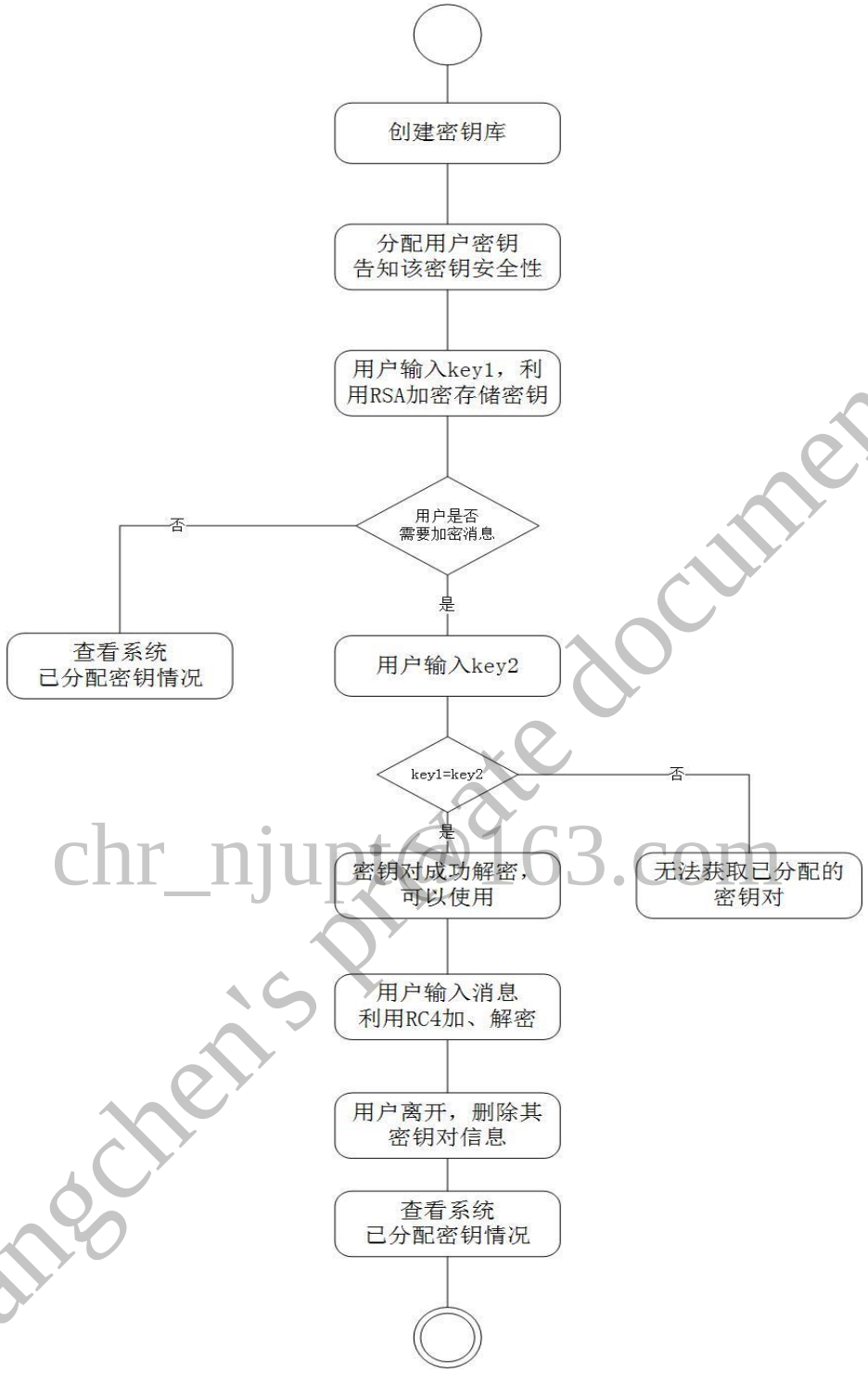

图 3-1 系统结构流程图

3.2 随机数及大素数的生成

为了生成一个随机数，本方案建立了一个随机数类，其私有部分是一个无符号的 64 位整型变量，命名为 randSeed；公有部分由两个函数组成，分别是 RandNumber 和 Random。这三部分相互配合，最终实现随机数的生成功能。

为了生成一个素数，我将素数生成功能划分为素数筛选和素数输出两部分来实现。在

素数的筛选方面，我选择了米勒拉宾法和小素数表配合使用的方式。

米勒拉宾法是筛选素数的一项有效方法，它是一个判断素性的多项式时间概率算法，这个算法是建立在费马小定理之上的，其具体的内容是：要测试N是否为质数，首先将$\mathrm{N}-1$分解为$2^s \times d$。在每次测试开始时，先随机选一个介于$[1,\mathrm{n}-1]$的整数 a，之后如果对所有的$\mathrm{rin}[0,\mathrm{s}-1]$，若$a^d(\mathrm{mod\ N}) <> 1$且$a^{2^{rd}}(\mathrm{mod\ N}) <> -1$，则N是合数。否则，N有$3/4$的机率为质数。虽然米勒拉宾法不能确定一个数一定是质数，但是如果尽可能地提高运行该算法的次数，确定一个数为素数的可能性也会随之扩大。在程序当中，米勒拉宾法的算法核心函数是 RabinMillerKnl 函数，该函数实现了上述内容。

为了提高整个函数的效率，本方案中引入了小素数表，这个素数表包含 3-100 之间的所有素数，在运行米勒拉宾核心函数 RabinMillerKnl 之前先用小素数表筛选一遍，可以减少 RabinMillerKnl 函数调用的次数，节省系统资源。这一部分功能由 RabinMiller 函数完成。一个数如果能全部通过米勒拉宾素数测试返回 1，否则返回 0。完整的素数筛选流程如下：

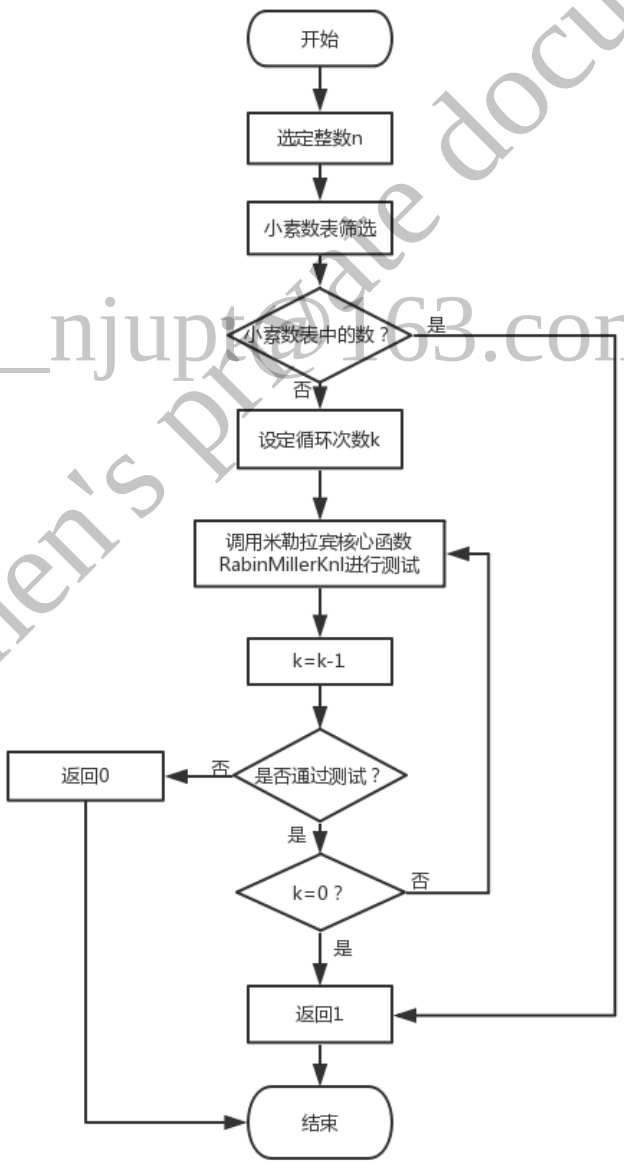

图 3-2 素数检测算法流程图

在完成了随机数生成和素数检测两部分的代码之后，我就着手于输出一个二进制位的素数。随机素数输出函数我命名为 RandomPrime。在这个函数中，首先定义一个无符号整型变量 base。因为二进制数仅由 0 和 1 构成，为了保证该素数位数一定，需要使得 base 的首位置 1。此步操作之后，base 与一个随机数相加。易知偶数不是素数（最大公约数至少为 2），因此 base 必须设定为奇数，即最后一位置 1。完成此项操作后，调用素数测试函数 RabinMiller30 次，若 base 通过全部测试，退出循环，输出我们需要的素数 base，否则，重复上述步骤，直至出现一个合适的素数 base。这一部分的程序框图如下：

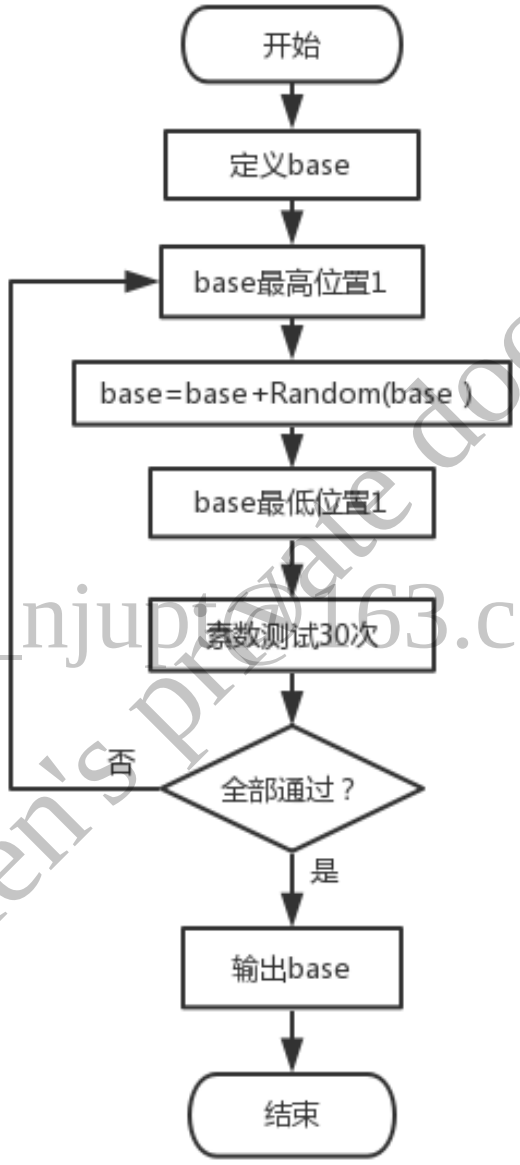

图 3-3 素数生成算法流程图

3.3 密钥管理

本方案在密钥管理系统的设计中，针对密钥这一主体，将其分为密钥分发、密钥保存、密钥查看、密钥删除和密钥应用这五个方面。从而实现管理员用户和普通用户的基本功能，满足他们对于密钥管理系统的需求。

3.3.1 密钥分发

对于整个系统，我使用两个文件进行密钥管理，分别为“密钥库”和“密钥保存”。在程

序中，设定每次启动时，生成 1000 个密钥对，以（公钥，私钥）的形式存储于“密钥库”文件中，为了后期取密钥的方便，设置其为按行存放，即每一行只存放一组密钥对。

当普通用户进入系统，立即给每个用户根据时间顺序设置一个编号 yonghu，用于后续密钥存储。当用户选择“1.密钥分发+密钥安全性评判”选项，系统以读的形式打开文件“密钥库”，随机选取某一行，作为该用户分配的密钥显示屏幕，同时进行密钥安全性判断，由于 RSA 密钥的公钥是公开的，安全性评判标准主要是私钥长度，由此我们将私钥长度划分区间，从而完成安全性的判断。

在从文件中取密钥的过程中，我选择已“，”作为分隔符，从而实现将公钥和私钥分别放入数组 gongyao[N]，siyao[N]，方便下面对于公钥和私钥的加解密操作。在这之间，也涉及到字符串和数组之间的转换、数组长度计算、字符串长度计算等操作，最终完美实现密钥分发的功能。相应代码实现如下（源码 474 到 500 行）：

```
yonghu++;   //用户编号
i=1;
srand((unsigned int)time(NULL)) ;
while (getline(inff, tmpLine))
{
    if (rand()%i == 0)
        lineData = tmpLine ;
    ++ i ;
}
inff.close() ;    //从密钥库中随机选取一行

strcpy(miyao, lineData.c_str());   //string 类型数据转化为数组型数据
length = sizeof(miyao)/sizeof(char);
i=0;
while (i<length)
{
    while ( miyao[i]!=',' && i==j)
    {
        gongyao[i]=miyao[i];
        i++;
        j++;

    }
siyao[i-j]=miyao[i+1];
i++;
}//实现公钥、私钥分别放入数组 gongyao[i]，miyao[i]
```


3.3.2 密钥保存

基于 2.2 中对于问题三的分析，本方案采用 RC4 加密算法进行密钥保存。RC4 于 1987 年提出，和 DES 算法一样。是一种对称加密算法，也就是说使用的密钥为单钥（或称为私钥）。但不同于 DES 的是。RC4 不是对明文进行分组处理，而是字节流的方式依次加密明文中的每个字节。解密的时候也是依次对密文中的每个字节进行解密。

首先初始化 S 数组$S[0]=0, S[1]=1, \ldots\ldots, S[255]=255$，建立临时数组 T，数组 K 保存用户输入密钥。若密钥 K 的长度为 256 比特，则 T =K，否则对于 *keylen* 字节长度的密

钥，从 K 复制 T 的前 *keylen* 个元素，然后一直重复 K 直到填满 T。相应伪代码如下：

$$for\ i\ =\ 0\ to\ 255\ do$$
$$S[i]\ =\ i$$
$$T[i]\ =\ K[i\ mod\ keylen]$$

然后是用 T 产生 S 的初始置换，从$S[0]$~$S[255]$，对每个$S[i]$，根据由$T[i]$确定的方案，并将$S[i]$置换为 S 的另一字节：

$$j\ =\ 0$$
$$for\ i\ =\ 0\ to\ 255\ do$$
$$j\ =\ (j\ +\ S[i]\ +\ T[i])\ (mod\ 256)$$
$$swap\ (S[i], S[j])$$

接着，执行伪随机子密钥生成算法，选代S[i]的所有元素，并对每个S[i]根据S的当前结构指定的方案将S[i]与S中的另一个字节交换。到达S[255]之后，流程继续，重新从S[0]]开始。相应伪代码如下：

$$i\ =\ j\ =\ 0$$
$$for\ each\ message\ byte\ Mi(Ci)$$
$$i\ =\ (i\ +\ 1)\ (mod\ 256)$$
$$j\ =\ (j\ +\ S[i])\ (mod\ 256)$$
$$swap(S[i], S[j])$$
$$t\ =\ (S[i]\ +\ S[j])\ (mod\ 256)$$
$$K\ =\ S[t]$$

加密时，将 k 值与明文的下一字节做异或。解密时，将 k 值与密文的下一字节异或即可。

具体 RC4 算法逻辑如下图所示：

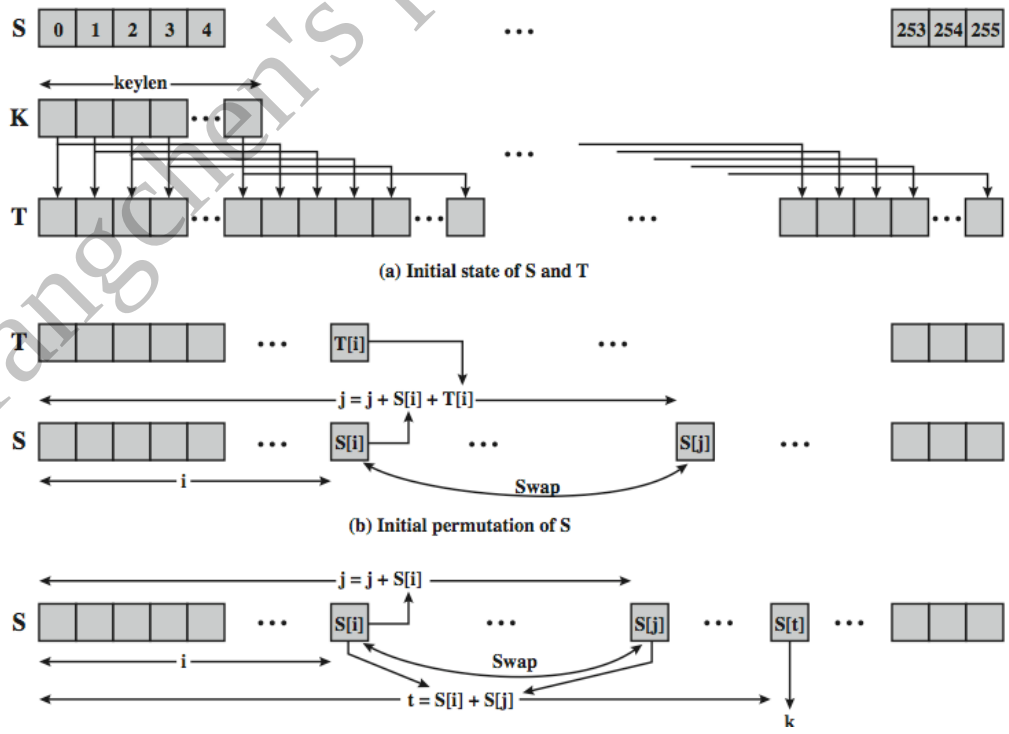

图 3-4　RC4 算法逻辑

用户选择“2.密钥加密存储”选项后，系统提示“请您输入秘密密钥”，用户输入自己指定密钥，作为上述算法中的 K 值，系统后台运行上述算法后，分别显示私钥、公钥加密后的内容。

当用户完成密钥加密后，即开始密钥存储工作。当该用户为第一位用户时，系统创建一个“密钥保存.txt”的文件，接下来的用户存储密钥时只要打开该文件就可以。接着，使用定义的 ofstream 变量 fout1 进行向文件中写数据的操作，即完成加密密钥的保存。相应代码实现如下（源码 608 行）：

```
if(yonghu==1)
{
    ofstream fout1("密钥保存.txt");        //创建一个密钥保存.txt 的文件
}
else
{
    fout1.open( "密钥保存.txt", ios::app );
}
fout1 << yonghu <<"          "<< ciphertext1<<"          "<< ciphertext2<<"\n";
fout1.close();
cout<<"\n\n 第"<<yonghu<<"位用户系统分配的秘钥已经加密存储";
cout<<endl;
```

3.3.3 密钥查看

该功能主要给管理员使用，查看用户已分配密钥情况。在用户完成密钥加密存储的基础上，在主菜单选择“3.查看系统用户保密密钥”选项，由于每个进入系统的用户都会分配一个编号 yonghu，此时，就使用用户编号进行用户密钥查看，在界面输入用户编号后，界面显示“用户编号　公钥加密密文　私钥加密密文”及相应编号用户分配的密钥对情况。

在读指定用户密钥对的过程中，使用 ReadText 自定义函数体，以用户编号作为传递参数，由存储时的规则可知，用户编号数就是其密钥对存放文件的行数，由此即可转化为读取“密钥保存.txt”文件中指定行问题。使用一个循环将文件中所有行都放入数组中即可。具体代码如下（源代码 396 行-409 行）：

```
string ReadText(int line)
{
    ifstream inff("密钥保存.txt") ;
    string strVec[100];
    int i = 0;
    while (!inff.eof())
    {
        string inbuf;
        getline(inff, inbuf, '\n');
        strVec[i] = inbuf;     //  所有行都放入数组
        i = i + 1;
    }
    return strVec[line - 1]; //返回指定行数据
}
```

3.3.4 密钥删除

密钥删除类似于密钥查看，通过输入用户编号进行删除，将文件中指定行的数据置空，即完成用户数据删除操作。具体代码如下：

```
cout <<"请输入您要删除的用户编号";
cin>>sc;
ifstream in;
in.open("密钥保存.txt");//以写的方式打开密钥保存文件
string strFileData = ""; //定义一个空字符串，用于替换
int line = 1;
char lineData[1024] = {0};
while(in.getline(lineData, sizeof(lineData)))//找出用户编号所对应的行
{
    if (line == sc)
    {
        strFileData += "\n";
    }
    else
    {
        strFileData += CharToStr(lineData);
        strFileData += "\n";
    }
    line++;
}
in.close();
ofstream out;
out.open("密钥保存.txt");//以输出的方式打开文件
out.flush();
out<<strFileData;//用空白替换原文件中指定行的数据
out.close();
```

操作完以上步骤，即完成某编号用户数据删除操作，为了验证是否确实删除，可以再次进行查看，输入刚刚删除用户的编号，进行查询，会发现输出内容为空，即表明删除成功。

3.4 RC4 算法实现——密钥加密、解密

在 3.3.2 中，已经介绍了 RC4 算法的基本内容和实现。在密钥的保存方面，用户输入加密密钥，通过 RC4 加密算法，实现系统分配密钥对的加密存储。当用户需要使用该密钥对进行加解密信息时，选择主菜单中的“4.解密用户保密密钥”选项，界面跳转，此时需要用户输入刚刚加密时输入的加密密钥，将此作为解密密钥，通过 RC4 解密算法进行密钥对的解密。如果解密密钥等于加密密钥，弹出正确解密后的密钥结果，否则，输出乱码，无法获得系统分配给指定用户的密钥。以此保证密钥保存的安全性。

4. 所遇到的问题及分析解决

（1）C++中进行文件流操作时，进行写文件操作时，写入的内容需要在文件关闭后才能正式生效，通过查阅资料，发现流式操作是通过缓冲区完成的，一般是围绕一个指针进行。与之对应的文件操作是 I/O 操作，它是通过直接存取文件来完成对文件的处理的。

（2）C++中使用 string 类型数据指定文件操作的文件名时，无法直接使用，需要使用.c_str()函数，比如：

```
string username;
cin >> username;
ofstream userfile;
userfile.open(username.c_str(),ios::out);
```

如果直接使用 userfile.open(username,ios::out)；会报 No matching function 的错误。查阅资料发现，open 函数的定义为：

```
open(const char* __s,ios_base::openmode __mode = ios_base::in | ios_base::out)
```

文件名位置对应的参数四 const char *，而 username 为 string 数据类型，因此会报错。在 username.c_str()方法后，返回的是指向正规 C 字符串的指针。

（3）使用流式操作读文件进行赋值时，使用的是流提取符号>>，通过查阅资料，发现 C++把输入和输出当做字节流，因此在输出时，程序将字节插入到输出流中，因此对于数据类型转换的要求降低，string 类型变量可以载入原本以 int 型写入文件的数据

（4）传统的 C 语言中比较字符串需要使用 strcmp 函数，而在 C++中，string 数据可以直接使用 == 运算符进行判断，查看 strcmp 函数定义：

```
int strcmp(char *str1,char *str2)
```

实际上比较的是两个指针，而 C++中的 == 运算符实质上是有可能会把两个相同的字符串字面量优化为存在同一个地方，所以在进行比较时，还是推荐使用 strcmp 函数

（5）ifstream 和 ofstream 是 fstream 的两个子类，ifstream 是指以输入方式打开文件，ofstream 是指以输出方式打开文件，ifstream 模式中 open 的默认模式为 ios_base::in，ofstream 模式中 open 的默认模式为 ios_base::out。因此在编码过程中，在不同的位置，需要以读的形式打开文件，需要以写的形式打开文件，需要细心考虑 ifstream 和 ofstream 的区别。

（6）在编码过程中，由于不同的模块使用的数据类型不同，比如当数据存入文件时，每一行定义为一个 string 数，要想分别得到公钥和私钥，就需要将字符串转化为数组，否则参数传递失败，查阅资料发现 strcpy(miyao, lineData.c_str())；即可很好的将字符串转化为数组。在整个过程中，还涉及了数组转化为 string 类型，字符串转化成 string 类型等类型转化操作，从而使系统顺利实现功能。

5. 测试

（1）系统界面设计

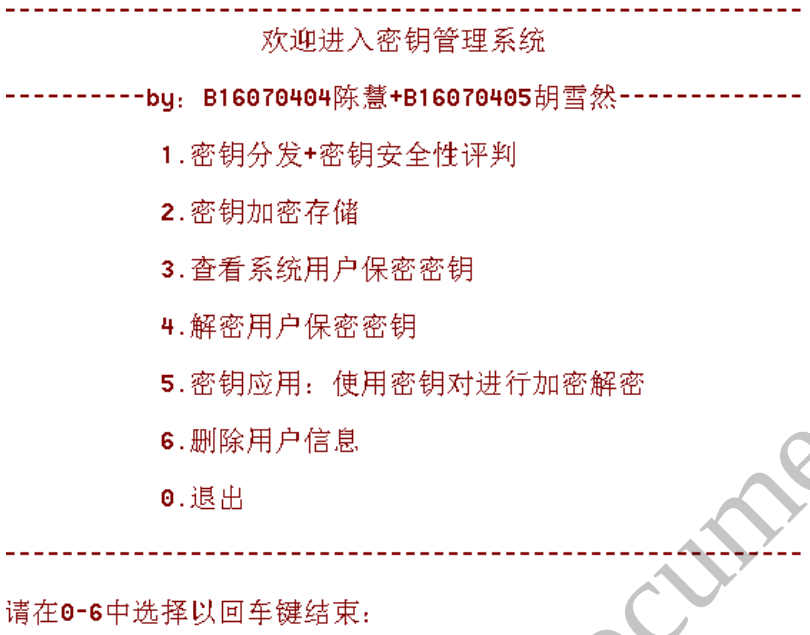

图 5-1 系统主菜单

（2）密钥分发及密钥安全性判断

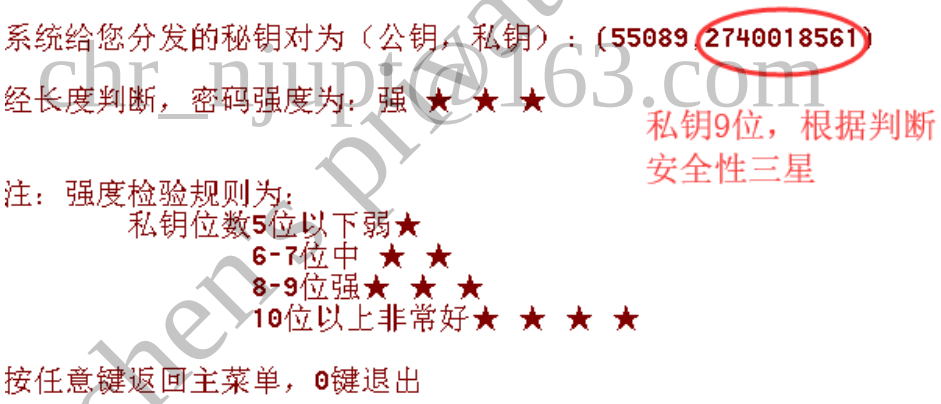

图 5-2 密钥分发及安全性判断

当用户选择分发密钥选项时，“密钥库”文件自动生成，并从中随机选择一对密钥分发：

图 5-3 “密钥库”文件

（3）密钥加密保存

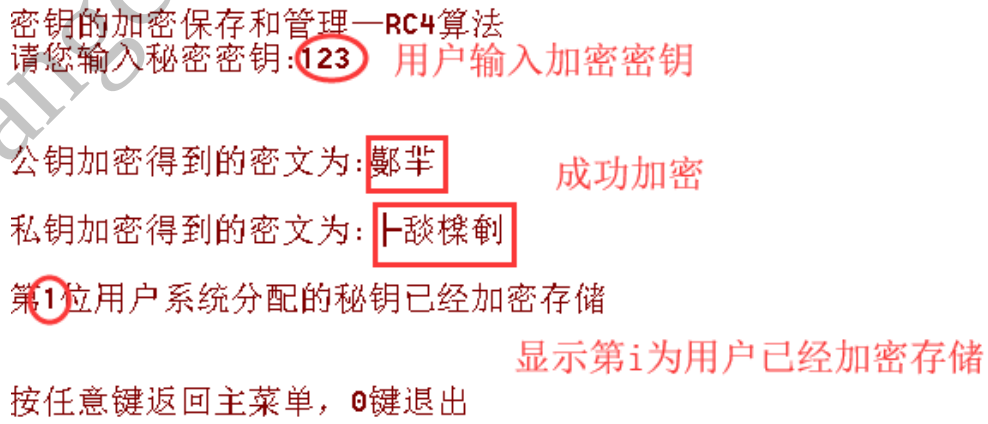

图 5-4　RC4 加密密钥

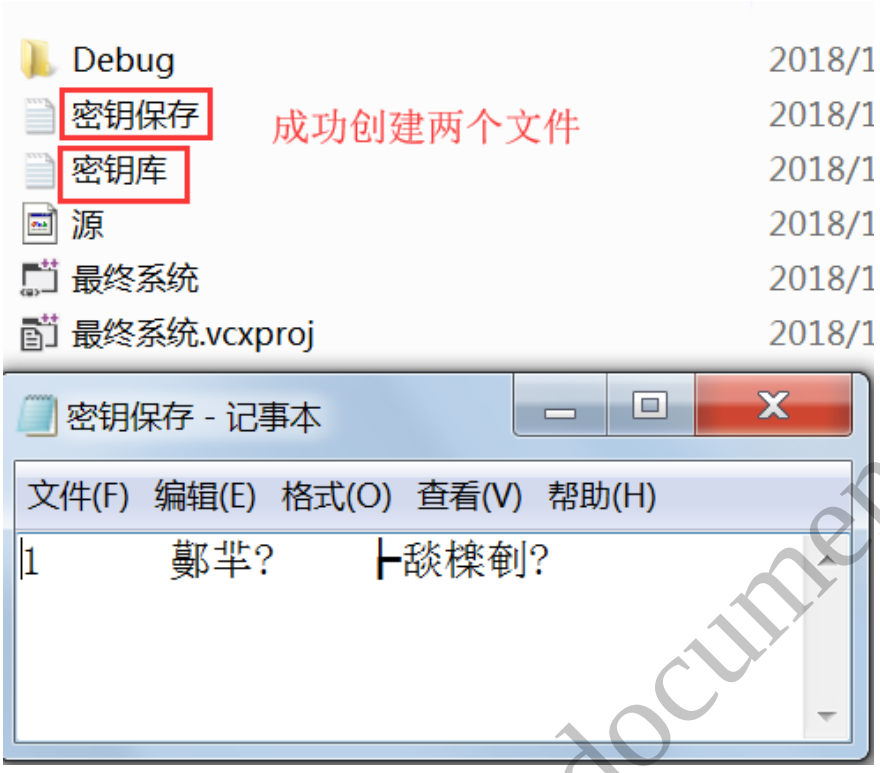

图 5-5 加密密钥存入“密钥保存”文件

（4）用户解密密钥

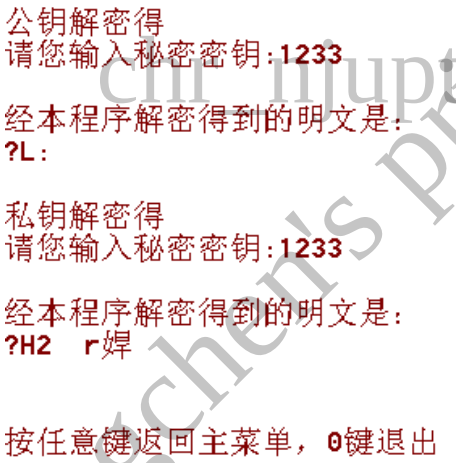

图 5-6 密码错误，无法获得

图 5-7 密码正确，成功解密

（5）查看用户保密密钥

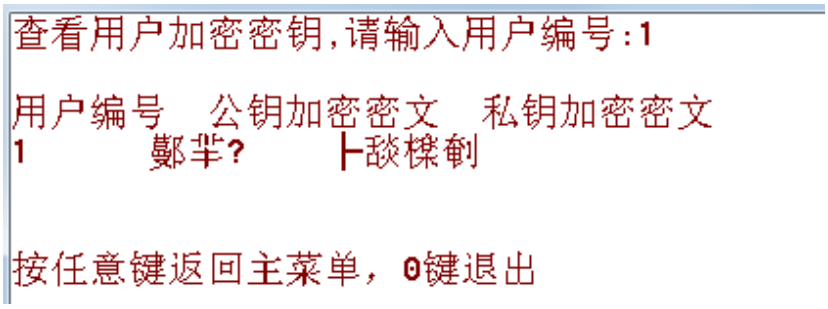

图 5-8 查看用户加密密钥

（6）删除用户

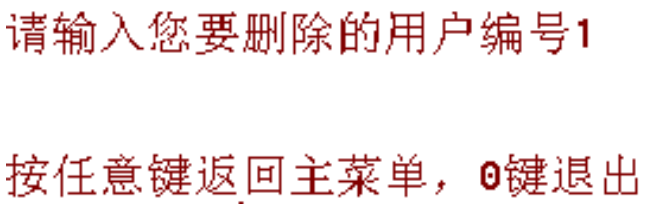

图 5-9 选择删除用户编号

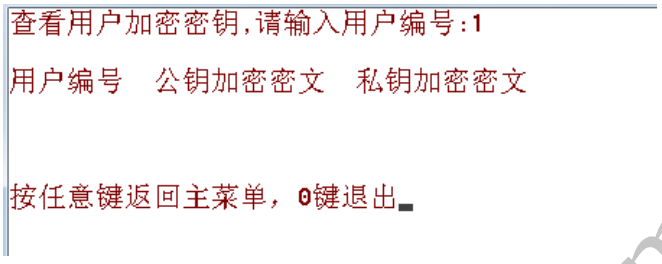

图 5-10 再次查看该用户密钥情况，没有显示，成功删除

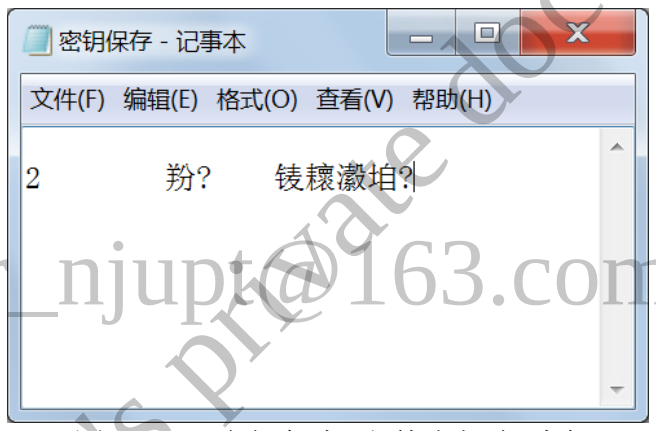

图 5-11 “密钥保存”文件也相应删除

（7）密钥应用

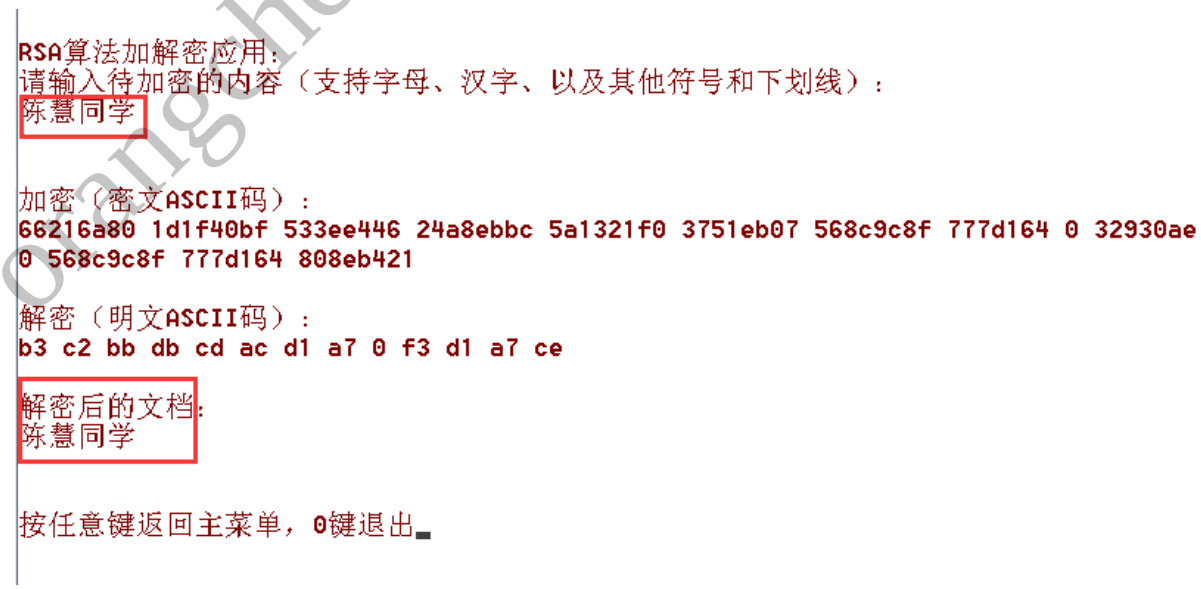

图 5-12 RSA 密钥加解密信息

所有功能测试成功，运行速度也很快，表明本方案成功完成系统全部功能。

6. 结论

通过本次网络安全课程设计，对于《密钥管理系统》这一课题，我们小组基本完成所需功能，成功实现了密钥生成，密钥分发，密钥加密保存，查看分发给用户的密钥，用户解密密钥，用户使用密钥加解密数据，用户离开删除信息这七大功能。我个人对于分工给自己的工作，也已成功完成。但是结合现实的技术和老师的意见，本方案系统还存在以下问题：

① 密钥库太小，仅用文档进行存放，不能满足大量密钥的存放工作，在后期需要我们认真学习数据库编程，用数据库代替文档存放数据。

② 用户仅用一个编号进行标记，没有考虑实际情况下，用户可能不记得自己编号，管理员也可能不知道用户和编号一一对应的关系，所以后期需要对于数据结构的构建和数据库用户属性的存储方面再深入学习，让系统更加人性化。

③ 对于一个系统，相应的图形化界面也是必不可少的，可是由于时间紧张，本方案还没有实现图形化界面的设计，但将终端主界面和流程界面设计的基本清晰流畅。在后期的设计中，需要使用 MFC 编程代替 C++编程。

最后，通过本次试验，我对于 RC4 算法，文档操作，系统构建等方面有了更深刻的认识。在编程的过程中，我体会到面对困难，一定不能退缩，需要我们静下心来认真思考，不懂的及时查找资料，在学习和操作中，解决问题。并且，在整个编程过程中，需要有条理，有逻辑，要细心的构建体系，只有各方面都做好了，最后的程序才能成功运行。

7. 参考文献

[1] 史欣慧.基于 KMIP 协议的密钥管理系统的设计与实现[D].山东大学.2018.

[2] 苏威积，汤敬浩，李剑.一种对称密钥的密钥管理方法及系统[J].信息安全研究.第 4 卷第 1 期.

8. 附录

8.1 安装说明

本“密钥管理系统”可以在 Visual Studio 2012 版本的编程环境中运行。

8.2 源代码

详细代码见源码文件

	评分项	优秀	良好	中等	差
评分	遵守机房规章制度				
	实验原理分析与设计				
	课题功能实现情况				
	设计验收与答辩				
	课程设计报告书写				
简短评语					
评分等级					
备注					

教师签名：_____

_____年____月____日