

**课 程 设 计 报 告**

**题目： 密码学课程设计**

**课程名称： 密码学课程设计**

**专业班级： 网安2002班**

**学 号： U202012043**

**姓 名： 范启航**

**指导教师：**

**报告日期：**

**教师评语：**

**分数：**

**网络空间安全学院**

目录

[一、设计过程 1](#_Toc3375)

[1.1 SPN实现（每一个题目不要超过一面） 1](#_Toc19395)

[1.2线性分析 2](#_Toc7230)

[1.3差分分析 3](#_Toc18077)

[1.4 SPN增强 4](#_Toc29376)

[1.5 RSA参数生成 5](#_Toc12421)

[1.6 模重复平方 6](#_Toc9431)

[1.7 中国剩余定理 7](#_Toc8811)

[1.8 PKCS7 8](#_Toc12226)

[二、实验心得 9](#_Toc29813)

[三、对课程设计内容和过程的建议 9](#_Toc9064)

## 一、设计过程

### 1.1 SPN实现（每一个题目不要超过一面）

（1）设计内容

SPN(代换-置换网络)是一个典型的迭代密码，这种密码明确定义了一个轮函数和一个秘钥编排方案，一个明文的加密将经过Nr轮相似的过程。

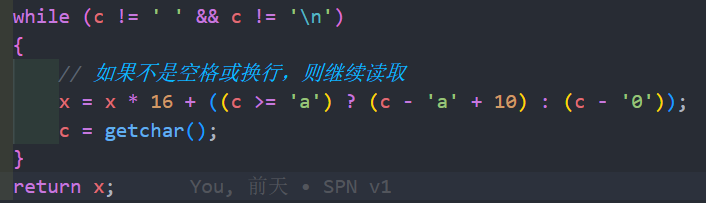
本题中将使用32位秘钥与16位明文，经过SPN加密得到16 位密文，并对密文做修改后，经过SPN解密得到16位明文。

本题中的秘钥为32位，轮秘钥为16位，共有5个，由秘钥生成，S盒为4位，P盒为16位。加密过程中，明文先通过3轮白化-S盒代换-P盒置换。最后一轮先白化后进行S盒代换，将结果与最后一个轮秘钥异或后得到结果。

（2）设计过程

SPN加解密的具体实现包括以下几个部分：1. 数据的读入输出2. 轮秘钥的生成 3. 白化 4. S盒代换 5. P盒置换 6. SPN主程序

1. 数据的读入输出：由于scanf,printf效率较低，故对循环getchar()进行封装实现对十六进制数字的快速读入



1. 轮秘钥生成采用移位与异或的方法，第一轮秘钥为0-16位，第二轮秘钥为4-20位，以此类推，第五轮秘钥为16-32位。
2. 白化采用异或操作，并对其进行封装。
3. S盒置换使用循环的方法，将明文或密文分为4组，分别对每一组进行代换后再将其使用或运算结合。
4. P盒置换使用循环的方法，i从1到16，先将第i位移动到最后一位再将其移动到P盒中指定的位置。速度较慢，优化方法：因数据量较小，采用打表输出。



1. SPN主程序，按照SPN流程，接收参数秘钥，明文或密文，加密或解密标志，依次调用以上函数实现SPN加解密。

（3）小结

运行结果，速度较快，最后一题为600ms左右，不足之处在于为保证速度，代码中存在冗余且不宜理解的地方。

### 1.2线性分析

1. 设计内容

线性密码分析的思想，原则上该思想可以被应用于任何迭代密码。假使能够在一个明文比特子集与最后一轮即将进行代换的输入状态比特子集之间找到一个线性关系，即存在一个比特子集使得其中元素的异或表现出非随机的分布（比如，该异或值以偏离1/2的概率取0）。

1. 设计过程

线性分析的具体实现包括以下几个部分：1. 对S盒分析，手动寻找两条线性逼近链。 2.读入大量明密文对。3. 分别对两条链进行线性分析，得到两个子秘钥。4. 穷举前16位秘钥与子秘钥组合进行测试，得出最终秘钥。

1. S盒分析，可采用手动或编程方法测试，选取一条链关联两个子秘钥，另一条链关联3个或4个子秘钥。

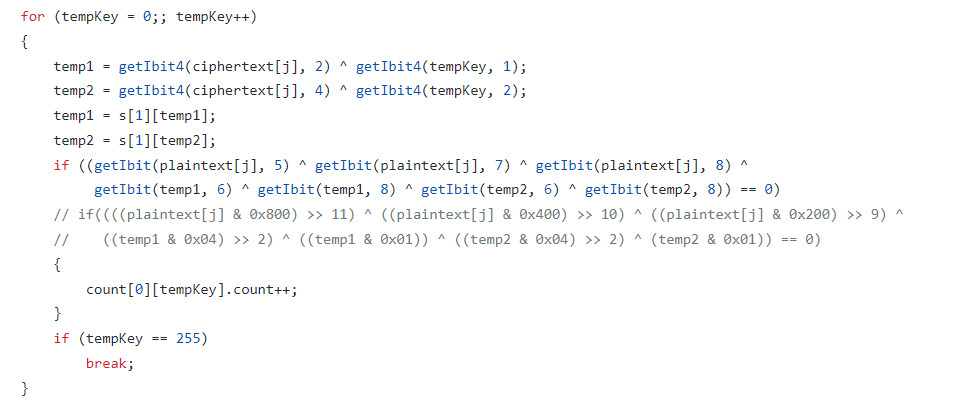


第一条链

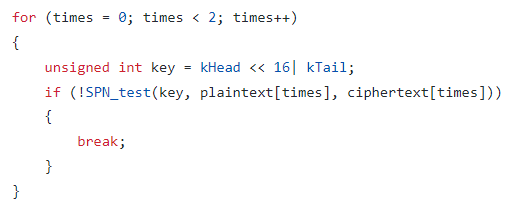


第二条链

1. 使用实验一中的快速读入大量明密文对。
2. 对链1进行线性分析，统计子秘钥对应的偏差并排序。对链2进行线性分析，统计子秘钥对应的偏差并排序。



1. 穷举前16位秘钥，依次取偏差率较大的子秘钥，组合成完整的秘钥，从明密文对中选取若干明密文进行测试，若均正确则选出正确的秘钥。



1. 小结

因本次实验数据量较大，需要反复对候选秘钥进行测试，因此效率较低，经过对输入输出、循环、编译等优化后可达到900ms通过。此外本题效率依赖线性链的选择，链1的偏差较小，误差较大。

### 1.3差分分析

1. 设计内容

差分密码分析与线性密码分析较为相似，主要差别在于差分密码分析包含了将两个输入的异或与其对应的两个输出的异或相比较。通过对S盒的输入异或，输出异或进行分析，发现对于特定的S盒，若输入值的异或一定，则输出值的异或分布偏向某一特定值，称为异或扩散率。对任何一轮的差分进行分析，得到一个差分链。通过对最后一轮的差分进行分析，即可找出最后一轮的秘钥。

1. 设计过程

差分分析的具体实现包括以下几个部分：1. 对S盒分析，手动寻找两条差分链。 2.读入大量明密文对。3. 分别对两条链进行差分分析，得到两个子秘钥。4. 穷举前16位秘钥与子秘钥组合进行测试，得出最终秘钥。

1. S盒分析，可采用手动或编程方法测试，选取两条链均关联两个子秘钥。



第一条链



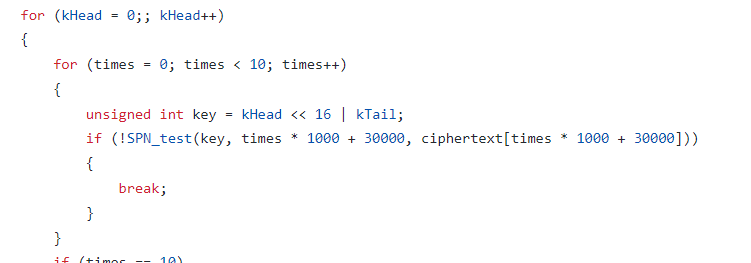
第二条链

1. 使用实验一中的快速读入大量明密文对。
2. 对链1进行线性分析，统计子秘钥对应的差分并排序。对链2进行线性分析，统计子秘钥对应的差分并排序。因差分攻击的效率较高，可知选取50-100个明密文对的值，此外可以丢弃不满足对应比特不相同的明密文组，过滤随机噪声，提高差分攻击效率。



过滤噪声

1. 穷举前16位秘钥，依次取偏差率较大的子秘钥，组合成完整的秘钥，从明密文对中随机选取若干明密文进行测试，若均正确则选出正确的秘钥。



结果验证

（3）小结

本次实验中，因差分攻击效率较高，因此采取了一些措施，如过滤噪声，减少取样数，增加检验数的方法来提高速率，实现了对秘钥的快速破解。此外，可以相对线性分析，差分分析可以选取两个独立的子秘钥链进行分析，效率相对更高。

### 1.4 SPN增强

（1）设计内容

对实验一中的SPN加解密进行优化，1.对秘钥长度，分组长度，S盒、P盒和轮数进行优化。2、 对轮秘钥生成算法进行优化。3、 使用分组密码加密模式对大量明文进行加密。

1. 提高效率。 5、 输出达到随机数检测标准。

本实验中 128 为秘钥， 64位为一个分组，明密文长度均为64位，使用16位S盒和64位P盒。轮数为4轮。为提高密文的随机性，采用了CBC模式，设定一个较大的初始向量，向量与明文进行异或后进行加密，加密结果作为下一个分组的向量。轮秘钥采用多次移位的方式产生。

（2）设计过程

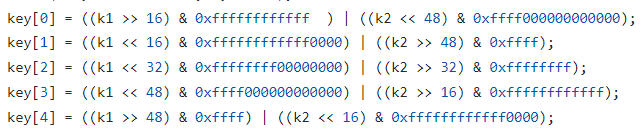
SPN增强加解密的具体实现包括以下几个部分：1. 数据的读入输出2. 轮秘钥的生成3. 分组密码模式 4. 白化 5. S盒代换 6. P盒置换 7. SPN主程序

1. 数据的读入输出：因为明密文的长度较长，使用scanf, printf的时间损耗较大，故使用fread、fwrtie来实现大量数据的读入，秘钥长度为128位，而C中unsigned long long最长为64位，使用两个unsigned long long 类型来存储初始秘钥。

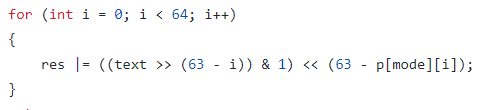




1. 轮秘钥的生成：使用多次移位来生成5个轮秘钥，如下图：



1. 分组密码模式：使用密码分组链接模式(CBC)，将每个明文组与前一个密文异或后再进行加密，初始向量为6420946196872948572。
2. 白化：同原SPN，将其扩展到64位。
3. S盒代换：同原SPN，将其扩展到64位。
4. P盒代换：同原SPN，将其扩展到64位。



1. SPN主程序：同原SPN，将调用函数改为对应64位操作。
2. 小结

本实验中为了能达到随机数检测标准作了一下优化，将秘钥长度增大到128位，同时轮秘钥的生成方式更加复杂，提高混乱性。采用CBC模式，将明文与上一轮加密的密文进行异或后加密，同时使用一个随机生成的大数作为初始向量，提高混乱性。

### 1.5 RSA参数生成

（1）设计内容

RSA密码体制中，n是两个不同的奇素数p和q的乘积，对于这样一个整数，φ(n) = (p-1)(q-1)。取随机数c(1<c<φ(n)),使得gcd(b,φ(n)) = 1，d = c ^ (-1) modφ(n)。其中c用于加密，(n,c)作为公钥 ,d用于解密,(p, q, d)作为私钥。

若已知参数p,q,c，求解参数d，只需先求出φ(n)后，求c对φ(n)的模逆即可，在本题中还需要适当检验参数的合法性，若参数不合法则需要输出ERROR。

1. 设计过程

求解参数d分为以下几个过程:1、读入参数e,p,q。2、判断参数是否合法。3、求φ(n)。4、求c对φ(n)的模逆，具体实现如下。

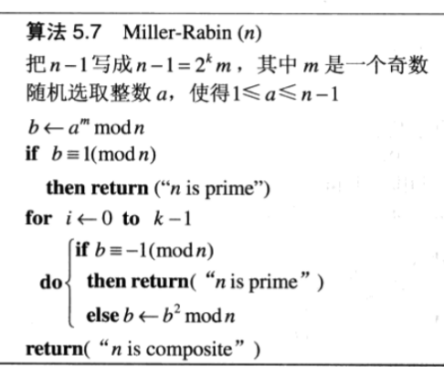
1. 读入大整数e,p,q。使用gmp自带的gmp\_scanf来读入。
2. 判断参数是否合法。具体有以下条件：
   1. e不能过小，若e小于10即为不合法。
   2. p, q必须是素数，若有一个不是即为不合法。
   3. p,q差值不能过小，若|p-q|的绝对值小于p/10即为不合法。
   4. p-1和q-1的公因数不能过大，若p-1, q-1的公因数大于100000即为不合法。
   5. e和φ(n)必须互质，若不互质即为不合法。

若满足以上条件即为合法

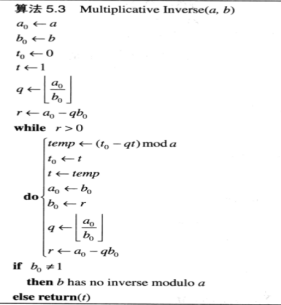
1. 求c对φ(n)的模逆。

在上述求解过程中

* 使用Miller-Rabin算法实现了素性检测。



* 使用辗转相除法实现了求解最大公因数。
* 使用Euclidean算法实现了求解模逆。



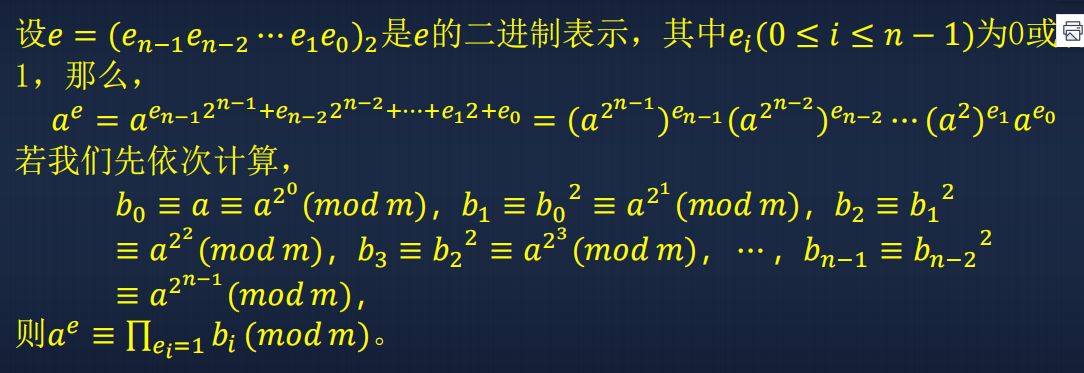
1. 小结

通过本实验，自己动手实现了求解RSA参数d的过程，使用c++ gmp库实现了一系列算法，加深了对于这些算法的理解和掌握。可以在判断参数是否合法处继续优化。

### 1.6 模重复平方

（1）设计内容

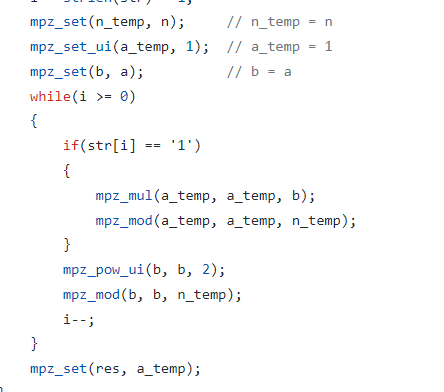
正确计算a^e(mod N)的值。计算方法如下：



对于计算expmod(m, e, pq)，a = m, e = e, m = pq代入。

1. 设计过程

将a, e, n作为参数传入函数expmod。a赋值1， b赋值a, 进入循环，判断 e是否为0，若e为0则退出循环。每一轮循环中，判断e模2是否等于1，若等于1，则计算a = a\* b mod n。后计算b = b ^ 2 mod n。循环结束后，a的值即为结果。代码如下：



为加速计算，可以先将e转化为2进制字符串的形式，避免了多次除以2，取余的操作，只需遍历字符串，即可判断是否为0或模2的结果。



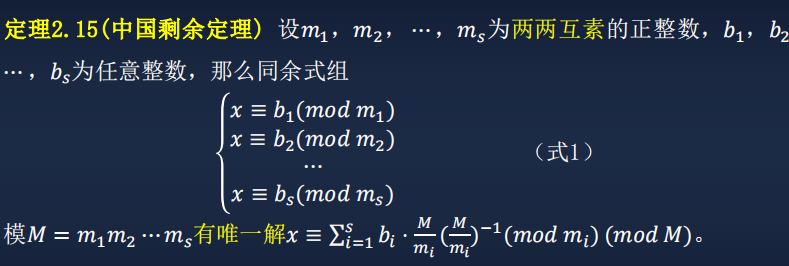
（3）小结

本次实验较为简单，只需简单复现书中的算法即可，但可以使用转换字符串的方法来提高计算速度。实验较难的部分是理解书中算法的含义。

### 1.7 中国剩余定理

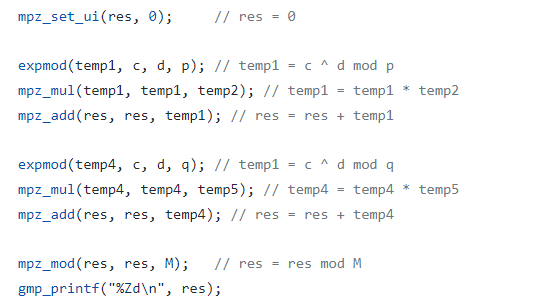
1. 设计内容

在进行解密的时候，直接计算c^d mod(pq)时，因p,q值较大，计算速度较慢，因p，q互素。因此可以通过中国剩余定理计算 c ^ d mod p \*q \*q ^(-1) mod p + c^d \*p \* p^(-1) mod q(mod pq)的值。中国剩余定理原理如下：



1. 设计过程

中国剩余定理计算分为以下几部分：1、计算参数d的值。2、计算c ^ d mod p的值。3、 计算 q ^(-1) mod p。4、 计算c ^ d mod p \*q \*q ^(-1) mod p 的值。 5、 计算c^d mod q的值。 6、 计算p ^ (-1) mod q。 7、计算 c^d \*p \* p^(-1) mod q。 8、将4、7步中的结果相加mod pq即得到结果。



因存在多个c，而p, q, e的值唯一，因此可以提前计算出q\* q ^(-1) mod p的值保存在temp2中，提前计算出p \* p^(-1) mod q的值，保存在temp5中，这样每轮循环中就不需要重复计算这些值，提高计算速率。

1. 小结

本次实验利用了中国剩余定理加速了解密的过程，灵活运用了前面所学的模重复平方法，求解参数d的方法。同时采用了一系列方法提高解密速度。

### 1.8 PKCS7

（1）设计内容

PKCS#7,也叫做加密消息的语法标准，由[RSA](https://baike.baidu.com/item/RSA/210678?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/PKCS%237/_blank)安全体系在[公钥加密系统](https://baike.baidu.com/item/%E5%85%AC%E9%92%A5%E5%8A%A0%E5%AF%86%E7%B3%BB%E7%BB%9F/2572570?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/PKCS%237/_blank)中交换数字证书产生的一种加密标准。

PKCS#7描述[数字证书](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E8%AF%81%E4%B9%A6?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/PKCS%237/_blank)的语法和其他加密消息——尤其是，[数据加密](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%8A%A0%E5%AF%86?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/PKCS%237/_blank)和[数字签名](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/PKCS%237/_blank)的方法，也包含了算法。当使用PKCS#7进行数字签名时，结果包含签名证书（一列相关证书撤回列表）和已证明路径上任何其他证书。如果使用PKCS#7加密数据，通常包含发行者的参考消息和证书的序列号，它与用于解密已加密数据的公共密钥相关。

（2）设计过程

参考资料中的代码，PKCS#7解密分为以下几个部分:1、初始化。2、将私钥转为EVP\_PKEY结构体，将证书转为X509结构体。3、读入密文。4、解密密文。5、验证签名。具体过程如下。

1. 初始化：调用ERR\_load\_crypto\_strings和OpenSSL\_add\_all\_algorithms函数初始化
2. 将私钥转为EVP\_PKEY结构体，将证书转为X509结构体。

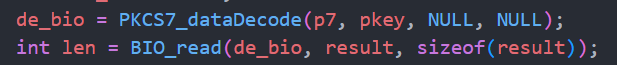




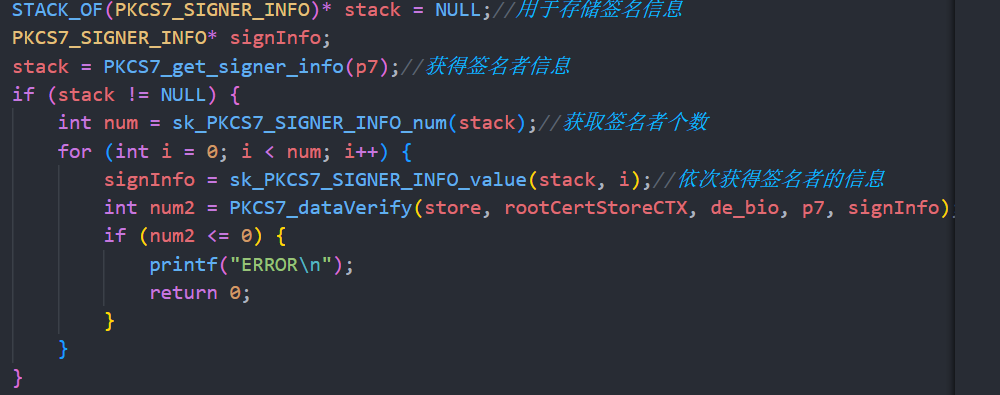
1. 读入密文：使用BIO\_new\_fd从stdin中读入密文。



1. 解密密文：使用读入的密文初始化PKCS7, 使用PKCS7解密密文，并写入result中



1. 验证签名：从pkcs7中获取签名者信息，依次遍历签名者信息，进行验证，若与证书匹配则输出解密密文，否则输出ERROR。



（3）小结

本次实验学习了对于openssl 中 crypt库的学习，学会了调用一些库函数实现对PKCS7的解密以及签名的验证。

### 1.9 彩虹表

（1）设计内容

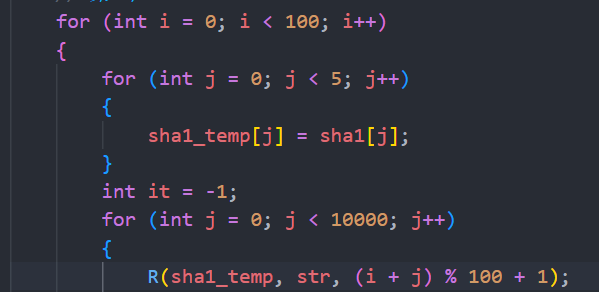
彩虹表是一个用于[加密散列函数](https://zh.m.wikipedia.org/wiki/%E5%8A%A0%E5%AF%86%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B0" \o ")逆运算的预先计算好的[表](https://zh.m.wikipedia.org/wiki/%E6%9F%A5%E6%89%BE%E8%A1%A8" \o "查找表)，常用于破解加密过的密码散列。 彩虹表常常用于破解长度固定且包含的字符范围固定的密码。这是[以空间换时间](https://zh.m.wikipedia.org/wiki/%E6%97%B6%E7%A9%BA%E6%9D%83%E8%A1%A1" \o "时空权衡)的典型实践，比[暴力破解](https://zh.m.wikipedia.org/wiki/%E6%9A%B4%E5%8A%9B%E7%A0%B4%E8%A7%A3" \o "暴力破解)（Brute-force attack）使用的时间更少，空间更多；但与储存密码空间中的每一个密码及其对应的[哈希值](https://zh.m.wikipedia.org/wiki/%E5%93%88%E5%B8%8C%E5%80%BC" \o "哈希值)（Hash）实现的[查找表](https://zh.m.wikipedia.org/wiki/%E6%9F%A5%E6%89%BE%E8%A1%A8" \o "查找表)相比，其花费的时间更多，空间更少。

（2）设计过程

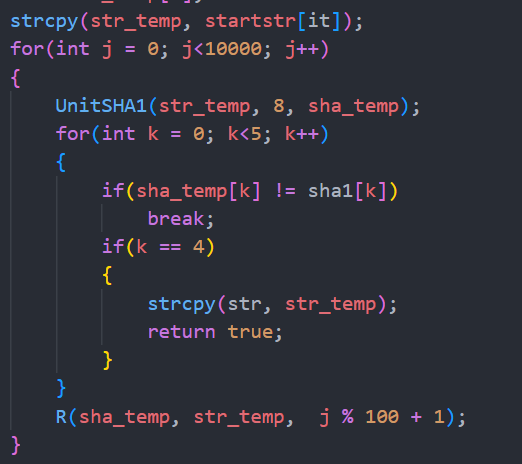
彩虹表中的一系列相关的衰减函数使用题目提供的衰减函数Ri，对SHA1加密进行破解。

彩虹表破解分为以下几个部分：1、读入彩虹表的条数m，起始字符串和终止字符串，并将终止字符串存入哈希表中。2、读入sha1散列后的值。3、遍历起始R函数从1到100，再将sha1往后递归10000次，过程中判断是否有字符串与终止字符串相同4、若所求值存在于某链中，则从头开始向后递归求得目的字符串。具体步骤如下：

1. 读入彩虹表的条数m，起始字符串和终止字符串。使用C++库unordered\_map作为哈希表，便于查找终止字符串。
2. 读入sha1散列后的值。
3. 使用双层循环，i为起始的R函数,j为递归次数。保证若所求散列值在链中，能够递归到结尾。递归过程中，可使用数组判断是否彩虹表是否循环来提高效率。



1. 从该链起始字符串向后递归，找出字符串的哈希值与所求哈希值相同。



1. 小结

通过本次实验，我学习到了如何在彩虹表中快速查找，破解所需的哈希值。使用C++库中的unordered\_map哈希表提高了查询速度。

## 二、实验心得

## 三、对课程设计内容和过程的建议