

**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：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**教师评语：**

**分数：**

**网络空间安全学院**

密码学课程设计检查表

班级： IS1803 姓名： 张帅 学号： U201814735

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **内容及分值** | **检查情况** | **得分** | **检查时间** |
| 1 | 原始SPN（5） |  |  |  |
| 2 | 线性密码分析（5） |  |  |  |
| 3 | 差分密码分析（5） |  |  |  |
| 4 | SPN增强（10） |  |  |  |
| 5 | RSA参数生成（10） |  |  |  |
| 6 | 模重复平方（5） |  |  |  |
| 7 | 中国剩余定理（10） |  |  |  |
| 8 | 蒙哥马利算法（10） |  |  |  |
| 9 | PKCS7（10） |  |  |  |
| 10 | 彩虹表（5） |  |  |  |
| 11 | 实验报告（20） |  |  |  |

合计：

密码学课程设计任务书

|  |
| --- |
| 课题内容：  （1）原始SPN（教材上）算法的实现。  （2）对上述算法进行线性密码分析及差分密码分析（求出所有32比特密钥）。  （3）增强以上SPN的安全性（如增加分组的长度、密钥的长度、S盒、轮数等）。  （4）对原始及增强的SPN进行随机性检测，对检测结果进行说明。  （5）生成RSA算法的参数（如p、q、N、私钥、公钥等）。  （6）快速实现RSA（对比模重复平方、蒙哥马利算法和中国剩余定理）。  （7）利用椭圆曲线密码算法、HASH函数、压缩函数、对称加密算法实现一个类似PGP的文件加解密及完整性校验功能。  （8）利用USBKEY和数字证书实现简单的文件加密保护。  （9）构造彩虹表破解HASH函数。 |
| 课题任务要求：   1. 掌握线性、差分分析的基本原理与方法。 2. 体会位运算、预计算在算法快速实现中的作用。 3. 可借助OpenSSL、GMP、BIGINT、FLINT等大数运算库的低层基本函数（加、减、乘、除、模），实现过程中必须体现模重复平方、中国剩余定理和蒙哥马利算法的过程。内容(7)的算法可以直接调用OpenSSL或者其它密码库。 4. 了解和掌握彩虹表构造的基本原理和方法，能够设计和实现约化函数（Reduction Function），针对特定的HASH函数构造彩虹表，进行口令破解，了解彩虹表的弱点及防范方法。 5. 独立完成课程设计内容，现场演示并讲解。 6. 掌握USBKEY和证书的基本使用方法。 7. 课程设计完成后一周内，提交课程设计报告。 |
| 主要参考文献（由指导教师选定）   1. 密码学原理与实践（第三版）. DouglasR.Stinson著，冯登国译，电子工业出版社，2009 2. 应用密码学：协议算法与C源程序（第二版）. Bruce Schneier 著，吴世忠等译，机械工业出版社，2014 |
| 同组设计者 无 |

**目录**

[1 SPN算法实现 6](#_Toc54628078)

[1.1 实验要求 6](#_Toc54628079)

[1.2 实验过程 6](#_Toc54628080)

[1.3 实验结果分析 7](#_Toc54628081)

[2 线性密码分析 8](#_Toc54628082)

[2.1 实验要求 8](#_Toc54628083)

[2.2 实验过程 8](#_Toc54628084)

[2.3 实验结果分析 9](#_Toc54628085)

[3 差分密码分析 10](#_Toc54628086)

[3.1 实验要求 10](#_Toc54628087)

[3.2 实验过程 10](#_Toc54628088)

[3.3 实验结果分析 11](#_Toc54628089)

[4 SPN增强 12](#_Toc54628090)

[4.1实验要求 12](#_Toc54628091)

[4.2实验过程 12](#_Toc54628092)

[4.3实验结果分析 13](#_Toc54628093)

[5 RSA参数计算 14](#_Toc54628094)

[5.1 实验要求 14](#_Toc54628095)

[5.2 实验过程 14](#_Toc54628096)

[5.3 实验结果分析 14](#_Toc54628097)

[6 模重复平方法 15](#_Toc54628098)

[6.1 实验要求 15](#_Toc54628099)

[6.2 实验过程 15](#_Toc54628100)

[6.3 实验结果分析 15](#_Toc54628101)

[7 中国剩余定理 16](#_Toc54628102)

[7.1 实验要求 16](#_Toc54628103)

[7.1 实验过程 16](#_Toc54628104)

[7.1 实验结果分析 16](#_Toc54628105)

[8 Montgomery 17](#_Toc54628106)

[8.1 实验要求 17](#_Toc54628107)

[8.2 实验过程 17](#_Toc54628108)

[8.3 实验结果分析 18](#_Toc54628109)

[9 PKCS7 19](#_Toc54628110)

[9.1 实验要求 19](#_Toc54628111)

[9.2 实验过程 20](#_Toc54628112)

[9.3 实验结果分析 20](#_Toc54628113)

[10 彩虹表 21](#_Toc54628114)

[10.1 实验要求 21](#_Toc54628115)

[10.2 实验过程 22](#_Toc54628116)

[10.3 实验结果分析 22](#_Toc54628117)

[11 实验总结 23](#_Toc54628118)

1 SPN算法实现

### 1.1 实验要求

内容：按照教材例3.1实现SPN加解密函数。  
要求：  
（1）正确实现算法的加解密过程  
（2）快速实现算法

### 1.2 实验过程

* 设l,m,Nr是正整数,P=C={0,1}lm
* K⊆({0,1}lm)Nr+1是由初始密钥k用密钥编排算法生成的所有可能的密钥编排方案集合，一个密钥编排方案记为(k1,k2,...,kNr+1)
* 状态值w长度为l×m，记为w1,w2,...,wlm；
* w可以看成m个长度为l的子串连接而成，记为
* w=w<1>,w<2>,...,w<m>，其中
* w<i>=w(i-1)l+1,w(i-1)l+2,...,w(i-1)l+l
* 加密过程使用如下算法描述：SPN(x,πs,πp,(k1,k2,...,kNr+1))

w0=x

for r=1 to Nr-1 {

ur=wr-1⊕kr  // 白化

for i=1 to m {

vr<i>=πs(ur<i>) // 代替

}

wr=(vrπp(1),vrπp(2),...,vrπp(lm)) // 置换

}

uNr=wNr-1⊕kNr

for i=1 to m {

vNr<i>=πs(uNr<i>) // 代替

}

y=vNr⊕kNr+1 // 白化

return y

### 1.3 实验结果分析

结果：AC

一开始的时候把加密解密过程中的S盒代换、P盒代换、密钥编排、加密解密部分都写了独立的函数，结果因为函数调用过多导致超时，后来仅定义加密、解密函数，其他的均在内部给出之后，达到了给定的时间要求，实现了实验。

2 线性密码分析

### 2.1 实验要求

内容：根据已知明密文对分析原始SPN的密钥。  
要求：  
（1）实现教材所给算法。  
（2）能根据所给8000对明密文对分析对应位置密钥。  
（3）分析出所有32比特密钥。

### 2.2 实验过程

采用已知明文攻击方法，如果掌握了足够多的明-密文对，即可求出轮密钥ki，进而根据轮密钥编排方案反向推导出加密密钥k。线性密码分析思路为找到足够多的明文-密文对，对可能的密钥进行穷举，计算相关随机变量的偏差，正确的密钥作用下，偏差的绝对值最大。

具体算法如下：线性攻击(T, T, πs-1)

for (L1,L2)=(0,0) to (F,F) { // L1,L2表示候选子密钥k5<2>和k5<4>

Count[L1,L2]=0 // 每个候选子密钥分配一个计数器并初始化为0

}

for each (x,y) ∈ T {

for (L1,L2)=(0,0) to (F,F) {

v4<2> = L1⊕y<2>

v4<4> = L2⊕y<4>

u4<2> = πs-1(v4<2>)

u4<4> = πs-1(v4<4>)

z = x5⊕x7⊕x8⊕u46⊕u48⊕u414⊕u416 // 计算随机变量值

if z=0 {

Count[L1,L2] ++;

}

}

max = -1

for (L1,L2)=(0,0) to (F,F) {

Count[L1,L2] = | Count[L1,L2] - T/2 |

if Count[L1,L2] > max {

max = Count[L1,L2]

maxkey = (L1,L2)

}

// maxkey即为所求子密钥

此算法理论上可行，但实际上只是缩小了穷举密钥的范围，算法本身也需要穷举候选密钥，因此和直接穷举密钥相比，总体上提高的效率有限，除非S盒有较大的线性缺陷。

### 2.3 实验结果分析

结果：AC

线性分析的一个难点是找到合适的线性链，24盒子的链教材上的就很合适，而13盒子的链经过漫长的测试发现并没有合适的，最后找了一个包含124盒子的线性链，实现了实验。

时限是另一个难点，一开始的时候只过了八个点，后来考虑了空间换时间的方法，把S、P盒的结果在一开始一次运算出来，存到s[0x10000]和p[0x10000]中，最后实现了实验。

3 差分密码分析

### 3.1 实验要求

内容：根据已知明密文对，选择明密文分析原始SPN的密钥。  
要求：  
（1）实现教材所给算法。  
（2）能根据所给明密文对分析对应位置密钥。

### 实验过程

找到足够多的四元组(x,x\*,y,y\*)，其中x’=x⊕x\*固定不变。对可能的密钥进行穷举，计算相关差分的扩散率，正确的密钥作用下，扩散率应最大。

和线性分析一样，不需要对全部密钥空间进行穷举，只需要对候选子密钥进行穷举即可。

具体算法如：差分攻击(T, T, πs-1)

for (L1,L2)=(0,0) to (F,F) // L1,L2表示候选子密钥k5<2>和k5<4>

{

Count[L1,L2]=0 // 每个候选子密钥分配一个计数器并初始化为0

}

for each (x,x\*,y,y\*) ∈ T

{

if (y<1>=y\*<1> and y<3>=y\*<3>)

{ // 只考虑y’<1>和y’<3>=0

for (L1,L2)=(0,0) to (F,F) {

v4<2> = L1⊕y<2>

v4<4> = L2⊕y<4>

u4<2> = πs-1(v4<2>)

u4<4> = πs-1(v4<4>)

(v4<2>)\*= L1⊕(y<2>)\*

(v4<4>)\* = L2⊕(y<4>)\*

(u4<2>)\* = πs-1((v4<2>) \*)

(u4<4>)\* = πs-1((v4<4>)\*)

(u4<2>)’=u4<2>⊕(u4<2>)\*

(u4<4>)’=u4<4>⊕(u4<4>)\*

if (u4<2>)’=0110 and (u4<4>)’ = 0110 {

Count[L1,L2] ++;

}

}

}

}

max = -1

for (L1,L2)=(0,0) to (F,F) {

if Count[L1,L2] > max {

max = Count[L1,L2]

maxkey = (L1,L2) }

}

// maxkey即为所求子密钥

### 实验结果分析

结果：PAC（42分），最后一个点超时。

差分分析仍然考虑了空间换时间的方法，把s\_box和p\_box提前给出，然而还是过不了最后一个点，后来也没想到足够合适的解决办法。

1. SPN增强

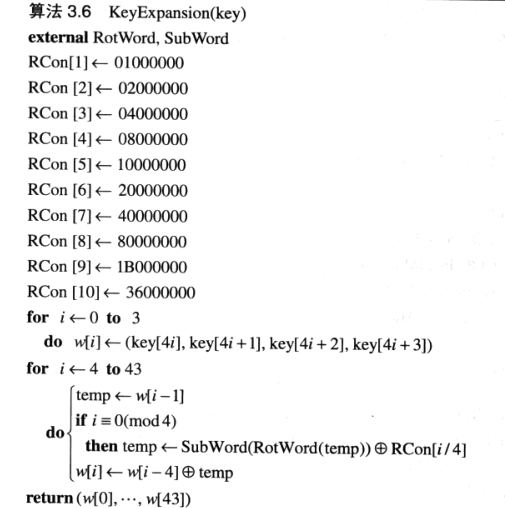
### 4.1实验要求

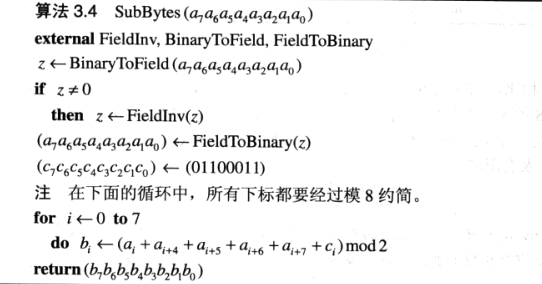
内容：对原始SPN进行改进。  
要求：  
（1）选择合适的密钥长度、分组长度、S盒、P置换、轮数。  
（2）效率较高  
（3）输出达到随机数检测标准

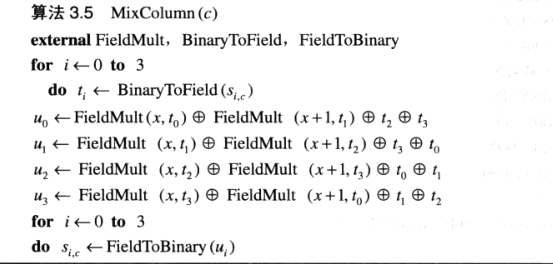
### 4.2实验过程

本次实验考虑采用AES实现：其主要算法有SubBytes()、MixColumn()、KeyExpansion()等。

AES为分组密码，分组密码也就是把明文分成一组一组的，每组长度相等，每次加密一组数据，直到加密完整个明文。在AES标准规范中，分组长度只能是128位，也就是说，每个分组为16个字节（每个字节8位）。密钥的长度可以使用128位、192位或256位。密钥的长度不同，推荐加密轮数也不同。







虽然是SPN增强，但我还是考虑用实现AES的办法考虑实现本题。

### 4.3实验结果分析

结果：PAC(85分)

不管怎么调整轮数都会出现一个点WA的情况，对这个也不是很理解，不过的确是实现了AES进而实现了SPN的增强。

5 RSA参数计算

### 5.1 实验要求

内容：求RSA参数d。

要求：

（1）利用加法、减法、乘法、模运算等基本运算。

（2）自己实现求逆,gcd。

（3）简单检查参数的合法性。

### 5.2 实验过程

RSA 算法描述如下：

（1）公钥

选择两个互异的大素数 p 和 q，n 是二者的乘积，即 n=p\*q，使 Φ(n)=(p-1)(q-1)为欧拉函数。随机选取正整数 e，使其满足 gcd (e, Φ(n)) =1，即 e 和 Φ(n) 互质，则将(n, e)作为公钥。

（2）私钥

求出正数 d，使其满足 e×d=1(mod Φ(n))，则将(n, d)作为私钥。

（3）加密算法

对于明文 M，由 C=Me(mod n)，得到密文 C。

（4）解密算法

对于密文 C，由 M=Cd(mod n)，得到明文 M。

如果窃密者获得了 n、e 和密文 C，为了破解密文必须计算出私钥 d，为此需要先分解 n。为了提高破解难度，达到更高的安全性，一般商业应用要求 n 的长度不小于 1024 位，更重要的场合不小于 2048 位。

实验过程中需要注意以下几点：

* + e不能太小: e的值越大，计算的难度越高。
  + p和q的间隔不能太小，更不能相等：在这里我的比较方式是-65536 < |p-q| < 65536。
  + (p-1)和(q-1)不能太平滑，指的是而这个最大公因数不能超过20。

只要注意到这三点，本次实验就能轻松完成（当然要阅读一下gmp库的文档）。

### 5.3 实验结果分析

结果：AC

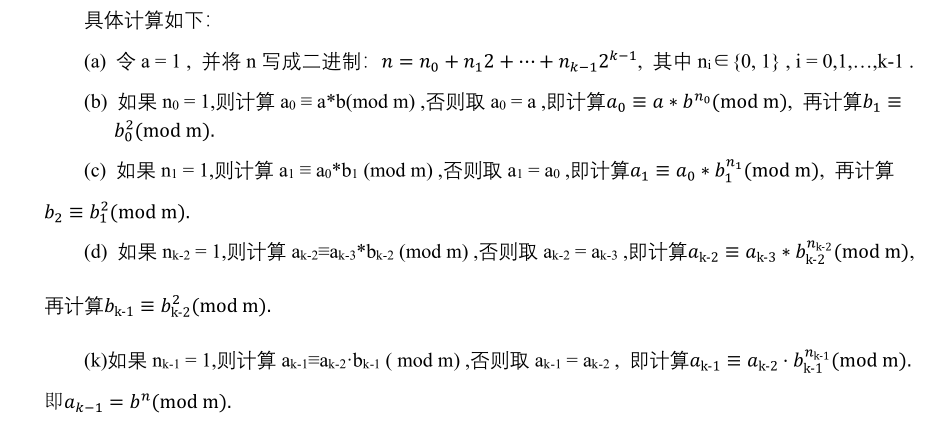
本次实验还是难度不大的，要注意判断参数的条件不能太强也不能太弱。

6 模重复平方法

### 6.1 实验要求

内容：正确计算a^e(modN)。  
要求：  
（1）利用加法、减法、乘法、模运算等基本运算。  
（2）自己实现expmod(a,e,n)

### 6.2 实验过程



这道题就是对算法进行翻译就好，很快就实现了。

### 6.3 实验结果分析

结果：AC

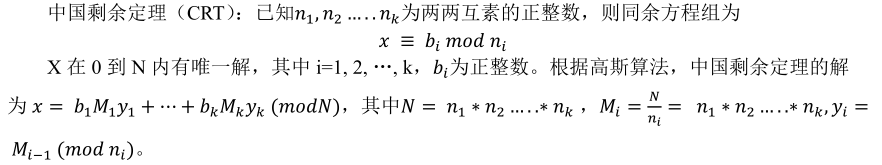
7 中国剩余定理

### 7.1 实验要求

内容：正确计算c^d(modpq)。  
要求：

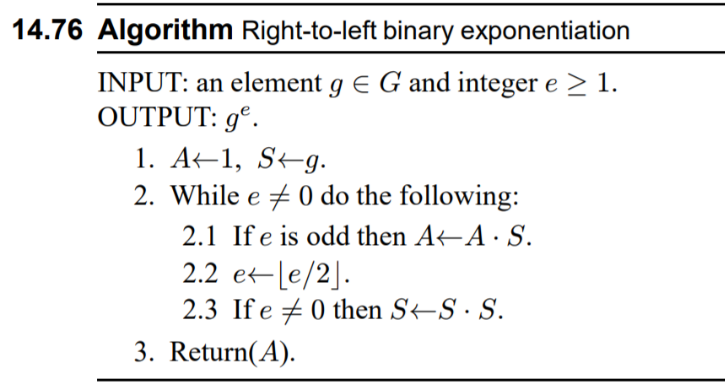
（1）利用V中的求逆运算从加密密钥e计算解密密钥d。  
（2）利用VI中实现的模幂运算和中国剩余定理计算c^d(modpq)。

### 7.1 实验过程



但是这道题并没有这么简单，如果只考虑实现常规意义的中国剩余定理，会导致最后两个点超时。

于是考虑用快速幂算法实现本题，算法如下：



改进之后快了400多ms, 通过了最后两个点。

### 7.1 实验结果分析

结果：AC

本次实验如果只是实现中国剩余定理还是比较容易的，但是为了对时间做出优化就变得难了起来，变量变得特别多。参考了网络上的代码，现在回看仍然有点一知半解。

8 Montgomery

### 8.1 实验要求

内容：正确计算a^d(modpq)。  
要求：  
（1）实现Montgomery算法。  
（2）Montgomery+中国剩余定理。

### 8.2 实验过程

由于 RSA 的核心算法是模幂运算，模幂运算又相当于模乘运算的循环，要提高 RSA 算法的效率，首要问题在于提高模乘运算的效率。

模乘过程中复杂度最高的环节是求模运算，因为一次除法实际包含了多次加法、减法和乘法，如果在算法中能尽量减少甚至避免除法，则算法的效率会大大提高。

Montgomery算法的核心思想在于将求 A\*B %N 转化为不需要反复取模的 A\*B\*R' %N（移位即可，因为 R 是 2^K,总之 R 是与进制相关的数），即不使用除法（用移位操作）而求得模乘运算的结果。

对于任意指数 E，都可采用以下算法计算 D=C\*\*E % N：

D=1

WHILE E>0{

IF（E%2=0）{

C=C\*C % N

17

E=E/2

}

ELSE{

D=D\*C % N

E=E-1

}

}

RETURN D

继续分析会发现，要知道 E 何时能整除 2，并不需要反复进行减一或除二的操作，只需验证 E的二进制各位是 0 还是 1 就可以了，从左至右或从右至左验证都可以，从左至右会更简洁，

设 E=Sum[i=0 to n](E\*2\*\*i)，0<=E<=1, 则：

D=1

FOR i=n to 0{

D=D\*D % N

IF（E=1）

D=D\*C % N

}

RETURN D

### 8.3 实验结果分析

结果：AC。

有了上一题的优化思路，本道题也AC了，虽然比上一题慢一点，有点不知道为什么。

9 PKCS7

### 9.1 实验要求

PKCS#7是PKI中用于消息加密的语法标准。可以用于给拥有公钥的用户发加密邮件、传送加密文件等。

现在假设我们有一个可信的根CA，其公钥证书为：

-----BEGIN CERTIFICATE-----

MIIB/zCCAaagAwIBAgIJAKKa0PAt9M1FMAoGCCqBHM9VAYN1MFsxCzAJBgNVBAYT

AkNOMQ4wDAYDVQQIDAVIdUJlaTEOMAwGA1UEBwwFV3VIYW4xDTALBgNVBAoMBEhV

U1QxDDAKBgNVBAsMA0NTRTEPMA0GA1UEAwwGY2Fyb290MB4XDTIwMDkyMDIwNTkx

OVoXDTMwMDkxODIwNTkxOVowWzELMAkGA1UEBhMCQ04xDjAMBgNVBAgMBUh1QmVp

MQ4wDAYDVQQHDAVXdUhhbjENMAsGA1UECgwESFVTVDEMMAoGA1UECwwDQ1NFMQ8w

DQYDVQQDDAZjYXJvb3QwWTATBgcqhkjOPQIBBggqgRzPVQGCLQNCAASJ8mm28JJR

bZKLr6DCo1+KWimpKEsiTfZM19Zi5ao7Au6YLosyN71256MWmjwkwXxJeLa0lCfm

kF/YWCX6qGQ0o1MwUTAdBgNVHQ4EFgQUAL5hW3RUzqvsiTzIc1gUHeK5uzQwHwYD

VR0jBBgwFoAUAL5hW3RUzqvsiTzIc1gUHeK5uzQwDwYDVR0TAQH/BAUwAwEB/zAK

BggqgRzPVQGDdQNHADBEAiAaZMmvE5zzXHx/TBgdUhjtpRH3Jpd6OZ+SOAfMtKxD

LAIgdKq/v2Jkmn37Y9U8FHYDfFqk5I0qlQOAmuvbVUi3yvM=

-----END CERTIFICATE-----

用户B是PKCS#7格式加密消息的接收者，用户B的私钥为：

-----BEGIN EC PARAMETERS-----

BggqgRzPVQGCLQ==

-----END EC PARAMETERS-----

-----BEGIN EC PRIVATE KEY-----

MHcCAQEEINQhCKslrI3tKt6cK4Kxkor/LBvM8PSv699Xea7kTXTToAoGCCqBHM9V

AYItoUQDQgAEH7rLLiFASe3SWSsGbxFUtfPY//pXqLvgM6ROyiYhLkPxEulwrTe8

kv5R8/NA7kSSvcsGIQ9EPWhr6HnCULpklw==

-----END EC PRIVATE KEY-----

### 9.2 实验过程

本题的难度并不大，难度在于环境的配置，在安装库的过程中耽搁了很多的时间，最后请同学帮忙弄好了实验的环境。

* 1. 先把根CA和私钥pkeyB存储好。
  2. 然后读取输入的PKCS7字符串并一次提取BIO，转换PKCS7类型，利用根CA的公钥检验输入的数据是否由根CA发出。这些步骤中如果有错误则输出“ERROR”。
  3. 解密：从B的密钥串中提取出密钥，利用密钥提取出BIO，再从BIO接口中将解密数据读取出来
  4. 利用B的公钥验证消息的签名。
  5. 验证成功后打印出消息

### 9.3 实验结果分析

结果：AC。

本次实验就是利用openssl对PKCS7字符串进行验证和解密，查询相关文档之后可以模仿来实现全部过程。

10 彩虹表

### 10.1 实验要求

已知SHA1彩虹表的100个R函数如下：

void getstr(unsigned n,char str[8])

{

str[0]='a';str[1]='0';str[2]='0';str[3]='0';str[4]='0';str[5]='0';str[6]='0';str[7]='0';

int i=2;

while(n)

{

unsigned tmp=n%36;

if(tmp<10)

str[i++]=tmp+'0';

else

{

str[i++]=tmp-10+'a';

}

n=n/36;

}

}

void R(unsigned sha1[5],char str[8],int i)

{

getstr((sha1[0]+sha1[1]\*i)%2176782336,str);

}

其中i=1,2,3,...,100。

现在有一些链头和链尾，每条链从链头开始，依次调用了100000次SHA1和R函数得到链尾。

希望从这些链中找到SHA1值对应的口令。

### 10.2 实验过程

本题的原理是：将彩虹表的链头和链尾存储起来，对于要查找的SHA1值，假设它是这条链上中间的一个值，我们从后往前开始尝试。假设它是链上第k个SHA1值，那么对它交叉做k次R函数和k-1次SHA1运算，将得到的值和各个链尾比较，如果某个链尾相同，说明它可能是这条链上一个节点，那么将对应的链首进行10000-k次SHA1和R函数交叉运算得到的这个值就是要查找的SHA1值对应的口令。

### 10.3 实验结果分析

结果：AC。

如果从首节点开始计算需要n条链从头开始向后同时计算，从中间的某个节点计算到尾节点只需要对这一条链进行计算，但每次循环都会从不同Ri开始，需要重新计算。估测它们时间复杂度差不多

在做的过程中发现如果从第一条链计算，很快就能出结果，这和测试样例有关，但这种方法在实际中用处不大，就不再考虑了。