江西理工大学理学院
《信息安全与密码学》实验指导书

主审人：刘建生　　**主撰人：尹宝勇**

信息与计算科学教研室

前　言

随着互联网应用的不断深入，信息系统成为了政府、企业等组织的必不可少的组成部分。信息安全问题越来越引起人们的重视。信息安全作为 IT 业的热门话题经久不衰，究其原因，第一、IT 技术不断发展，新技术总有新问题；第二、安全问题的难度。根据木桶原则，系统中只要有一个部分存在安全缺陷，或者说只要存在一处漏洞，　那么整个系统就是不安全的。这对于设计安全系统来说，有如恶梦。系统中的每个部分、每个细节都要考察，而且新的攻击方式层出不穷，未来可能出现什么攻击手段还是未知数。尽管难度很大，但是还是有一定的规律可循。密码学作为信息安全领域的基本工具，一直是研究的热门。计算机的计算能力不断提高，也就伴随着已有的密码安全性逐渐降低，甚至是变得不再安全，此时也就需要有新的密码取而代之。

《信息安全与密码学》课程作为专业选修课程，主要讲解密码学的原理，同时介绍一些必要的安全技术。本指导书通过实验，加深对课程内容的理解。实验共 8 个学时，主要针对密码学部分。希望诸位同学通过动手、总结、思考的过程，加深对课程的理解，体会安全设计的不同之处。

绪　论

本实验指导书是根据《信息安全与密码学》课程实验教学大纲编写，适用于信息与科学计算专业。

一、　本课程实验的作用与任务

通过实验课程，加深对理论课程的理解，掌握基本的密码算法。理解设计安全密码所需考虑的方方面面。

二、　本课程实验教学项目及要求

序号	实验项目名称	学时	实验类别	实验要求	实验类型	每组人数	目的和要求
1	实验 1　古典密码体制	2	基础	必修	验证	1	实现古典密码，并加深古典密码学攻击过程的理解
2	实验 2　AES 密码体制	6	基础	必修	设计	1	实现 AES 密码，掌握其加密过程，理解 AES 的设计原则
3	实验 3　RC4 算法	6	基础	必修	设计	1	通过实现 RC4 算法的实现，加深对序列密码体制的理解

.

实验 1　古典密码体制

一．实验目的

通过编程实现古典密码体制中的维吉尼亚密码体制，并了解古典密码的密码分析方式，加深对古典密码体制的理解。

二．实验环境：

硬件设备：PC 机
操作系统：windows 2000 操作系统
编程工具：Matlab，VC++

三．维吉尼亚（Vigenere）密码

维吉尼亚（Vigenere）密码是一种多表代换密码算法。它选择一个词组作为密钥，密钥中每个字母用来确定一个代换表，每个密钥字母用来加密一个明文字母，例如密钥字母为 a，明文字母为 c，则密文字母为 0+2 (mod26)=2，也就是 c。直到所有的密钥字母用完，后再从头开始，使用第一个密钥字母加密。也就是说，密钥循环使用。

例如：明文为 attack begins at five，密钥为 cipher，

```
   attack begins at five      明文
＋ cipher cipher ci pher      密钥
--------------------------
＝ cbihgb dmvprj cb upzv      密文
```

去除空格后为 cbihgbdmvprjcbupzv

四．实验内容

1. 使用 matlab 或者其它的编程语言，实现维吉尼亚（Vigenere）密码算法。要求写入报告中。
2. 利用你的程序，针对下列明文，应用加密算法，选择相应的密钥，运行程序，记录密文。
   a) 明文：are you going to scarborough fair
      密钥：paul simon
      密文：
   b) 明文：without no seams nor needlework
      密钥：thyme
      密文：
3. 下列密文使用了维吉尼亚（Vigenere）密码，密钥长度为 5，由字母组成。写个小程序，用穷举搜索法（亦称暴力法）破解密码。记录你的程序花费的时间。要求将程序写入报告中。

   明文：scientists try to answer questions about the world around us

密文：zgtpbamdeg avj ec hrdhsy ufpgamzyg hfzfh alp hcypo lfvyyo iz
密钥：

五．思考（写入报告）

一、NSA 想雇佣你做密码分析工作。这份工作主要分析使用维吉尼亚（Vigenere）加密的密文，但是仅有密文。要求你能分析出明文。为了表明你能胜任这份工作，请回答下面的问题：

1） 如何分析维吉尼亚（Vigenere）密码体制加密的密文？
2） 暴力破解法是否有效，在什么情况下可以使用暴力破解法？
3） 如果允许你借助计算机辅助分析，但是你只能自己动手设计程序，这个程序该如何设计？
4） 给出你的工作方案

二、（选作）作为测试，NSA 要求你破译下面的这段密文，给出明文和密钥，要求写出分析过程：

ALPGLECTALZHLASDHVPPZOTCNXSTKIGDAIPHVJNXCMWGPKSIZASTUATASCZJIIDPAM
DUPIOLLGLCUIGTYFPHHXTHMMPSHWWDUKLHALPCLKCDPWEWLZTRAMXDMXSTBRDELEVPIPP
WVVCDYWZUWSWXJIMGBXLAPXJLLGLCUIGTYFPHHXTHMMPSHWWDUKLHVYCQVHTTZLPPCCHX
ALEWLJLIPKFTVJEGHZPAJEYCVXRPPRWDKKTCNMYIOIXDAIWHVJEWLLTVOALNZEYSALPWV
XPAZSQIOINXAMPHDINPURZIIIDPAMDUPIOPZPZCNEDPUIRGVMYBPWDXZWTEWMNPURZICS
ETHROPUIRGVMYCLAJDYOMTSMPKLWSTOEDCVXSXUKQDYASXJLEDCSETUSYDDILGLRZIZEE
XZJTTKEYSDIHXSPYDAFPHHXTHMMPSBREXSNFHAMNTYSWAZHZLUPTZLALILVDPUHCXNLET
VYDCLWDAPOPPTMRWACDIYILBPEXCVXFCTMYSMYWIOEEHVQPDMCZJOEGTJSXTOICTVYEDM
KCTHXEGPEWHHROIYMMJSEEXVRDHVQPDMCZJOEGTJSXTMVPHOJCDTRLGYSHYHMWRLPWHHR
OHVQPDMCZJOEGTJSXTMVZBHVPPZASTYIJDBVBJLWEFBIDIMSCUYIPSVQWTMXJDBFLIAIC
TKFJIOIDIVVXHVJATYWPRBXTDUEYSZXLVNICTKFJIOIHXUHDDMTZAPGPQYYEPSMENFSFW
HZPQLIYIOIGTAICPUWZUJVPPAMGTZYQULVTCNGZCAMYJLXZLVVVLPXSIOIQPPXSIOEEJU
ILGUIOHBJQTYMYVPWCTKIXEAMGTNSMPJOEDTMDHPWDXWTTVVFLRRXZPSEMPTERDIENZAS
DDBXSRHVZAPRLVVFLRRXZVLSCVPERDIENZASWDBMDXHRLVVFLRRXZIOIDABQDPUHRWLXE
DZSQDBVYDYXSTYRNXAMPHRRZLPRRIOEEHVQPWVAEWPWDXAYLIPSYRHRLCKATASFPROEYV
LH

提示：明文是英文，加密体制为维吉尼亚（Vigenere）密码体制，密钥第 2 个字母为 E。

六．附录

1．英文字母频率表

A	B	C	D	E	F	G	H	I	J
0.082	0.015	0.028	0.043	0.127	0.022	0.020	0.061	0.070	0.002
K	L	M	N	O	P	Q	R	S	T
0.008	0.040	0.024	0.067	0.075	0.019	0.001	0.060	0.063	0.091
U	V	W	X	Y	Z				
0.028	0.010	0.023	0.001	0.020	0.001				

2．字母可以分为五组：

E	0.127	TAOINSHR	0.06~0.09
DL	0.04	CUMWFGYPB	0.015~0.023
VKJXQZ	小于 0.01		

3．最常见的两字母组合，依照出现次数递减的顺序排列：TH、HE、IN、ER、AN、RE、DE、ON、ES、ST、EN、AT、TO、NT、HA、ND、OU、EA、NG、AS、OR、TI、IS、ET、IT、AR、TE、SE、HI、OF。

4．最常见的三字母组合，依照出现次数递减的顺序排列：THE、ING、AND、HER、ERE、ENT、THA、NTH、WAS、ETH、FOR、DTH。

实验 2　AES 密码体制

一．实验目的

通过实现 AES 算法，以及对实际的数据进行加密和解密，加深对 AES 密码体制原理的理解。

二．实验环境

硬件设备：PC 机
操作系统：windows 2000 操作系统
编程工具：Matlab，VC++或 Eclipse

三．AES 简介

信息加密根据采用的密钥类型可以划分为对称密码算法和非对称密码算法。对称密码算法是指加密系统的加密密钥和解密密钥相同，或者虽然不同，但是可以从其中任意一个推导出另一个，更形象的说就是用同一把钥匙开锁和解锁。在对称密码算法的发展历史中曾出现过多种优秀的算法，包括 DES、3DES、AES 等。

2000 年 10 月，NIST（美国国家标准和技术协会）宣布通过从 15 种候选算法中选出的一项新的高级加密标准（AES）规范 Rijndael。Rijndael 这个名字是从它的两个发明者 Rijmen 和 Daemen 的名字得来的。

AES 是迭代、对称密钥分组的密码体制，它可以使用 128、192 和 256 位密钥，并且用 128 位（16 字节）分组加密和解密数据。基于代换与置换运算。置换是对数据重新进行安排，代换是将一个数据单元替换为另一个。详细的加解密过程请参考官方标准：

http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

四、实验步骤

1. 根据附录提供的算法与代码，实现AES加密算法，使算法能够对128位的数据加密。要求能够处理密钥为128位，数据分组为128位，也即16字节的情况。
2. 使用下面数据测试你的算法：

定义加密过程轮数为 Nr=10
密钥
0X2B,0X7E,0X15,0X16,　0X28,0XAE,0XD2,0XA6,　0XAB,0XF7,0X15,0X88,
X09,0XCF,0X4F,0X3C
明文
0X32,0X43,0XF6,0XA8,　0X88,0X5A,0X30,0X8D,　0X31,0X31,0X98,0XA2,
0XE0,0X37,0X07,0X34
密文
0X39,0X25,0X84,0X1D,　0X02,0XDC,0X09,0XFB,　0XDC,0X11,0X85,0X97,
0X19,0X6A,0X0B,0X32

3. 修改加密算法，令 Nr ＝ 10，密钥的所有字节都是 0。使用步骤 2 中的明文，计算并记

录密文。
（答案：e5 27 93 6d 04 9f 88 87 2a 49 03 30 5b 97 5b d1）

4. 将步骤 2 中的明文，改变 1 位，其它不变。记录密文中有多少字节发生变化。多重复几次，观察 AES 的扩散和混乱的效果。
5. 实现 AES 解密算法，要求同步骤 1。使用步骤 2 中的数据测试算法。
6. 修改解密算法，令 Nr=10，Key 为全 0，密文为步骤 2 中的密文。运行解密算法，记录明文。

注意，要求将源代码以及每一步的结果写入报告，

五、思考题

研究 AES 密文的随机性，探讨使用加密算法做为随机数发生器的可行性。

附录

1. GF(2)多项式乘法运算

GF(2)上的多项式系数仅有 0，1 两种情况。GF（2）的 7 次多项式可以使用 8 位二进制数表示：例如 x7+x5+1 可以表示为（10100001）。算法中的多项式的输入输出都是以这种形式表示的。不可约多项式，一般的对于有限域 GF（2^8）上都是 8 次多项式，其最高阶的项的系数比为 1，因此可以省略表示，于是用一个 8 位二进制数表示。

输入：两个最高次数为 7 次的多项式 A(x),B(x)，一个 GF(2)上的 8 次多项式 M(x)
输出：A(x)*B(x) modM(x)
具体 C 语言形式代码如下：

```
BYTE Multi_P(BYTE Ax, BYTE Bx, BYTE Mx)
{
    BYTE Cx=0;
    for(int i=0; i<8;i++)
    {
        Cx = ((Cx & 0x80) ==0)?(Cx<<1):(Mx^(Cx<<1));
        if((Ax&0x80)!=0)
        {
            Cx = Cx ^ Bx;
        }
        Ax = Ax<<1;
    }
    return Cx;
}
```

2. 有限域 GF（2^8）上的运算

有限域 GF（2^8）的元素为多项式，最高次数小于等于 7
加法：多项式加法，也就是对应的系数相加模 2。若以二进制数表示一个多项式，那么两个多项式相加相当于两个二进制数相异或。
代码如下：

```
BYTE GF_add(BYTE x,BYTE y){  return ( x ^ y ); }
```

乘法：多项式 A(x),B(x),不可约多项式 M(x)。
A(x)*B(x) mod M(x)
计算方法有两种：
1） 通过直接调用函数 Multi_P 实现。
2） 根据有限域的性质，采用指数形式计算。

即：每个元素 a 都可以表示为 p^x，p 是域中的本原元，x=log(a)。元素 a,b 相乘的结果相当于：

$$a \bullet b = \begin{cases} p^{\log(a)} \bullet p^{\log(b)} = p^{[\log(a)+\log(b)] \bmod 2^8-1}, & a,\text{b全不为0} \\ 0, & \text{，} a,b\text{至少有一个为0} \end{cases}$$

于是 ，如果建立 GF（2^8）所有元素的对数表，及指数表，那么就可以通过查表得到结果。显然同第一种方法相比，这种计算方法更快。

代码如下：

```
BYTE GF_Mul(BYTE x, BYTE y)
{
    if(x!=0 && y!=0)
    {
        // GF_2_8_pow 是指数表，GF_2_8_log 是对数表
        return GF_2_8_pow[(GF_2_8_log[x] + GF_2_8_log[y])%0xFF];
    }
    else
    {
        return 0;
    }
}
```

至于指数表、对数表的建立，可以通过调用 Multi_P 来实现，比如

$$p^3 \bmod m(x)=(((p*p) \bmod m(x))*p) \bmod m(x)$$

调用两次函数 Multi_P 就可以得到 $p^3 \bmod m(x)$。通过这种方式就可以得到指数表，然后由指数表反推就得到对数表。这里需要特别注意对于 0 的处理。

逆运算：已知任意一个非零元素 a，求其逆元 a^{-1}，使得 $a \cdot a^{-1}=1$。

求逆元的快速的方法，就是通过查表，先求出 log(a),然后，计算逆元的指数 log(a)＝，最后通过查指数表求出 p，也就是 a 的逆元。

特别地：0 没有逆元，直接返回 0。

算法如下：

```
BYTE GF_Reverse(BYTE x)
{
    // GF_2_8_pow 是指数表，GF_2_8_log 是对数表
    return x!=0?GF_2_8_pow[(0xFF-GF_2_8_log[x])]:0;
}
```

3. AES 基本运算

为了简化运算，接下来介绍的算法都是基于这样的假设：

a) 算法密钥为 128 位，每个块为 128 位，也就是 16 个字节。
b) 每种变换算法的输入都是一个块，也就是每次处理 16 个字节

AES 加密过程由 4 个基本基本动作组成，包括：S 盒变换、行移位变化、列混合变换、轮密钥加法变换。接下来，我们看看这些细节是如何实现的。

1） S 盒变换

算法：通过查表 SBox 实现。以 16 个字节为基本单位，分别对每个字节查表替换以相应的表中内容。

```
void SubBytes(BYTE state[16])
{
    for(int i=0; i<16;i++)
    {
        state[i] = SBox[state[i]];
    }
    return;
```

```
}
```

SBox 可以通过计算生成。也可以以全局数组的形式直接写入代码中。

以下方法用来计算 SBox[x]

```
BYTE c= 0x63;
求字节 x 的逆：b = GF_Reverse(x)
然后，依次计算每一位
for i=0 to 7
    //bit(b,i)返回字节 b 的第 i 位的值：0 或 1。⊕为异或预算
    B= bit(b, i) ⊕bit(b, (i+4)mod8) ⊕ bit(b, (i+5) mod8)
        ⊕ bit(b, (i+6)mod8) ⊕bit(b, (i+7)mod8) ⊕ bit(c, i)
    setbit(SBox[x], i, B) //设置 SBox[x]的第 i 位为 B
endfor
```

2） 行移位变换

这个变换很简单：输入为 state[16]。state[0]-state[3]不变，state[4]-state[7]循环左移 1 个字节，state[8]-state[11] 循环左移 2 个字节，state[12]-state[15] 循环左移 3 个字节

算法如下：

输入：16 个字节构成的一个块 state[16]

输出：变换后的结果保存在 state[16]

```
Void ShiftRows( BYTE state[16])
{
    BYTE t[4];
    for(int i=1; i<4; i++)
    {
        for( int j=0; j<4; j++ )
        {
            t[j] = state[ i*4 + (j+i)%4 ];
        }
        memcpy(state+4*i, t, 4);
    }
 }
```

3） 列混合变换

输入：16 个字节构成的一个块：state[16]

输出：变换后的结果保存在 state[16]

算法：按列处理，每列表示为 $s(x) = S[3][i]x^3+ S[2][i]x^2+ S[1][i]x+ S[0][i]$，然后计算 $[\{03\}x^3+\{01\}x^2+\{01\}X+\{02\}]*s(x) \mod (x^4+1)$ 得到新的多项式，其系数对应 S[0][i]-S[3][i]变换后的字节值。

代码如下：

```
Void MixColumns(BYTE state[16])
{
    BYTE temp[4][4];
    memcpy(temp, state, 16);            //将 state 复制到 temp 中，temp[i][j]=state[i*4+j]
    for(int i=0; i<4; i++)              //第 i 列
    {
        for(int j=0; j<4; j++)          //第 j 行，⊕为异或运算
        {
            state[j*4+i] =
                GF_Mul(0x2, temp[j][i])
                ⊕GF_ Mul (0x3, temp[(j+1) mod 4][i])
                ⊕temp[ (j+2)mod 4 ][i] ⊕temp[ (j+3)mod 4 ][i] ;
        }
    }
}
```

4） 轮密钥加法变换

每个块的加密都要经过 Nr（不妨取 Nr＝10）轮变换。在第 i 轮变换，使用第 i 个扩展密钥 k(i)

变换是输入与密钥异或的过程

代码如下：

```
void AddRoundKey(BYTE state[16],BYTE key[16])
{
    for(int i=0; i<4; i++)
    {
        for( int j=0; j<4; j++ )
        {
            state[ i*4 + j] = state[ i*4 + j]  ⊕key[j*4+i];
        }
    }
}
```

5） 密钥扩展算法

选取 128 位的密钥，也就是 16 个字节的密钥，通过扩展算法，生成 Nr 个扩展密钥。

对于第 r 轮(r 从 1 开始计数)用的密钥

输入： r 表示当前要扩展第 r 轮密钥，r>0

数组 key 的前 16 字节放入第 r-1 轮密钥

输出： 数组 key 的后 16 字节写入第 r 轮密钥

注意： 第 0 轮的密钥也就是初始密钥

```
void make_key(BYTE key[32], int r)
{
    const BYTE RC[10]={0x1,0x2,0x4,0x8,0x10,0x20,0x40,0x80,0x1B,0x36};
    BYTE *temp = key;
    DWORD *dw_key= (( DWORD*)(temp));
    //存储第四个双字，也就是最后 4 个字节
    DWORD dw3= dw_key[3];
    //对第一个字，旋转并替换
    BYTE t = temp[4*3+0];
    temp[4*3+0] = gSBox[temp[4*3+1]];
    temp[4*3+1] = gSBox[temp[4*3+2]];
    temp[4*3+2] = gSBox[temp[4*3+3]];
    temp[4*3+3] = gSBox[t];
    //对第一个字，异或
    temp[4*3] = temp[4*3]^RC[r-1];
    //第 i－1 个字 与 第 i-3 个字
    for(int i=4; i<7;i++)
    {
        dw_key[i] = dw_key[i-1]^dw_key[i-4];
    }
    dw_key[7] = dw_key[6]^dw3;
    dw_key[3] = dw3;
}
```

通过反复调用函数 make_key 就可以生成任意轮扩展密钥。对于 128 位的密钥，可以取 Nr=10。

4. AES 加密算法

这里仅仅描述对 1 个块的加密，对更长的数据，则需要依据不同的模式，使用加密算法。

算法如下：

输入：明文 in ， 密钥 key

输出：密文 out

AESEncrypt(BYTE in[16],BYTE out[16]，BYTE key[16])

```
{
    BYTE state[16];
    int Nr = 10;
    将 in 看作 4×4 的矩阵，做转置运算：state[ i*4+j ] = in[ j*4+i ]
    生成 10 轮扩展密钥，记录在数组 key_expand[11][16]中
    AddRoundKey（state, key_expand[0], 1）
    for( round = 1; round<Nr; round++)
    {
        SubBytes(state);
        ShiftRows(state);
        MixColumns(state);
        AddRoundKey(state, key_expand[round]);
    }
    将 state 看作 4×4 的矩阵，做转置运算：out[ i*4+j ] = state[ j*4+i ]
}
```

5. AES 解密算法

解密算法同加密算法的过程基本相同，具体运算需要 S 盒变换、行移位、列混合的逆运算。扩展密钥使用顺序相反，先用第 Nr 轮扩展密钥，在使用 Nr－1 轮扩展密钥，直到第 0 轮扩展密钥。过程不再详述。

报告要求：

1） 要求字数不少于 1000 字

2） 列出参考文献。包括作者，标题，出处，日期等。

3） 可以两人合作。请在报告中注明你们是如何分工的。

实验 3　RC4 密码体制

一．实验目的

通过实现 RC4 算法的实现，以及对实际的数据进行加密和解密，加深对序列密码体制的理解。

二．实验环境

硬件设备：PC 机

操作系统：windows 2000、XP 或 windows 7 操作系统

编程工具：Matlab，VisualStudio 2005 或 Eclipse

三．RC4 简介

RC4 是 Ron Rivest 在 1987 年设计的密钥长度可变的序列密码。RC4 是 WEP[1]中采用的

[1] WEP：全称 Wireless Encryption Protocol(无线加密协议)，是 1999 年 9 月通过的 IEEE 802.11 标准的一部分。WEP 的设计是要提供和传统有线的局域网路相当的机密性，不过密码分析学家已经找出有线等效加密几个弱点，因此在 2003 年被实现大部分 IEEE 802.11i 标准的 Wi-Fi Protected Access 淘汰，又在 2004 年由实现完整 IEEE

加密算法，也是安全套接层（SSL）可采用的算法之一。RC4 的密钥长度可变，范围是 1～255 字节。RC4 一个字节一个字节地加解密。由于异或运算的对合性，RC4 加密解密使用同一套算法。

RC4 由两个部分组成。

（1）初始化长度为 256 的 S 盒。第一个 for 循环将 0 到 255 的互不重复的元素装入 S 盒。第二个 for 循环根据密钥打乱 S 盒。

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for( i=0 ; i<256 ; i++)
    j := (j + S[i] + Key[i mod keylength]) % 256
    swap(S[i], S[j])
endfor
```

（2）下面 i,j 是两个指针。每收到一个字节，就进行 while 循环。通过一定的算法（a,b）定位 S 中的一个元素，并与输入字节 InputByte 异或，得到 OutputByte。循环中还改变了 S(c)。如果输入的是明文，输出的就是密文；如果输入的是密文，输出的就是明文。

```
i := 0
j := 0
while (true):
    i := (i + 1) mod 256                    //a
    j := (j + S[i]) mod 256                 //b
    swap(S[i], S[j])                        //c
    k := InputByte ^ S[(S[i] + S[j]) % 256]
    output OutputByte
endwhile
```

四．实验任务

1.使用 C 语言或者其它编程语言，实现 RC4 加密解密算法。

2.若选取密钥 K= 0F 01 0E 02 (十六进制表示的 4 个字节)

消息 M= Attackatdawn（注意区分大小写，按 ASCII 码转换为十六进制数做加密变换）

以十六进制形式写出对应的密文。

3.研究 RC4 密码关于雪崩效应的特性，即当密钥 K 中有 1 位发生变化时，密钥序列（前 100 字节）有多少位发生变化？

密钥（十六进制表示）	发生变化的位	密钥序列变化位数（前 100 字节）

802.11i 标准的 WPA2 所取代。有线等效加密虽然有些弱点，亦足以吓阻非专业人士的窥探。在 2005 年，美国联邦调查局的一组人展示了用公开可得的工具可以在三分钟内破解一个用 WEP 保护的网络。

	(从低向高计数)	
K=0F 01 0E 02	1	
	2	
	3	
	4	
	5	
	6	
	…	…
	30	
	31	
	32	

4.研究密钥序列的随机性：

选定密钥 K= 0F 01 0E 02　(十六进制表示)，生成密钥序列的前 2500 个字节

（1）统计序列中的 1 的个数（注意：共有 20000 位），看看是否在 9651～10346 之间。

（2）游程测试(详见附录 1)：统计序列中的长度为 1，2，3，4，5 和 6 以上的游程数。看看是否符合附录中游程检验标准的表格。游程是指连续的 1 序列。

例如在 0110111101 中，游程长为 1 的个数：1；游程长为 2 的个数：1；游程长为 3 的个数：0；游程长为 4 的个数：1；游程长为 5 的个数：0；游程长为 6+的个数：0。

五．附录

1.FIPS 140-1 中的关于伪随机数检验的内容[2]

[2] （http://csrc.nist.gov/publications/fips/fips140-1/fips1401 .pdf）

The Monobit Test

1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X.

2. The test is passed if 9,654 < X < 10,346.

The Poker Test

1. Divide the 20,000 bit stream into 5,000 contiguous 4 bit segments. Count and store the number of occurrences of each of the 16 possible 4 bit values. Denote *f(i)* as the number of each 4 bit value *i* where $0 \le i \le 15$.

2. Evaluate the following:

$$X = (16/5000) * \left(\sum_{i=0}^{15} [f(i)]^2 \right) - 5000$$

3. The test is passed if 1.03 < X < 57.4.

The Runs Test

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros, which is part of the 20,000 bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.

2. The test is passed if the number of runs that occur (of lengths 1 through 6) is each within the corresponding interval specified below. This must hold for both the zeros and ones; that is, all 12 counts must lie in the specified interval. For the purpose of this test, runs of greater than 6 are considered to be of length 6.

Length of Run	*Required Interval*
1	2,267 - 2,733
2	1,079 - 1,421
3	502 - 748
4	223 - 402
5	90 - 223
6+	90 - 223

The Long Run Test

1. A long run is defined to be a run of length 34 or more (of either zeros or ones).

2. On the sample of 20,000 bits, the test is passed if there are NO long runs.

2.位运算

1）统计一个字节 B 中的 1 的个数

```
int count =0;
for(i=0; i<8; i++)
{
    if( ( B & 0x80 ) != 0 )
    {//B 的最高字节等于 1
        count++;
```

```
        }
        else
        {//B 的最高字节等于 0

        }
        B = B<<1; //左移 1 位
    }
```

2) 将一个字节转化为二进制形式的字符串。结果写入 buffer 字符数组

```
char    buffer[8]
B=0xA5;
for( i =0; i<8 ;i++)
{
        if( (B & 0x80) ==0)
                buffer[i]='0';
        else
                buffer[i]='1';
        B=B<<1;
}
```