分组密码算法AES

1. 程序设计

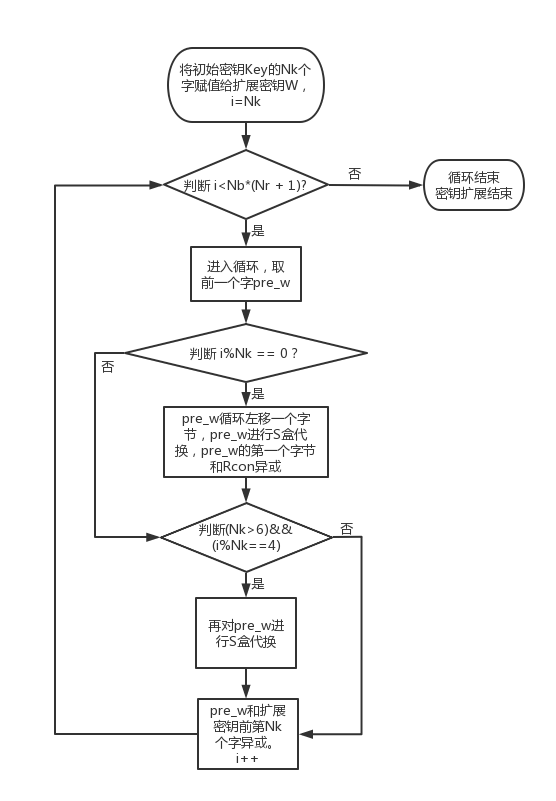
存储格式：本程序中输入的明文、密钥，加密生成的密文，以及解密生成的明文的存储格式都是unsigned char类型，每个char存储一个字节，且存储的数据形式为16进制数字。

整个程序构造了一个AES类，其中包含成员变量明文分组Nb，密钥分组Nk，加密轮数Nr，初始密钥、扩展后密钥。针对不同的密钥长度Nk应当有不同的加密轮数Nr：当密钥长度为128位、192位或256位，相应的迭代轮数为10轮、12轮、14轮，因此在程序的开头设置了一个enum常量数组，用来标记初始密钥的长度。还包含常数数组S盒、逆S盒、常数Rcon。且其他的函数均为成员函数，主要的函数是密钥扩展函数、加密函数、解密函数。

下面重点介绍扩展函数、AES加密函数、AES解密函数的实现：

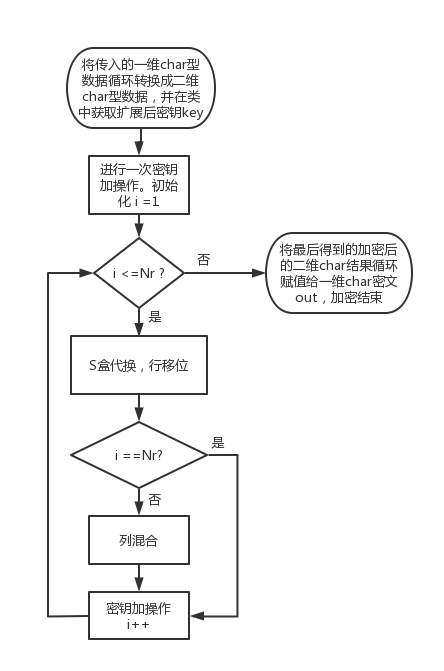
1. 密钥扩展函数：

密钥扩展函数，首先将初始密钥的Nk个字赋值给扩展函数，之后进入循环进行扩展密钥，循环标记i从Nk开始，直到第Nb\*(Nr + 1)个字停止。进入循环后，先取前一个字，当i%Nk == 0时，进行行移位、S盒代换、并且让前一个字的第一个字节和常数Rcon异或，当Nk>6时，且有i%Nk==4的情况时，让前一个字再进行一次S盒代换。最后让进行完上述操作的前一个字和前Nk个字进行异或操作。

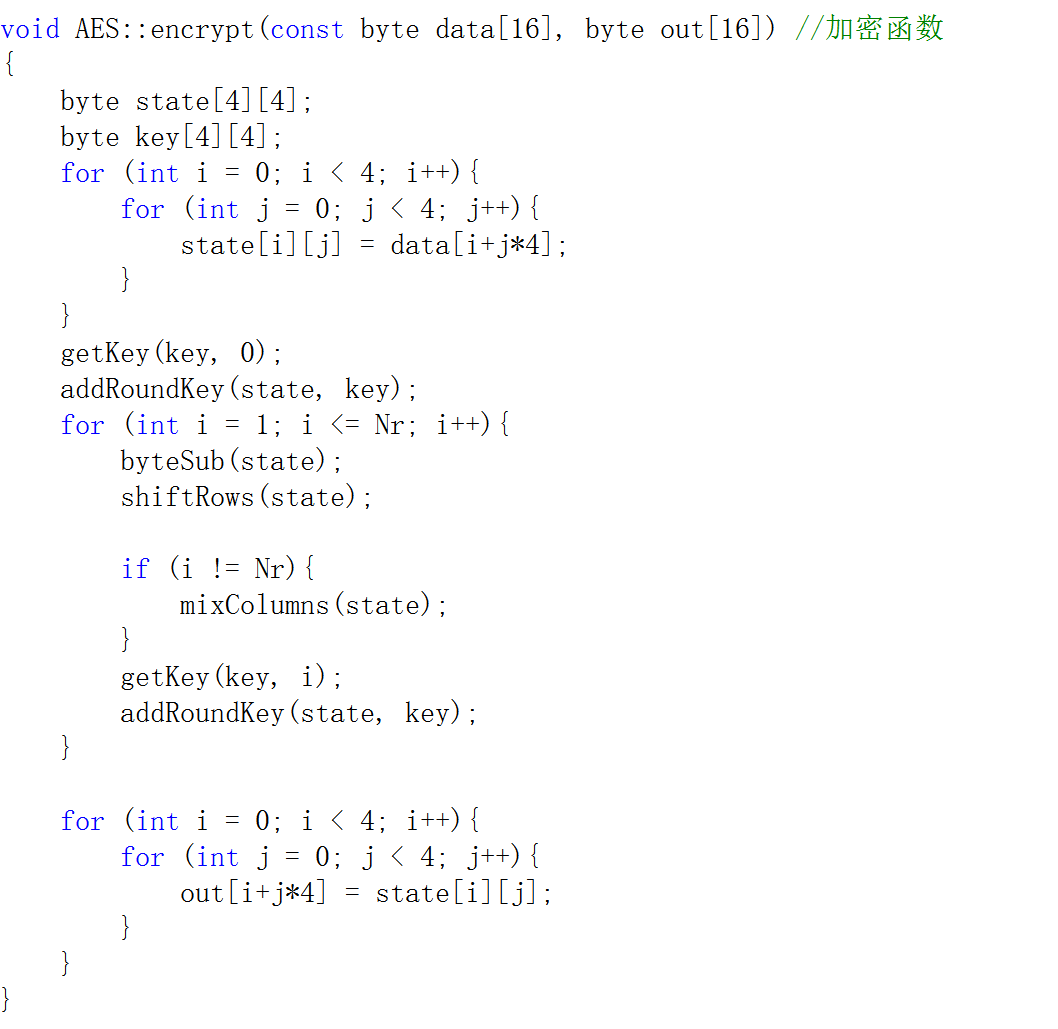


（二）AES加密

加密流程：对明文状态的一次密钥加操作，轮轮加密和末尾轮轮加密，最后得到密文。其中轮轮加密每一轮有四个部件，包括字节代换部件ByteSub、行移位变换ShiftRow、列混合变换NixColumn和一个密钥加AddRoundKey部件，末尾轮加密和前面轮加密类似，只是少了一个列混合变换NixColumn部件。

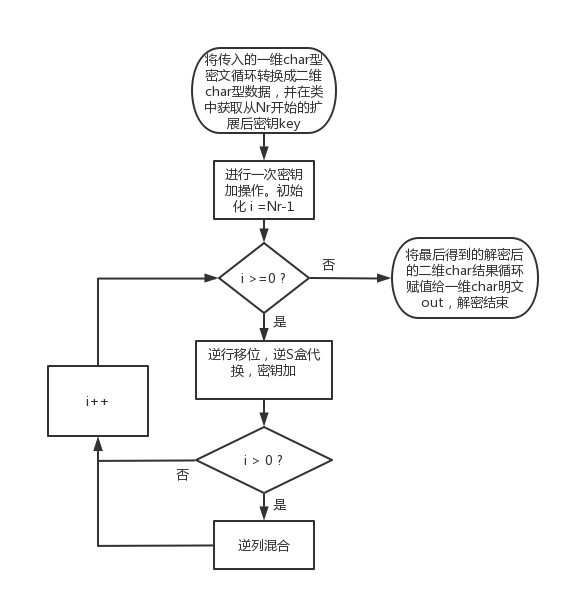


函数实现代码如下：

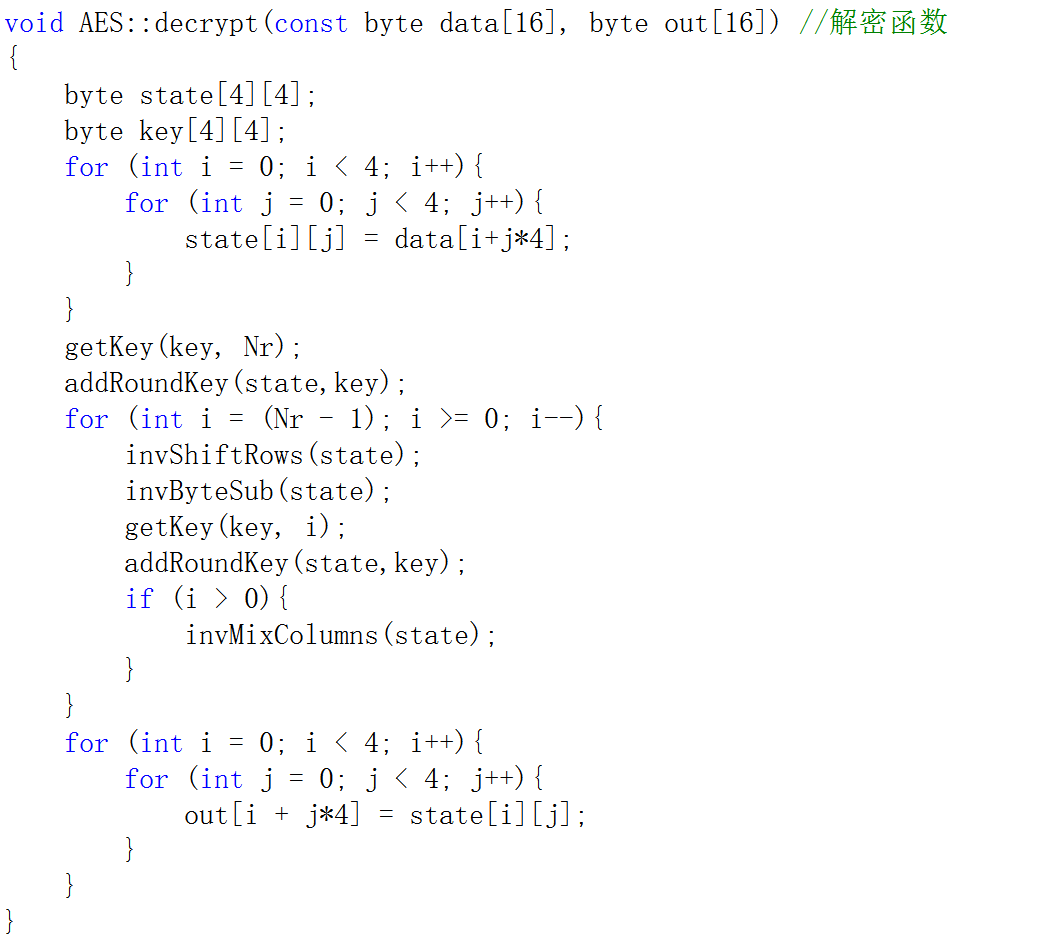


（三）AES解密

解密过程和加密过程流程基本一样，先经过一个密钥加，然后进行轮轮解密和末尾轮轮解密，最后得到明文。和加密不同的是轮轮解密每一轮四个部件都需要用到它们的逆运算部件，包括字节代换部件的逆运算、行移位变换的逆变换、逆列混合变换和一个密钥加部件，末尾轮加密和前面轮加密类似，只是少了一个逆列混合变换部件。且解密时的子密钥是从扩展密钥中倒着选取的。



解密函数代码如下：



1. 实验结果展示

在主函数中编写了测试程序，用到两组测试数据如下：

第一组：

Keysize=128

明文（16进制）：0001, 0001, 01a1, 98af, da78, 1734, 8615, 3566

密钥（16进制）：0001, 2001, 7101, 98ae, da79, 1714, 6015, 3594

密文（16进制）：6cdd, 596b, 8f56, 42cb, d23b, 4798, 1a65, 422a

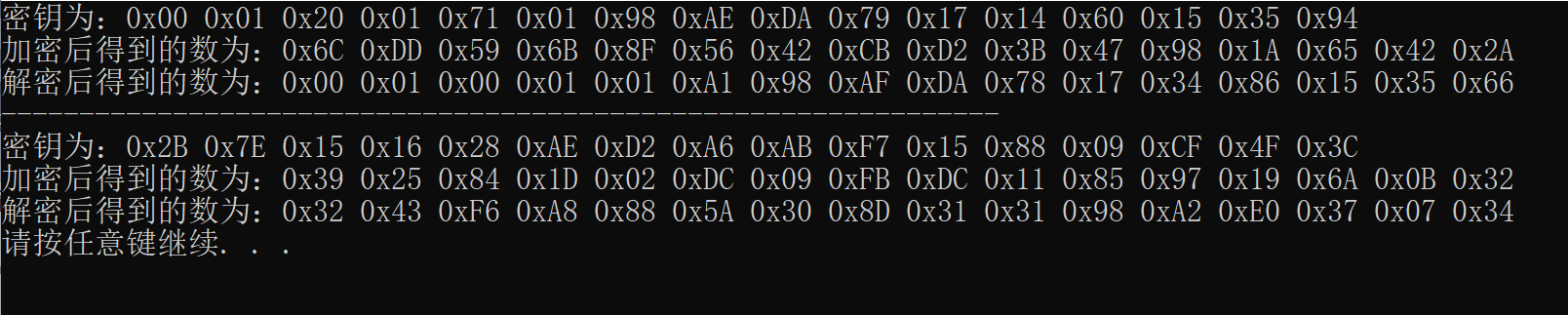
第二组：

Keysize=128

明文（16进制）：3243, f6a8, 885a, 308d, 3131, 98a2, e037, 0734

密钥（16进制）：2b7e, 1516, 28ae, d2a6, abf7, 1588, 09cf, 4f3c

密文（16进制）：3925, 841d, 02dc, 09fb, dc11, 8597, 196a, 0b32



1. 雪崩效应检验

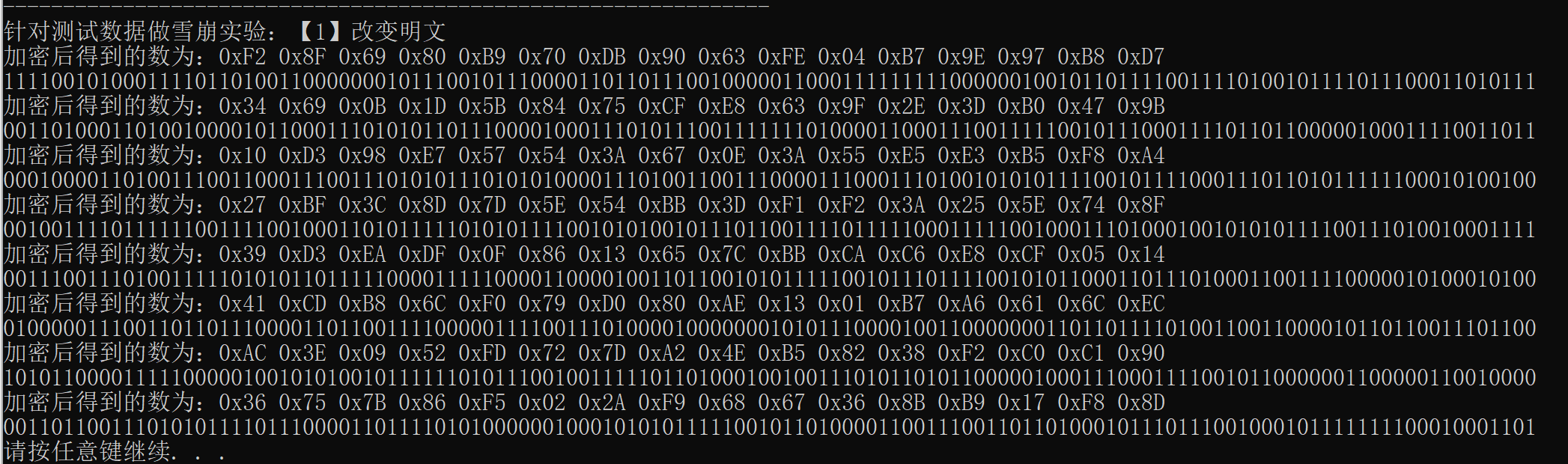
对第一组测试数据进行雪崩效应检验，

1、固定密钥密钥（16进制）：0001, 2001, 7101, 98ae, da79, 1714, 6015, 3594

初始明文：0001, 0001, 01a1, 98af, da78, 1734, 8615, 3566

分别改变第一个16进制数为1，2，4，8，以及改变第二个16进制数为1，2，4，8

改变后得到的密文：



编写程序，统计和原来的密文相比，改变的bit位数：

char s0[129] = "01101100110111010101100101101011100011110101011001000010110010111101001000111011010001111001100000011010011001010100001000101010";

char s1[129] = "11110010100011110110100110000000101110010111000011011011100100000110001111111110000001001011011110011110100101111011100011010111";

char s2[129] = "00110100011010010000101100011101010110111000010001110101110011111110100001100011100111110010111000111101101100000100011110011011";

char s3[129] = "00010000110100111001100011100111010101110101010000111010011001110000111000111010010101011110010111100011101101011111100010100100";

char s4[129] = "00100111101111110011110010001101011111010101111001010100101110110011110111110001111100100011101000100101010111100111010010001111";

char s5[129] = "00111001110100111110101011011111000011111000011000010011011001010111110010111011110010101100011011101000110011110000010100010100";

char s6[129] = "01000001110011011011100001101100111100000111100111010000100000001010111000010011000000011011011110100110011000010110110011101100";

char s7[129] = "10101100001111100000100101010010111111010111001001111101101000100100111010110101100000100011100011110010110000001100000110010000";

char s8[129] = "00110110011101010111101110000110111101010000001000101010111110010110100001100111001101101000101110111001000101111111100010001101";

double sum = 0;

int temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s1[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s2[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s3[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s4[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s5[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s6[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s7[i])

temp++;

}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

temp = 0;

for(int i = 0; i <128; i++){

if(s0[i]!=s8[i])

temp++;

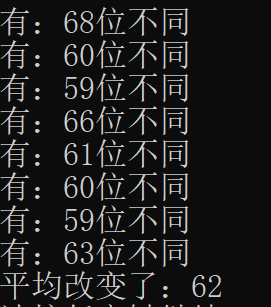
}

cout<<"有："<<temp<<"位不同"<<endl;

sum +=temp;

cout<<"平均改变了："<<sum/8<<endl;

结果显示：



如图所示，平均改变的位数位62位。

1. 固定明文：0001, 0001, 01a1, 98af, da78, 1734, 8615, 3566

初始密钥：0001, 2001, 7101, 98ae, da79, 1714, 6015, 3594

分别改变密钥的第一个16进制数为1，2，4，8，以及改变第二个16进制数为1，2，4，8

编程方法和上述相同：

得到结果如下，平均改变的位数为63.5位。

