实验四 AES实验

> 17341125 蒙亚愿 信息安全

一、实验目的

> 深入了解AES密码体制，掌握加密解密的算法，实现给定明文的加密。

二、实验要求

> 实验要求：
>
> 实现128位高级加密标准AES对以下信息进行加密并验证：
>
> 加密字符串：
>
> ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschoolyoumultiplythesquarefootageofthewallsbythecubiccontentsofthefloorandceilingcombinedanddoubleityouthenallowhalfthetotalforopeningssuchaswindowsanddoorsthenyouallowtheotherhalfformatchingthepatternthenyoudoublethewholethingagaintogiveamarginoferrorandthenyouorderthepaper（实验1第1题结果,全小写无空格无标点）
>
> 密钥：姓名全拼 例：田杨童->tianyangtong
>
> 工作模式：CBC
>
> 密钥偏移量IV：123（字符串"123"）
>
> 补码方式：PKCS5Padding
>
> 密文编码方式：16进制

三、实验介绍

3.1 实验综述

我采用的是AES-128-CBC，所以密钥和$Block$是16个字节，需要迭代10轮。

加密：输入明文字符串，先把明文字符串按照ASCII码转换成易于计算的整型数组，接着进行分组，每16个字节一块。对于每一块逐次进行加密，其中包括：轮密钥加，字节代换，行变换，列混合。其中，轮密钥是经过初始密钥$mengyayuan$经过密钥调度算法得到11组密钥，用于10轮的加密。由于使用了CBC模式，第一块需要先与制定的偏移量iv进行异或，经过10轮之后，得出的密文再作为下一块明文的偏移量，与该块明文异或，继续进行10轮的加密。最终使所有的$Block$都得到了加密，输出密文。

解密：解密就是加密的逆过程，并且使用同样的轮密钥。已知密文字符串，同样把密文转类型、分组。逆过程，即需要轮密钥也是反方向使用，也就是从最后一个密钥开始。同样，对于每一个密文块，都经过逆向轮密钥加、逆向行移位、逆向字节代换、逆向列混淆。并且，对于每一块经过十轮之后的得到的密文块，都要与前一块密文块异或，才能得到最后的明文块。第一块密文块则是在最后与偏移量iv异或得到。

3.2 开发环境

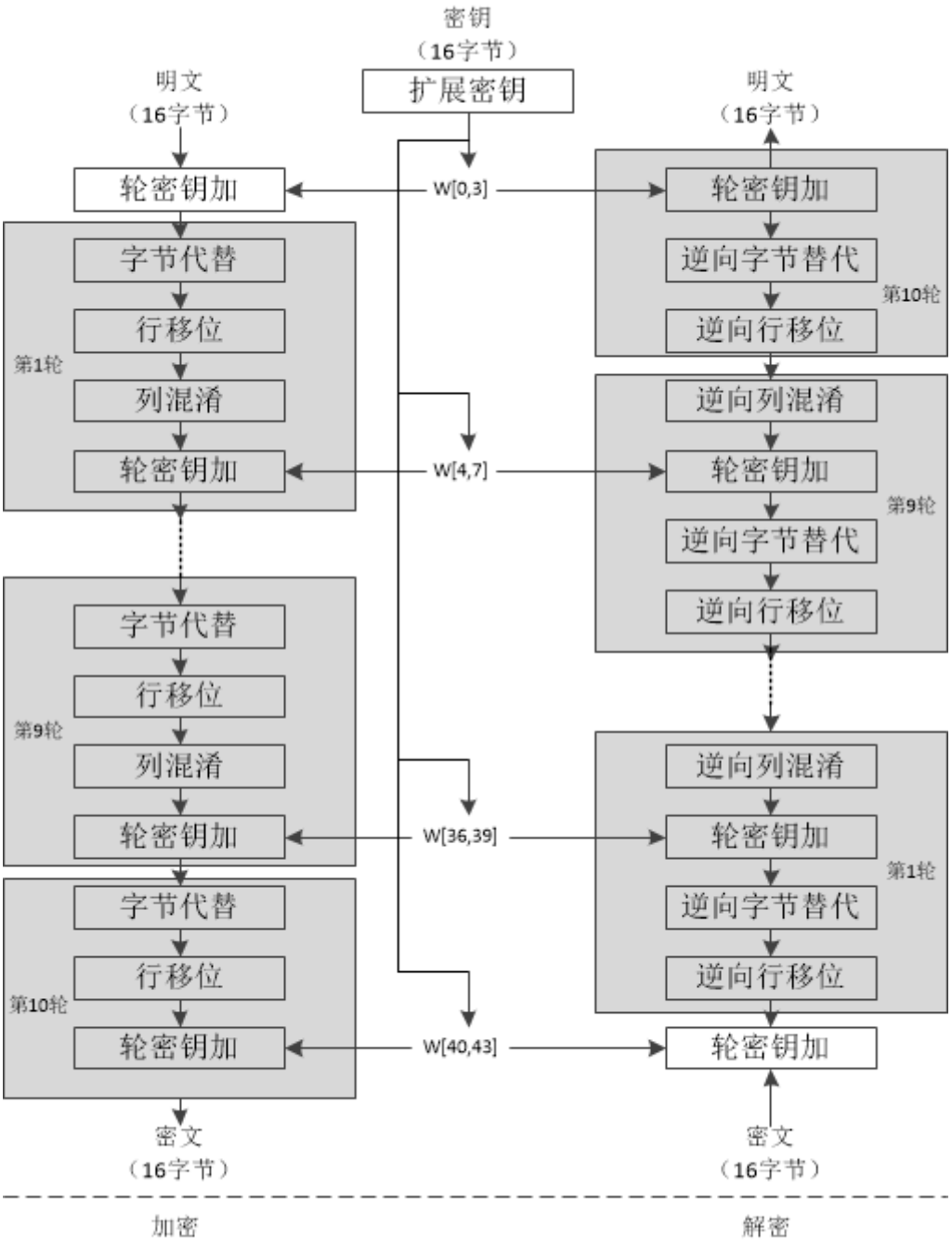
运行环境：Windows

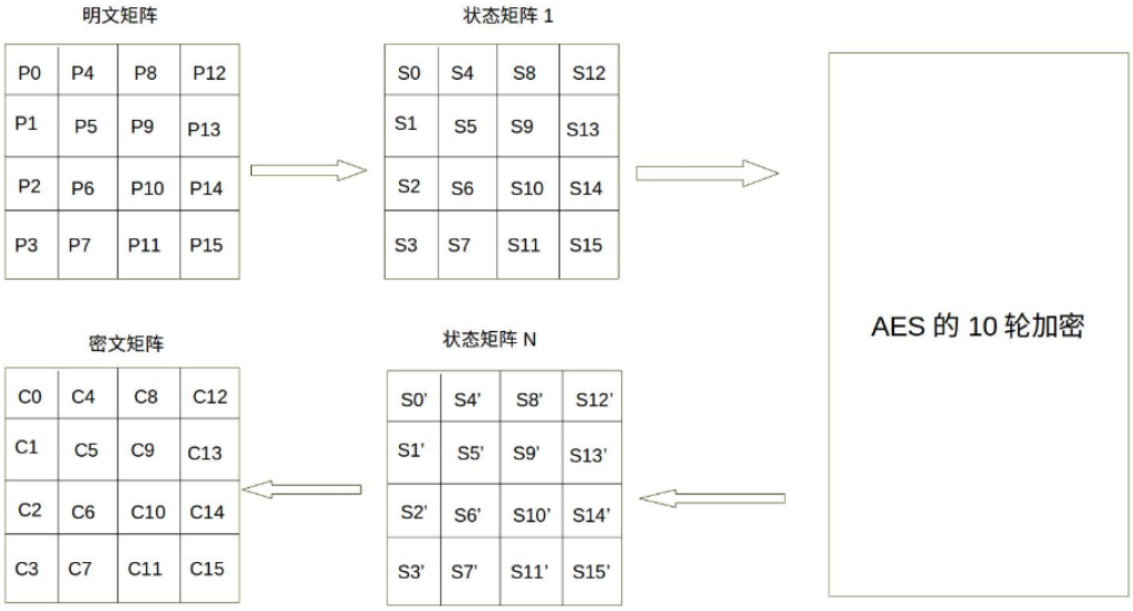
使用语言：C++

四、算法原理

加密解密的关系及步骤如下图：可见加密和解密互为逆过程。解密的每一步都对应加密的逆操作；加密解密所有操作的顺序使相反的。所以，保证了明文能够正确地恢复成明文。

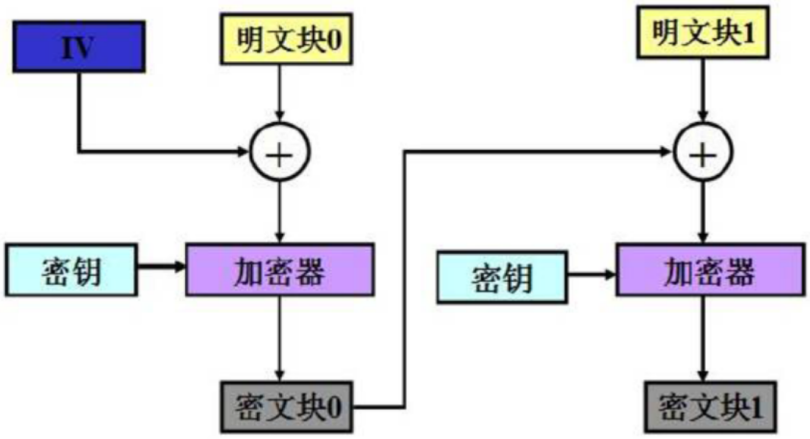
加密解密中每轮的密钥都是由初始密钥拓展得到的，算法中16字节的明文、密文和轮密钥都用矩阵表示。初始偏移量$iv_0 = '123'||0000000000000$（4个字节），初始密钥为$[roundKey_0，roundKey_3] = 'mengyayuan'||000000$（$roundKey_i$为4个字节）。





4.1 CBC_偏移量

加密：由于使用CBC模式，在每一块明文块进行10轮的加密迭代之前，都需要与偏移量IV异或，定义初始$IV_0 = 123$,由于每一块有16个字节，还需要在IV_0后补充补充13个数字0，IV_0与明文块$Block_0$异或，$Block_0$经过10轮的加密之后得到的密文块，作为下一个明文块$Block_1$的偏移量IV_1，依此类推到最后一块。



解密：解密时，与偏移量异或同样也是逆操作。从最后一块开始，前一块的密文就作为其偏移量IV_{Nr}，与经过十轮的解密迭代之后得到的密文块异或，就得到最后一块对应的明文。以此往前推，第一块的偏移量就是初始偏移量IV_0，与密文异或，即得到第一块明文。

4.2 轮密钥加

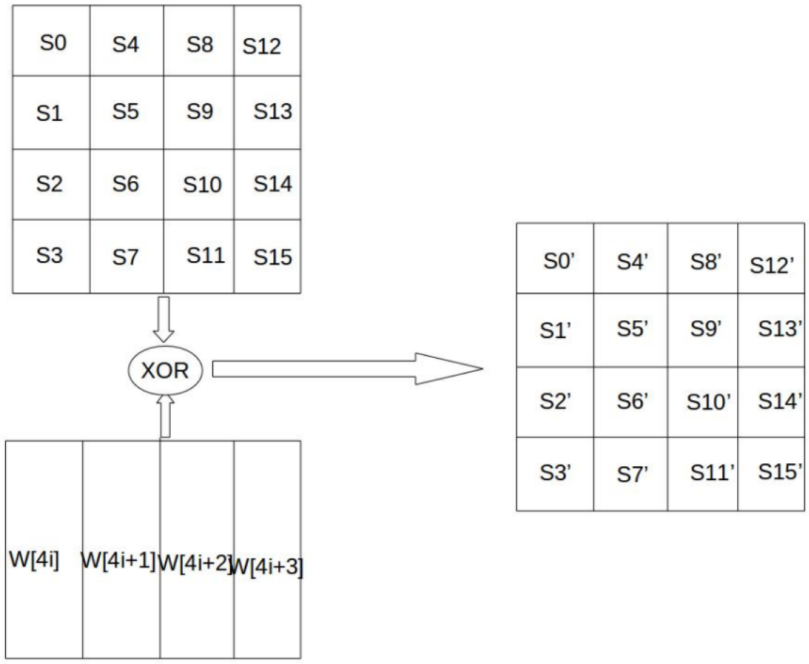
加密：对于由初始密钥调度生成的11个轮密钥，都用在每一块的10轮加密中。在最开始，明文在与偏移量IV异或之后，还要在与初始密钥$roundKey_0$异或。在接下来的十轮中，则与图中对应的轮密钥异或。

解密：由于是从最后一块开始的，按照逆过程，则最后一块密文块就是与最后一轮的轮密钥$roundKey[40，43]$异或，之后的十轮的轮密钥异或则是从$roundKey[36，39]$开始，倒序异或。

```
// function--addRoundKey
// 加密
for i 0 to Block_Cnt do
    block[i] ^= roundKey[0]||roundKey[1]||roundKey[2]||roundKey[3]
    for round 1 to 10 do
        block[i] ^= roundKey[4*round]||roundKey[4*round+1]||
                    roundKey[4*round+2]||roundKey[4*round+3]
    end
end
```

```
// function--addRoundKeyInv
// 解密
for i Block_Cnt-1 to 0 do
    block[i] ^= roundKey[43]||roundKey[42]||roundKey[41]||roundKey[40]
    for round 9 to 1 do
        block[i] ^= roundKey[4*round]||roundKey[4*round+1]||
                    roundKey[4*round+2]||roundKey[4*round+3]
     end
 end
```

其中的异或可以看作是每个字的逐位异或。S_i是一个字节，$Wi = roundKey[4i]\ ||\ roundKey[4i+1]\ ||\ roundKey[4i+2]\ ||\ roundKey[4i+3]$是一个字，$S_0, S_1, S_2, S_3$组成的32位的字与$W_i$异或。



4.3 字节代换

通过运算可以把域$GF[2^8]$中所有的对应的代换全部写出，提高效率。于是，字节代换就成了简单的查表操作，定义S盒：

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

在状态矩阵中，对于每一个字节$S_{a,b}$，$x = S_{a,b}/16, y = S_{a,b}\ mod\ 16$，所以映射之后的$S'_{a,b} = SBox(x, y)$。

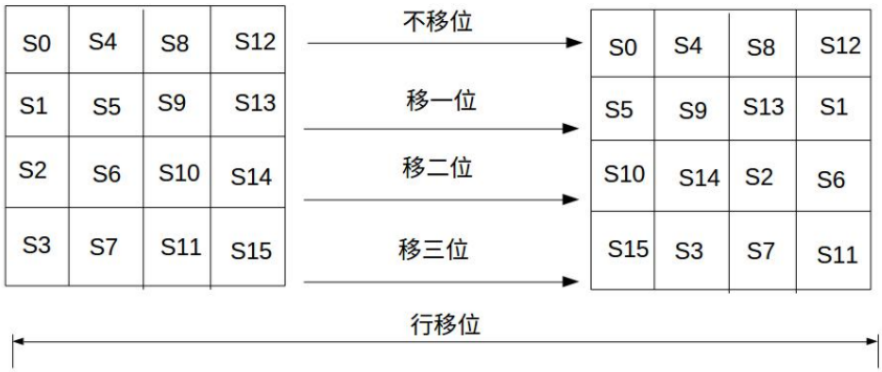
同样的，在解密中，也定义了逆S盒：

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

解密也是逆字节代换以同样的方式映射。

4.4 行移位

加密：每个状态矩阵中的每一行循环**左移**。第i行循环左移位i位（$i = 0, 1, 2, 3$)。如下图：



解密：每个状态矩阵中的每一行循环**右移**。第i行循环右移i位（$i = 0, 1, 2, 3$）

4.5 列混合

加密：列混合是通过矩阵相乘来实现的，在域$GF[2^8]$中，与混合矩阵相乘，得到新的状态矩阵。

$$\text{混合矩阵}mixMatrix=\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

$$\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

以上的矩阵相乘，其实是看作多项式相乘，把每一个数值化成二进制，就可以得到每个数值对应的在域中的元素，相乘之后得到的矩阵需要再模本原多项式，其中的乘法和模运算都是域中的运算。

以上次的有限域运算实验作为参考，在域$GF[2^8]$中，本原多项式为$p(x) = x^8 + x^4 + x^3 + x + 1$，在矩阵相乘的时候，都是把每个数化成有限域中的元素，利用有限域乘法，得出的结果再模$p(x)$，所以乘法和模运算的算法可以轻易得到（详见上次实验报告）。

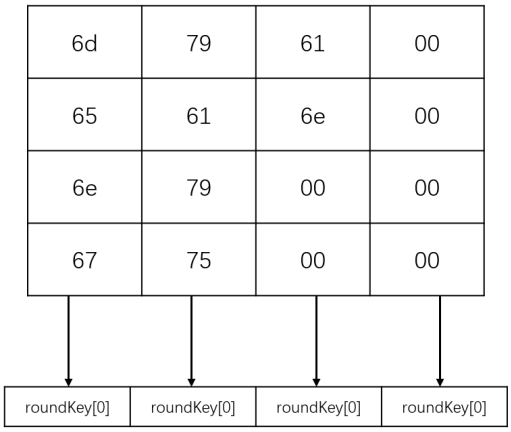
解密：解密与加密唯一的不同就是混合矩阵不同。在解密中的混合矩阵实际上是加密混合矩阵的逆。

$$\text{逆混合矩阵}mixMatrixInv = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \cdot \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{bmatrix}$$

4.6 密钥拓展

轮密钥$roundKey$是一个44×4的数组，最开始的4×4是我们的初始密钥：$mengyayuan$，由于不够16位，需要填充6个数字0，以形成一个4×4的$Block$，用于与明文加密的的最开始的轮密钥加以及接下来的密钥拓展。



这个矩阵的每一列的4个字节组成一个字，对这个数组扩充40个新列，组成一个44×4的拓展密钥数组。拓展方式如下：

对于即将拓展的第i列，

1. **i=4k**

$$roundKey[i] = roundKey[i-4] \bigoplus f(roundKey[i-1])$$

f(rk[i-1])包括以下步骤：

字循环

将rk[i-1]字中的4个字节循环左移1个字节，即

$$[k_0, k_1, k_2, k_3] -> [k_1, k_2, k_3, k_0]$$

字节代换

对字循环之后的结果使用$SBox$进行代换，方法同4.3字节代换。

轮常量异或

轮常量是一个字：由于在十轮的计算中，需要11个轮密钥块，也就是有44个（列）轮密钥，$[0, 43]$中，4的倍数有10个，所以有10个轮常量。

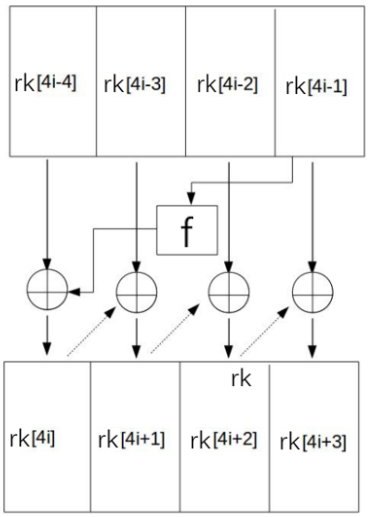
j(轮数)	**1**	**2**	**3**	**4**	**5**
Rcon[j]	01 00 00 00	02 00 00 00	04 00 00 00	08 00 00 00	10 00 00 00
j(轮数)	**6**	**7**	**8**	**9**	**10**
Rcon[j]	20 00 00 00	40 00 00 00	80 00 00 00	1b 00 00 00	36 00 00 00

前两步的结果$t[i-1]$，与$Rcon[i/4]$异或，得到最后的结果。

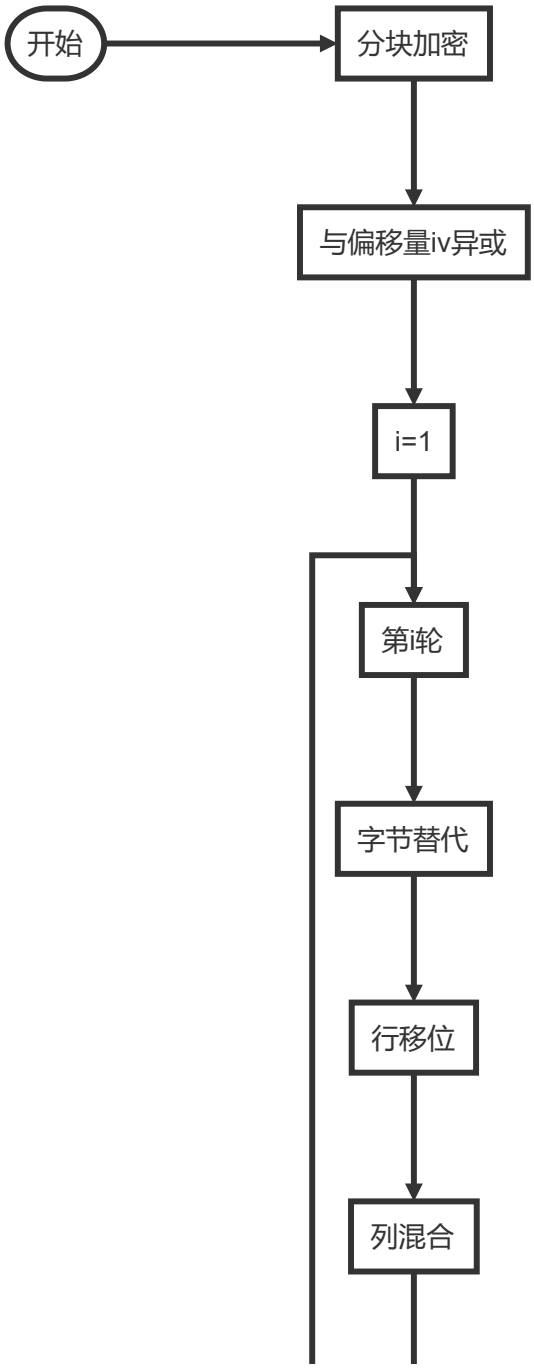
所以，第i列的密钥就可以计算出。

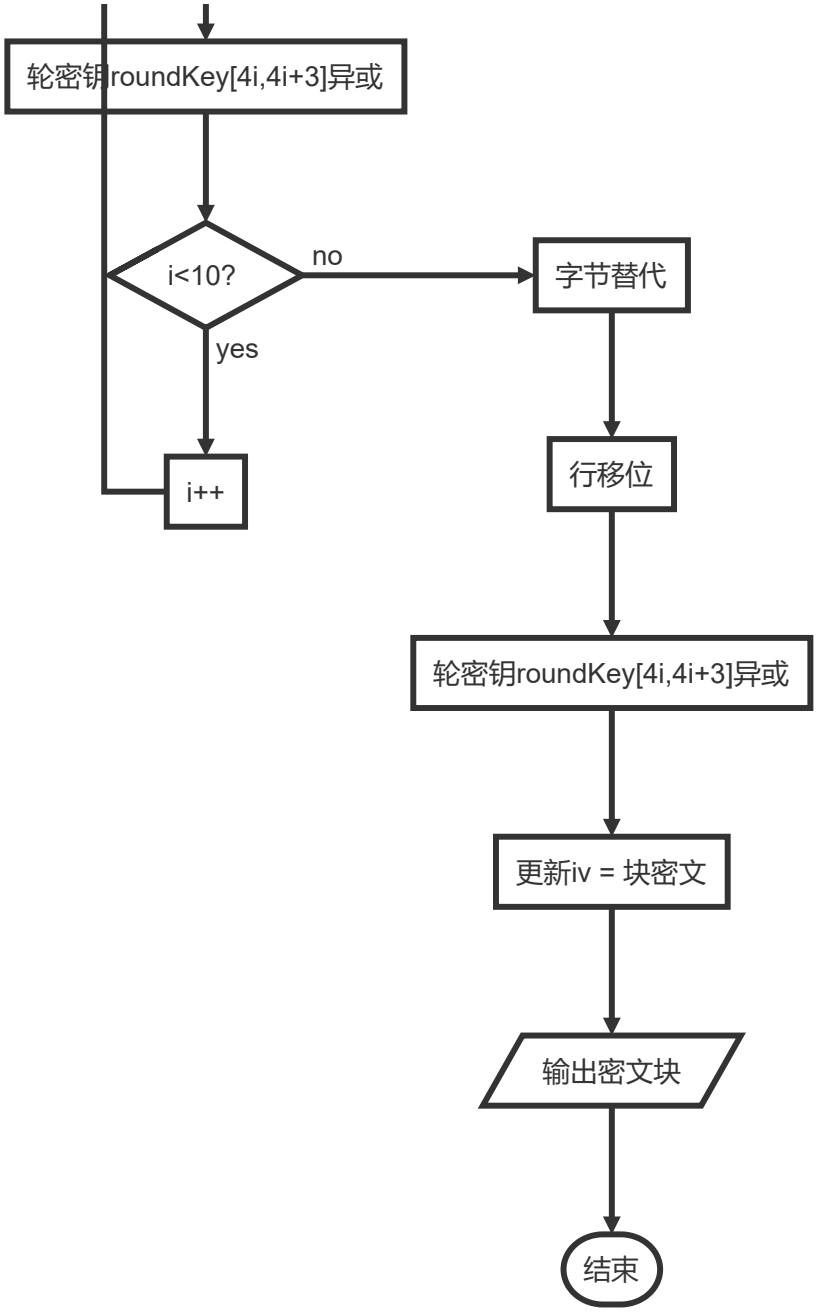
2. **i≠4k**

$$roundKey[i] = roundKey[i-4] \bigoplus roundKey[i-1]$$


rk[4i-4]
rk[4i-3]
rk[4i-2]
rk[4i-1]
f
rk
rk[4i]
rk[4i+1]
rk[4i+2]
rk[4i+3]

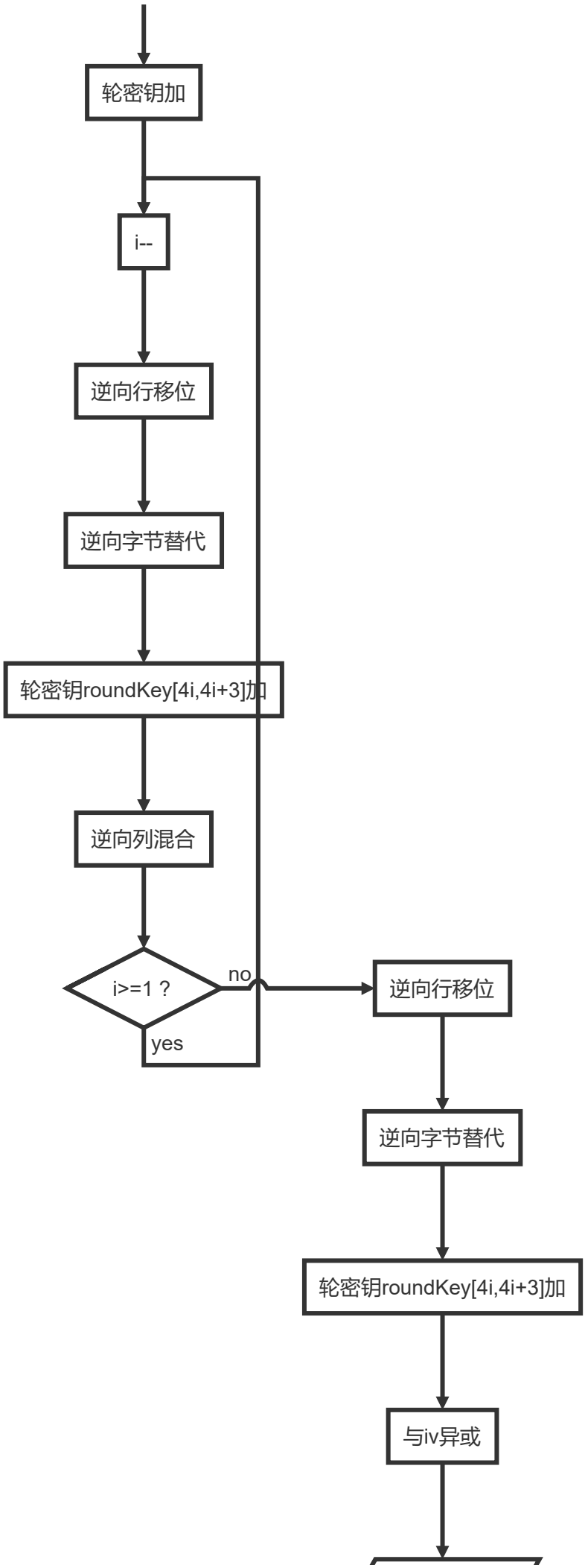
加密：

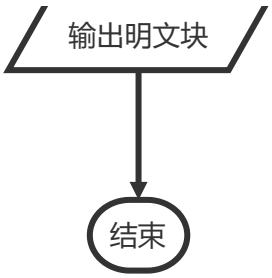

开始
分块加密
与偏移量iv异或
i=1
第i轮
字节替代
行移位
列混合


轮密钥roundKey[4i,4i+3]异或
i<10?
no
yes
字节替代
i++
行移位
轮密钥roundKey[4i,4i+3]异或
更新iv = 块密文
输出密文块
结束

解密:


开始
分块解密
i=10


轮密钥加
i--
逆向行移位
逆向字节替代
轮密钥roundKey[4i,4i+3]加
逆向列混合
i>=1 ?
no
yes
逆向行移位
逆向字节替代
轮密钥roundKey[4i,4i+3]加
与iv异或



5、代码实现

5.1 main.cpp

在main函数中，输入需要加密的明文，然后就会创建一个class AES，调用AES对象的加密函数，就可以得到加密之后的密文，其中，加密函数需要输入明文以及明文的长度。之后会解密验证，如果能正确解密，则测试成功。

解密函数需要输入密文。

```
int main(){
                cout<<"\n\n\n AES密码体制\n\n 密钥：mengyayuan\n\n 偏移量：123"
                <<"\n\n 密文编码：十六进制 \n\n 补码方式：PKCS5Padding\n\n\n"
                <<"--------------------------------\n\n"
                <<" 请输入您要加密的密文：";
                string sMEG;  cin>>sMEG ;   //输入明文
                int len=sMEG.length() ;     //明文长度

                const char *cMEG ={0} ;
                cMEG = sMEG.data() ;

                int iMEG[len+1];
                for(int i=0;i<len;i++)  iMEG[i]=cMEG[i];   //转成int数组

                myyAES data(iMEG,len) ;       //创建一个AES对象
                int text[1024]={-1};
                len = data.encrypt(text);       //AES加密
                cout<<"\n 密文：";
                for(int i=0;i<len;i++)  printf("%02x",text[i]);

                cout<<"\n\n\n 解密：";
                data.decryption(text, len);     //AES解密
                for(int i=0;text[i]!='\0';i++)  printf("%c",text[i]);
                cout<<"\n\n\n"<<endl;

                return 0;

}
```

5.2 myyAES.h

```
#ifndef MYYAES_H
```

```
#define MYYAES_H

class myyAES{
    private:
        int MEG[1024] ;
        int len ;
        int megBlocks[512][4][4];
        int blockCnt ;
        int roundKey[44][4];
        int round;
        static int iv[4][4] ;
        static int MixMatrix[4][4] ;
        static int paiS[16][16] ;
        const static int rcon[10][4];

        int lenInv ;
        int cipherBlocks[512][4][4] ;
        int blockCntInv ;
        int ivInv[4][4] ;
        static int paiSInv[16][16] ;
        static int MixMatrixInv[4][4] ;

    public:
        void print(int round);
        myyAES(int meg[1024], int len);
        ~myyAES() ;
        void group(int len) ;
        void creatRoundKey() ;
        int _xor(int, int);
        void f(const int*, int*, int);
        void addRoundKey(int number, int round);
        void subBytes(int number) ;
        void shiftRows(int number) ;
        void mixColumns(int number) ;
        int encrypt(int *text) ;

        void decryption(int *text, int);
        void groupInv(int *text) ;
        void addRoundKeyInv(int, int) ;
        void shiftRowsInv(int) ;
        void subBytesInv(int) ;
        void mixColumnsInv(int) ;
        void printInv(int round) ;
};

#endif
```

5.3 myyAES.CPP

构造函数：$myyAES()$

输入参数：$int* meg$：需要加密的明文对应的ASCII码

$int\ length$：明文字符串的长度

```
myyAES::myyAES(int *meg, int length){ //初始化各个变量
```

```
    len = length ;

    for(int i=0;i<len;i++)  MEG[i]=meg[i];

    blockCnt = 0 ;

    memset(roundKey,0,176);
    memset(ivInv,0,16) ;
    round = 0;

    group(len);  // 分组

    creatRoundKey() ;

    cout<<"轮密钥"<<endl;
    for(int i=0;i<44;i++){
        for(int j=0;j<4;j++){
            printf("%x ", roundKey[i][j]);
        }
        cout<<endl;
        if((i+1)%4==0)  cout<<endl;
    }


    lenInv = 0 ;
}
```

group函数

此函数主要用于对明文串进行填充和分组。由于我们使用的补码形式是PKCS5Padding，意思就是16个字节为一组，如果最后一组不满16个字节，则缺多少字节就补多少个字节，比如缺少5个字节，就在后面补上5个十进制的数字5；而如果最后一组恰好满16个字节，就补上16个$0x10$作为最后一组。

分组的时候，我们应该是竖着填入4×4的矩阵。

```
void myyAES::group(int len){
    // 填充
    int remain=16;
    if(len%16==0){  //正好够16个，就补充16个16
        for(int i=0;i<16;i++)
            MEG[i+len] = 0x10 ;
    }
    else{  // 最后一块不够16个，补充16个remain
        int had = len%16 ;
        if(had<0)
            had +=16 ;

        remain = 16-had ;
        for(int i=0;i<remain;i++)
            MEG[i+len] = remain ;
    }

    len+=remain;
    blockCnt = len/16 ;
    //竖着放
```

```
    int  cnt=0;
    for(int i=0;i<blockCnt;i++){
        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                    megBlocks[i][k][j] = MEG[cnt++] ;
            }
        }
    }
}
```

creatRoundKey()函数

这个函数主要用于拓展密钥，与之后的加密解密的$Block$异或。

首先是初始化一个$Block$密钥，由于$mengyayuan$不满16个，则需要补充6个数字0，并且，是竖着初始化。

对于接下来要拓展的密钥$roundKey[i]$，如果i不是4的倍数，就通过$roundKey[i-1] \bigoplus roundKey[i-4]$得到。否则，$roundKey[i-1]$就需要变换：循环左移，SBox字节代换，与轮常量$Rcon[i/4-1]$异或。之后再与$roundKey[i-4]$异或得到。

```
void myyAES::creatRoundKey(){
    //初始密钥mengyayuan
    roundKey[0][0]=0x6d;
    roundKey[0][1]=0x65;
    roundKey[0][2]=0x6e;
    roundKey[0][3]=0x67;
    roundKey[1][0]=0x79;
    roundKey[1][1]=0x61;
    roundKey[1][2]=0x79;
    roundKey[1][3]=0x75;
    roundKey[2][0]=0x61;
    roundKey[2][1]=0x6E;
    roundKey[2][2]=roundKey[2][3]=roundKey[3][0]=
    roundKey[3][1]=roundKey[3][2]=roundKey[3][3]=0x0;



    for(int i=4;i<44;i++){

        if(i%4!=0){
            for(int j=0;j<4;j++){  //异或
                roundKey[i][j] = roundKey[i-4][j] ^ roundKey[i-1][j] ;
            }
        }

        else{
            //k[i] = k[i-4] xor tk[i-1]
            //字循环  字节代换  轮常量异或
            int a[4]={0};
            f(roundKey[i-1], a, i) ;
            for(int j=0;j<4;j++){
                roundKey[i][j] = roundKey[i-4][j]^a[j] ;
            }
```

```
        }

    }


}
```

```
void myyAES::f(const int rk[4], int *a, int number){
    //字循环  字节代换  轮常量异或
    //字循环  一个字中的四个字节循环左移一个字节
    for(int j=0;j<4;j++){
        a[j] = rk[(j+1)%4];
    }

    //字节代换
    for(int j=0;j<4;j++){
        int shi = a[j]/16 ;
        int ge = a[j]%16 ;
        a[j] = paiS[shi][ge] ;
    }

    //轮常量异或  用于第j轮的密钥就和con[j]异或  j=number/4
    for(int j=0;j<4;j++){  //有四个字节  一个字节8位
        int temp = a[j] ;
        a[j] = temp ^ rcon[number/4-1][j] ;
    }

}
```

```
const int myyAES::rcon[10][4]={  0x01, 0x00, 0x00, 0x00,
                                 0x02, 0x00, 0x00, 0x00,
                                 0x04, 0x00, 0x00, 0x00,
                                 0x08, 0x00, 0x00, 0x00,
                                 0x10, 0x00, 0x00, 0x00,
                                 0x20, 0x00, 0x00, 0x00,
                                 0x40, 0x00, 0x00, 0x00,
                                 0x80, 0x00, 0x00, 0x00,
                                 0x1b, 0x00, 0x00, 0x00,
                                 0x36, 0x00, 0x00, 0x00
} ;
```

析构函数:

```
myyAES::~myyAES(){
        delete [] MEG;
        delete []megBlocks;
        blockCnt = 0 ;
        delete []roundKey ;
        round = 0 ;
        delete []iv ;
        delete []MixMatrix ;
        delete []paiS ;
        delete []rcon ;
```

```
        lenInv = 0;
        delete []cipherBlocks ;
        blockCntInv = 0 ;
        delete []ivInv;
        delete []paiSInv ;
        delete []MixMatrixInv ;
}
```

加密

加密流程同以上的流程图。

函数名：$encryption$

参数：$int* text$：需要加密的明文对应的对应的ASCII码数组

返回值：加密后的密文长度

值得注意的是，每一块$Block$在加密之前，都需要与偏移量iv异或，而只有第一块$Block$的偏移量是初始偏移量

$$iv_0 = \begin{bmatrix} 31 & 0 & 0 \\ 32 & 0 & 0 \\ 33 & 0 & 0 \end{bmatrix}$$

而接下来每一块的偏移量，则是前一块加密之后的密文。

此外，在最后一轮，也就是第十轮，只有字节代换，行移位，轮密钥加，**没有列混合。**

```
int myyAES::encrypt(int *text){
    for(int i=0;i<blockCnt;i++){

        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                megBlocks[i][k][j]^=iv[k][j];
            }
        }

        round=0;
        addRoundKey(i,round);
        round++;


        while(round<10){
    //      cout<<"in 字节代换"<<endl;
            subBytes(i);

    //      cout<<"in 行移位"<<endl;
            shiftRows(i) ;

    //      cout<<"in 列混合"<<endl;
            mixColumns(i);

    //      cout<<"in 轮密钥加"<<endl;
            addRoundKey(i,round);
```

```
            round++;
        }


    //  cout<<"最后一轮"<<endl;
    //  cout<<"in 字节代换"<<endl;
        subBytes(i);

    //  cout<<"in 行移位"<<endl;
        shiftRows(i) ;

    //  cout<<"in 轮密钥加"<<endl;
        addRoundKey(i,round);


        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                iv[k][j] = megBlocks[i][k][j] ;
            }
        }

    }

//  cout<<"密文十六进制"<<endl;
    int n=0;
    for(int i=0;i<blockCnt;i++){
        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                text[n++] = megBlocks[i][k][j] ;
            }
        }
    }
    return n;
}
```

解密

由于解密是加密的逆过程，从最后一块密文块开始，也从最后一块轮密钥开始异或，经过逆向行移位、逆向字节替代、逆向轮密钥加和逆向列混合之后的9轮，第十轮也是没有逆向列混合，经过十轮之后，并没有得到明文，由于是CBC模式，还需要属于上一块明文块进行异或，之后得到真正的明文块。

而第一块明文块的偏移量，则也是iv_0。

```
void myyAES::decryption(int *text, int l){
    lenInv = l;
    groupInv(text);
    memset(text,0,1024) ;

    blockCntInv = lenInv/16 ;

    for(int i=blockCntInv-1;i>=0;i--){
        round = 10;
        addRoundKeyInv(i, round) ;
        round-- ;
```

```
        while(round>=1){
    //      cout<<"逆向行移位"<<endl;
            shiftRowsInv(i) ;

    //      cout<<"逆向字节替代"<<endl;
            subBytesInv(i) ;

    //      cout<<"逆向轮密钥加"<<endl;
            addRoundKeyInv(i, round) ;

    //      cout<<"逆向列混合"<<endl;
            mixColumnsInv(i) ;
            round-- ;
        }

        //最后一轮
            shiftRowsInv(i) ;
            subBytesInv(i) ;
            addRoundKeyInv(i, round) ;

        // CBC
        if(i==0){
            ivInv[0][0] = 0x31;
            ivInv[1][0] = 0x32;
            ivInv[2][0] = 0x33;
            ivInv[3][0] = ivInv[0][1] = ivInv[0][2] = ivInv[0][3] =
            ivInv[1][1] = ivInv[1][2] = ivInv[1][3] = ivInv[2][1] =
            ivInv[2][2] = ivInv[2][3] = ivInv[3][1] = ivInv[3][2] = ivInv[3][3]
= 0x00 ;
        }
        else{
            for(int j=0;j<4;j++){
                for(int k=0;k<4;k++){
                    ivInv[j][k] = cipherBlocks[i-1][j][k] ;
                }
            }
        }

        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                cipherBlocks[i][j][k] ^= ivInv[j][k] ;
            }
        }

    }

    int cnt=0;
    for(int i=0;i<blockCntInv;i++){
        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                text[cnt++] = cipherBlocks[i][k][j] ;
            }
        }
    }

    int padding = text[--cnt];
    // 去填充
```

```
    for(int i=0;i<padding;i++){
        if(text[lenInv-i-1]!=padding) {
            cout<<"解密失败! i="<<i<<endl;
            return ;
        }
        text[lenInv-i-1] = '\0' ;
    }
}
```

此外，由于在加密时，我们使用了PKCS5Padding填充，所以对于解密出来的明文，我们还需要去填充。我们可以先查看明文数组的最后一个数字x，就可以知道填充了x个数字。接着从最后开始，去掉x个数字即可。

不过如果在去填充的时候，如果存在要去掉的数字并不等于x时，就说明了解密出错了。

groupInv函数

只需要分组，同样也是按列分组。

```
void myyAES::groupInv(int *text){
    blockCntInv = lenInv/16 ;
    int t=0;

    for(int i=0;i<blockCntInv;i++){
        for(int j=0;j<4;j++){
            for(int k=0;k<4;k++){
                cipherBlocks[i][k][j] = text[t++] ;
            }
        }
    }

}
```

轮密钥加addRoundKey

对于每块明文/密文4×4块，都采用字节与字节异或的方式，与轮密钥异或。

加密过程：

```
void myyAES::addRoundKey(int number, int round){
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            int t = megBlocks[number][j][i] ;
            megBlocks[number][j][i] = t ^ roundKey[4*round+i][j] ;
        }
    }
}
```

解密过程：

```
void myyAES::addRoundKeyInv(int number, int round){
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            int t = cipherBlocks[number][j][i] ;
            cipherBlocks[number][j][i] = t ^ roundKey[4*round+i][j] ;
        }
    }
}
```

字节代换subBytes

在域$GF(2^8)$中，可以计算出每一个多项式对应的代换，把所有的列成表，就可以对每个十六进制数直接查表不用计算，提高效率。

```
void myyAES::subBytes(int number) {
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            int a = megBlocks[number][i][j]/16 ;
            int b = megBlocks[number][i][j]%16 ;
            int t =  paiS[a][b] ;
            megBlocks[number][i][j] = t ;
        }
    }

}
```

由于存的是十进制数，可以通过除16、模16得到对应的十六进制数，然后查表。

```
void myyAES::subBytesInv(int number){

    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            int t = cipherBlocks[number][i][j] ;
            int a = t/16 ;
            int b = t%16;
            cipherBlocks[number][i][j] = paiSInv[a][b] ;
        }
    }

}
```

行移位shiftRows

加密时，第i循环左移i位（$i=0,1,2,3$）。

```
void myyAES::shiftRows(int number){
    for(int i=1;i<4;i++){
        int temp[4] ;
        for(int j=0;j<4;j++)  temp[j] = megBlocks[number][i][j] ;
        for(int k=0;k<4;k++){
            megBlocks[number][i][k] = temp[(k+i)%4] ;
        }
    }

}
```

解密时，第i循环右移i位（$i = 0, 1, 2, 3$）。

```
void myyAES::shiftRowsInv(int number){
    int temp[4] ;

    for(int i=1;i<4;i++){
        for(int j=0;j<4;j++)   temp[j] = cipherBlocks[number][i][j] ;
        for(int k=0;k<4;k++){
            int pos = (k-i)%4;
            if(pos<0)  pos+=4 ;
            cipherBlocks[number][i][k] = temp[pos] ;
        }
    }

}
```

列混合mixColumns

参数：$int\ number$：需要进行操作的$Block$的序号。

加密：混合矩阵$MixMatrix$与加密的明文块$Block$相乘。其中，乘法和加法都是在域$GF(2^8)$中进行的。

```
void myyAES::mixColumns(int number) { //state, mixmatrix
    //一行一行的算
    int bloc[4][4];
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++)
            bloc[i][j] = megBlocks[number][i][j] ;
    }

    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            megBlocks[number][i][j] = 0 ;
            for(int k=0;k<4;k++){
                bitset<2*Max> a(MixMatrix[i][k]), b(bloc[k][j]) ;
                bitset<2*Max> t = multiple(a,b) ;
                megBlocks[number][i][j] ^= t.to_ulong()  ;
            }
        }
    }
```

```
}
```

先把两个要相乘的十进制数转换成二进制，变成域中的元素。

两个数a，b相乘就是从低位开始，对于b中的每一位，如果第i位是1，a就左移i位，再和原来的a异或。得到的多项式再模本原多项式$p(x)=x^8+x^4+x^3+x+1$，再转成十进制，就可以保证该数是8位，即一个字节。

```
int degree(const bitset<2*Max> a){
    for(int i=2*Max-1;i>=0;i--){
        if(a[i]==1)  return i ;
    }
    return 0 ;
}


bitset<2*Max> mod(bitset<2*Max> a){
    bitset<Max+1> px;
    bitset<Max> rx;
    px[8] = px[4] = px[3] = px[1] = px[0] = 1;

    rx[4] = rx[3] = rx[1] = rx[0] = 1;

    bitset<2*Max> _r = bitset<2*Max>(rx.to_string()) ;
    bitset<2*Max> ans = bitset<2*Max>(a.to_string().substr(Max)); // 原来的前面个
不要

    while(degree(a)>Max-1){
        ans.reset();
        ans = bitset<2*Max>(a.to_string().substr(Max));
        for(int i=Max;i<2*Max;i++){
            if(a[i])  ans^=_r<<(i-Max);
        }
        a=ans;
    }

    return ans;
}

bitset<2*Max> multiple(bitset<2*Max> a, bitset<2*Max> b){
    bitset<2*Max> ans ;
    bitset<2*Max> temp(a.to_string()) ;

    for(int i=0;i<2*Max;i++){
        if(b[i])  ans^=temp<<i ;

    }

    return mod(ans) ;
}
```

解密：即逆混合矩阵与密文块相乘，算法同上，只是矩阵变了。

```
void myyAES::mixColumnsInv(int number){
    int bloc[4][4];
```

```
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++)
            bloc[i][j] = cipherBlocks[number][i][j] ;
    }

    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            cipherBlocks[number][i][j] = 0 ;
            for(int k=0;k<4;k++){
                bitset<2*Max> a(MixMatrixInv[i][k]), b(bloc[k][j]) ;
                bitset<2*Max> t = multiple(a,b) ;
                cipherBlocks[number][i][j] ^= t.to_ulong()  ;
            }
        }
    }

}
```

混合矩阵和逆混合矩阵相乘得单位矩阵。

```
int myyAES::MixMatrix[4][4]={0x02,0x03,0x01,0x01,
                             0x01,0x02,0x03,0x01,
                             0x01,0x01,0x02,0x03,
                             0x03,0x01,0x01,0x02
} ;

int myyAES::MixMatrixInv[4][4]={0x0e, 0x0b, 0x0d, 0x09,
                                0x09, 0x0e, 0x0b, 0x0d,
                                0x0d, 0x09, 0x0e, 0x0b,
                                0x0b, 0x0d, 0x09, 0x0e} ;
```

6、实验结果

情况1：明文不满16个字节

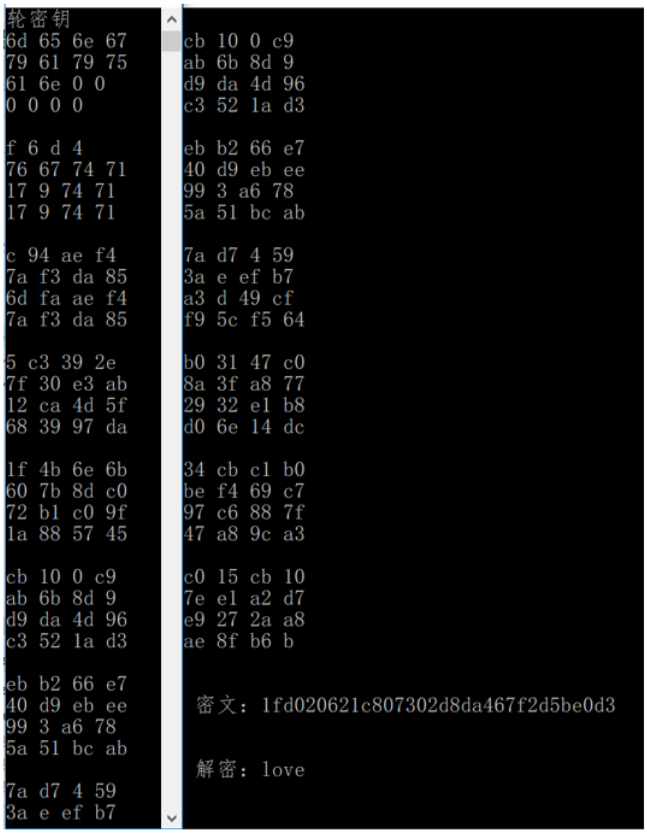
输入：love

```
AES密码体制
密钥：mengyayuan
偏移量：123
密文编码：十六进制
补码方式：PKCS5Padding

--------------------------------

请输入您要加密的密文：love
```

输出轮密钥、加密后的密文，以及通过该密文加密出来的明文：



明文	love
密文	1fd020621c807302d8da467f2d5be0d3

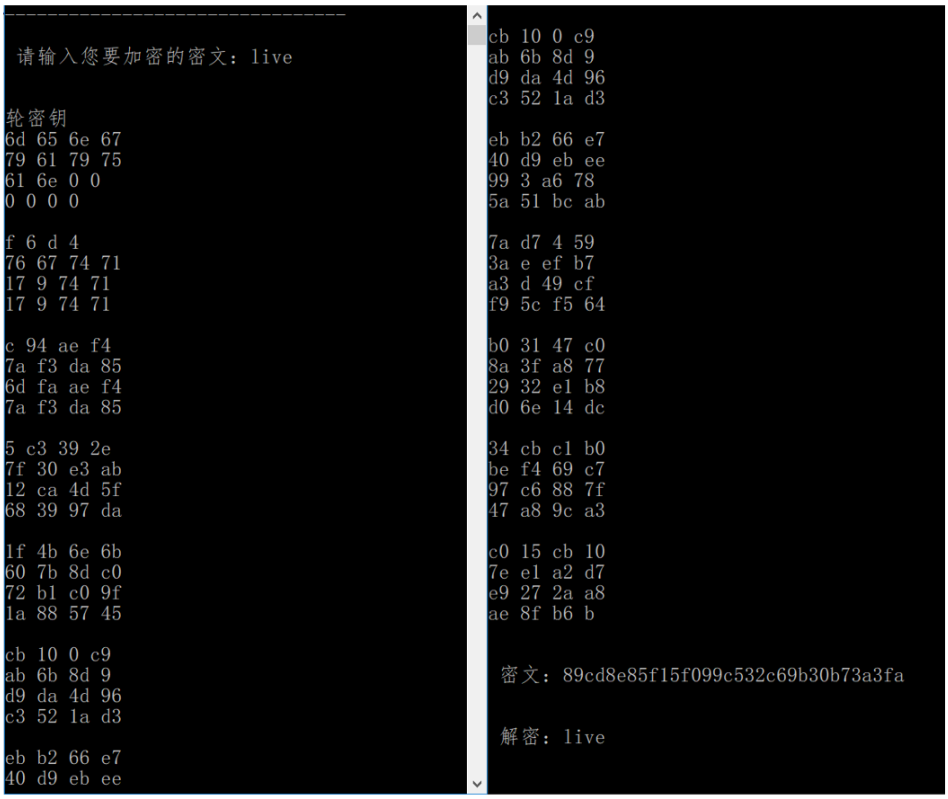
网站验证：1fd020621c807302d8da467f2d5be0d3

明文串长度为4，填充之后长度为16，每个ASCII码都输出十六进制，两个字节，所以密文一共有32个字节。

经验证，加密成功！

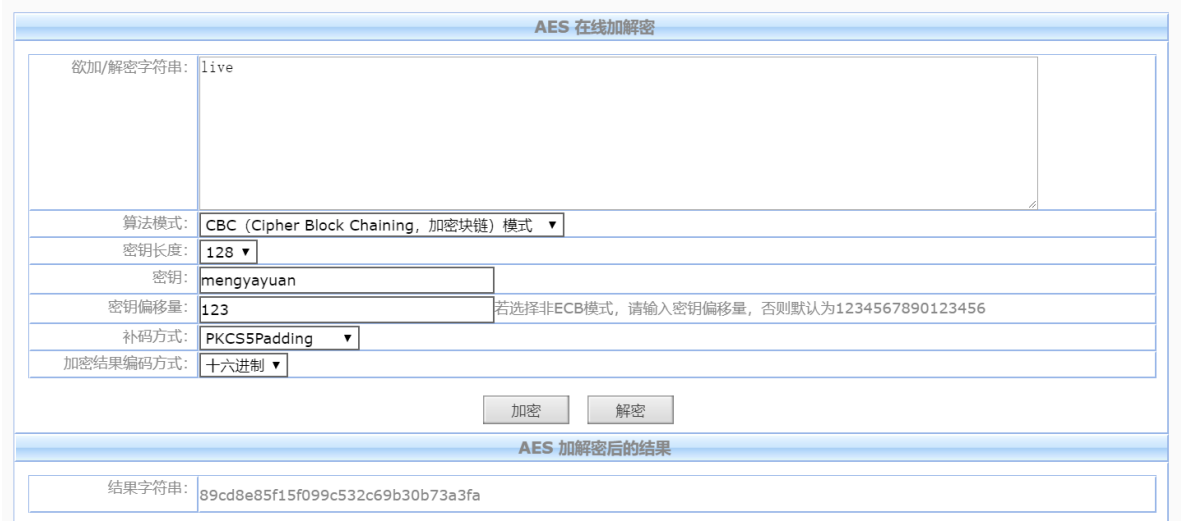
情况2：加密字符相似

输入：live



明文	live
密文	89cd8e85f15f099c532c69b30b73a3fa

网站验证：89cd8e85f15f099c532c69b30b73a3fa

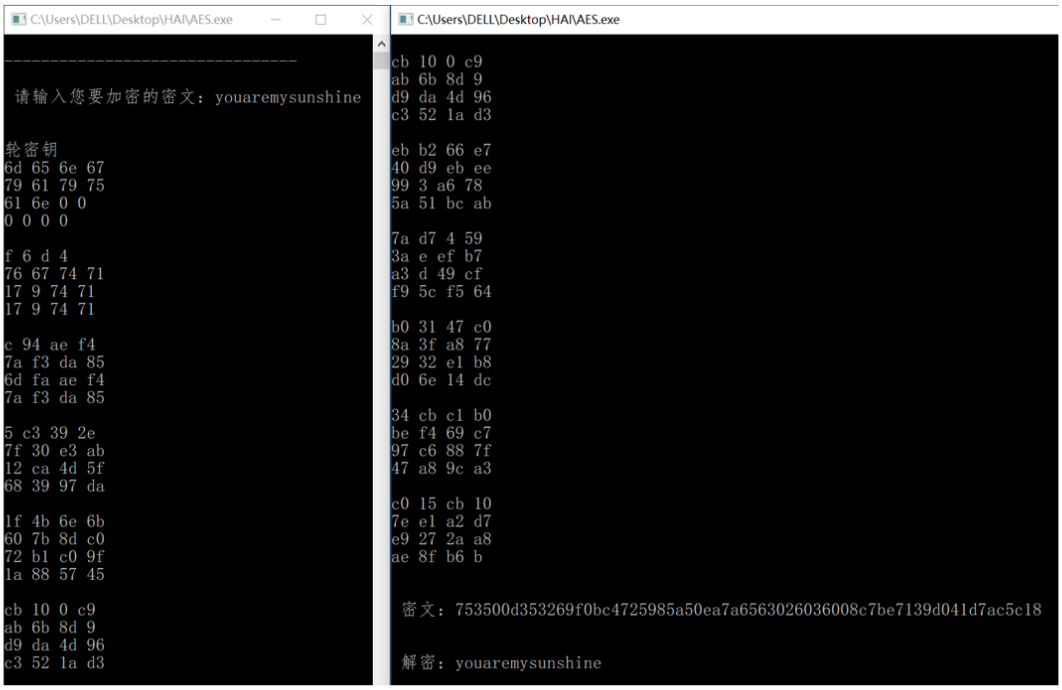


密文长度同样是32。

经验证，加密成功！

情况3：加密的字符刚刚好16个字节

输入：youaremysunshine



明文	youaremysunshine
密文	753500d353269f0bc4725985a50ea7a6563026036008c7be7139d041d7ac5c18

网站验证：753500d353269f0bc4725985a50ea7a6563026036008c7be7139d041d7ac5c18

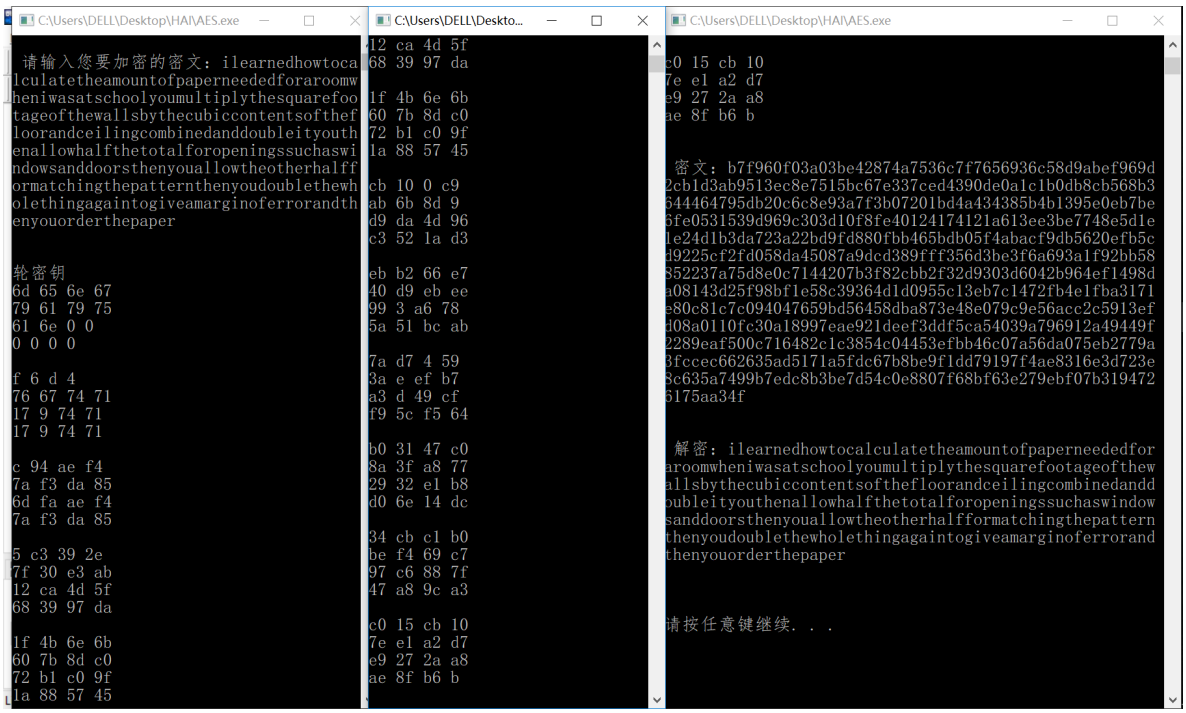
明文串长度为16，填充后长度为32，输出十六进制之后，长度为64，正确。

经验证，加密成功！

情况4：实验要求测试

输入：

ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschoolyoumultiplythesquarefootageofthewallsbythecubiccontentsofthefloorandceilingcombinedanddoubleityouthenallowhalfthetotalforopeningssuchaswindowsanddoorsthenyouallowtheotherhalfformatchingthepatternthenyoudoublethewholethingagaintogiveamarginoferrorandthenyouorderthepaper



明文：

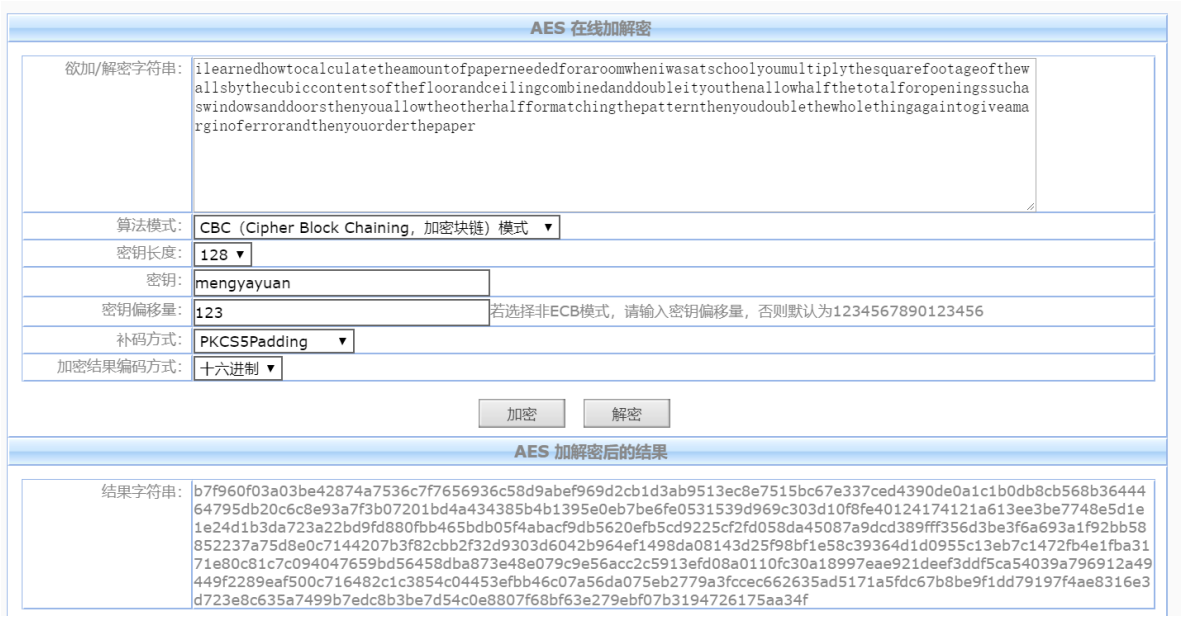
ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschoolyoumultiplythesquarefootageofthewallsbythecubiccontentsofthefloorandceilingcombinedanddoubleityouthenallowhalfthetotalforopeningssuchaswindowsanddoorsthenyouallowtheotherhalfformatchingthepatternthenyoudoublethewholethingagaintogiveamarginoferrorandthenyouorderthepaper

密文：

b7f960f03a03be42874a7536c7f7656936c58d9abef969d2cb1d3ab9513ec8e7515bc67e337ced4390de0a1c1b0db8cb568b3644464795db20c6c8e93a7f3b07201bd4a434385b4b1395e0eb7be6fe0531539d969c303d10f8fe40124174121a613ee3be7748e5d1e1e24d1b3da723a22bd9fd880fbb465bdb05f4abacf9db5620efb5cd9225cf2fd058da45087a9dcd389fff356d3be3f6a693a1f92bb58852237a75d8e0c7144207b3f82cbb2f32d9303d6042b964ef1498da08143d25f98bf1e58c39364d1d0955c13eb7c1472fb4e1fba3171e80c81c7c094047659bd56458dba873e48e079c9e56acc2c5913efd08a0110fc30a18997eae921deef3ddf5ca54039a796912a49449f2289eaf500c716482c1c3854c04453efbb46c07a56da075eb2779a3fccec662635ad5171a5fdc67b8be9f1dd79197f4ae8316e3d723e8c635a7499b7edc8b3be7d54c0e8807f68bf63e279ebf07b3194726175aa34f

网站加密：

b7f960f03a03be42874a7536c7f7656936c58d9abef969d2cb1d3ab9513ec8e7515bc67e337ced4390de0a1c1b0db8cb568b3644464795db20c6c8e93a7f3b07201bd4a434385b4b1395e0eb7be6fe0531539d969c303d10f8fe40124174121a613ee3be7748e5d1e1e24d1b3da723a22bd9fd880fbb465bdb05f4abacf9db5620efb5cd9225cf2fd058da45087a9dcd389fff356d3be3f6a693a1f92bb58852237a75d8e0c7144207b3f82cbb2f32d9303d6042b964ef1498da08143d25f98bf1e58c39364d1d0955c13eb7c1472fb4e1fba3171e80c81c7c094047659bd56458dba873e48e079c9e56acc2c5913efd08a0110fc30a18997eae921deef3ddf5ca54039a796912a49449f2289eaf500c716482c1c3854c04453efbb46c07a56da075eb2779a3fccec662635ad5171a5fdc67b8be9f1dd79197f4ae8316e3d723e8c635a7499b7edc8b3be7d54c0e8807f68bf63e279ebf07b3194726175aa34f



经验证，加密成功！

7、实验总结

- AES加密能把一段简短的明文加密成比较长的、看似毫无规律的乱码，达到了混乱与扩散的效果。通过轮密钥加、字节代换、行移位、列混合，使得密钥、明文和密文之间的依赖关系十分复杂，并且具有雪峰效应，很厉害。不过，根据密文长度可以推导出明文长度，我觉得可能会对密文识别造成一定的影响。
- 在列混合运算中，我是直接把上次有限域的实现乘法和模部分直接粘贴过来的，于是我发现上次的实现的有限域的模好像有点错误（捂脸）。我百度学习的时候发现，网上很多大佬都是利用了分配律来计算，虽然没看懂，也没有仔细研读，但是我总觉得那样的方法虽然简单，但是不是很耐用？所以还是用了已有的有限域的方法来计算。
- 这次的代码虽然长，但是很多都是可以举一反三地，写出了加密，解密自然就很好写。但是我写的时候真的是太坎坷了，先是纠结填充时应该填充'0'字符的ASCII码还是直接是一个数字0；然后就是CBC模式没有理解好，我原来以为的是在每一轮，第一块明文块与初始偏移量iv_o异或，然后得出的密文作为下一块的偏移量，也就是每一轮都在与偏移量异或，所以加密出来的一直都是错的，甚至debug了一个星期，还问了ta姐姐……最后问了一个大佬才知道，原来是第一块与初始偏移量异或之后，经过十轮的加密得到的密文才作为下一块的偏移量，不是一轮……这件事告诉我下次要好好了解实验中的每一个要求，不要自己为就是那样子，而不去验证考究……
- 最后一点就是，下次实验要早点打，不然谁知道debug要多久……最后感谢ta姐姐，辛苦你们啦~