哈尔滨工业大学（深圳）

通信系统仿真

实验报告

题　　目　AES 算法的 MATLAB 实现

专　　业　　通信工程

学　　号　　12306

姓　　名　　程二狗

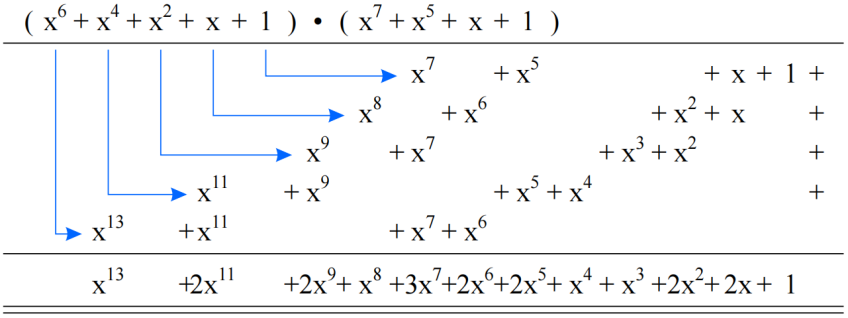
日　　期　　2021.10.24

一、实验原理

2.1 有限域算法

2.1.1 多项式乘法

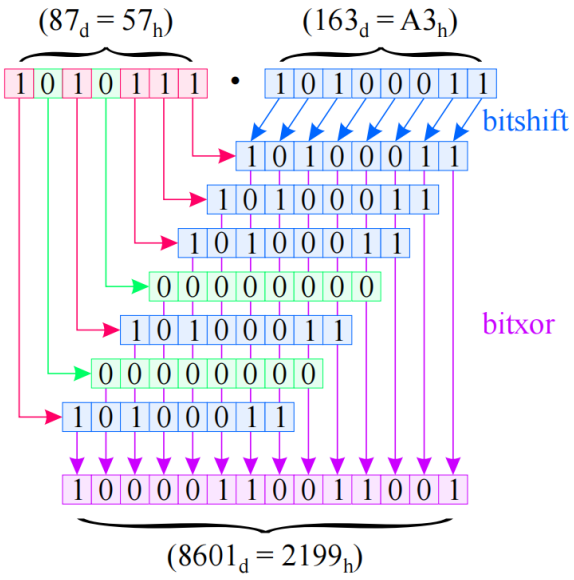
下图中两式相乘得到结果 $x^{13}+2x^{11}+2x^{9}+x^{8}+3x^{7}+2x^{6}+2x^{5}+x^{4}+x^{3}+2x^{2}+2x+1$ 。



由于是基于 $GF(2)$ 的运算，故我们需要对省略每个具有偶数系数的幂，并将每个奇数系数减少为 1，从而得到下列多项式：

$$x^{13}+x^{8}+x^{7}+x^{4}+x^{3}+1$$

用位级操作也可以得到同样的结果：



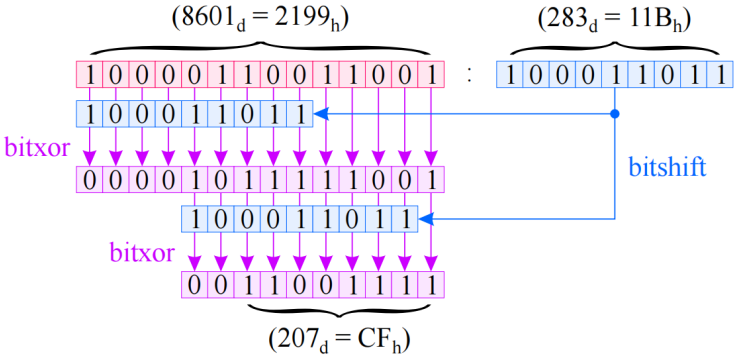
2.1.2 多项式除法

以下用具体例子说明：分子的最大幂 x^{13} 除以分母的最大幂 x^8，从而得到第一个结果项 x^5。该项乘以分母得到 $x^{13}+x^8+x^7+x^4+x^3+1$，从分子中减去该项，得到一个新的分子 $-x^9+x^7-x^6-x^5+x^4+x^3+1$。

$$
\begin{array}{l}
(x^{13}+x^8+x^7+x^4+x^3+1) : (x^8+x^4+x^3+x+1) = x^5 - x \\
\underline{-(x^{13}+x^9+x^8+x^6+x^5)} \\
-x^9+x^7-x^6-x^5+x^4+x^3+1 \\
\underline{-(-x^9-x^5-x^4-x^2-x)} \\
x^7-x^6+2x^4+x^3+x^2+x+1
\end{array}
$$

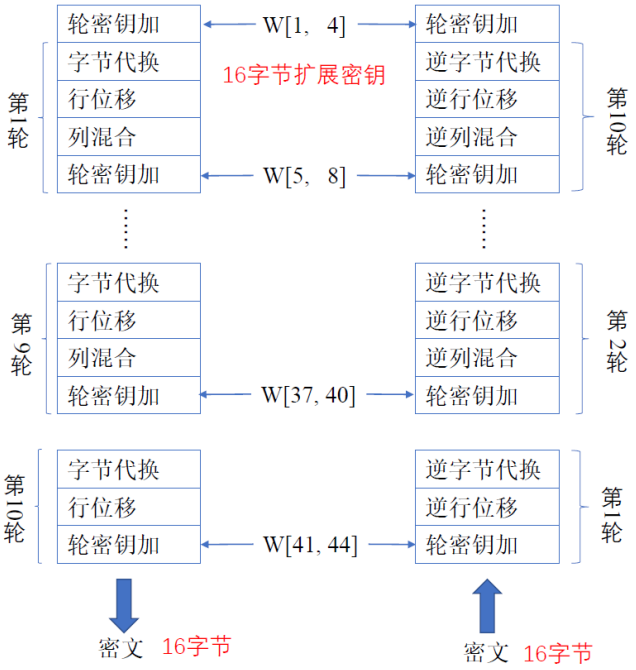
重复此过程，直到新分子的最大幂已经小于分母的最大幂。最终的分子 $x^7-x^6+2x^4+x^3+x^2+x+1$ 记为余式。运用 GF(2) 的运算法则（即异或规则，偶数系数→0，奇数系数→1），得到最终的余式为 $x^7+x^6+x^3+x^2+x+1$。

用位级操作也可以得到同样的结果：分母向左偏移，直到其最高位(即左数第一位)与分子的最高位相匹配。然后通过异或执行减法，从而得到一个新的、更小的分子。重复移动和异或，直到生成的分子（余式）长度在一个字节以内。



2.2 AES 算法原理

2.2.1 AES 基本结构



实验流程框图

对明文进行轮密钥加，再进行十轮加密，输出密文；接着再对密文进行十轮解密，得到解密后的明文，对比两者是否相同。

2.2.2 字节替换

S 盒的生成

Step. 1 求每个在 $GF(2^8)$ 中的元素的逆

Step. 2 仿射变换

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

等效表达式：$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$（这里的<<<表示循环移位）

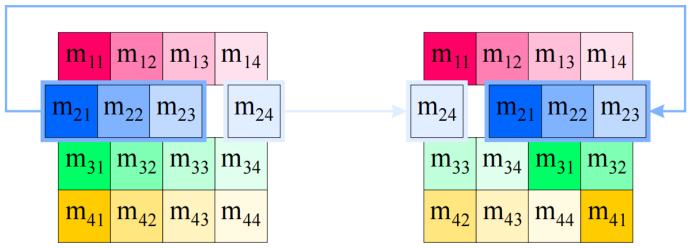
$s_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$

$s = \left(b \times 31_{10} \bmod 257_{10}\right) \oplus 99_{10}$【本实验中采取该种等效方式】

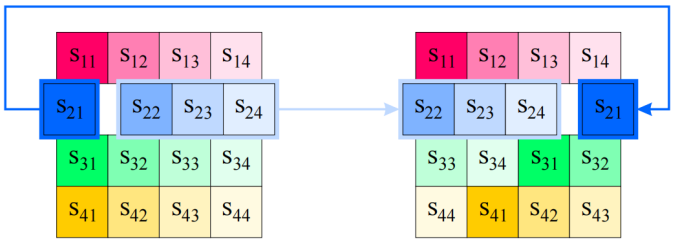
由于等效表达式的运算中包含取模的操作，故乘法等效为循环移位乘法。

2.2.3 行位移

行移位是一个简单的左循环移位操作。当密钥长度为 128 比特时，状态矩阵的第 i 行向左循环移位 i 字节，如下图所示：



逆向行移位是一个简单的右循环移位操作。当密钥长度为 128 比特时，状态矩阵的第 i 行向右循环移位 i 字节，如下图所示：



2.2.4 列混淆

列混淆变换通过矩阵相乘来实现，将行移位后的状态矩阵与固定的矩阵相乘，得到混淆后的状态矩阵，如下图的公式所示：

$$\begin{bmatrix} s'_{11} & s'_{12} & s'_{13} & s'_{14} \\ s'_{21} & s'_{22} & s'_{23} & s'_{24} \\ s'_{31} & s'_{32} & s'_{33} & s'_{34} \\ s'_{41} & s'_{42} & s'_{43} & s'_{44} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \bullet \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

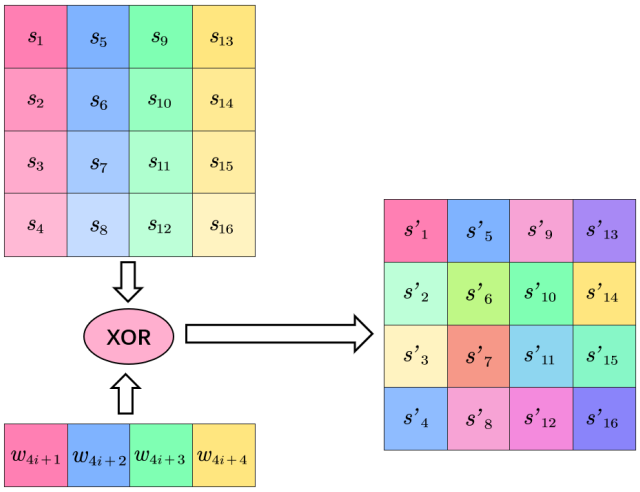
左侧的元素 s'_{ij} 由右侧第一个矩阵第 i 行的向量与的二个矩阵的第 j 列向量点乘得到。其中，矩阵元素的乘法和加法都是定义在基于 $GF(2^8)$ 上的二元运算，即需要对 $GF(2^8)$ 的不可约多项式（本原多项式）$m(x) = x^8 + x^4 + x^3 + x + 1$ 进行取模。

对正向列混淆变换矩阵求逆得到逆列混淆变换的变换矩阵，故可逆向列混淆变换由下图定义：

$$\begin{bmatrix} s'_{11} & s'_{12} & s'_{13} & s'_{14} \\ s'_{21} & s'_{22} & s'_{23} & s'_{24} \\ s'_{31} & s'_{32} & s'_{33} & s'_{34} \\ s'_{41} & s'_{42} & s'_{43} & s'_{44} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \bullet \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

2.2.5 轮密钥加

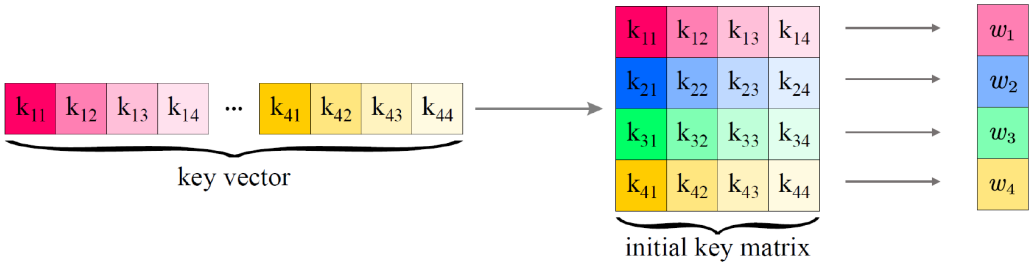
轮密钥加是将 128 位轮密钥 Ki 与状态矩阵中的数据进行逐位异或操作，如下图所示：



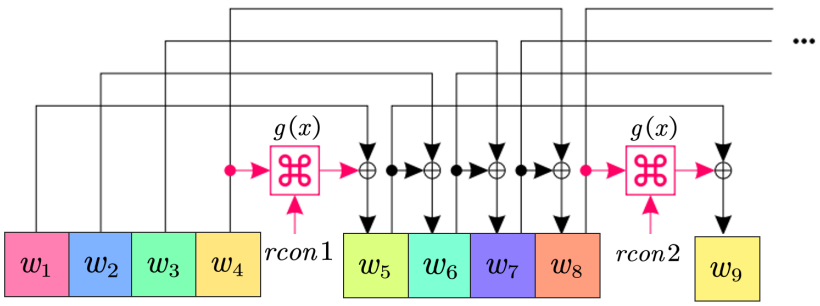
其中，密钥 Ki 中每个字 W[4i]，W[4i+1]，W[4i+2]，W[4i+3]为 32 位比特字，包含 4 个字节，他们的生成算法下面在下面介绍。轮密钥加过程可以看成是字节逐位异或的结果，也可以看成字节级别或者位级别的操作。也就是说，可以看成 S0，S1，S2，S3 组成的 32 位字与 W[4i]的异或运算。轮密钥加的逆运算同正向的轮密钥加运算完全一致，这是因为二进制异或的逆操作是其自身。轮密钥加操作很简单，但却能够影响 S 数组中的每一位。

密钥扩展过程

Step. 1 将初始密钥按照下图排列成矩阵形式，排列后用 4 个 32 比特的字表示，分别记为 $w[1]$， $w[2]$， $w[3]$， $w[4]$。

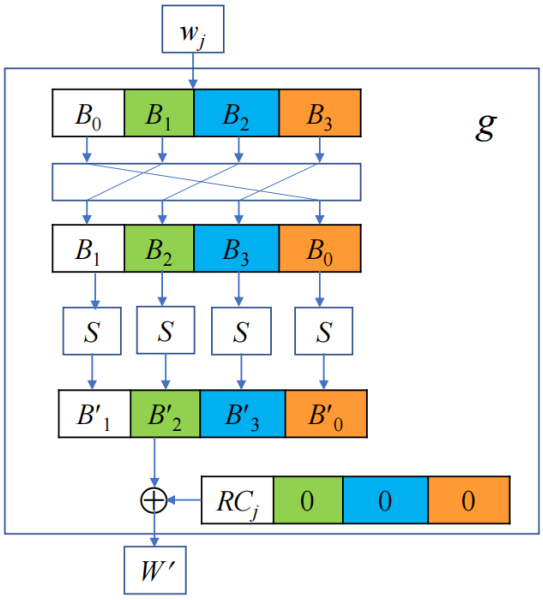


Step. 2 依次求解 w_j，其中 $j=5,6,...,44$。求解方法如下：若 $\mathrm{mod}(j,4)=1$，则 $w_j=w_{j-4}\oplus g(w_{j-1})$，否则 $w_j=w_{j-4}\oplus w_{j-1}$



函数 g 的运算过程

函数 g 的运算包括行循环移位、S 盒替换、圆常数异或三个步骤，细节如下图所示：

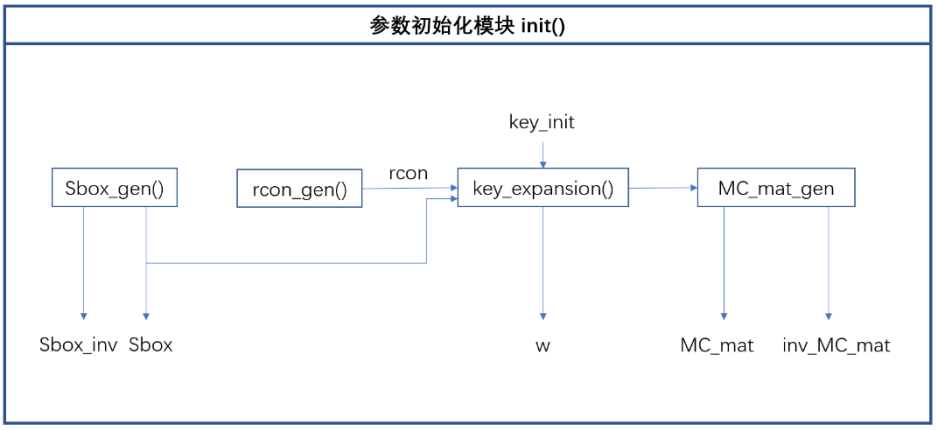


二、程序分析

以下分模块实现 AES 的加解密过程

3.1 参数初始化

程序框图如下所示：



主函数及其子函数如下所示：

```
function [Sbox,Sbox_inv,w,MC_mat,MC_mat_inv] = init()
% 初始化函数，产生 S 盒与逆 S 盒，圆常数 rcon，密钥 key，用于列混淆中的多项式矩阵
%S 盒与逆 S 盒的生成
[Sbox,Sbox_inv] = Sbox_gen;
%圆常数生成
```

```
rcon = rcon_gen;
%密钥的生成与扩展
key_init_hex = {'00' '01' '02' '03' '04' '05' '06' '07'...
                '08' '09' '0a' '0b' '0c' '0d' '0e' '0f'};
key_init = hex2dec(key_init_hex);
w = key_expansion(key_init,Sbox,rcon);
%用于列混淆的矩阵及其逆矩阵的生成
[MC_mat,MC_mat_inv] = MC_mat_gen;    % MC 为 MixColumns 的简写，表示用于列混淆的矩阵的生成
end
```

S 盒的生成

Step.1 寻找元素的逆

```
function inv = find_inverse(in,mod_pol)
% 函数功能：对于特定的余数 mod_pol，在 2^8 的范围内，找到输入参数 in 的逆，使之满足 mod(in * in^(-1) , mod_pol)=1。（溢出则取余）
    for i = 1:255
        prodcut = poly_mult(in,i,mod_pol);
        if prodcut == 1
            inv = i;
            break
        end
    end
end
```

Step.2 仿射变换

```
function out = aff_trans(in)
% 函数功能：实现仿射变换
% 参考资料：维基百科 https://en.wikipedia.org/wiki/Rijndael_S-box
    mod_pol = bin2dec('100000001');
    mult_pol = bin2dec('00011111');
    add_pol = bin2dec('01100011');
    temp = poly_mult(in,mult_pol,mod_pol);
    out = bitxor(temp,add_pol);
end
```

1. 逆 S 盒的生成

以 i 作为索引值查询得到 S 盒的结果值作为逆 S 盒的索引值；由该索引值查询逆 S 盒得到的结果值即为 S 盒的索引值。

结果为 i-1 是由于逆 S 盒的结果值在 [0,255] 之间。逆 S 盒的索引为 Sbox(i)+1 是因为 MATLAB 索引值必须为正数。

如果索引值 i=1，Sbox(i)=(63)16，则 Sbox_inv(Sbox(i)+1)= Sbox_inv(100) = (00)16

```
function Sbox_inv = Sbox_inversion(Sbox)
% 函数功能：生成 S 盒子的逆
    for i = 1:256
        Sbox_inv(Sbox(i)+1) = i-1;
    end
end
```

值得注意的是，在代码 Sbox_gen 中，生成的 S 盒与逆 S 盒都是 1×256 的行向量，而非矩阵，但同样可以实现索引功能，并能利用索引功能进行逆 S 盒的求解。举例来说，对于数 (a)10 = (bc)16。当我们通过十六进制得到第 b 行，第 c 列得到元素 d，我们利用十进制行向量形式的 S 盒的索引 a 同样能找到元素 d，这是由于两者满足 $b\times16+c=a$。所以在行向量形式 S 盒中的第 a 个元素，恰好就是 16×16 矩阵形式 S 盒中第 b 行，第 c 列元素。

2. 密钥扩展函数 key_expansion

```
function w = key_expansion(key_init,Sbox,rcon)
% 函数功能：实现对密钥的扩展，将原本 4×4 的密钥矩阵扩展为 44×4
% 输入参数：
%       key_init：初始的 32 位密钥
%       Sbox：S 盒
%       rcon：圆常数
% 输出参数：
%       w：扩展后的密钥
    w = (reshape(key_init,4,4))';          % 按行填入，第 i 行代表 wi
    for i = 5:44
        temp = w(i-1,:);                   % 提取上一行的密钥
        if mod(i,4)==1                     % 如果对 4 的余数为 1，则对 w_(i-1)进行 g
```

```
变换，得到 temp=g(w_(i-1))
            temp = round_shift(temp);    % 进行向左循环位移一字节
            temp = Sbox(temp+1);         % 进行 S 盒替换，由于 MATLAB 索引值要＞
0，故需要 temp+1。
            r = rcon(floor((i-1)/4),:); % 提取每一轮的圆常数
            temp = bitxor(temp,r);
        end
        w(i,:) = bitxor(w(i-4,:),temp)  % w_i = w_i-4⊕w_i-1 或者 w_i = w_i-
4⊕g(w_i-1)
    end
end
```

3. 圆常数 rcon 的生成

在扩展密钥中使用到了圆常数。它是一个 10×4 的零矩阵，除了第一列，其字节由 2 的幂次构成。

圆常数初始值为 1，在 Rijndael 有限域（即 $GF(2^8)$）中通过前一位乘 2 迭代计算得到圆常数 rcon 的每一位，最后在行中添加三个零列。第一列的计算式如下：

$$\mathrm{rcon}(i) = 2^{i-1} = 2 * 2^{i-2} = 2 * \mathrm{rcon}(i-1)$$

等价式：$\mathrm{rcon}(i) = b^{i-1} \bmod x^8 + x^4 + x^3 + x + 1$

```
function rcon = rcon_gen()
% 函数功能：生成圆常数 rcon
    mod_pol = bin2dec('100011011');
    rcon(1) = 1;
    for i = 2:10
        rcon(i) = poly_mult(rcon(i-1),2,mod_pol);%迭代算出下一位圆常数
    end
    rcon = [rcon(:),zeros(10,3)];%在后面补三列零行
end
```

4. 列混淆变换矩阵及逆列混淆变换矩阵的生成

```
function [MC_mat,MC_mat_inv] = MC_mat_gen()
% 函数功能：生成用于列混淆的多项式矩阵
```

```
% 逆列混淆矩阵的构造
row_hex = {'02' '03' '01' '01'};     % 第一行
row = hex2dec(row_hex)';             % 转置后才是行向量
rows = repmat(row,4,1);              % 复制第一行，生成 4×4 矩阵
MC_mat = cycle(rows,'right');      % 向右循环移位

% 逆列混淆矩阵的构造
row_inv_hex = {'0e' '0b' '0d' '09'};% 与列混淆矩阵第一行点乘，乘积为 1
row_inv = hex2dec(row_inv_hex)';
rows_inv = repmat(row_inv,4,1);
MC_mat_inv = cycle(rows_inv,'right');%向右循环移位
end
```

5. **矩阵的循环移位**

```
function matrix_out = cycle(matrix_in,direction)
%   实现输入矩阵第 n 行向左或者向右移动 n 字节
%% 做法一，比较巧妙，参照网上的资料
% if strcmp(direction,'left')
%     col = (0:5:15)';% 左移
% else
%     col = (16:-3:7)';% 右移
% end
% row = 0:4:12;
% cols = repmat(col,1,4);
% rows = repmat(row,4,1);
% mat = mod(rows + cols,16)+1;%要+1 否？
% matrix_out = matrix_in(mat);

%% 做法二，易于理解
matrix_out = zeros(size(matrix_in));
matrix_out(1,:) = matrix_in(1,:);
if strcmp(direction,'left')
```

```
    for i = 2:4
        matrix_out(i,:) = round_shift(matrix_in(i,:),'left',i-1)
    end
else
    for i = 2:4
        matrix_out(i,:) = round_shift(matrix_in(i,:),'right',i-1)
    end
end
end
```

6. 向量的循环位移

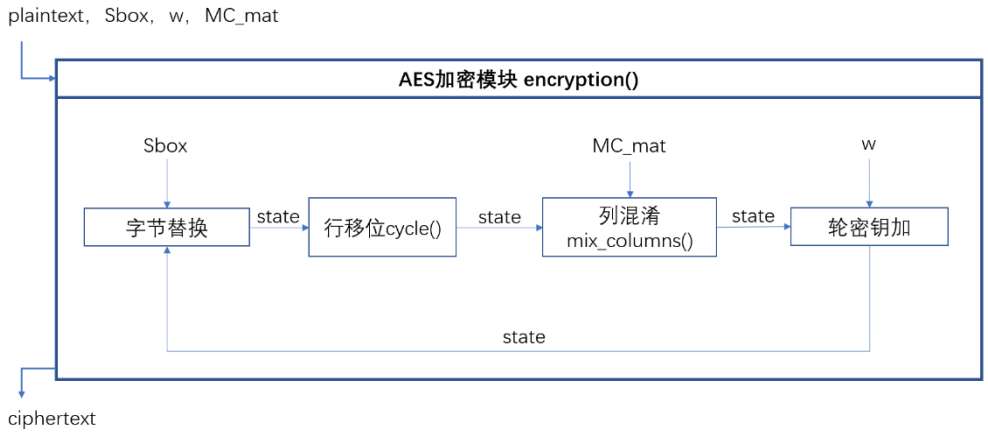
```
function [w_out] = round_shift(w_in,direction,step)
% 函数功能：现密钥 wj 向量向左或者向右移指定字节长度，若无输入方向，则默认向左循环位移。
% 备注：由于每一位是[0~256]的十进制数组成，对应一个字节的长度（8 位二进制数）。故该函数将 wj 循环左移 8 比特。
if nargin==1
    w_out = w_in([2 3 4 1]);     % 如果不输入 direction，则默认循环左移
elseif nargin==2                 % 如果不输入位移步数 step，默认只位移一位
    if strcmp(direction,'left') % 左移
        w_out = w_in([2 3 4 1]);
    else
        w_out = w_in([4 1 2 3]);
    end
else
         w_out = w_in;
    if strcmp(direction,'left') % 循环左移
        while(step)              % 循环移位，直至步数 step 降到 0
            w_out = w_out([2 3 4 1]);
            step = step-1;
        end
    else                         % 循环右移
        while(step)
```

```
            w_out = w_out([4 1 2 3]);
            step = step-1;
        end
    end
end

end
```

3.2 AES 加密

程序框图如下所示:



主函数及其子函数如下所示：

```
function ciphertext = encryption(plaintext,w,Sbox,MC_mat)
% 函数功能：实现加密
% 输入参数：
%       plaintext：明文
%       w：密钥
%       Sbox：S 盒
%       MC_mat：用于列混淆变换的矩阵
    state = reshape(plaintext,4,4);%将明文进行重排
%% 预轮密钥加
    round_key = (w(1:4,:))'%密钥，根据图示需要转置后再异或
    state = bitxor(state,round_key);%轮密钥加


%% 一~九轮加密
```

```
    for i = 1:9
        state =  Sbox(state + 1);%字节替换
        state = cycle (state, 'left');%行移位
        state = mix_columns(state,MC_mat);%列混淆
        round_key = (w((1:4)+4*i,:))';%取出每一轮对应的圆常数
        state = bitxor(state,round_key);%轮密钥加
    end
    %% 第十轮，没有列混淆
    state = Sbox(state + 1);%字节替换
         state = cycle (state, 'left');%行移位
    round_key = (w(41:44,:))';%取出最后一轮对应的圆常数
         state = bitxor(state,round_key);%轮密钥加
    ciphertext = reshape(state,1,16);
end
```

1. **列混淆**

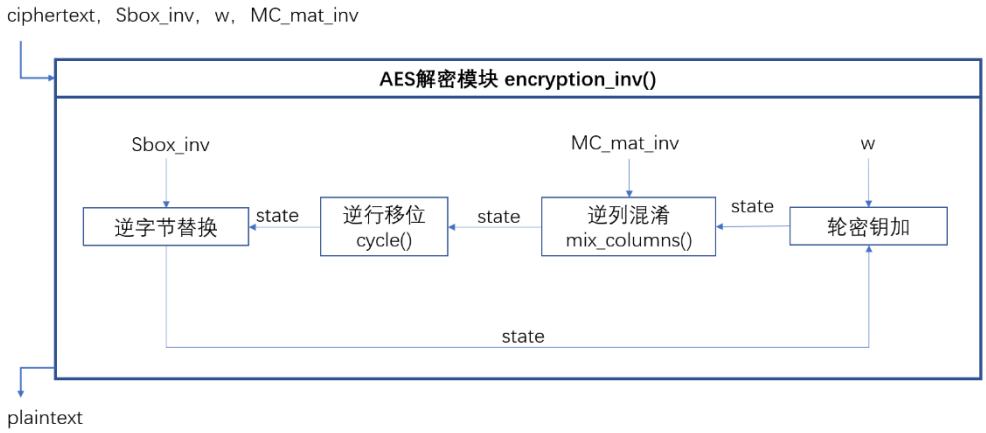
AES 涉及到$GF(2^8)$的不可约多项式（本原多项式）为：$m(x)=x^8+x^4+x^3+x+1$。多项式 m(x) 对应的二进制数为 '100011011'，故我们其赋值给模值多项式参数 mod_pol。

```
function state_out = mix_columns(state_in,MC_mat)
% 函数功能：实现列混淆
% 输入参数：
%       state_in：待进行列混淆的矩阵
%       MC_mat：列混淆变换矩阵
% 输出参数：
%       state_out：列混淆后的矩阵
    mod_pol = bin2dec('100011011');
    for col = 1:4
        for row = 1:4
            temp = 0;
            for inner = 1:4      % 对矩阵中的元素逐个相乘，
                prod = poly_mult(...
                        MC_mat(row,inner),...
                        state_in(inner,col),...
```

```
                    mod_pol);
                %利用定义好的多项式乘法与取余函数，依照矩阵乘法对逐个元素进行相乘
                %由于取余，可保证结果小于设定的 mod_pol
                temp = bitxor(temp,prod);%对结果累计异或（等效于求和），得到行向量与列向量点乘后的结果
            end
            state_out(row,col) = temp;
        end
    end
end
```

3.3 AES 解密

程序框图如下所示：



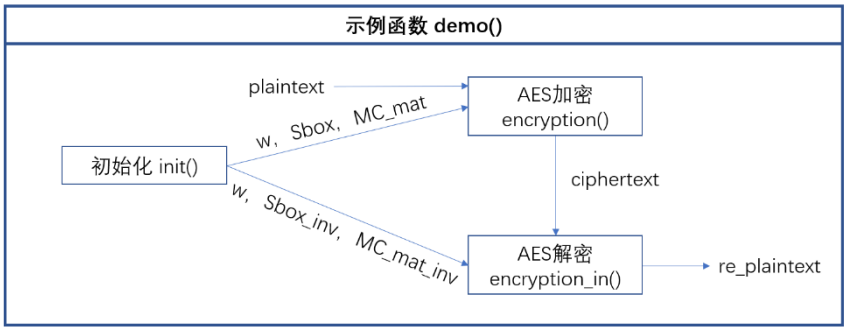
主函数及其子函数如下所示：

```
function plaintext = encryption_inv(ciphertext,w,Sbox_inv,MC_mat_inv)
% 函数功能：实现 AES 解密
% 输入参数：
%       ciphertext：密文
%       w：密钥
%       Sbox_inv：逆 S 盒
%       MC_mat_inv：用于逆列混淆变换的矩阵
% 输出参数：
%       plaintext：解密后的明文
```

```
    state = reshape(ciphertext,4,4);%将密文进行重排
    %% 第一轮解密（对应第十轮加密）
    round_key = (w(41:44,:))';%密钥，根据图示需要转置后再异或
    state = bitxor(state,round_key);%轮密钥加
    state = cycle (state, 'right');%逆行移位
    state =  Sbox_inv(state + 1);%逆字节替换
    %% 二~十轮解密（对应第九~一轮加密）
    for i = 9:-1:1
        round_key = (w((1:4)+4*i,:))';%取出每一轮对应的圆常数
        state = bitxor(state,round_key);%逆轮密钥加
        state = mix_columns(state,MC_mat_inv);%逆列混淆
        state = cycle (state, 'right');%逆行移位
         state =  Sbox_inv(state + 1);%逆字节替换
    end
    %% 轮密钥加（对应 AES 加密的预轮密钥加）
    round_key = (w(1:4,:))';%取出每一轮对应的圆常数
    state = bitxor(state,round_key);%逆轮密钥加
    plaintext = reshape(state,1,16);
end
```

3.4 示例函数

程序框图如下所示:



主函数及其子函数如下所示：

```
function [] = demo()
% 函数功能：对设定的明文进行 AES 加密，并解密
% 初始化
[Sbox,Sbox_inv,w,poly_mat,poly_mat_inv] = init;
% 设定明文
plaintext_hex = {'00' '11' '22' '33' '44' '55' '66' '77'...
```

```
                    '88' '99' 'aa' 'bb' 'cc' 'dd' 'ee' 'ff'};
plaintext = hex2dec(plaintext_hex);
% 加密
ciphertext = encryption(plaintext,w,Sbox,poly_mat);
% 解密
re_plaintext = encryption_inv(ciphertext,w,Sbox_inv,poly_mat_inv);
% 输出
fprintf('明文为：');
disp(plaintext')
fprintf('密文为：');
disp(ciphertext)
fprintf('解密文为：');
disp(re_plaintext)
end
```

三、仿真结果

运行程序发现能够正确加解密，好耶ヾ(✿ﾟ▽ﾟ)ノ

```
明文为：     0    17    34    51    68    85   102   119   136   153   170   187   204   221   238   255

密文为：   105   196   224   216   106   123     4    48   216   205   183   128   112   180   197    90

解密文为：     0    17    34    51    68    85   102   119   136   153   170   187   204   221   238   255
```

参考资料：

[1]https://blog.csdn.net/gulang03/article/details/81175854《现代密码学教程》

[2]chrome-extension://ikhdkkncnoglghljlkmcimlnlhkeamad/pdf-viewer/web/viewer.html?file=https%3A%2F%2Fir.nctu.edu.tw%2Fbitstream%2F11536%2F41079%2F3%2F751403.pdf#=&zoom=130（S 盒与逆 S 盒的生成）

[3]https://zh.wikipedia.org/wiki/Rijndael%E5%AF%86%E9%92%A5%E7%94%9F%E6%88%90%E6%96%B9%E6%A1%88（圆常数的生成）

[4]chrome-extension://ikhdkkncnoglghljlkmcimlnlhkeamad/pdf-viewer/web/viewer.html?file=https%3A%2F%2Fcs.au.dk%2F~ivan%2Frijndael.pdf#=&zoom=130（AES Proposal: Rijndael）