浙江大学

本科实验报告

课程名称：网络与通信安全

姓　　名：蔡松成

学　　院：信息与电子工程学院

专　　业：信息工程

学　　号：3200103584

指导教师：谢磊、陈惠芳

2022 年 6 月 13 日

专业：信息工程
姓名：蔡松成
学号：3200103584
日期：2023.6.13

浙江大学实验报告

课程名称： 网络与通信安全 **成绩：**

实验名称： AES 与基于 AES 的 CMAC **实验类型：** 设计型

一、实验目的

1. 熟悉 AES 加密的知识，掌握其算法实现；
2. 熟悉基于 AES 的 CMAC 生成，掌握其算法实现。

二、实验任务

1. 目标：用 MATLAB 或 C 实现加解密算法及其应用：AES 和基于 AES 的 CMAC
2. 附加要求：

(1) 加密密钥：自己的姓名汉字用 UNICODE 编码后扩展成需要的位数，例如“蔡松成”经过 UNICODE 编码后变成 6 字节十六进制数 85 21 67 7E 62 10，本实验采用重复扩展，将其扩展为 16 字节密钥 85 21 67 7E 62 10 85 21 67 7E 62 10 85 21 67 7E。

(2) 加密与认证消息：Information Security is a multidisciplinary area of study and professional activity which is concerned with the development and implementation of security mechanisms of all available types to keep information in all its locations and, consequently, information systems, where information is created, processed, stored, transmitted and destroyed, free from threats. This project is finished by CAISONGCHENG.

三、实验原理

1. AES 原理

AES（Advanced Encryption Standard）是一种对称加密算法，用于保护敏感

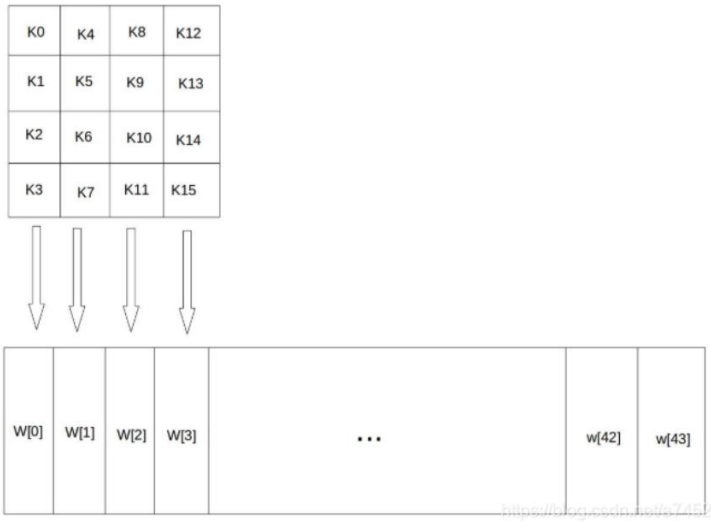
数据的机密性。它是目前广泛应用的加密标准之一。

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，如下表所示：

AES	密钥长度（32位比特字)	分组长度(32位比特字)	加密轮数
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

这里实现的是 AES-128，也就是密钥的长度为 128 位，加密轮数为 10 轮。AES 的加密公式为 C = E(K,P)，在加密函数 E 中，会执行一个轮函数，并且执行 10 次这个轮函数，这个轮函数的前 9 次执行的操作是一样的，只有第 10 次有所不同。也就是说，一个明文分组会被加密 10 轮。AES 的核心就是实现一轮中的所有操作。

AES 的处理单位是字节，128 位的输入明文分组 P 和输入密钥 K 都被分成 16 个字节，分别记为 P = P0 P1 … P15 和 K = K0 K1 … K15。一般地，明文分组用字节为单位的正方形矩阵描述，称为状态矩阵。在算法的每一轮中，状态矩阵的内容不断发生变化，最后的结果作为密文输出。该矩阵中字节的排列顺序为从上到下、从左至右依次排列。



如上图所示：类似地，128 位密钥也是用字节为单位的矩阵表示，矩阵的每

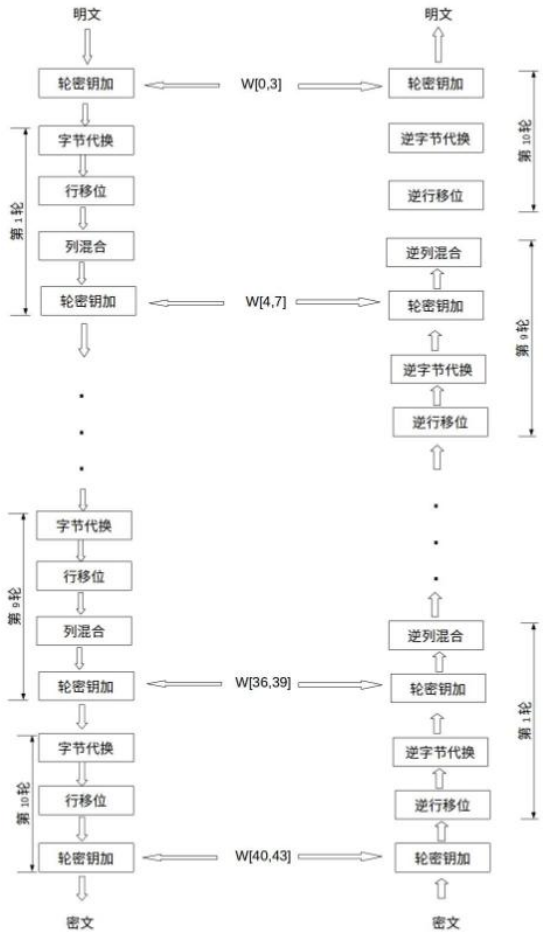
一列被称为 1 个 32 位比特字。通过密钥编排函数该密钥矩阵被扩展成一个 44 个字组成的序列 W[0],W[1], … ,W[43], 该序列的前 4 个元素 W[0],W[1],W[2],W[3]是原始密钥，用于加密运算中的初始密钥加（下面介绍）；后面 40 个字分为 10 组，每组 4 个字（128 比特）分别用于 10 轮加密运算中的轮密钥加。后面 10 个新列以如下的递归方式产生：如果 i 不是 4 的倍数，那么第 i 列由如下等式确定：W[i]=W[i-4]⊕W[i-1]；如果 i 是 4 的倍数，那么第 i 列由如下等式确定：W[i]=W[i-4]⊕T(W[i-1])；

其中，T 是一个有点复杂的函数。函数 T 由三部分组成：字循环、字节代换和轮常量异或，具体操作如下：

a. 字循环：将 1 个字中的 4 个字节循环左移 1 个字节。即将输入字[b0, b1, b2, b3]变换成[b1,b2,b3,b0]。

b. 字节代换：对字循环的结果使用 S 盒进行字节代换。

c. 轮常量异或：将前两步的结果同轮常量 Rcon[j]进行异或，其中 j 表示轮数。



AES 的整体结构如上图所示，其中的 W[0,3]是指 W[0]、W[1]、W[2]和 W[3]

串联组成的 128 位密钥。加密的第 1 轮到第 9 轮的轮函数一样，包括 4 个操作：字节代换、行位移、列混淆和轮密钥加。最后一轮迭代不执行列混淆。另外，在第一轮迭代之前，先将明文和原始密钥进行一次异或加密操作。下面将介绍这四种操作的具体原理：

① 字节代换（SubBytes）：

在这一步骤中，AES 将输入数据（通常为 16 字节）看作一个 4x4 的字节矩阵。对矩阵中的每个字节进行一个固定的非线性代换操作，称为 S 盒替换。S 盒是一个 256 字节的查找表，将每个输入字节映射为一个输出字节，把输入字节的高 4 位作为行值，低 4 位作为列值，取出 S 盒或者逆 S 盒中对应的行的元素作为输出。这个代换操作使得每个输入字节都被替换为一个新的字节值。

② 行移位变换（ShiftRows）：

在这一步骤中，AES 对字节矩阵的每一行进行循环左移操作。第一行保持不变，第二行左移 1 个字节，第三行左移 2 个字节，第四行左移 3 个字节。这个操作使得字节矩阵中的数据在列之间发生移动，增加了混淆度。行移位的逆变换是将状态矩阵中的每一行执行相反的移位操作，例如 AES-128 中，状态矩阵的第 0 行右移 0 字节，第 1 行右移 1 字节，第 2 行右移 2 字节，第 3 行右移 3 字节。

③ 列混淆（MixColumns）：

在这一步骤中，AES 对字节矩阵的每一列进行线性变换。每个字节与固定的多项式进行乘法和加法操作，以实现列之间的混淆。这个操作使得字节矩阵中的数据在列之间进行了混淆和扩散，增加了密码算法的安全性。具体线性变换可由下面的矩阵运算表示：

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

逆向列混合变换可由下图的矩阵乘法定义：

$$\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

④ 轮密钥加（AddRoundKey）：

在这一步骤中，AES 将轮密钥与字节矩阵进行逐字节的异或操作。轮密钥是通过之前介绍的密钥扩展算法从主密钥生成的，每一轮使用不同的轮密钥。这个操作相当于对字节矩阵中的每个字节进行密钥的加密，将密钥的信息混入到加密数据中。轮密钥加的逆运算同正向的轮密钥加运算完全一致，这是因为异或的逆操作是其自身。轮密钥加非常简单，但却能够影响 S 数组中的每一位。

2. 基于 AES 的 CMAC 原理

CMAC（Cipher Block Chaining-Message Authentication Code），也简称为 CBC_MAC，它是一种基于对称秘钥分组加密算法的消息认证码。由于其是基于“对称秘钥分组算法”的，故可以将其当做是对称算法的一种操作模式。

CMAC 可以应用的算法主要有：AES、DES、3DES 等。本实验实现的是基于 128bitAES 的 CMAC，下面对其内在原理进行探讨：

(1) 生成子密钥 K1、K2；

a. 使用 AES 算法对 128 位全 0 消息加密，得到加密后的 128 位消息 L；

b. 若 L 的最高位为 0，则 K1 = L<< 1。否则，K1 = (L << 1)再与 Rb 异或；

c. 若 K1 的最高位为 0，则 K2 = K1<< 1。否则，K2 = (K1 << 1)再与 Rb 异或。

其中，Rb 为给定的常量字符串，对于 AES 算法，其值为 128 位的数据，前面 120 位都为 0，最后 8 位为 10000111。

(2) 将数据按分组长度 16 字节分块，若分块数为 n，则对前 n-1 个分块执行 AES 算法；

(3) 若最后一个块是完整块，则将其与 K1 异或后作为输入数据，并执行 AES 算法；

(4) 若最后一个块为非完整块，则用 10...0 模式填充该块，再与 K2 异或后作为最后一个块的输入数据，并执行 AES 算法；

(5) 将最后一个分组的计算结果作为 CMAC 输出值；

四、AES 算法实现

1. 字节代换

根据传入的 S 盒或者逆 S 盒，将 4*4 的输入矩阵映射为新的矩阵。代码如下：

```
function bytes_out = sub_bytes (bytes_in, s_box)
% 根据生成的 S_BOX 进行字节代换
for i=1:4
    for j=1:4
        bytes(i,j)=hex2dec(bytes_in(i,j));
        x=mod(bytes(i,j),16)+1;
        y=fix(bytes(i,j)/16)+1;
        bytes_out(i,j) = s_box (y,x);
    end
end
```

其中 S 盒的生成与逆 S 盒的生成代码如下：

```
function s_box = generate_AES_SBox()
    % 生成 AES 中的 S 盒

    % 定义 S 盒生成中使用的常量
    mod_value = bin2dec('100011011'); % 模数
    mult_value = bin2dec('00011111'); % 乘法常量
    add_value = bin2dec('01100011'); % 加法常量
```

```
    % 生成中间值 temp_x
    temp_x(1) = 0;
    for i = 1 : 255
        for j = 1 : 255
            temp = 0;
            for bit_pos = 1 : 8
                if bitget(i, bit_pos)
                    temp = bitxor(temp, bitshift(j, bit_pos-1));
                end
            end
            for bit_pos = 16 : -1 : 9
                if bitget(temp, bit_pos)
                    temp = bitxor(temp, bitshift(mod_value, bit_pos - 9));
                end
            end
            if temp == 1
                bx = j;
                break
            end
        end
        temp_x(i + 1) = bx;
    end
```

```
    mod_value = bin2dec('100000001');
    % 根据 temp_x 生成 S 盒
    s_box = cell(16, 16);
    for i = 1 : 256
        temp = 0;
        for bit_pos = 1 : 8
            if bitget(temp_x(i), bit_pos)
                temp = bitxor(temp, bitshift(mult_value, bit_pos-1));
            end
        end
        for bit_pos = 16 : -1 : 9
            if bitget(temp, bit_pos)
                temp = bitxor(temp, bitshift(mod_value, bit_pos-9));
            end
        end
        s_box{floor((i - 1) / 16) + 1, mod(i - 1, 16) + 1} = dec2hex(bitxor(temp, add_value), 2);
    end
end
function inv_sbox = generate_inverse_AES_SBox()
 sbox = [
        '52' '09' '6A' 'D5' '30' '36' 'A5' '38' 'BF' '40' 'A3' '9E' '81' 'F3' 'D7' 'FB';
        '7C' 'E3' '39' '82' '9B' '2F' 'FF' '87' '34' '8E' '43' '44' 'C4' 'DE' 'E9' 'CB';
        '54' '7B' '94' '32' 'A6' 'C2' '23' '3D' 'EE' '4C' '95' '0B' '42' 'FA' 'C3' '4E';
        '08' '2E' 'A1' '66' '28' 'D9' '24' 'B2' '76' '5B' 'A2' '49' '6D' '8B' 'D1' '25';
        '72' 'F8' 'F6' '64' '86' '68' '98' '16' 'D4' 'A4' '5C' 'CC' '5D' '65' 'B6' '92';
        '6C' '70' '48' '50' 'FD' 'ED' 'B9' 'DA' '5E' '15' '46' '57' 'A7' '8D' '9D' '84';
        '90' 'D8' 'AB' '00' '8C' 'BC' 'D3' '0A' 'F7' 'E4' '58' '05' 'B8' 'B3' '45' '06';
        'D0' '2C' '1E' '8F' 'CA' '3F' '0F' '02' 'C1' 'AF' 'BD' '03' '01' '13' '8A' '6B';
        '3A' '91' '11' '41' '4F' '67' 'DC' 'EA' '97' 'F2' 'CF' 'CE' 'F0' 'B4' 'E6' '73';
        '96' 'AC' '74' '22' 'E7' 'AD' '35' '85' 'E2' 'F9' '37' 'E8' '1C' '75' 'DF' '6E';
        '47' 'F1' '1A' '71' '1D' '29' 'C5' '89' '6F' 'B7' '62' '0E' 'AA' '18' 'BE' '1B';
        'FC' '56' '3E' '4B' 'C6' 'D2' '79' '20' '9A' 'DB' 'C0' 'FE' '78' 'CD' '5A' 'F4';
        '1F' 'DD' 'A8' '33' '88' '07' 'C7' '31' 'B1' '12' '10' '59' '27' '80' 'EC' '5F';
        '60' '51' '7F' 'A9' '19' 'B5' '4A' '0D' '2D' 'E5' '7A' '9F' '93' 'C9' '9C' 'EF';
        'A0' 'E0' '3B' '4D' 'AE' '2A' 'F5' 'B0' 'C8' 'EB' 'BB' '3C' '83' '53' '99' '61';
        '17' '2B' '04' '7E' 'BA' '77' 'D6' '26' 'E1' '69' '14' '63' '55' '21' '0C' '7D'
    ];

    inv_sbox = cell(16, 16);

    for i = 1:16
        for j = 1:16
            inv_sbox{i, j} = sbox(i, 2*(j-1)+1:2*(j-1)+2);
        end
```

```
    end
end
```

2. 行移位变换

根据之前讨论的原理，行移位与逆行移位就是根据行数进行循环左移或者右移，代码如下：

```
function out = shift_rows (in)
% 按照第几行进行行移位
for i=1:4
    for j=1:4
        if (i-j) <= 0
            out(i,1-i+j) = in(i,j);
        else
            out(i,5-i+j) = in(i,j);
        end
    end
end
function out = inv_shift_rows (in)
% 按照第几行进行逆向行移位
for i = 1:4
    for j = 1:4
        if i == 2
            out(i, j) = in(i, mod(j-2, 4) + 1);
        elseif i == 3
            out(i, j) = in(i, mod(j-3, 4) + 1);
        elseif i == 4
            out(i, j) = in(i, mod(j-4, 4) + 1);
        else
            out(i, j) = in(i, j);
        end
    end
end
```

3. 列混淆

对 4*4 的矩阵进行列混淆或者逆列混淆就是按照原理对矩阵在有限域进行矩阵线性变换，代码如下（传入的 P 是列混淆矩阵，需要自己定义）：

```
function out = mix_columns(in, p)
modulus = bin2dec('100011011');
```

```
for i = 1:4
    for j = 1:4
        x = 0;
        for k = 1:4
            ab = 0;
            a_val = hex2dec(p(i, k));
            b_val = hex2dec(in(k, j));

            % 矩阵乘法
            for bit = 1:8
                if bitget(a_val, bit)
                    ab = bitxor(ab, bitshift(b_val, bit - 1));
                end
            end

            % 模运算
            for bit = 16:-1:9
                if bitget(ab, bit)
                    ab = bitxor(ab, bitshift(modulus, bit - 9));
                end
            end

            x = bitxor(x, ab);
        end
        out(i, j) = cellstr(dec2hex(x, 2));
    end
end
```

4. 轮密钥加

轮密钥加无论正逆都是 state 矩阵与轮密钥进行异或操作即可，代码实现如下：

```
function out = xor_round_key(a, b)
% 进行轮密钥加操作，输入参数 a 和 b 为两个 4x4 的十六进制值的单元格数组
% 返回结果为 4x4 的十六进制值的单元格数组 out
    for i = 1:4
        for j = 1:4
            % 将十六进制值转换为十进制，并进行异或运算
            xor_result(i,j) = bitxor(hex2dec(a(i,j)), hex2dec(b(i,j)));
            % 将异或结果转换为两位的十六进制字符串，并存储到 out 数组中
            out(i,j) = cellstr(dec2hex(xor_result(i,j), 2));
        end
    end
end
```

5. AES 加密实现

利用上述实现的功能函数，组合起来实现十轮加密，密钥为“蔡松成”的 Unicode 编码扩展 16 字节，加密信息由自己输入，代码实现如下：

```
% 输入明文
inputChar = input('请输入字符串：', 's');
% 转换为 ASCII 编码
input_code = double(inputChar);
% 计算补零的个数
numPadding = 16 - mod(length(input_code), 16);
paddingzero = double(zeros(1,numPadding));
% 添加补零
input_code_Padded = [input_code, paddingzero];
% 转换为十六进制表示
hex_input_code_Padded = dec2hex(input_code_Padded);
% 创建一个 n*16 的全 0 cell
input_matrix = cell(size(hex_input_code_Padded,1)/16, 16);

% 对每个 cell 进行赋值
for i = 1:size(hex_input_code_Padded,1)/16
    for j = 1:16
        value = hex_input_code_Padded((i-1)*16+j,1:2);
        input_matrix{i, j} = value;
    end
end
```

```
% 名字密钥转换为 Unicode 编码
% name_code = unicode2native('蔡松成');
% hex_name_code = cellstr(dec2hex(name_code));
% hex_name_code = repmat(hex_name_code, ceil(16/length(hex_name_code)), 1);
% hex_name_code = hex_name_code(1:16, :);
% hex_name_code = reshape(cellstr(hex_name_code), 4, 4);
% fid=fopen('key.txt','wt');
% for i=1:4
%    for j=1:4
%       a = hex_name_code(i,j);
%       fwrite(fid,cell2mat(a));
%       fwrite(fid,' ');
%    end
%       fprintf(fid,'\n');
% end
% sta=fclose(fid);
```

```
% 定义列混淆矩阵
```

```
p={'2' '3' '1' '1';'1' '2' '3' '1'; '1' '1' '2' '3';'3' '1' '1' '2'};
% 生成 S 盒
s_box = generate_AES_SBox();
for group = 1:size(input_matrix,1)
    state = reshape(input_matrix(group,1:16),4,4);
    writeout (' INPUT :', state, group);
    round_key = AES_key(1);
    writeout ('  KEY  :', round_key, group);
    state = xor_round_key (state, round_key);
    for round = 1:10
        writeout ('********  ROUND  ********',round, group);
        writeout ('      Start of round    ', state, group);
        state = sub_bytes (state, s_box);
        writeout ('      After sub_bytes     ', state, group);
        state = shift_rows (state);
        writeout ('     After  shift_rows    ', state, group);
        if round <10
            state = mix_columns (state, p);
            writeout ('     After mix_columns    ', state, group);
        end
        round_key=AES_key(round+1);
        writeout ('        Round   key       ', round_key, group);
        state = xor_round_key (state, round_key);
    end
    writeout ('        Final  state        ', state, group);
end
```

6. AES 解密实现

与加密类似，注意每轮的解密用的密钥使用顺序与加密相反，按逆行变换，逆字节代换，轮密钥加，逆列混淆的顺序，解密前 9 轮，最后一轮没有逆列混淆。，代码实现如下：

```
% 从 outx.txt 文件中读取密文
folder = 'result';
files = dir(fullfile(folder,'*.txt'));
% 创建存放密文的 cell
ciphertext = cell(length(files),16);
for i = 1:length(files)
    filepath = ['result/out',num2str(i),'.txt'];
    fid=fopen(filepath);
    data=textscan(fid,'%s','Delimiter','\n');
    for x = 1:4
        for y = 1:4
```

```
            ciphertext{i,(x-1)*4+y} = data{1,1}{380+y,1}(1+(x-1)*3:2+(x-1)*3);
        end
    end
    fclose(fid);
end

% 定义列混淆矩阵
p={'0E' '0B' '0D' '09';'09' '0E' '0B' '0D'; '0D' '09' '0E' '0B';'0B' '0D' '09' '0E'};
% 生成 S 盒
inv_sbox = generate_inverse_AES_SBox();
fid=fopen('decryption.txt','at+');
for group = 1:length(files)
    state = reshape(ciphertext(group,1:16),4,4);
    round_key = AES_key(11);
    state = xor_round_key (state, round_key);
    for round = 1:10
        state = inv_shift_rows (state);
        state = sub_bytes (state, inv_sbox);
        round_key=AES_key(11-round);
        state = xor_round_key (state, round_key);
        if round <10
            state = mix_columns (state, p);
        end
    end
    for i = 1:4
        for j = 1:4
            disp(char(hex2dec(state{j,i})));
            fwrite(fid,char(hex2dec(state{j,i})));
        end
    end
end
sta=fclose(fid);
```

五、CMAC 算法实现

先输入认证消息，并且判断是否为刚好分好组，来决定使用 K1 还是 K2(这里因为事先判断输入的信息不是刚好分组)，先算出 K2，再对分组进行加密，加密后的密文作为输入与明文异或后再加密，重复至最后一轮后介绍，获得 MAC。

```
% 输入明文
inputChar = input('请输入字符串：', 's');
% 转换为 ASCII 编码
input_code = double(inputChar);
% 计算补零的个数
numPadding = 16 - mod(length(input_code), 16) - 1;
padding = double(zeros(1,numPadding));
padding = [128,padding];
input_code_Padded = [input_code, padding];
% 转换为十六进制表示
hex_input_code_Padded = dec2hex(input_code_Padded);
% 创建一个 n*16 的全 0 cell
input_matrix = cell(size(hex_input_code_Padded,1)/16, 16);

% 对每个 cell 进行赋值
for i = 1:size(hex_input_code_Padded,1)/16
    for j = 1:16
        value = hex_input_code_Padded((i-1)*16+j,1:2);
        input_matrix{i, j} = value;
    end
end
```

```
% 定义列混淆矩阵
p={'2' '3' '1' '1';'1' '2' '3' '1'; '1' '1' '2' '3';'3' '1' '1' '2'};
% 生成 S 盒
s_box = generate_AES_SBox();
% 对 key 使用全 0 密钥加密
state = cell(4,4);
for i = 1:4
    for j = 1:4
        state{i,j} = '00';
    end
end
round_key=AES_key(1);
state = xor_round_key (state, round_key);
for round = 1:10
    state = sub_bytes (state, s_box);
    state = shift_rows (state);
    if round <10
       state = mix_columns (state, p);
    end
    round_key=AES_key(round+1);
    state = xor_round_key (state, round_key);
end
```

```
% 生成 K2
const_Rb = textread('const_Rb.txt','%s')';
const_Rb = reshape(const_Rb,4,4)';
L = '';
for i = 1:4
    for j = 1:4
        L = [L,state{j,i}];
    end
end
if L(1) >= '8'
   binarySequence = dec2bin(hex2dec(reshape(L, 2, [])'), 8);
   binarySequence = reshape(binarySequence', 1, []);
   binarySequence = [binarySequence(2:128),'0'];
   binarySegments = reshape(binarySequence, 8, 16)';
   decNums = bin2dec(binarySegments);
   hexSeqs = dec2hex(decNums);
   L_shift = cell(4,4);
   for i = 1:4
        for j = 1:4
            L_shift{i,j} = hexSeqs((j-1)*4+i,1:2);
        end
   end
   K1 = xor_round_key(L_shift,const_Rb);
else
   binarySequence = dec2bin(hex2dec(reshape(L, 2, [])'), 8);
   binarySequence = reshape(binarySequence', 1, []);
   binarySequence = [binarySequence(2:128),'0'];
   binarySegments = reshape(binarySequence, 8, 16)';
   decNums = bin2dec(binarySegments);
   hexSeqs = dec2hex(decNums);
   L_shift = cell(4,4);
   for i = 1:4
        for j = 1:4
            L_shift{i,j} = hexSeqs((j-1)*4+i,1:2);
        end
   end
   K1 = L_shift;
end
K1_Sequence = '';
for i = 1:4
    for j = 1:4
        K1_Sequence = [K1_Sequence,K1{j,i}];
    end
end
```

```
if K1_Sequence(1) >= '8'
   binarySequence = dec2bin(hex2dec(reshape(K1_Sequence, 2, []).'), 8);
   binarySequence = reshape(binarySequence.', 1, []);
   binarySequence = [binarySequence(2:128),'0'];
   binarySegments = reshape(binarySequence, 8, 16)';
   decNums = bin2dec(binarySegments);
   hexSeqs = dec2hex(decNums);
   K1_shift = cell(4,4);
   for i = 1:4
        for j = 1:4
            K1_shift{i,j} = hexSeqs((j-1)*4+i,1:2);
        end
   end
   K2 = xor_round_key(K1_shift,const_Rb);
else
   binarySequence = dec2bin(hex2dec(reshape(K1_Sequence, 2, []).'), 8);
   binarySequence = reshape(binarySequence.', 1, []);
   binarySequence = [binarySequence(2:128),'0'];
   binarySegments = reshape(binarySequence, 8, 16)';
   decNums = bin2dec(binarySegments);
   hexSeqs = dec2hex(decNums);
   K1_shift = cell(4,4);
   for i = 1:4
        for j = 1:4
            K1_shift{i,j} = hexSeqs((j-1)*4+i,1:2);
        end
   end
   K2 = K1_shift;
end
% 存放上次加密的密文
temp = cell(4,4);
% CMAC
for group = 1:size(input_matrix,1)
    if group == 1
        state = reshape(input_matrix(group,1:16),4,4);
    elseif group < size(input_matrix,1)
        state = xor_round_key(reshape(input_matrix(group,1:16),4,4),temp);
    elseif group == size(input_matrix,1)
        state = xor_round_key(reshape(input_matrix(group,1:16),4,4),temp);
        state = xor_round_key(state, K2);
    end
    round_key=AES_key(1);
    state = xor_round_key (state, round_key);
    for round = 1:10
```

```
        state = sub_bytes (state, s_box);
        state = shift_rows (state);
        if round <10
            state = mix_columns (state, p);
        end
        round_key=AES_key(round+1);
        state = xor_round_key (state, round_key);
    end
    temp = state;
end
disp('CMAC 认证码：');
disp(state);
```

六、实验结果

1. AES 加密结果

AES 加密过程都写在 result 文件夹的 out.txt 文件中，每轮的操作后的矩阵都显示在文件中，最后的 Final State 下的 4*4 矩阵是该分组的明文加密后的密文，out 文件的序号代表第几个分组。例如第一个分组加密后的密文如下，写在 out1.txt 文件中：

Final state

54 88 6E A4
31 19 AC 02
38 9A F0 48
D2 E9 8F AC

2. AES 解密结果

读取 result 文件夹内的 out 文件中的密文，对其使用 AES 解密算法，求出每组明文，并将其转换为字符，解密结果写在 decryption.txt 文件中。输出如下：

Information Security is a multidisciplinary area of study and professional activity which is concerned with the development and implementation of security mechanisms of all available types to keep information in all its locations and, consequently, information systems, where information is created, processed, stored, transmitted and destroyed, free from threats. This project is finished by CAISONGCHENG.

与加密前的明文一模一样，证明 AES 算法正确。

3. CMAC 生成

生成的 16 字节 CMAC 认证码如下，写在 CMAC_output.txt 文件中：

'EC' '1E' 'A0' '52'

'46' '13' 'AD' '9A'

'72' 'E4' '4F' 'D0'

'25' '5D' '22' '31'

七、实验心得

本次实验让我对上课所学到的 AES 与 CMAC 算法更加熟悉，也有了更加深刻的理解，在 debug 过程中我也发现了之前对这两个算法的一些误解，比如加解密的四个操作的进行顺序是不一样的，否则无法得出最终正确的结果，通过查阅网上资料，我了解了解密证明的过程，也明白了解密的操作顺序的原因，受益匪浅。