**信息科学与工程学院**

**2021－2022学年第一学期**

实 验 报 告

课程名称： 信息安全

实验名称： 实验五

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2021年10月23日

实验报告

### 【实验目的】

掌握AES加密的原理。

### 【实验要求】

编写128bit AES加密程序。

### 【源代码】

### (1) AES()

|  |  |
| --- | --- |
|  | function cipher\_text=AES(M,K) |
|  | %  --------------------------------------------- |
|  | %   数据加密标准AES算法的Matlab实现 |
|  | %   M:输入明文，32位16进制数，字符串格式 |
|  | %   K:输入密钥，32位16进制数，字符串格式 |
|  | %  --------------------------------------------- |
|  |  |
|  | %  ------------第一步：密钥扩展------------ |
|  | p(1,:)=K(1:8); |
|  | p(2,:)=K(9:16); |
|  | p(3,:)=K(17:24); |
|  | p(4,:)=K(25:32); |
|  | for i=1:1:10 |
|  | temp=dec2hex(bitxor(hex2dec(p(1+(i-1)\*4,:)),g\_function(p(4\*i,:),i))); |
|  | temp=[zeros(1,8-length(temp)),temp]; |
|  | temp(abs(temp)==0)='0'; |
|  | p(1+i\*4,:)=temp; |
|  | for j=2:1:4 |
|  | temp=dec2hex(bitxor(hex2dec(p(j+(i-1)\*4,:)),hex2dec(p(j+i\*4-1,:)))); |
|  | temp=[zeros(1,8-length(temp)),temp]; |
|  | temp(abs(temp)==0)='0'; |
|  | p(j+i\*4,:)=temp; |
|  | end |
|  | end |
|  |  |
|  | %  ------------第二步：十轮循环------------ |
|  | %初始轮密相加 |
|  | temp=add\_key(M,p,0); |
|  | %九轮循环 |
|  | for j=1:1:9 |
|  | %字节代换 |
|  | for s\_i=[1:1:4] |
|  | for s\_j=[1:2:8] |
|  | temp(s\_j:s\_j+1,s\_i)=sbox(temp(s\_j:s\_j+1,s\_i)); |
|  | end |
|  | end |
|  | %行移位 |
|  | temp=row\_trans(temp,0); |
|  | %列混合 |
|  | M(1:8)=mix\_col(temp(:,1)); |
|  | M(9:16)=mix\_col(temp(:,2)); |
|  | M(17:24)=mix\_col(temp(:,3)); |
|  | M(25:32)=mix\_col(temp(:,4)); |
|  | %轮密相加 |
|  | temp=add\_key(M,p,j); |
|  | end |
|  | %第十轮 |
|  | %字节代换 |
|  | for s\_i=[1:1:4] |
|  | for s\_j=[1:2:8] |
|  | temp(s\_j:s\_j+1,s\_i)=sbox(temp(s\_j:s\_j+1,s\_i)); |
|  | end |
|  | end |
|  | %行变换 |
|  | temp=row\_trans(temp,0); |
|  | %轮密相加 |
|  | M=[temp(:,1)',temp(:,2)',temp(:,3)',temp(:,4)']; |
|  | temp=add\_key(M,p,10); |
|  |  |
|  | %  ------------第三步：输出------------ |
|  | cipher\_text=[temp(:,1)',temp(:,2)',temp(:,3)',temp(:,4)']; |
|  | end |

### (2) add\_key()

|  |  |
| --- | --- |
|  | function data=add\_key(M,KEYs,num) |
|  | %  --------------------------------------------- |
|  | %   轮密相加子函数 |
|  | %   M:当前轮状态矩阵展成的字符串 |
|  | %   KEYs:扩展密钥矩阵 |
|  | %   nums:轮数 |
|  | %  --------------------------------------------- |
|  | key(1,:)=KEYs(1+num\*4,:); |
|  | key(2,:)=KEYs(2+num\*4,:); |
|  | key(3,:)=KEYs(3+num\*4,:); |
|  | key(4,:)=KEYs(4+num\*4,:); |
|  | %轮密相加 |
|  | temp1=dec2hex(bitxor(hex2dec(M(1:8)),hex2dec(key(1,:)))); |
|  | temp6(:,1)=[zeros(1,8-length(temp1)),temp1]; |
|  | temp2=dec2hex(bitxor(hex2dec(M(9:16)),hex2dec(key(2,:)))); |
|  | temp6(:,2)=[zeros(1,8-length(temp2)),temp2]; |
|  | temp3=dec2hex(bitxor(hex2dec(M(17:24)),hex2dec(key(3,:)))); |
|  | temp6(:,3)=[zeros(1,8-length(temp3)),temp3]; |
|  | temp4=dec2hex(bitxor(hex2dec(M(25:32)),hex2dec(key(4,:)))); |
|  | temp6(:,4)=[zeros(1,8-length(temp4)),temp4]; |
|  | temp6(abs(temp6)==0)='0'; |
|  | data=temp6; |
|  | end |

### (3) g\_function()

|  |  |
| --- | --- |
|  | function W1=g\_function(W,num) |
|  | %  --------------------------------------------- |
|  | %   密钥扩展中的g函数 |
|  | %   W:输入的字，长度8字节 |
|  | %   nums:轮数 |
|  | %  --------------------------------------------- |
|  | RCON=['01000000';'02000000';'04000000';'08000000';'10000000';'20000000';  '40000000';'80000000';'1b000000';'36000000']; |
|  | B0=sbox(W(1:2)); |
|  | B1=sbox(W(3:4)); |
|  | B2=sbox(W(5:6)); |
|  | B3=sbox(W(7:8)); |
|  | W1=strcat(B1,B2,B3,B0); |
|  | W1=bitxor(hex2dec(RCON(num,:)),hex2dec(W1)); |
|  | end |

### (4) mix\_col()

|  |  |
| --- | --- |
|  | function y=mix\_col(s) |
|  | %  --------------------------------------------- |
|  | %   列混淆子函数 |
|  | %   s:所需操作的列，长度4字节 |
|  | %  --------------------------------------------- |
|  | s=s'; |
|  | s0=s(1:2); |
|  | s1=s(3:4); |
|  | s2=s(5:6); |
|  | s3=s(7:8); |
|  | %只需实现乘2子函数，再进行拆分即可 |
|  | a=dec2hex(bitxor(bitxor(bitxor(bitxor(hex2dec(xtime(s0)),hex2dec(xtime(s1)  )),hex2dec(s1)),hex2dec(s2)),hex2dec(s3))); |
|  | b=[zeros(1,2-length(a)),a]; |
|  | b(abs(b)==0)='0'; |
|  | y(1:2)=b; |
|  | a=dec2hex(bitxor(bitxor(bitxor(bitxor(hex2dec(s0),hex2dec(xtime(s1))),hex2  dec(xtime(s2))),hex2dec(s2)),hex2dec(s3))); |
|  | b=[zeros(1,2-length(a)),a]; |
|  | b(abs(b)==0)='0'; |
|  | y(3:4)=b; |
|  | a=dec2hex(bitxor(bitxor(bitxor(bitxor(hex2dec(s0),hex2dec(s1)),hex2dec(xti  me(s2))),hex2dec(xtime(s3))),hex2dec(s3))); |
|  | b=[zeros(1,2-length(a)),a]; |
|  | b(abs(b)==0)='0'; |
|  | y(5:6)=b; |
|  | a=dec2hex(bitxor(bitxor(bitxor(bitxor(hex2dec(xtime(s0)),hex2dec(s0)),hex2  dec(s1)),hex2dec(s2)),hex2dec(xtime(s3)))); |
|  | b=[zeros(1,2-length(a)),a]; |
|  | b(abs(b)==0)='0'; |
|  | y(7:8)=b; |
|  | end |

### (5) row\_trans()

|  |  |
| --- | --- |
|  | function S=row\_trans(temp,flag) |
|  | %  --------------------------------------------- |
|  | %   行移位子函数 |
|  | %   temp:某轮的状态矩阵 |
|  | %   flag:'1'代表右移，'0'代表左移 |
|  | %  --------------------------------------------- |
|  | for i=[1:2:8] |
|  | k=(i-1)/2; |
|  | for j=[0:1:1] |
|  | str=temp(i+j,:); |
|  | if(flag) |
|  | str=[str(end-k+1:end),str(1:end-k)]; |
|  | else |
|  | str=[str(k+1:end),str(1:k)]; |
|  | end |
|  | S(i+j,:)=str; |
|  | end |
|  | end |
|  | end |

### (6) sbox()

|  |  |
| --- | --- |
|  | function D=sbox(data) |
|  | %  --------------------------------------------- |
|  | %   字节代换子函数 |
|  | %   data:输入待替换数据，长度2字节 |
|  | %  --------------------------------------------- |
|  | S=[ 0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76; |
|  | 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0; |
|  | 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15; |
|  | 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75; |
|  | 0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84; |
|  | 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf; |
|  | 0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8; |
|  | 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2; |
|  | 0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73; |
|  | 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb; |
|  | 0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79; |
|  | 0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08; |
|  | 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a; |
|  | 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e; |
|  | 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf; |
|  | 0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16]; |
|  | D=['00',dec2hex(S(hex2dec(data(1))+1,hex2dec(data(2))+1))]; |
|  | D=D(end-1:end); |
|  | end |

### (7) xtime()

|  |  |
| --- | --- |
|  | function y=xtime(b) |
|  | %  --------------------------------------------- |
|  | %   GF(2^8)上的乘二子函数 |
|  | %   b:待操作的字节 |
|  | %  --------------------------------------------- |
|  | a=dec2bin(hex2dec(b)); |
|  | c=[zeros(1,8-length(a)),a]; |
|  | c(abs(c)==0)='0'; |
|  | switch c(1) |
|  | case  '0' |
|  | temp='00'; |
|  | case '1' |
|  | temp='FF'; |
|  | end |
|  |  |
|  | d=dec2hex(bitxor(bin2dec(strcat(c(2:8),'0')),bitand(hex2dec('1B'),hex2dec(  temp)))); |
|  | b=[zeros(1,2-length(d)),d]; |
|  | b(abs(b)==0)='0'; |
|  | y=b; |
|  | end |

### 【实验过程】

从AES()函数可以比较清晰的看出AES加密整个流程的大体原理，其中列混淆部分的原理比较复杂，其他部分都较为简明，下对列混淆实现作简要分析。

列混淆实质上是定义在上的矩阵运算，但是状态矩阵中的第列的列混合可以表示为下式：

这一运算是定义在的运算，加减可用异或进行，乘法则可作拆分：

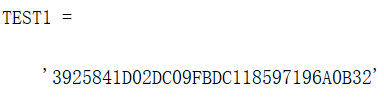
方便起见，这里我们仅实现上的乘2算法，公式如下：

该部分代码的实现在mix\_col()和xtime()中。

最后，我们对代码进行测试：

TEST1=AES('3243F6A8885A308D313198A2E0370734','2B7E151628AED2A6ABF7158809CF4F3C')

得到如下结果：



下用密码学工具cryptool进行验证，可见该结果是正确的：

