



# AES 算法实现

山东大学

网络安全学院 2021 级第 11 组

2023 年 3 月 26 日

## 目录

AES 算法实现 .....	1
一、 组员分工与环境配置 .....	3
(1) 组员分工 .....	3
(2) 环境配置 .....	3
二、 AES 算法的实现 .....	3
(1) 实验要求 .....	3
(2) 实验思路 .....	3
(3) 实验步骤 .....	3
(4) 实验结果 .....	6
(5) 全部代码 .....	7
(6) 实验心得 .....	12
三、 算法加密和密码库加密的时间比较 .....	12
(1) 实验要求 .....	12
(2) 实验思路 .....	12
(3) 实验步骤 .....	13
(4) 实验结果 .....	15
(5) 实验心得 .....	17

## 一、组员分工与环境配置

### (1) 组员分工

单灵 202100460039: 实现 AES-128 加密算法, 并设明文和密钥均为学号

张铭珊 202100460111: 实现 AES-128 加密算法, 并设明文和密钥均为学号

李佳琪 202100460158: 同一个密钥分别加密 100 组明文集合, 比较算法和密码库调用时间

宾芸萱 202100460131: 同一个密钥分别加密 100 组明文集合, 比较算法和密码库调用时间

### (2) 环境配置

Operating system version: WIN 10

CPU instruction set: X64

Software: Visual Studio 2019, Python 3.11

## 二、AES 算法的实现

### (1) 实验要求

实现 AES-128 加密算法, 并设明文和密钥均为学号(按 ASCII 码表示, 每位数字对应 8-bit, 不满 128-bit 的在最后填充足够多的 0, 明文最左侧为最低位), 求对应的密文, 依次加密小组成员的学号。

### (2) 实验思路

我们对于本实验的设想如下: 因为 AES 算法条理明晰, 于是我们准备先将各步骤的函数先实践完成, 然后在最后的加密过程中对它们进行调用即可。

### (3) 实验步骤

1、初始化 S 盒、实现轮密钥的列表矩阵以及矩阵乘的矩阵

2、写 S 盒代换函数:

这个函数比较简单, 我们的思路是传入一个列表后对于列表中的每一个元素取其 i、j 值, 即这个数字的高四位和低四位, 然后传入 S 盒中进行查找, 并将查找到的值返回。

局部代码如下:

```
#S盒做替换
def subBytes(text, matrix):
    return [matrix[i][j] for i, j in [(t//0xf, t%0xf) for t in text]]
```

### 3、写行代换函数：

因为每次传入的列表数目固定，所以我们直接写了代换之后的返回值。局部代码如下：

```
#行代换
def shiftRows(text):
    return [text[0], text[1], text[2], text[3],
            text[5], text[6], text[7], text[4],
            text[10], text[11], text[8], text[9],
            text[15], text[12], text[13], text[14]]
```

### 4、列混淆函数

对于列混淆函数，我们一开始觉得没有头绪，后来决定从它的原理出发，即编写函数实现  $gf_2^8$  有限域上的乘法。由于我们编写代码的能力不足，所以我们决定将其逐步拆分成很小的步骤，先实现单个元素之间的乘法，只关乎多项式之间指数相乘，再进行行列之间相乘出的多项式进行相加和取模（指数列表为[0,1,3,4,8]），最后调用这些小的函数进行两个矩之的列混淆。局部代码如下：

```
def record_index(b):
    index_list=[]
    count=0
    while b!=0:
        if b&0x01==1:#b每一位对1进行位运算 若位次为1则记录下对应指数 如1111 1001记录为[0, 3, 4, 5, 6, 7]
            index_list.append(count)
            count+=1
            b=b>>1
    return index_list
#行上的一个元素与列上的对应元素相乘，去掉多于元素并将大于8的指数去掉
def single_multy(hangyuansu, lieyuansu):
    list1=record_index(hangyuansu)
    list2=record_index(lieyuansu)
    list3=[]
    for i in list1:
        for j in list2:
            list3.append(i*j)
    list4=list(set(list3))#去重复元素
    for l in list4:
        if l>8:
            list4.remove(l)
    return list4
#每行每列逐项相乘再相加最后进行模运算 将列表作为索引值赋予0000 0000 返回后即是列混淆后对应空位上的元素
#可以发现这个域上的多项式的模运算就是两个集合的并集减去它们的交集 即对称差集
#xiecuel..... caonm
def hang_lie_multy(hang, lie):
    list1=[]
    for i in hang:
        for j in lie:
            list1.append(single_multy(i, j))
    list2=low_den(list1)#降维
    stage=set(list2)#去重复元素并作为集合运算
    mo_n={0, 1, 3, 4, 8}
    last_stage=stage|mo_n-stage&mo_n
    one_last=list(last_stage)
    for i in one_last:
        if i==8:
            one_last.remove(i)#模8的时候是出现了8的超出有限域的时候
    num=0
    for i in one_last:
        num+=2**i
    return num
def mixColumns(liehunhe_matrix, text):
    text_state=[]
    for i in range(4):
        text_state.append(text[i::4])
    return_list=[]#应该返回的列表
    num1=0
    num2=0
    list1=[]
    list2=[]
    for i in liehunhe_matrix:
        list1=liehunhe_matrix[num1:len(liehunhe_matrix)]
        num1+=1
        num2=0#恢复初始值 重新计数
        for j in text_state:
            list2=text_state[num2:len(text_state)]
            t=hang_lie_multy(low_den(list1), low_den(list2))
            return_list.append(t)
            num2+=1
    return return_list
```

## 5、轮密钥加函数

对于密钥来说，在每次迭代中要进行新的生成。为了方便，我们先把 44 列的密钥都生成出来，每次加密时根据轮数选取密钥。对于生成密钥数列，我们先定义列表间对应元素彼此异或的函数 yihuo，便于接下来的写作。然后定义 T 函数，它写起来较为简单。将初始的四组密钥按顺序放在 one\_list 空列表中，然后进行密钥的生成。局部代码如下：

```
#轮密钥加 每一列有4个16进制数字 初始一共有4轮 在迭代过程中使得最终能够有 44组别
#异或函数 两列逐项异或 t表示这个一维数组有多少元素
def yihuo(list1, list2, t):
    list3=[]
    for i in list1:
        for j in list2:
            list3.append(i^j)
    list4=list3[::t]
    return list4

def T(list1, matrix, R_list):#matrix就是S盒 R_list是轮常量.
    #1字节向左移动一个位置
    list2=[list1[1], list1[2], list1[3], list1[0]]
    #2对于list2进行S盒代换
    list2=subBytes(list2, S_BOX)
    return list2
#在W[i]中i%4==0时传入W[i-1]、以及S盒还有R[j]. (j表示轮数)
#i%4==0时 W[i]=W[i-4]^T(W[i-1])
#i%4!=0时 W[i]=W[i-4]^W[i-1]

#制造44列，每次返回四轮次作为密钥
#one_list[]是存储44列密钥的列表 每个元素就是一列
one_list=[]
for i in range(4):
    one_list.append(K_list[i::4])

i=4
b=4
count=0
while(i>=4 and i<44):
    list1=one_list[i-4]
    list2=one_list[i-1]
    if(i%4!=0):
        one_list.append(yihuo(list1, list2, 4))
    else:
        one_list.append(yihuo(list1, T(list2, S_BOX, R_list[b-4]), 4))
    count+=1
    i+=1
    if(count%4==0):
        b+=1 #每生成一周期 R_list发生改变

#print(one_list)
```

## 6、调用以上函数并进行AES加密

局部代码如下：



```

P_list=[0x31, 0x09, 0x07, 0x23, 0x01, 0x01, 0x34, 0x36, 0x30, 0x10, 0x19, 0x39, 0x00, 0x00, 0x00, 0x00]
lunshu=0
print("明文是：", P_list)
print("密钥初始值为", K_list)
#1 初始值进行第一次加密
Encry_text=yihuo(P_list, K_list, 16)

#2 开始加密前9轮
for k in range(9):
    Encry_text=subBytes(Encry_text, S_BOX)
    print(Encry_text)
    Encry_text=shiftRows(Encry_text)
    print(Encry_text)
    Encry_text=mixColumns(MIX_C, Encry_text)
    print(Encry_text)
    lunshu+=1
    for i in range(4):
        K_list.append(one_list[lunshu*4+i])
    K_list=K_list[16::]
    K_list=low_den(K_list)
    print("K_list是：")
    print(K_list)
    Encry_text=yihuo(Encry_text, K_list, 16)

#3开始加密第十轮
Encry_text=subBytes(Encry_text, S_BOX)
Encry_text=shiftRows(Encry_text)
lunshu=10
for i in range(4):
    K_list.append(one_list[lunshu*4+i])
K_list=K_list[16::]
K_list=low_den(K_list)
print("K_list是：")
print(K_list)
Encry_text=yihuo(Encry_text, K_list, 16)
print(Encry_text)

```

#### (4)实验结果

```

明文是： [50, 48, 50, 49, 48, 48, 52, 54, 48, 48, 51, 57, 0, 0, 0, 0]
密钥初始值为 [50, 48, 50, 49, 48, 48, 52, 54, 48, 49, 49, 49, 0, 0, 0, 0]
K_list是：
[32, 34, 34, 18, 16, 16, 17, 32, 34, 36, 33, 16, 19, 20, 19, 34]
K_list是：
[121, 123, 123, 75, 105, 105, 104, 89, 75, 77, 72, 121, 88, 95, 88, 105]
K_list是：
[52, 54, 54, 6, 93, 93, 92, 109, 22, 16, 21, 36, 78, 73, 78, 127]
K_list是：
[215, 213, 213, 229, 138, 138, 139, 186, 156, 154, 159, 174, 210, 213, 210, 227]
K_list是：
[198, 196, 196, 244, 76, 76, 77, 124, 208, 214, 211, 226, 2, 5, 2, 51]
K_list是：
[173, 175, 175, 159, 225, 225, 224, 209, 49, 55, 50, 3, 51, 52, 51, 2]
K_list是：
[55, 53, 53, 5, 214, 214, 215, 230, 231, 225, 228, 213, 212, 211, 212, 229]
K_list是：
[207, 205, 205, 253, 25, 25, 24, 41, 254, 248, 253, 204, 42, 45, 42, 27]
K_list是：
[203, 201, 201, 249, 210, 210, 211, 226, 44, 42, 47, 30, 6, 1, 6, 55]
K_list是：
[183, 181, 181, 133, 101, 101, 100, 85, 73, 79, 74, 123, 79, 72, 79, 126]
密文是 [116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116, 116]

```

可以看出，加密结果应该是错误的。对于这个错误事件，我们逐步调试，最后发现在第一轮

的函数中，明文在 9 轮迭代中，经过 S 盒代换与行代换后均输出正常，但在第一次进行列混淆后，所有的密文值均变成相等，这样以后每轮次都是相等的了，于是我们推测是列混淆函

数写作出现问题，但单独调用列混淆函数时，对于数字差异较大的数组，其输出值并不相等，如下：

```
[127, 191, 255, 251, 127, 191, 255, 251, 127, 191, 255, 251, 127, 191, 255, 251]
```

但我们并没有找出列混淆函数哪里写得有问题。故本次实验做得并不成功，在实验报告上交后我们仍将尝试发现问题。

### (5)全部代码

代码与思路注释如下：

#S 盒

```
S_BOX = [[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]]
```

#矩阵乘的矩阵

```
MIX_C = [[0x2, 0x3, 0x1, 0x1],
          [0x1, 0x2, 0x3, 0x1],
          [0x1, 0x1, 0x2, 0x3],
          [0x3, 0x1, 0x1, 0x2]]
```

#轮密钥加的轮密钥

```
R_list=[[0x01,0x00,0x00,0x00],[0x02,0x00,0x00,0x00],[0x04,0x00,0x00,0x00],[0x08,0x00,0x00,0x00],
[0x01,0x00,0x00,0x00],[0x02,0x00,0x00,0x00],[0x04,0x00,0x00,0x00],[0x08,0x00,0x00,0x00],
[0x1B,0x00,0x00,0x00],[0x36,0x00,0x00,0x00]]
K_list=[0x32,0x30,0x32,0x31,0x30,0x30,0x34,0x36,0x30,0x30,0x33,0x39,0x00,0x00,0x00,0x00]
P_list=[0x32,0x30,0x32,0x31,0x30,0x30,0x34,0x36,0x30,0x30,0x33,0x39,0x00,0x00,0x00,0x00]
```

#降维函数

```
def low_den(twoden_list):
    list1=[]
    for i in twoden_list:
        for j in i:
            list1.append(j)
    return list1
```

#S 盒做替换

```
def subBytes(text,matrix):
    return [matrix[i][j] for i,j in [(t//0xf,t%0xf) for t in text]]
```

#行代换

```
def shiftRows(text):
    return [text[0],text[1],text[2],text[3],
            text[5],text[6],text[7],text[4],
            text[10],text[11],text[8],text[9],
            text[15],text[12],text[13],text[14]]
```

#列混淆

#1、 $gf_{2^8}$  有限域乘法 模  $:x^8+x^4+x^3+x+1$

'''

作为多项式的指数相乘，多的系数取为 1，然后模 8 4 3 1 0 最后求得的多项式指数记载在八位二进制中，就是列混淆中[i,j]的结果

'''

```
def record_index(b):
    index_list=[]
    count=0
    while b!=0:
        if b&0x01==1:#b 每一位对 1 进行位运算 若位次为 1 则记录下对应指数 如 1111
            1001 记录为[0,3,4,5,6,7]
```



```

        index_list.append(count)
    count+=1
    b=b>>1
    return index_list
#行上的一个元素与列上的对应元素相乘，去掉多于元素并将大于 8 的指数去掉
def single_multy(hangyuansu,lieyuansu):
    list1=record_index(hangyuansu)
    list2=record_index(lieyuansu)
    list3=[]
    for i in list1:
        for j in list2:
            list3.append(i*j)
    list4=list(set(list3))#去重复元素
    for l in list4:
        if l>8:
            list4.remove(l)
    return list4
#每行每列逐项相乘再相加最后进行模运算 将列表作为索引值赋予 0000 0000 返回后即是
列混淆后对应空位上的元素
#可以发现这个域上的多项式的模运算就是两个集合的并集减去它们的交集 即对称差集
#xiecuol..... caonm
def hang_lie_multy(hang,lie):
    list1=[]
    for i in hang:
        for j in lie:
            list1.append(single_multy(i,j))
    list2=low_den(list1)#降维
    stage=set(list2)#去重复元素并作为集合运算
    mo_n={0,1,3,4,8}
    last_stage=stage|mo_n-stage&mo_n
    one_last=list(last_stage)
    for i in one_last:
        if i==8:
            one_last.remove(i)#模 8 的时候是出现了 8 的超出有限域的时候
    num=0
    for i in one_last:
        num+=2**i
    return num
def mixColumns(liehunhe_matrix,text):
    text_state=[]
    for i in range(4):
        text_state.append(text[i::4])
    return_list=[]#应该返回的列表
    num1=0

```

```

num2=0
list1=[]
list2=[]
for i in liehunhe_matrix:
    list1=liehunhe_matrix[num1::len(liehunhe_matrix)]
    num1+=1
    num2=0#恢复初始值 重新计数
    for j in text_state:
        list2=text_state[num2::len(text_state)]
        t=hang_lie_multy(low_den(list1),low_den(list2))
        return_list.append(t)
        num2+=1
    return return_list

#轮密钥加 每一列有 4 个 16 进制数字 初始一共有 4 轮 在迭代过程中使得最终能够有 44
组别
#异或函数 两列逐项异或 t 表示这个一维数组有多少元素
def yihuo(list1,list2,t):
    list3=[]
    for i in list1:
        for j in list2:
            list3.append(i^j)
    list4=list3[::t]
    return list4

def T(list1,matrix,R_list):#matrix 就是 S 盒 R_list 是轮常量.
    #1 字节向左移动一个位置
    list2=[list1[1],list1[2],list1[3],list1[0]]
    #2 对于 list2 进行 S 盒代换
    list2=subBytes(list2,S_BOX)
    return list2
#在 W[i]中 i%4==0 时传入 W[i-1]、以及 S 盒还有 R[j].(j 表示轮数)
#i%4==0 时 W[i]=W[i-4]^T(W[i-1])
#i%4!=0 时 W[i]=W[i-4]^W[i-1]

#制造 44 列，每次返回四轮次作为密钥
#one_list[]是存储 44 列密钥的列表 每个元素就是一列
one_list=[]
for i in range(4):
    one_list.append(K_list[i::4])

i=4

```

```

b=4
count=0
while(i>=4 and i<44):
    list1=one_list[i-4]
    list2=one_list[i-1]
    if(i%4!=0):
        one_list.append(yihuo(list1,list2,4))
    else:
        one_list.append(yihuo(list1,T(list2,S_BOX,R_list[b-4]),4))
    count+=1
    i+=1
    if(count%4==0):
        b+=1 #每生成一周期 R_list 发生改变

```

```
#print(one_list)
```

```
#传入轮数
```

```
'''
```

```

- 128 位密钥需要加密 10 轮
- 在第一轮加密之前之前需要进行一次轮密钥加 addRoundKey
- 前 9 轮循环执行以下操作：
- subBytes
- shiftRows
- mixColumns
- addRoundKey
- 最后一轮(即第 10 轮没有列混淆):
- subBytes
- shiftRows
- addRoundKey
'''

```

```
lunshu=0
```

```
print("明文是：",P_list)
```

```
print("密钥初始值为",K_list)
```

```
#1 初始值进行第一次加密
```

```
Encry_text=yihuo(P_list,K_list,16)
```

```
#2 开始加密前 9 轮
```

```
for k in range(9):
```

```
    Encry_text=subBytes(Encry_text,S_BOX)
```

```
    Encry_text=shiftRows(Encry_text)
```

```
    Encry_text=mixColumns(MIX_C,Encry_text)
```

```
    lunshu+=1
```

```

    for i in range(4):
        K_list.append(one_list[lunshu*4+i])
    K_list=K_list[16::]
    K_list=low_den(K_list)
    print("K_list 是: ")
    print(K_list)
    Encry_text=yihuo(Encry_text,K_list,16)

#3 开始加密第十轮
Encry_text=subBytes(Encry_text,S_BOX)
Encry_text=shiftRows(Encry_text)
lunshu=10
for i in range(4):
    K_list.append(one_list[lunshu*4+i])
K_list=K_list[16::]
K_list=low_den(K_list)
print("K_list 是: ")
print(K_list)
Encry_text=yihuo(Encry_text,K_list,16)
print('密文是: ',Encry_text)

```

## (6)实验心得

本次实验的失败令我们决心更好地学习密码学。这次实验令我们深刻地意识到自己编程水平的不足，以至于我们逻辑上的想法和实践需要通过编写一些非常繁琐、甚至重复的小逻辑代码块才能实现，感到惭愧，我们决心要好好提高我们编程的写作能力。

## 三、算法加密和密码库加密的时间比较

### (1) 实验要求

对同一个密钥，分别加密 100 组满足以下结构的 256 个明文组成的集合（提前生成好），估算算法加密一次的运行时间，并与直接调用算法库进行加密的运行时间进行比较。256 个明文满足：第 0 字节遍历 256 种可能，其余字节取任意常数。

### (2) 实验思路

首先设计函数利用循环结构生成符合题目条件的 256 个明文，第 0 字节遍历 256 种可能，对于其余 15 个字节利用 rand 函数进行随机数生成，利用数组收集 100 组这样的明文组成的集合。然后编写 AES-128 算法加密对 100 组明文进行加密并输出运行时间。由于直接调用密码库进行加密每次只能够加密一组 256 个明文组成的集合，在此次实验中选择加密十组取运行时间平均值作为数据参考进行比较。

### (3) 实验步骤

#### 1、生成 100 组 256 个明文组成的集合

因为 256 个明文满足的结构对于第 0 字节有要求，则生成过程中采取直接生成字节的形式，而不是按字符输入。设置 unsigned int cc 作为“中间人”接收随机形成的 256 个单字节明文。

```
for (int p = 0; p < 100; p++) {  
    cout << "输入的明文为:" << endl;  
    for (int b = 0; b < 16 * 256; b++) {  
        char c = b % 16;  
        unsigned int d;  
        d = b / 16;  
        unsigned int cc;  
        if (c == 0) {  
            P[b] = d;  
        }  
        if (c > 0 && c < 16) {  
            cc = rand() % 256;  
            P[b] = cc;  
        }  
    }  
}
```

#### 2、设计 AES-128 加密算法对明文进行加密

*PS. 在此仅放出头文件函数定义，详细代码见附件*

```
#ifndef _AES_H_  
#define _AES_H_  
  
// S盒  
extern unsigned char S[256];  
  
// AES-128轮常量  
static const unsigned int rcon[10] = {  
    0x01000000UL, 0x02000000UL, 0x04000000UL, 0x08000000UL, 0x10000000UL,  
    0x20000000UL, 0x40000000UL, 0x80000000UL, 0x1B000000UL, 0x36000000UL  
};  
  
//列混淆时用到的正矩阵  
extern unsigned char positive_matrix[4][4];  
  
//密钥扩展  
extern unsigned int W[44];  
  
//可输入明文的最大长度  
static const int MAX_LENGTH = 1e6;  
  
//明文  
extern unsigned char P[MAX_LENGTH];  
  
//分组后的128明文  
extern unsigned char P128[16];
```

```

//密文
extern unsigned char C[MAX_LENGTH];
//分组后的128密文
extern unsigned char C128[16];

//将128明文转换为状态矩阵
void array_to_mat(unsigned char p[], unsigned char state_mat[][4]);

//将状态矩阵转换为128密文
void mat_to_array(unsigned char state_mat[][4], unsigned char c[]);

//将1个32位的密钥，转换为4个8位密钥
void key32_to_key8(unsigned int key32, unsigned char* key8);

//将4个8位的密钥，转换为1个32位密钥
unsigned int key8_to_key32(unsigned char* key8);

//字节替换
unsigned char SubBytes(unsigned char input);

//行位移
void ShiftRows(unsigned char state_mat[][4]);

//有限域上的乘法
unsigned char multi_finite_field(unsigned char a, unsigned char b);

//列混合
void MixColumns(unsigned char state_mat[][4]);

//轮密钥加，cnt标记这是第几轮循环
void AddRoundKey(unsigned char state_mat[][4], int cnt);

//密钥扩展时的T函数，cnt代表轮数
unsigned int T(unsigned int input, int cnt);

//密钥扩展函数
void KeyExpansion(unsigned char* init_key);

//加密
void encryption();
#endif

```

### 3、调用密码库对明文进行加密

PS. 在此仅放出修改部分，详细代码见附件

```

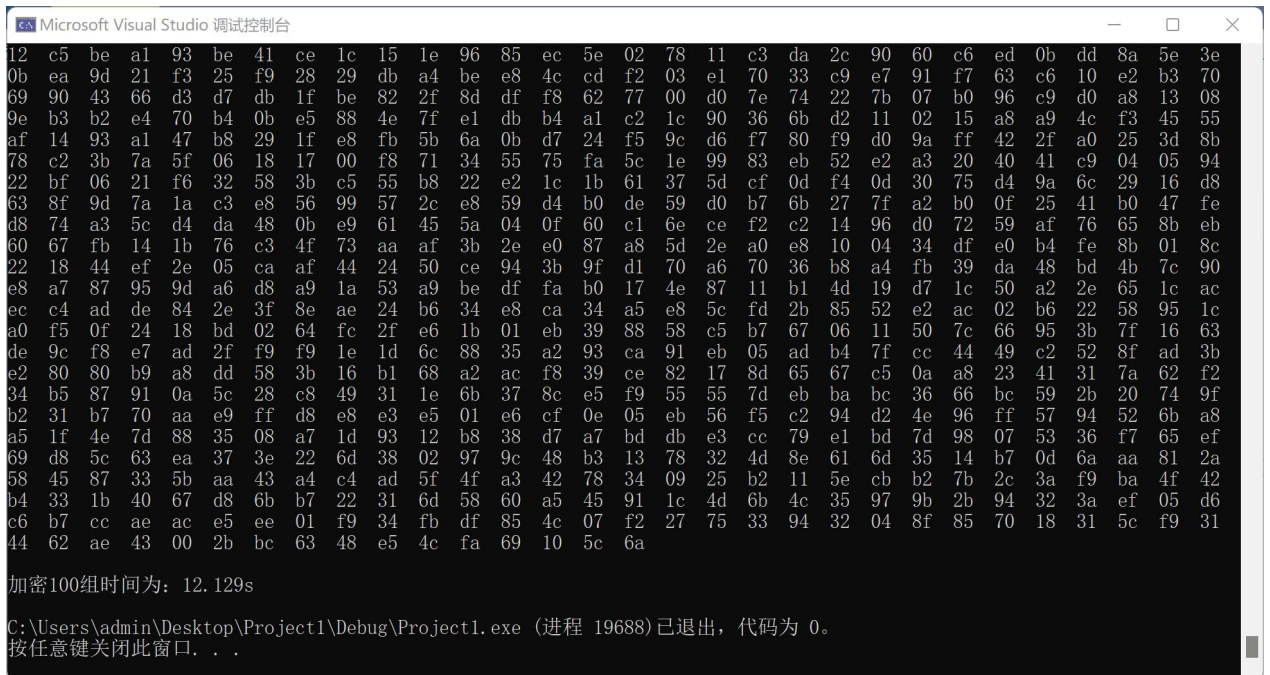
    end = clock();
    printf("\r\nsource context: %s%02X\r\n", input);
    dump_buf((uint8_t*)input, 256 * 16);
    printf("cipher name: %s block size is: %d\r\n", mbedtls_cipher_get_nam
    dump_buf(output_buf, olen);
    printf("The time is %f s", (double)(end - begin));

```



#### (4) 实验结果

##### 1、AES-128 加密算法运行时间输出



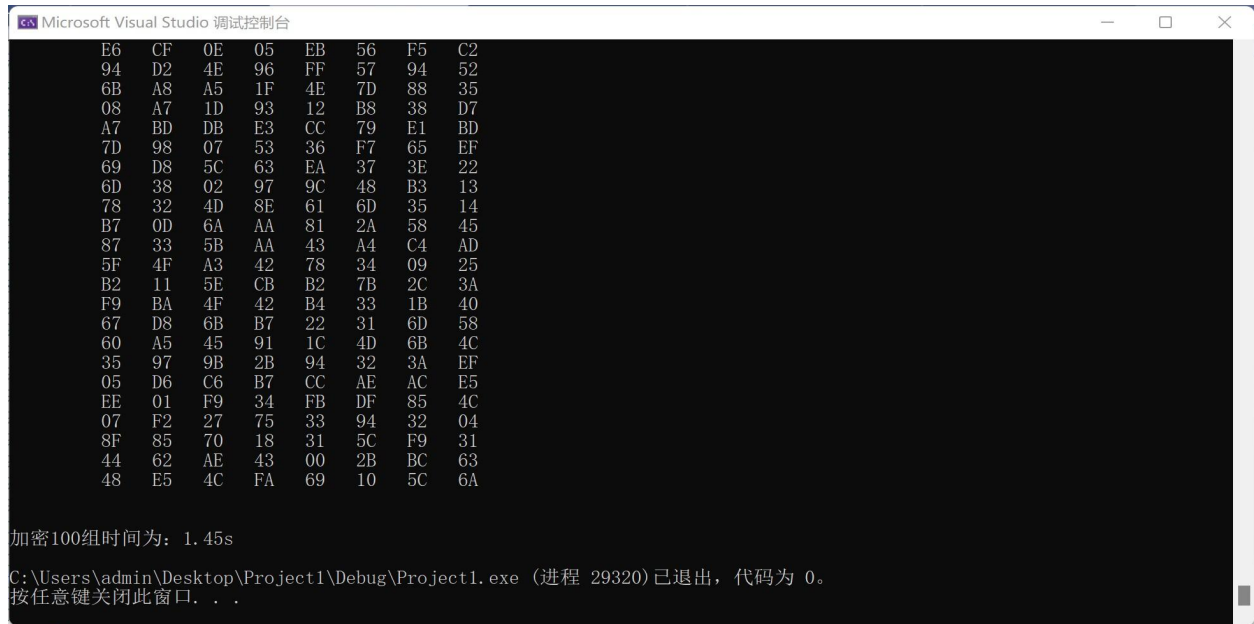
```
Microsoft Visual Studio 调试控制台

12 c5 be a1 93 be 41 ce 1c 15 1e 96 85 ec 5e 02 78 11 c3 da 2c 90 60 c6 ed 0b dd 8a 5e 3e
0b ea 9d 21 f3 25 f9 28 29 db a4 be e8 4c cd f2 03 e1 70 33 c9 e7 91 f7 63 c6 10 e2 b3 70
69 90 43 66 d3 d7 db 1f be 82 2f 8d df f8 62 77 00 d0 7e 74 22 7b 07 b0 96 c9 d0 a8 13 08
9e b3 b2 e4 70 b4 0b e5 88 4e 7f e1 db b4 a1 c2 1c 90 36 6b d2 11 02 15 a8 a9 4c f3 45 55
af 14 93 a1 47 b8 29 1f e8 fb 5b 6a 0b d7 24 f5 9c d6 f7 80 f9 d0 9a ff 42 2f a0 25 3d 8b
78 c2 3b 7a 5f 06 18 17 00 f8 71 34 55 75 fa 5c 1e 99 83 eb 52 e2 a3 20 40 41 c9 04 05 94
22 bf 06 21 f6 32 58 3b c5 55 b8 22 e2 1c 1b 61 37 5d cf 0d f4 0d 30 75 d4 9a 6c 29 16 d8
63 8f 9d 7a 1a c3 e8 56 99 57 2c e8 59 d4 b0 de 59 d0 b7 6b 27 7f a2 b0 0f 25 41 b0 47 fe
d8 74 a3 5c d4 da 48 0b e9 61 45 5a 04 0f 60 c1 6e ce f2 c2 14 96 d0 72 59 af 76 65 8b eb
60 67 fb 14 1b 76 c3 4f 73 aa af 3b 2e e0 87 a8 5d 2e a0 e8 10 04 34 df e0 b4 fe 8b 01 8c
22 18 44 ef 2e 05 ca af 44 24 50 ce 94 3b 9f d1 70 a6 70 36 b8 a4 fb 39 da 48 bd 4b 7c 90
e8 a7 87 95 9d a6 d8 a9 1a 53 a9 be df fa b0 17 4e 87 11 b1 4d 19 d7 1c 50 a2 2e 65 1c ac
ec c4 ad de 84 2e 3f 8e ae 24 b6 34 e8 ca 34 a5 e8 5c fd 2b 85 52 e2 ac 02 b6 22 58 95 1c
a0 f5 0f 24 18 bd 02 64 fc 2f e6 1b 01 eb 39 88 58 c5 b7 67 06 11 50 7c 66 95 3b 7f 16 63
de 9c f8 e7 ad 2f f9 f9 1e 1d 6c 88 35 a2 93 ca 91 eb 05 ad b4 7f cc 44 49 c2 52 8f ad 3b
e2 80 80 b9 a8 dd 58 3b 16 b1 68 a2 ac f8 39 ce 82 17 8d 65 67 c5 0a a8 23 41 31 7a 62 f2
34 b5 87 91 0a 5c 28 c8 49 31 1e 6b 37 8c e5 f9 55 55 7d eb ba bc 36 66 bc 59 2b 20 74 9f
b2 31 b7 70 aa e9 ff d8 e8 e3 e5 01 e6 cf 0e 05 eb 56 f5 c2 94 d2 4e 96 ff 57 94 52 6b a8
a5 1f 4e 7d 88 35 08 a7 1d 93 12 b8 38 d7 a7 bd db e3 cc 79 e1 bd 7d 98 07 53 36 f7 65 ef
69 d8 5c 63 ea 37 3e 22 6d 38 02 97 9c 48 b3 13 78 32 4d 8e 61 6d 35 14 b7 0d 6a aa 81 2a
58 45 87 33 5b aa 43 a4 c4 ad 5f 4f a3 42 78 34 09 25 b2 11 5e cb b2 7b 2c 3a f9 ba 4f 42
b4 33 1b 40 67 d8 6b b7 22 31 6d 58 60 a5 45 91 1c 4d 6b 4c 35 97 9b 2b 94 32 3a ef 05 d6
c6 b7 cc ae ac e5 ee 01 f9 34 fb df 85 4c 07 f2 27 75 33 94 32 04 8f 85 70 18 31 5c f9 31
44 62 ae 43 00 2b bc 63 48 e5 4c fa 69 10 5c 6a

加密100组时间为: 12.129s

C:\Users\admin\Desktop\Project1\Debug\Project1.exe (进程 19688) 已退出, 代码为 0。
按任意键关闭此窗口. . .
```

(图 1: 有输出密文的算法加密时间)



```
Microsoft Visual Studio 调试控制台

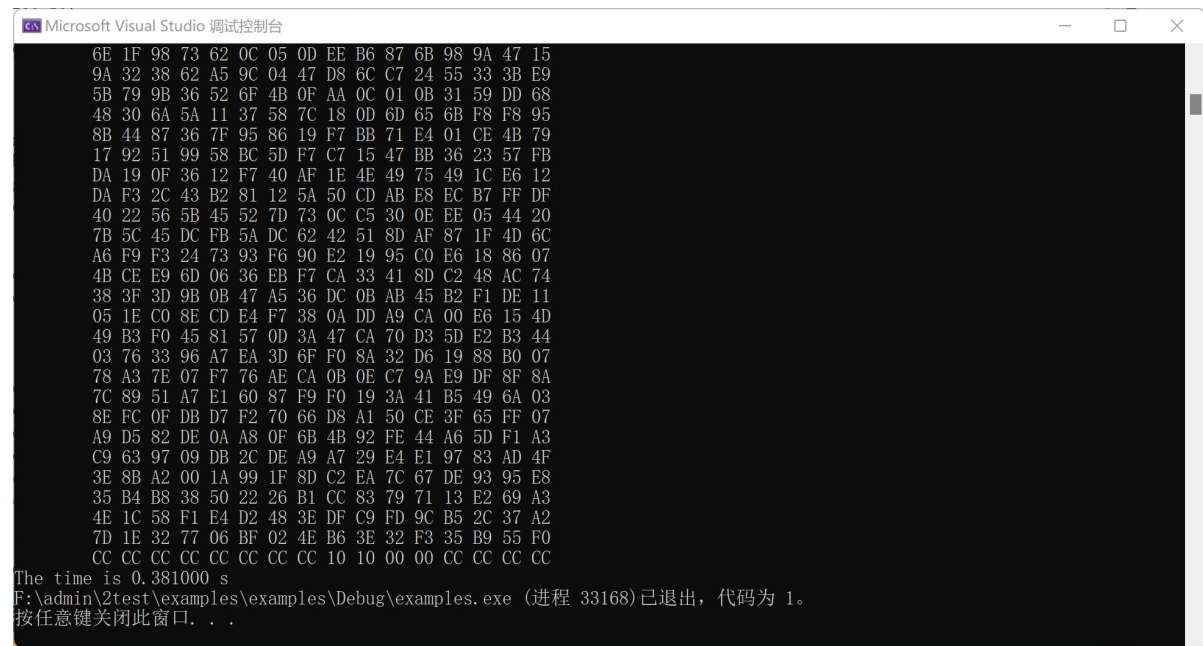
E6 CF 0E 05 EB 56 F5 C2
94 D2 4E 96 FF 57 94 52
6B A8 A5 1F 4E 7D 88 35
08 A7 1D 93 12 B8 38 D7
A7 BD DB E3 CC 79 E1 BD
7D 98 07 53 36 F7 65 EF
69 D8 5C 63 EA 37 3E 22
6D 38 02 97 9C 48 B3 13
78 32 4D 8E 61 6D 35 14
B7 0D 6A AA 81 2A 58 45
87 33 5B AA 43 A4 C4 AD
5F 4F A3 42 78 34 09 25
B2 11 5E CB B2 7B 2C 3A
F9 BA 4F 42 B4 33 1B 40
67 D8 6B B7 22 31 6D 58
60 A5 45 91 1C 4D 6B 4C
35 97 9B 2B 94 32 3A EF
05 D6 C6 B7 CC AE AC E5
EE 01 F9 34 FB DF 85 4C
07 F2 27 75 33 94 32 04
8F 85 70 18 31 5C F9 31
44 62 AE 43 00 2B BC 63
48 E5 4C FA 69 10 5C 6A

加密100组时间为: 1.45s

C:\Users\admin\Desktop\Project1\Debug\Project1.exe (进程 29320) 已退出, 代码为 0。
按任意键关闭此窗口. . .
```

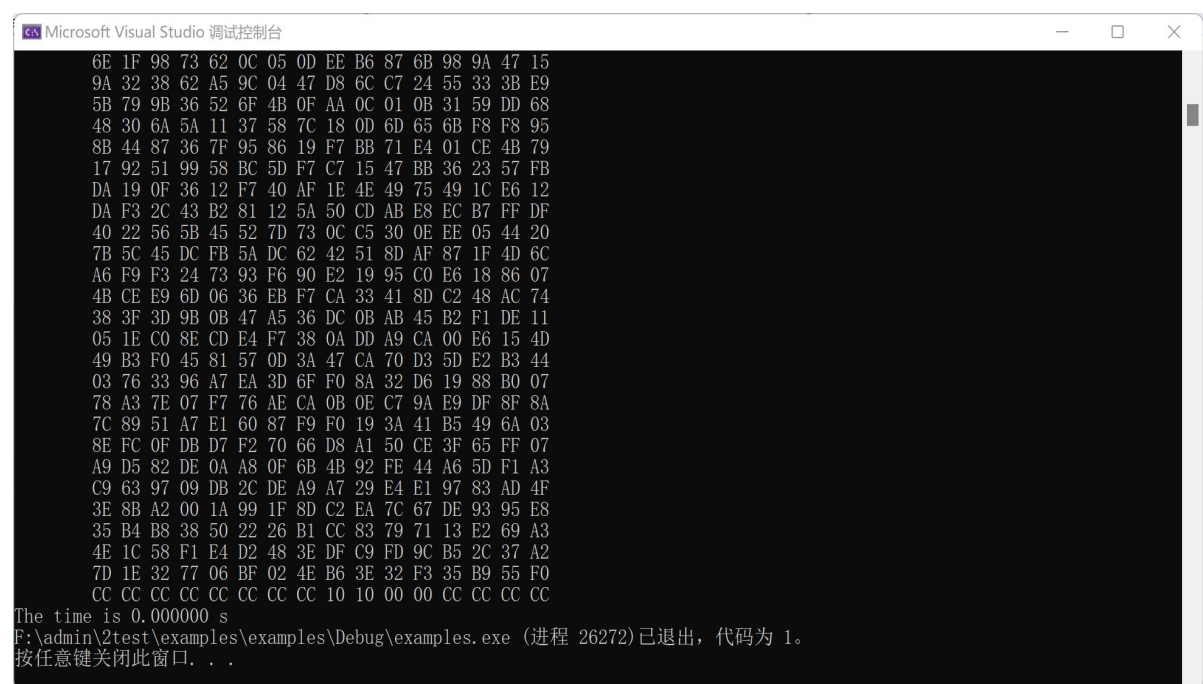
(图 2: 仅加密不输出密文的算法加密时间)

## 2、调用密码库加密运行时间输出



```
Microsoft Visual Studio 调试控制台
6E 1F 98 73 62 0C 05 0D EE B6 87 6B 98 9A 47 15
9A 32 38 62 A5 9C 04 47 D8 6C C7 24 55 33 3B E9
5B 79 9B 36 52 6F 4B 0F AA 0C 01 0B 31 59 DD 68
48 30 6A 5A 11 37 58 7C 18 0D 6D 65 6B F8 F8 95
8B 44 87 36 7F 95 86 19 F7 BB 71 E4 01 CE 4B 79
17 92 51 99 58 BC 5D F7 C7 15 47 BB 36 23 57 FB
DA 19 0F 36 12 F7 40 AF 1E 4E 49 75 49 1C E6 12
DA F3 2C 43 B2 81 12 5A 50 CD AB E8 EC B7 FF DF
40 22 56 5B 45 52 7D 73 0C C5 30 0E EE 05 44 20
7B 5C 45 DC FB 5A DC 62 42 51 8D AF 87 1F 4D 6C
A6 F9 F3 24 73 93 F6 90 E2 19 95 C0 E6 18 86 07
4B CE E9 6D 06 36 EB F7 CA 33 41 8D C2 48 AC 74
38 3F 3D 9B 0B 47 A5 36 DC 0B AB 45 B2 F1 DE 11
05 1E C0 8E CD E4 F7 38 0A DD A9 CA 00 E6 15 4D
49 B3 F0 45 81 57 0D 3A 47 CA 70 D3 5D E2 B3 44
03 76 33 96 A7 EA 3D 6F F0 8A 32 D6 19 88 B0 07
78 A3 7E 07 F7 76 AE CA 0B 0E C7 9A E9 DF 8F 8A
7C 89 51 A7 E1 60 87 F9 F0 19 3A 41 B5 49 6A 03
8E FC 0F DB D7 F2 70 66 D8 A1 50 CE 3F 65 FF 07
A9 D5 82 DE 0A A8 0F 6B 4B 92 FE 44 A6 5D F1 A3
C9 63 97 09 DB 2C DE A9 A7 29 E4 E1 97 83 AD 4F
3E 8B A2 00 1A 99 1F 8D C2 EA 7C 67 DE 93 95 E8
35 B4 B8 38 50 22 26 B1 CC 83 79 71 13 E2 69 A3
4E 1C 58 F1 E4 D2 48 3E DF C9 FD 9C B5 2C 37 A2
7D 1E 32 77 06 BF 02 4E B6 3E 32 F3 35 B9 55 F0
CC CC CC CC CC CC CC CC 10 10 00 00 CC CC CC CC
The time is 0.381000 s
F:\admin\2test\examples\examples\Debug\examples.exe (进程 33168)已退出，代码为 1。
按任意键关闭此窗口。...
```

(图三：密码库加密有输出运行时间)



```
Microsoft Visual Studio 调试控制台
6E 1F 98 73 62 0C 05 0D EE B6 87 6B 98 9A 47 15
9A 32 38 62 A5 9C 04 47 D8 6C C7 24 55 33 3B E9
5B 79 9B 36 52 6F 4B 0F AA 0C 01 0B 31 59 DD 68
48 30 6A 5A 11 37 58 7C 18 0D 6D 65 6B F8 F8 95
8B 44 87 36 7F 95 86 19 F7 BB 71 E4 01 CE 4B 79
17 92 51 99 58 BC 5D F7 C7 15 47 BB 36 23 57 FB
DA 19 0F 36 12 F7 40 AF 1E 4E 49 75 49 1C E6 12
DA F3 2C 43 B2 81 12 5A 50 CD AB E8 EC B7 FF DF
40 22 56 5B 45 52 7D 73 0C C5 30 0E EE 05 44 20
7B 5C 45 DC FB 5A DC 62 42 51 8D AF 87 1F 4D 6C
A6 F9 F3 24 73 93 F6 90 E2 19 95 C0 E6 18 86 07
4B CE E9 6D 06 36 EB F7 CA 33 41 8D C2 48 AC 74
38 3F 3D 9B 0B 47 A5 36 DC 0B AB 45 B2 F1 DE 11
05 1E C0 8E CD E4 F7 38 0A DD A9 CA 00 E6 15 4D
49 B3 F0 45 81 57 0D 3A 47 CA 70 D3 5D E2 B3 44
03 76 33 96 A7 EA 3D 6F F0 8A 32 D6 19 88 B0 07
78 A3 7E 07 F7 76 AE CA 0B 0E C7 9A E9 DF 8F 8A
7C 89 51 A7 E1 60 87 F9 F0 19 3A 41 B5 49 6A 03
8E FC 0F DB D7 F2 70 66 D8 A1 50 CE 3F 65 FF 07
A9 D5 82 DE 0A A8 0F 6B 4B 92 FE 44 A6 5D F1 A3
C9 63 97 09 DB 2C DE A9 A7 29 E4 E1 97 83 AD 4F
3E 8B A2 00 1A 99 1F 8D C2 EA 7C 67 DE 93 95 E8
35 B4 B8 38 50 22 26 B1 CC 83 79 71 13 E2 69 A3
4E 1C 58 F1 E4 D2 48 3E DF C9 FD 9C B5 2C 37 A2
7D 1E 32 77 06 BF 02 4E B6 3E 32 F3 35 B9 55 F0
CC CC CC CC CC CC CC CC 10 10 00 00 CC CC CC CC
The time is 0.000000 s
F:\admin\2test\examples\examples\Debug\examples.exe (进程 26272)已退出，代码为 1。
按任意键关闭此窗口。...
```

(图四：密码库仅加密不输出运行时间)

## 3、运行时间比较

可以明显看出调用密码库进行加密的运行时间要远低于 AES-128 算法进行加密，输出密文在加密过程中占了相当大一部分的时间开销。密码库的加密是非常快的，在不输出的情况下，只对一组 256 个明文组成的集合进行加密时间极短，显示运行时间无限趋近于 0。

### (5) 实验心得

对称加密的加密速度很快，通过此次实验，我们发现 AES 加密算法的时间效率会随着明文长度的改变而正比例增加，调用密码库的加密所用时间比估计的密码算法时间要更短，主要时间花费在明文密文打印中。在密码加密算法中，明文或者密文的分组是相当重要的。字符输入和字节输入主要的差异是进制上存在的不同，以及长度的收集划分。在本次实验中，字节的长度获取以及其他相关函数有关部分的调整花费了较多的时间。对于 AES 算法而言，涉及到有关有限域的部分更需要深入的思考。密码库相较于大段的算法编程而言无疑大量的节省了明文加密的运行时间，由此可见密码库在密码安全这一部分的重要作用。