密码学实验三实验报告

姓名：孟衍璋 学号：16337183

一、 实验目的

对 AES 进行仿真实现。

二、 实验内容

使用 AES 的加密规则，对字符串“School of data science and computer, Sun Yat-sen University.”进行加密。并且选用 CBC 模式，密钥为 sysu，密钥偏移量 IV 为 123（10 进制），补码填充方式为 PKCS5Padding。

三、 实验及算法原理

本次实验中，需要完整实现 AES 的加密过程。加密过程中有几个步骤比较重要。在编写代码时，用几个独立的函数来分别实现。

首先，每轮加密都是对 state 分组进行操作，每个分组 128bits，所以在实现过程中，下列函数的输入输出的结构均如下图所示：

```
['82', '50', '43', '8f']
['a8', 'b7', 'b7', 'a2']
['33', 'ef', '34', 'a8']
['92', '34', 'b7', 'ef']
```

将 128bits 分为 16 组，每组 8bits，用两个十六进制数来表示。

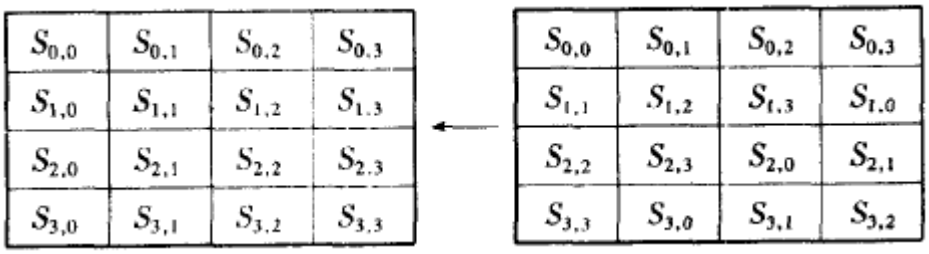
第一个要实现的函数是SubBytes，它使用一个S盒 π_S对state的每一个字节都进行独立的代换，其中 π_S是$\{0,1\}^8$的一个置换。我采用字典的结构来完成这个置换的实现。其中AES的S盒如下：

```
S_box = {
    '00':'63', '01':'7c', '02':'77', '03':'7b', '04':'f2', '05':'6b', '06':'6f', '07':'c5',
    '08':'30', '09':'01', '0a':'67', '0b':'2b', '0c':'fe', '0d':'d7', '0e':'ab', '0f':'76',
    '10':'ca', '11':'82', '12':'c9', '13':'7d', '14':'fa', '15':'59', '16':'47', '17':'f0',
    '18':'ad', '19':'d4', '1a':'a2', '1b':'af', '1c':'9c', '1d':'a4', '1e':'72', '1f':'c0',
    '20':'b7', '21':'fd', '22':'93', '23':'26', '24':'36', '25':'3f', '26':'f7', '27':'cc',
    '28':'34', '29':'a5', '2a':'e5', '2b':'f1', '2c':'71', '2d':'d8', '2e':'31', '2f':'15',
    '30':'04', '31':'c7', '32':'23', '33':'c3', '34':'18', '35':'96', '36':'05', '37':'9a',
    '38':'07', '39':'12', '3a':'80', '3b':'e2', '3c':'eb', '3d':'27', '3e':'b2', '3f':'75',
    '40':'09', '41':'83', '42':'2c', '43':'1a', '44':'1b', '45':'6e', '46':'5a', '47':'a0',
    '48':'52', '49':'3b', '4a':'d6', '4b':'b3', '4c':'29', '4d':'e3', '4e':'2f', '4f':'84',
    '50':'53', '51':'d1', '52':'00', '53':'ed', '54':'20', '55':'fc', '56':'b1', '57':'5b',
    '58':'6a', '59':'cb', '5a':'be', '5b':'39', '5c':'4a', '5d':'4c', '5e':'58', '5f':'cf',
    '60':'d0', '61':'ef', '62':'aa', '63':'fb', '64':'43', '65':'4d', '66':'33', '67':'85',
    '68':'45', '69':'f9', '6a':'02', '6b':'7f', '6c':'50', '6d':'3c', '6e':'9f', '6f':'a8',
    '70':'51', '71':'a3', '72':'40', '73':'8f', '74':'92', '75':'9d', '76':'38', '77':'f5',
    '78':'bc', '79':'b6', '7a':'da', '7b':'21', '7c':'10', '7d':'ff', '7e':'f3', '7f':'d2',
    '80':'cd', '81':'0c', '82':'13', '83':'ec', '84':'5f', '85':'97', '86':'44', '87':'17',
    '88':'c4', '89':'a7', '8a':'7e', '8b':'3d', '8c':'64', '8d':'5d', '8e':'19', '8f':'73',
    '90':'60', '91':'81', '92':'4f', '93':'dc', '94':'22', '95':'2a', '96':'90', '97':'88',
    '98':'46', '99':'ee', '9a':'b8', '9b':'14', '9c':'de', '9d':'5e', '9e':'0b', '9f':'db',
    'a0':'e0', 'a1':'32', 'a2':'3a', 'a3':'0a', 'a4':'49', 'a5':'06', 'a6':'24', 'a7':'5c',
    'a8':'c2', 'a9':'d3', 'aa':'ac', 'ab':'62', 'ac':'91', 'ad':'95', 'ae':'e4', 'af':'79',
    'b0':'e7', 'b1':'c8', 'b2':'37', 'b3':'6d', 'b4':'8d', 'b5':'d5', 'b6':'4e', 'b7':'a9',
    'b8':'6c', 'b9':'56', 'ba':'f4', 'bb':'ea', 'bc':'65', 'bd':'7a', 'be':'ae', 'bf':'08',
    'c0':'ba', 'c1':'78', 'c2':'25', 'c3':'2e', 'c4':'1c', 'c5':'a6', 'c6':'b4', 'c7':'c6',
    'c8':'e8', 'c9':'dd', 'ca':'74', 'cb':'1f', 'cc':'4b', 'cd':'bd', 'ce':'8b', 'cf':'8a',
    'd0':'70', 'd1':'3e', 'd2':'b5', 'd3':'66', 'd4':'48', 'd5':'03', 'd6':'f6', 'd7':'0e',
    'd8':'61', 'd9':'35', 'da':'57', 'db':'b9', 'dc':'86', 'dd':'c1', 'de':'1d', 'df':'9e',
    'e0':'e1', 'e1':'f8', 'e2':'98', 'e3':'11', 'e4':'69', 'e5':'d9', 'e6':'8e', 'e7':'94',
    'e8':'9b', 'e9':'1e', 'ea':'87', 'eb':'e9', 'ec':'ce', 'ed':'55', 'ee':'28', 'ef':'df',
    'f0':'8c', 'f1':'a1', 'f2':'89', 'f3':'0d', 'f4':'bf', 'f5':'e6', 'f6':'42', 'f7':'68',
    'f8':'41', 'f9':'99', 'fa':'2d', 'fb':'0f', 'fc':'b0', 'fd':'54', 'fe':'bb', 'ff':'16',
}
```

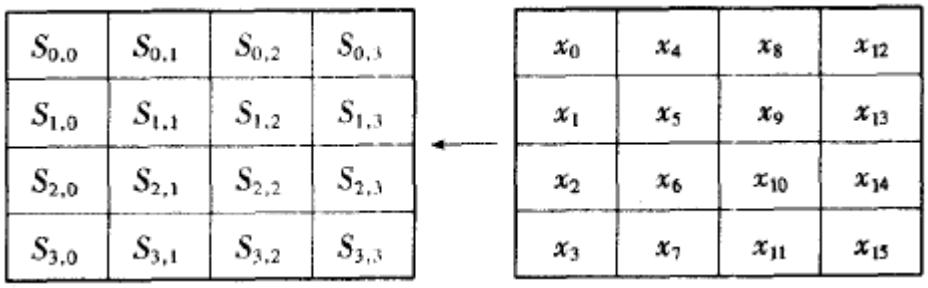
代码部分如下：

```
def SubBytes(state):
    newstate = [[],[],[],[]]  # 为了返回4*4的列表
    for i in range(len(state)):
        for j in range(len(state[0])):
            result = S_box[state[i][j]]
            newstate[i].append(result)
    return newstate
```

第二个要实现的函数是ShiftRows，是对state进行了行变换，变换操作如下：



在实现这个函数的过程中，最开始对一个地方理解错误，所以进行了错误的变换，后来检查到了这个错误之处。出错的原因在于刚开始忽略了 state 是如何编排成 4×4 字节的矩阵的：



本来应该是以列为单位，分组中的比特依次从左到右摆放，而我错误地理解成了一行一行依次摆放，所以刚开始写的变换是有问题的。

更改之后的代码如下：

```python
def ShiftRows(state):
    newstate = [[],[],[],[]]
    newstate[0].append(state[0][0])
    newstate[0].append(state[1][1])
    newstate[0].append(state[2][2])
    newstate[0].append(state[3][3])
    newstate[1].append(state[1][0])
    newstate[1].append(state[2][1])
    newstate[1].append(state[3][2])
    newstate[1].append(state[0][3])
    newstate[2].append(state[2][0])
    newstate[2].append(state[3][1])
    newstate[2].append(state[0][2])
    newstate[2].append(state[1][3])
    newstate[3].append(state[3][0])
    newstate[3].append(state[0][1])
    newstate[3].append(state[1][2])
    newstate[3].append(state[2][3])
    return newstate
```

第三个要实现的函数是 **MixColumns**，即对 state 的每一列都进行操作，替换为新的列，这个新列由原列乘上域F_{2^8}中的元素组成的矩阵而来。

因为对于 AES 来说，域乘法只需要算与 x 和 x+1 的乘积，乘以 x 相当于左移一位,乘以 x+1 相当于左移一位之后与自身相异或。但是由于是在域F_{2^8}中，还要考虑模一个多项式之后的结果，所以还要考虑到这点，不能简单地移位。刚开始也是忽略了这一点，所以出了些错误。

为了完成 MixColumns 还需要完成下列函数：

```python
def XOR(a,b):
    c = ''
    for i in range(len(a)):
        c += str(int(a[i]) ^ int(b[i]))
    return c
```

```python
def shift_8(s):
    result = ''
    if(s[0] == '0'):
        result += s[1:8]
        result += '0'
    else:
        result += s[1:8]
        result += '0'
        result = XOR(result,'00011011')
    return result
```

```python
def FieldMult(c):
    new_c = []
    t0 = c[0]
    t1 = c[1]
    t2 = c[2]
    t3 = c[3]
    # u0 = reduce(XOR, [shift_8(t0), XOR(shift_8(t1),t1), t2, t3])
    u0 = XOR(shift_8(t0), XOR(shift_8(t1),t1))
    u0 = XOR(u0, t2)
    u0 = XOR(u0, t3)
    # u1 = reduce(XOR, [shift_8(t1), XOR(shift_8(t2),t2), t3, t0])
    u1 = XOR(shift_8(t1), XOR(shift_8(t2),t2))
    u1 = XOR(u1, t3)
    u1 = XOR(u1, t0)
    # u2 = reduce(XOR, [shift_8(t2), XOR(shift_8(t3),t3), t0, t1])
    u2 = XOR(shift_8(t2), XOR(shift_8(t3),t3))
    u2 = XOR(u2, t0)
    u2 = XOR(u2, t1)
    # u3 = reduce(XOR, [shift_8(t3), XOR(shift_8(t0),t0), t1, t2])
    u3 = XOR(shift_8(t3), XOR(shift_8(t0),t0))
    u3 = XOR(u3, t1)
    u3 = XOR(u3, t2)
    new_c.append(u0)
    new_c.append(u1)
    new_c.append(u2)
    new_c.append(u3)
    return new_c
```

```python
def Hex2Bin(state):
    transform = {'0':'0000', '1':'0001', '2':'0010', '3':'0011', '4':'0100', '5':'0101', '6':'0110',
        '7':'0111', '8':'1000', '9':'1001', 'a':'1010', 'b':'1011', 'c':'1100', 'd':'1101', 'e':'1110', 'f':'1111'}
    state_bin = [[],[],[],[]]
    for i in range(len(state)):
        for j in range(len(state[0])):
            result = transform[state[i][j][0]] + transform[state[i][j][1]]
            state_bin[i].append(result)
    return state_bin

def Bin2Hex(state):
    transform = {'0000':'0', '0001':'1', '0010':'2', '0011':'3', '0100':'4', '0101':'5', '0110':'6',
        '0111':'7', '1000':'8', '1001':'9', '1010':'a', '1011':'b', '1100':'c', '1101':'d', '1110':'e', '1111':'f'}
    state_hex = [[],[],[],[]]
    for i in range(len(state)):
        for j in range(len(state[0])):
            result = transform[state[i][j][:4]] + transform[state[i][j][-4:]]
            state_hex[i].append(result)
    return state_hex
```

MixColumns 的代码如下：

```python
def MixColumns(state):
    state = Hex2Bin(state)
    newcol = []
    for i in range(len(state)):
        newcol.append(FieldMult(state[i]))
    newcol = Bin2Hex(newcol)
    return newcol
```

第四个要实现的函数是 **keyExpansion**，即 AES 的密钥编排方案。要将 128bits 的种子密钥扩展为 11 个轮密钥，每个轮密钥由 16 个字节组成。最后返回的扩展密钥由 44 个字组成，表示为 w[0],…,w[43]，其中每个 w[i]都是一个字。

为了实现 keyExpansion 操作，还需要完成两个操作：RotWord 和 SubWord。RotWord 对 4 个字节进行循环移位，SubWord 对 4 个字节使用 AES 的 S 盒，实现分别如下：

```python
def RotWord(t):
    new_t = ''
    new_t += t[2:8]
    new_t += t[:2]
    return new_t
```

```python
def SubWord(t):
    new_t = ''
    new_t += S_box[t[:2]]
    new_t += S_box[t[2:4]]
    new_t += S_box[t[4:6]]
    new_t += S_box[t[6:8]]
    return new_t
```

为了完成 keyExpansion 操作，还需要增加如下函数：

```python
def Hex2Bin_str(s):
    transform = {'0':'0000', '1':'0001', '2':'0010', '3':'0011', '4':'0100', '5':'0101', '6':'0110',
        '7':'0111', '8':'1000', '9':'1001', 'a':'1010', 'b':'1011', 'c':'1100', 'd':'1101', 'e':'1110', 'f':'1111'}
    new_s = ''
    for i in range(len(s)):
        new_s += transform[s[i]]
    return new_s
```

```python
def Bin2Hex_str(s):
    # print(s)
    transform = {'0000':'0', '0001':'1', '0010':'2', '0011':'3', '0100':'4', '0101':'5', '0110':'6',
        '0111':'7', '1000':'8', '1001':'9', '1010':'a', '1011':'b', '1100':'c', '1101':'d', '1110':'e', '1111':'f'}
    new_s = ''
    for i in range(len(s) // 4):
        new_s += transform[s[4*i:4*i+4]]
    return new_s
```

```python
def XOR_hex(a,b):
    c_bin = ''
    a_bin = Hex2Bin_str(a)
    b_bin = Hex2Bin_str(b)
    for i in range(len(a_bin)):
        c_bin += str(int(a_bin[i]) ^ int(b_bin[i]))
    c = Bin2Hex_str(c_bin)
    return c
```

用来完成相应操作与结构的统一。

最后 keyExpansion 的代码如下：

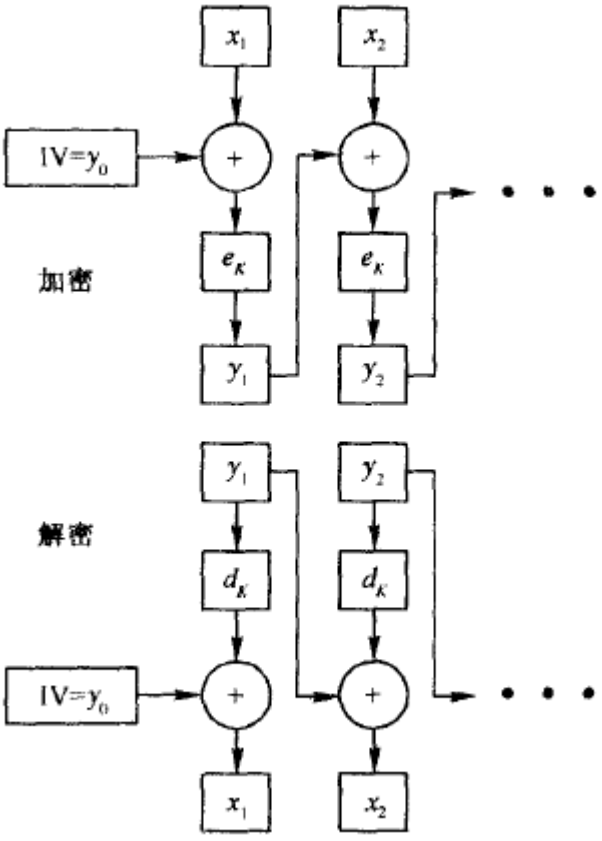
```python
def KeyExpansion(key):
    w = [] # 用来储存输出的轮密钥
    for i in range(44):
        w.append('')
    RCon = []
    RCon.append('01000000')
    RCon.append('02000000')
    RCon.append('04000000')
    RCon.append('08000000')
    RCon.append('10000000')
    RCon.append('20000000')
    RCon.append('40000000')
    RCon.append('80000000')
    RCon.append('1b000000')
    RCon.append('36000000')
    for i in range(4):
        w[i] += key[4*i]
        w[i] += key[4*i+1]
        w[i] += key[4*i+2]
        w[i] += key[4*i+3]
        # print(w[i])

    for i in range(4,44):
        temp = w[i - 1]
        if(i % 4 == 0):
            temp = XOR_hex(SubWord(RotWord(temp)), RCon[int(i/4) - 1])
        w[i] = XOR_hex(w[i - 4], temp)
        # print(w[i])
    return w
```

分别实现了上述函数之后，就要考虑执行 AES 加密的工作。首

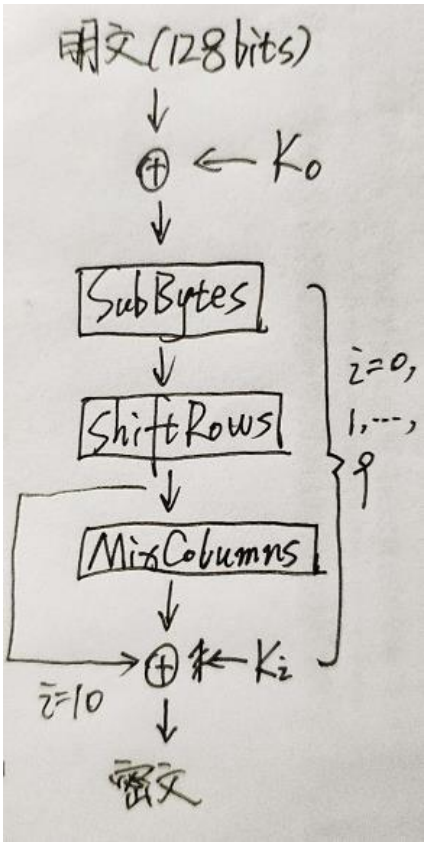
先要理解 padding 的部分，密钥和密钥偏移量转换为二进制之后，只需要在后面补 0 补至 128 位。明文转换为二进制后，需要分为 128bits 一组的多个分组，剩下的一组如果不足 128bits，并且还差 x 比特到 128 比特，就在后面补 t=（x/32）个‘0x0t’。

然后就要理解 CBC 模式的含义，每一个密文分组在用密钥 K 加密之前，都要先跟下一个明文分组相异或。比如开头密钥偏移量 IV 与第一个明文分组异或之后再用密钥 K 进行加密。

CBC 模式示意图如下：



加密过程的示意图如下：



最后，经过 10 轮迭代，得到的 state 即为最终的密文。

四、 实验代码

AES.py 2018/10/20 20:46 Python File 11 KB

五、 运行截图

```
db5d034554088b2e896988e616290411ce9b0a11f0a78fe19d00da39161aa1e3258
46723fc1c55461037fe2c166d45ae94f41d456b95c8187ac6336fe3cd6f85
[Finished in 0.3s]
```

六、 结果分析

在 AES 在线解密的网站上运行如下：

与我的实验结果相符。

七、 总结

这次实现 AES 的实验，过程看似复杂，但其实如果将过程一步步列举出来，先实现几个重要的函数，再按照加密的流程一步步走，最后总能完整地再现 AES 的过程。

在完成作业的过程中，开始对 AES 不太熟悉，后来将书上有关 AES 的内容读了几遍，才完全清楚流程。这次实验最大的难点应该在于 AES 的过程过于繁琐，所以在实现过程中哪怕出了一点小小的差错也会失之毫厘、差之千里，只有对照着过程一步一步核对，但想要找出错误十分困难。

刚开始的时候理解错了一些地方，比如对 PKCS5Padding 的理解不够透彻，后来发现形成明文分组的时候出了问题。还有对 state 结构的错误理解导致 ShiftRows 的操作错误。

总而言之， 在这次 AES 的实现过程中，对这种加密算法有了很透彻的理解，了解了其复杂的结构，也明白了为何对现在的所

有已知攻击而言，AES 是安全的。感觉这次的实验让我收获颇丰。