CBC和CTR模式下的AES实验报告

学号：SA20225172 姓名：郭俊勇

实现目的

- 了解分组密码的结构特点；
- 掌握传统分组密码结构AES，以及AES在两种工作模式CBC和CTR下的实现；

编程语言

Python

实验内容

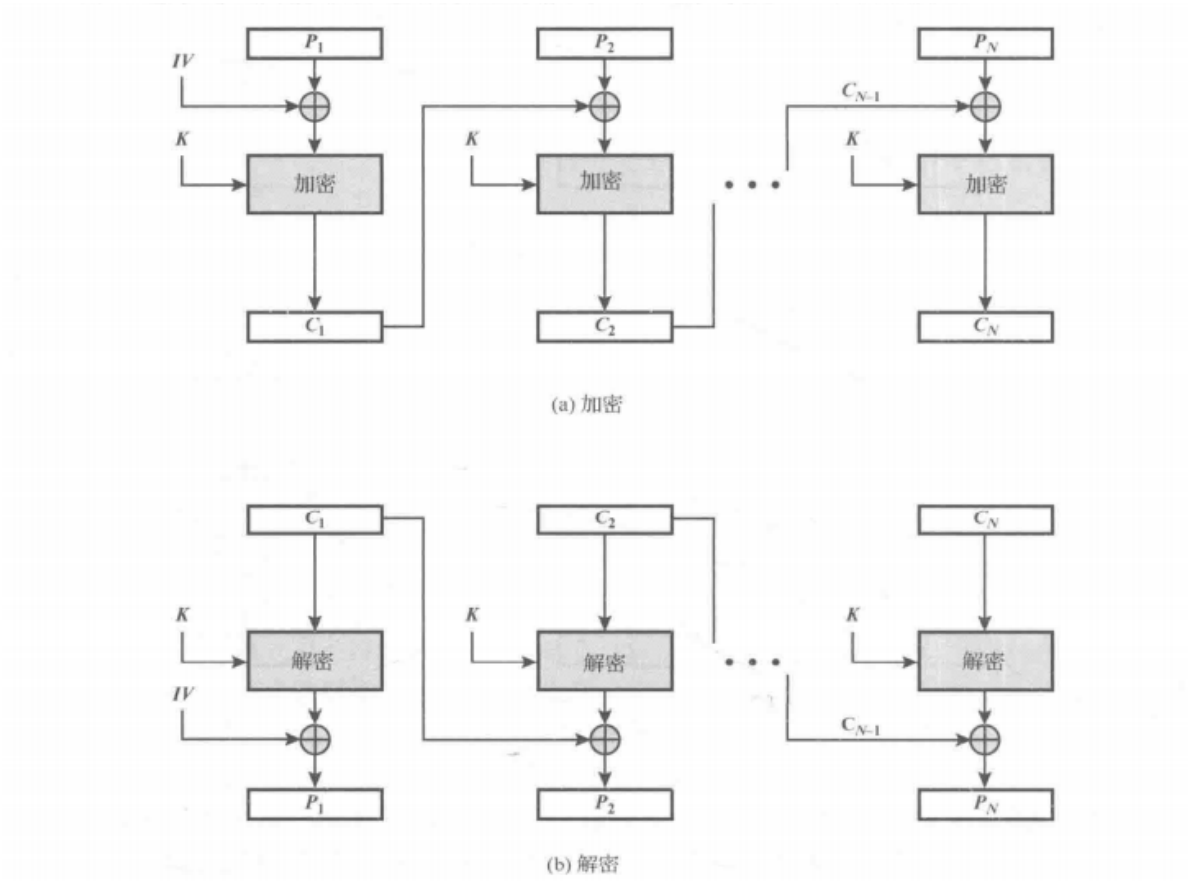
- 在本次实验中，需要实现两个加密/解密系统，一个在密文分组链接模式（CBC）下使用AES，另一个在计数器模式（CTR）中使用AES；
- 完成程序后，使用附件的test.txt中给出的四组密钥和密文（十六进制形式）来验证你的代码。

实验原理分析

预备知识：

PKCS5 是按 8 字节分组对数据进行填充的：如果要填充 1 个字节，那填入的值就是 0x01；如果要填充 2 个字节，那么填入的值就是 0x02，以此类推。但若待加密数据长度正好为 8 的整数倍时，则需要填入 8 个 0x08。

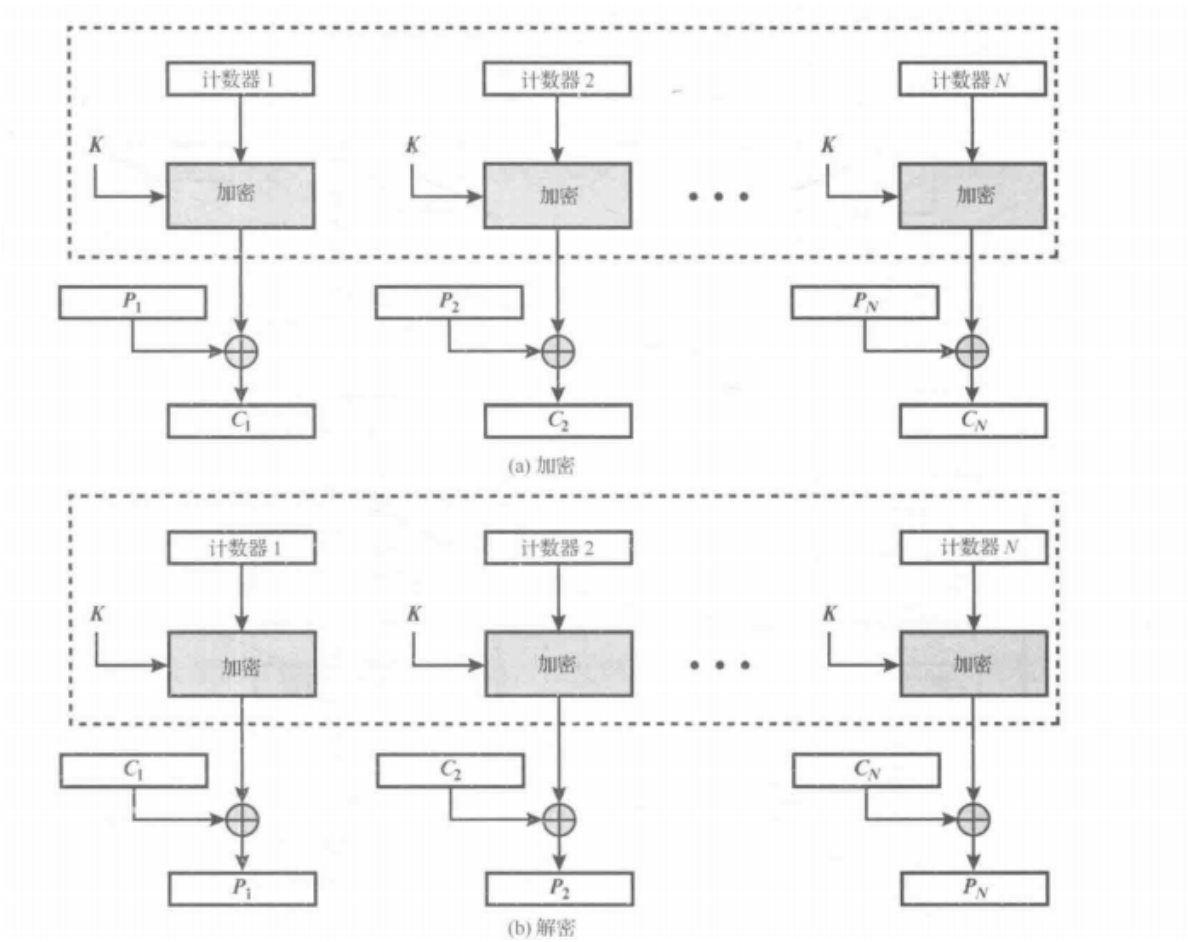
CBC加解密原理分析：



加解密通式：

CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}])$ $j=2,\cdots,N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1}$ $j=2,\cdots,N$

CTR加密原理分析：



加解密通式：

CTR	$C_j = P_j \oplus E(K, T_j)$ $j=1,\cdots,N-1$ $C_N^* = P_N^* \oplus MSB_u[E(K, T_N)]$	$P_j = C_j \oplus E(K, T_j)$ $j=1,\cdots,N-1$ $P_N^* = C_N^* \oplus MSB_u[E(K, T_N)]$

Python代码实现（源码）

```
import Crypto.Cipher.AES as AES
import operator
from binascii import a2b_hex
from Crypto import Random


# 进行异或(bytes ^ bytes) 按位异或，迭代器
def xor_block(left, right):
    return map(operator.xor, left, right)


# CBC模式实现：
class CBC_Cipher(object):
    def __init__(self, key):
        self.my_cipher = AES.new(key, AES.MODE_ECB)
```

```
        self.block_size = AES.block_size

    def encrypt(self, plainText, iv):
        plainText = bytes(plainText)
        blockSZ = self.block_size
        iv = bytes(iv)
        # PKCS#7填充规则
        padLen = (blockSZ - len(plainText) % blockSZ)
        plainText = plainText + bytes([padLen] * padLen)
        # 填充密文内容
        cipherText = bytearray(blockSZ + len(plainText))
        # 将密文前blockSz位置填上iv
        cipherText[0:blockSZ] = iv
        for i in range(blockSZ, len(cipherText), blockSZ):
            # 块长度
            j = i + blockSZ
            # 异或操作
            after_xor = bytes(xor_block(plainText[i - blockSZ:i], cipherText[i -
blockSZ:i]))
            # 调用加密接口
            cipherText[i:j] = self.my_cipher.encrypt(after_xor)

        return cipherText

    def decrypt(self, cipherText):
        blockSZ = self.block_size
        cipherText = bytes(cipherText)
        after_decrypt = self.my_cipher.decrypt(cipherText[blockSZ:])
        blocks = xor_block(after_decrypt, cipherText[:-blockSZ])
        plainText = bytes(blocks)
        return plainText[:-plainText[-1]]


# ***************************************
# CTR模式实现
# 转化为bytes类型
def int_to_bytes(x):
    return x.to_bytes((x.bit_length() + 7) // 8, 'big')


# bytes转为十进制整数
def int_from_bytes(xbytes):
    return int.from_bytes(xbytes, 'big')


class CTRCipher(object):
    def __init__(self, key):
        self._cipher = AES.new(key, AES.MODE_ECB)
        self.block_size = AES.block_size

    def encrypt(self, plainText, count):
        count = bytes(count)
        # 各个计数器值
        counters = self._get_timers(count, len(plainText))
        blocks = xor_block(self._cipher.encrypt(counters), plainText)
        ciphertext = bytes(blocks)
        return count + ciphertext[:len(plainText)]
```

```
    def decrypt(self, cipherText):
        blockSZ = self.block_size
        # 加密和解密只有输入不同
        pt = self.encrypt(cipherText[blockSZ:], cipherText[:blockSZ])
        return pt[blockSZ:]

    # 生成各个计数器的值
    def _get_timers(self, iv, msgLen):
        # iv: 计时器初值
        # msgLen: 密文长度(明文)
        blockSZ = self.block_size
        blocks = int((msgLen + blockSZ - 1) // blockSZ)
        timer = int_from_bytes(iv)
        timers = iv
        for i in range(1, blocks):
            timer += 1
            timers += int_to_bytes(timer)
        return timers


if __name__ == '__main__':
    # CBC模式
    cbc_key = "140b41b22a29beb4061bda66b6747e14"
    cbc_cipher1 =
"4ca00ff4c898d61e1edbf1800618fb2828a226d160dad07883d04e008a7897ee2e4b7465d5290d0
c0e6c6822236e1daafb94ffe0c5da05d9476be028ad7c1d81"
    cbc_cipher2 =
"5b68629feb8606f9a6667670b75b38a5b4832d0f26e1ab7da33249de7d4afc48e713ac646ace36e
872ad5fb8a512428a6e21364b0c374df45503473c5242a253"
    cbc_key = a2b_hex(cbc_key)
    # 初始化
    decryptor1 = CBC_Cipher(cbc_key)
    print("CBC模式下的解密：")
    print("第一个明文结果：", decryptor1.decrypt(a2b_hex(cbc_cipher1)).decode('utf-
8'))
    print("第二个明文结果：", decryptor1.decrypt(a2b_hex(cbc_cipher2)).decode('utf-
8'))
    iv = Random.new().read(AES.block_size)
    # print("iv随机生成值", iv)
    a = decryptor1.encrypt(b"hello world!", iv)
    print("CBC模式下加密", a)
    print("加密后解密结果:", decryptor1.decrypt(a).decode('utf-8'))


 print("-----------------------------------------------------------------------------
-----------------------------------------------------")

    # CTR模式
    ctr_key = "36f18357be4dbd77f050515c73fcf9f2"
    ctr_cipher1 =
"69dda8455c7dd4254bf353b773304eec0ec7702330098ce7f7520d1cbbb20fc388d1b0adb5054db
d7370849dbf0b88d393f252e764f1f5f7ad97ef79d59ce29f5f51eeca32eabedd9afa9329"
    ctr_cipher2 =
"770b80259ec33beb2561358a9f2dc617e46218c0a53cbeca695ae45faa8952aa0e311bde9d4e017
26d3184c34451"
    decryptor2 = CTRCipher(a2b_hex(ctr_key))
    print("CTR模式下的解密：")
```

```
    print("第一个明文结果：", decryptor2.decrypt(a2b_hex(ctr_cipher1)).decode('utf-8'))
    print("第二个明文结果：", decryptor2.decrypt(a2b_hex(ctr_cipher2)).decode('utf-8'))
    a = decryptor2.encrypt(b"hello world!", iv)
    print("CTR加密后的结果：", a)
    print("CTR解密后", decryptor2.decrypt(a).decode('utf-8'))
```

运行结果