B10615023 四資工三甲

- **分工**

  高楚云：ECB、Cool Mode、Debuging

  楊傑安：CBC、File in/out、Padding、Data Structure

- **建置環境**

  MacOS 10.14.6
  Python 3.7

- **操作方式**

  執行方式

  %> python3 hw3.py [InputFilePath] [OutputFilePath] [encrypt/decrypt] [mode] [key]

  [encrypt/decrypt] 輸入檔案路徑

  [encrypt/decrypt] 輸出檔案路徑

  輸入輸出檔案皆為.png格式

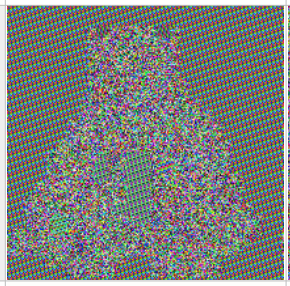
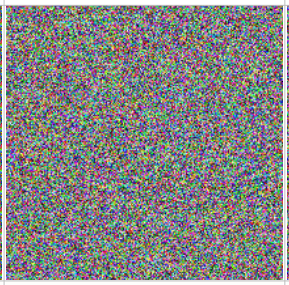
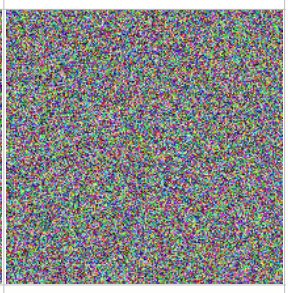
  [encrypt/decrypt]={encrypt,decrypt} 選擇執行加密或解密

  [mode]={ecb,cbc,cool} 選擇block cipher運作方式

  [key] 用以加密的key,以hex表示,可以為128/192/256 bits

  e.g. %> python3 hw3.py test.png test_ecb_en.png encrypt ecb 000102030405060708091211121314151617181920212223

- **執行結果**

	ECB	CBC	COOL
encrypt			
decrypt			

B10615023 四資工三甲

- **程式碼及解說**

```
from Crypto.Cipher import AES
from PIL import Image
from sys import argv
import copy

# 如果要加密的data沒有滿一個block則需要padding
def pad(text):
    padding = 16 - (len(text) % 16)
    return text + bytes([padding] * padding)

# 一次加密一個block
def AES_encrypt_one_block(plain,key):
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(plain)
    return ciphertext

#一次解密一個block
def AES_decrypt_one_block(cipher,key):
    plain = AES.new(key, AES.MODE_ECB)
    plaintext = plain.decrypt(cipher)
    return plaintext

# bytes type的XOR
def bytes_xor(b1, b2): # use xor for bytes
    parts = []
    for b1, b2 in zip(b1, b2):
        parts.append(bytes([b1 ^ b2]))
    return b''.join(parts)
```

B10615023 四資工三甲

```python
# 資料結構用來處理PPM的內容
class AESppm():
    def __init__(self,ppm_bin):
        #編碼形式
        self.p_number = str(ppm_bin.readline(),"utf-8")
        string = ppm_bin.readline()
        # 圖片的長寬
        self.length = int(string.split()[0])
        self.width = int(string.split()[1])
        # 每一個pixel的位元深度
        self.depth = int(ppm_bin.readline())
        # 每一個pixel的內容的binary data
        self.pixels_bin = ppm_bin.read()

    # return 整個資料結構的bytes
    def __bytes__(self):
        to_return = bytes(self.p_number+"\n","utf-
8")+bytes(str(self.length),'utf-8')+bytes(" ","utf-
8")+bytes(str(self.width),'utf-8')+bytes("\n","utf-
8")+bytes(str(self.depth),'utf-8')+bytes("\n","utf-8")+self.pixels_bin
        return to_return

    # ecb的加密、解密
    def ecb(self,key,en_de) : #0=encrypt 1=decrypt
        # 先做pad
        self.pixels_bin = pad(self.pixels_bin)
        to_return = self
        # 避免side effect
        pixels_bin_origin = self.pixels_bin
        to_return.pixels_bin=bytes()
        # encrypt
        if en_de==0:
            cipherblock = bytes()
            # 對ppm的pixel_bin切成一個一個block加密
            for i in range(int(len(pixels_bin_origin)/16)) :
                block = pixels_bin_origin[16*i:16*i+16]
                cipherblock += AES_encrypt_one_block(block,key)
            to_return.pixels_bin += cipherblock
        # decrypt
        elif en_de==1:
            plainblock = bytes()
            # 對ppm的pixel_bin切成一個一個block加密
            for i in range(int(len(pixels_bin_origin)/16)) :
                block = pixels_bin_origin[16*i:16*i+16]
                plainblock += AES_decrypt_one_block(block,key)
            to_return.pixels_bin += plainblock
        return to_return
```

B10615023 四資工三甲

```
    # cbc的加密、解密
    def cbc(self,key,en_de) :
        # 先做pad
        self.pixels_bin = pad(self.pixels_bin)
        # 避免side effect
        to_return = copy.copy(self)
        to_return.pixels_bin=bytes()
        processed_block=bytes()
        # 用key的前128bits作為initial vector
        vector = key[0:16]
        if(en_de==0) :
            # encrypt
            for i in range(int(len(self.pixels_bin)/16)) :
                # 切成一個一個block
                block=self.pixels_bin[16*i:16*i+16]
                # 做XOR
                block = bytes_xor(block,vector)
                processed_block = AES_encrypt_one_block(block,key)
                vector = processed_block
                to_return.pixels_bin+=processed_block
        else :
            # encrypt
            for i in range(int(len(self.pixels_bin)/16)) :
                # 切成一個一個block
                block=self.pixels_bin[16*i:16*i+16]
                # 一個block的decrypt
                processed_block = AES_decrypt_one_block(block,key)
                # 做XOR
                processed_block = bytes_xor(processed_block,vector)
                # cipherblock 取代initial vector
                vector = block
                to_return.pixels_bin+=processed_block
        return to_return
```

B10615023 四資工三甲

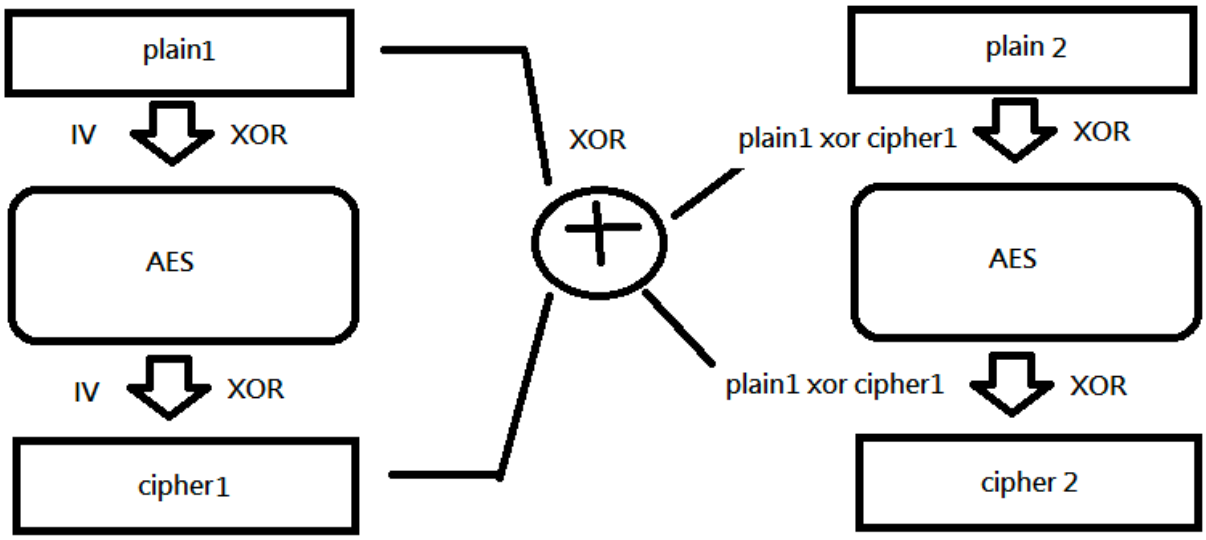
```
    # cool的方式的加密、解密
    def cool(self,key,en_de) :
        # 先做pad
        self.pixels_bin = pad(self.pixels_bin)
        # 避免side effect
        to_return = copy.copy(self)
        to_return.pixels_bin=bytes()
        processed_block=bytes()
        # 用key的前128bits作為initial vector
        vector = key[0:16]
        for i in range(int(len(self.pixels_bin)/16)) :
            # 切成一個一個block
            block=self.pixels_bin[16*i:16*i+16]
            # 做第一次XOR
            xored_block = bytes_xor(block,vector)
            # 一個block的en/decrypt
            if(en_de==0) :
                processed_block = AES_encrypt_one_block(xored_block,key)
            else :
                processed_block = AES_decrypt_one_block(xored_block,key)
            # 做第二次XOR
            processed_block = bytes_xor(processed_block,vector)
            to_return.pixels_bin+=processed_block
            # plainblock跟cipherblock取代initial vector
            vector = bytes_xor(processed_block,block)
        return to_return
```

B10615023 四資工三甲

```python
if __name__ == "__main__":
    # 參數數量檢查
    if(len(argv)!=6):
        print("Arguments error")
        exit()
    # 檔案路徑
    input_png_path = argv[1]
    output_png_path = argv[2]
    #加解密判斷
    if(argv[3].lower()=="encrypt") :
        en_de=0;
    elif(argv[3].lower()=="decrypt") :
        en_de=1;
    else :
        print("encyrpt or decrypt")
        exit()
    # blockcipher模式
    mode = argv[4].lower()
    # 把key轉為bytes type
    key = bytes.fromhex(argv[5])
    # 將input轉成ppm格式
    with Image.open(input_png_path) as input_png_file :
        input_ppm_path = input_png_path[:-4]+".ppm"
        input_png_file.save(input_ppm_path)
    # 讀入轉好的ppm
    with open(input_ppm_path,'rb') as input_ppm_bin :
        ppm_file = AESppm(input_ppm_bin)
        # 判斷blockcipher方式然後執行加解密
        if(mode=="ecb") :
            ppm_file=ppm_file.ecb(key,en_de)
        elif(mode=="cbc") :
            ppm_file=ppm_file.cbc(key,en_de)
        elif(mode=="cool") :
            ppm_file=ppm_file.cool(key,en_de)
        output_ppm_path = "./output.ppm"
        #儲存成ppm格式
        with open(output_ppm_path,'wb') as output_ppm_bin :
            output_ppm_bin.write(bytes(ppm_file))
        #將儲存的ppm轉成png格式
        with Image.open(output_ppm_path) as output_ppm :
            output_ppm.save(output_png_path,output_png_path[-3:])
```

B10615023 四資工三甲

- ## Cool mode解說



1. IV為key的前128bits
2. plainblock和IV做XOR
3. 結果做加/解密
4. 再對加密結果和IV做XOR即得到cipherblock
5. IV用plainblock和cipherblock做XOR的結果取代,與下個block做上述的處理

Advantage:

結構對稱,只需要將AES module切換en/decrypt即可切換加解密

運算只有XOR,速度快

Disadvantage:

無法平行化

- ## 遇到困難與心得

1. 因為這次需要做檔案轉換有不少的檔案要管理,在路徑的處理上花了不少時間.
2. 這次作業包含對binary檔案的處理(bytes),之前沒有處理過,也花了不少時間在找bytes型別轉換的資料
3. 對於Python的parameter傳遞的方式不熟悉,因為side effect而在debug也花了不少時間