Information Security HW3

- 分工

  四資工三 B10615033 王璽禎 Decrypt
  四資工三 B10615035 陳靜萱 Encrypt

- 開發環境

  Visual studio Code 使用 Python 撰寫

- 操作方式

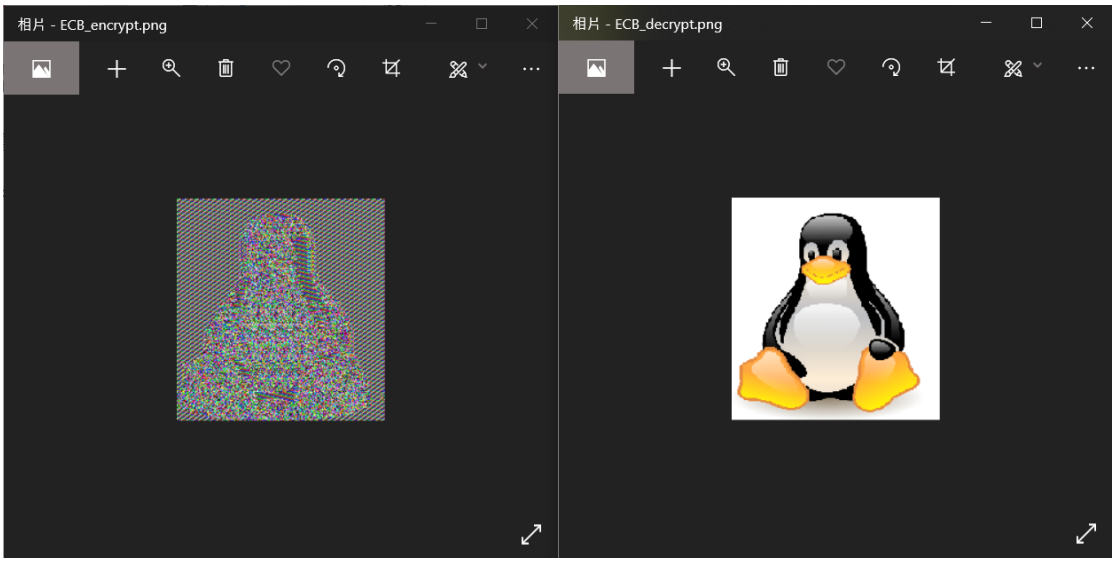
```
python block_cipher.py ECB "C:\Users\HP AY111TX\Desktop\IS\penguin.png"
```

  ☆ 輸入參數：
  python block_cipher.py (mode) (path)
  argv[1] = mode 種類 (ECB/CBC/cool)
  argv[2] = 圖片路徑 (path)

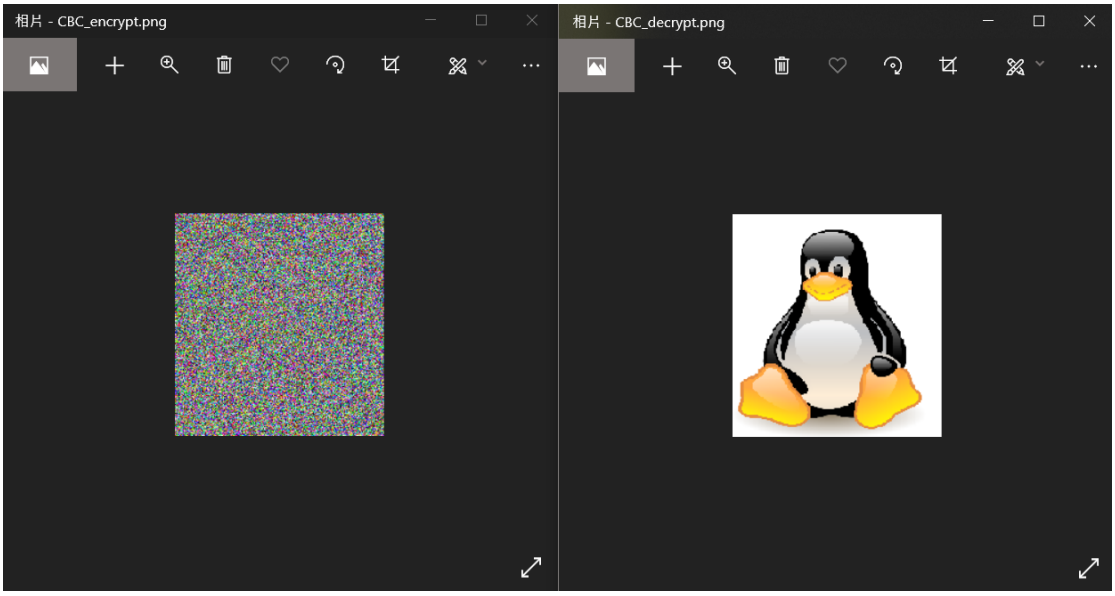
  執行完後會在目錄下產生加密與解密的 png 圖檔

- 執行結果圖

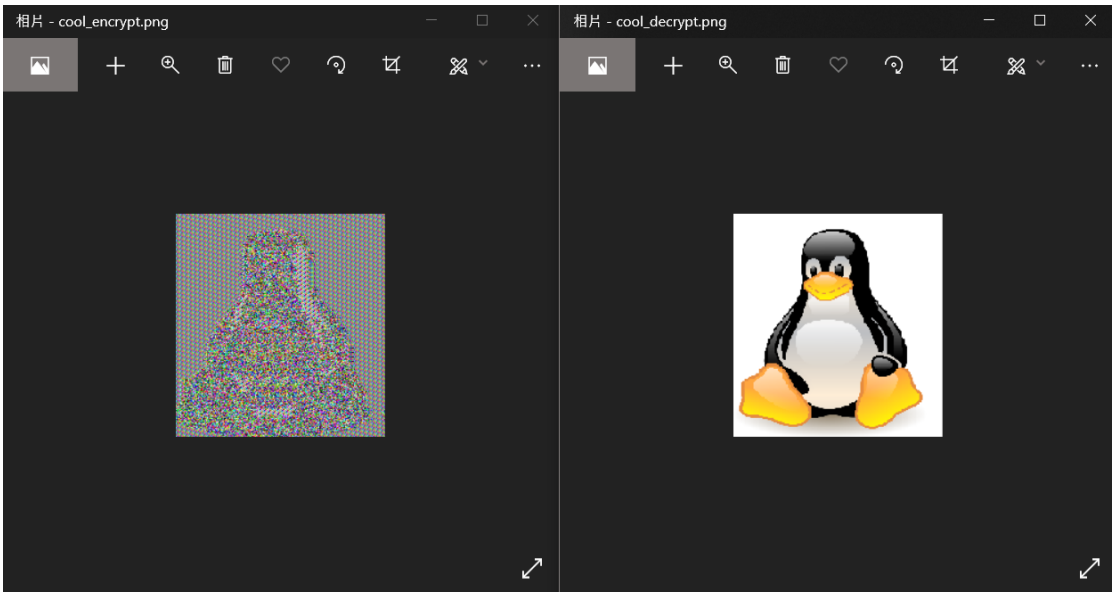
  ☆ **ECB encrypt / decrypt**



  ☆ **CBC encrypt / decrypt**



☆ **cool encrypt / decrypt**



- 程式碼

```python
if __name__ == "__main__":
    mode = sys.argv[1]
    path = sys.argv[2]

    input_image = Image.open(path)
    input_byte = input_image.convert("RGB").tobytes()
    input_byte = pad(input_byte)    # 將不足16bytes的block補滿

    key = os.urandom(16)

    if mode == 'ECB':
        ECB(input_byte, key)
        ECB_encrypt = Image.open('./ECB_encrypt.png')
        ECB_encrypt_byte = ECB_encrypt.convert("RGB").tobytes()
        ECB_decrypt(ECB_encrypt_byte, key)
        print("Done")
    elif mode == 'CBC':
        IV = os.urandom(16)
        CBC(input_byte, key, IV)
        CBC_encrypt = Image.open('./CBC_encrypt.png')
        CBC_encrypt_byte = CBC_encrypt.convert("RGB").tobytes()
        CBC_decrypt(CBC_encrypt_byte, key, IV)
        print("Done")
    elif mode == 'cool':
        IV = os.urandom(16)
        cool(input_byte, key, IV)
        cool_encrypt = Image.open('./cool_encrypt.png')
        cool_encrypt_byte = cool_encrypt.convert("RGB").tobytes()
        cool_decrypt(cool_encrypt_byte, key, IV)
        print("Done")
```

將圖片轉換成 byte，並根據長度做 pad。
Key 和 IV 為隨機產生的 16bytes 長度的 data。
先做加密，將加密出來的圖片讀取後轉成 byte，再做解密。

☆ 用到的 **Function**

```python
def byte_xor(ba1, ba2):
    return bytes([_a ^ _b for _a, _b in zip(ba1, ba2)])
```

直接將 byte 做 xor，不需要轉成 bit

```
def pad(text):
    while len(text)%16 != 0:
        text += bytes([0])
    return text
```

手動做 pad，將不足 16bytes 的部分補 0。

☆ **ECB**

```
def ECB(input_byte, key):
    cipher = AES.new(key, AES.MODE_ECB)
    cipher_image = b''
    for i in range(0, len(input_byte), 16):
        plaintext = input_byte[i:i+16]
        ciphertext = cipher.encrypt(plaintext)
        cipher_image += ciphertext

    encrypt_image = Image.frombytes("RGB", input_image.size, cipher_image)
    encrypt_image.save('./ECB_encrypt.png')
```

將 key 結合 ECB mode 利用 AES 加密技術變成一個 block cipher。一個 block 為 16bytes，將一次 16bytes 的圖片長度放進去加密，出來的結果為 16bytes 的 ciphertext。利用 for 迴圈將所有圖片的 input byte 跑一遍，最後加密後的圖片為所有 16bytes 的 ciphertext 的總和。
利用 PIL 中的 Image.frombytes，參數為(mode, image_size, byte_buffer)，將 pixel data 轉成 image object，模式為 RGB(3*8 bit pixels, true color)。

☆ **ECB decryption**

```
def ECB_decrypt(input_byte, key):
    cipher = AES.new(key, AES.MODE_ECB)
    origin_image = b''
    for i in range(0, len(input_byte), 16):
        ciphertext = input_byte[i:i+16]
        plaintext = cipher.decrypt(ciphertext)
        origin_image += plaintext

    decrypt_image = Image.frombytes("RGB", ECB_encrypt.size, origin_image)
    decrypt_image.save('./ECB_decrypt.png')
```

decryption 為 encryption 的相反，由於 ECB 的 Block 之間沒有關係，因此將 encrypt 後的 ciphertext 和 plaintext 交換後，出來的結果就是原來的圖片 。

☆ **CBC**

```
def CBC(input_byte, key, IV):
    cipher = AES.new(key, AES.MODE_ECB)
    cipher_image = b''
    for i in range(0, len(input_byte), 16):
        plaintext = input_byte[i:i+16]
        xor = byte_xor(plaintext, IV)
        ciphertext = cipher.encrypt(xor)
        cipher_image += ciphertext
        IV = ciphertext

    encrypt_image = Image.frombytes("RGB", input_image.size, cipher_image)
    encrypt_image.save('./CBC_encrypt.png')
```

CBC 比 ECB 多了一個 xor 的過程，將隨機產生出來的 IV，和 plaintext 做 xor，xor 出來的結果丟到 cipher block 做加密後就是 ciphertext。下一個 IV 更新成 ciphertext 的資料。

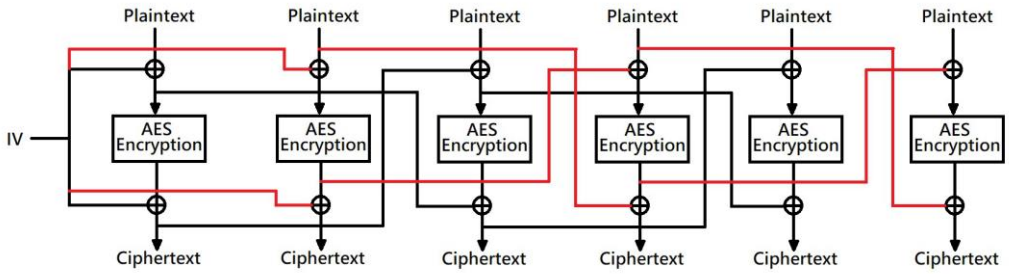
☆ **CBC decryption**

```
def CBC_decrypt(input_byte, key, IV):
    cipher = AES.new(key, AES.MODE_ECB)
    origin_image = b''
    for i in range(0, len(input_byte), 16):
        ciphertext = input_byte[i:i+16]
        block = cipher.decrypt(ciphertext)
        plaintext = byte_xor(IV, block)
        origin_image += plaintext
        IV = ciphertext

    decrypt_image = Image.frombytes("RGB", CBC_encrypt.size, origin_image)
    decrypt_image.save('./CBC_decrypt.png')
```

decryption 為 encryption 的相反，將 ciphertext 放入 cipher block 做解密，出來的結果再和 IV 做 xor，就是 plaintext。由於 ECB 的 Block 之間沒有關係，因此將 encrypt 後的 ciphertext 和 plaintext 交換後，出來的結果就是原來的圖片。

☆ **Cool mode**



```python
def cool(input_byte, key, IV):
    cipher = AES.new(key, AES.MODE_ECB)
    cipher_image = b''
    n = 1
    for i in range(0, len(input_byte), 16):
        odd1 = IV
        odd2 = IV
        even1 = IV
        even2 = IV
        if n%2==1:  # 奇數個block
            plaintext = input_byte[i:i+16]
            xor_in = byte_xor(plaintext, odd1)
            block = cipher.encrypt(xor_in)
            xor_out = byte_xor(block, odd2)
            odd1 = xor_out
            odd2 = xor_in
            cipher_image += xor_out
        else:   # 偶數個block
            plaintext = input_byte[i:i+16]
            xor_in = byte_xor(plaintext, even1)
            block = cipher.encrypt(xor_in)
            xor_out = byte_xor(block, even2)
            even1 = block
            even2 = plaintext
            cipher_image += xor_out
        n+=1

    encrypt_image = Image.frombytes("RGB", input_image.size, cipher_image)
    encrypt_image.save('./cool_encrypt.png')
```

自己設計出的 block cipher 的特色是將 block 分為奇數和偶數個位置。

奇數位的 block，xor_in 和 plaintext IV 做 xor，xor_out 和 block IV 做 xor，接下來下一個 block 的 IV 更新為上一個 block 的 xor_out 和 xor_in。

偶數位的 block，xor_in 和 plaintext IV 做 xor，xor_out 和 block IV 做 xor，接下來下一個 block 的 IV 更新為上一個 block 的 block 和 plaintext

☆ **Cool mode decryption**

```python
def cool_decrypt(input_byte, key, IV):
    cipher = AES.new(key, AES.MODE_ECB)
    origin_image = b''
    n = 1
    for i in range(0, len(input_byte), 16):
        odd1 = IV
        odd2 = IV
        even1 = IV
        even2 = IV
        if n%2==0:  # 偶數個block
            ciphertext = input_byte[i:i+16]
            xor_in = byte_xor(ciphertext, odd1)
            block = cipher.decrypt(xor_in)
            plaintext = byte_xor(block, odd2)
            odd1 = plaintext
            odd2 = xor_in
            origin_image += plaintext
        else:   # 奇數個block
            ciphertext = input_byte[i:i+16]
            xor_in = byte_xor(ciphertext, even1)
            block = cipher.decrypt(xor_in)
            plaintext = byte_xor(block, even2)
            even1 = block
            even2 = ciphertext
            origin_image += plaintext
        n+=1

    decrypt_image = Image.frombytes("RGB", cool_encrypt.size, origin_image)
    decrypt_image.save('./cool_decrypt.png')
```

decryption 為 encryption 的相反，奇數位的 block 的 IV 更新為上一個 block 的 block 和 ciphertext，偶數位的 block 的 IV 更新為上一個 block 的 plaintext 和 xor_in。

- **心得與遇到的困難**

一開始操作的時候比較不知道檔案的讀取跟放進 AES 的格式，不斷把 bytes 轉成 bit 來操作，但是後來發現不需要這麼麻煩，可以用 bytes 直接運算。

還有原本做 padding 是想用 Crypto.Util.Padding 的 pad，但是不知道為什麼會出現錯誤，之後改成自己寫 pad 的 function，在 bytes 存成圖片的部分和第三題題目設計也花蠻多時間查資料。