四資工三甲 B10715029 陳彥瑋

IS hw3 說明文件

1. 分工

B10715029 陳彥瑋（我） ：所有的解密部分
B10715029 馬孝傑 ：所有的加密部分

2. 建置環境 / 依賴套件

測試環境 ：Ubuntu 20.04.2 LTS
Python 版本 ：3.7.10
依賴套件 ：Pillow (7.1.2)
pycryptodome (3.10.1)
numpy (1.19.5)
io (原生函式庫)
os (原生函式庫)
sys (原生函式庫)
typing (原生函式庫)

3. 操作方式

安裝依賴套件：make install
加密 ：make enc
解密 ：make dec

加密程式執行：python3 enc.py -key <自訂 key> -iv <自訂 iv>

- 此指令會將 linux.jpeg 使用指定的 key 和 iv 以 ECB, CTR, 自訂方法 (PCBC) 加密後分別儲存到 test_enc 資料夾下的 ECB.png, CTR.png, Custom.png

解密程式執行：python3 dec.py --key <自訂 key> --iv <自訂 iv> --mode <加密模式>

- 此指令會將 test_enc 資料夾下的 <加密模式>.png 使用指定的 key 和 iv 以自訂模式解密後儲存到 test_dec 資料夾下的 <加密模式>.png
- 模式僅支援 ECB, CTR, Custom 三種

4. 程式碼解說

```
94  def main():
95      mode = "Custom"
96      key = b'\xd5\x07V\x148\xd4\xa78r\xf5\x05\x8a\xad\xf2}]'
97      iv = b'\xa8\xf4\x07t\x83j,\xceK^\x97/\xb2\x1c\x16\xec'
98
99      for i in range(1, len(sys.argv)):
100         if sys.argv[i] == '--iv':
101             i = i + 1
102             #check if the iv and key length is correct
103             if len(sys.argv[i]) == AES.block_size:
104                 iv = str.encode(sys.argv[i])
105         elif sys.argv[i] == '--key':
106             i = i + 1
107             if len(sys.argv[i]) == AES.block_size:
108                 key = str.encode(sys.argv[i])
109         elif sys.argv[i] == "--mode":
110             i += 1
111             mode = sys.argv[i]
112
113     imgPath = "./test_enc/" + mode + ".png"
114     savePath = "./test_dec/" + mode + ".png"
115
116     height, width, data = imgToBytes(imgPath, maxLen=16)
117     if(mode == "ECB"):
118         dataDec = ECB_decrypt(data, key)
119     elif(mode == "CTR"):
120         dataDec = CTR_decrypt(data, key, iv)
121     elif(mode == "Custom"):
122         dataDec = PCBC_decrypt(data, key, iv)
123
124     if(not os.path.exists('./test_dec')):
125         os.makedirs('./test_dec')
126
127     bytesToPng(savePath, dataDec, height, width)
128
129 if __name__ == "__main__":
130     main()
```

main 為主函式，在第 129 ~ 130 行表示當程式被啟動時就執行 main 函式。

在第 95~ 111 行為設定 mode, key, iv 參數，第 95 ~ 97 行為預設的參數，而第 99 ~ 111 行是將使用者在命令列設定的參數讀入並設定。

第 113 ~ 114 行是設定輸入的圖片位置和要儲存的解密圖片的位置。

第 116 行會呼叫 imgToBytes 函式讀取圖片及將圖片做分段。

第 117 ~ 122 行會根據 mode 選擇用不同的解密方式做解密。

第 124 ~ 125 行會檢查 test_dec 這個路徑是否存在，如果不存在則創建。

第 127 行會將解密後的資料轉換成圖片並儲存。

```python
def imgToBytes(path:str, maxLen:int=16) -> Tuple[int, int, List[bytes]]:
    retBytes = []

    img = Image.open(path)
    imgArray = np.asarray(img)

    imgH, imgW = img.height, img.width
    temp = []
    for i in range(imgH):
        for j in range(imgW):
            pixel = imgArray[i,j]
            for c in range(len(pixel)):
                temp.append(pixel[c])

            if(len(temp) >= maxLen):
                seg = temp[:maxLen]
                retBytes.append(bytes(seg))
                temp = temp[maxLen:]

    if(len(temp) != 0):
        while(len(temp) < maxLen):
            temp.append(0) # pad 0 at tail

        retBytes.append(bytes(temp))
        temp.clear()
    return (imgH, imgW, retBytes)
```

imgToBytes 函式會讀取一張圖片，然後切割成固定長度的 bytes 資料，在回傳時除了會回傳資料外也會回傳圖片的長寬資訊。

第 10 ~ 11 行為讀取圖片及將圖片的 RGB 資料值存進 numpy 的 ndarray 中，shape 為 (height, width, channel)。

第 14 ~ 31 行會將資料值以每 maxLen 長度做一次切割，這邊不會考慮到 RGB 的通道而是直接切割。

第 26 ~ 31 行主要是預防圖片資料會有剩下一點點不夠做切割，這邊 padding 的方式是直接補 0 在最後面。

```
64  def _AESEncBlock(plain: bytes, key: bytes) -> bytes:
65      assert(len(plain) == 16)
66      cipher = AES.new(key, AES.MODE_ECB)
67      return cipher.encrypt(plain)
68  def _AESDecBlock(c: bytes, key: bytes) -> bytes:
69      assert(len(c) == 16)
70      cipher = AES.new(key, AES.MODE_ECB)
71      return cipher.decrypt(c)
72  def ECB_decrypt(cipherData:List[bytes], key:bytes) -> List[bytes]:
73      plainData = []
74      for segment in cipherData:
75          plainData.append(_AESDecBlock(segment, key))
76      return plainData
77  def CTR_decrypt(cipherData:List[bytes], key:bytes, iv:bytes) -> List[bytes]:
78      plainData = []
79      for segment in cipherData:
80          civ = _AESEncBlock(iv, key)
81          plainSeg = bytes([b1 ^ b2 for b1, b2 in zip(segment, civ)])
82          plainData.append(plainSeg)
83          iv = (int.from_bytes(iv, "big") + 1).to_bytes(16, "big")  # iv += 1
84      return plainData
85  def PCBC_decrypt(cipherData:List[bytes], key:bytes, iv:bytes) -> List[bytes]:
86      plainData = []
87      for segment in cipherData:
88          pxiv = _AESDecBlock(segment, key)
89          plainSeg = bytes([b1 ^ b2 for b1, b2 in zip(pxiv, iv)])
90          plainData.append(plainSeg)
91          iv = bytes([b1 ^ b2 for b1, b2 in zip(segment, plainSeg)])
92      return plainData
```

第 64 ~ 92 行是加密用的一些函式， _AESEncBlock 和 _AESDecBlock 是一個簡單的 AES 加解密的模組，輸入長度只會是單一個 segment 的長度（=16Bytes =128bits）。

ECB_decrypt, CTR_decrypt, PCBC_decrypt 則是對應模式的解密函式，其中在第 81, 89, 91 行的$\text{bytes}([b1\ \hat{}\ b2\ \text{for}\ b1, b2\ \text{in}\ \text{zip}(<A>, <B>)])$ 代表的是對 <A> 和 <B> 做 XOR 的操作；而在第 83 行的 $(int.from\_bytes(iv, "big") + 1).to\_bytes(16, "big")$ 則是做 iv+1 的動作。

其中例外一個要關注的點是在做 CTR 解密時，所使用到的 AES 操作是加密而其它的是解密，如第 80 行所示。

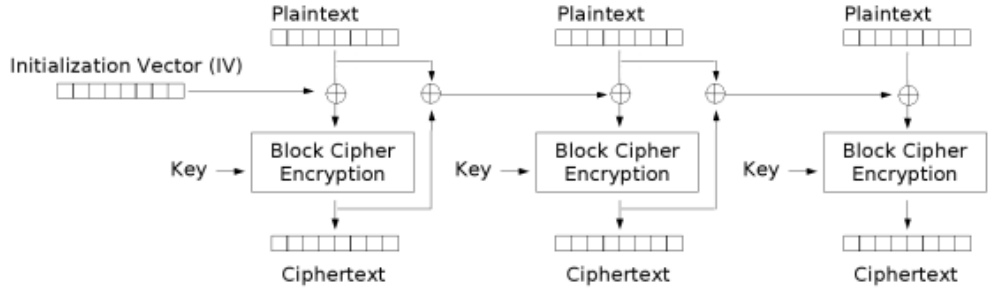
```python
def _bytesToNpArray(data:List[bytes], height:int, width:int) -> np.ndarray:
    imgData = np.ndarray((height, width, 3), dtype=np.uint8)
    h = 0
    w = 0
    ch = 0
    end = False
    for segment in data:
        for byte in segment:
            imgData[h,w,ch] = byte
            ch += 1
            if(ch >= 3):
                w += 1
                ch = 0
            if(w >= width):
                h += 1
                w = 0
            if(h >= height):
                end = True
                break
        if(end):
            break
    return imgData

def bytesToPng(path:str, data:List[bytes], height:int, width:int) -> np.ndarray:
    imgArray = _bytesToNpArray(data, height, width)
    img = Image.fromarray(imgArray)
    img.save(path, "png")
    return imgArray
```

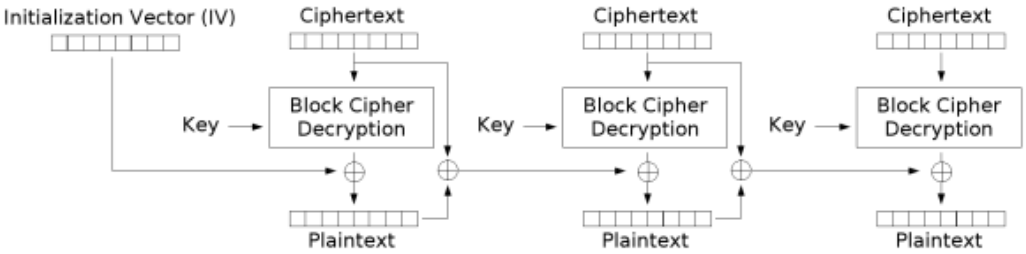
最後的部分是將解密後的資料轉換回圖片，所使用的函式是 bytesToPng，他首先會把解密後的資料用 _bytesToNpArray 這個函式轉換成 PIL 接受的 ndarray 格式，再透過 Image.fromarray 轉換成 PIL 圖片，就可以做儲存了。

而 _bytesToNpArray 的內容中，它會一直去讀入 data 在每個 segment 的每個位置的資料，並儲存在 imgData 中，最後傳回。

5. 自行設計的 Block Cipher Operation 的架構圖



Propagating Cipher Block Chaining (PCBC) mode encryption



Propagating Cipher Block Chaining (PCBC) mode decryption

圖片來源：維基百科

6. 遇到困難與心得

困難：

Debug 花很久時間

圖片資料的轉換很麻煩

心得：

在解密部分的 debug 時，可以先自己單獨寫加密和解密的程式，如果加密後解密不出來的話可以使用組員的加密程式也做一次加密，再使用線上的圖片差異辨識程式比較自己和組員加密出來的圖片有沒有差異，有的話可以判斷是自己寫的加密部分程式出現問題，如果圖片沒問題的話則是解密的程式有問題。如此一來就可以縮小要 debug 的範圍，更容易找出錯誤。

如果自己能解密自己寫的加密但解不出組員的加密圖片的話，多半是因為圖片的前處理不一致（如一個是直接切 16Byte 而另一個有將 RGB 通道切開分別加密），或是一些參數如 key 和 iv 設定有問題。