

Block Cipher Operation

分工

組員A: B10730007 楊芯蘋

組員B: B10730027 曾瑄翎

	ECB	CTR	Custom
組員A		V	V
組員B	V		V
說明	前期共同討論, 過程中互相幫忙解決問題, 最後一起整合		

建置環境 / 依賴套件

- **Pillow**
將圖片轉為 ppm 格式
- **pycryptodome**
使用AES進行加密
- **bitstring**

操作方式

Make

```
make install  
make enc
```

```
make dec
```

Input & Output

加密

```
./enc.py {filename}.jpeg {mode}(ECB,CTR,CUSTOM)
```

```
./enc.py linux.jpeg ECB  
./enc.py linux.jpeg CTR  
./enc.py linux.jpeg CUSTOM
```

加密完成的圖片將存放在test_enc目錄

解密

```
./dec.py {mode}(ECB,CTR,CUSTOM)
```

```
./dec.py ECB  
./dec.py CTR  
./dec.py CUSTOM
```

解密完成的圖片將存放在test_dec目錄

程式碼解說

加密大綱


> 輸入原始圖片、模式

```
11  pic_file = sys.argv[1]
12  mode = sys.argv[2]
```

> 將原始圖片轉成ppm

```
14  # Convert to ppm
15  ppmPicture = "./input_enc.ppm"
16  im = Image.open(sys.argv[1])
17  im.save(ppmPicture)
18  tmpPic = "./tmp.ppm"
```

> 加密模式定義


```
>  ecb
>  ctr
>  custom
```

解密大綱

> 輸入模式

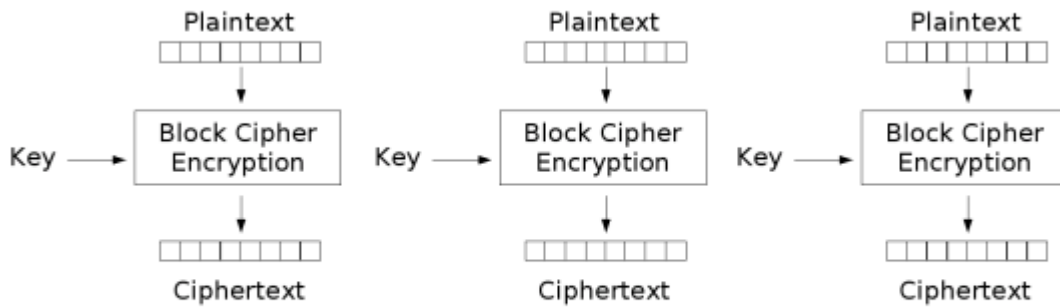
```
11  pic_file = sys.argv[1]
12  mode = sys.argv[1]
```

> 解密模式定義

```
>  ecb
>  ctr
>  custom
```

ECB 模式

加密



Electronic Codebook (ECB) mode encryption

1. 設定輸入和輸出檔案、key、AES cipher、Padding

```

23     # Read ppm as a binary file
24     bin_data = open(ppmPicture, 'rb').read()
25
26     t = open(tmpPic, "wb")
27
28     key = b'MORECOFFEEPLEASE'
29     cipher = AES.new(key, AES.MODE_ECB)
30
31     # Add null bytes
32     while (len(bin_data) % 16 != 0):
33         bin_data += b'\x00'
34
35     t.write(bin_data)
36     # clear output
37     output = open("out_ecb_enc.ppm", "wb")
38     output.close()
39
40     output = open("out_ecb_enc.ppm", 'ab')
41     line_count = 0

```

2. 對每塊明文加密, 寫檔

```

43     with open(tmpPic, 'rb') as f:
44         while True:
45             buf = f.read(16)
46             if line_count == 0:
47                 msg = buf
48             else:
49                 #對每塊明文加密
50                 msg = cipher.encrypt(buf)
51             output.write(msg)
52             line_count += 1
53             if not buf:
54                 break

```

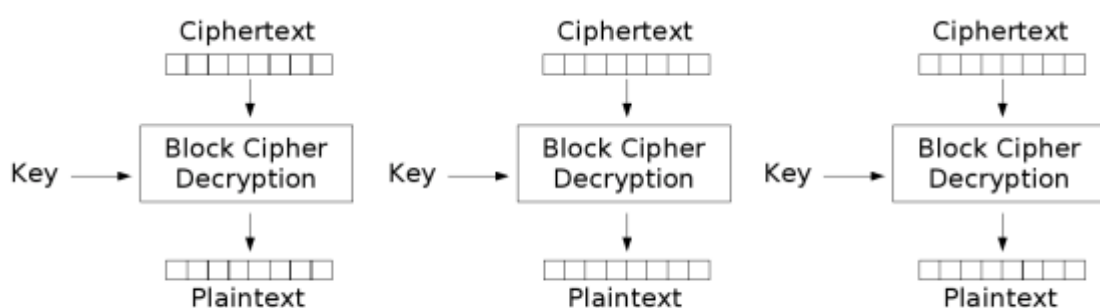
3. 將ppm轉成jpeg, 存檔

```

56     # Convert to jpeg
57     Picture = "./out_ecb_enc.ppm"
58     im = Image.open(Picture)
59     im.save("./test_enc/ECB.jpeg")

```

解密



Electronic Codebook (ECB) mode decryption

1. 設定輸入和輸出檔案、key、AES decipyer

```

16     input = "./out_ecb_enc.ppm"
17
18     output = open("out_ecb_dec.ppm", "wb")
19     output.close()
20     output = open("out_ecb_dec.ppm", 'ab')
21     line_count = 0
22     key = b'MORECOFFEEPLEASE'
23     decipher = AES.new(key, AES.MODE_ECB)

```

2. 對每塊密文解密，寫檔

```

25     with open(input, 'rb') as f:
26         while True:
27             buf = f.read(16)
28             if line_count == 0:
29                 msg = buf
30             else:
31                 #對每塊密文解密
32                 msg = decipher.decrypt(buf)
33             output.write(msg)
34             line_count += 1
35             if not buf:
36                 break

```

3. 將ppm轉成jpeg, 存檔

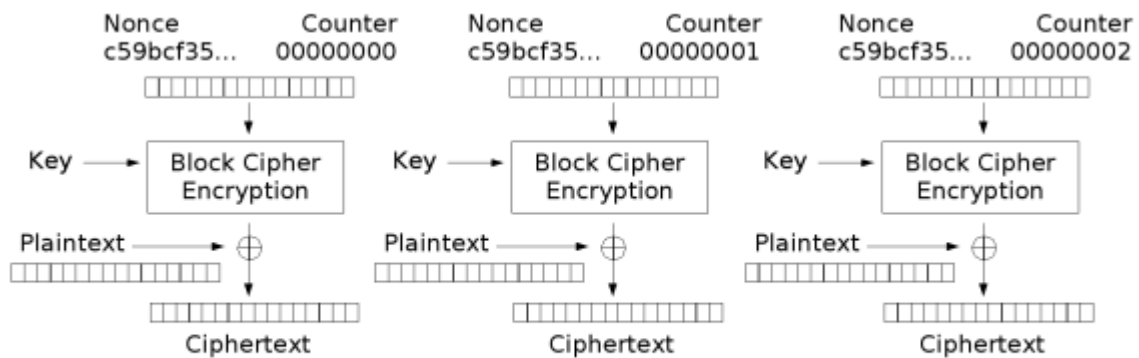
```

38     # Convert to jpeg
39     Picture = "./out_ecb_dec.ppm"
40     im = Image.open(Picture)
41     im.save("./test_dec/ECB.jpeg")

```

CTR 模式

加密



Counter (CTR) mode encryption

1. 定義functions: make_iv(合併nonce和counter)、bxor

```

72  ✓   def make_iv(n, cnt):
73       tmp_iv = bytearray(16)
74  ✓   for i in range(16):
75  ✓       if i < 8:
76           tmp_iv[i] = n[i]
77  ✓       else:
78           tmp_iv[i] = cnt[i-8]
79       return tmp_iv
80
81       # xor for bytes
82  ✓   def bxor(b1, b2):
83       result = b""
84  ✓   for b1, b2 in zip(b1, b2):
85       result += bytes([b1 ^ b2])
86       return result
87

```

2. 設定vector、nonce、counter

```

89       # create a 16 byte array to hold IV
90       initial_vector = bytearray(16)
91
92       # Randomly generate a nonce
93       nonce = b',\xd5{\xe95\x86B\xa1'
94
95       # Initialize counter
96       counter = 0
97       counter_bytes = counter.to_bytes(8, 'big')
98
99       # Initialize nonce
100      initial_vector = make_iv(nonce, counter_bytes)
101

```

3. 明文切塊

```

102      # divide plaintext into 16-byte blocks
103      plain_text_blocks = []
104  ✓   with open(tmpPic, 'rb') as f:
105       while True:
106           buf = f.read(16)
107           if buf != b'':
108               plain_text_blocks.append(buf)
109           if not buf:
110               break

```

4. 設定key、AES cipher、輸出檔

```

113     key = b'MORECOFFEEPLEASE'
114     cipher = AES.new(key, AES.MODE_ECB)
115     output = open("out_ctr_enc.ppm", "wb")
116     output.close()
117     output = open("out_ctr_enc.ppm", "ab")

```

5. 對每塊加密vector後和明文xor, 得到密文, 寫檔

```

118     for i in range(len(plain_text_blocks)):
119         if i == 0:
120             output.write(plain_text_blocks[i])
121         else:
122             msg = cipher.encrypt(initial_vector)
123             msg_bin = BitArray(hex=msg.hex()).bin
124             cipher_text_block = bxor(msg, plain_text_blocks[i])
125             output.write(cipher_text_block)
126             counter += 1
127             initial_vector = make_iv(nounce, counter.to_bytes(8, 'big'))

```

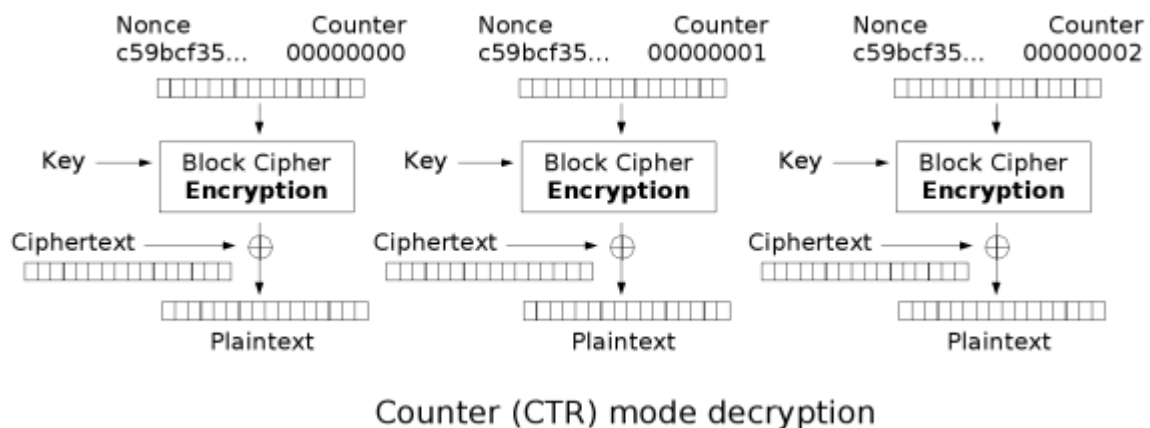
6. 將ppm轉成jpeg, 存檔

```

129     # Convert to jpeg
130     Picture = "./out_ctr_enc.ppm"
131     im = Image.open(Picture)
132     im.save("./test_enc/CTR.jpeg")

```

解密



1. 設定輸入和輸出檔案、key、AES cipher、nonce、counter

```

46     input = "./out_ctr_enc.ppm"
47
48     output = open("out_ctr_dec.ppm", "wb")
49     output.close()
50     output = open("out_ctr_dec.ppm", 'ab')
51     line_count = 0
52     key = b'MORECOFFEEPLEASE'
53     nonce = b'\xd5{\xe95\x86B\xa1'
54     cipher = AES.new(key, AES.MODE_ECB)
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78     # Initialize counter
79     counter = 0
80     counter_bytes = counter.to_bytes(8, 'big')

```

2. 定義functions: make_iv(合併nonce和counter)、bxor

```

57     def make_iv(n, cnt):
58         tmp_iv = bytearray(16)
59         for i in range(16):
60             if i < 8:
61                 tmp_iv[i] = n[i]
62             else:
63                 tmp_iv[i] = cnt[i-8]
64         return tmp_iv
65
66     # xor for bytes
67     def bxor(b1, b2):
68         result = b""
69         for b1, b2 in zip(b1, b2):
70             result += bytes([b1 ^ b2])
71         return result

```

3. 對每塊加密vector後和密文xor, 得到明文, 寫檔

```

82     initial_vector = make_iv(nounce, counter_bytes)
83     with open(input, 'rb') as f:
84         while True:
85             buf = f.read(16)
86             if counter == 0:
87                 output.write(buf)
88             else:
89                 msg = cipher.encrypt(initial_vector)
90                 decipher_text = bxor(msg, buf)
91                 output.write(decipher_text)
92             counter += 1
93             initial_vector = make_iv(nounce, counter.to_bytes(8, 'big'))
94             if not buf:
95                 break

```

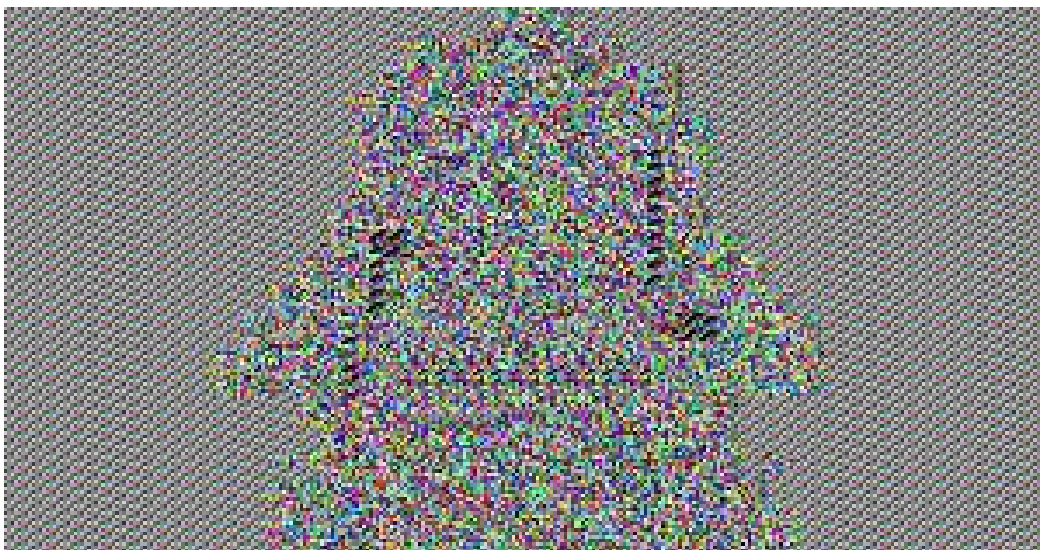
4. 將ppm轉成jpeg, 存檔

```

97     # Convert to jpeg
98     Picture = "./out_ctr_dec.ppm"
99     im = Image.open(Picture)
100    im.save("./test_dec/CTR.jpeg")

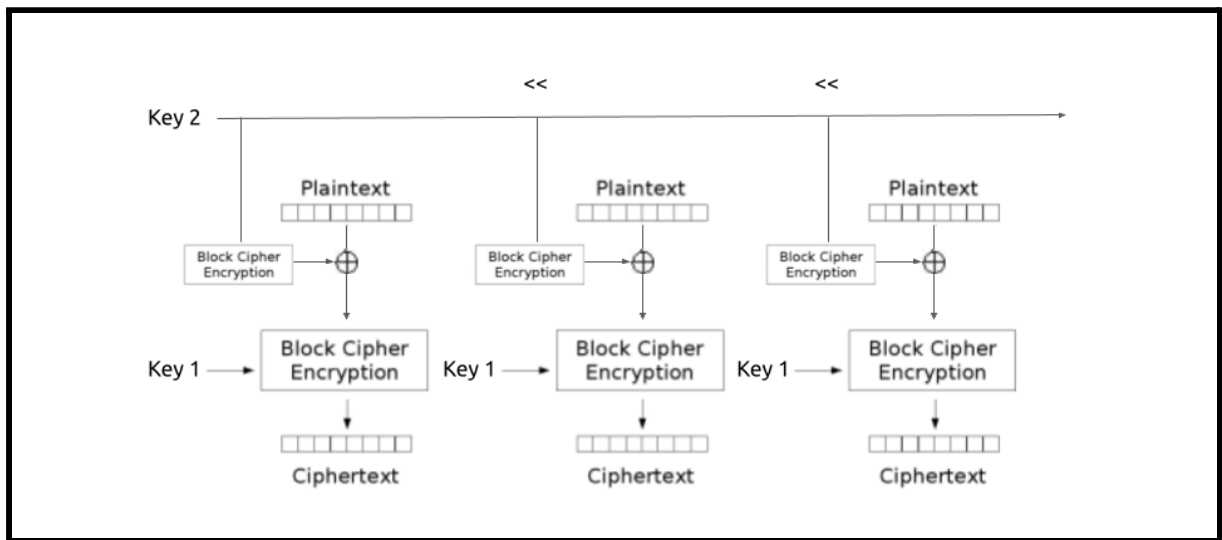
```

CUSTOM 模式(自行設計)

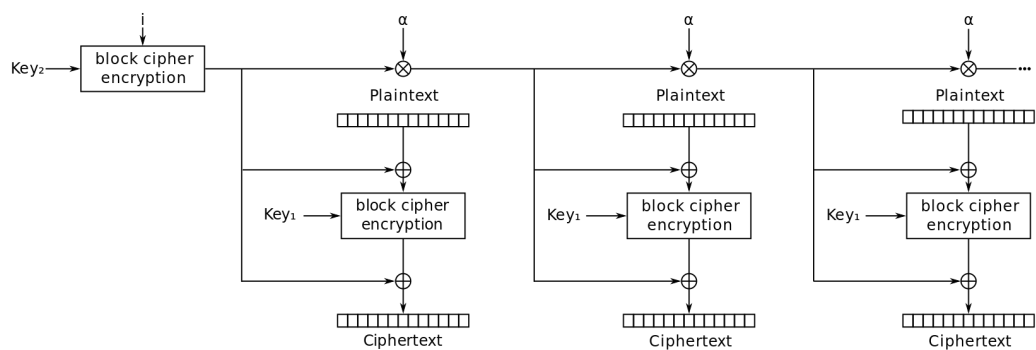


ECB模式下, 同樣的明文塊會被加密成相同的密文塊, 加密後(上圖)仍然可以看出原圖輪廓, 無法提供嚴格的資料保密性, 於是我們決定設計新的模式改善這點。

加密



參考XTS模式的作法, 如下圖



加入key2並持續更新key2(我們採用循環左移的方式)、加密key2, 將加密後的key2和每塊明文進行xor, 再對處理過後的明文加密, 得到具偽隨機性的密文(見下圖)。



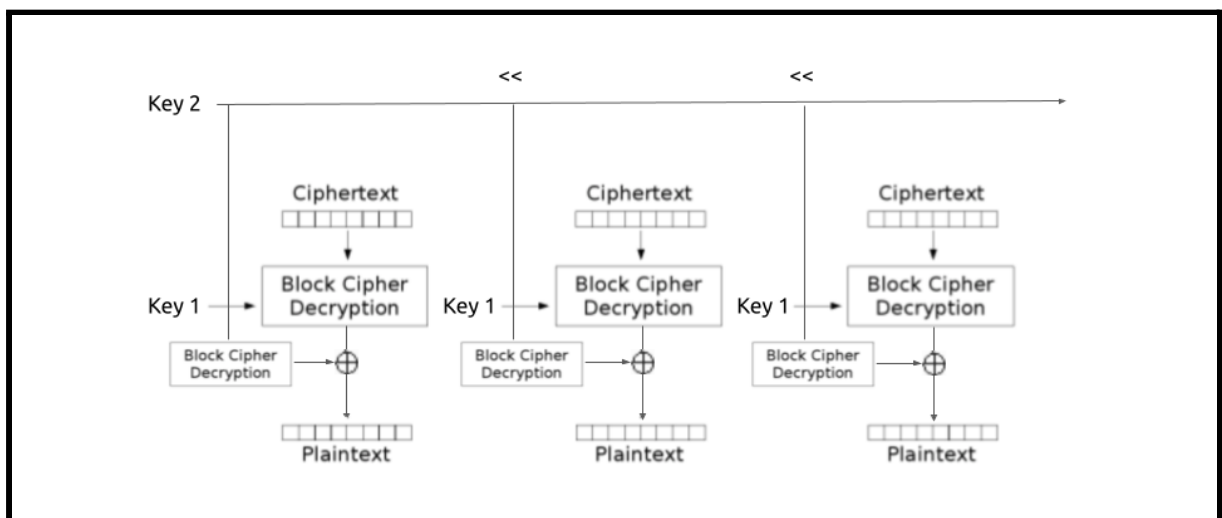
實作如下：

```

171     with open(tmpPic, 'rb') as f:
172         while True:
173             buf = f.read(16)
174             if line_count == 0:
175                 msg = buf
176             else:
177                 #對key2加密
178                 key2 = cipher.encrypt(key2)
179                 #加密後的key2和明文xor
180                 buf = byte_xor(buf, key2)
181                 #對處理後的明文加密
182                 msg = cipher.encrypt(buf)
183                 #key2循環左移
184                 key2 = make_key2(key2)
185             output.write(msg)
186             line_count += 1
187             if not buf:
188                 break

```

解密



解密過程和加密類似，一樣持續更新key2(循環左移)、加密key2，先將密文解密，再與加密後的key2進行xor，得到明文。

實作如下：

```

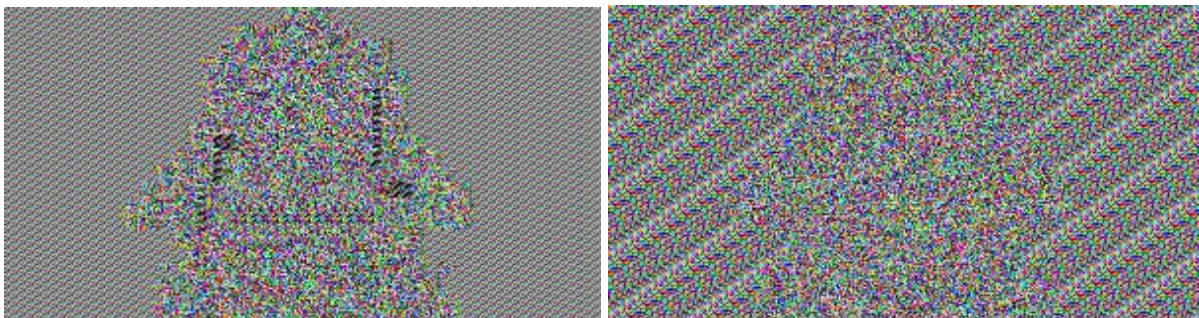
128     with open(input, 'rb') as f:
129         while True:
130             buf = f.read(16)
131             if line_count == 0:
132                 msg = buf
133             else:
134                 #解密key2
135                 key2 = decipher.encrypt(key2)
136                 #解密密文
137                 msg = decipher.decrypt(buf)
138                 #兩者xor
139                 msg = byte_xor(msg, key2)
140                 #key2循環左移
141                 key2 = make_key2(key2)
142             output.write(msg)
143             line_count += 1
144             if not buf:
145                 break

```

困難與心得

剛開始嘗試改善ECB的的時候加入了key2的概念，使用循環左移更新key2，但結果不盡理想，讓我們有點灰心，持續腦力激盪才得到CUSTOM最終版本。

左為ECB加密結果、右為初版CUSTOM加密結果：



初版架構如下：

