網路安全概論

期中專題報告

DES 實作

系級：電機系統三

姓名：B0721251 楊仁傑

日期：2021/05/20

壹、DES 原理：

DES 加密算法主要可以分為以下四個步驟：

（1）初始置換

（2）生成子密鑰

（3）迭代過程

（4）逆置換

一、初始置換(IP 置換)：

初始替換是將原始明文通過 IP 置換表處理。IP 置換表如下：

```
# 初始置換表
IP = [58, 50, 42, 34, 26, 18, 10, 2,
      60, 52, 44, 36, 28, 20, 12, 4,
      62, 54, 46, 38, 30, 22, 14, 6,
      64, 56, 48, 40, 32, 24, 16, 8,
      57, 49, 41, 33, 25, 17, 9,  1,
      59, 51, 43, 35, 27, 19, 11, 3,
      61, 53, 45, 37, 29, 21, 13, 5,
      63, 55, 47, 39, 31, 23, 15, 7]
```

圖 1. IP 置換表

其中，IP 置換表中的數字指的是指原文位置，例如 58 指將 M 第 58 位放置第 1 位。

例如：

輸入 64 位明文數據 M（64 bits）：

明文 M（64 bits）=

0110001101101111010110110101110000011101010111010001100101011100 10

將 M 經過 IP 置換後為 M’

M’（64 bits）=

1111111110111000011101100101011100000000111111110000011010000011

二、生成子密鑰

在 DES 加密中會執行 16 次迭代，每次重複過程的數據長度為 48bits，因此需要 16 個 48bits 的子密鑰來進行加密，生成子密鑰的過程如下：

I. 第一輪置換：

A. PC1 置換：

密鑰 K = 0001001100110100010101110111100110011011101111001101111111110001 經過 PC-1 表置換，可得到 K'。

```
# 金鑰置換表，將64位金鑰變成56位
PC_1 = [57, 49, 41, 33, 25, 17, 9,
        1,  58, 50, 42, 34, 26, 18,
        10, 2,  59, 51, 43, 35, 27,
        19, 11, 3,  60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7,  62, 54, 46, 38, 30, 22,
        14, 6,  61, 53, 45, 37, 29,
        21, 13, 5,  28, 20, 12, 4]
```

圖 2. PC1 置換表

K'（56 位）= 11110000110011001010101011111010101010110011001111100011111

取 K'的前 28 位作為 C0，且取 K'的後 28 位作為 D0，，則有

C0（28 位）= 11110000110011001010101011111

D0（28 位）= 01010101011001100111100011111

B. leftShift 置換：

生成 C0，D0 後進行左移操作，需要查詢移動位數表：

每輪移動移動位數表如下：

```
# 每輪左移的位數
shiftBits = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
```

圖 3. 左移置換表

進行第一輪移位，輪數為 1，查表得左移位數為 1。

C0 左移 1 位為 C1，且 D0 左移 1 位為 D1：

C1（28 位）= 11100001100110010101010111111

D1（28 位）= 10101010110011001111100011110

C. PC2 置換：

將 C1 和 D1 合併後，經過 PC-2 表置換得到子密鑰 K1。

由於 PC-2 表為 6x8 的表，經 PC-2 置換後的數據為 48 位，置換後得到密鑰 K1，

```
# 壓縮置換，將56位金鑰壓縮成48位子金鑰
PC_2 = [14, 17, 11, 24, 1,  5,
        3,  28, 15, 6,  21, 10,
        23, 19, 12, 4,  26, 8,
        16, 7,  27, 20, 13, 2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32]
```

圖 4. PC2 置換表

K1（48 位）= 000110110000001011101111111111000111000001110010

II. 第二輪置換

A. leftShift 置換：

C1 和 D1 再次左移，輪數 = 2，查表左移位數 = 1，則 C1 和 D1 左移 1 位得到 C2 和 D2。

C2（28 位）= 1100001100110010101010111111

D2（28 位）= 0101010110011001111000111101

B. PC2 置換：

C2 和 D2 合併後為 56 位，經過 PC-2 表置換得到密鑰 K2（48 位）

K2（48 位）= 011110011010111011011001110110111100100111100101

重複進行加密動作，即可得到 K3-K16 子密鑰，且須注意 Ci 和 Di 左移的位數。

C3（28 位） = 0000110011001010101011111111

D3（28 位） = 0101011001100111100011110101

K3（48 位） = 010101011111110010001010010000101100111110011001

C4（28 位） = 0011001100101010101111111100

D4（28 位） = 0101100110011110001111010101

K4（48 位） = 011100101010110111010110110110011010100011101

C5（28 位） = 1100110010101010111111110000

D5（28 位） = 0110011001111000111101010101

K5（48 位） = 011111001110110000000111111010110101001110101000

C6（28 位） = 001100101010101111111111000011

D6（28 位） = 100110011110001111010101010101

K6（48 位） = 011000111010010100111110010100000111101100101111

C7（28 位） = 110010101010111111111100001100

D7（28 位） = 011001111000111101010101010110

K7（48 位） = 111011001000010010110111111110110000011000101111100

C8（28 位） = 001010101011111111110000110011

D8（28 位） = 100111100011110101010101011001

K8（48 位） = 111101111000101000111010110000010011101111111011

C9（28 位） = 010101010111111111100001100110

D9（28 位） = 001111000111101010101010110011

K9（48 位） = 111000001101101111101011111011011110011111000000001

C10（28 位） = 010101011111111100000110011001

D10（28 位） = 111100011110101010101011001100

K10（48 位） = 101100011111001101000111101110100100011001001111

C11（28 位） = 010101111111100001100110010 1

D11（28 位） = 110001111010101010110011001 1

K11（48 位） = 001000010101111111010011110111101101001110000110

C12（28 位） = 010111111110000110011001010 1

D12（28 位） = 000111101010101011001100111 1

K12（48 位） = 011101010111000111110101100101000110011111101001

C13（28 位） = 011111111000011001100101010 1

D13（28 位） = 011110101010101100110011110 0

K13（48 位） = 100101111100010111010001111110101011101001000001

C14（28 位） = 111111100001100110010101010 1

D14（28 位） = 111010101010110011001111000 1

K14（48 位） = 010111110100001110110111111100101110011100111010

C15（28 位） = 1111100001100110010101010111

D15（28 位） = 1010101010110011001111000111

K15（48 位） = 101111111001000110001101001111010011111100001010

C16（28 位） = 1111000011001100101010101111

D16（28 位） = 0101010101100110011110001111

K16（48 位） = 110010110011111011000101100001110000101111111010 1

三、Feistel 函式：

Feistel 函式主要可以由下列四個部分組成：

I. 擴展置換 E：

右半部分 Ri 的位數為 32 bits，而密鑰長度 Ki 為 48 bits，為了使 Ri 與 Ki 可以進行 xor 運算，所以需要利用擴展置換表 E 將 Ri 由 32 bits 擴充為 48 bits。

```
# 擴充置換表，將 32bits 擴充至 48bits
E = [32, 1,  2,  3,  4,  5,
     4,  5,  6,  7,  8,  9,
     8,  9,  10, 11, 12, 13,
     12, 13, 14, 15, 16, 17,
     16, 17, 18, 19, 20, 21,
     20, 21, 22, 23, 24, 25,
     24, 25, 26, 27, 28, 29,
     28, 29, 30, 31, 32, 1]
```

圖 5. 擴充置換表 E

L0（32 位） = 11111111101110000111011001010111

R0（32 位） = 00000000111111110000011010000011

R0（32 位）經過擴展置換後變為 48 bits 數據：

E(R0)（48 位） = 100000000001011111111110100000001101010000000110

II. XOR 運算：

若兩輸入相異，輸出為 0；若兩輸出相同，輸出為 0。

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

圖 6. XOR 真值表

將 E(R0) 與 K1 作 XOR 運算：

E(R0) = 100000000001011111111110100000001101010000000110

K1 = 000110110000001011101111111111000111000001110010

E(R0)^K1 = 100110110001010100010001011111001010010001110100

III. S 盒置換：

置換運算由 8 個不同的置換盒（S 盒）完成。每個 S 盒有 6 bits 輸入，4 bits 輸出。運算流程如下：

若 S-盒 1 的輸入為 110111，第一位與最後一位構成 11，十進位值為 3，則對應第 3 行，中間 4 位為 1011 對應的十進位值為 11，則對應第 11 列。查找 S-盒 1 表的值為 14，則 S-盒 1 的輸出為 1110。8 個 S 盒將輸入的 48 位數據輸出為 32 位數據。

```
# S盒，每個S盒是4x16的置換表，6位 -> 4位
S = [
    [
        [14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
        [0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
        [4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
        [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]
    ],
    [
        [15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
        [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
        [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
        [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]
    ],
    [
        [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
        [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
        [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
        [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]
    ],
    [
        [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
        [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
        [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
        [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]
    ],
    [
        [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
        [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
        [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
        [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]
    ],
    [
        [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
        [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
        [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
        [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]
    ],
    [
        [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
        [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
        [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
        [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]
    ],
    [
        [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
        [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
        [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
        [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]
    ]
]
```

圖 7. S 盒置換表

將 E(R0)^K1（48 bits）= 100110110001010100010001011111001010010001110100，通過 S 盒替換得到輸出為 10001011110001000110001011101010（32 bits）。

IV. P 置換：

將 S 盒置換的輸出結果作為 P 盒置換的輸入

```
# P置換，32位 -> 32位
P = [16, 7,  20, 21,
     29, 12, 28, 17,
     1,  15, 23, 26,
     5,  18, 31, 10,
     2,  8,  24, 14,
     32, 27, 3,  9,
     19, 13, 30, 6,
     22, 11, 4,  25]
```

圖 8. P 置換表

將 S 盒輸出 10001011110001000110001011101010（32 bits）經過 P 置換，得到輸出 01001000101111110101010110000001（32 bits）。

綜上所述，將 Feistel 函式重複執行 16 次即可得到 L16 與 R16。

第一次疊代過程 f(R0,K1) = 01001000101111110101010110000001

計算 L1（32 位）= R0 = 00000000111111110000011010000011

計算 R1（32 位）= L0 ^ f(R0,K1) = 10110111000001110010001111010110

經過 16 次疊代後輸出：

L16（32 位）= 00110000100001001101101100101000

R16（32 位）= 10110001011001010011000000011000

四、逆置換(IP^{-1})：

逆置換是初始置換的逆運算。從初始置換規則中可以看到，原始數據的第 1 位置換到了第 40 位，第 2 位置換到了第 8 位。則逆置換就是將第 40 位置換到第 1 位，第 8 位置換到第 2 位。以此類推，逆置換規則表如下：

```
# 結尾置換表
IIP = [40, 8, 48, 16, 56, 24, 64, 32,
       39, 7, 47, 15, 55, 23, 63, 31,
       38, 6, 46, 14, 54, 22, 62, 30,
       37, 5, 45, 13, 53, 21, 61, 29,
       36, 4, 44, 12, 52, 20, 60, 28,
       35, 3, 43, 11, 51, 19, 59, 27,
       34, 2, 42, 10, 50, 18, 58, 26,
       33, 1, 41, 9,  49, 17, 57, 25]
```

圖 9. IP^{-1} 置換表

將 L16 與 R16 構成 64 位數據，經過逆置換表輸出密文為：

L16+R16　：0011000010000100110110110010100010110001011001010011000000011000

Ciphertext　：0101100000001000001100000000101111001101110101100001100001101000

貳、DES 各模式示意圖：

1. ECB 模式：

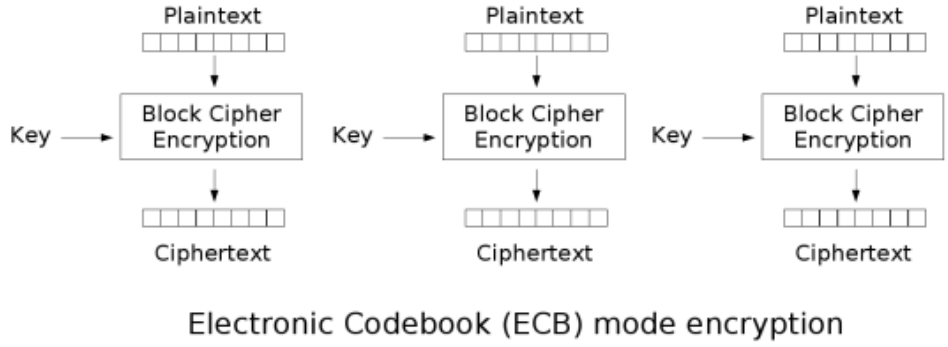


圖 1. ECB 加密模式示意圖

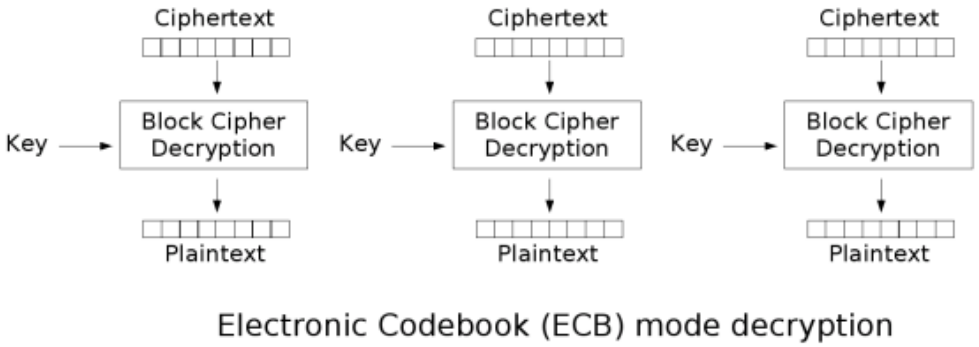


圖 2. ECB 解密模式示意圖

2. CBC 模式：

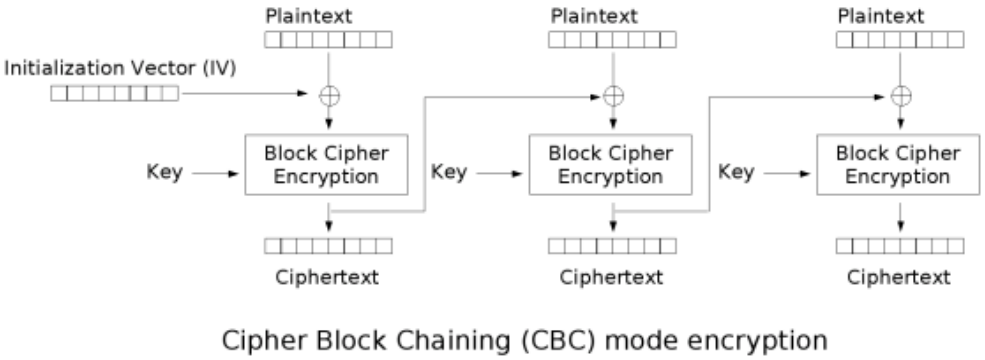


圖 3. CBC 加密模式示意圖

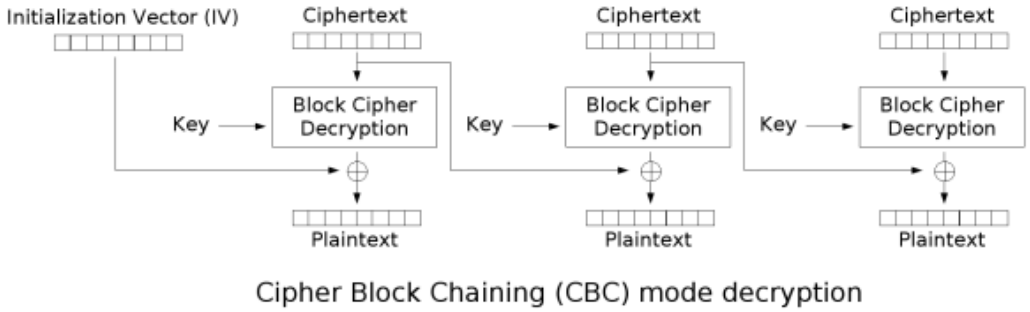


圖 4. CBC 解密模式示意圖

3. CFB 模式：

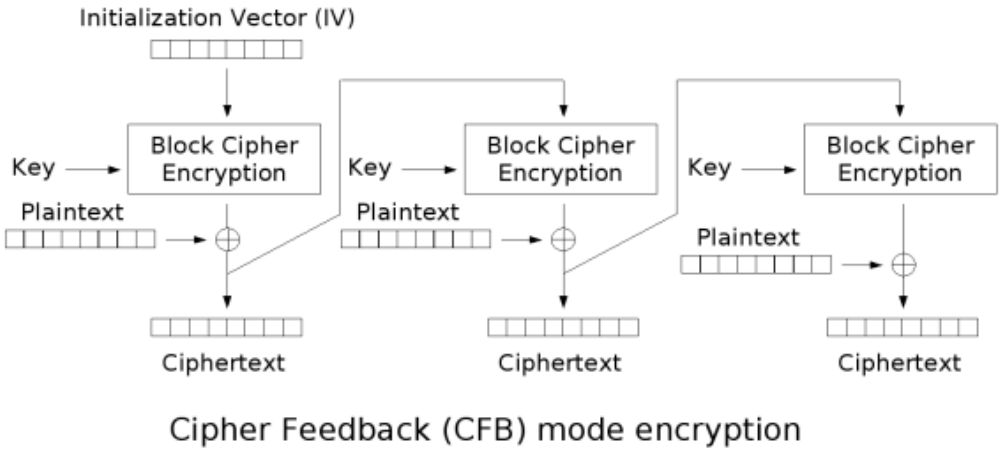


圖 5. CFB 加密模式示意圖

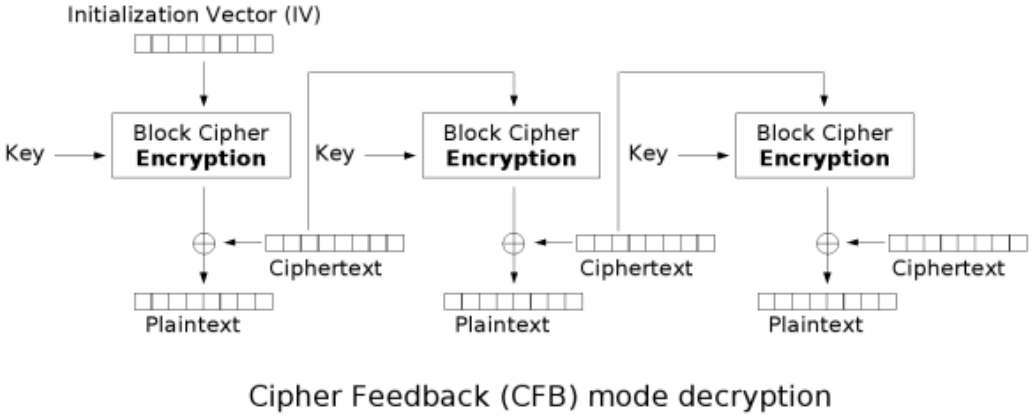


圖 6. CFB 解密模式示意圖

4. OFB 模式：

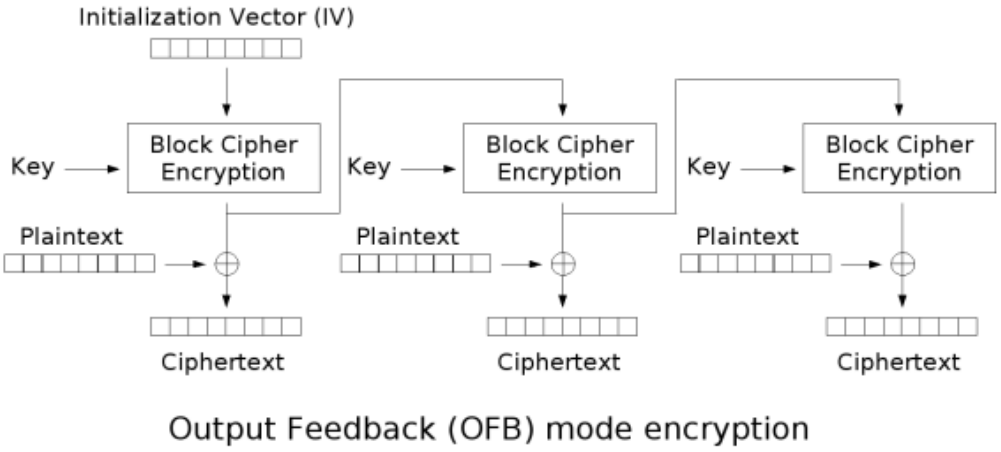


圖 7. OFB 加密模式示意圖

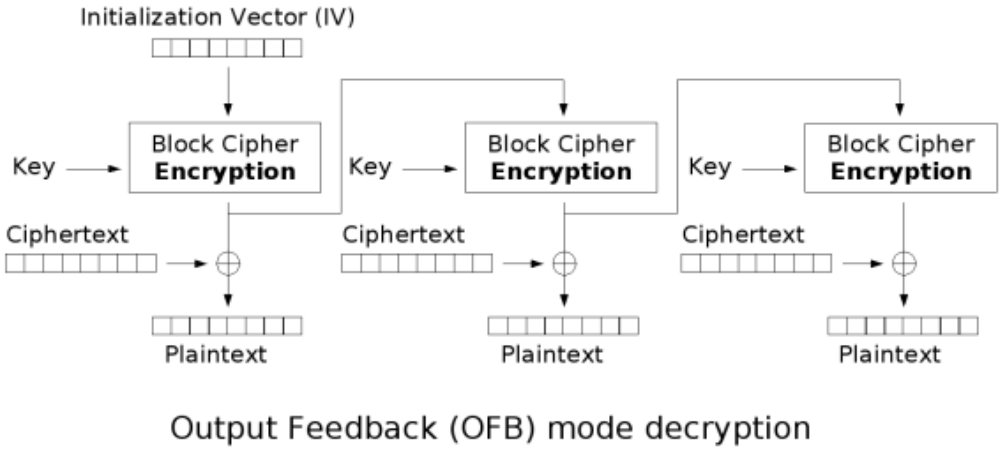


圖 8. OFB 解密模式示意圖

5. CTR 模式：

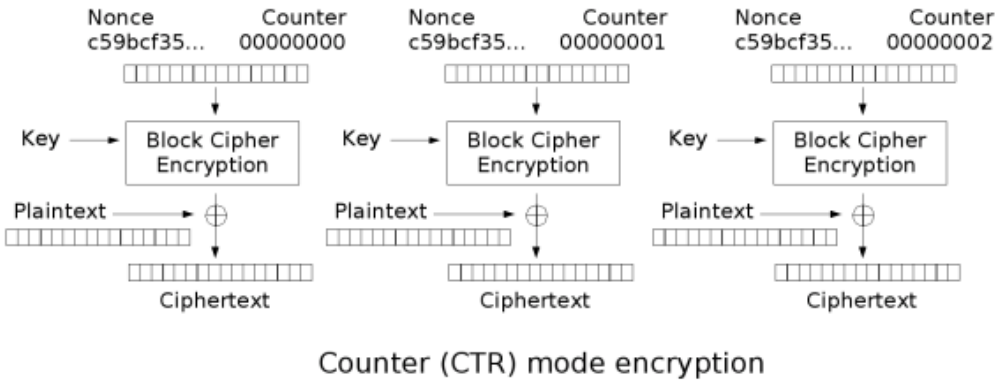


圖 9. CTR 加密模式示意圖

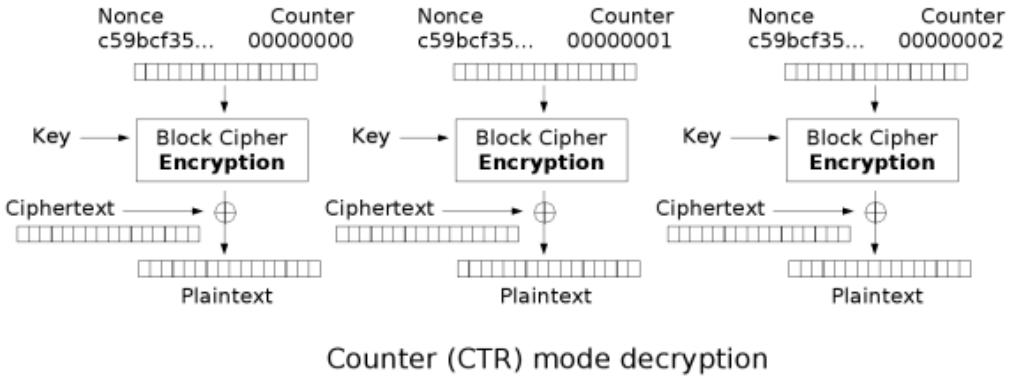
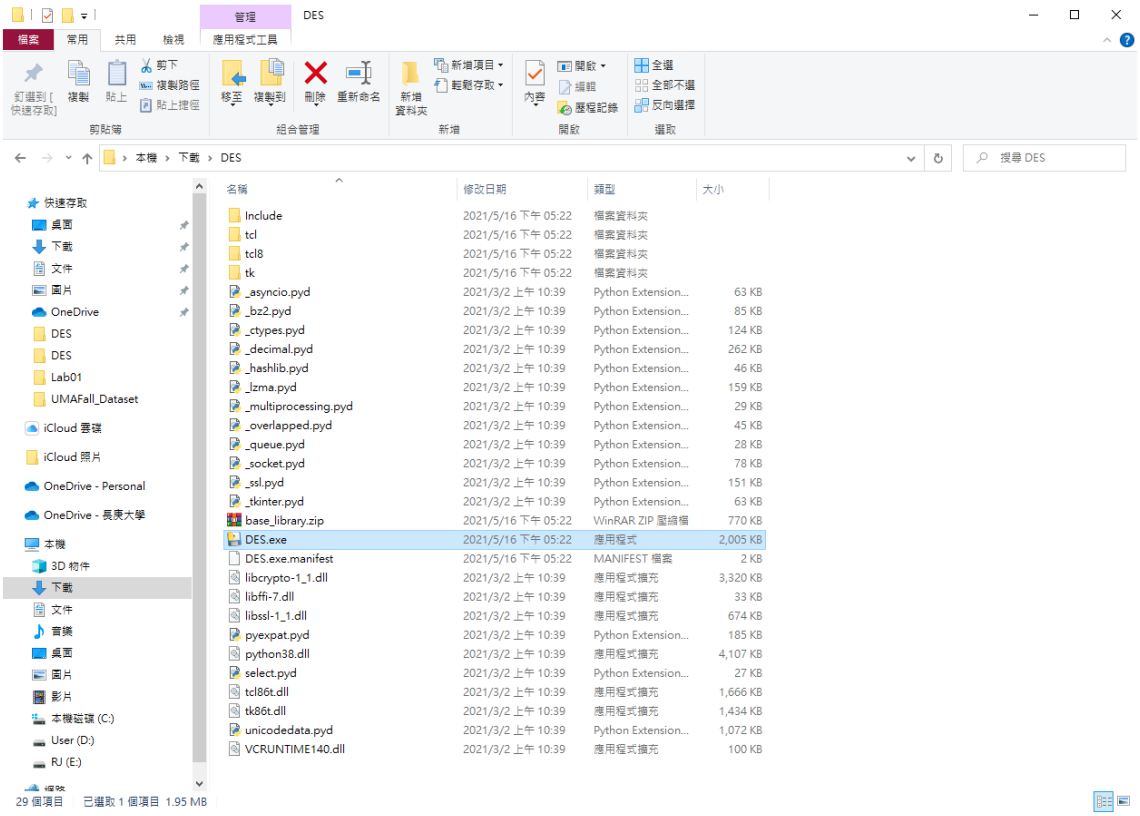


圖 10. CTR 解密模式示意圖

參、實作操作：

1. 將 DES.rar 解壓縮後，選取資料夾內的 DES.exe。



2. 打開 DES.exe 並選取特定檔案後即可使用。

3. 加密功能展示：

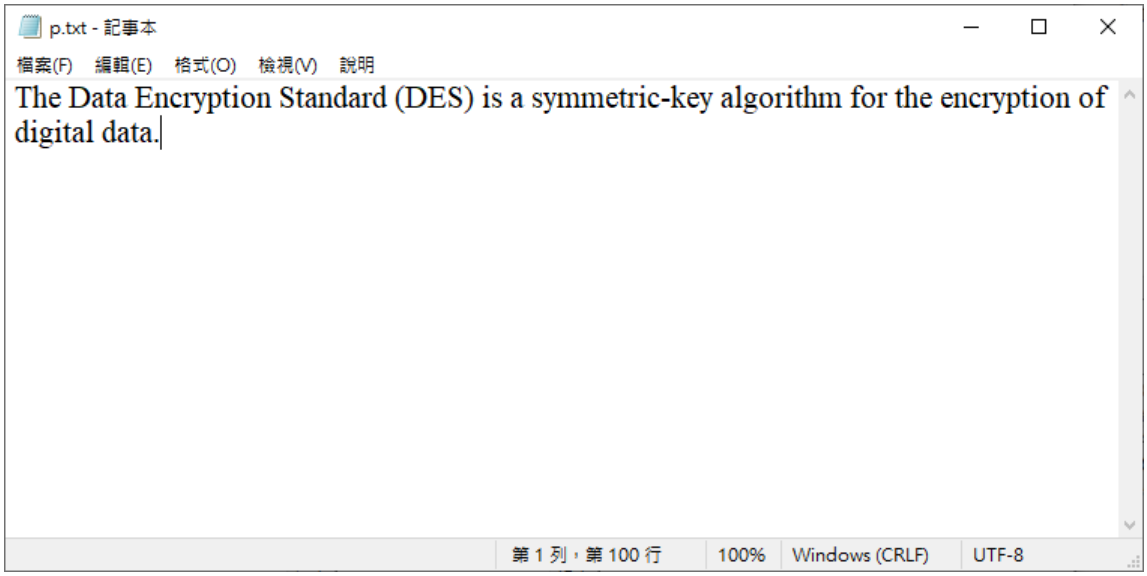


圖 11. 原始明文內容(plaintext)

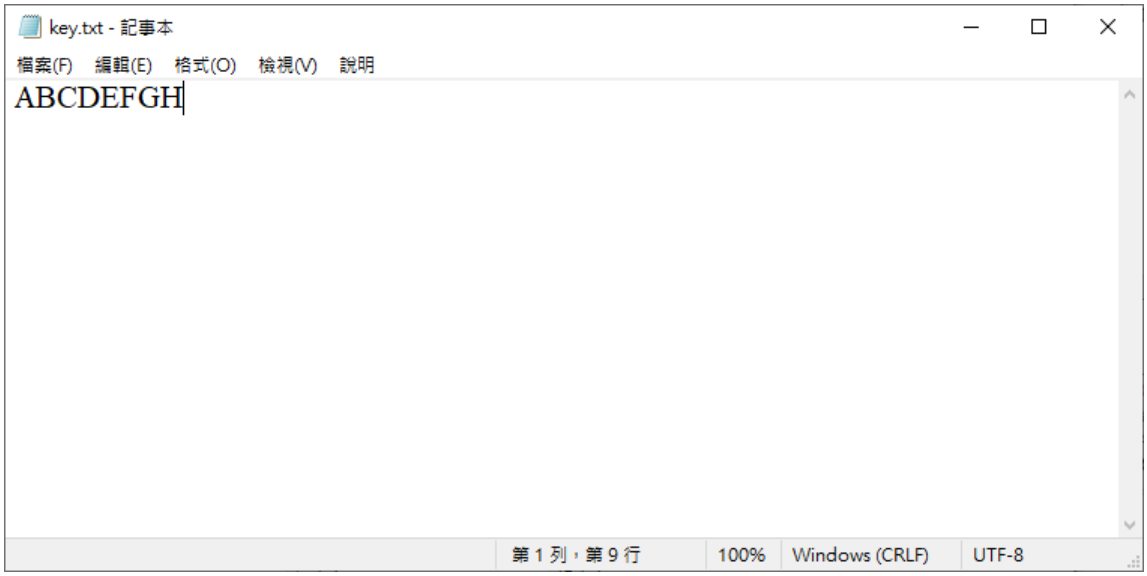


圖 12. 金鑰內容(key)

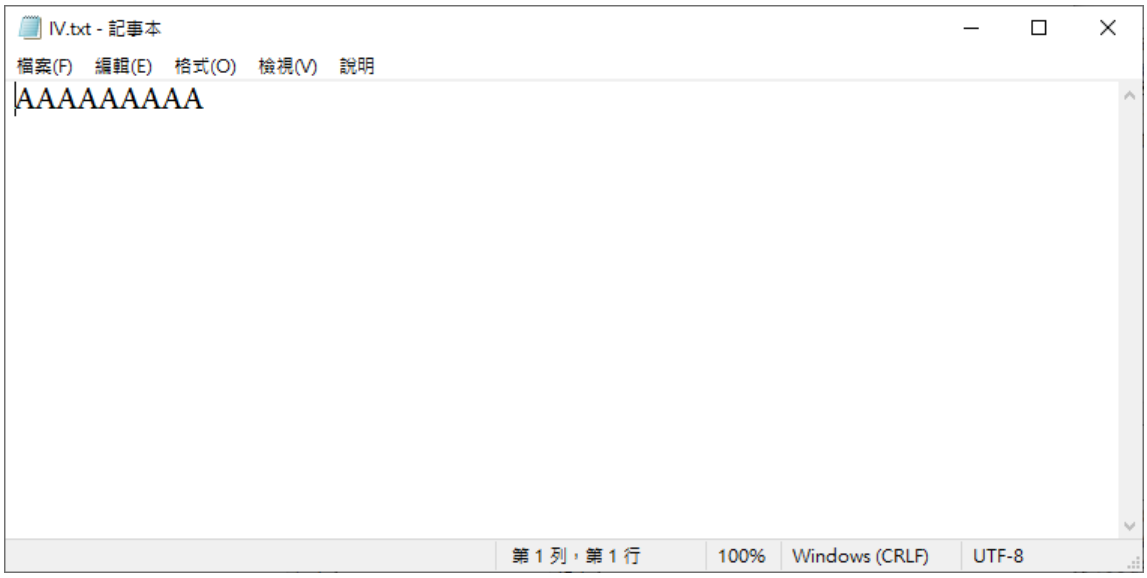


圖 13. 向量內容(IV)

A. ECB 模式

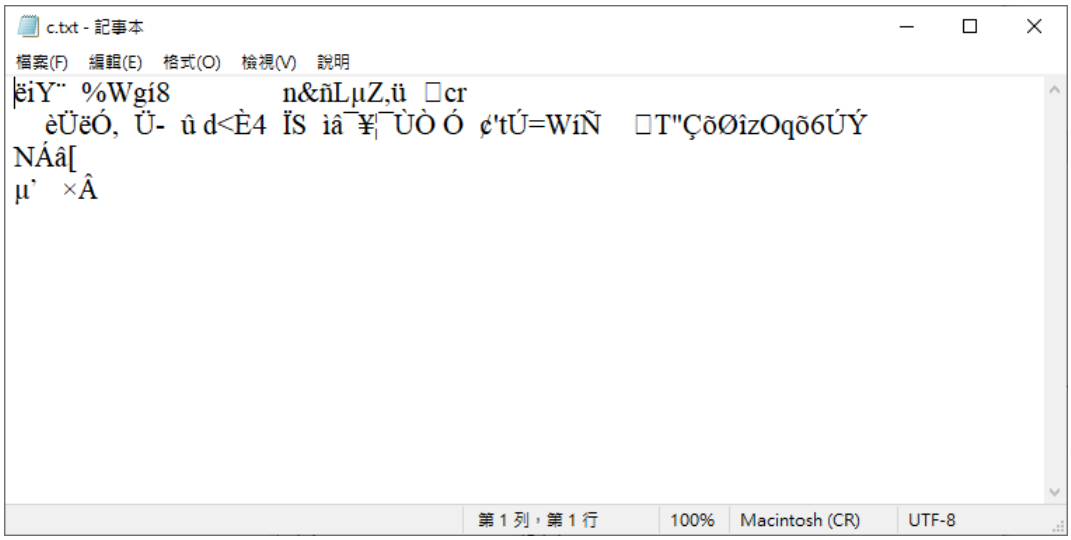


圖 12. 由 ASCII 編碼輸出的密文內容

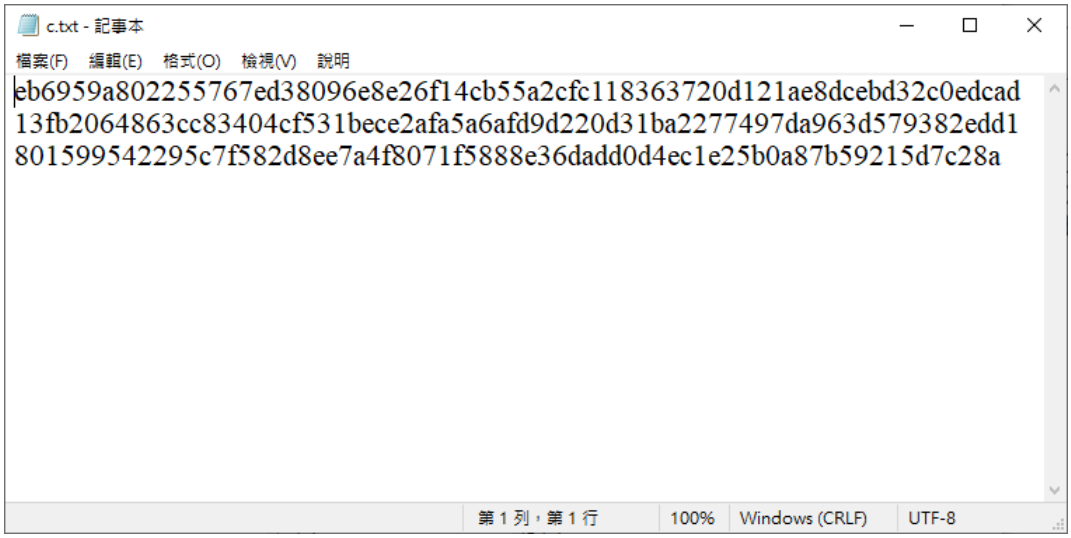


圖 13. 由 Hex 編碼輸出的密文內容

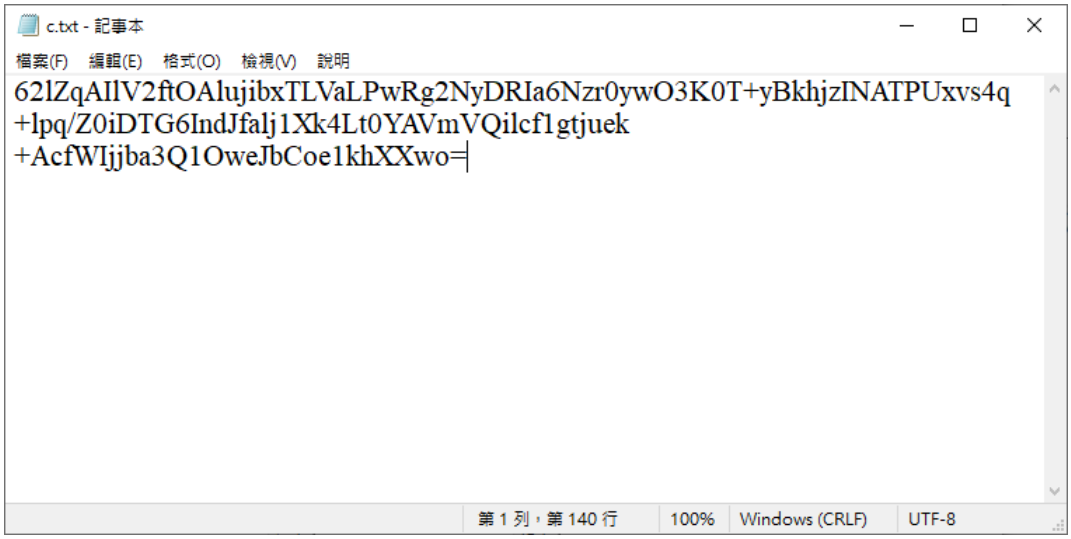


圖 14. 由 Base64 編碼輸出的密文內容

B. CBC 模式

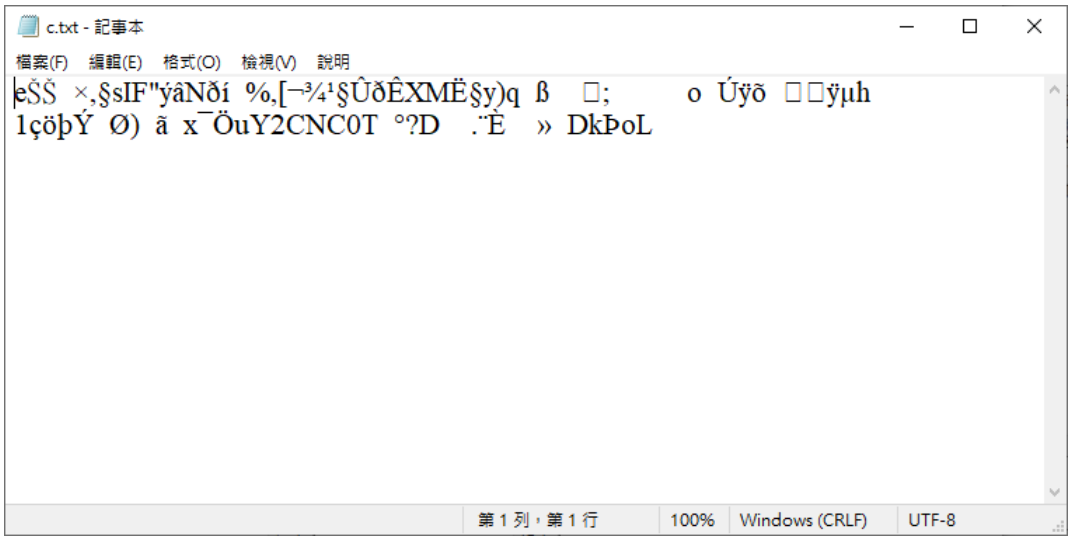


圖 12. 由 ASCII 編碼輸出的密文內容

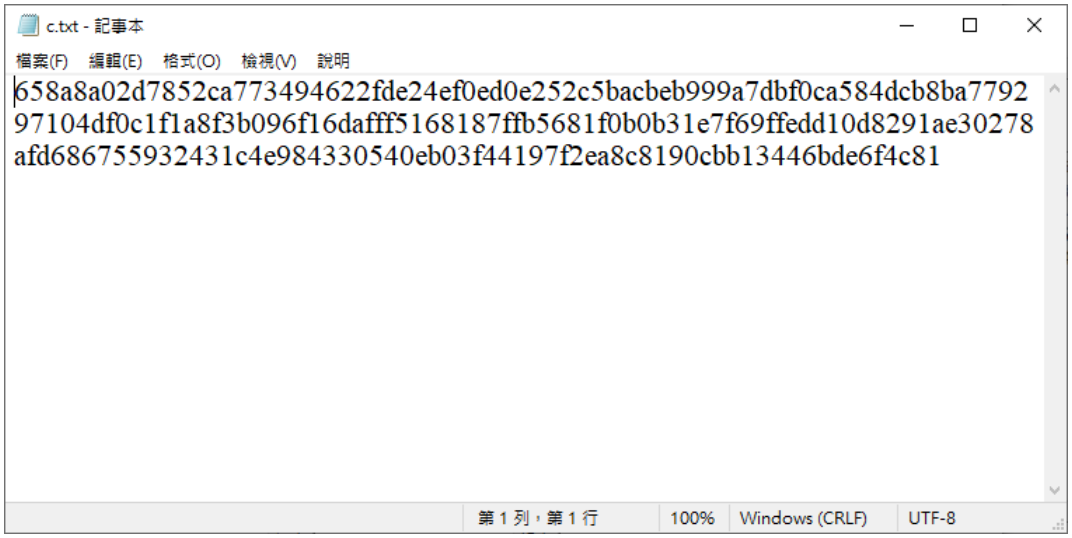


圖 13. 由 Hex 編碼輸出的密文內容

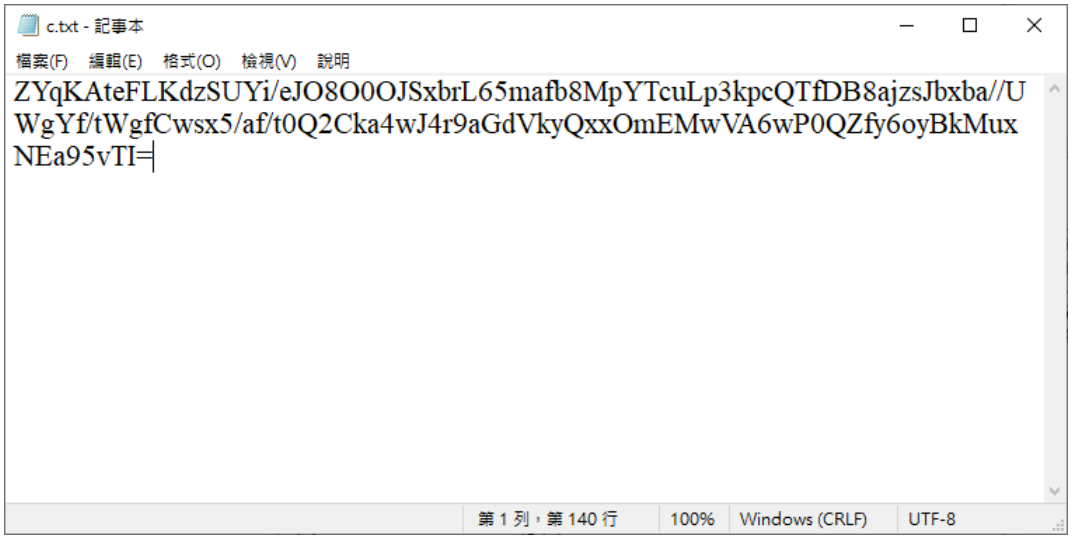


圖 14. 由 Base64 編碼輸出的密文內容

C. CFB 模式

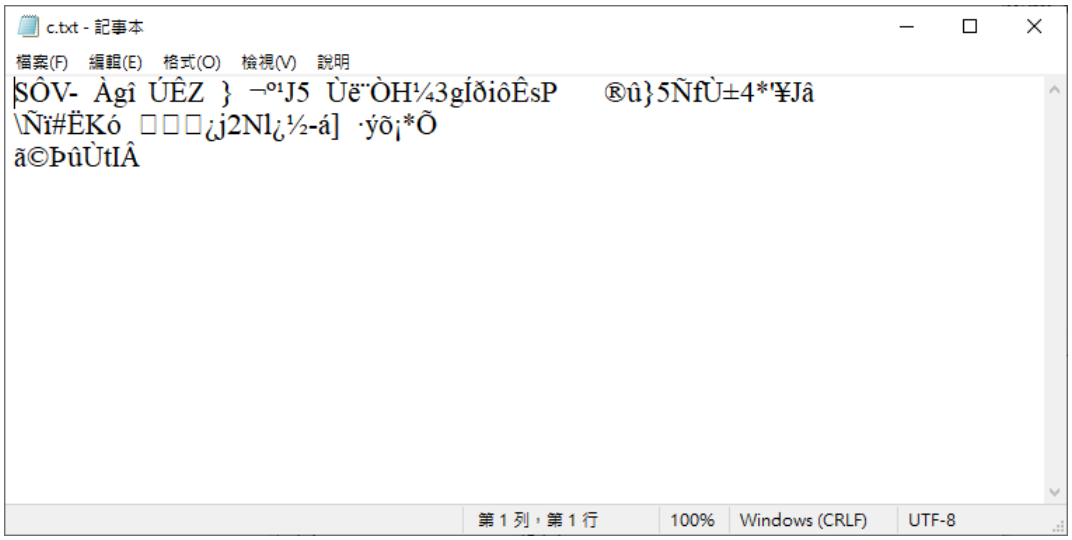


圖 12. 由 ASCII 編碼輸出的密文內容

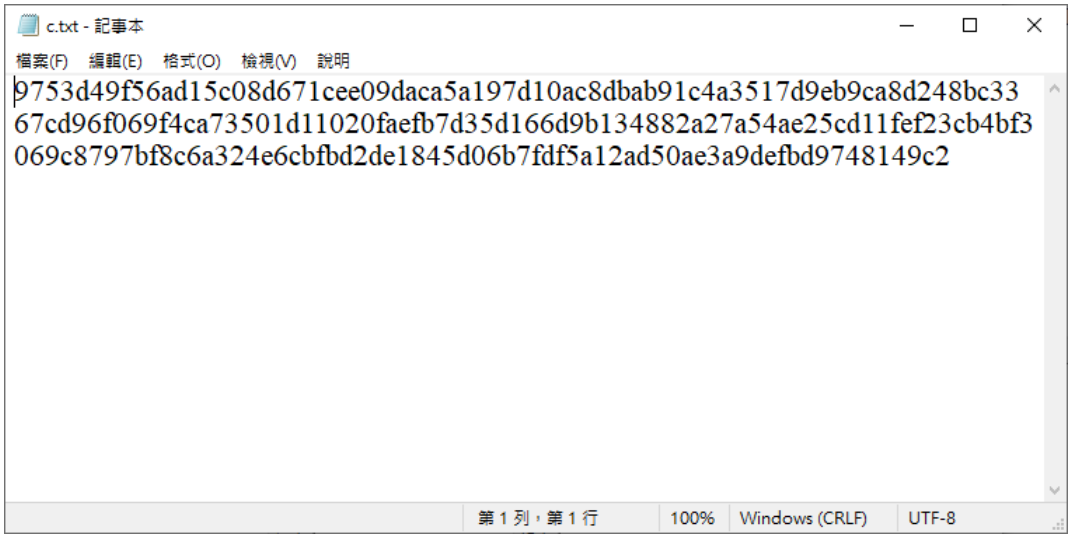


圖 13. 由 Hex 編碼輸出的密文內容

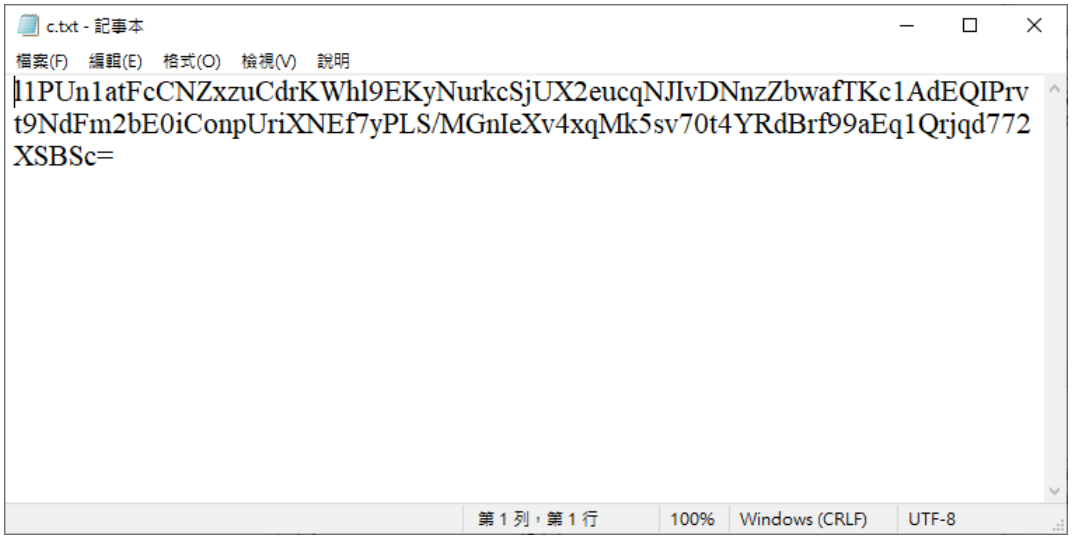


圖 14. 由 Base64 編碼輸出的密文內容

D. OFB 模式

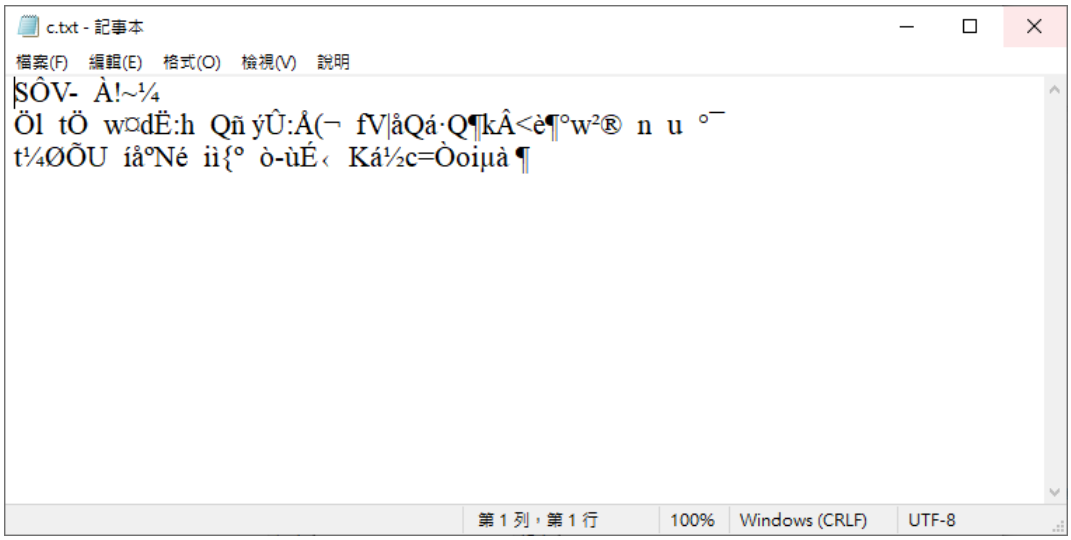


圖 12. 由 ASCII 編碼輸出的密文內容

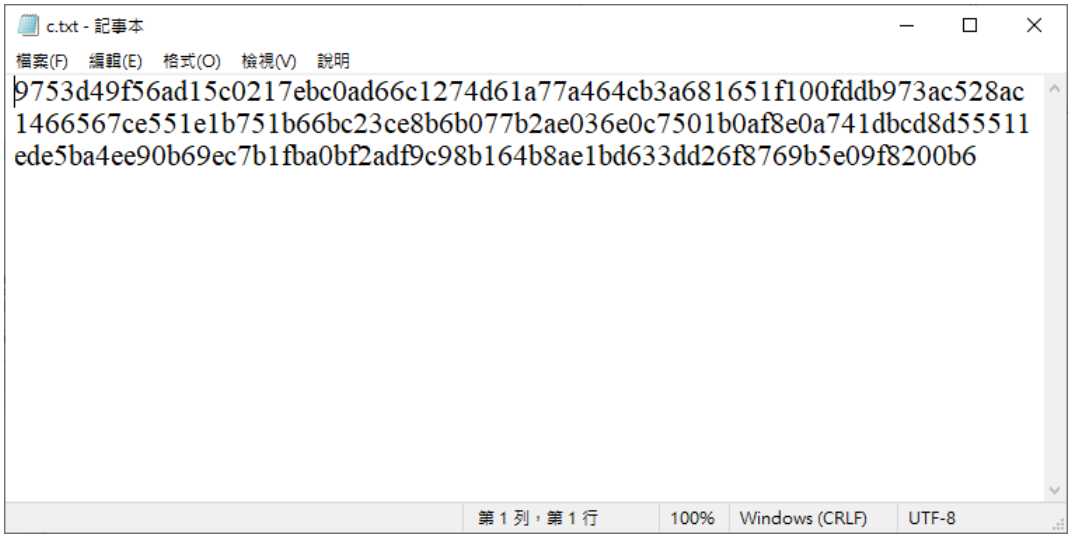


圖 13. 由 Hex 編碼輸出的密文內容

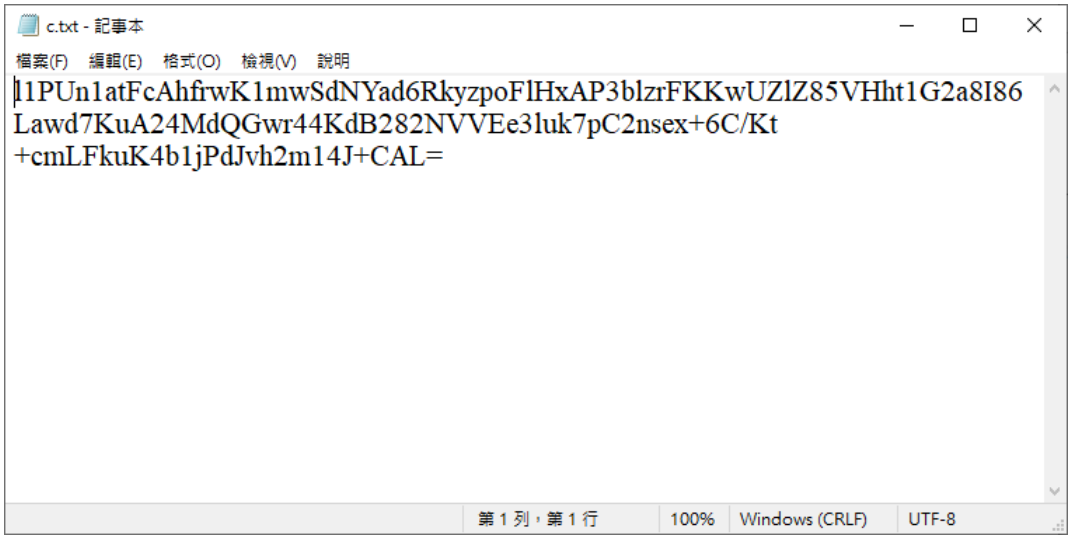


圖 14. 由 Base64 編碼輸出的密文內容

E. CTR 模式

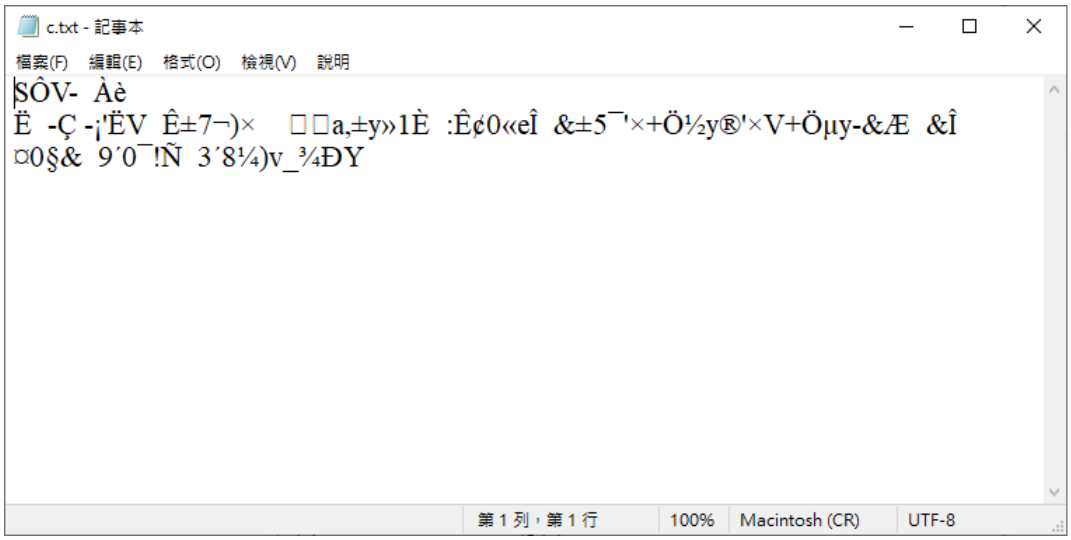


圖 12. 由 ASCII 編碼輸出的密文內容

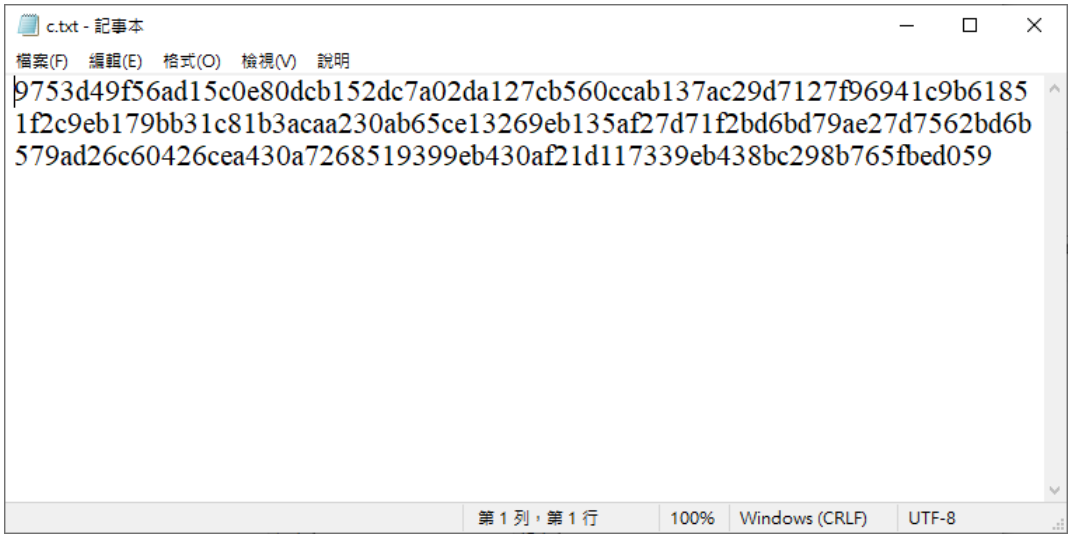


圖 13. 由 Hex 編碼輸出的密文內容

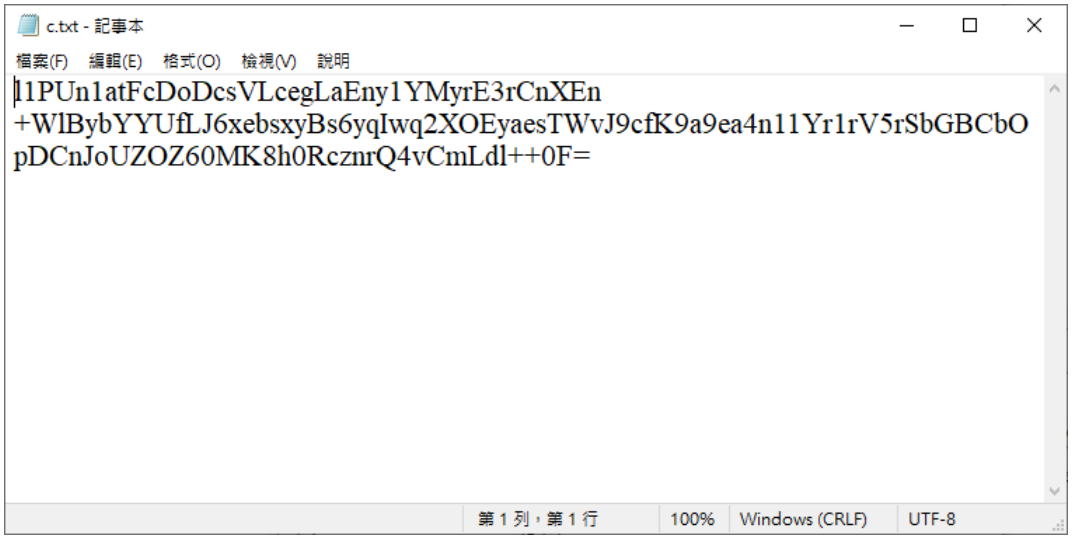


圖 14. 由 Base64 編碼輸出的密文內容

肆、心得：

在本次的期中專題中，遇到的問題主要有兩個部分。第一部分是對於 DES 算法以及 Python 的不熟悉，導致在撰寫程式時會有各種莫名的錯誤，例如在 Feistel 函式中會因為格式上的不同，導致不斷報錯；抑或是在編寫程式碼時，沒有注意到細節，導致在加密過程中，無法輸出想要的結果；又或者是在封裝副函式時不夠精簡，導致大量的程式碼重複利用，造成儲存空間及應用上的冗餘。

第兩部分則是 GUI 圖形介面的封裝。由於在之前撰寫 Python 時，都是以命令列為優秀控制，並沒有將其封裝成 GUI 介面的經歷。故在這次專題中，需要從零開始，經由課外書籍、網際網路補充設定 GUI 的相關知識，並且多次嘗試後，選擇適當的函式，並不斷的調試成最佳化。對我而言，這次封裝 GUI 是一次充實的經歷，使我對於基本的圖形介面操作有一定程度的了解。

總體來說，在本次專題中，需要改進的部分有加速程式碼的執行效率，撰寫程式碼的時間效率，以及優化 GUI 圖形介面使其更加好看。若之後還有相關的專題，希望自己能夠完成的更加良好。