3.1

給定兩組 cipher(c1, c1)、(c3, c4)

(E1, D1)解密條件是必須 D(k, c1) = D(k, c2)，因此可以藉由檢查回傳的訊息是否為 D(c1)，找到挑戰者是否選擇(c1, c1)，若選擇的是(c3, c4)則解密失敗，可以此攻破 CCA-secure，所以(E1, D1)不為 AE-secure。

(E2, D2)加密後，得到(c, c)，解密也需要兩個 c 相同才解的開，即無法使用上述做法攻破 CCA-secure，所以(E2,D2)為 AE-secure。

3.2

(1)

由 Carmichael's theorem

L (c^{λ} mod n^2) $*$ μ mod n = L(c^{λ} mod n^2) /L(g^{λ} mod n^2) mod n

$c^{\lambda} = (g^m r^n)^{\lambda} = g^{m\lambda} * r^{n\lambda} = g^{m\lambda}$

$((1+n)^x) \equiv 1 + nx \pmod{n^2}$

$g^{m\lambda} = ((1+n)^{\alpha\beta n})^{\lambda m} = (1+n)^{\alpha\lambda m\beta n\lambda m} \equiv (1+\alpha\lambda mn) \bmod n^2$

代回原式得:

$$\frac{L(c^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n = \frac{L(1+\alpha\lambda mn)}{L(1+\alpha\lambda m)} \bmod n = \frac{\alpha\lambda mn}{\alpha\lambda n} \bmod n = m$$

(2)

設 c = E_k(m, r)，對 c 乘上一個 0 的隨機加密

得:

$cr_1^n \bmod n^2 = E_k(m, rr_1 \bmod n)$

3.3

(1)

由 g 生成的 subgroup 的基數除以 Z*N = φ(N) = p*q 的基數

因為 r 是從 Z^*_{pq} 中選出的，所以 gcd(r, p*, q) = 1

h 與 g 的順序相同，所以 g = h，根據 descrite logarithm problem，因此，如果 r 被從 Z_{pq} 中選出，則 $g^m h^r$ 將會難以區分

(2)

$c = g^m h^r = g^m (g^k)^r = g^m g^{kr}$，已知 c, g, m，因此如果能在有效率的時間內找出 kr 則能破解此系統，但解 kr 為 descrite logarithm problem 並不存在有效率的解法，因此此問題為 computationally impossible

3.4

假設 E: $\{0,1\}^n \times \{0,1\}^l$

$h_i = E(m_i, H_i) \oplus H_i$, $h_j = E(m_j, H_j) \oplus H_j$

若要造成碰撞，只有在攻擊者試圖對 E 查詢 (m_i, H_i)，且獲得 E(m_i,

H_i) = $h_j \oplus H_i$ ，此事件發生的機率最高為 $1/(2^l - (i-1))$，根據生日悖論，在同一個 group，要找到不同的 m 具有相同的 hash 直，所需查詢的次數 q 會小於等於 $2^{l/2}$。

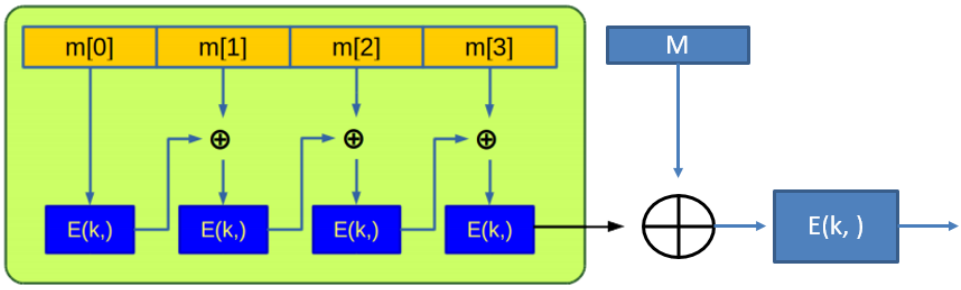
因為 i <= q <= $2^{l/2}$ ，所以 $h_i = h_j$ 的機率最高為 $2/(2^l - 2^{l/2})$，只要 l 夠大，該機率值可忽略，即可達成 collision resistant。

參考:

http://cs.wellesley.edu/~cs310/lectures/18.1_hash_revisited_slides_handouts.pdf(第三頁開始)

3.5

若無 last stage，可使用最後的 E(k,)，將想要的內容與之作 XOR，即可偽造簽名產生想要的訊息。



3.6

明文： All warfare is based on deception.

由於每次都只會得到倒數第二個 bit，因此將 c 乘上 2^e 將整個密文往左移後，再送入 oracle，即可獲得最後一個 bit，剩下的部分即可視為 LSB oracle 來進行。

LSB 解法:

假設密文為 c，明文為 m，oracle 最後一個 bit

每次送入 oracle 後即可取得最後一個 bit，因此在本題中，直接送入 c，即可取得倒數第二個 bit，接著將 c 乘上$(2^{-1})^e$ 將整個密文右移，此時 m 會變成 $2^{-1}m$

此時，我們要求的部分為 x1，前面的部分此處設為 y2，因此可以看成

$2^{-1}\ m = 2^{-1}x_0 + x_1 + 2y_2$

$r \equiv 2^{-1}x_0 + x_1 + 2y_2 \pmod 2$

由於前面的 $2y_2$ 並不會影響 x_1 的 bit，因此此處先不管

所以 $r \equiv 2^{-1}x_0 + x_1 \pmod 2$

$x_1 \equiv r - 2^{-1}x_0 \pmod 2$，依此類推即可由右往左依序取得所有明文 m 的

bit

程式碼:

```
import requests
import libnum
from Crypto.Util.number import *



N =
20330970518755266242385905909493999137973424517956750150773387949450532738688140473016222
24716963535789710862443698543805591403077337663589954137331842700192633609661964808898926
66344916558126256741851517291029922834976497563902217333771729967101036049938534525861868
81931950804787900352965945845764611738623330333991461000238591512750434491144339028770922
92230446203120992624650342535688784407833207991637407185100598329880030827523559313218059
14693636091553425020056741155485595996089729505271434927528288363505735195072563272004104
47961225666115908557731068901843936258478611053735330129457700258684949083496466237700
9
e =
17601719593773062356212374759207801318915519707405328064977013647467296877821574582351788
51869137569410221372453911375978073501005670173963578850510137788744016839082061467635493
24783662154975115658507034180653906388362323548765829068065774694214943095348097082200
11810705634550172763254560746975968588313259 3
url = "http://140.122.185.210:8080/oracle/"



def get_message():
    c =
74735327552606417518519580075808812638395960311889051049881911607008730481186830864609170
48233183580620071267702465699659368846043441508499408174074721199816517942387597065088661
09408827249419107167173461501484172151420007545281765851600717584006412112097188970137690
95205931176610213218689597247412920241755970999047775871710434929765272172589894268598275
00956973545739556786392893508222989463612768269690954526982275598416240996192338630459952
34591810378274120072474441261916710365782186699930359888954475372361778670415221378284989
28346191935811165654380882721353014240186958187924710330789430458314409240814800700
    inv_2 = inverse(2, N)  # 2^-1 to N
    neg_2e = pow(inv_2, e, N)  # 2^(-e) mod N

    # get last bit
    m = str(requests.get(url + str((c * pow(2, e, N)) % N)).text)[0]
```

```
    need_sub = 0
    for i in range(34*8-1):  # teacher said the string length = 34
        s = str(requests.get(url + str(c)).text)[0]  # get from oracle
        bit = (int(s) - need_sub) % 2

        # add the last bit into need_sub
        if i == 0:
            need_sub = inv_2 * int(m) % N

        # next round should take off
        need_sub = inv_2 * (need_sub + bit) % N

        # save the message bit
        m = str(bit) + m
        print(m)

        # right shift
        c = (c * neg_2e) % N

    with open('message.txt', 'w', encoding='UTF-8') as f:
        print(m, file=f)
    return m


def get_plaintext(m=None):
    if m is None:
        with open('message.txt', 'r', encoding='UTF-8') as f:
            for line in f:
                m = line.strip()
        print(m)
    print(libnum.b2s(m))


def main():
    get_plaintext(get_message())
    # get_plaintext()


main()
```

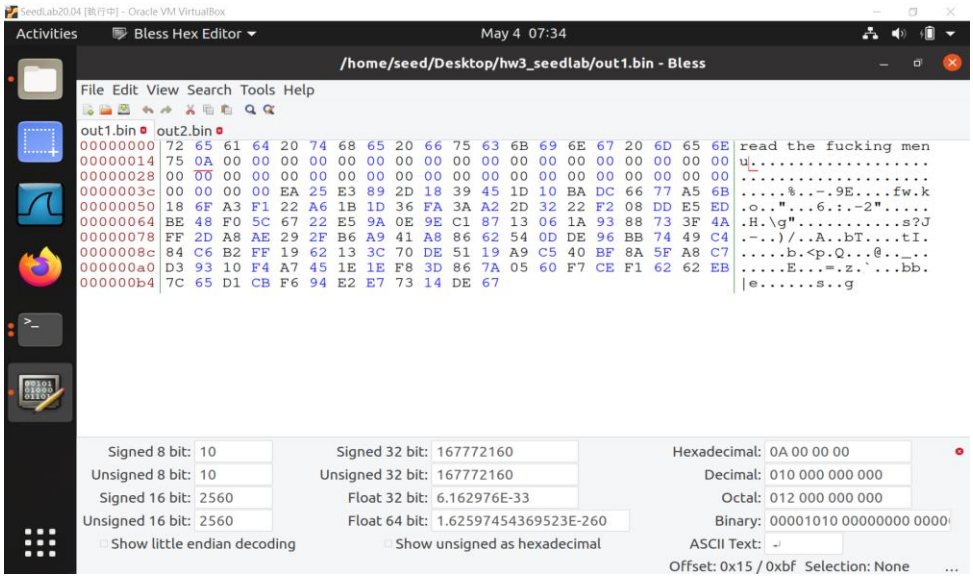
參考：

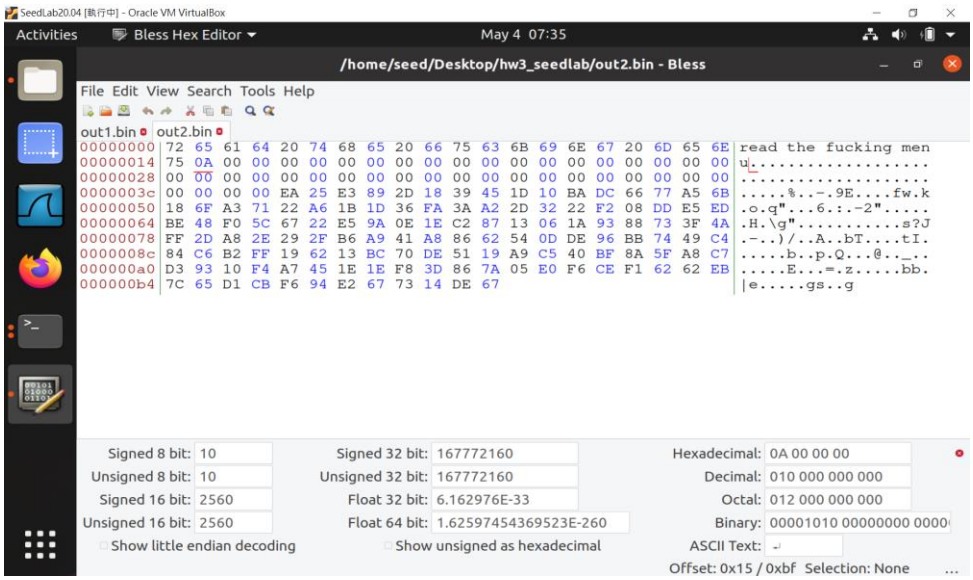
https://kruztw.tw/week02-crypto-rsa-lsb-oracle-attack/

3.7

Task1

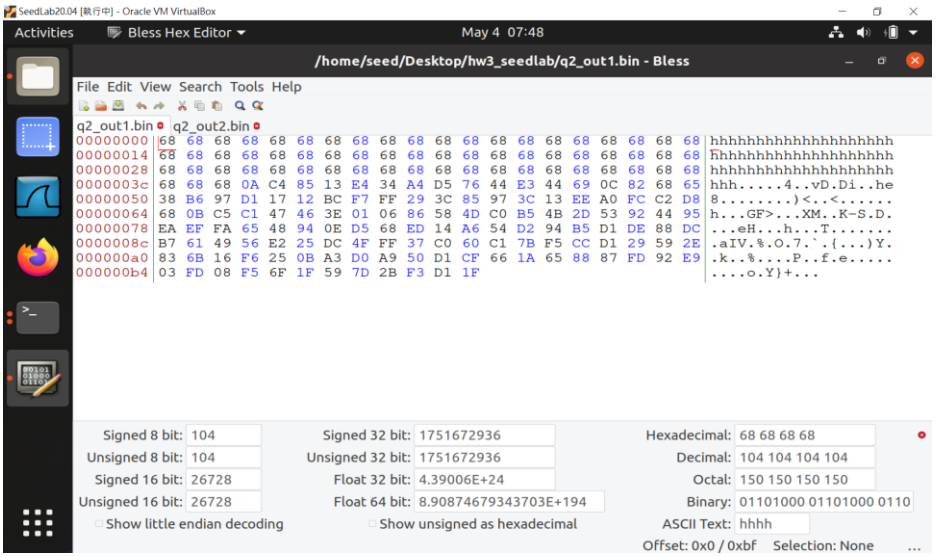
Q1

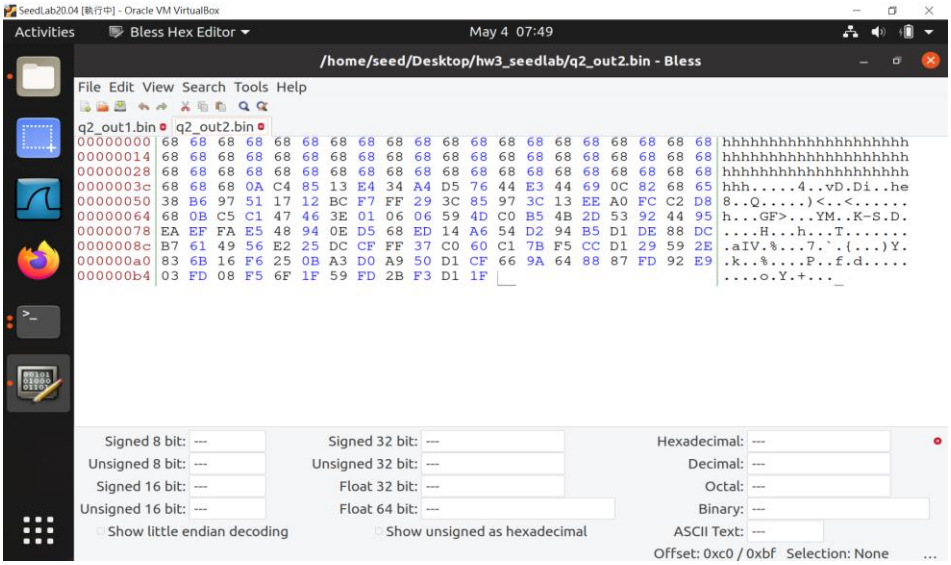




若不為 64 的倍數，會在 prefix 後，補 00 直到滿足 64 的倍數

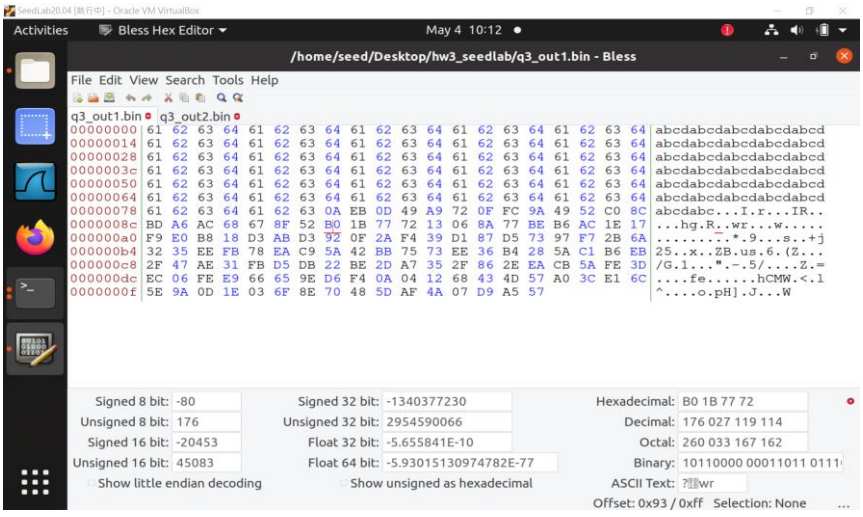
Q2

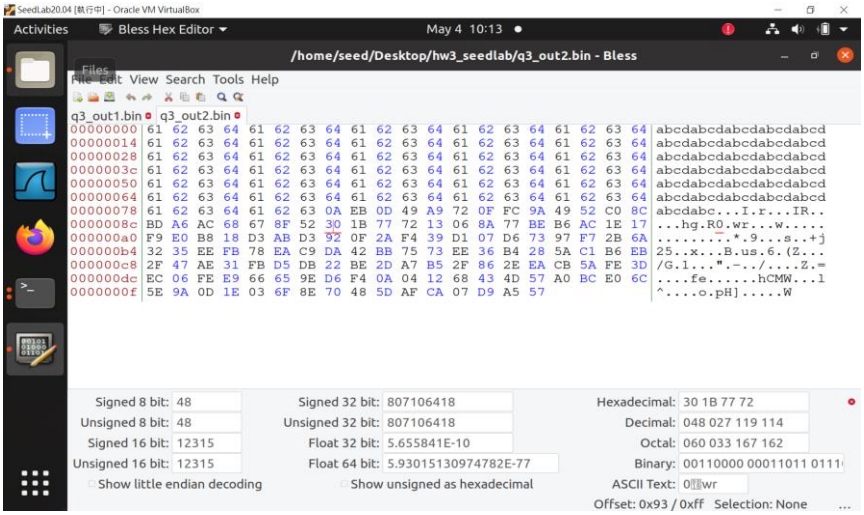




若剛好為 64bytes，則不會有 padding

Q3






```
[05/04/21]seed@VM:~/.../hw3_seedlab$ diff q3_out1.bin q3_out2.bin
Binary files q3_out1.bin and q3_out2.bin differ
[05/04/21]seed@VM:~/.../hw3_seedlab$ md5sum q3_out1.bin
0072fcfcb5630c4205d74e1ca37d91f9  q3_out1.bin
[05/04/21]seed@VM:~/.../hw3_seedlab$ md5sum q3_out2.bin
0072fcfcb5630c4205d74e1ca37d91f9  q3_out2.bin
```

不同的 byte 都發生在 p 和 q 的部分

下圖為使用 python 比較每個 byte 的結果:

```
out1:
                    B0
                                  87 D5
                    5A
                              35
                                           3C E1
                              4A
------------------------------------------------------------
out2:
                    30
                                  07 D6
                    DA
                              B5
                                           BC E0
                              CA
```

下圖使用 xxd 指令將 bin 寫入 txt 檔，再用 diff 比較的結果

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ xxd q3_out1.bin > q3_o1.txt
[05/05/21]seed@VM:~/.../hw3_seedlab$ xxd q3_out2.bin > q3_o2.txt
[05/05/21]seed@VM:~/.../hw3_seedlab$ diff q3_o1.txt q3_o2.txt
10,12c10,12
< 00000090: 678f 52b0 1b77 7213 068a 77be b6ac 1e17  g.R..wr...w.....
< 000000a0: f9e0 b818 d3ab d392 0f2a f439 d187 d573  .........*.9...s
< 000000b0: 97f7 2b6a 3235 eefb 78ea c95a 42bb 7573  ..+j25..x..ZB.us
---
> 00000090: 678f 5230 1b77 7213 068a 77be b6ac 1e17  g.R0.wr...w.....
> 000000a0: f9e0 b818 d3ab d392 0f2a f439 d107 d673  .........*.9...s
> 000000b0: 97f7 2b6a 3235 eefb 78ea c9da 42bb 7573  ..+j25..x...B.us
14,16c14,16
< 000000d0: be2d a735 2f86 2eea cb5a fe3d ec06 fee9  .-.5/....Z.=....
< 000000e0: 6665 9ed6 f40a 0412 6843 4d57 a03c e16c  fe......hCMW.<.l
< 000000f0: 5e9a 0d1e 036f 8e70 485d af4a 07d9 a557  ^....o.pH].J...W
---
> 000000d0: be2d a7b5 2f86 2eea cb5a fe3d ec06 fee9  .-../....Z.=....
> 000000e0: 6665 9ed6 f40a 0412 6843 4d57 a0bc e06c  fe......hCMW...l
> 000000f0: 5e9a 0d1e 036f 8e70 485d afca 07d9 a557  ^....o.pH].....W
[05/05/21]seed@VM:~/.../hw3_seedlab$
```

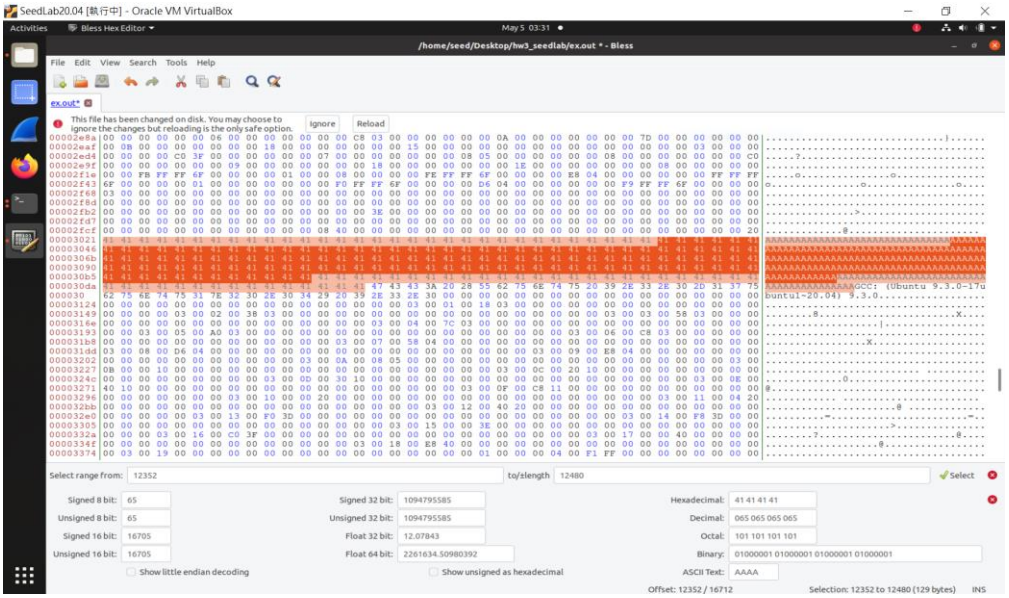
Task2

此處使用上一個任務取得的 q3_o1.txt 與 q3_o2.txt 將兩個內容相連成一個檔案

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ cat q3_o1.txt q3_o2.txt > task2.txt
[05/05/21]seed@VM:~/.../hw3_seedlab$ cat task2.txt
00000000: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000010: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000020: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000030: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000040: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000050: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000060: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000070: 6162 6364 6162 6364 6162 6364 6162 630a  abcdabcdabcdabc.
00000080: eb0d 49a9 720f fc9a 4952 c08c bda6 ac68  ..I.r...IR.....h
00000090: 678f 52b0 1b77 7213 068a 77be b6ac 1e17  g.R..wr...w.....
000000a0: f9e0 b818 d3ab d392 0f2a f439 d187 d573  .........*.9...s
000000b0: 97f7 2b6a 3235 eefb 78ea c95a 42bb 7573  ..+j25..x..ZB.us
000000c0: ee36 b428 5ac1 b6eb 2f47 ae31 fbd5 db22  .6.(Z.../G.1..."
000000d0: be2d a735 2f86 2eea cb5a fe3d ec06 fee9  .-.5/....Z.=....
000000e0: 6665 9ed6 f40a 0412 6843 4d57 a03c e16c  fe......hCMW.<.l
000000f0: 5e9a 0d1e 036f 8e70 485d af4a 07d9 a557  ^....o.pH].J...W
00000000: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000010: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000020: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000030: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000040: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000050: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000060: 6162 6364 6162 6364 6162 6364 6162 6364  abcdabcdabcdabcd
00000070: 6162 6364 6162 6364 6162 6364 6162 630a  abcdabcdabcdabc.
00000080: eb0d 49a9 720f fc9a 4952 c08c bda6 ac68  ..I.r...IR.....h
00000090: 678f 5230 1b77 7213 068a 77be b6ac 1e17  g.R0.wr...w.....
000000a0: f9e0 b818 d3ab d392 0f2a f439 d107 d673  .........*.9...s
000000b0: 97f7 2b6a 3235 eefb 78ea c9da 42bb 7573  ..+j25..x...B.us
000000c0: ee36 b428 5ac1 b6eb 2f47 ae31 fbd5 db22  .6.(Z.../G.1..."
000000d0: be2d a7b5 2f86 2eea cb5a fe3d ec06 fee9  .-../....Z.=....
000000e0: 6665 9ed6 f40a 0412 6843 4d57 a0bc e06c  fe......hCMW...l
000000f0: 5e9a 0d1e 036f 8e70 485d afca 07d9 a557  ^....o.pH].....W
```

Task3

此處 char 使用 0x41 產生 out 檔，用 bless 查看的結果如下圖



從開頭的 A 開始計算，此處取到 12352byte(64 倍數)做 prefix

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ head -c 12352 ex.out > prefix
```

間隔 128byte 後，取 suffix

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ tail -c +12480 ex.out > suffix
```

透過 md5collgen 生成 p 和 q，並各取 128byte

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ md5collgen -p prefix -o t3_o1.bin t3_o2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 't3_o1.bin' and 't3_o2.bin'
Using prefixfile: 'prefix'
Using initial value: 65322d4c1c0b5802cdf6d6b729393313

Generating first block: ..............
Generating second block: S11.....................................................
Running time: 17.214 s

[05/05/21]seed@VM:~/.../hw3_seedlab$ tail -c 128 t3_o1.bin > p
[05/05/21]seed@VM:~/.../hw3_seedlab$ tail -c 128 t3_o2.bin > q
```

最終結果:

```
[05/05/21]seed@VM:~/.../hw3_seedlab$ cat prefix p suffix > t3_g1
[05/05/21]seed@VM:~/.../hw3_seedlab$ cat prefix q suffix > t3_g2
[05/05/21]seed@VM:~/.../hw3_seedlab$ MD5sum t3_g1
MD5sum: command not found
[05/05/21]seed@VM:~/.../hw3_seedlab$ md5sum t3_g1
0593fa0257785ae6ddc33679dff98580  t3_g1
[05/05/21]seed@VM:~/.../hw3_seedlab$ md5sum t3_g2
0593fa0257785ae6ddc33679dff98580  t3_g2
[05/05/21]seed@VM:~/.../hw3_seedlab$ diff t3_g1 t3_g2
Binary files t3_g1 and t3_g2 differ
[05/05/21]seed@VM:~/.../hw3_seedlab$
```

Task4

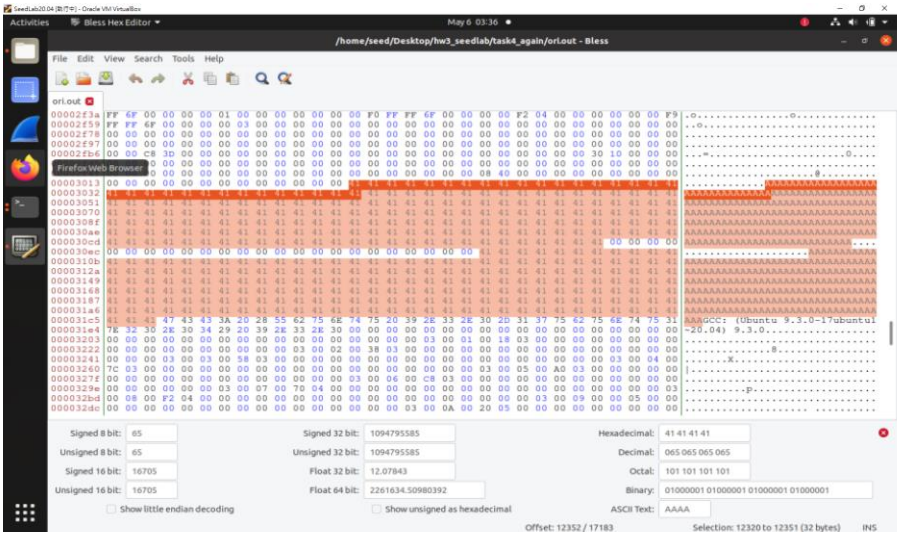
C code:

```c
#include<stdio.h>
char x[200] = {
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0>
};

char y[200] = {
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0>
};

int main()
{
  int same = 1;
  for(int i = 0; i<200; ++i)
  {
    if (x[i]!=y[i])
    {
      same = 0;
      break;
    }
  }
  if (same)
    printf("run benign code\n");
  else
    printf("run malicious code\n");
}
```

產生執行檔，並測試結果，以確認是相同內容的兩個陣列 x, y

```
[05/06/21]seed@VM:~/.../task4_again$ sudo nano task4.c
[05/06/21]seed@VM:~/.../task4_again$ gcc task4.c -o ori.out
[05/06/21]seed@VM:~/.../task4_again$ ./ori.out
run benign code
```

透過 bless 查看 ori.out 的內容



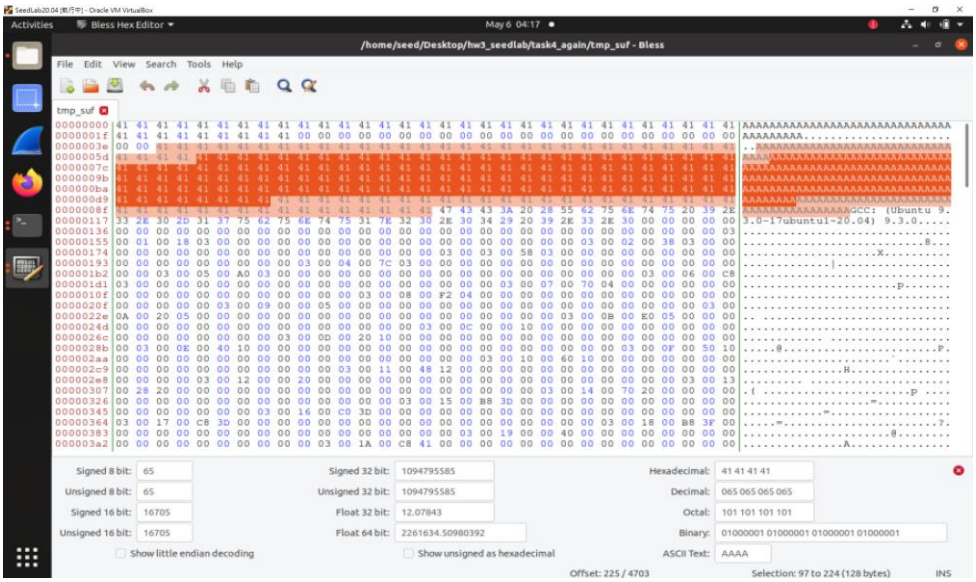
此處取 12352(64 的倍數)作為 prefix，中間間隔 128byte 後(要用來存放 p 或 q)，把剩下部份另外保存到 tmp_suf(此處的 tail 由於是從前面數，所以要算+12481 在這邊算錯卡超久 QQ)

```
[05/06/21]seed@VM:~/.../task4_again$ head -c 12352 ori.out > pre
[05/06/21]seed@VM:~/.../task4_again$ tail -c +12481 ori.out > tmp_suf
[05/06/21]seed@VM:~/.../task4_again$ md5collgen -p pre -o p_o1 p_o2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'p_o1' and 'p_o2'
Using prefixfile: 'pre'
Using initial value: a45c38b3f51303c244132121094ad70e

Generating first block: ..............
Generating second block: W.......
Running time: 9.9909 s
```

透過 md5collgen 從 prefix 產生不同的 p 和 q，並取後面的 128byte，作為之後要用的 p 和 q

```
[05/06/21]seed@VM:~/.../task4_again$ tail -c 128 p_o1 > p
[05/06/21]seed@VM:~/.../task4_again$ tail -c 128 p_o2 > q
```

由於要讓 x 與 y 兩個陣列內容相同，要幫 y 陣列也加上 p，此處觀察 tmp_suf 的 byte，找出對應的位置切出 mid 與 suf




```
[05/06/21]seed@VM:~/.../task4_again$ bless tmp_suf
Gtk-Message: 03:23:33.838: Failed to load module "canberra-gtk-module"
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find file "/home/seed/.config/bless/export_patterns"
[05/06/21]seed@VM:~/.../task4_again$ head -c 96 tmp_suf > mid
[05/06/21]seed@VM:~/.../task4_again$ tail -c +225 tmp_suf > suf
```

組合最終結果，並將產生的檔案 g1 和 g2 轉換成執行檔，確認結果是否正確

```
[05/06/21]seed@VM:~/.../task4_again$ cat pre p mid p suf > g1
[05/06/21]seed@VM:~/.../task4_again$ cat pre q mid p suf > g2
[05/06/21]seed@VM:~/.../task4_again$ chmod +x g1
[05/06/21]seed@VM:~/.../task4_again$ chmod +x g2
[05/06/21]seed@VM:~/.../task4_again$ ./g1
run benign code
[05/06/21]seed@VM:~/.../task4_again$ ./g2
run malicious code
```

```
[05/07/21]seed@VM:~/.../task4_again$ diff g1 g2
Binary files g1 and g2 differ
[05/07/21]seed@VM:~/.../task4_again$ xxd g1 > g1_txt
[05/07/21]seed@VM:~/.../task4_again$ xxd g2 > g2_txt
[05/07/21]seed@VM:~/.../task4_again$ diff g1_txt g2_txt
774,776c774,776
< 00003050: b545 6b66 8b4d 5922 3402 3a5d 0d30 230e  .Ekf.MY"4.:].0#.
< 00003060: eed9 feca 63e4 93b7 6b62 50fc 1e72 516a  ....c...kbP..rQj
< 00003070: c453 4a7f 734f 0da1 dbd0 0354 8d64 4f59  .SJ.sO.....T.dOY
---
> 00003050: b545 6be6 8b4d 5922 3402 3a5d 0d30 230e  .Ek..MY"4.:].0#.
> 00003060: eed9 feca 63e4 93b7 6b62 50fc 1ef2 516a  ....c...kbP...Qj
> 00003070: c453 4a7f 734f 0da1 dbd0 03d4 8d64 4f59  .SJ.sO.......dOY
778,780c778,780
< 00003090: b40f 6e47 3205 064e ee68 e675 cb69 4d8f  ..nG2..N.h.u.iM.
< 000030a0: 758f f3d2 8fad 9065 f6a2 517a 8cdc 3e9f  u......e..Qz..>.
< 000030b0: e76c 1551 71bc d3ca a756 e1f0 2297 1907  .l.Qq....V.."...
---
> 00003090: b40f 6ec7 3205 064e ee68 e675 cb69 4d8f  ..n.2..N.h.u.iM.
> 000030a0: 758f f3d2 8fad 9065 f6a2 517a 8c5c 3e9f  u......e..Qz.\>.
> 000030b0: e76c 1551 71bc d3ca a756 e170 2297 1907  .l.Qq....V.p"...
```