資訊安全 HW1

40647027S 陳冠潁

1. Prove that: H(X,Y) = H(X) + H(Y|X)

Proof:

$$\mathrm{H(X)} + \mathrm{H(Y|X)} = \mathrm{H(X)} + \sum_{x\in X} p(x)H(Y|X=x)$$

$$= H(X) - \sum_{x\in X} p(x) \sum_{y\in Y} p(y|x)log_2 p(y|x) = \mathrm{H(X)}$$

$$- \sum_{x\in X, y\in Y} p(x,y)log_2 p(y|x)$$

$$= H(X)$$

$$- \sum_{x\in X, y\in Y} p(x,y)log_2 \frac{p(x,y)}{p(x)}$$

$$= H(X)$$

$$- \left(\sum_{x\in X, y\in Y} p(x,y)log_2 p(x,y) - \sum_{x\in X, y\in Y} p(x,y)log_2 p(x) \right)$$

$$= H(X) + H(X,Y) - H(X) = H(X,Y)$$

得證。

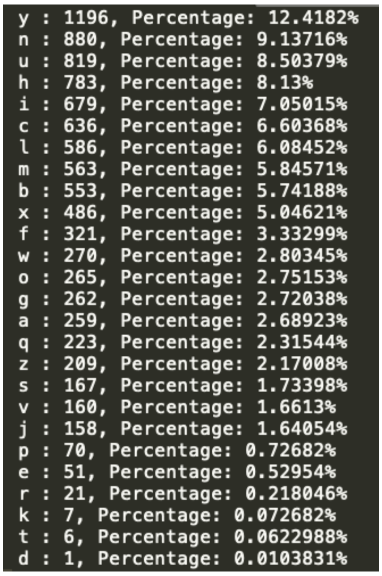
Proof:

$$H(X_1, X_2, \dots, X_n) = \sum_{i=1}^{n} H(X_i |\, X_{i-1}, \dots, X_1) \le \sum_{i=1}^{n} H(X_i)$$

If and only if $X_1, X_2, \dots, X_n$ are independent 時，左式等於右式。

2. (1), (3), (4)

3. 第一步：我先讀取檔案並且把出現平率由大到小排好，如圖。

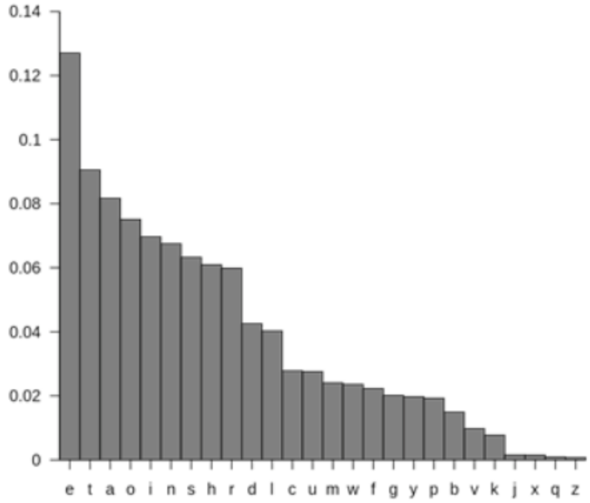


```
y : 1196, Percentage: 12.4182%
n : 880, Percentage: 9.13716%
u : 819, Percentage: 8.50379%
h : 783, Percentage: 8.13%
i : 679, Percentage: 7.05015%
c : 636, Percentage: 6.60368%
l : 586, Percentage: 6.08452%
m : 563, Percentage: 5.84571%
b : 553, Percentage: 5.74188%
x : 486, Percentage: 5.04621%
f : 321, Percentage: 3.33299%
w : 270, Percentage: 2.80345%
o : 265, Percentage: 2.75153%
g : 262, Percentage: 2.72038%
a : 259, Percentage: 2.68923%
q : 223, Percentage: 2.31544%
z : 209, Percentage: 2.17008%
s : 167, Percentage: 1.73398%
v : 160, Percentage: 1.6613%
j : 158, Percentage: 1.64054%
p : 70, Percentage: 0.72682%
e : 51, Percentage: 0.52954%
r : 21, Percentage: 0.218046%
k : 7, Percentage: 0.072682%
t : 6, Percentage: 0.0622988%
d : 1, Percentage: 0.0103831%
```

第二步：到網路查字母出現頻率。

參考： https://inventwithpython.com/hacking/chapter20.html

Figure 20-2. Letter frequency of normal English, sorted.



將查到的字母出現頻率映射到我計算密文的字母頻率，使用 std::map 實作。

第三步：除錯過程

I. 估計 nby 是 the, 但轉出來是 tre，把 b 對到 h，原本是 m 對到 h，因此 m 姑且先對到 r。

II. machnoe 估計 machine(guwbchy)，懷疑 c 對到 i，h 對到 n。

III. cseated 懷疑是 created(wlyunyx) ，l 對到 r。

IV. dircovesed 懷疑是 discovered(xcmwipylyx) ，m 對到 s, l 對到 r。

V. aii 懷疑是 all(uff) ，f 對到 l。

VI. turinw 懷疑是 turing(nolcha) ，a 對到 g，原本是 z 對到 g，姑且 a 跟 z 先交換。

VII.attriputes 懷疑是 attributes(unnlcvonym) ，v 對到 b。
VIII.　ejisting 懷疑是 existing(yrcmncha)，r 對到 x。
IX.　sowtfare 懷疑是 software（miznquly），z 對到 f。

第四部：感覺解的差不多了，貼到 Google 搜尋一下，確認全對，完成。

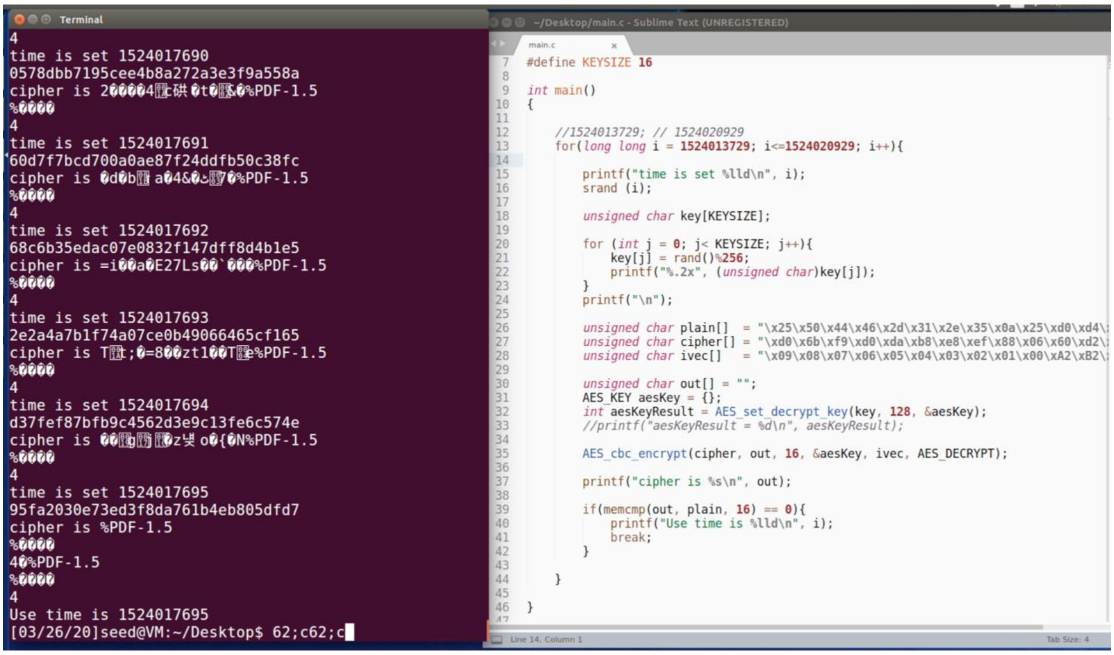
備註：Frequency-Analysis-Attack 程式執行方式請看資料夾內 README.md，解出來的明文在”Problem3 plain text.txt”。

4. 目標明文大概為: No matter how perfect the crime, as long as people do there is no solution don't open.

5.

Task1. 原本每次執行產出的 key 都不相同，但是註解調 srand(time(NULL))之後，每次產生的 key 都一樣，rand 每次 call 都是針對上一個值來產生，因此用 srand()並帶入時間作為參數改變一開始的值。

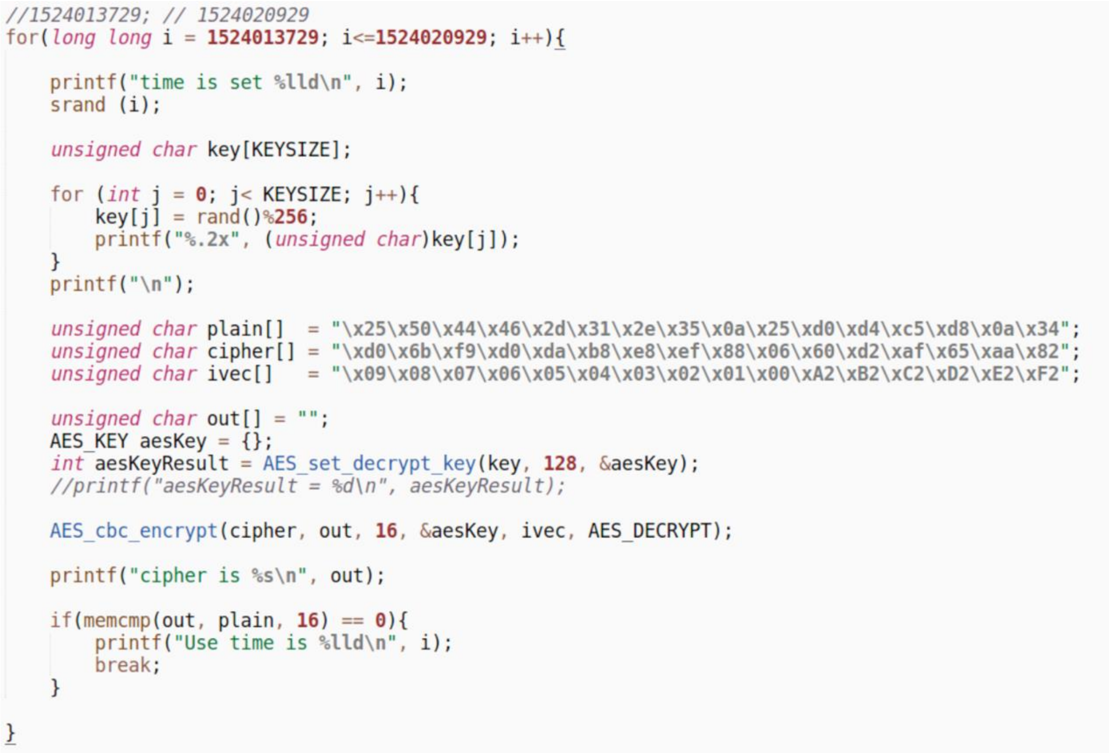
Task2. 說明給了一個時間區段有可能為 srand 帶入的 time 參數，因此暴力試試看那個時間段的所有可能，當 cipher 透過 AES 解密與提供的 plain 相同，則代表找到了該文件在哪個時間點做加密。



```
4
time is set 1524017690
0578dbb7195cee4b8a272a3e3f9a558a
cipher is 2�����4�t硄�t��&�%PDF-1.5
%����
4
time is set 1524017691
60d7f7bcd700a0ae87f24ddfb50c38fc
cipher is �d�b� a�4&�ح�7�%PDF-1.5
%����
4
time is set 1524017692
68c6b35edac07e0832f147dff8d4b1e5
cipher is =i��a�E27Ls��`���%PDF-1.5
%����
4
time is set 1524017693
2e2a4a7b1f74a07ce0b49066465cf165
cipher is T�t;�=8��zt1��T�e%PDF-1.5
%����
4
time is set 1524017694
d37fef87bfb9c4562d3e9c13fe6c574e
cipher is ���g�j��z볶 o�{�N%PDF-1.5
%����
4
time is set 1524017695
95fa2030e73ed3f8da761b4eb805dfd7
cipher is %PDF-1.5
%����
4�%PDF-1.5
%����
4
Use time is 1524017695
[03/26/20]seed@VM:~/Desktop$ 62;c62;c
```

```c
#define KEYSIZE 16

int main()
{

    //1524013729; // 1524020929
    for(long long i = 1524013729; i<=1524020929; i++){

        printf("time is set %lld\n", i);
        srand (i);

        unsigned char key[KEYSIZE];

        for (int j = 0; j< KEYSIZE; j++){
            key[j] = rand()%256;
            printf("%.2x", (unsigned char)key[j]);
        }
        printf("\n");

        unsigned char plain[]  = "\x25\x50\x44\x46\x2d\x31\x2e\x35\x0a\x25\xd0\xd4\
        unsigned char cipher[] = "\xd0\x6b\xf9\xd0\xda\xb8\xe8\xef\x88\x06\x60\xd2\
        unsigned char ivec[]   = "\x09\x08\x07\x06\x05\x04\x03\x02\x01\x00\xA2\xB2\

        unsigned char out[] = "";
        AES_KEY aesKey = {};
        int aesKeyResult = AES_set_decrypt_key(key, 128, &aesKey);
        //printf("aesKeyResult = %d\n", aesKeyResult);

        AES_cbc_encrypt(cipher, out, 16, &aesKey, ivec, AES_DECRYPT);

        printf("cipher is %s\n", out);

        if(memcmp(out, plain, 16) == 0){
            printf("Use time is %lld\n", i);
            break;
        }

    }

}
```

主要 Code:



```
//1524013729; // 1524020929
for(long long i = 1524013729; i<=1524020929; i++){

    printf("time is set %lld\n", i);
    srand (i);

    unsigned char key[KEYSIZE];

    for (int j = 0; j< KEYSIZE; j++){
        key[j] = rand()%256;
        printf("%.2x", (unsigned char)key[j]);
    }
    printf("\n");

    unsigned char plain[]  = "\x25\x50\x44\x46\x2d\x31\x2e\x35\x0a\x25\xd0\xd4\xc5\xd8\x0a\x34";
    unsigned char cipher[] = "\xd0\x6b\xf9\xd0\xda\xb8\xe8\xef\x88\x06\x60\xd2\xaf\x65\xaa\x82";
    unsigned char ivec[]   = "\x09\x08\x07\x06\x05\x04\x03\x02\x01\x00\xA2\xB2\xC2\xD2\xE2\xF2";

    unsigned char out[] = "";
    AES_KEY aesKey = {};
    int aesKeyResult = AES_set_decrypt_key(key, 128, &aesKey);
    //printf("aesKeyResult = %d\n", aesKeyResult);

    AES_cbc_encrypt(cipher, out, 16, &aesKey, ivec, AES_DECRYPT);

    printf("cipher is %s\n", out);

    if(memcmp(out, plain, 16) == 0){
        printf("Use time is %lld\n", i);
        break;
    }

}
```

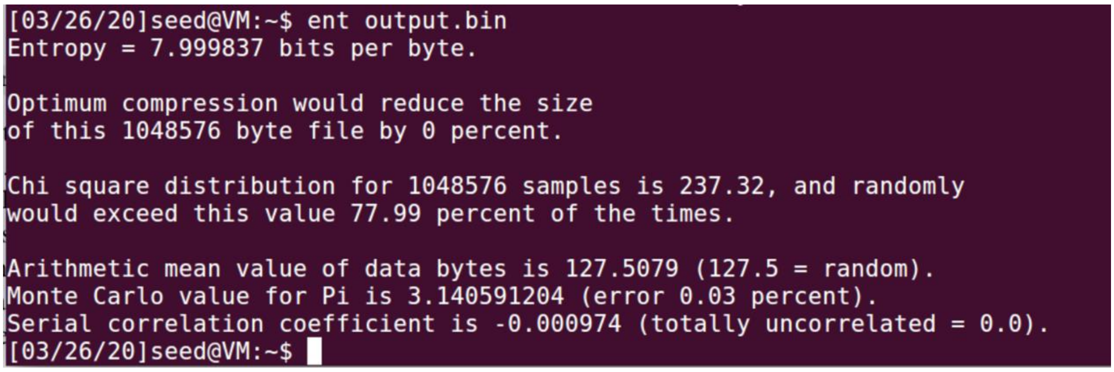
Task3. 使用 `watch -n .1 cat /proc/sys/kernel/random/entropy_avail` 指令後，在每次移動滑鼠或使用鍵盤，entropy 數量會增加。

Task4. 當滑鼠鍵盤不在動作時，entropy 增加會極為緩慢，導致數量不夠/dev/random 產生 pseudo random numbers，因此 hexdump 出來會停滯，直到 entropy 夠/dev/random 產生 pseudo random numbers。

Question: If a server uses /dev/random to generate the random session key with a client. Please describe how you can launch a Denial-Of-Service (DOS) attack on such a server.

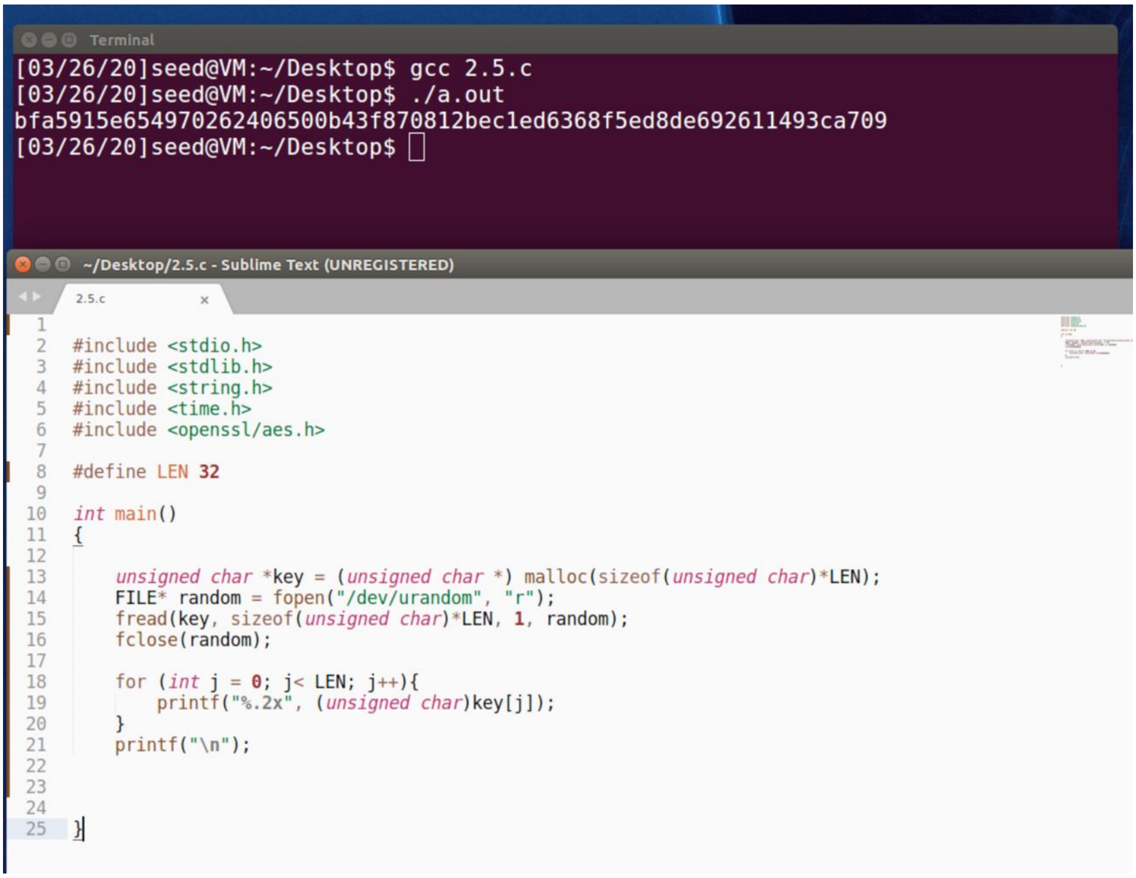
My answer: 持續耗盡 SERVER 的 entropy 導致目標無法產生新的 pseudo random numbers。

Task5. 使用 `cat /dev/urandom | hexdump` 指令會一直不斷顯示 pseudo random numbers 無關於滑鼠與鍵盤的動作。



```
[03/26/20]seed@VM:~$ ent output.bin
Entropy = 7.999837 bits per byte.

Optimum compression would reduce the size
of this 1048576 byte file by 0 percent.

Chi square distribution for 1048576 samples is 237.32, and randomly
would exceed this value 77.99 percent of the times.

Arithmetic mean value of data bytes is 127.5079 (127.5 = random).
Monte Carlo value for Pi is 3.140591204 (error 0.03 percent).
Serial correlation coefficient is -0.000974 (totally uncorrelated = 0.0).
[03/26/20]seed@VM:~$ 
```

產生 256-bits 長度的 Key



```
[03/26/20]seed@VM:~/Desktop$ gcc 2.5.c
[03/26/20]seed@VM:~/Desktop$ ./a.out
bfa5915e654970262406500b43f870812bec1ed6368f5ed8de692611493ca709
[03/26/20]seed@VM:~/Desktop$ 
```

```c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <openssl/aes.h>

#define LEN 32

int main()
{

    unsigned char *key = (unsigned char *) malloc(sizeof(unsigned char)*LEN);
    FILE* random = fopen("/dev/urandom", "r");
    fread(key, sizeof(unsigned char)*LEN, 1, random);
    fclose(random);

    for (int j = 0; j< LEN; j++){
        printf("%.2x", (unsigned char)key[j]);
    }
    printf("\n");



}
```