

# **CSI Rapport For Lab 3 Exo 3 and 7**

**Guebli Ayoub Abdessami Group 2**

Exo 03 :

To do our calculations ,first we have this 2 method , to convert string to number and number to string:

```
def LtoN(a):  
    return [ord(c) - ord('A') for c in a.upper()]  
  
def NtoL(a):  
    return "".join(chr(num + ord('A')) for num in a)
```

So after this we can calculate, we will need to calculate the diff between this 2 word we have ( "HQQYAJT" and "RJAJPWG" ) so we used this method to do that

```
def sub_mod_26(t1, t2):  
    n1 = LtoN(t1)  
    n2 = LtoN(t2)  
    return [(num1 - num2) % 26 for num1, num2 in zip(n1, n2)]
```

So after calculating the diff between this words , we opened the dictionary and started adding the diff to all words of this dictionary by the next method :

```
def add_mod_26(t1, diff):  
    n1 = LtoN(t1)  
    return NtoL([(num1 + d) % 26 for num1, d in zip(n1, diff)])
```

and checking if the word is true by checking in the dictionary again , so the code will be like this

exo3.py X

exo3.py > add\_mod\_26

```
1 def LtoN(a):
2     return [ord(c) - ord('A') for c in a.upper()]
3
4 def NtoL(a):
5     return "".join(chr(num + ord('A')) for num in a)
6
7 def sub_mod_26(t1, t2):
8     n1 = LtoN(t1)
9     n2 = LtoN(t2)
10    return [(num1 - num2) % 26 for num1, num2 in zip(n1, n2)]
11
12 def add_mod_26(t1, diff):
13     n1 = LtoN(t1)
14     return NtoL([(num1 + d) % 26 for num1, d in zip(n1, diff)])
15
16 w1 = "HQQYAJT"
17 w2 = "RJAJPWG"
18 diff = sub_mod_26(w1, w2)
19 possible_pairs = []
20
21 with open('fr.txt', 'r') as file:
22     words = [line.strip().upper() for line in file if len(line.strip()) == 7]
23     words_set = set(words)
24
25 for word1 in words:
26     expected_word2 = add_mod_26(word1, diff)
27     if expected_word2 in words_set:
28         possible_pairs.append((word1, expected_word2))
29
30 for pair in possible_pairs:
31     print(pair)
32
33
34
35
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
env (env)-(kali@kali) - [~/Desktop/Mater1s2/CSI/tp]
• $ /home/kali/Desktop/Mater1s2/CSI/tp/env/bin/python /home/kali/Desktop/Mater1s2/CSI/tp/exo3.py
('MASQUER', 'CHIFFRE')

env (env)-(kali@kali) - [~/Desktop/Mater1s2/CSI/tp]
o $
```

## Exo 07 :

### 1- everytime a new key

```
env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ /home/kali/Desktop/Mater1s2/CSI/tp/env/bin/python /ho
('MASQUER', 'CHIFFRE')

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ gcc exo07.c -o exo07

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979186
9d0b6d7b10a3e0686b8631ef63ff08729

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979188
b27b7448aec5f54d0a24192bbd1f3bb3

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979190
5bc4c48ef603044221335635f3ff1752

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979191
4eb9690a624fe02dacfbcb56f3bdd11c5

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $
```

### 2- same key everytime

```
env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ gcc exo07.c -o exo07

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979361
67c6697351ff4aec29cdbaabf2fbe346

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979363
67c6697351ff4aec29cdbaabf2fbe346

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979364
67c6697351ff4aec29cdbaabf2fbe346

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $ ./exo07
1714979365
67c6697351ff4aec29cdbaabf2fbe346

env (env)-(kali@kali)-[~/Desktop/Mater1s2/CSI/tp]
• $
```

So the srand is responsible to create a key using the time, and every second the time changed so a new key

3- to get the right key, we first generate all the keys that are probably the ones, and searching the one by testing all those keys, so by that way, we created all the keys we can generate on that day by this C code (3 hours before, I didn't find the key so I did all the day) and I saved the time on Unix type, to make it easy to know what time the key generated:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16

int main() {
    int i;
    long long start_time, end_time;
    unsigned char key[KEYSIZE];

    struct tm timeinfo = {0};
    timeinfo.tm_year = 2018 - 1900; // Year since 1900
    timeinfo.tm_mon = 3;           // Month (0-11)
    timeinfo.tm_mday = 18;         // Day of the month
    timeinfo.tm_hour = 22;         // Hour
```

```

timeinfo.tm_min = 1;           // Minute
timeinfo.tm_sec = 49;          // Second
end_time = mktime(&timeinfo);
printf("Unix Timestamp: %lld\n", (long long)end_time);

start_time = end_time - 24 * 3600;

FILE *fp = fopen("keys.txt", "w");
if (fp == NULL) {
    printf("Failed to open file for writing.\n");
    return 1;
}

for (long long t = start_time; t <= end_time; t++) {
    srand(t);
    for (i = 0; i < KEYSIZE; i++) {
        key[i] = rand() % 256;
    }

    fprintf(fp, "%lld,", t);
    for (i = 0; i < KEYSIZE; i++) {
        fprintf(fp, "%.2x", (unsigned char) key[i]);
    }
    fprintf(fp, "\n");
}

fclose(fp);
printf("Done generating keys.\n");

return 0;
}

```

So i got the keys on the file "keys.txt", then i tried to use the Crypto library, to use the AES algorithm, then by that iv i have, and every key, i try to generate cipher by AES.new, and try to decrypt the cipher text and i should have the plain text to know that key is right so i did this

```

from Crypto.Cipher import AES

plaintext = bytes.fromhex('255044462d312e350a25d0d4c5d80a34')
ciphertext = bytes.fromhex('d06bf9d0dab8e8ef880660d2af65aa82')

```

```

iv = bytes.fromhex('09080706050403020100A2B2C2D2E2F2')

def try_decrypt(ciphertext, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted = cipher.decrypt(ciphertext)
    return decrypted

with open('keys.txt', 'r') as file:
    hex_keys = file.readlines()

for hex_key in hex_keys:
    date, key_hex = hex_key.split(',')
    key_hex = key_hex.strip()
    key = bytes.fromhex(key_hex)
    decrypted = try_decrypt(ciphertext, key, iv)
    print(decrypted)
    print ( plaintext )
    if decrypted == plaintext:
        print(f'Found key: {key.hex()}, timestamp: {date}')
        break
    else:

        print(f'Key {key.hex()} did not work.')

```

So by execute the next code i got this

**Found key: 95fa2030e73ed3f8da761b4eb805dfd7, timestamp: 1524017695**

And by using a website to get the time from unix type I got this time :

**GMT: Wednesday, April 18, 2018 2:14:55 AM**

4- Use a cryptographically secure randomnumber generator and avoid using predictable seeds like timestamps for cryptographic key generation.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key faef197aae8cdb97b23a02efaa666b01 did not work.
b'v\x8e\xd4\xbdn \xad,LXA\x0b8\xffw\x1e'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key bdad2df707a1205delle6c4b81e1271b did not work.
b'c5\xd4\x8e\x0e"\x82\xd6\x80r-\t\xd9|\x85\xc9'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key 9e924b7ea3fe8886bb58d38ea2bd78a4 did not work.
b'2\xbb\xc8\xfc\x8b4\x1ac\xe7\xa1\x94\xd0t\xbf\x02&'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key 0578dbb7195cee4b8a272a3e3f9a558a did not work.
b'\xaad\xf1b\x1e\xd7\x96a\xea46\xb4\xd9\xb9\x037'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key 60d7f7bcd700a0ae87f24ddf50c38fc did not work.
b'=i\xdc\xf7a\xd1E27Ls\xcb\xc2'\xeb\xa0'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key 68c6b35edac07e0832f147dff8d4b1e5 did not work.
b'T\x1at;\xe5=8\xcd\xfazt1\x92\xcbT\x04'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key 2e2a4a7b1f74a07ce0b49066465cf165 did not work.
b'\xd4\xe1\x19g\x17j\x1c\xfez\xeb\x84\x82o\x94{\xfa'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Key d37fef87fb9c4562d3e9c13fe6c574e did not work.
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
b'%PDF-1.5\n%\xd0\xd4\xc5\xd8\n4'
Found key: 95fa2030e73ed3f8da761b4eb805dfd7, timestamp: 1524017695

env (env)-(kali@kali) [~/Desktop/Mater1s2/CSI/tp]
└─$

```