

Applied Cryptography (UE20CS314)

Lab 3

Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

Section: F

Task 1: Generate Encryption Key in a Wrong Way

Step 1:-

Code:

```
seed@VM: ~/.../lab3
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
void main() {
    int i;
    char key[KEYSIZE];
    printf("%lld\n", (long long) time(NULL));
    srand (time(NULL));
    for (i = 0; i< KEYSIZE; i++) {
        key[i] = rand()%128;
        printf("%.2x", (unsigned char)key[i]);
    }
    printf("\n");
}
~
~
~
~
~
~
~
~
"task1.c" 15L, 298C written
```

Output:

```
[09/27/22] seed@VM: ~/.../lab3$ gcc task1.c -o task1
[09/27/22] seed@VM: ~/.../lab3$ ./task1
1664250219
30146a371a257e2305557c2971695126
[09/27/22] seed@VM: ~/.../lab3$ ./task1
1664250219
30146a371a257e2305557c2971695126
[09/27/22] seed@VM: ~/.../lab3$ ./task1
1664250220
6146736b2d79791866414803474f1600
[09/27/22] seed@VM: ~/.../lab3$
```

Observation:

A new key is generated every second as the seed value we pass is the time elapsed in seconds till that point.

Step 2:-

Code:

```
seed@VM: ~/.../lab3
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
void main() {
    int i;
    char key[KEYSIZE];
    printf("%lld\n", (long long) time(NULL));
    //srand (time(NULL));
    for (i = 0; i < KEYSIZE; i++) {
        key[i] = rand()%128;
        printf("%.2x", (unsigned char)key[i]);
    }
    printf("\n");
}
~
~
~
~
~
~
~
~
"task1.c" 15L, 300C written
```

Output:

```
[09/27/22] seed@VM:~/.../lab3$ gcc task1.c -o task1
[09/27/22] seed@VM:~/.../lab3$ gcc task1.c -o task1
[09/27/22] seed@VM:~/.../lab3$ ./task1
1664250261
67466973517f4a6c294d3a2b727b6346
[09/27/22] seed@VM:~/.../lab3$ ./task1
1664250262
67466973517f4a6c294d3a2b727b6346
[09/27/22] seed@VM:~/.../lab3$ ./task1
1664250262
67466973517f4a6c294d3a2b727b6346
[09/27/22] seed@VM:~/.../lab3$
```

Observation:

Here the same key is generated as there's no new seed value being passed to the rand function.

Task2: Guessing the key

Step 1:-

Output:

```
[09/27/22] seed@VM:~/.../lab3$ date -d "2018-04-17 21:08:49" +%s
1523979529
[09/27/22] seed@VM:~/.../lab3$ date -d "2018-04-17 23:08:49" +%s
1523986729
[09/27/22] seed@VM:~/.../lab3$
```

Observation:

The output is the time elapsed in seconds between the current time and the time specified in the argument.

Step 2:-

Code:

```
seed@VM: ~/.../lab3
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
void main() {
    int i, j;
    FILE *f;
    char key[KEYSIZE];
    int value1 = 1524013729, value2 = 1524020929;
    // value1 = output of date -d "2018-04-17 21:08:49" +%s ;
    // value2 = output of date -d "2018-04-17 23:08:49" +%s ;
    f = fopen("keys.txt", "w");
    for (j = value1; j <= value2; j++) {
        srand (j);
        for (i = 0; i < KEYSIZE; i++) {
            key[i] = rand()%256;
            fprintf(f, "%.2x", (unsigned char)key[i]);
        }
        fprintf(f, "\n");
    }
}
~
~
"task2.c" 21L, 507C                                     16,23-44
```

Output:

```
[09/27/22] seed@VM:~/.../lab3$ gcc task2.c -o task2
[09/27/22] seed@VM:~/.../lab3$ ./task2
[09/27/22] seed@VM:~/.../lab3$ head keys.txt
9a6226fc01a201b82b7d42caa7de3e05
12d494f3e5506c3fc152d68ae5d35bc8
64b838761768baa431899b84dc5bbed0
fd9b1b3ae04452506a7f269b77d95e8e
9a45a8c0eea61d185e2e896ea1e96167
0d858bd80cb81b68981fc6ab1f6b6ff0
405350cf2bf03e912e03bba28a2a3cc6
66b32d34c8315750343b34fbf329b8b5
794885b8757f4791ee06970ed2c1f92b
c270b9219acd47d50997e8404ef066f3
[09/27/22] seed@VM:~/.../lab3$ wc -l keys.txt
7201 keys.txt
```

Observation:

The output is all the possible keys that could have been generated between the time of value1 and value2.

Step 3:-

Code:

```
seed@VM: ~/.../lab3
# decrypt.py
from Crypto import Random
from Crypto.Cipher import AES
file = open("keys.txt", "r")
ciphertext = "d06bf9d0dab8e8ef880660d2af65aa82"
for i in range(0,7200):
    str = file.readline()
    key = bytes.fromhex(str[:-1]).decode("hex")
    IV = bytes.fromhex("09080706050403020100A2B2C2D2E2F2".lower())
    plaintext1 = bytes.fromhex("255044462d312e350a25d0d4c5d80a34")
    cipher = AES.new(key, AES.MODE_CBC, IV)
    encrypted = cipher.encrypt(plaintext1)
    print("Encrypted: ",encrypted.hex())
    if ciphertext == encrypted.hex():#.encode("hex")[0:32]:
        print("")
        print("Match found")
        print("key: "+str[:-1])
        print("Ciphertext: " + ciphertext)
        print("Encrypted: " + (encrypted).hex())
        print("")
~
~
~
"task2.py" [New] 20L, 756C written 15,4
```

Output:

```
seed@VM: ~/.../lab3
seed@VM: ~/.../lab3
[09/27/22] seed@VM: ~/.../lab3$ python3 task2.py
Encrypted: 00097874f2a1109ab6871dbff7599c4c
Encrypted: af821fc8ddfa989c3a441e6ae0b523f2
Encrypted: f1b153c3020d352cf4f139d9cffa4e4e
Encrypted: 5a6a81b179fb48bb3758ad4e6125fa58
Encrypted: 840ad9469fbda67c416950d96bc035c5
Encrypted: ab4b2801ad7ae46929ba3f40beef0498
Encrypted: 1e22e24ef879e93781764267eccd2ea7
Encrypted: 92b5f95cb71210a8fbbd46dbd57f88e7
Encrypted: 96e32a089b2a0e5a8ba29fd7f02cc328
Encrypted: 4c5dbb20950b0f84c3deaa210e15a1d9
Encrypted: 97e3ed76ed9817358fe6ed776ea3853d
Encrypted: d3b7d7e9ee4fa61c33d422bb67be9c6c
Encrypted: bf627543c69902bb2b0d072d8c85ed59
Encrypted: 52012a0f9c0664ac8048265145f7a0b5
Encrypted: ff2031fc1fef52480bba80ebc227b58
Encrypted: 86e44d02fc350adcfff0176c907f79e3
Encrypted: c831cec4f70735a90cde6afdae32640b
Encrypted: aea5bfff68745e7e38f43073d6cb460e1
Encrypted: 47eb38ae5ea269ea489f9ace32656124
Encrypted: 87341c1a8506c63a30c57ebe512d7f59
Encrypted: 4fe1b9c421ad1ae74f05a5f8e7e7b498
Encrypted: 7779d04434d140a9485cbcdee16eafa5
Encrypted: 16322a1c86a6ea0aedb7705eca1d468c
```

Observation:

Gives the output of encrypted plaint text over all the stored keys in keys.txt file.

Code:

```
seed@VM: ~/.../lab3 x seed@VM: ~/.../lab3 x seed@VM: ~
# decrypt.py
from Crypto import Random
from Crypto.Cipher import AES
file = open("keys.txt", "r")
ciphertext = "d06bf9d0dab8e8ef880660d2af65aa82"
for i in range(0,7200):
    str = file.readline()
    key = bytes.fromhex(str[:-1]).decode("hex")
    IV = bytes.fromhex("09080706050403020100A2B2C2D2E2F2".lower())
    plaintext1 = bytes.fromhex("255044462d312e350a25d0d4c5d80a34")
    cipher = AES.new(key, AES.MODE_CBC, IV)
    encrypted = cipher.encrypt(plaintext1)
    #print("Encrypted: ",encrypted.hex())
    if ciphertext == encrypted.hex():#.encode("hex")[0:32]:
        print("")
        print("Match found")
        print("key: "+str[:-1])
        print("Ciphertext: " + ciphertext)
        print("Encrypted: " + (encrypted).hex())
        print("")
~
~
~
"task2.py" 20L, 757C written 13,5
```

Output:

```
[09/27/22] seed@VM: ~/.../lab3$ python3 task2.py

Match found
key: 95fa2030e73ed3f8da761b4eb805dfd7
Ciphertext: d06bf9d0dab8e8ef880660d2af65aa82
Encrypted: d06bf9d0dab8e8ef880660d2af65aa82

[09/27/22] seed@VM: ~/.../lab3$
```

Observation:

Commenting out the unwanted output we now check if the cipher text we need is present in the list of encrypted values.

Task 3: Measure the Entropy of Kernel

Code:

```
[09/27/22] seed@VM: ~/.../lab3$ watch -n .1 cat /proc/sys/kernel/random/entropy_aval  
[09/27/22] seed@VM: ~/.../lab3$
```

Output:

```
seed@VM: ~/.../lab3 seed@VM: ~/.../lab3  
Every 0.1s: cat /proc/sys/kernel/random/entropy... VM: Tue Sep 27 10:08:29 2022  
3060
```

Observation:

The entropy changes quickly when the mouse is moved faster and slower if it is not moved around.

Task 4: Get Pseudo Random Numbers from /dev/random

Step 1:-

Output:

```
[09/27/22] seed@VM: ~/.../lab3$ watch -n .1 cat /proc/sys/kernel/random/entropy_aval
```

```
seed@VM: ~/.../lab3  
Every 0.1s: cat /proc/sys/kernel/random/entropy... VM: Tue Sep 27 17:00:17 2022  
1432
```



```
seed@VM: ~/.../lab3  seed@VM: ~/.../lab3  seed@VM: ~/.../lab3
[09/27/22] seed@VM: ~/.../lab3$ cat /dev/random | hexdump
00000000 5c62 7870 4e39 5e78 6041 bdaa b304 d613
00000010 8a3e 0177 d12e 60ef dd95 e9c2 c043 6241
00000020 4737 849e 53f3 bf20 b581 a77a ebe0 3b4e
00000030 cd7c c19f 7489 ac9b e2c8 e073 1536 2c8b
00000040 d728 424f 2143 5a42 e852 6cfc 10c8 b9ac
00000050 f264 9797 4deb 97a5 1140 bc59 1426 3e69
00000060 10dc a248 714d c4b6 a99c dfe8 d2bd 0eb5
00000070 6e19 856e 0c7a a471 491f ca85 c5ff de12
00000080 aa61 3e22 1b22 cf9f 62a0 0cc9 28ef 396c
00000090 5927 10b5 a9f6 f76e e956 4fd0 9f22 92ac
000000a0 99fe d535 2062 545c 3489 c628 2156 5345
000000b0 18fb 440a c9f8 c2c5 be22 5d4e ea74 f227
000000c0 e0fa 19bf 9474 7a7d 67c2 8ab3 1a75 8b80
000000d0 9ad2 e17e 26c2 40fa 51a4 368b b280 284e
000000e0 0635 21c5 0a93 98a2 0555 7cee 7c10 87d2
000000f0 24b4 f1a3 183e c165 8c19 9041 70f5 7764
00001000 3063 e619 8c68 502a 3a7d e6e7 3268 f498
00001100 63d5 204e 0fd4 ce21 631a 23e7 0add ca69
```

Observation:

Movement of mouse and keyboard increases the entropy and low activity reduces the change rate of entropy.

Task 5: Get Pseudo Random Numbers from /dev/urandom

Step 1:-

Output:

```
seed@VM: ~/.../lab3  seed@VM: ~/.../lab3  seed@VM: ~/.../lab3
[09/27/22] seed@VM: ~/.../lab3$ cat /dev/urandom | hexdump
00000000 a72e 5270 42d3 b338 2194 aae8 0cc8 e7aa
00000010 48c3 515d 48f1 22bb 078c 2e7e ee8f ccf6
00000020 dd04 bd9f ee7f 4d58 770e c3c9 2c95 68ad
00000030 f285 57bb 4d89 d9f7 35fd b7f8 c0ec 5cc9
00000040 6aea cb4d 75b8 5b6b 4fdf e229 472c 249e
00000050 a258 84b3 b5f5 d5a5 e335 9840 28b4 9529
00000060 d9d4 5184 8a94 68b0 6972 3b17 b550 22b1
00000070 4fe4 d6f6 7fbc 85b3 a0e2 f384 8310 639b
00000080 6b26 6c52 2c5e b305 fb22 ab74 ad4f 69c1
00000090 5851 4c51 f61d 58a0 650f 830a 97b2 e12c
000000a0 a14d fc5f e053 26f3 6789 b859 4a07 2cc8
000000b0 c2b4 17e4 d2aa f520 9e7b 9508 c2c8 4d5f
000000c0 b66e b715 5dbc 4833 92fc 0f02 2d8c d20e
000000d0 10d2 fa54 2fb7 b7d6 dc04 44c0 dd6c 273d
000000e0 8fde 53f6 fb04 1d38 c957 6759 e149 8031
000000f0 7333 0872 bf4f 98fa a68c 5aca 47dd b7c8
00001000 9e58 32ec 73cd 3f83 cfec acf2 3725 9ca4
00001100 ede0 830c 2d63 3fb8 36c8 99d8 8efc d4ad
00001200 3dba 9d61 63cd 66c2 d9c1 1835 1f2c 5968
00001300 a471 f5b4 7bf5 8940 7950 3bf9 df0b 5a1f
00001400 07ff a7fd d83a 27ef 36b1 d2ef 06b2 0bb6
00001500 4c10 12fc 5f09 f9e8 bb4a e09a b6ba 04ba
00001600 1bb9 4883 5cdb bde6 1c6f 9295 8373 58d7
```

```
seed@VM: ~  
Every 0.1s: cat /proc/sys/kernel/random/entropy... VM: Tue Sep 27 13:31:47 2022  
795
```

Step 2:-

Output:

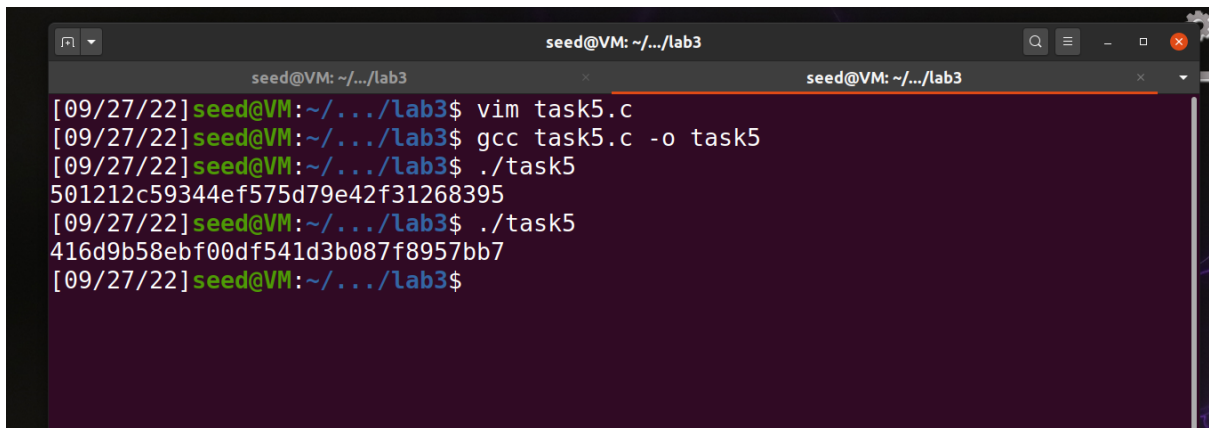
```
seed@VM: ~  
[09/27/22] seed@VM:~$ head -c 1M /dev/urandom > output.bin  
[09/27/22] seed@VM:~$ ent output.bin  
Entropy = 7.999855 bits per byte.  
  
Optimum compression would reduce the size  
of this 1048576 byte file by 0 percent.  
  
Chi square distribution for 1048576 samples is 210.96, and randomly  
would exceed this value 97.96 percent of the times.  
  
Arithmetic mean value of data bytes is 127.5175 (127.5 = random).  
Monte Carlo value for Pi is 3.140499651 (error 0.03 percent).  
Serial correlation coefficient is -0.000820 (totally uncorrelated = 0.0).  
[09/27/22] seed@VM:~$
```

Step 3:-

Code:

```
seed@VM: ~/.../lab3  
#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
#define KEYSIZE 16  
void main() {  
    int i;  
    FILE *random;  
    unsigned char *key = (unsigned char *) malloc (sizeof (unsigned char) *  
KEYSIZE);  
    random = fopen("/dev/urandom", "r");  
    for (i = 0; i < KEYSIZE; i++) {  
        fread(key, sizeof(unsigned char) * KEYSIZE, 1, random);  
        printf("%.2x", *key);  
    }  
    printf("\n");  
    fclose(random);  
}  
~  
~  
~  
~  
~  
~  
"task5.c" [New] 16L, 385C written 15,3-10 All
```

Output:



```
seed@VM: ~/.../lab3
[09/27/22] seed@VM: ~/.../lab3$ vim task5.c
[09/27/22] seed@VM: ~/.../lab3$ gcc task5.c -o task5
[09/27/22] seed@VM: ~/.../lab3$ ./task5
501212c59344ef575d79e42f31268395
[09/27/22] seed@VM: ~/.../lab3$ ./task5
416d9b58ebf00df541d3b087f8957bb7
[09/27/22] seed@VM: ~/.../lab3$
```

Observation:

We use a better way of generating key using PRNG rather than relying on seed and time values.