

# Applied Cryptography

## Lab -03

Name	Naman Choudhary
SRN	PES2UG20CS209
Section	D

### Task 1: Generate Encryption Key in a Wrong Way

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
int 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()%256;
        printf("%.2x", (unsigned char)key[i]);
    }
    printf("\n");
    return 0;
}
```

Screenshot:

```
Lab3 — zsh — zsh (figterm) - zsh — 80x24
[naman2341@Namans-MacBook-Pro Lab3 % gcc task1.c -o task1
[naman2341@Namans-MacBook-Pro Lab3 % ./task1
1664251648
e1ec5ab5c0f240fb62a07c0196b32ef3
[naman2341@Namans-MacBook-Pro Lab3 % ./task1
1664251649
88dd34e043ba18faa6ee14562395e13b
[naman2341@Namans-MacBook-Pro Lab3 % ./task1
1664251650
2fce0e0ac583f0f9ea3cadacb0779482
[naman2341@Namans-MacBook-Pro Lab3 % ]
```

After commenting `srand(time(NULL));`

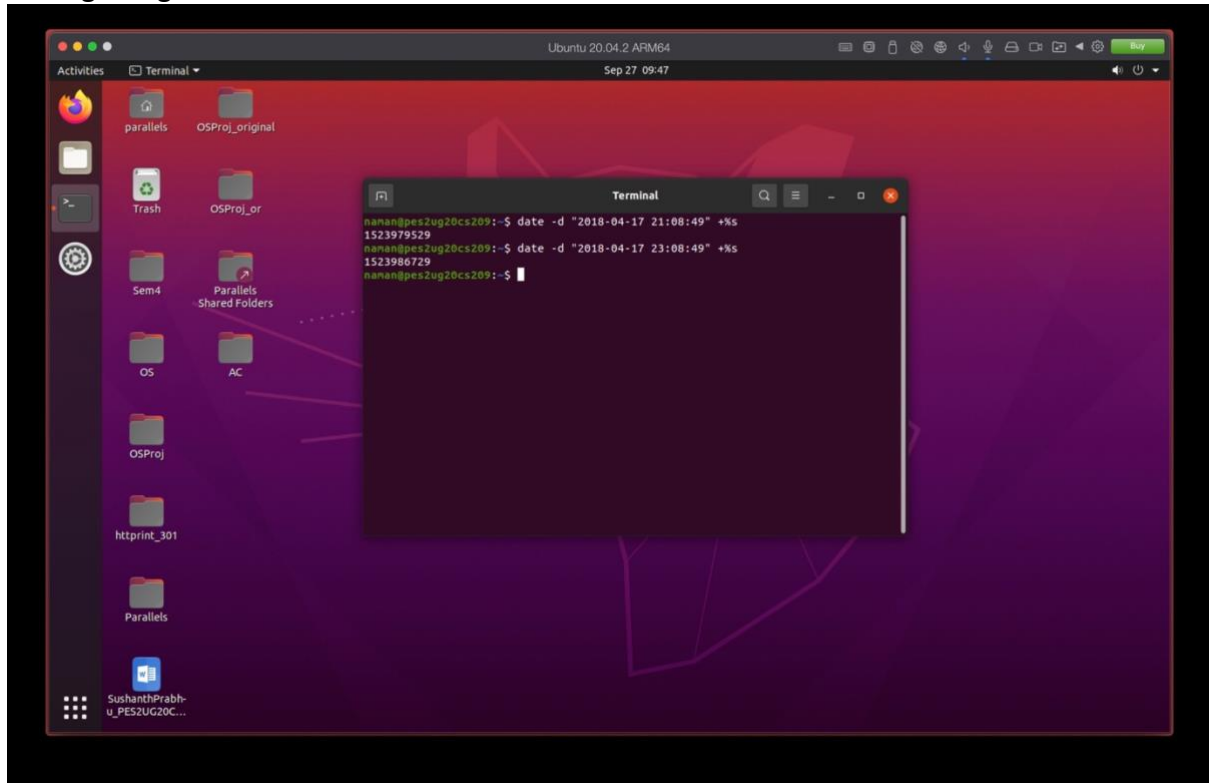
Screenshot:

```
Lab3 — zsh — zsh (figterm) - zsh — 80x24
[naman2341@Namans-MacBook-Pro Lab3 % gcc task1.c -o task1commented
[naman2341@Namans-MacBook-Pro Lab3 % ./task1commented
1664251701
a7f1d92a82c8d8fe434d98558ce2b347
[naman2341@Namans-MacBook-Pro Lab3 % ./task1commented
1664251703
a7f1d92a82c8d8fe434d98558ce2b347
[naman2341@Namans-MacBook-Pro Lab3 % ./task1commented
1664251703
a7f1d92a82c8d8fe434d98558ce2b347
[naman2341@Namans-MacBook-Pro Lab3 % ]
```

Observation: In case 1, we got different outputs every time, but in case 2, we get the same output on different runs

## Task 2: Guessing the Key

After getting date values:



Code:

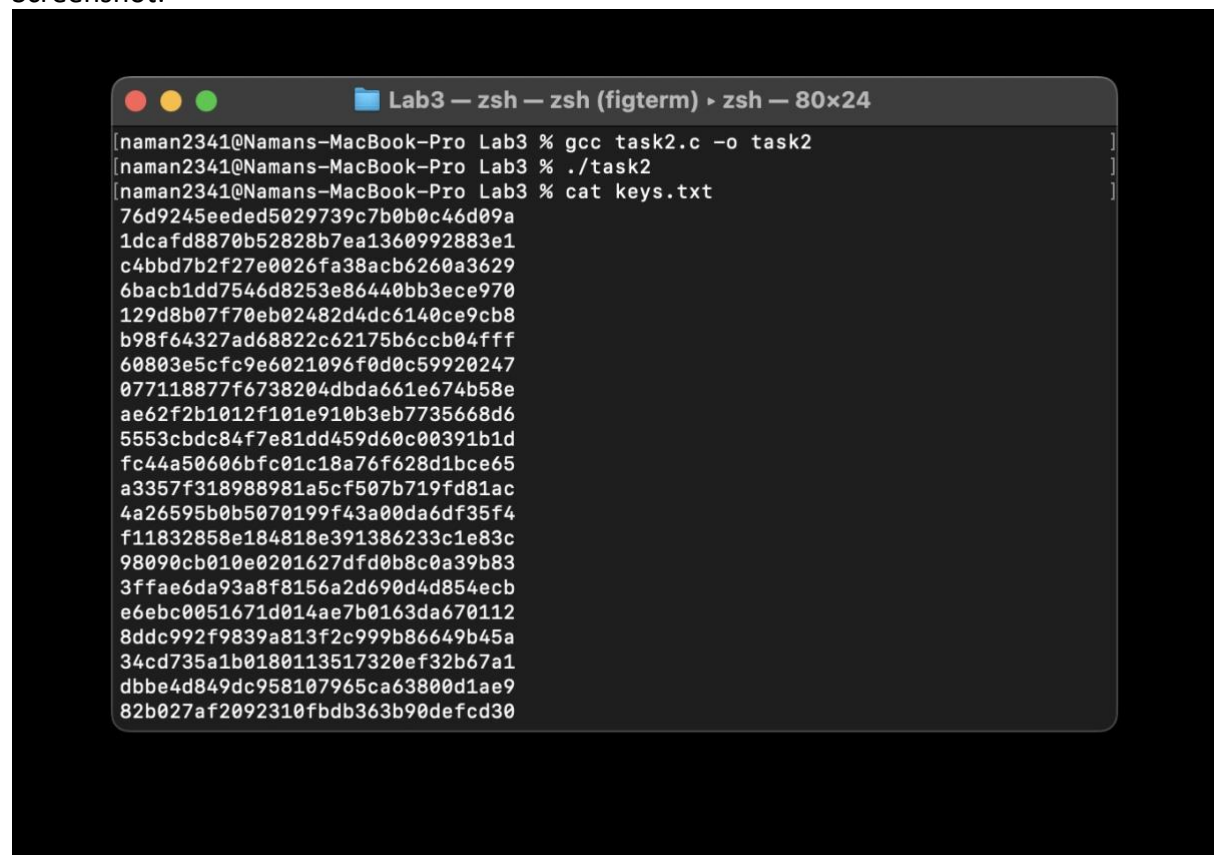
```
/* task2.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define KEYSIZE 16
int main() {
    int i, j;
    FILE *f;
    char key[KEYSIZE];
    int value1, value2;
    value1 = 1523979529;
    value2 = 1523986729;
    f = fopen("./keys.txt", "w");
```

```

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");
}
return 0;
}

```

Screenshot:



The screenshot shows a terminal window titled "Lab3 — zsh — zsh (figterm) • zsh — 80x24". The user is logged in as "naman2341@Namans-MacBook-Pro". The terminal shows the following commands and output:

```

[naman2341@Namans-MacBook-Pro Lab3 % gcc task2.c -o task2
[naman2341@Namans-MacBook-Pro Lab3 % ./task2
[naman2341@Namans-MacBook-Pro Lab3 % cat keys.txt
76d9245eeded5029739c7b0b0c46d09a
1dcafd8870b52828b7ea1360992883e1
c4bbd7b2f27e0026fa38acb6260a3629
6bacb1dd7546d8253e86440bb3ece970
129d8b07f70eb02482d4dc6140ce9cb8
b98f64327ad68822c62175b6ccb04fff
60803e5cfc9e6021096f0d0c59920247
077118877f6738204dbda661e674b58e
ae62f2b1012f101e910b3eb7735668d6
5553cbdc84f7e81dd459d60c00391b1d
fc44a50606bfc01c18a76f628d1bce65
a3357f318988981a5cf507b719fd81ac
4a26595b0b5070199f43a00da6df35f4
f11832858e184818e391386233c1e83c
98090cb010e0201627dfd0b8c0a39b83
3ffae6da93a8f8156a2d690d4d854ecb
e6ebc0051671d014ae7b0163da670112
8ddc992f9839a813f2c999b86649b45a
34cd735a1b0180113517320ef32b67a1
dbbe4d849dc958107965ca63800d1ae9
82b027af2092310fbdb363b90defcd30

```

Observation: Keys were generated in keys.txt

Code 2:

```

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() or (True):#.encode("hex")[0:32]:

        print("")

        print("Match found")

        print("key: "+str[:-1])

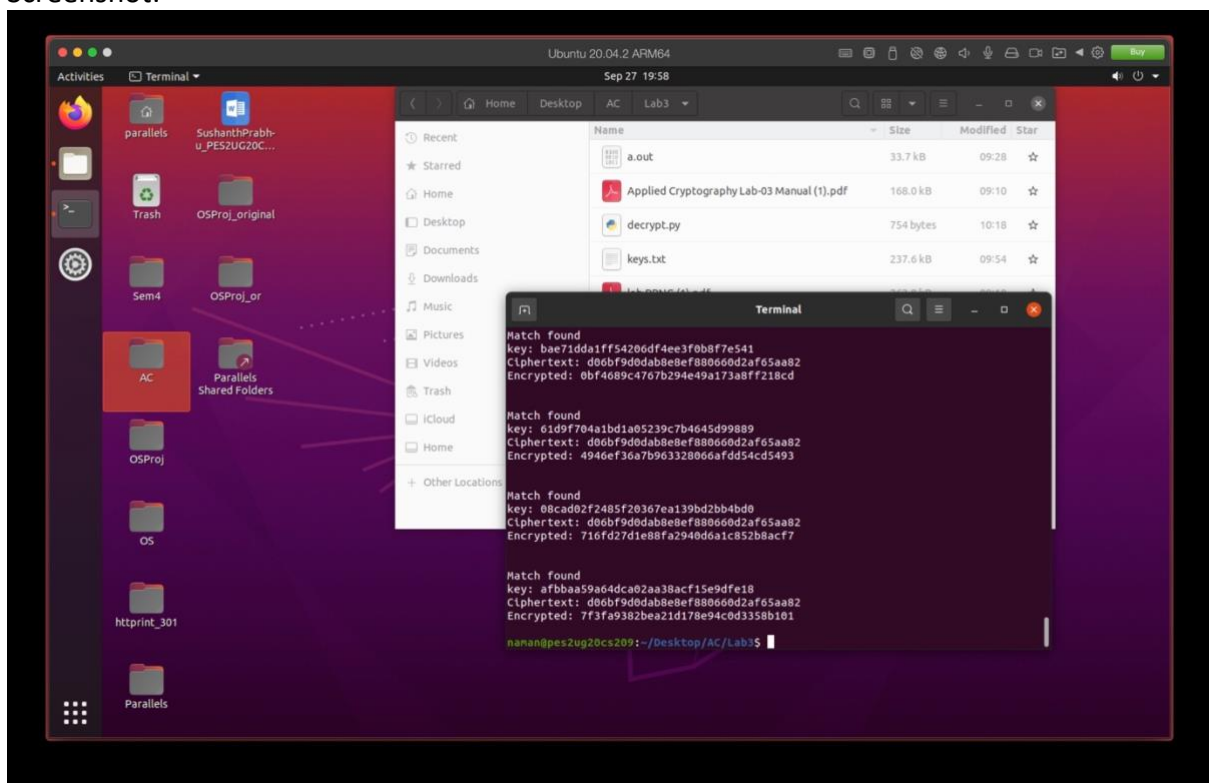
        print("Ciphertext: " + ciphertext)

        print("Encrypted: " + (encrypted).hex())

        print("")

```

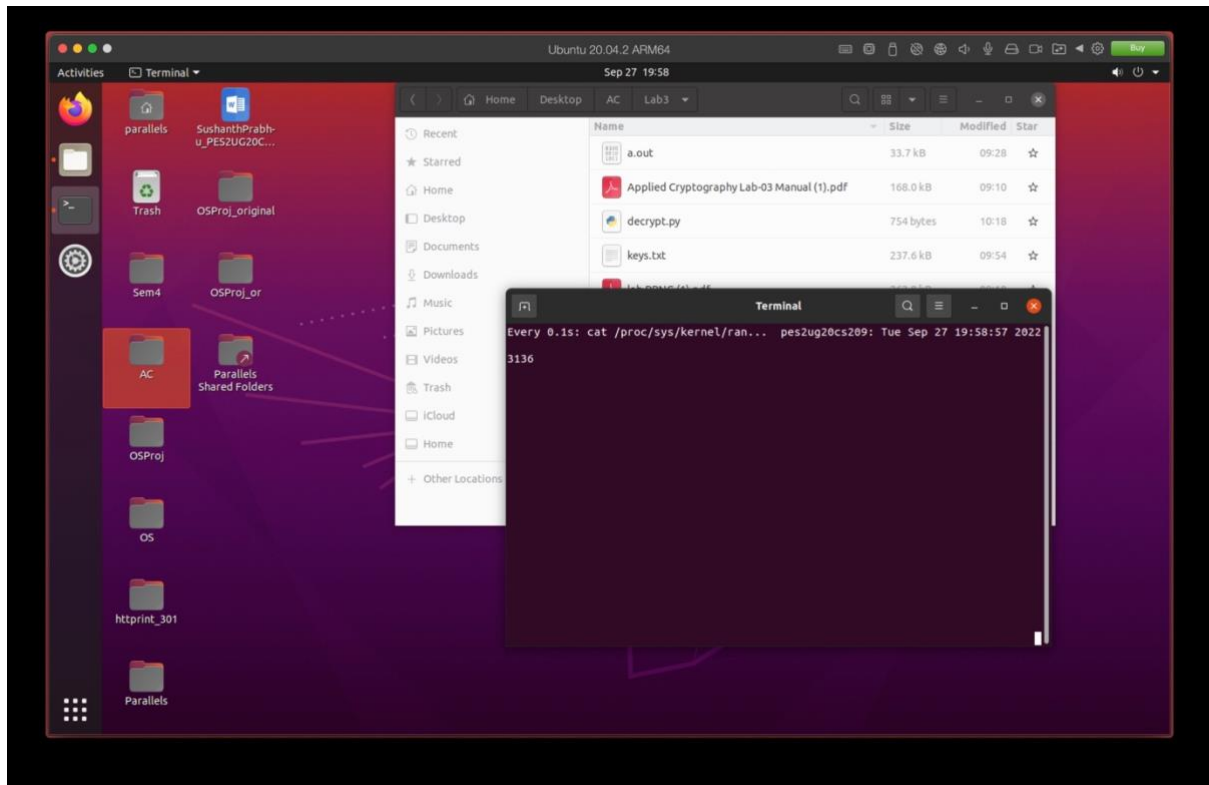
Screenshot:



Observation: The keys were generated correctly

### Task 3: Measure the Entropy of Kernel

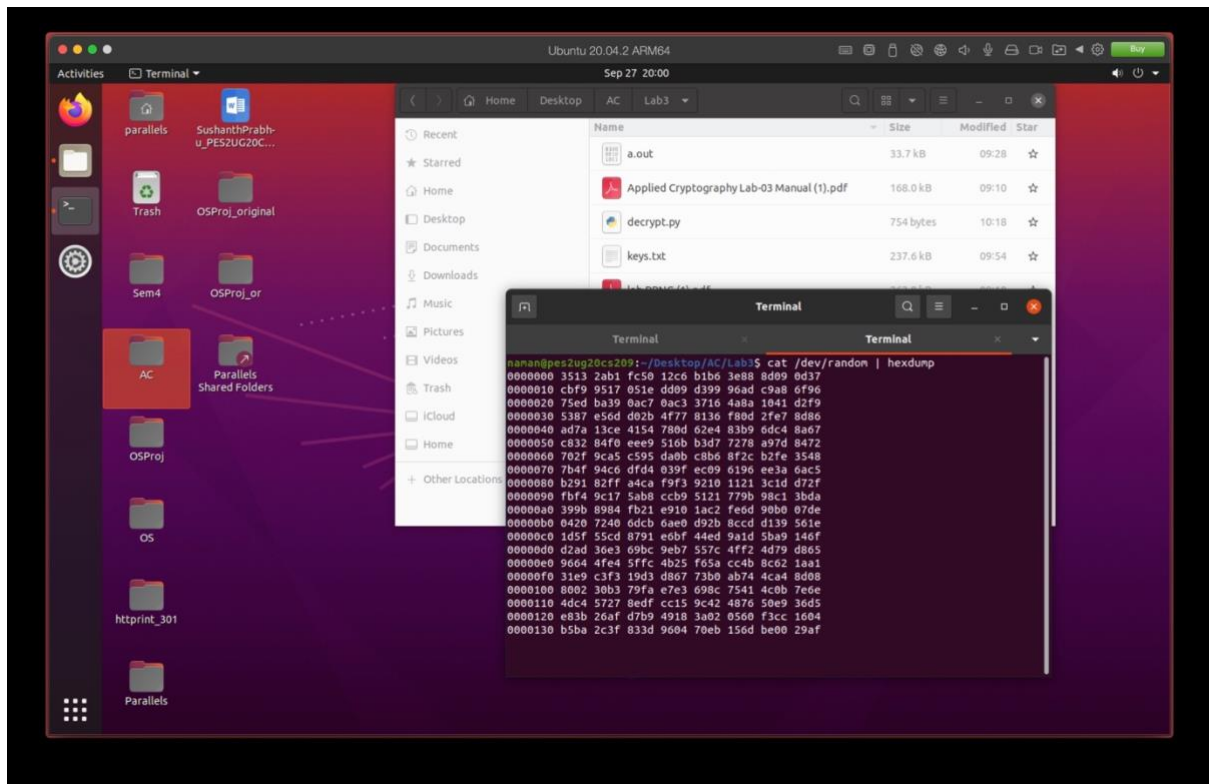
Screenshot:



Observation: The entropy increases with mouse movements and key presses, and comes back to 0 after a particular value

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

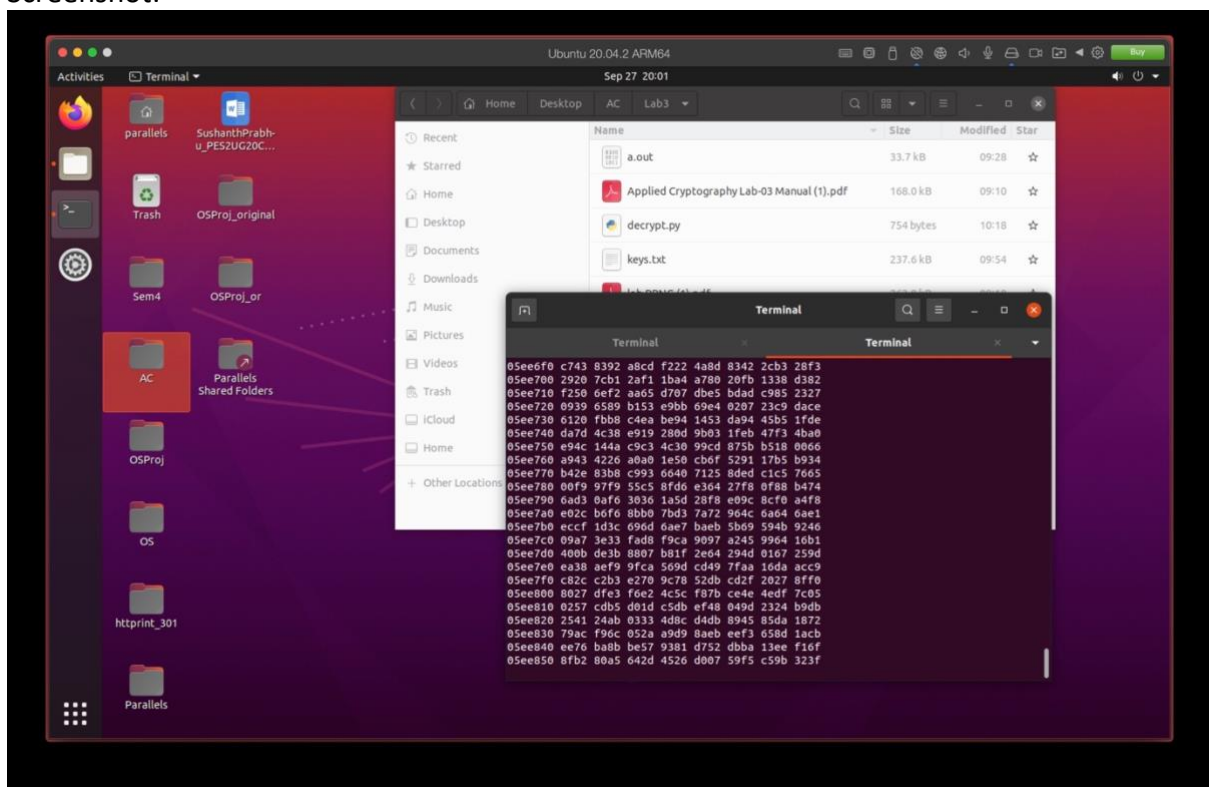
Screenshot:



Observation: hexdump generated hexes of the entropy, which increased after a large amount of mouse movement

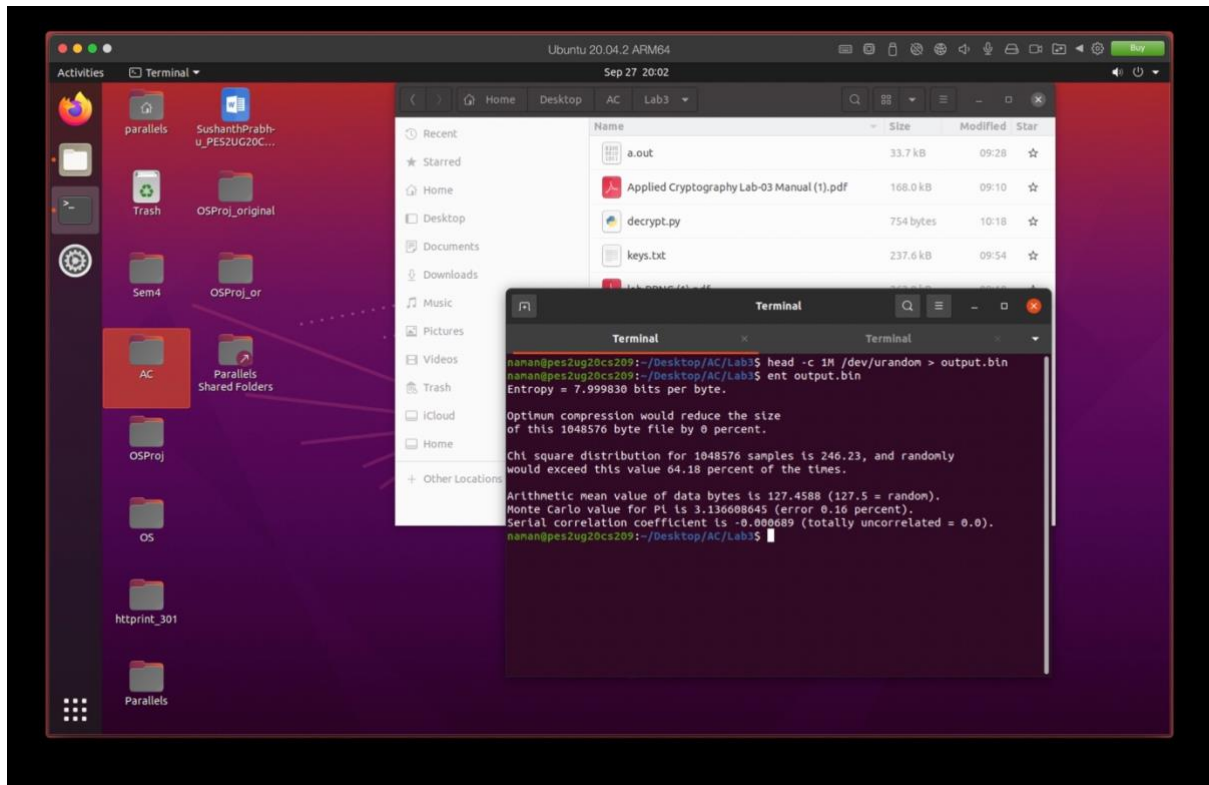
## Task 5: Get Pseudo Random Numbers from /dev/urandom

Screenshot:



Observation: hexdump generated hexes of the entropy, which increased after a very short amount of mouse movement

Screenshot:



Observation: We note that the command ent gives us the entropy per byte, with other details about the entropy

Screenshot:



