**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]);

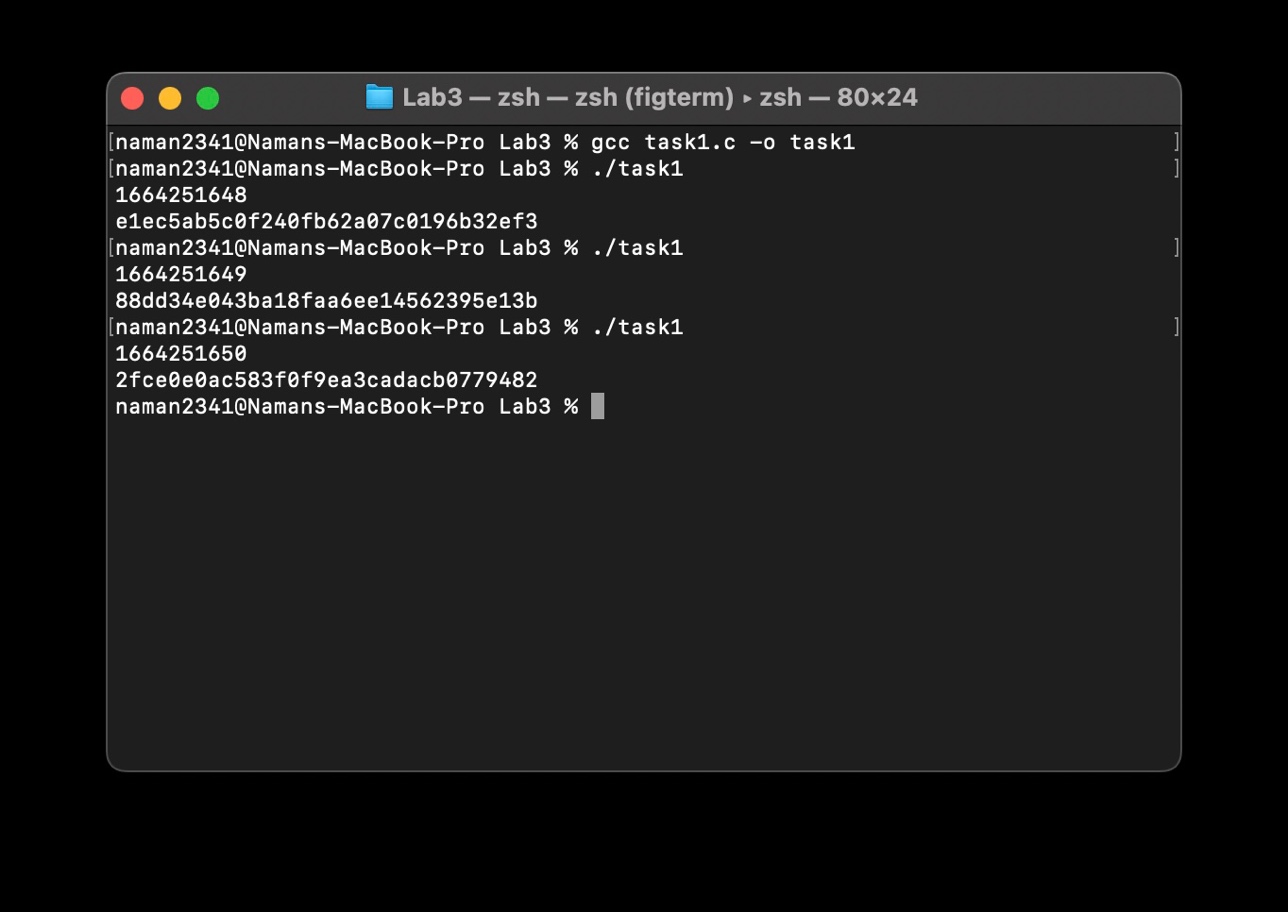
}

printf("\n");

return 0;

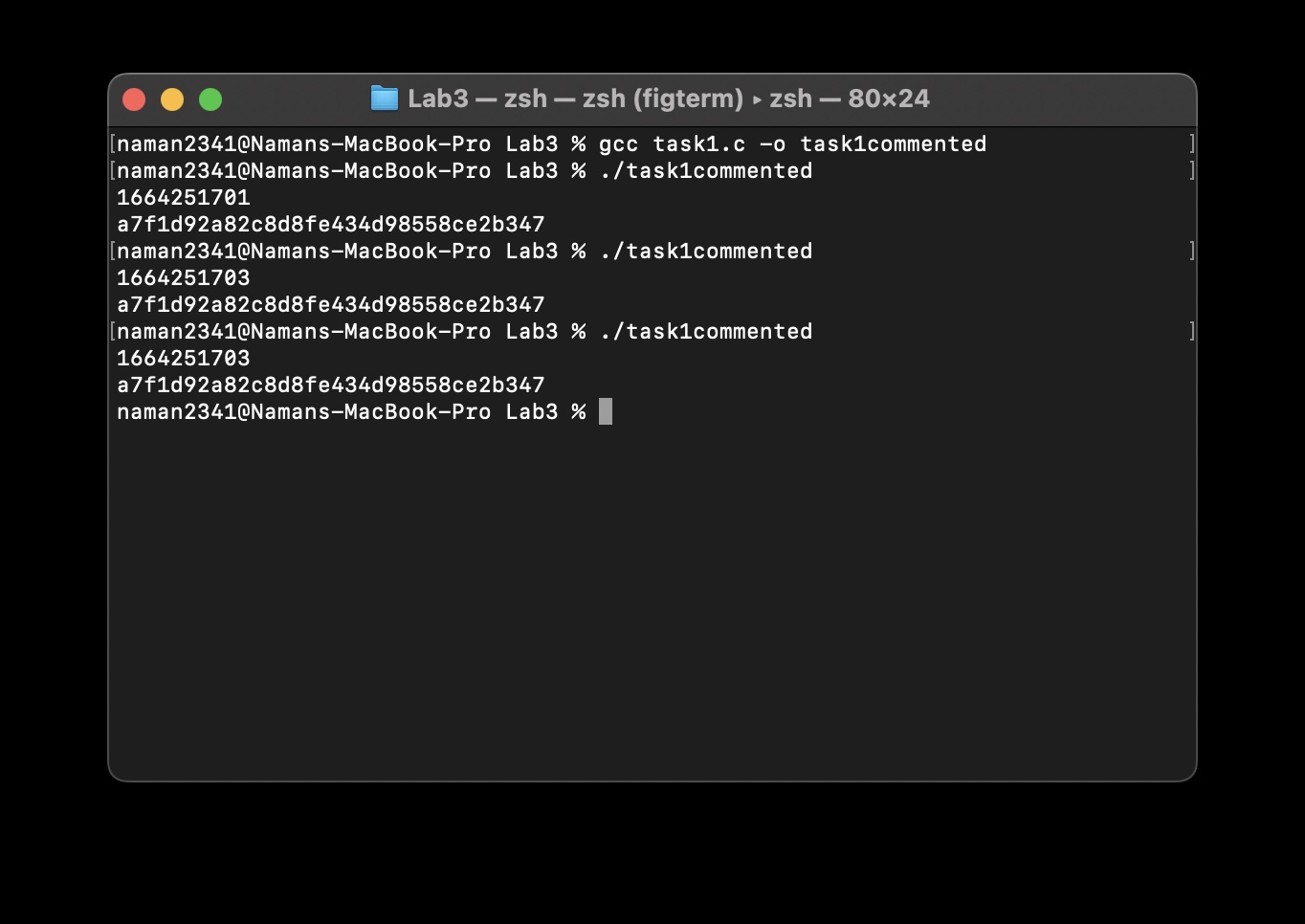
}

Screenshot:



After commenting srand(time(NULL));

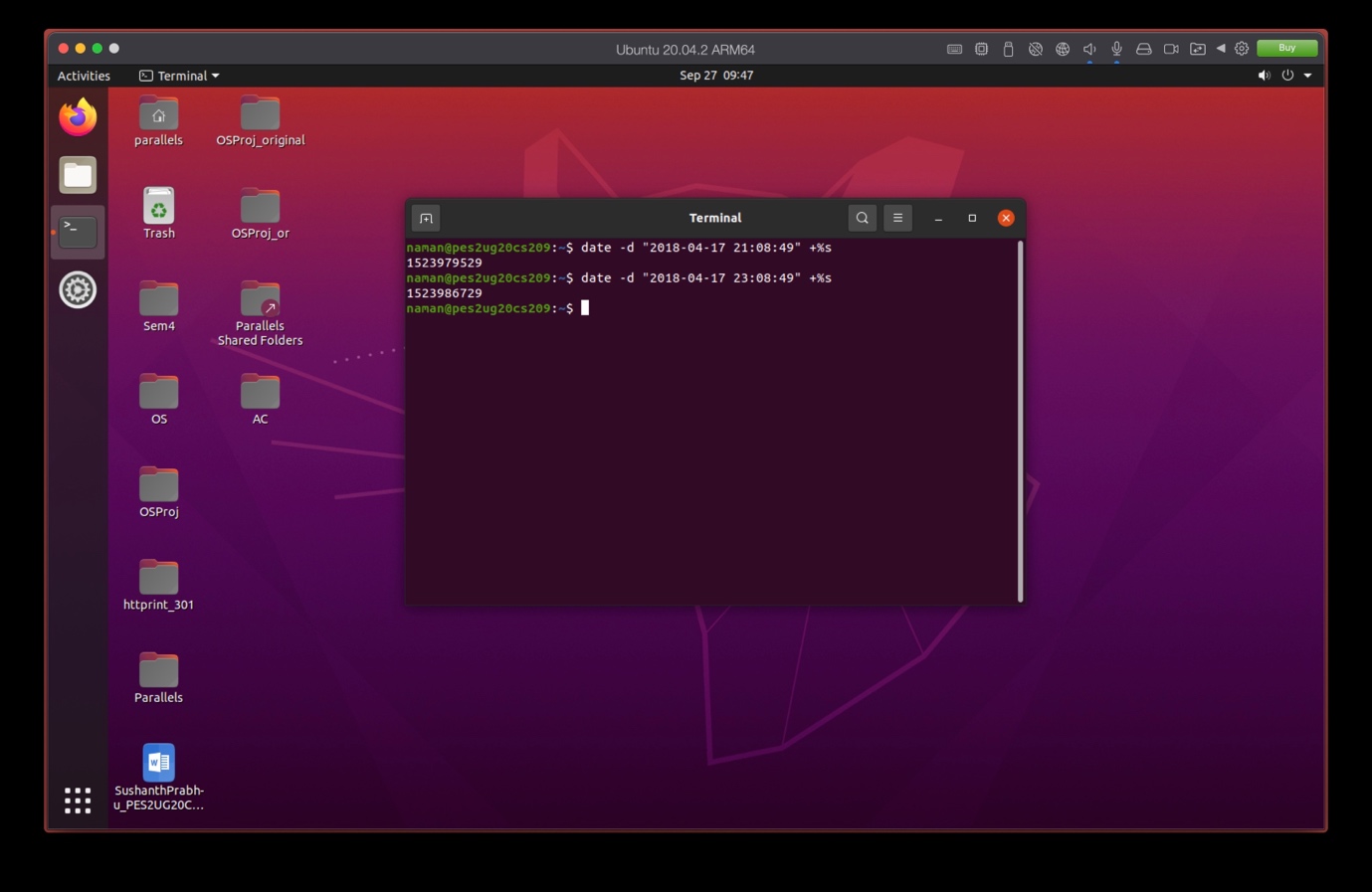
Screenshot:



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]);

}

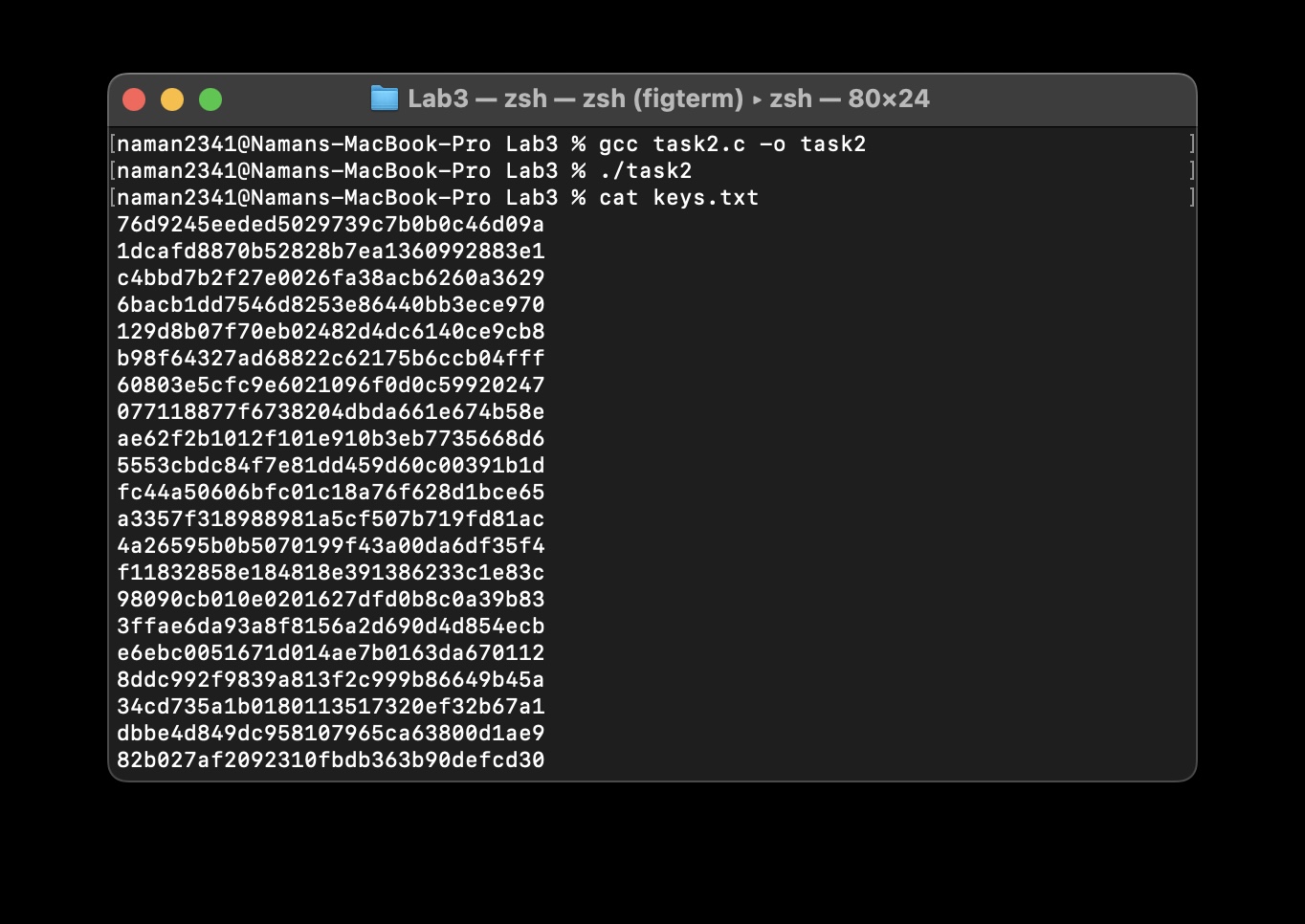
fprintf(f,"\n");

}

return 0;

}

Screenshot:



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")

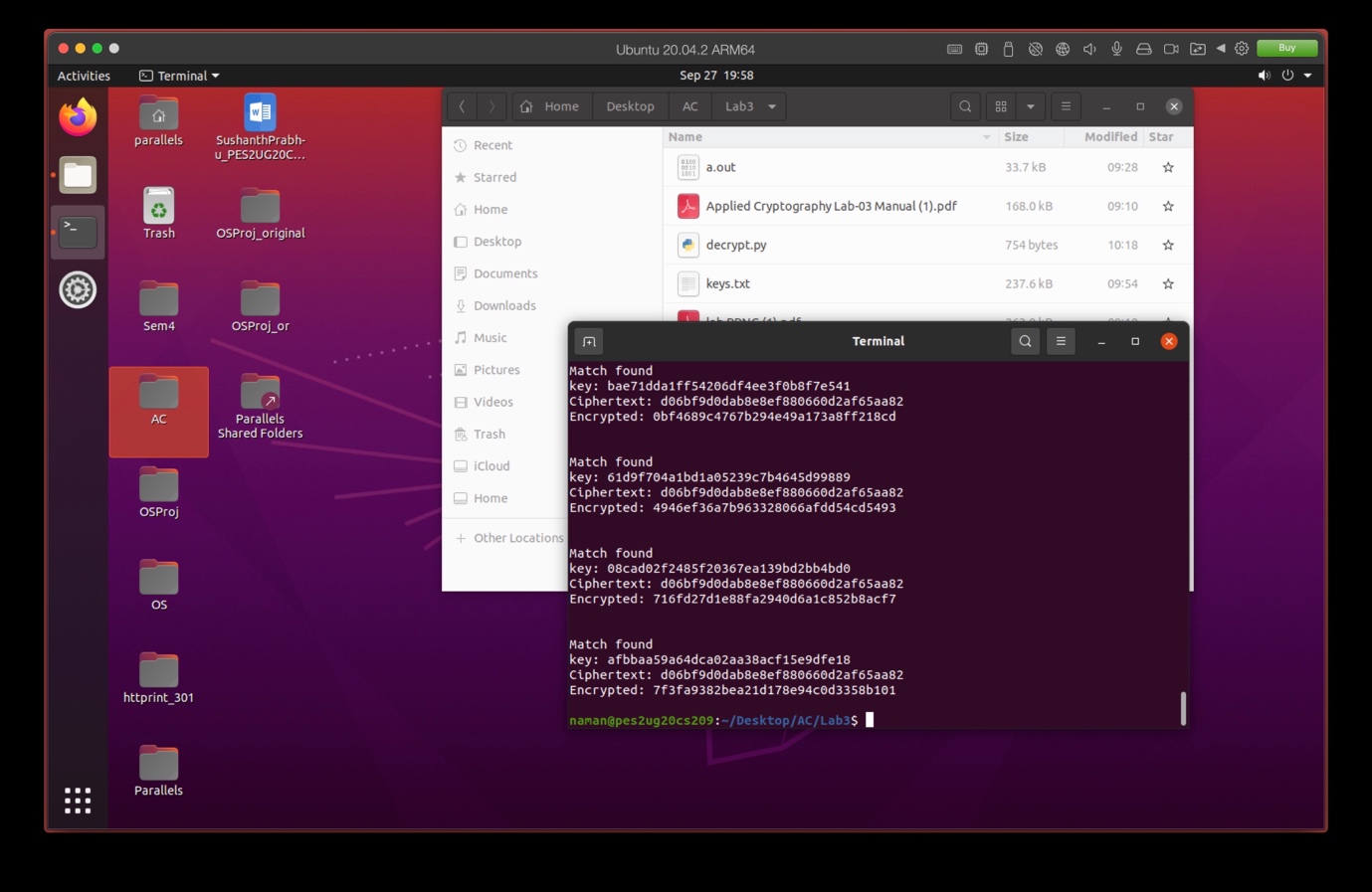
print("key: "+str[:-1])

print("Ciphertext: " + ciphertext)

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

print("")

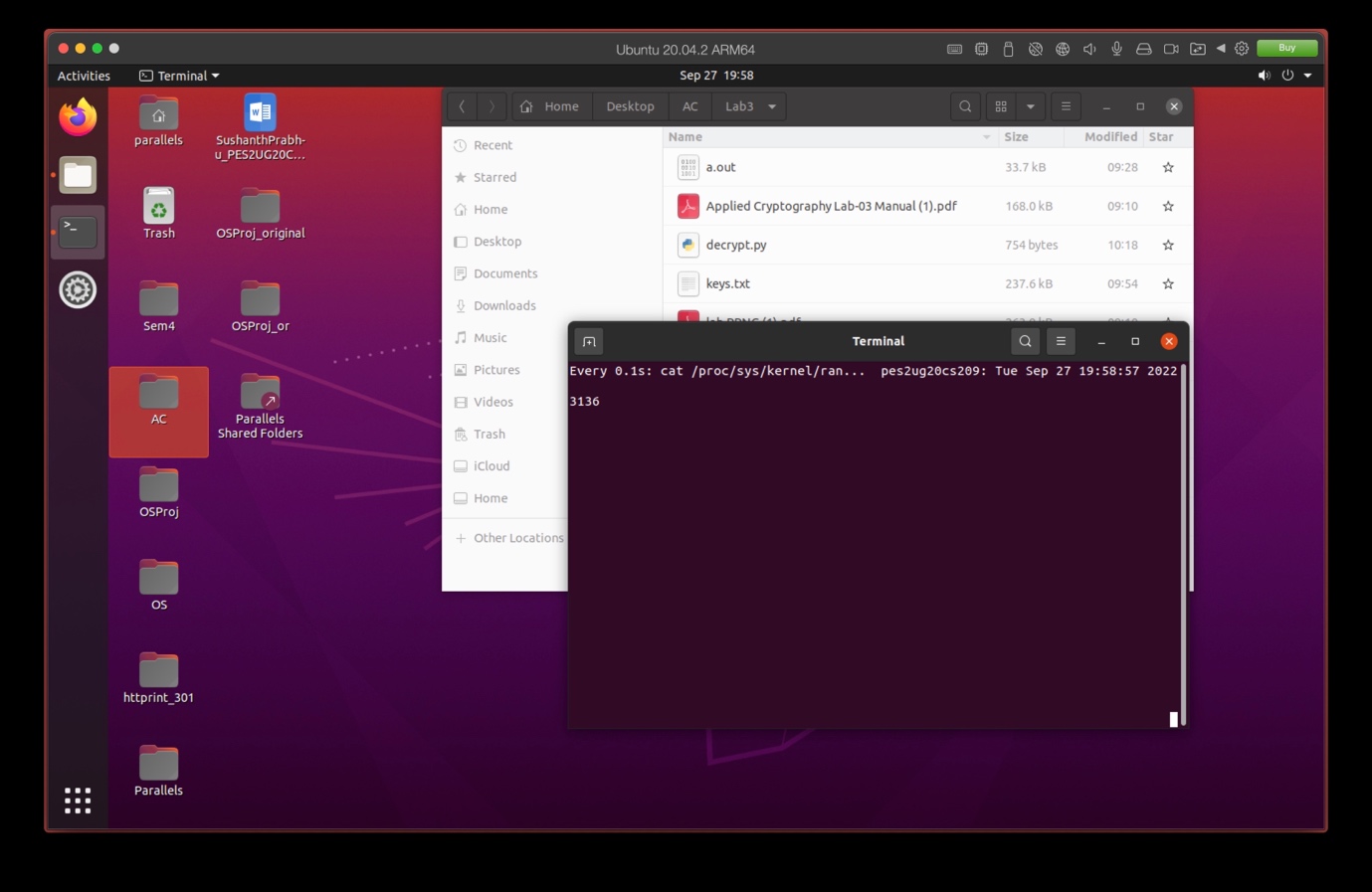
Screenshot:



Observation: The keys were generated correctly

Task 3: Measure the Entropy of Kernel

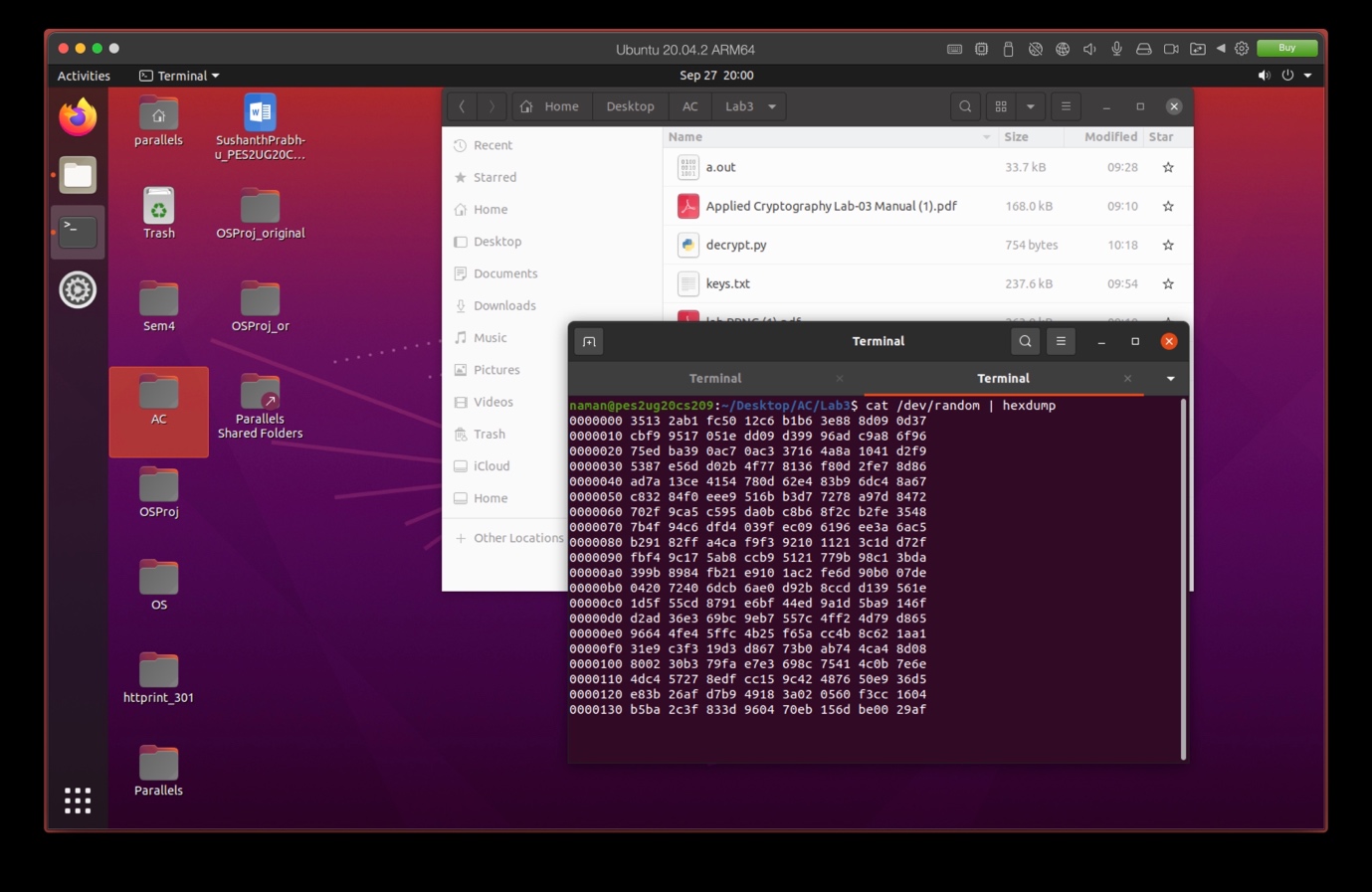
Screenshot:



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

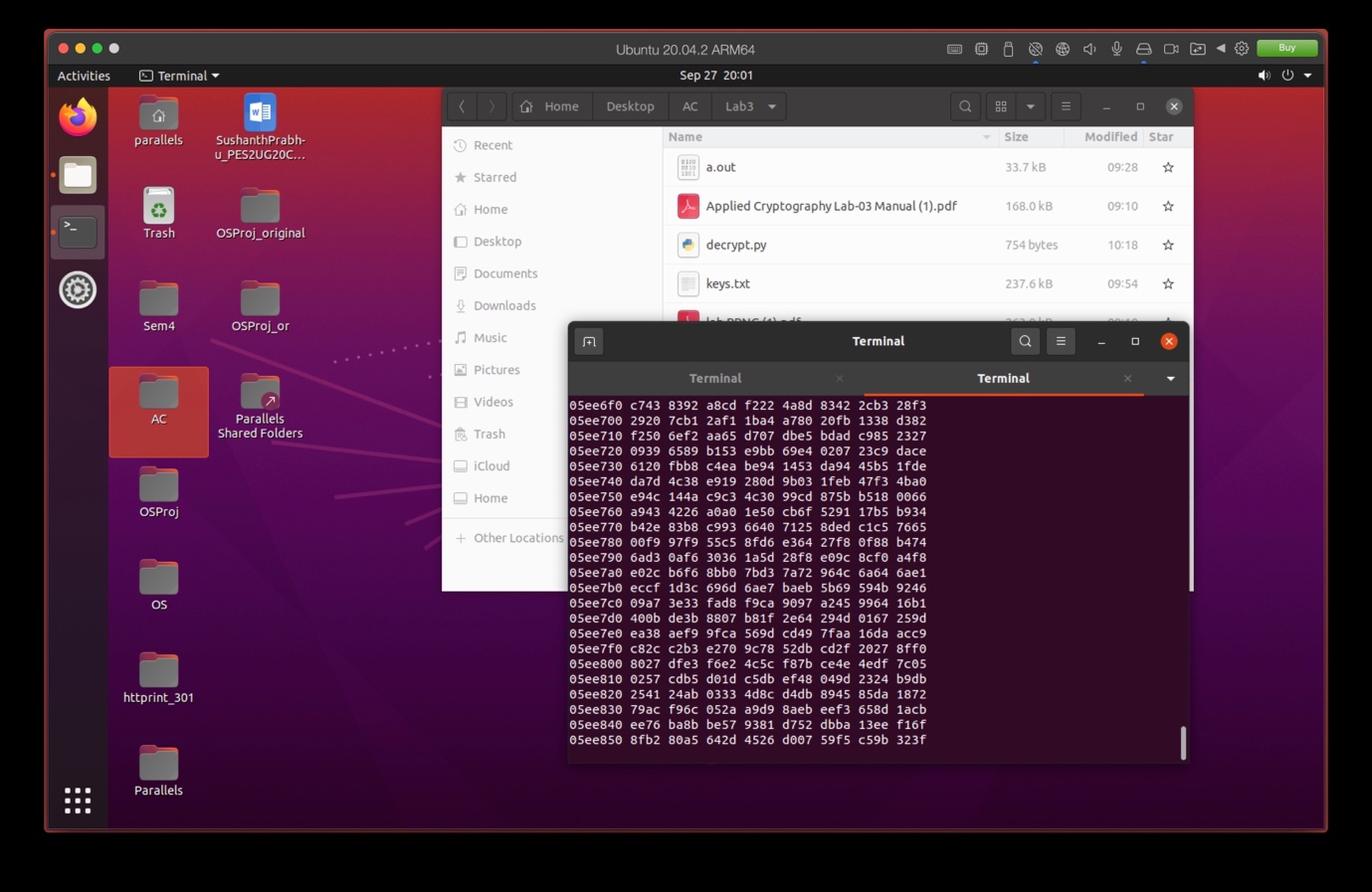
Screenshot:



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

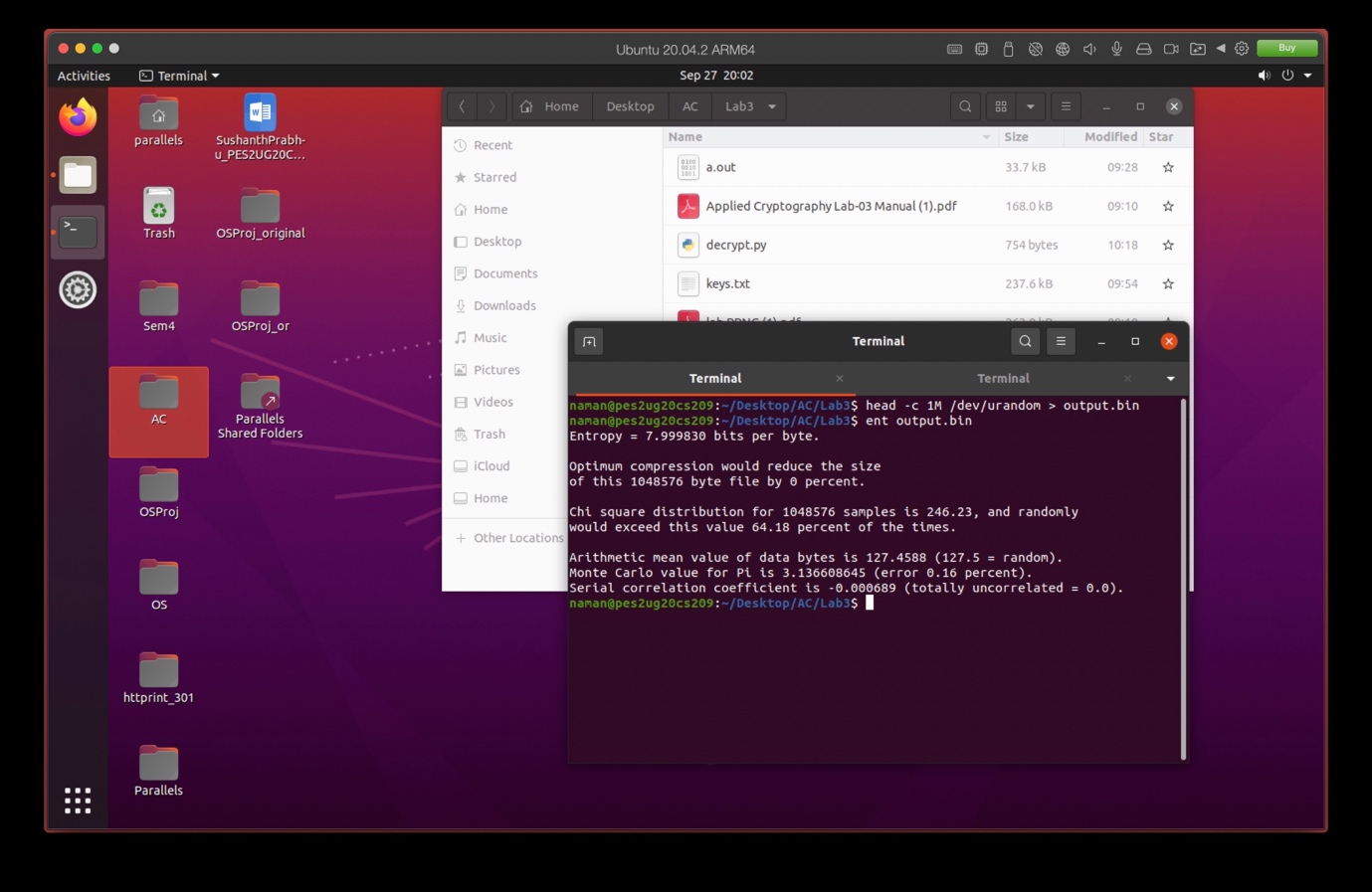
Task 5: Get Pseudo Random Numbers from /dev/urandom

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:

