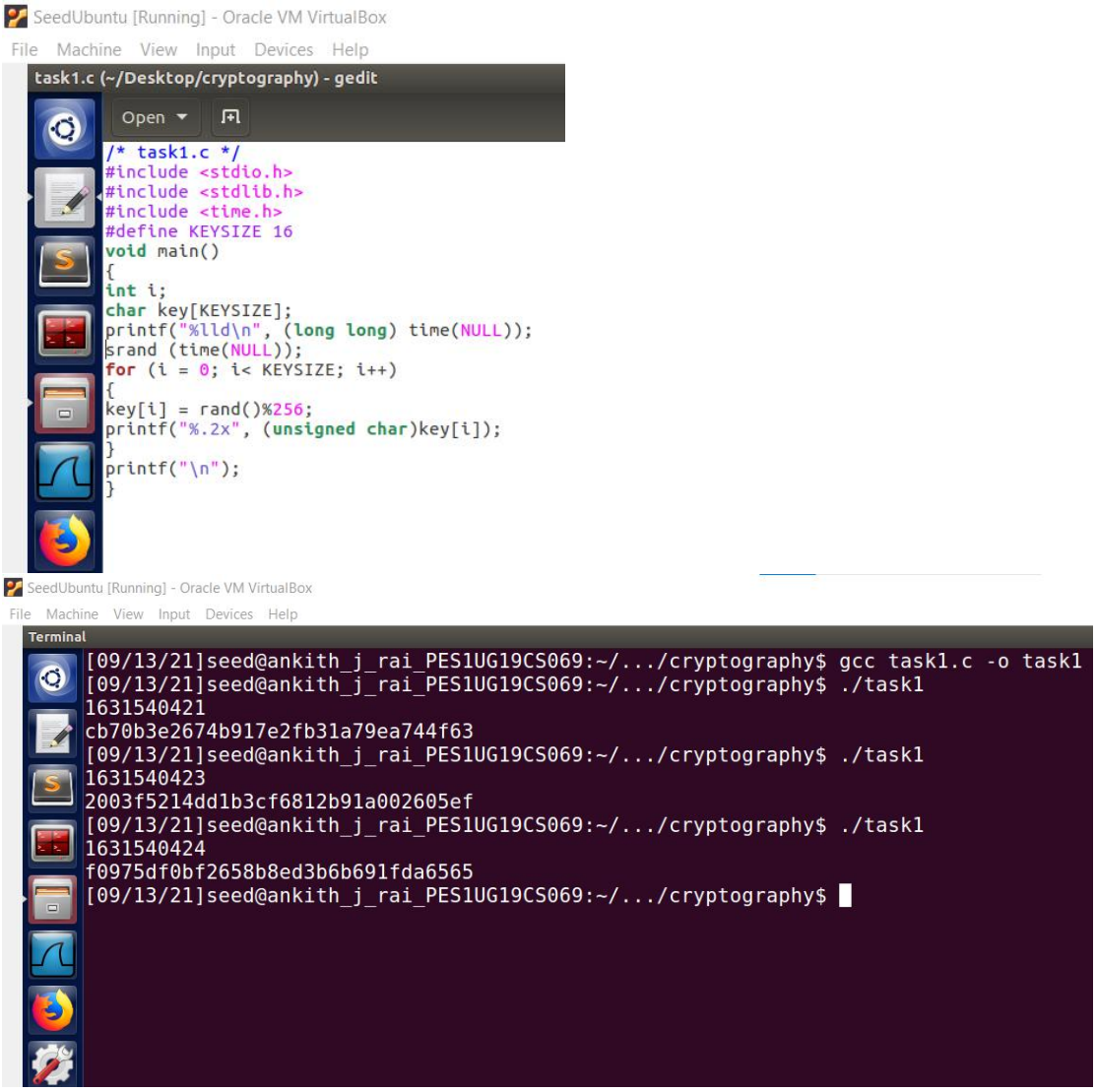
CRYPTOGRAPHY LAB- WEEK 1
Pseudo Random Number Generation

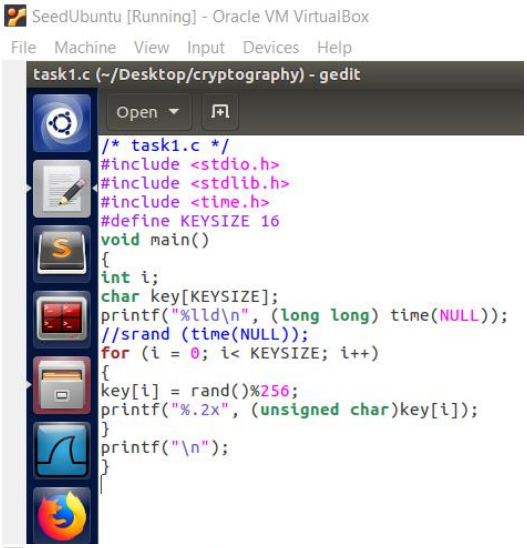
NAME : Ankith J Rai
SRN : PES1UG19CS069
SEC : B

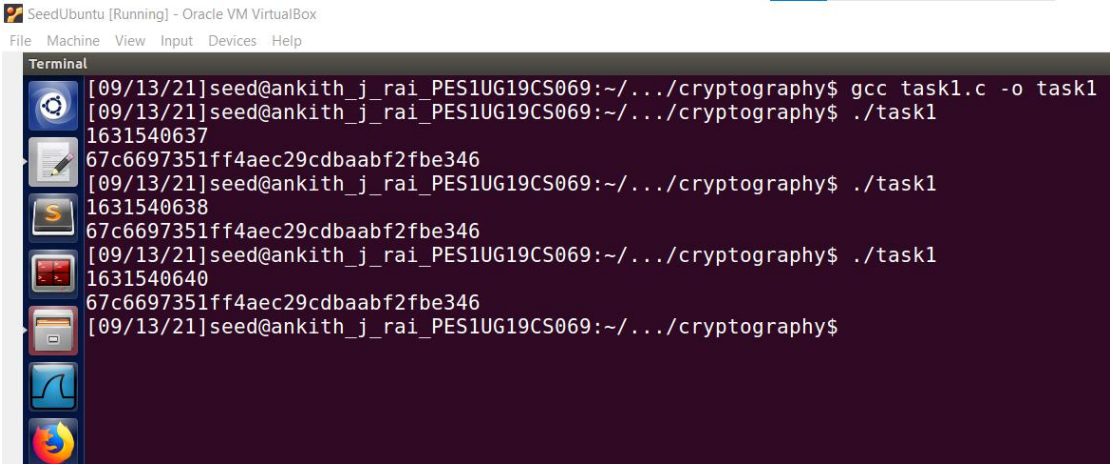
Task 1: Generate Encryption Key in a Wrong Way

Step 1: In this step a 128 bit encryption key is generated using the computer time as parameter.Hence each time the program is run a different key is generated.



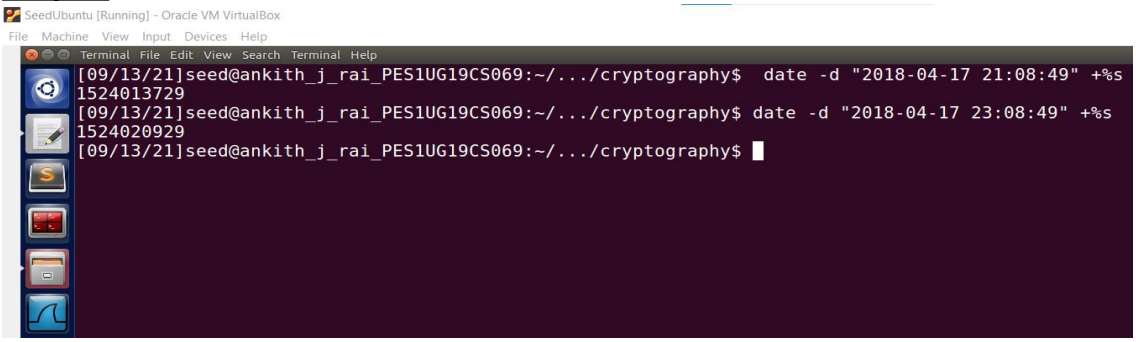
Step 2: Here also a 128 bit encryption key is generated but the same key is generated even after running the program multiple times this is because the srand(time(NULL)) has been commented out.





Task 2: Guessing the Key

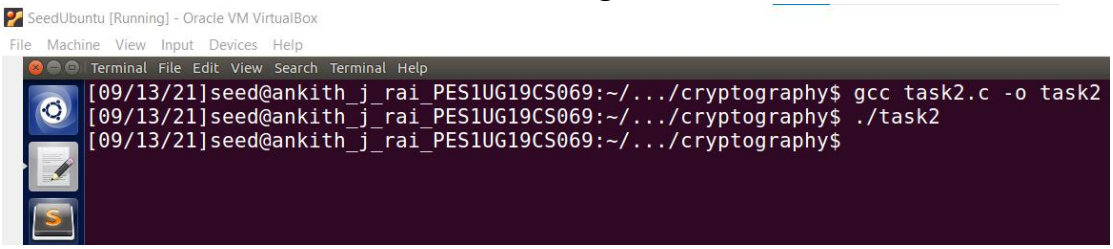
Step 1:



Here we calculate the time in seconds till the specified date.Hence we get **Value1** as **1524013729** and **Value2** as **1524020929**.

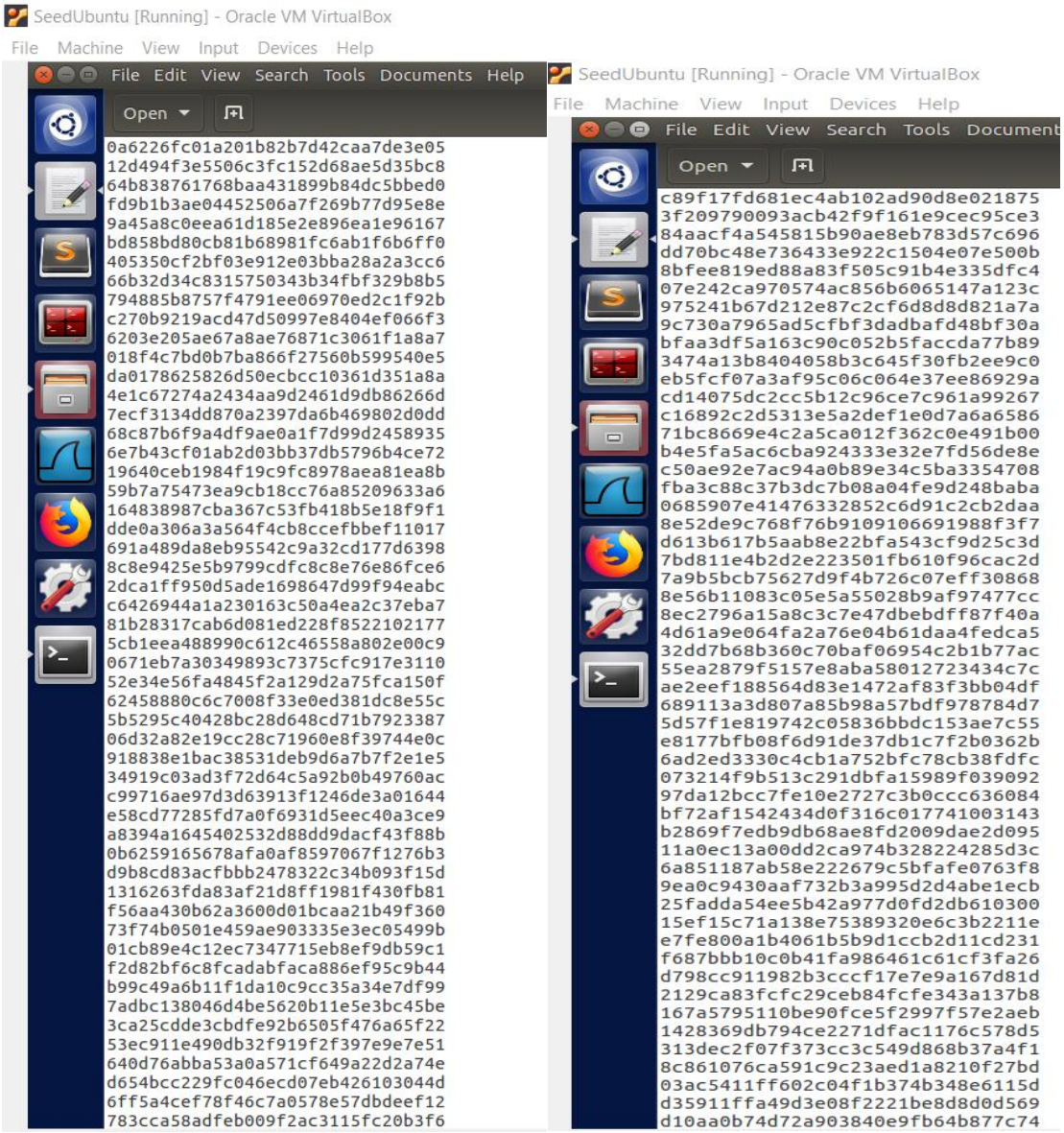
Step 2:

Screenshot of the terminal after running task2.c

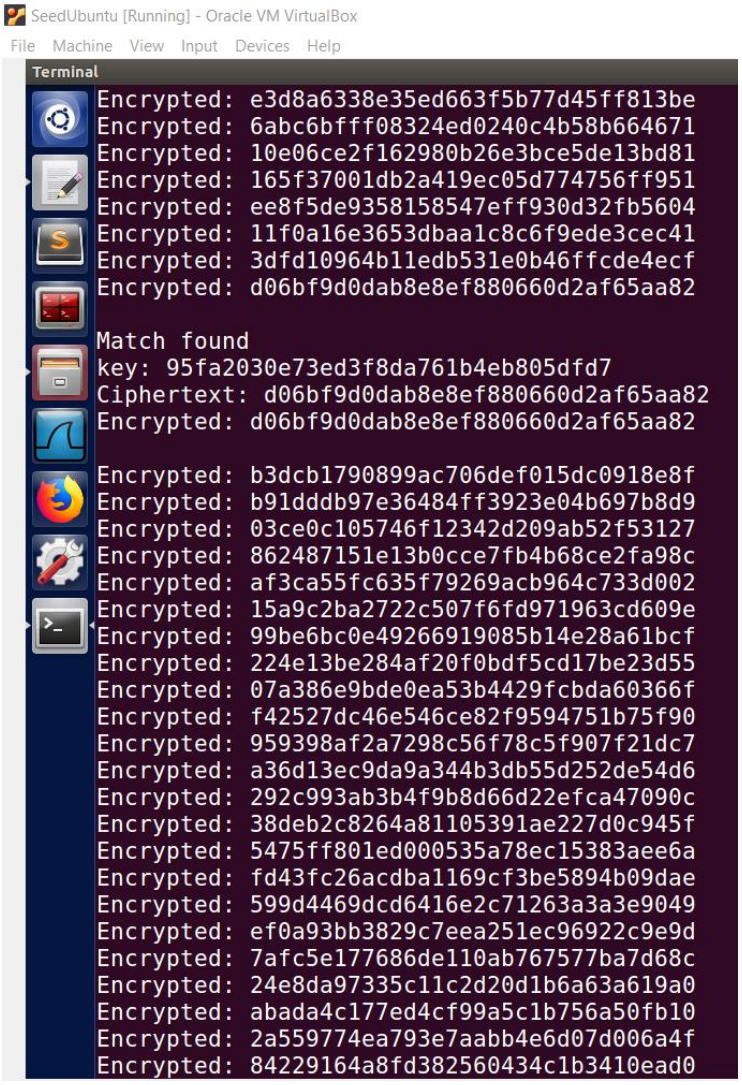


Once we run task2.c all the keys that are possible using the range of time values between value1 and value2 is generated and stored in a file called keys.txt .

Some of the screenshot of the keys.txt file which contains all possible keys.

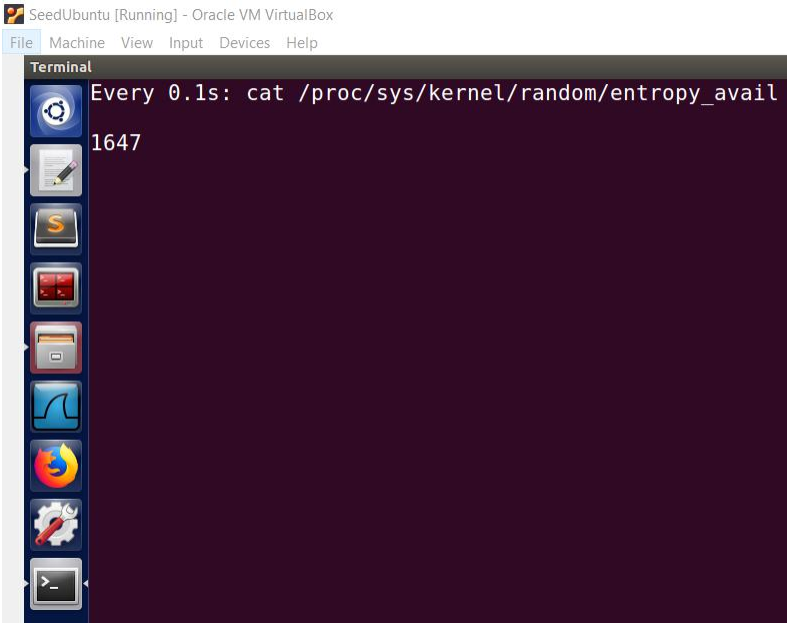


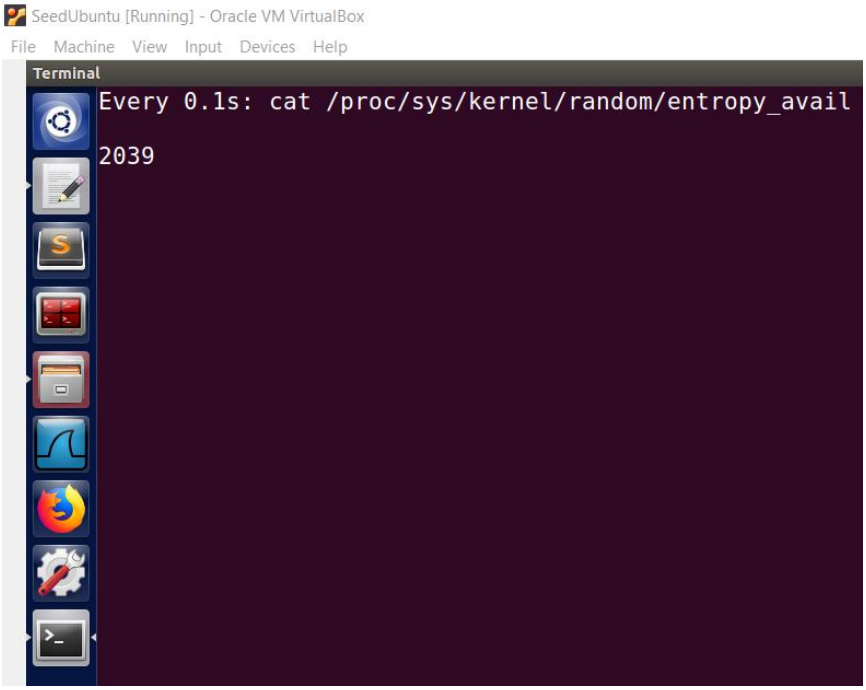
Step 3: Here we use brute force method to guess the key. For this we run the decrypt.c to compare the cipherText and the encryption of the plain text using the keys present in the keys.txt . When a match is found we can get to know the key used in encryption.



Task 3: Measure the Entropy of Kernel

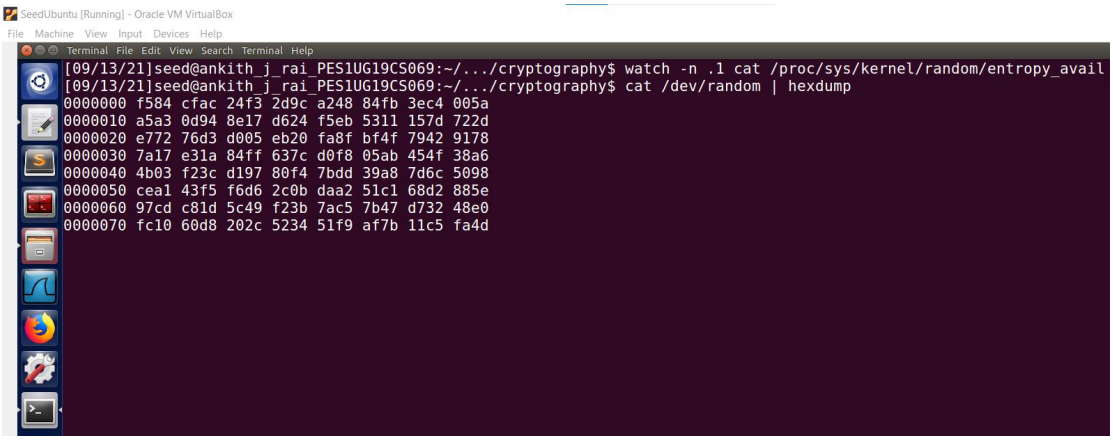
In this task we find the entropy of the kernel which is the number of bits of random numbers the system currently has.

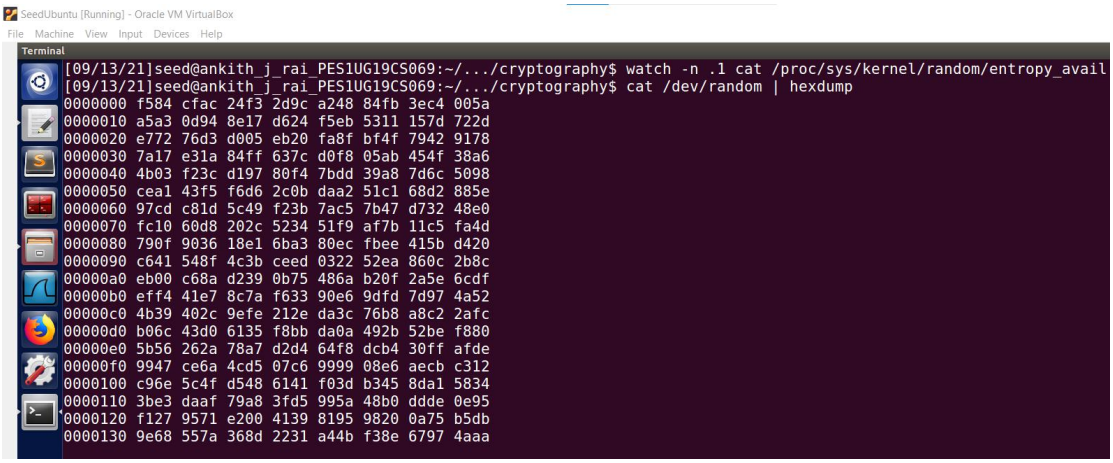




By moving the mouse and pressing keys it is noted that the entropy of the kernel has been increased.

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

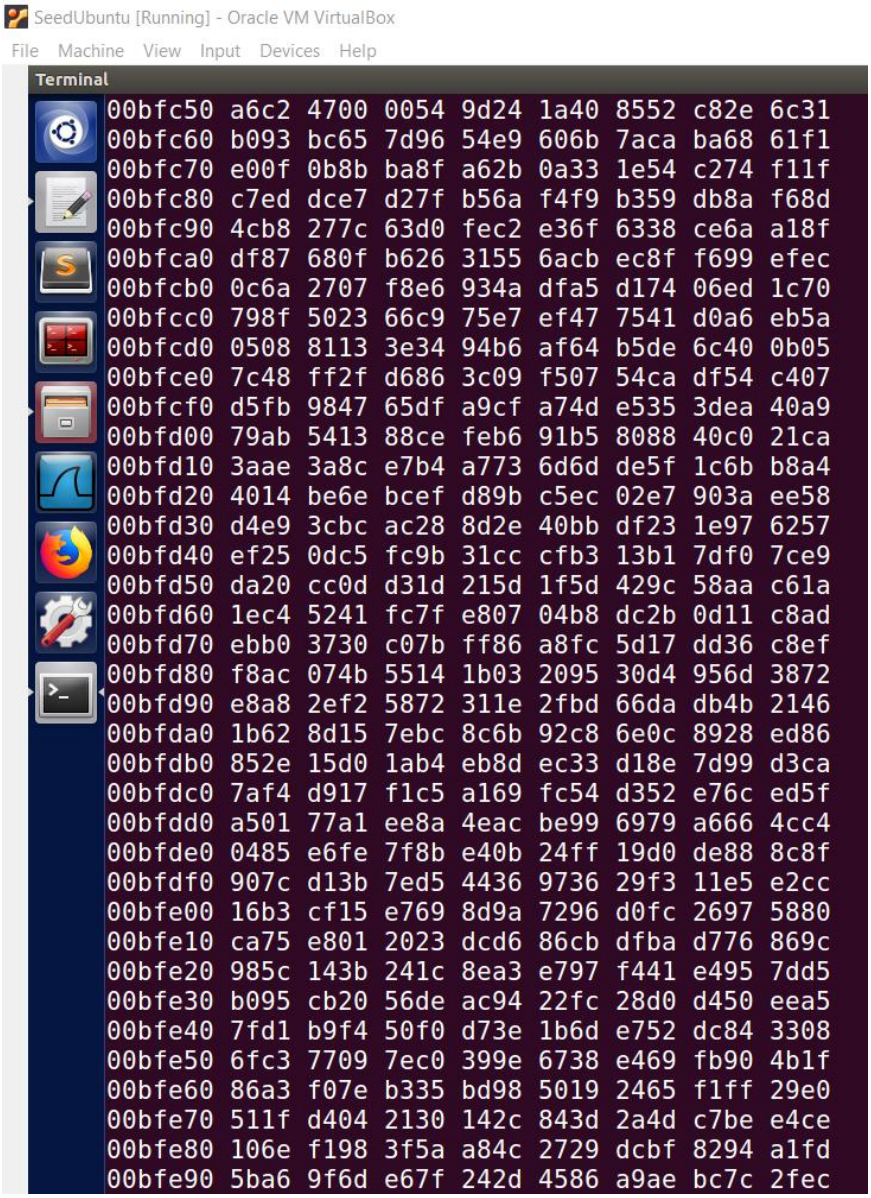




It is noted that if we do not move mouse or press any key it is noted that random numbers are not generated as entropy has become 0.
It is also noted that if we move the mouse or press any key the entropy increases hence random numbers are generated.

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

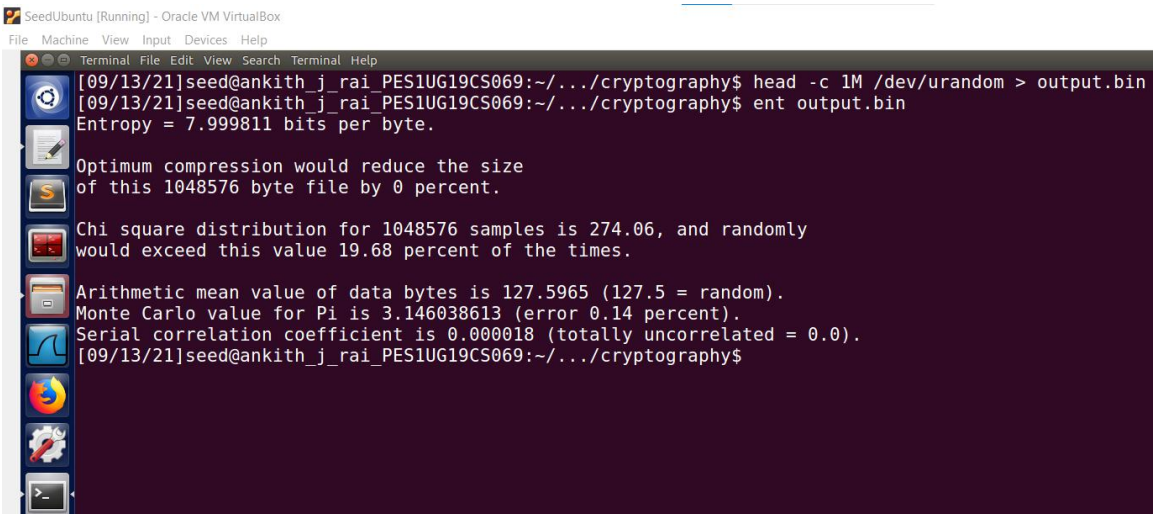
Step 1:



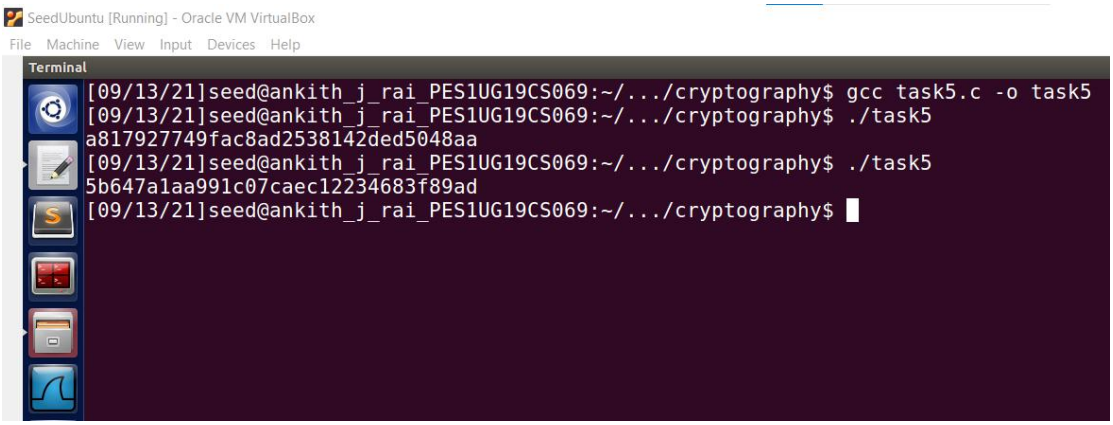
From the above screenshot we can notice that when we run /dev/urandom a lot of random numbers are generated.

Step 2: Measuring the quality of the random number

ent tool is used to measure the quality of the random number generated.

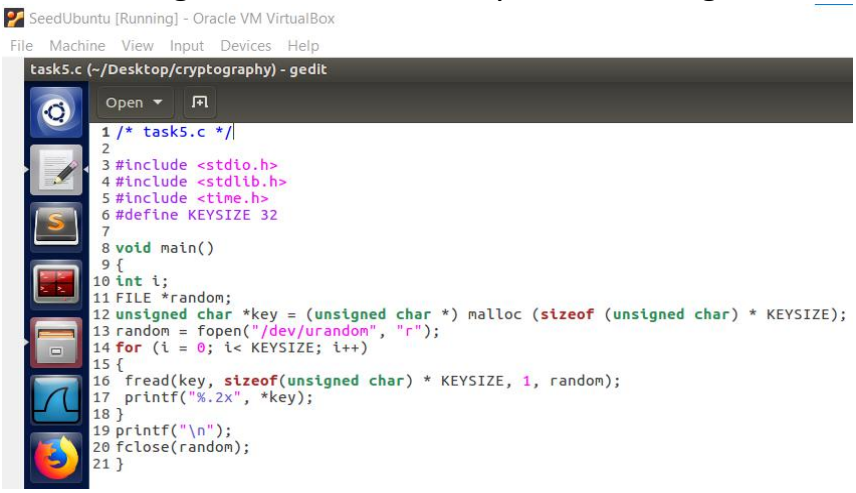


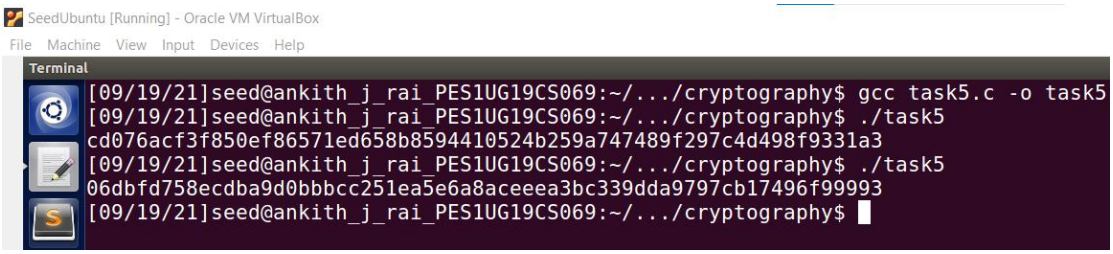
Step 3:



From the above screenshot it can be seen that a 128 bit key is generated.

In order to generate a 256 bit key the following modification is done:





From the above screenshot we can be seen that a 256 bit key is generated.