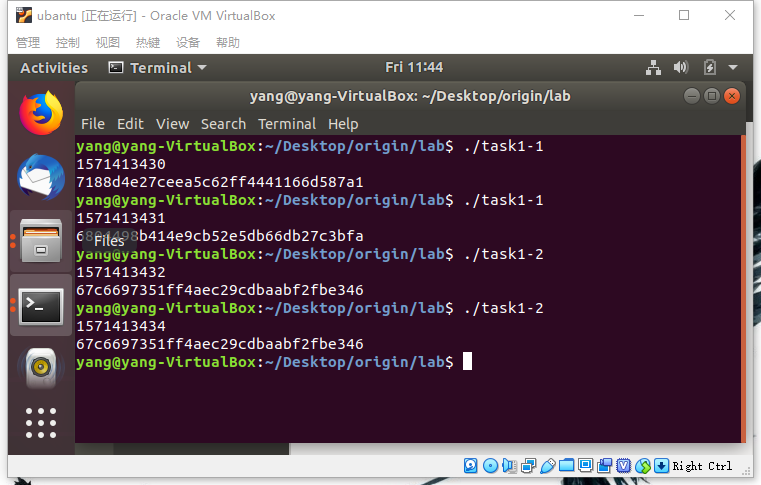
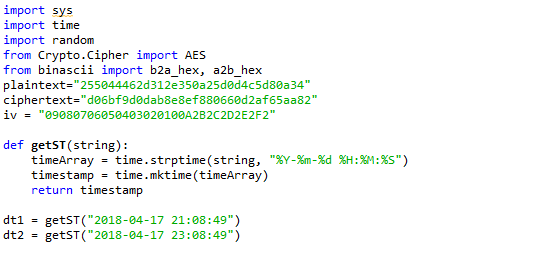
**1 Task1: Generate Encryption Key in a Wrong Way**

The task1-1 is the original code’s output, and the task1-2 is the code that command out the “srand()” line. I found that in task1-1 the key will change with the timestamp but in the task1-2 the key will be the same no matter whether the time change. “time(NULL)” is the function to give a timestamp, and the “srand()” use the current timestamp as seed to generate random number. If command out the line, the seed will be the same, which means the random number generate will output that same sequence of number.



**2 Task2: Guessing the Key**

The following is my Python code to break the key. The code use the timestamp as seed to generate key, and then traverse all key to encrypt the plaintext message. Comparing the output and the ciphertext, if they are the same, it means the key generator get the correct key that Alice used. As shown below, the key is 2129322710651590.





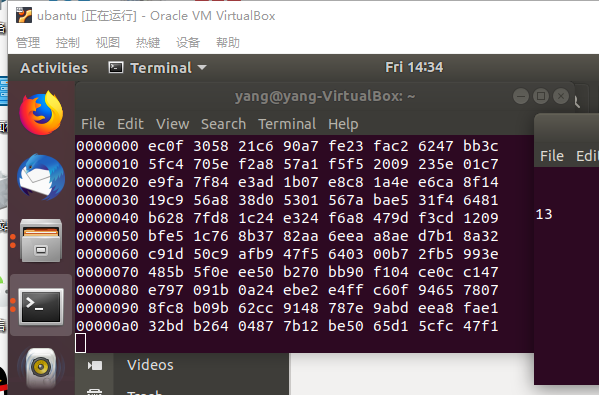


**2.3 Task3: Measure the Entropy of Kernel**

When moving the mouse, the number increased by 2 or 3 each time. The click action will make number increase by 3 or 4. The typing will make number increase 5. The reading would not make entropy increase unless using mouse or keyboard to turn page. And visiting a website will make number keep going by 1 without other actions. If we only consider the step of the number, typing increased the entropy significantly, but if we consider the all action we need to do, visiting a website will be the champion.

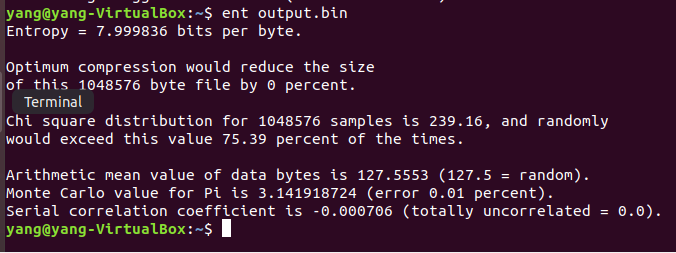
**2.4 Task4: Get Pseudo Random Numbers from /dev/random**

If we do not do any action, the number will only increase 1 each time, and if we randomly move mouse the number will increase much faster. No matter how faster the number increases, it will go back to 1 once it reach 64, with the increase in first 6 bit of output of hexdump. The output of hexdump is hex. The reason that number increase faster is that the increase is the sum of default number and the number triggered by actions.



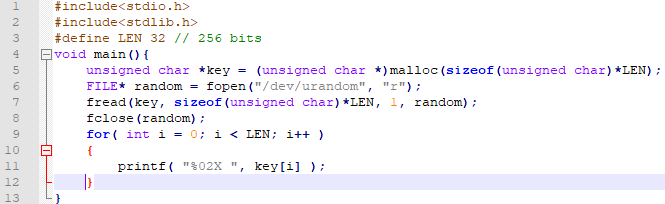
**2.5 Task5: Get Random Numbers from /dev/urandom**

In this case, the number generate very fast, and it seems that there is no affect by the moving the mouse on the outcome. The following is the outcome of random number analysis.



It shows that the quality of random number is good, because the number it generate is close to the truly random number.

The following is the code to generate a 256-bit encryption key.



This screenshot shows the key that generate by the code.

