1. Created directory hw3.
2. Implemented a program (generate\_keys) that will randomly generate 3072-bit RSA Key and will save public key in pub\_key and Private Key in priv\_key under the folder hw3 created in Step #1.

**Q0**: I used Python programming language. I used RSA library.

**Q1**: Using hexdump displayed pub\_key and priv\_key  
**pub\_key**:

Text

Description automatically generated

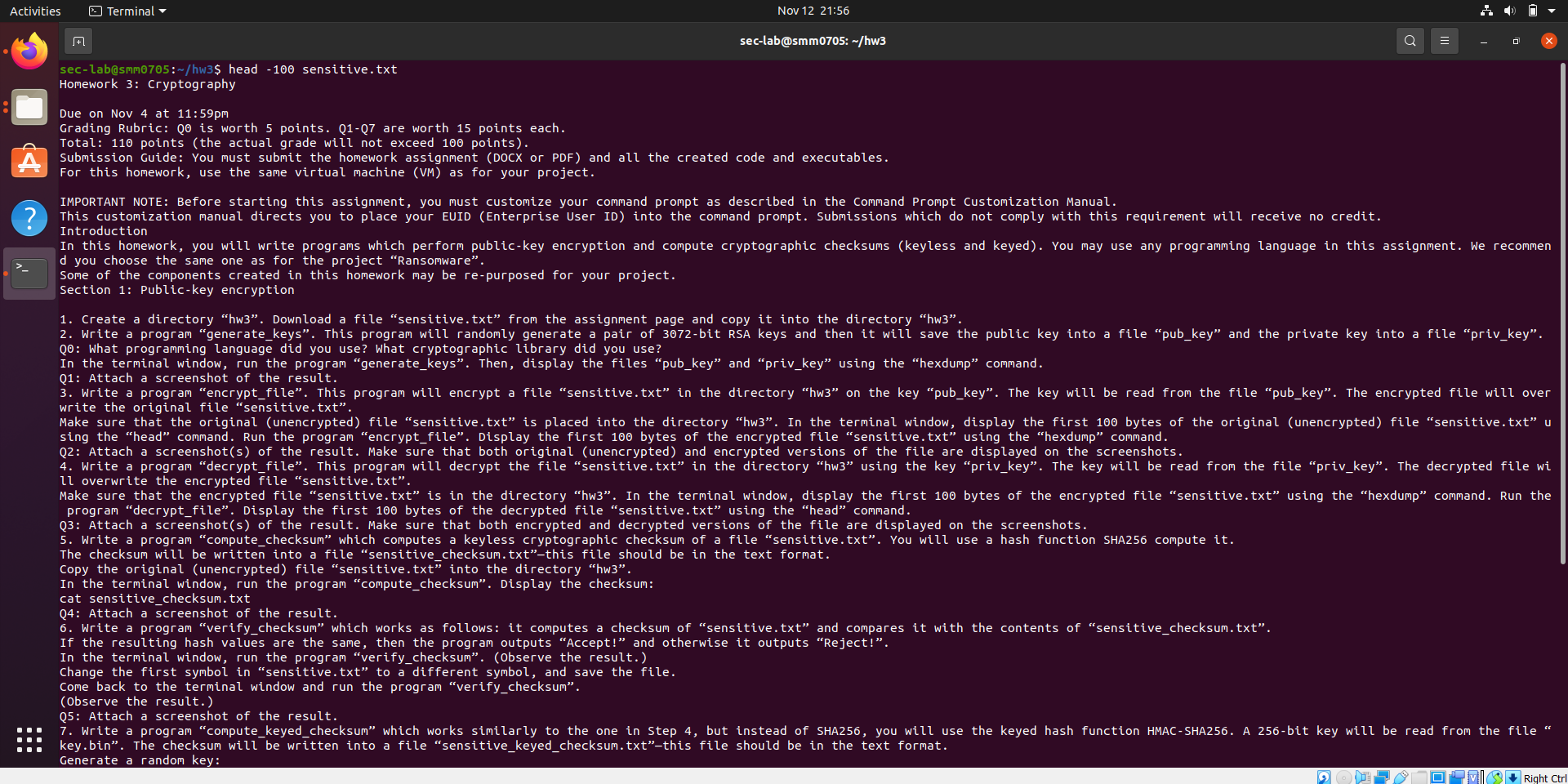
**Priv\_key**:

A screenshot of a computer

Description automatically generated with medium confidence

1. Implemented the program encrypt\_file that encrypts the given file sensitive.txt with the help of key Public Key available in pub\_key

**Q2**: Screenshots of encrypted and unencrypted (original files):  
  
***Original file***:



***Encrypted file***:

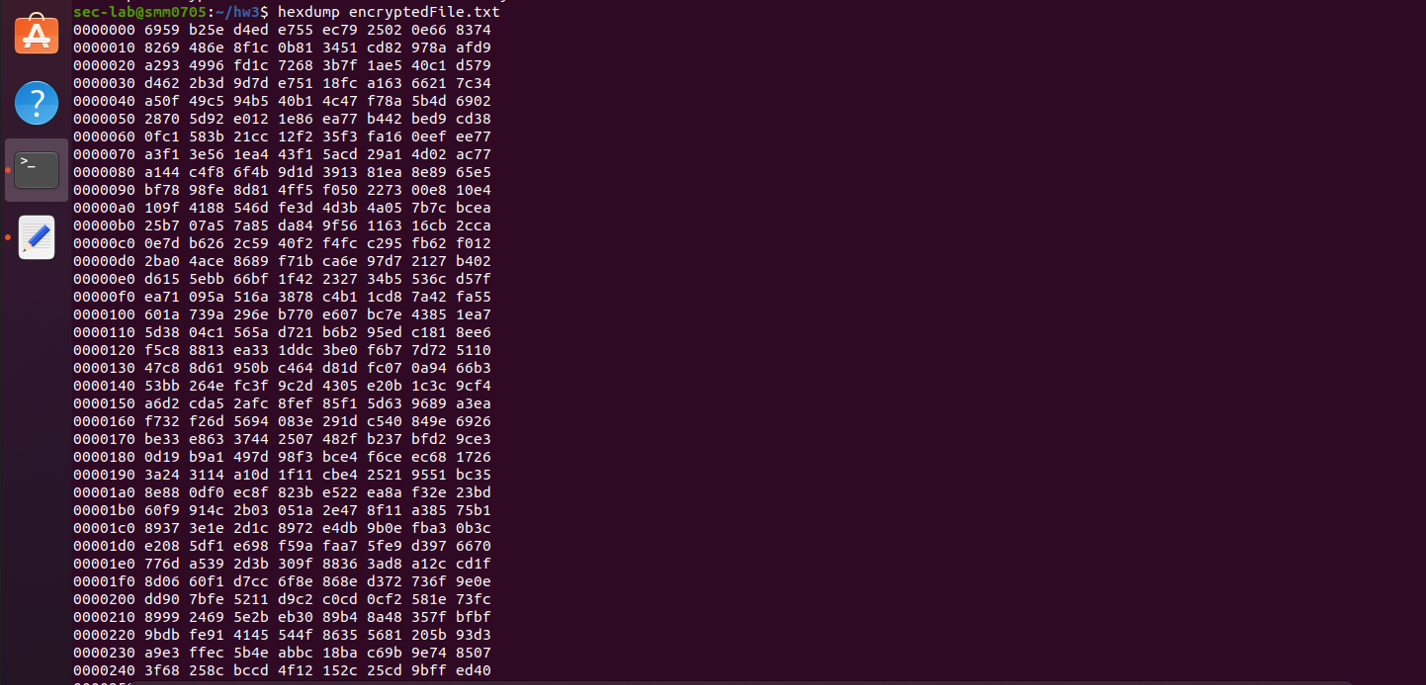
A screenshot of a computer

Description automatically generated with medium confidence

1. Implemented the program decrypt\_file python program that reads the priv\_key and decrypts the encrypted file

**Q3**: Executed the decrypt\_file program on the terminal.

*Encrypted data:*



Result post execution of decrypt\_file program:

A screenshot of a computer

Description automatically generated with medium confidence

1. Implemented the program compute\_checksum using python programming language (used hash function SHA256 as instructed) and the check sum is written to sensitive\_checksum.txt file.

**Q4**: Result of the check sum post compute\_checksum execution on the terminal.

Text

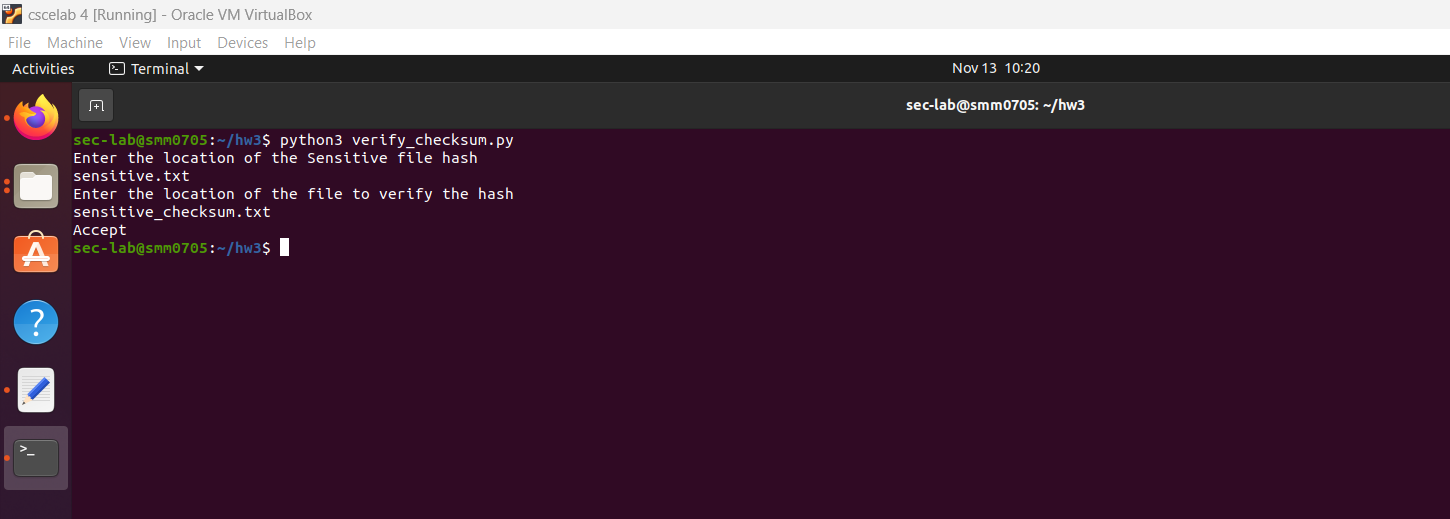
Description automatically generated

1. Implemented the python program verify check sum as per the given instructions.

**Q5**: Comparing the sensitive\_checksum (obtained from step #5 and Q4) with check\_sum of the sensitive.txt file.

*Comparison with original checksum on sensitive.txt file*:  
  
A picture containing timeline

Description automatically generated



*Comparison on altered sensitive.txt file and sensitive\_checksum.txt file:*

Text

Description automatically generated

1. Implemented the compute\_keyed\_checksum as per the instructions in python using keyed hash function.

*Generate a random key:*head -c 256 /dev/urandom > key.bin

Text

Description automatically generated

**Q6**: The result of compute\_keyed\_checksum post execution using cat command:

Text

Description automatically generated

1. Implemented verify\_keyed\_checksum in python using HMAC-SHA256.

**Q7**: Comparing original file and file generated from #Q6 sensitive\_keyed\_checksum and got ‘Accept’ result.

**Original file**:

A picture containing timeline

Description automatically generated

*Verifying keyed check sum of original file with compute keyed checksum of sensitive file:*

Graphical user interface, text

Description automatically generated

Changed sensitive.txt file computed keyed check sum and then compared it with sensitive\_keyed\_checksum.txt file:

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated