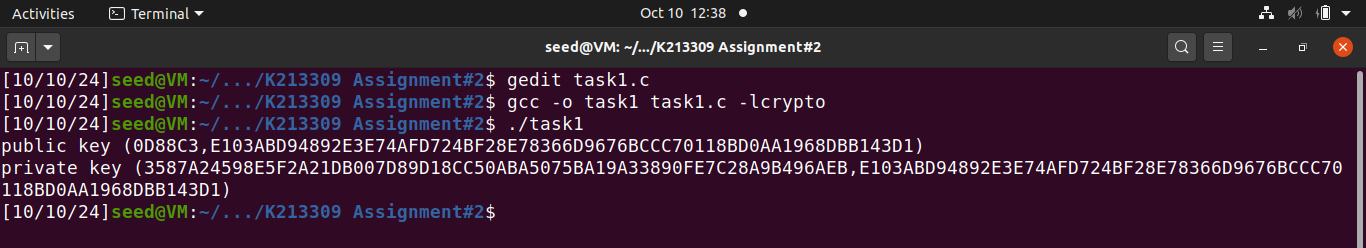
# Task 1 - Deriving the Private Key

Let p, q, and e be three prime numbers. Let n = p\*q. We will use (e, n) as the public key. Please

calculate the private key d. So, we will first have to create BIGNUM variables for p, q, e, d and other temp variables. The code is also shared on GCR.

After running the C code, this is the output.



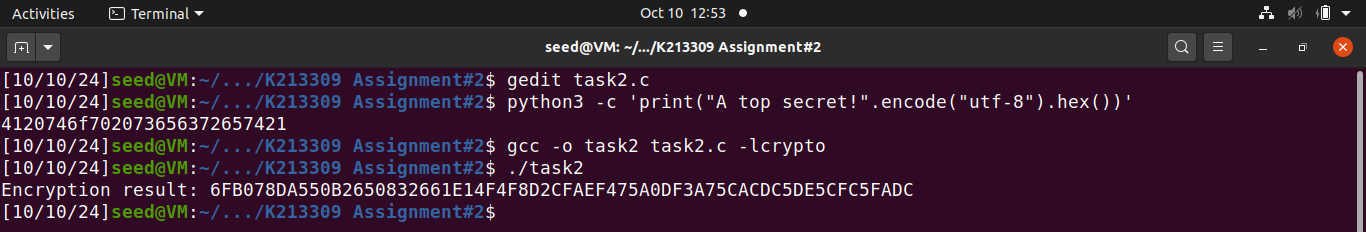
Therefore, the value of d is 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB.

# Task 2 - Encrypting a Message

Firstly, we will use the python commands given in the manual to convert a plain text to a hex string.

$ python3 -c ’print("A top secret!".encode("utf-8").hex())’

We then write a C code to convert the hex string to BIGNUM and execute the code. The code is shared on GCR.

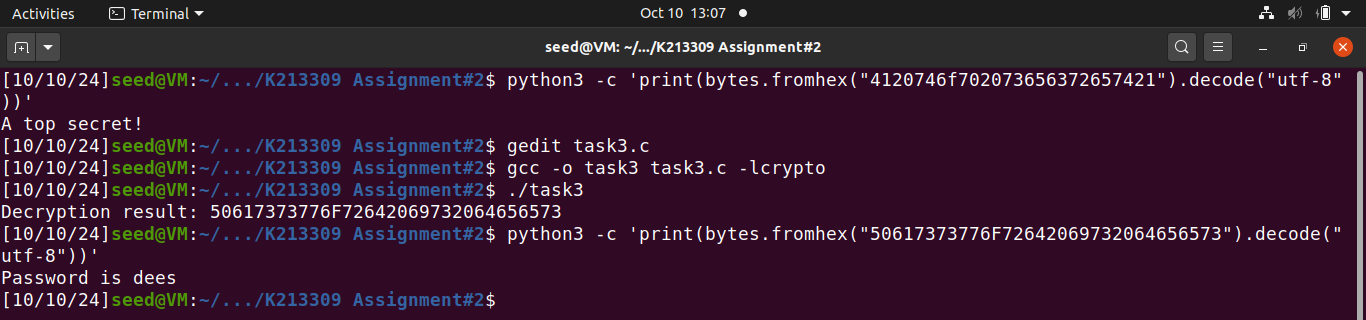


# Task 3 - Decrypting a Message

Firstly, we will use the python command given in the manual.

$ python3 -c ’print(bytes.fromhex("4120746f702073656372657421").decode("utf-8"))’

We then write a C code to decrypt the cipher text and convert it back to ASCII string and execute the code. Then we use the python decode command to see the actual message. The code is shared on GCR.

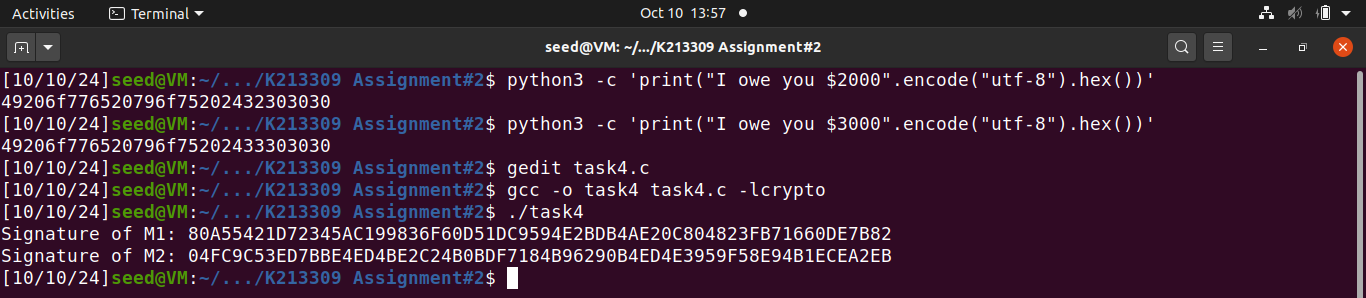


# Task 4 - Signing a Message

In this task we will generate signatures for messages. The two messages will be:

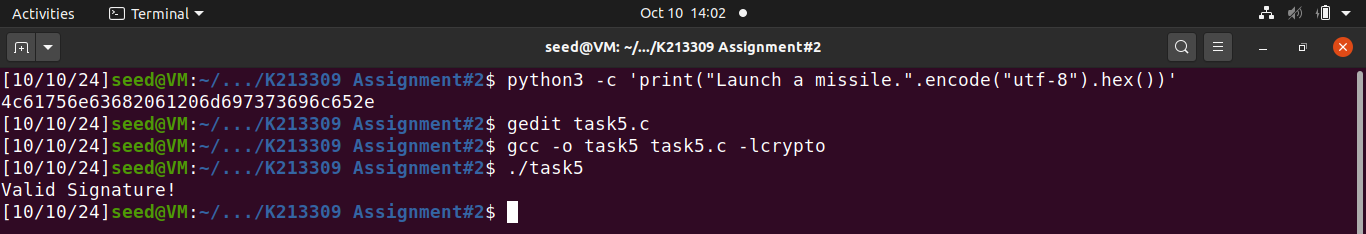
* I owe you $2000.
* I owe you $3000.

The code is shared on GCR.



# Task 5 - Verifying a Signature

In this task we will verify whether the signature belongs to Alice or not. The code is shared on GCR.



If a valid S happens to be corrupted, even just in one byte, our program will also reject the message.

# Task 6 - Manually Verifying an X.509 Certificate

Here are all the commands that were specified in the manual. Other relevant codes are submitted on GCR.

