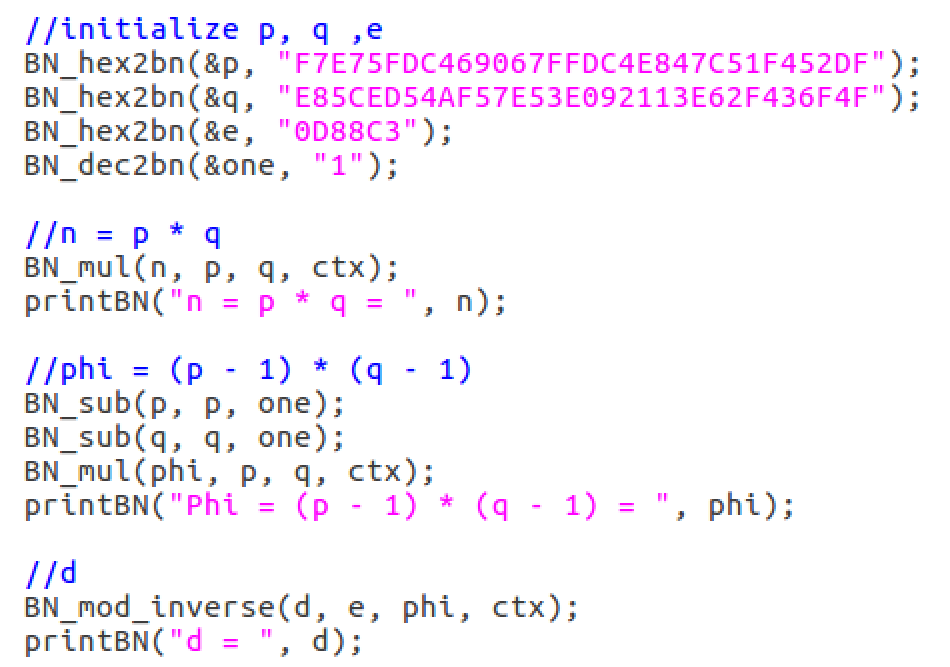
Mark Gameng

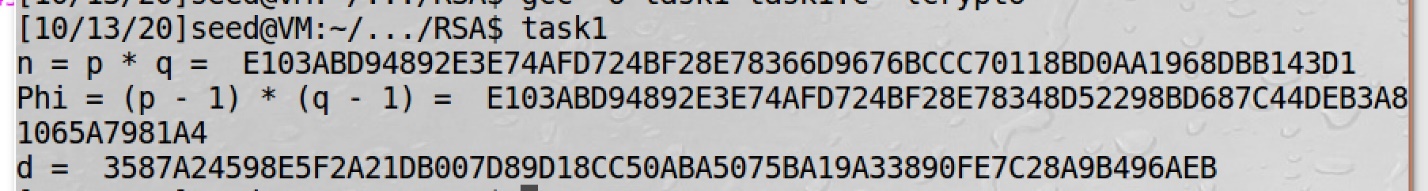
CS 458 – Dong Jin

RSA

## 3.1 – Task 1: Deriving the private key

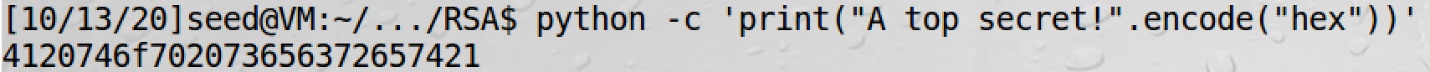


Given **p**, **q**, and **e**, I made a similar program in the lab description and found **n** which is **p** \* **q**. I then calculated the private key **d** by first calculating **phi**.

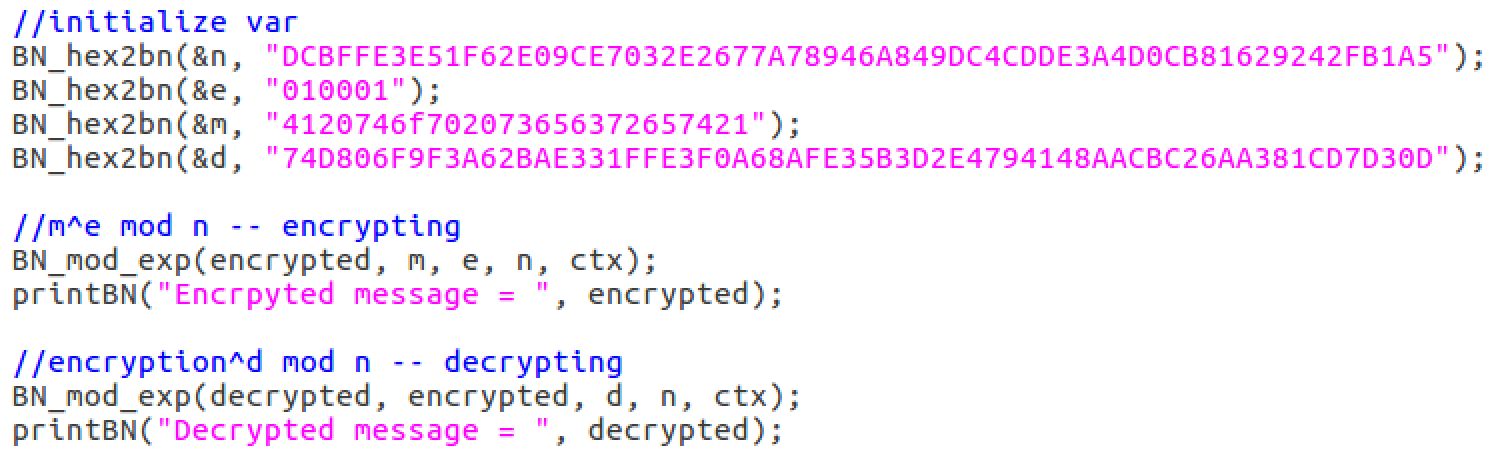
  
It was very easy to create the private key. I wonder if there is an easier process than this but also as secure or even better?

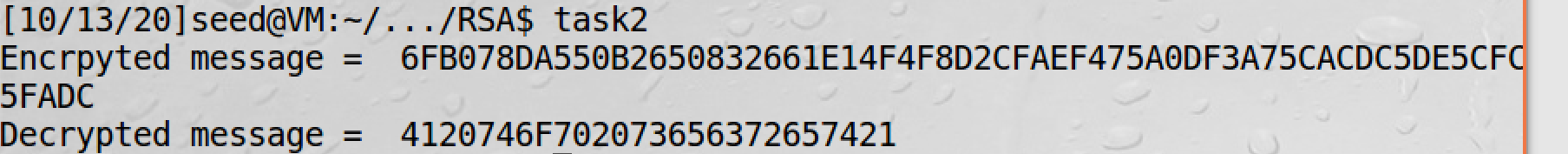
## 3.2 – Task 2: Encrypting a message

We want to encrypt “A top secret!” but first it needs to be turned in to a hexadecimal.



Public keys are given so we can encrypt it. Encrypted = message ^ e mod n. The private key was also given so we can double check by decrypting and it should give the hex of the message.

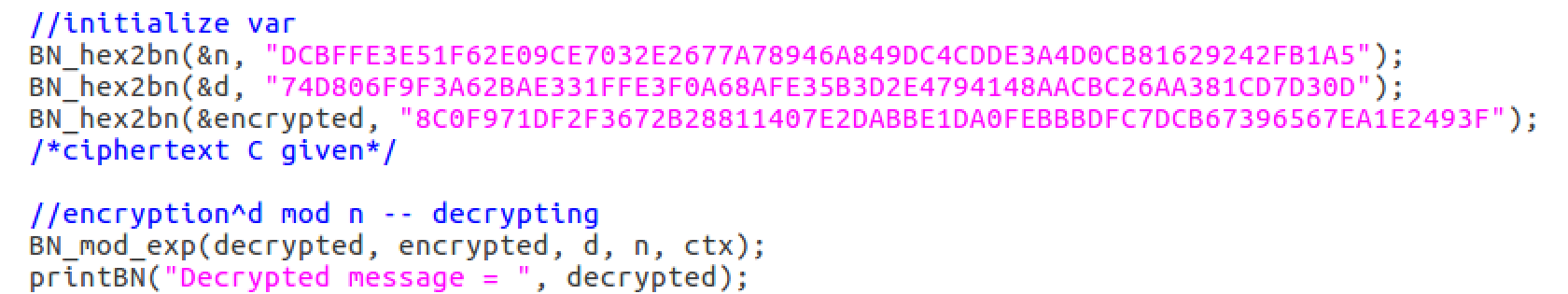


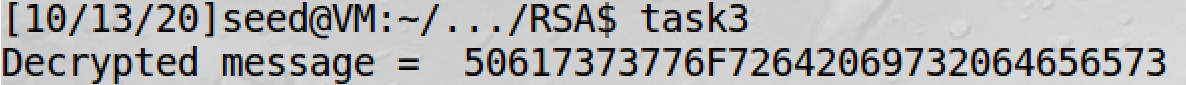


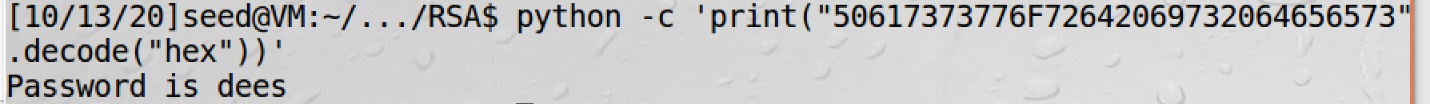
Here we see that “A top secret!” was encrypted and decrypting using the private key **d**, we get the same hex of “A top secret!”. Thus, we have successfully encrypted a message and decrypted it using a private key.

## 3.3 – Task 3: Decrypting a message

We are given ciphertext, **C,** to decode and using the same values, we can decrypt it.







Getting the decrypted hex and then converting it to ASCII, we get that the decrypted ciphertext **C**, is “Password is dees”.

## 3.4 – Task 4: Signing a message

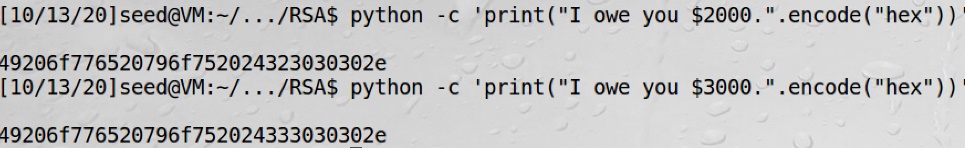
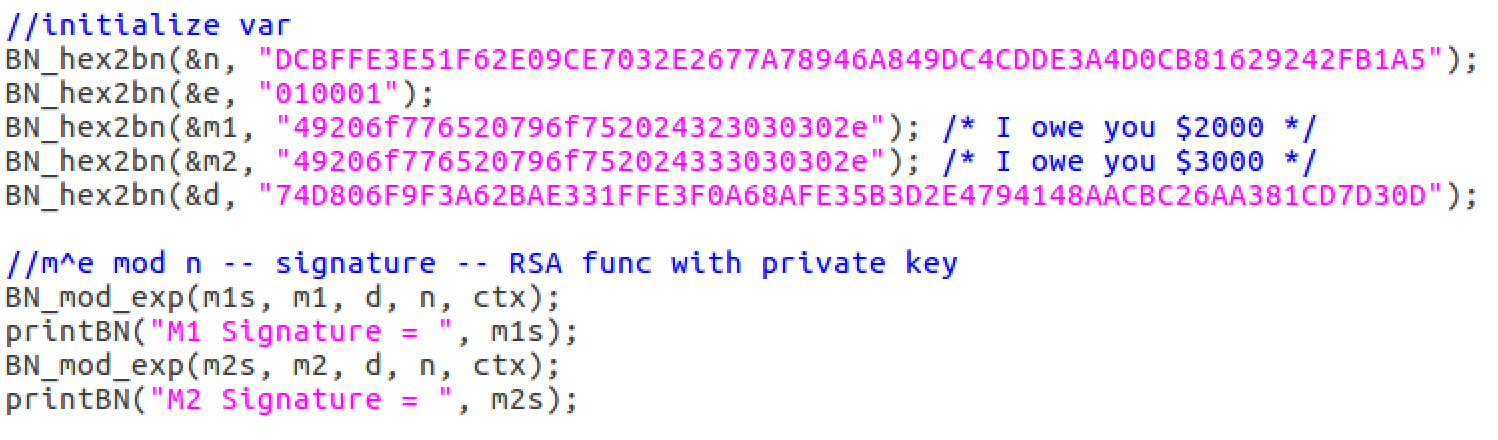


Image above shows two messages with its hex values. Those two will be used to compare signatures. Signature is just RSA function with the private key.

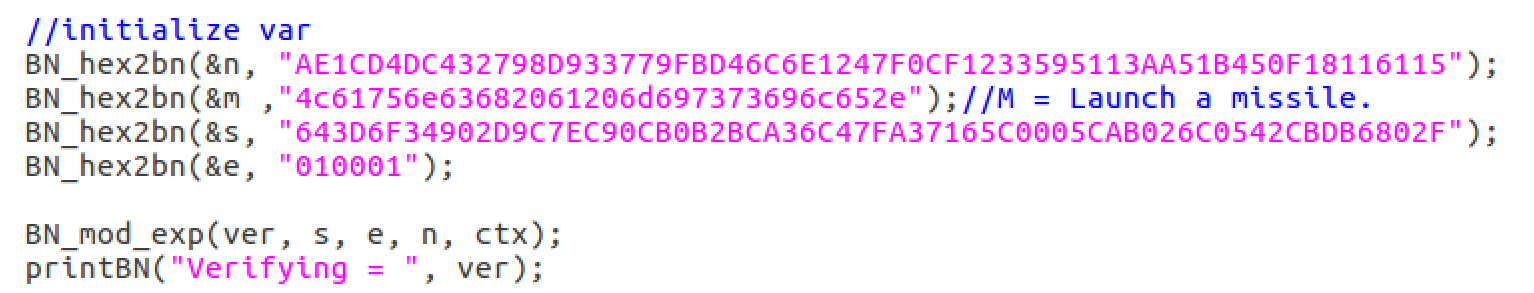


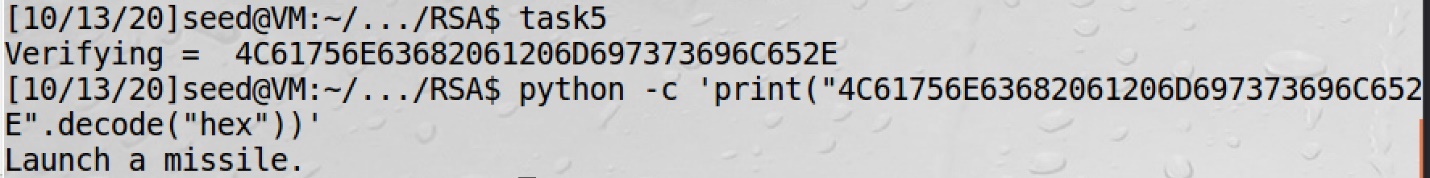


Here we see that even though we only made a slight change to the message **M**, we get very different signatures. I thought it would be quite similar because the hex of the messages are similar except for one byte. That’s shows how good the function is and use of exponent and modulus.

## 3.5 Task 5: Verifying a signature

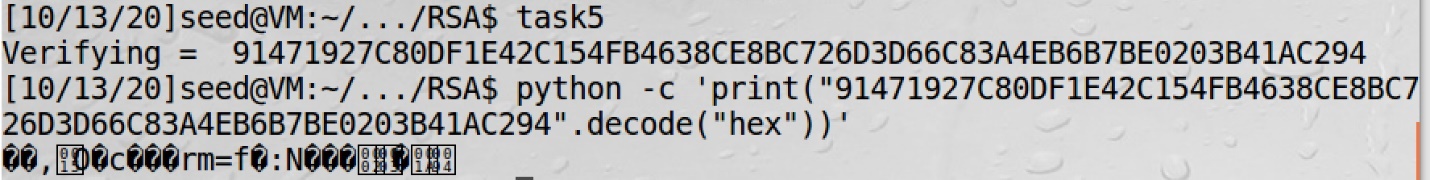
Given a message with signature and public key, we can verify the signature is legit.





To verify the signature, we do signature^e mod n, and we get the message. Cross checking the message, we see that the messages are equal, so the signature is indeed Alices.

Changing the last byte of the signature, this is what happens:



We can clearly see that the signature is not Alice’s or is corrupted because the hex is different and decoding it got something random. This is surprising because we only changed the last byte of the signature, and it completely changed it. I would have thought it would only made a small difference.

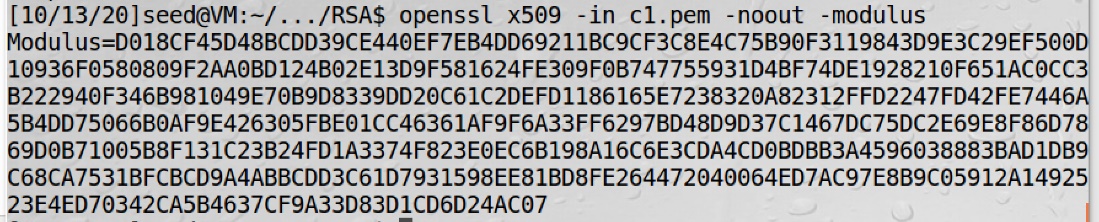
## 3.6 – Task 6

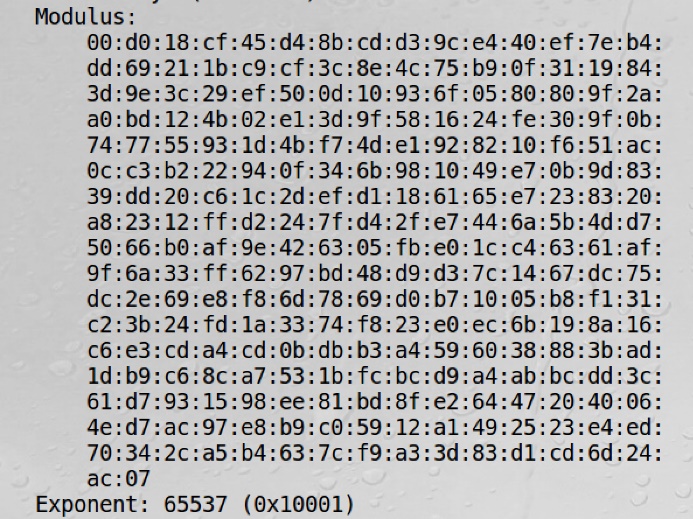
**Step 1:**

I used [www.google.com](http://www.google.com) and got the certificate and copied it to c0.pem and c1.pem.

**Step 2:**

Can extract the value of **n** and **e** using the following commands, **openssl x509 -in c1.pem -noout -modulus**, and **openssl x509 -in c1.pem -text -noout**

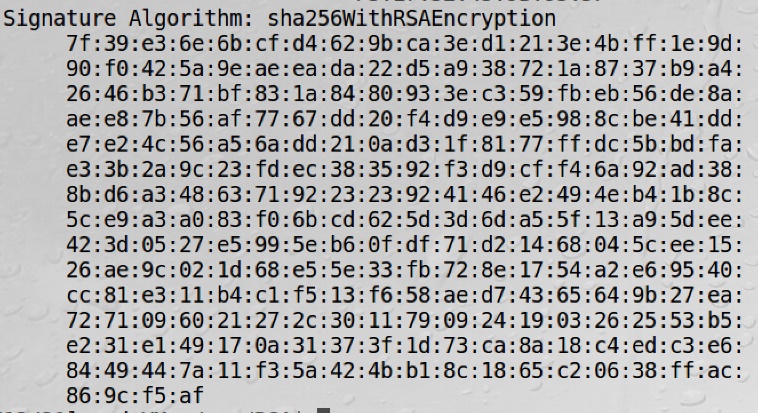




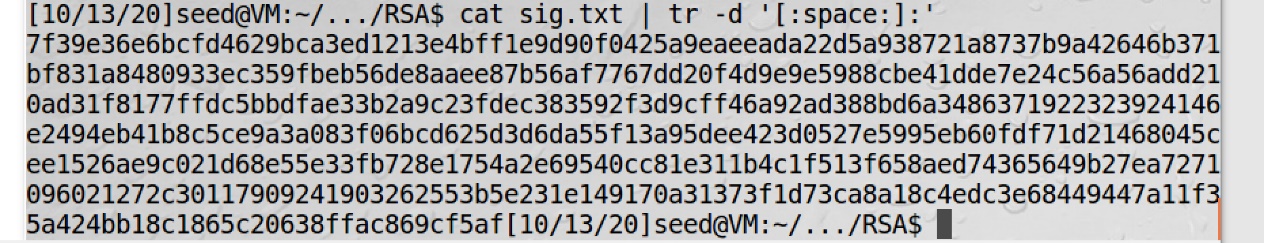
From those, we now have the **n** and **e**.

**Step 3:**

To get the signature, I did openssl x509 -in c0.pem -text -noout

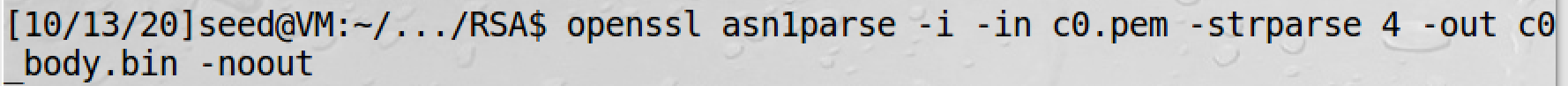


Inputting those values in a text file and removing the spaces and colons, we get the signature.

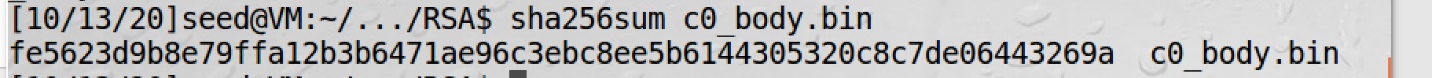


**Step 4:**

To get the body of the certificate, without the signature block, and put it in a file:



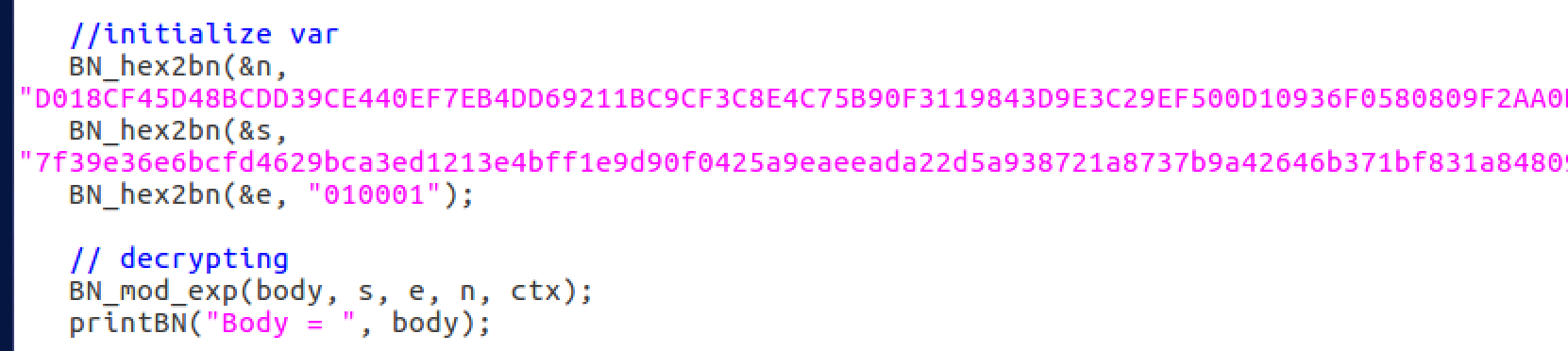
Calculating the hash of the body of the certificate:



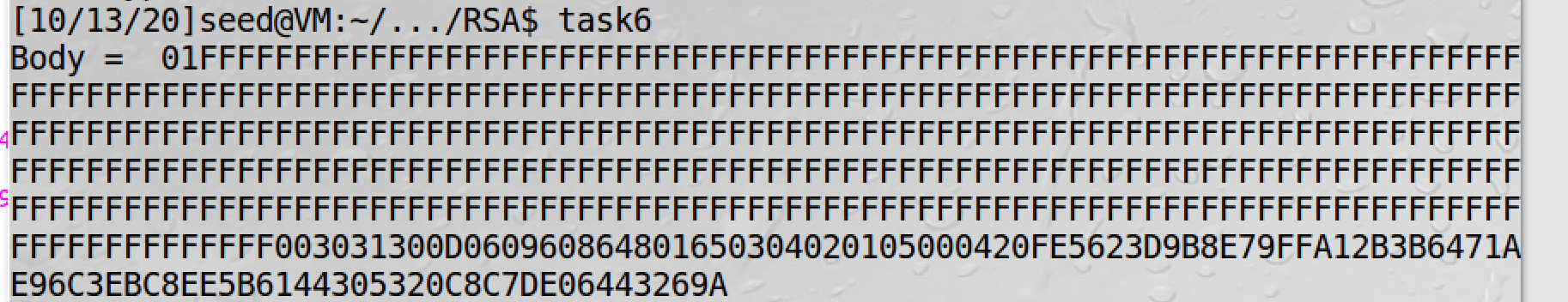
This will be used to verify the certificate, similar to task 5.

**Step 5:**

Using a similar program in task 5, but using the **n**, **e**, and **signature** above, we can verify the signature:



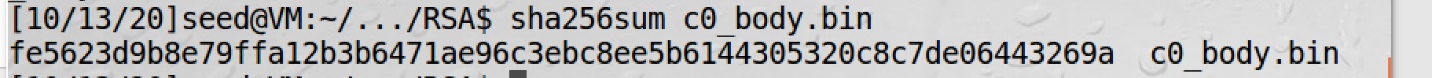
Running the program results in:





Comparing with the hash in step 5:





The hash looks different, but at the end, they are the same. Thus, we have verified that the signature is valid. I wonder why we get all those irrelevant? hex values in front.



## Final Thoughts

This was very mind opening and quite surprising at certain points. In task 5, I knew changing just one byte would make a difference, but never expected it to be that big of a difference. It went from being a legible message to being gibberish due to one byte being off in the signature. It just goes to show how secure RSA. RSA is secure and easy to implement, however I wonder in a security perspective, would it better to have a cipher that is as secure but harder to implement?