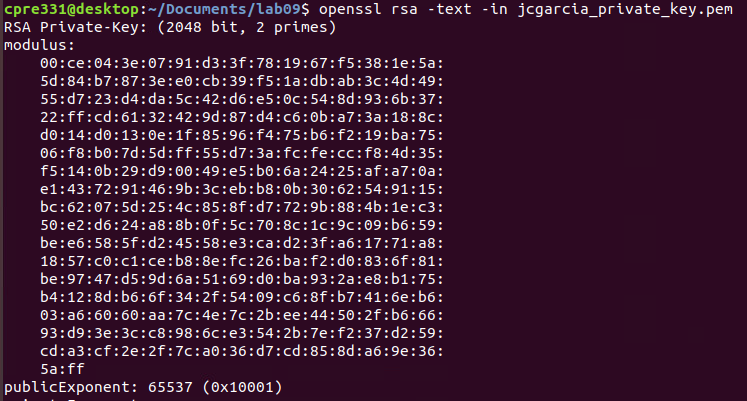
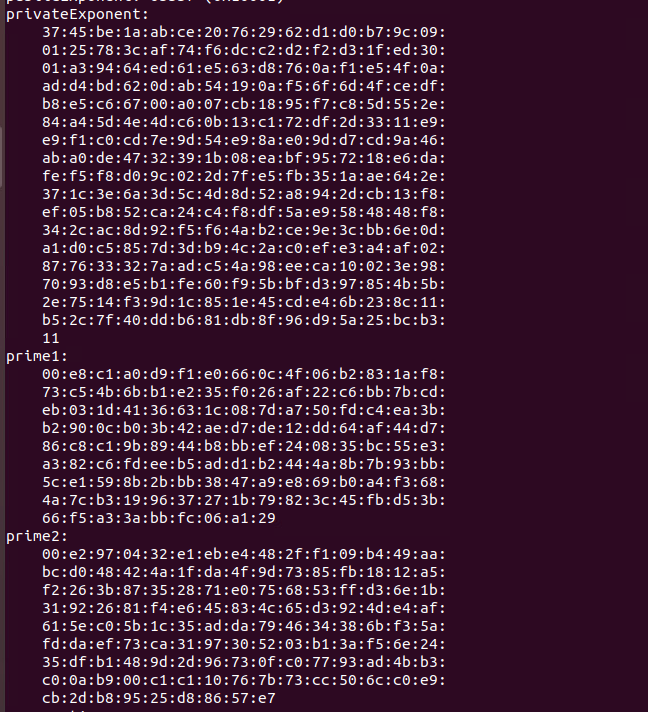
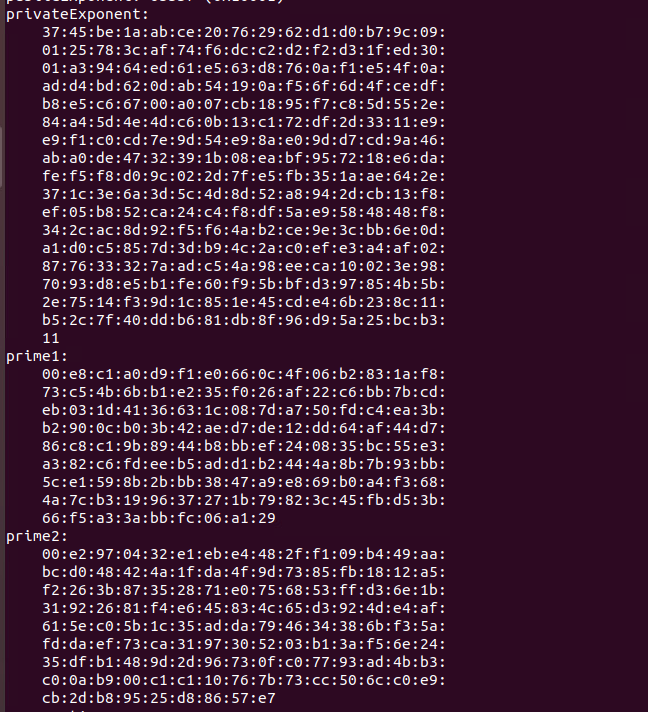
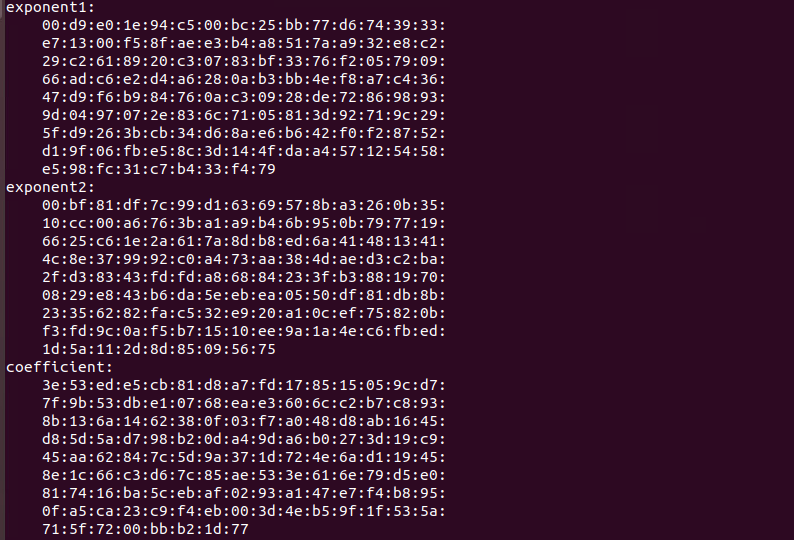
**Part 1:**

1. **Screenshot of the output from** openssl rsa -text -in <netid>\_private\_key.pem
   1. This was an extremely long output, excluding the key - here is our output from the command:

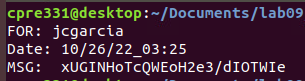




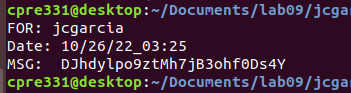




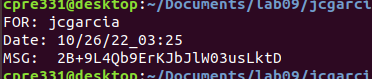
1. **Comparison of the same/different values observed across the extra generated keys**
   1. Which values are constant?
      1. It looks like the public exponent is constant across the two generated values.
   2. Which ones vary?
      1. Everything aside from the public exponent is different.
   3. What do these values represent?
      1. Based on what we know about rsa, here is the terminology:
         1. modulus: p \* q
         2. public exponent: e
         3. private exponent: d
         4. prime 1: p
         5. prime 2: q
         6. exponent 1: dp
         7. exponent 2: dq
         8. coefficient: dInv
2. **Discussion of the differences between FTP and SFTP.**
   1. Why would you want one over the other?
      1. FTP does not encrypt files as they are transferred from client to server/client - whereas SFTP encrypts these files using SSH. You would prefer SFTP to avoid transferring plaintext over the internet where it can be read using packet sniffing.
   2. Why did we need to specify our private key?
      1. We need to specify our private key to allow the server to authenticate us as a user. Since we uploaded our public key earlier, one would notice that we did not have to use a password to authenticate with the server when we logged in - this is because SFTP uses a private key and a public key combination to authenticate users and grant access.
   3. What protection does this offer?
      1. This offers the guarantee of nonrepudiation, unless someone shares their private key or it gets leaked, as we can confirm that the user we are allowing to access our server is who they say they are, providing access to the correct personnel - insuring the other four pillars of information assurance.
3. **Screenshot of the five messages [netid]1.txt, [netid]2.txt, … [netid]5.txt**
   1. Text 1: Not verified



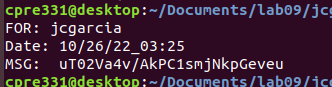
* 1. Text 2: Not verified



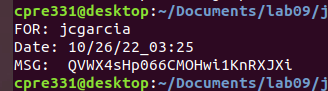
* 1. Text 3: Verified



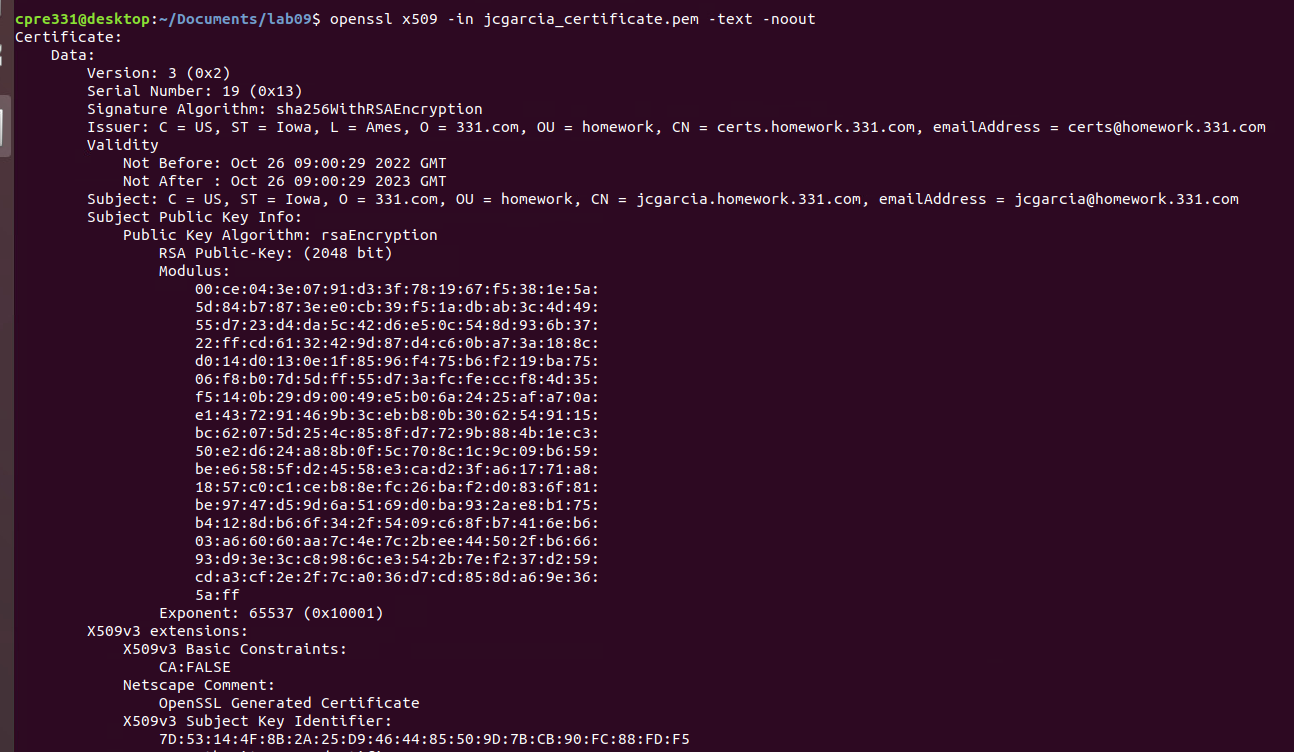
* 1. Text 4: Not verified

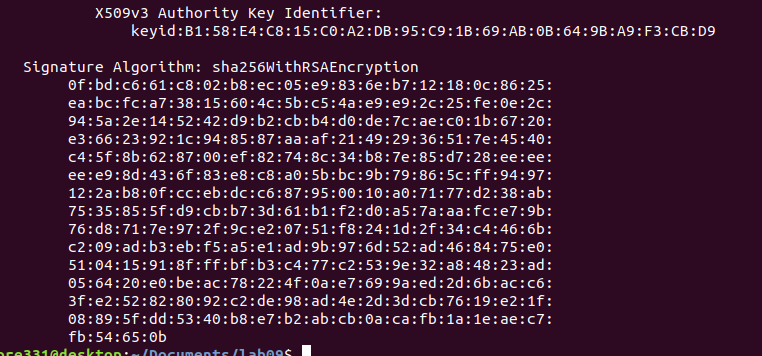


* 1. Text 5: Not verified



1. **Discussion on hash verification**
   1. What is known about the message?
      1. Pertaining to the hash verification message, we know that the message was signed with a private key and verified using the public key pair. Therefore, if the message was modified, it would prove to be not verified as we saw with 4 of the 5 messages. Here we saw that message 3 was not modified after being signed by the private key.
   2. What is the message protected against and what is it vulnerable to?
      1. The message in general protects the idea of non-repudiation as we know for certain who sent the message, in theory. Unless their private key was leaked.
      2. Additionally, we know that this hash method is vulnerable to a brute force hash attack, in which the attacker will attempt multiple values until they retrieve the correct encryption key in order to gain the private key of the sender.
2. **Discussion on what the message generated in step 8e protected against and what it is vulnerable to (compared to the message we downloaded in step 6).**
   1. **Since this message is encrypted using RSA, this message is protected against plaintext leaks to someone who does not have the private key/public key combo. The issue with this message is that it is not signed, and therefore could fall victim to a man in the middle type of attack as someone could pose as the recipient, decrypt the message using a public key method, etc.**
3. **Screenshot of the signed certificate ([netid]\_certificate.pem) when looked at through openssl**

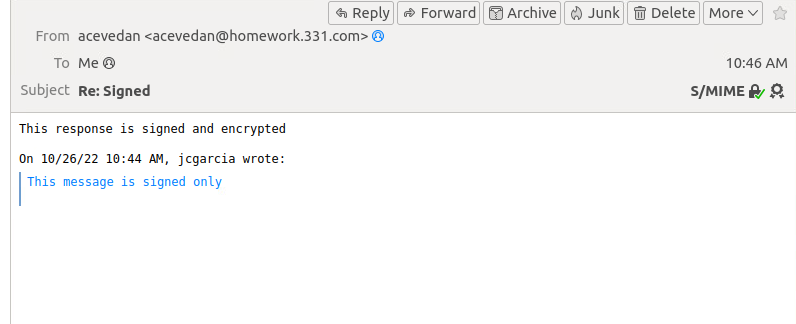




1. **Discussion from step 12**
   1. Do any parts of the certificate match with your private key? If so, why?
   2. What was happening during the Certificate Signing process? Why did you need to submit it for signing?

**Part 2:**

**a. Screenshot of signed and encrypted message received from a classmate**



**b. Explain why you couldn’t send an encrypted message straight away - why did you need to send a signed-only message first?**

Similar to how SSL works, we needed to complete a handshake in order to send an encrypted message. When we sent the signed-only message, we are essentially signing something with our “public” key. When our classmate “acknowledged” this - they were able to retrieve our certificate from our email. Only with this, were we able to send a message encrypted with our private key, to be decrypted with our public key that our classmate had previously received. Otherwise, we wouldn’t have their key/certificate and our classmates would then find themselves unable to decrypt.