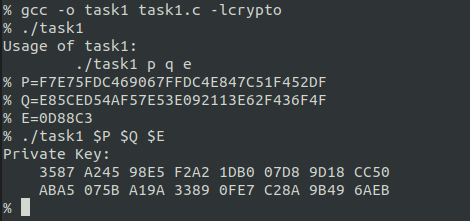
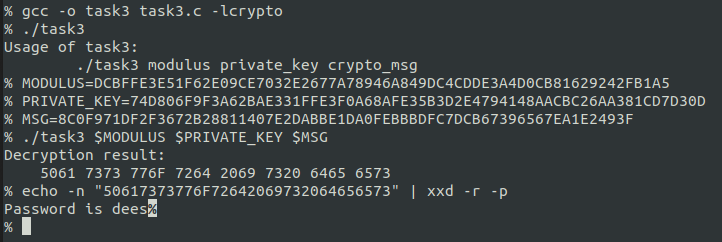
# Task 1: Deriving the Private Key



# Task 2: Encrypting a Message



# Task 3: Decrypting a Message



# Task 4: Signing a Message

To prepare to sign the message, I first set the modulus and private (d) key values to variables for easy reference. Then I compiled my signing app and ran it without arguments to show the usage info.

A picture containing food

Description automatically generated

I converted the requested message to its hexadecimal equivalent. I had to add an escape for the “$” since it is a special character in bash. I ran my signing app with the provided values and the computed signature is displayed in the screenshot.

A black sign with white text

Description automatically generated

Next, I made the requested modification to the message (changing $2000 to $3000) and ran the signing tool. The result was a completely different signature despite only a single byte of the message changing. As such, it would be very difficult (if not impossible) for an attacker to make a change to the message and predict the new signature. Thus, if the message and signature match then it is near certain that the message has not been altered.

A black sign with white text

Description automatically generated

# Task 5: Verifying a Signature

To verify the message and signature provided, I first converted the message to its hexadecimal equivalent and saved it to a variable for easy reference. I stored the signature, public key and modulus hex values in variables as well. Next, I compiled my verifier app and ran it without arguments to show the usage info. I ran the app again with the requested arguments, which displays the both the received and signed messages. The received message is the hex value of the message I received, and the signed message is the computed hex value of the message that was originally signed.

In this case, the two values do not match and so the signature is invalid. The issue appears to be a missing character at the 14th byte of the received message. When converted back to text you can see that in the originally signed message missile was spelled correctly, while in the received message it is missing its second letter “i.”

A screenshot of a cell phone

Description automatically generated

As requested, I modified the last byte of the signature and re-ran the verification program. The result was a completely different signed message. This suggests that it would be nearly impossible for an attacker to modify a signature to match a message without knowing the original private key.

A close up of a black background

Description automatically generated