**HW-10**

**Task – 1**

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

The hexadecimal values of p, q, and e are listed below

p = F7E75FDC469067FFDC4E847C51F452DF

q = E85CED54AF57E53E092113E62F436F4F

e = 0D88C3

We need to calculate the private key d.

Code

A screenshot of a computer

Description automatically generated

A black and purple background

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

The private key d is **3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB**

**Task – 2**

In this task let (e, n) be the public key. We have to encrypt the message "A top secret!"

We use the following python command to convert a plain ASCII string to a hex string.

A black background with white text

Description automatically generated

The public keys are listed in the followings (hexadecimal and the private key d to verify encryption result.

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

e = 010001 (this hex value equals to decimal 65537)

M = A top secret!

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

The hex string is converted to a BIGNUM using the hex-to-bn API BN hex2bn()

A screenshot of a computer program

Description automatically generated



We use the given value of the private key to verify if our encrypted result is right by decrypting the encrypted result.

A screenshot of a computer

Description automatically generated

The encrypted result is correct, because on decryption, we get the same value of M.

**Task – 3**

The public/private keys used in this task are the same as the ones used in Task 2. Please decrypt the following ciphertext C

C = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F

Convert it back to a plain ASCII string.

Given:

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

e = 010001 (this hex value equals to decimal 65537)

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

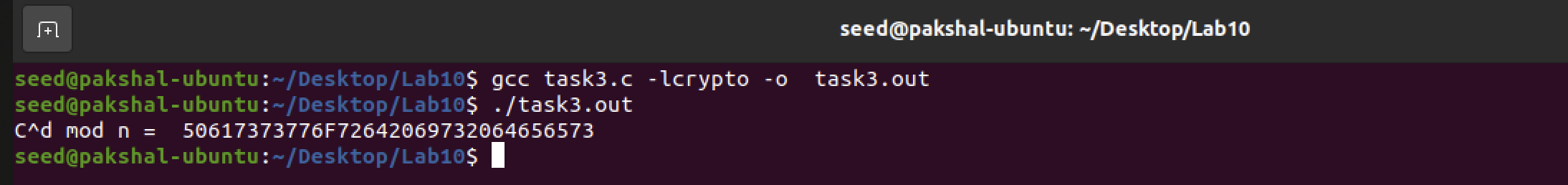
C = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F

M = ? which is (C ^ d) mod n

**Code**

A screenshot of a computer program

Description automatically generated



The hex string needs to be converted back to the ASCII message string as shown below

A screenshot of a computer

Description automatically generated

Thus, the ciphertext has been decrypted and the original message was ‘Password is dees’.

**Task – 4**

In this task we need to generate a signature for the following message

M1 = I owe you $2000.

After making a slight change to the message M, such as changing $2000 to $3000.

M2 = I owe you $3000.

Given

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

e = 010001 (this hex value equals to decimal 65537)

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

A screenshot of a computer

Description automatically generated

M1 and M2 in hexadecimal.

M1 = 49206f776520796f752024323030302e

M2 = 49206f776520796f752024333030302e

Both M1 and M2 differ by couple of bits.

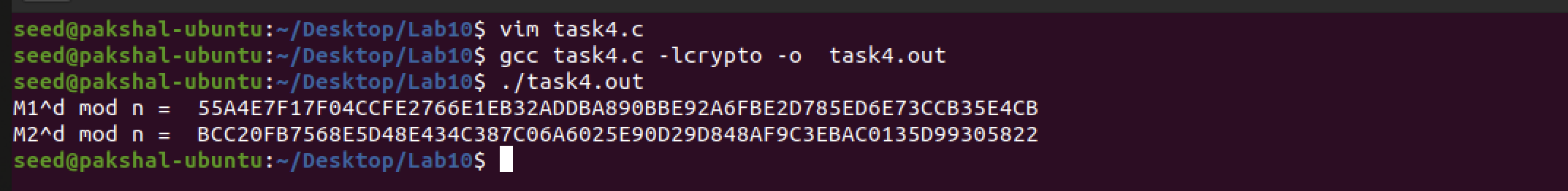
**Code**

A screenshot of a computer program

Description automatically generated

A black and purple background

Description automatically generated with medium confidence



We can see that even though there was only one single digit difference in the message M1 and M2, there generated signatures vary a lot even for single digit being different.

**Task – 5**

Bob receives the message from Alice with her signature S. We know Alice public key (e,n). verify whether the signature is indeed Alice’s or not.

Given.

M = Launch a missile.

S1 = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F

e = 010001 (this hex value equals to decimal 65537)

n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115

After corrupting the signature by changing one bit from 2F to 3F

S2 = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F

Message M in hexadecimal

A computer screen shot of a computer code

Description automatically generated

**Code**

A screenshot of a computer program

Description automatically generated

A black and purple background

Description automatically generated with medium confidence

A computer screen with white text

Description automatically generated

S1 is the correct signature because it gives back the original message value. Even if we modify the signature by even one bit, it completely changes the generated message as shown above.

Task – 6

In this task, we have to manually verify an X.509 certificate using our program. An X.509 contains data about a public key and an issuer’s signature on the data. We will download a real X.509 certificate from a web server, get its issuer’s public key, and then use this public key to verify the signature on the certificate.

Task 6.1

In this task we use the given command to download the X.509 certificate from the paypal web server.

A screenshot of a computer

Description automatically generated

we copy and paste each of the certificates (the text between the line containing "Begin CERTIFICATE" and the line containing "END CERTIFICATE", including these two lines) to two files. We call the first one paypal.pem and the second one paypal\_2.pem.

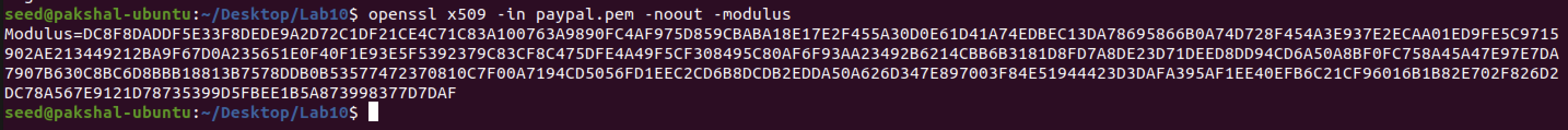
A screenshot of a computer screen

Description automatically generated

**Task 6.2**

In this sub-task, we need to extract the public key (e, n) from the issuer’s certificate. Openssl provides commands to extract certain attributes from the x509 certificates.

We can extract the value of n using the flag -modulus.



We can print all the fields and find the value of e.

A screenshot of a computer

Description automatically generated

Value of e in hexadecimal is 65537 (0x10001) as shown above.

**Task 6.3**

Here we extract the signature from the server’s certicate.

A black and purple background

Description automatically generated with medium confidence

We need to remove the spaces and colons from the data, so we can get a hex-string that we can feed into our program. The following command commands can achieve this goal. The tr command is a Linux utility tool for string operations.

A screenshot of a computer

Description automatically generated

In this case, the -d option is used to delete ":" and "space" from the data.

The signature S is



**Task 6.4**

In this task we extract the body of the server’s certificate. A Certificate Authority (CA) generates the signature for a server certificate by first computing the hash of the certificate, and then sign the hash.

To verify the signature, we also need to generate the hash from a certificate. Since the hash is generated before the signature is computed, we need to exclude the signature block of a certificate when computing the hash

X.509 certificates are encoded using the ASN.1 (Abstract Syntax Notation.One) standard, so if we can parse the ASN.1 structure, we can easily extract any field from a certificate. Openssl has a command called asn1parse used to extract data from ASN.1 formatted data, and is able to parse our X.509 certificate.

A screenshot of a computer program

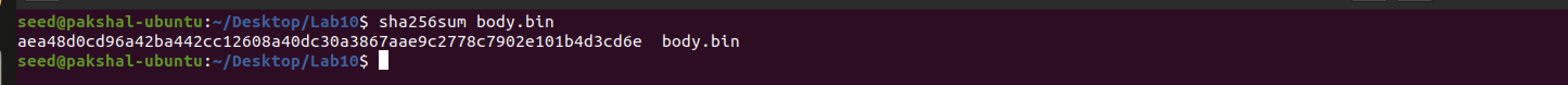
Description automatically generated

Now we can use the -strparse option to get the field from the offset 4, which will give us the body of the certificate, excluding the signature block.

A screenshot of a computer screen

Description automatically generated

Once we get the body of the certificate, we can calculate its hash using the following command:



**Task 6.5**

Verify the signature. Now we have all the information, including the CA’s public key, the CA’s signature, and the body of the server’s certificate. We can run our own program to verify whether the signature is valid or not.

**Code**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

We have successfully verified the signature to be the valid signature from the CA.