BIRZEIT UNIVERSITY

Faculty of Information Technology

Computer Systems Engineering Department

APPLIED CRYPTOGRAPHY
ENCS4320

RSA Public-Key Encryption and Signature Lab

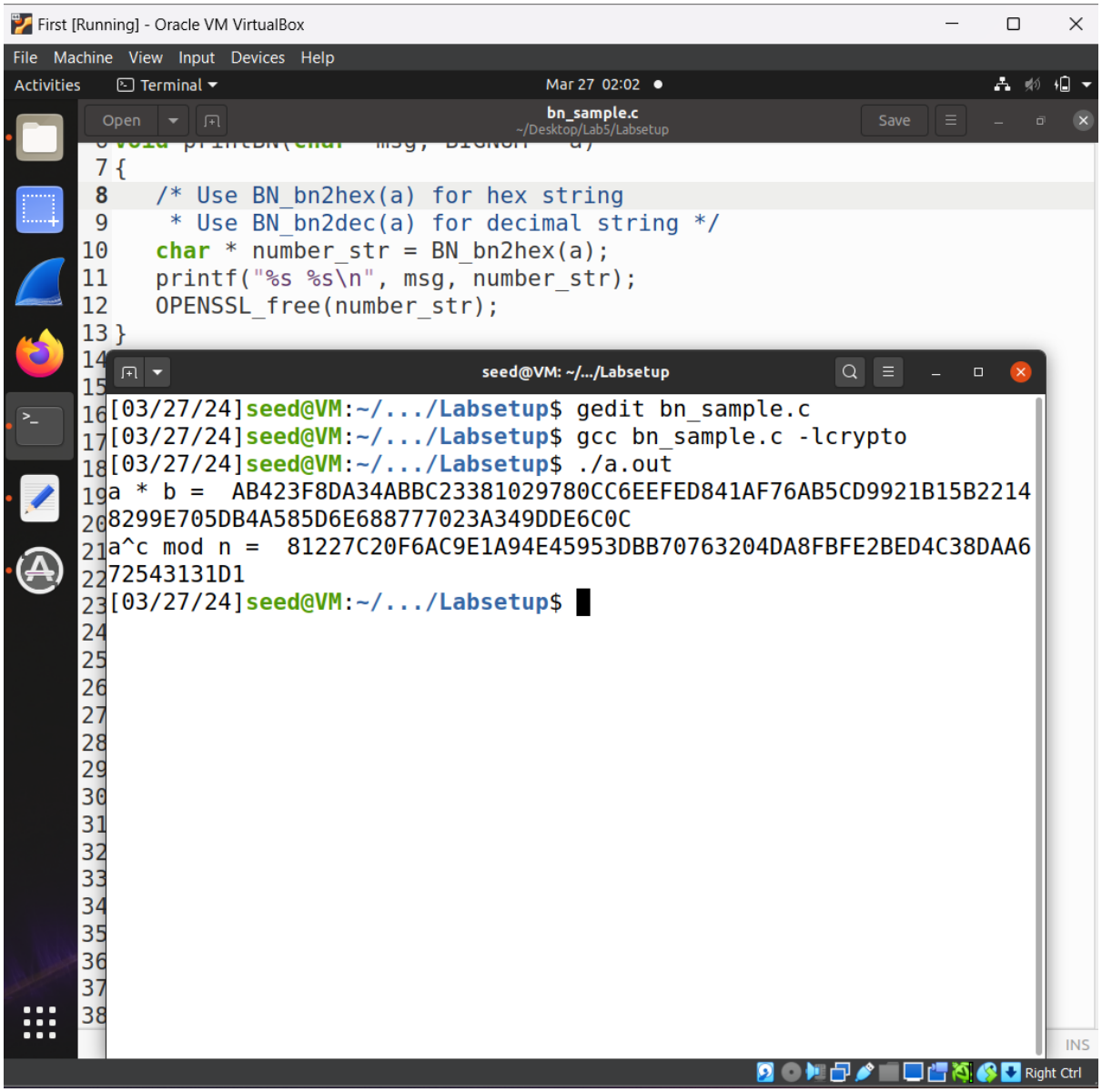
Prepared by: Dana Bornata – 1200284

Instructor: Dr. Ibrahim Nemer

Teacher Assistant: **Mohammed Balawi**

Section: 1

1.A Complete Example:



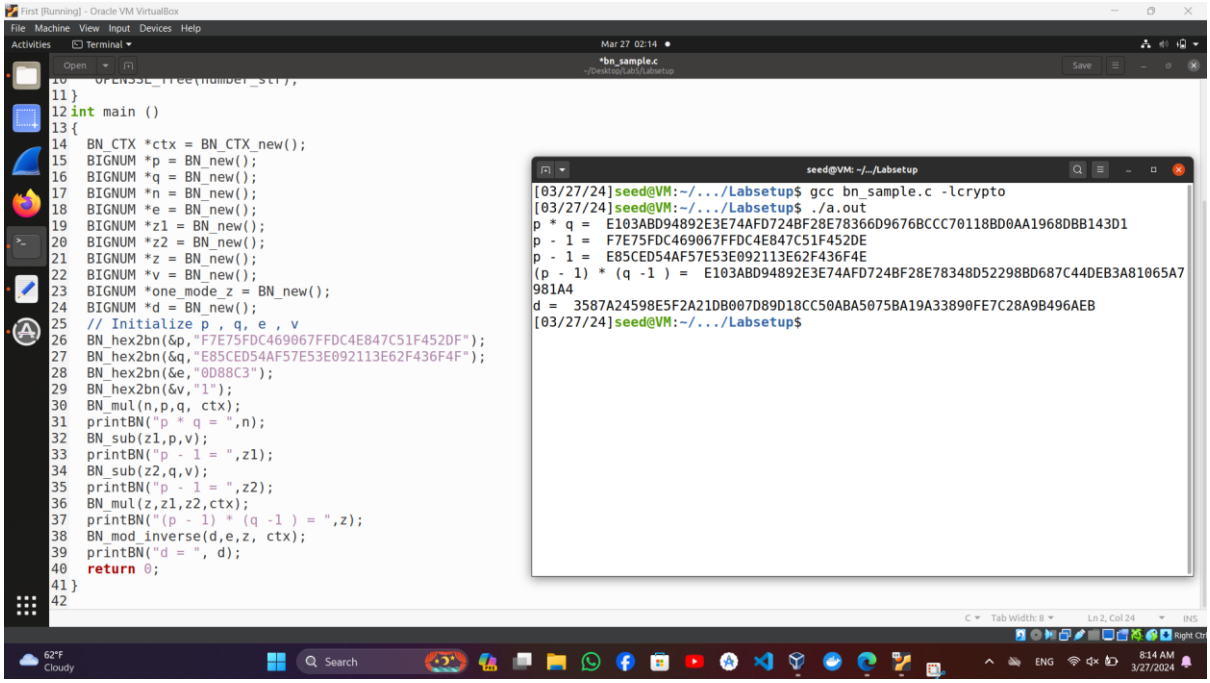
Task 1: Deriving the Private Key

3.1 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. The hexadecimal values of p, q, and e are listed in the following. It should be noted that although p and q used in this task are quite large numbers, they are not large enough to be secure. We intentionally make them small for the sake of simplicity. In practice, these numbers should be at least 512 bits long (the one used here are only 128 bits).

```
p = F7E75FDC469067FFDC4E847C51F452DF
q = E85CED54AF57E53E092113E62F436F4F
e = 0D88C3
```

ed = 1 mod $(p-1)(q-1)$



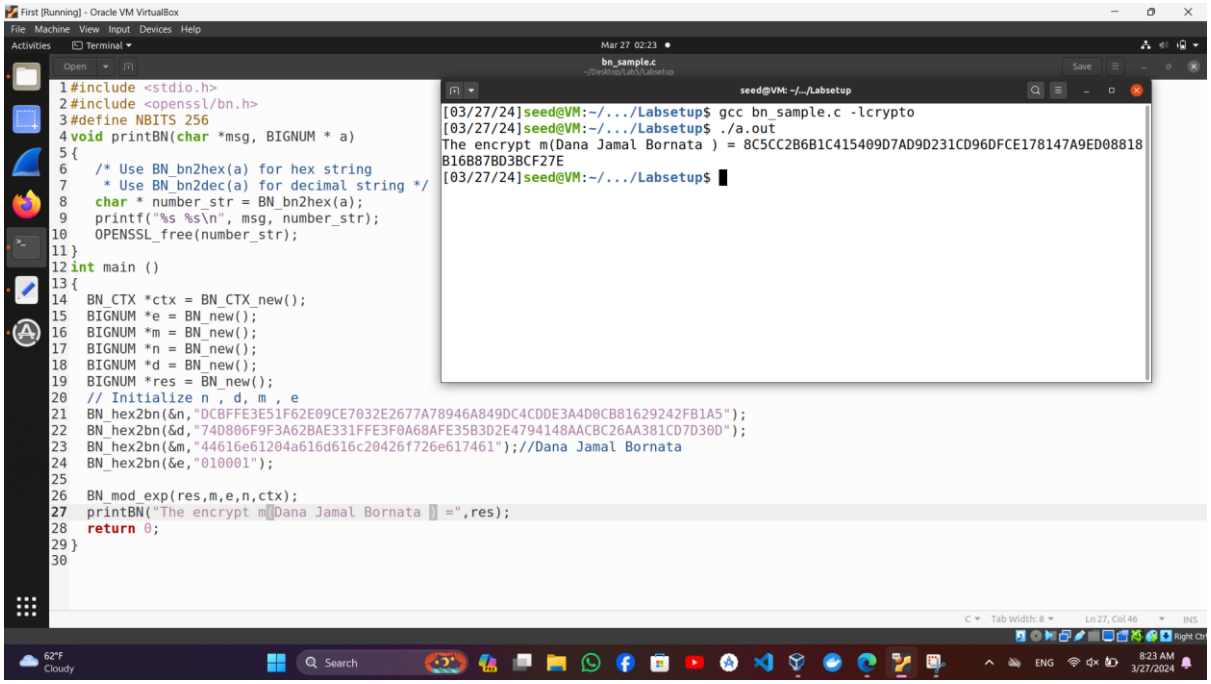
Task 2: Encrypting a message

3.2 Task 2: Encrypting a Message

Let `(e, n)` be the public key. Please encrypt the message `"A top secret!"` (the quotations are not included). We need to convert this ASCII string to a hex string, and then convert the hex string to a BIGNUM using the hex-to-bn API `BN_hex2bn()`. The following `python` command can be used to convert a plain ASCII string to a hex string.

```
$ python  -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421
```

C=m^e mod n



Home » Text to Hexadecimal

Text to Hexadecimal

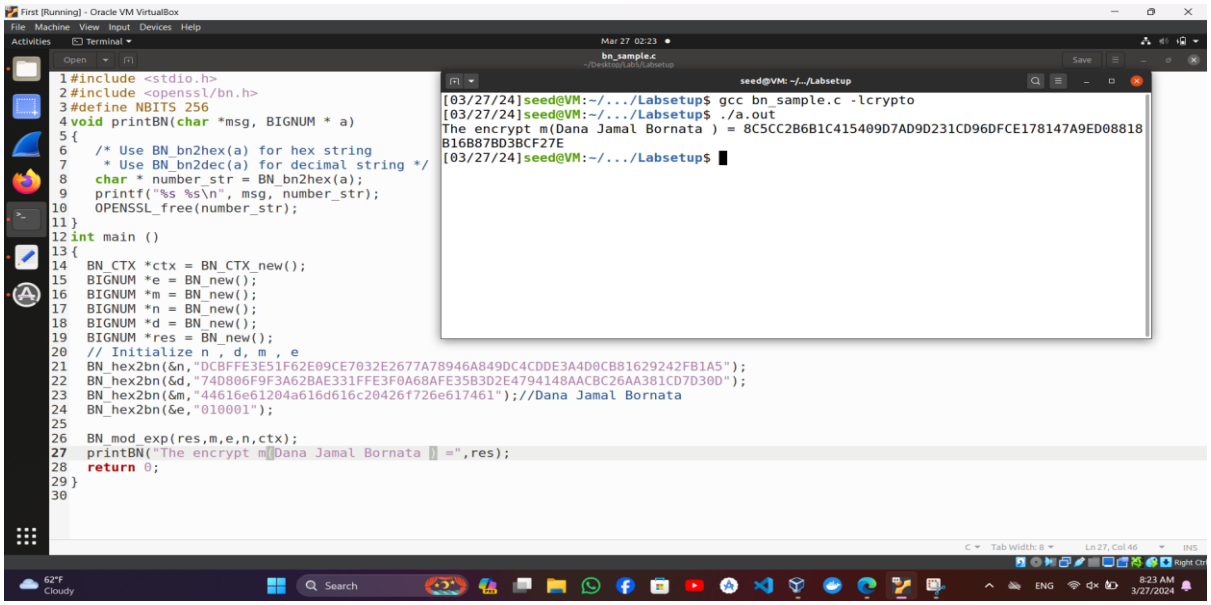
Online Text to Hexadecimal converter tool that convert any text into hex code.

Dana Jamal Bornata

Convert

Text to Hex

44616e61204a616d616c20426f726e617461



The encrypt m(Dana Jamal Bornata) = 8C5CC2B6B1C415409D7AD9D231CD96DFCE178147A9ED08818B16B87BD3BCF27E

Task 3: Decrypting a message:

3.3 Task 3: Decrypting a Message

The public/private keys used in this task are the same as the ones used in Task 2. Please decrypt the following ciphertext C, and convert it back to a plain ASCII string.

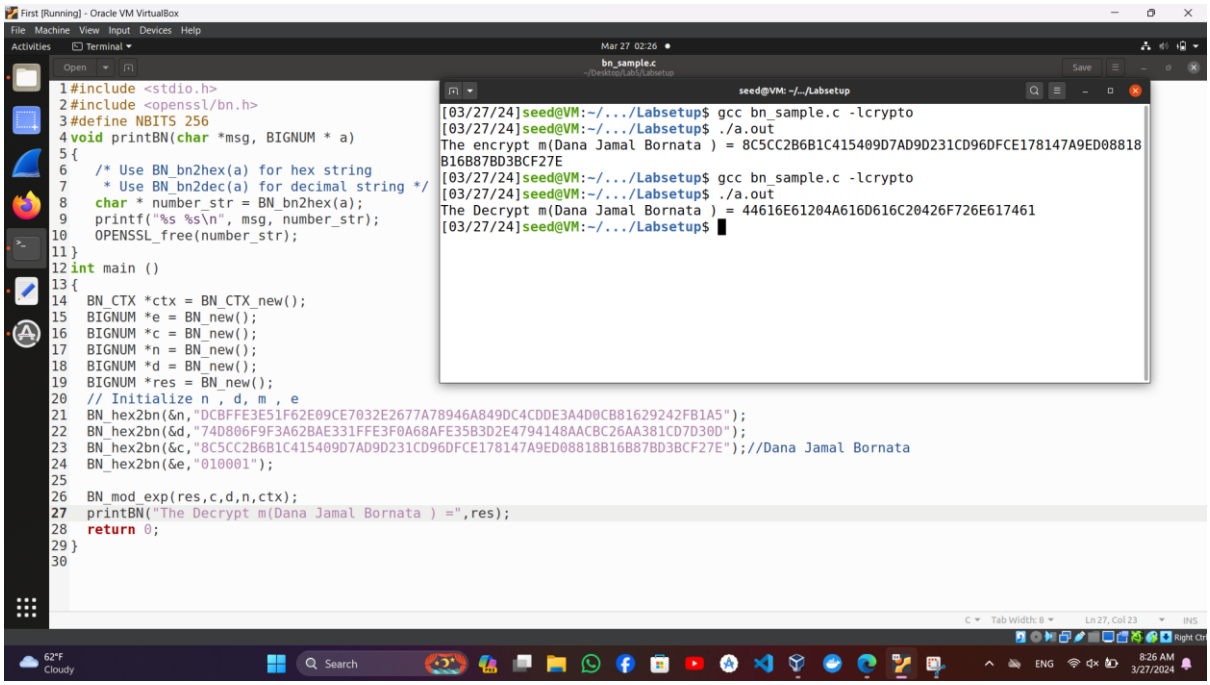
```
C = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F
```

You can use the following python command to convert a hex string back to to a plain ASCII string.

```
$ python  -c 'print("4120746f702073656372657421".decode("hex"))'
A top secret!
```

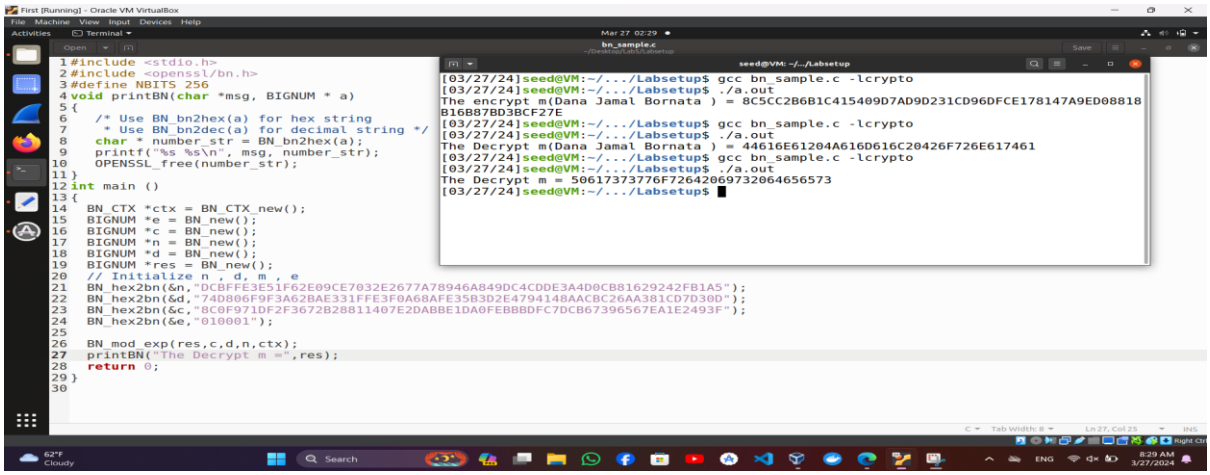
m=c^e mod n

C=
8C5CC2B6B1C415409D7AD9D231CD96DFCE178147A9ED08818B16B87BD3BCF27E

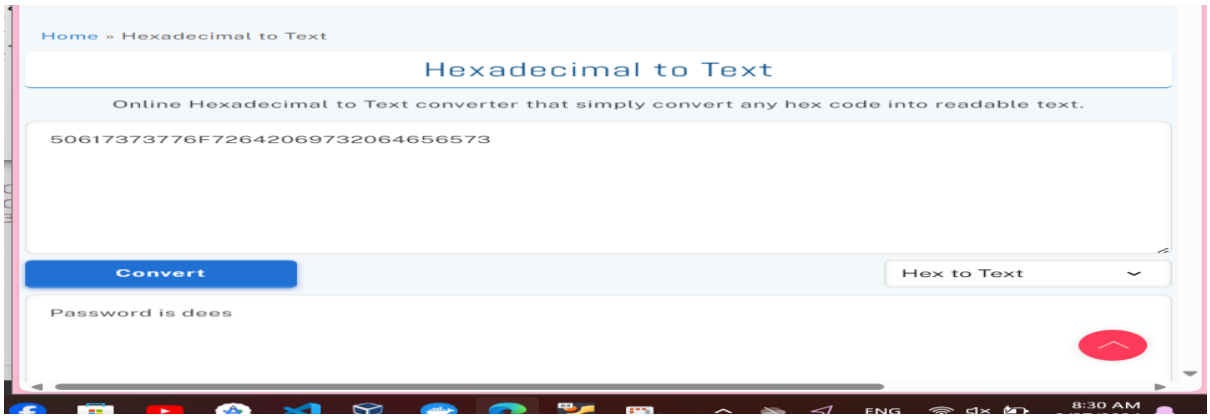


The Decrypt m(Dana Jamal Bornata) = 44616E61204A616D616C20426F726E617461

When the C is:
8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2 493F



The Decrypt m = 50617373776F726420697320646565 73



Task 4: Signing a Message

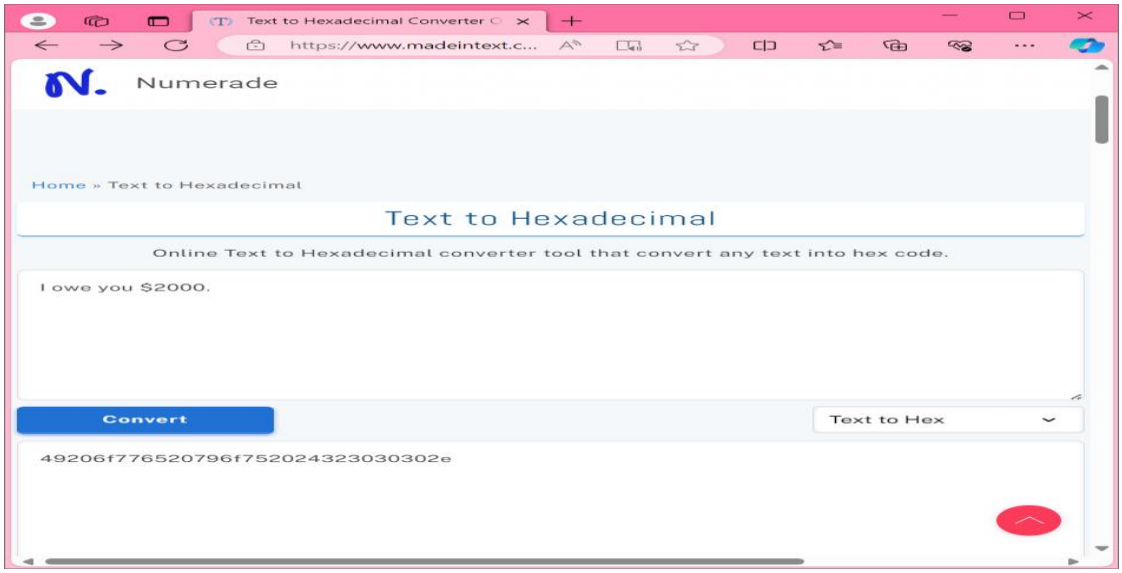
3.4 Task 4: Signing a Message

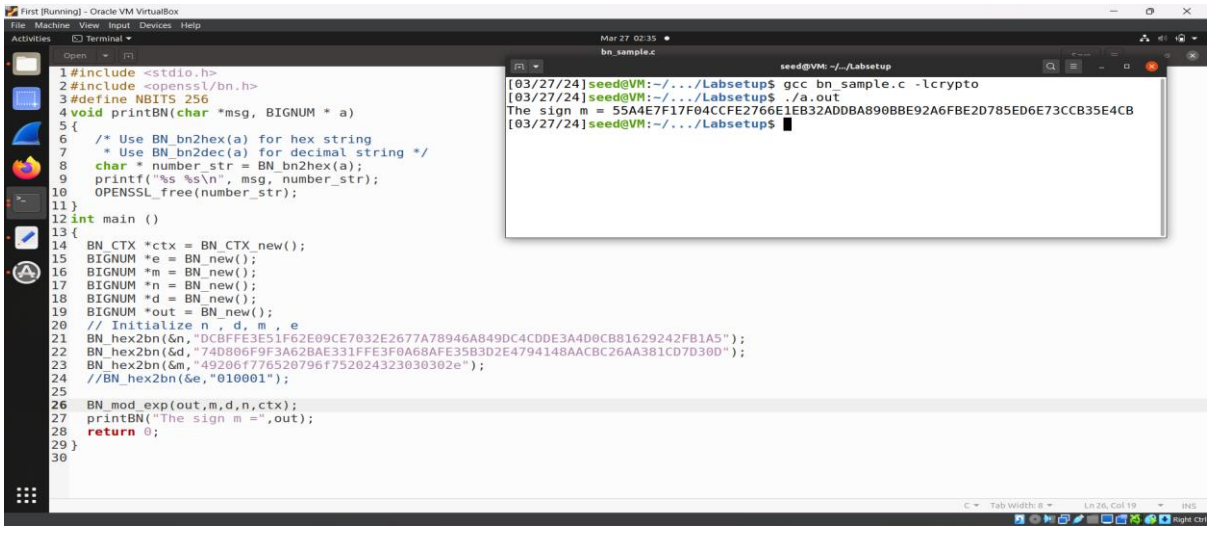
The public/private keys used in this task are the same as the ones used in Task 2. Please generate a signature for the following message (please directly sign this message, instead of signing its hash value):

```
M = I owe you $2000.
```

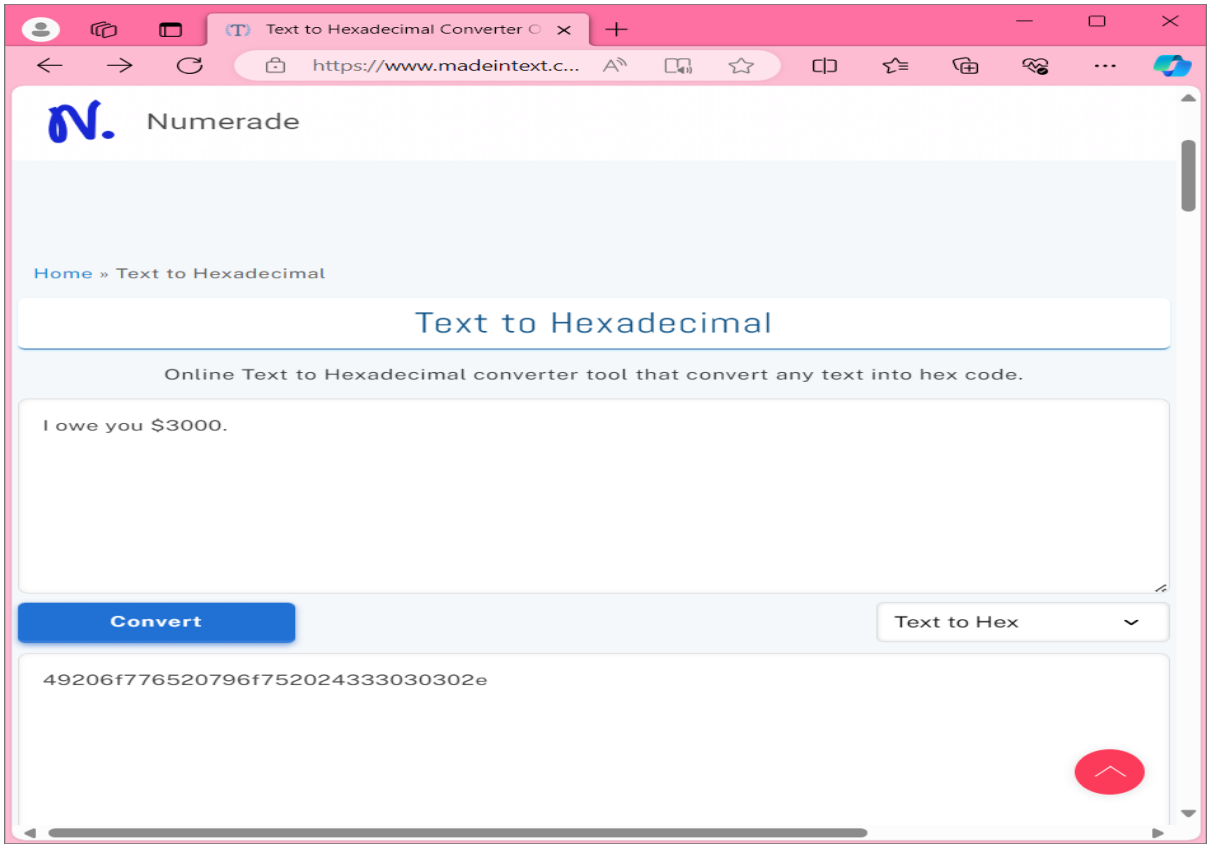
Please make a slight change to the message `M`, such as changing `$2000` to `$3000`, and sign the modified message. Compare both signatures and describe what you observe.

$2000

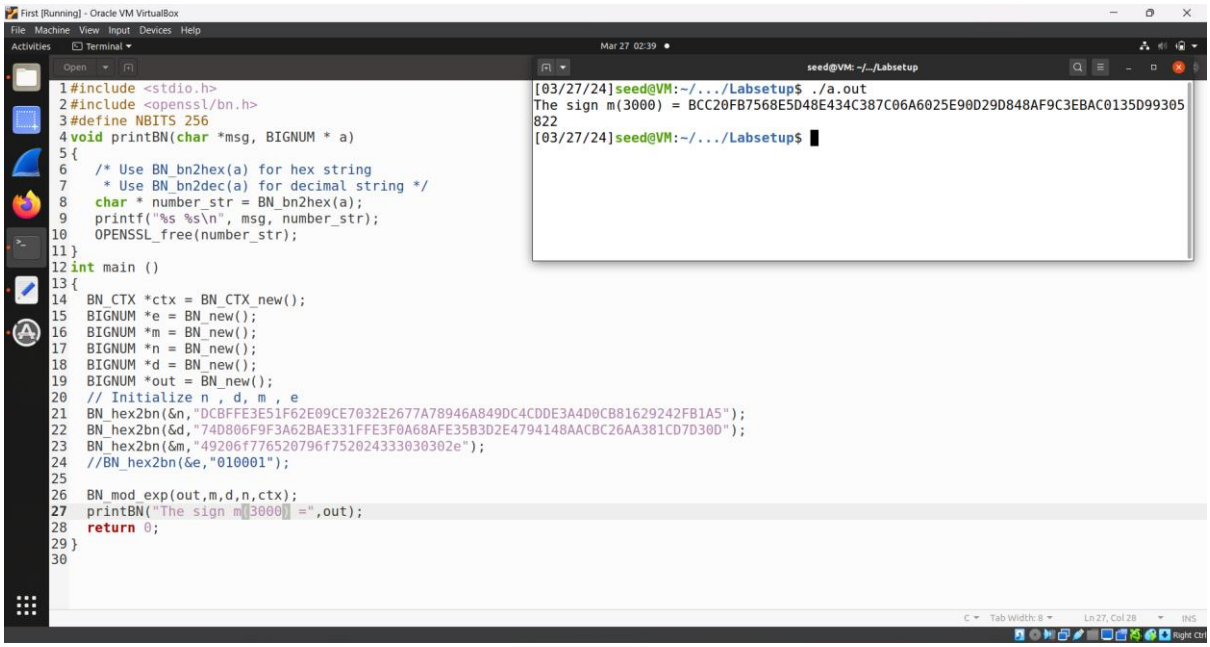




$3000



The sign m(3000) =
BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822



Task 5: Verifying a Signature

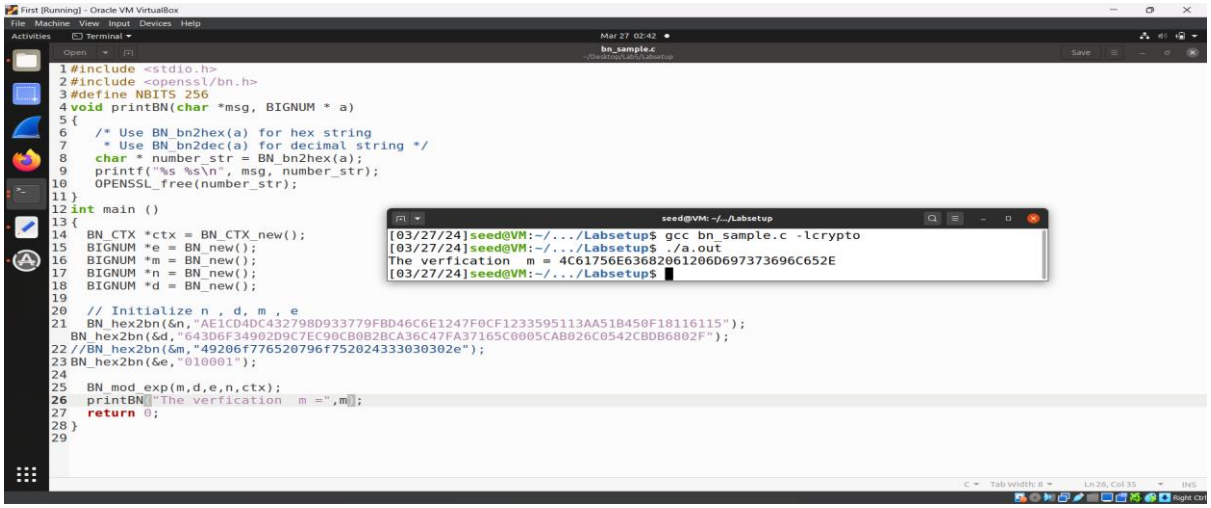
3.5 Task 5: Verifying a Signature

Bob receives a message M = "Launch a missile." from Alice, with her signature S. We know that Alice's public key is (e, n). Please verify whether the signature is indeed Alice's or not. The public key and signature (hexadecimal) are listed in the following:

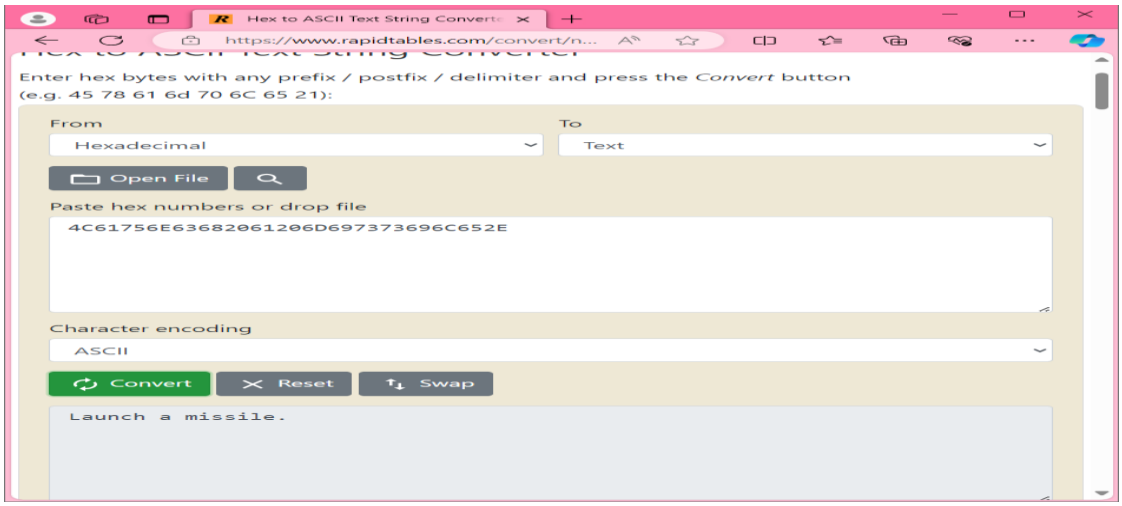
```
M = Launch a missile.
S = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
e = 010001 (this hex value equals to decimal 65537)
n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115
```

Suppose that the signature above is corrupted, such that the last byte of the signature changes from 2F to 3F, i.e, there is only one bit of change. Please repeat this task, and describe what will happen to the verification process.

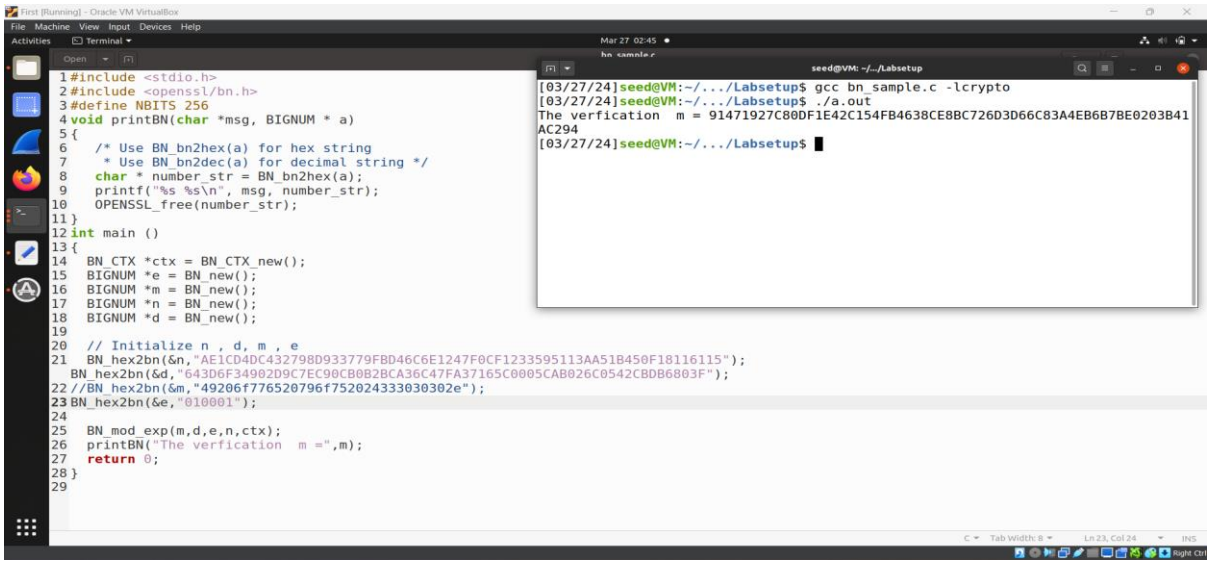
M= *Se* mod n



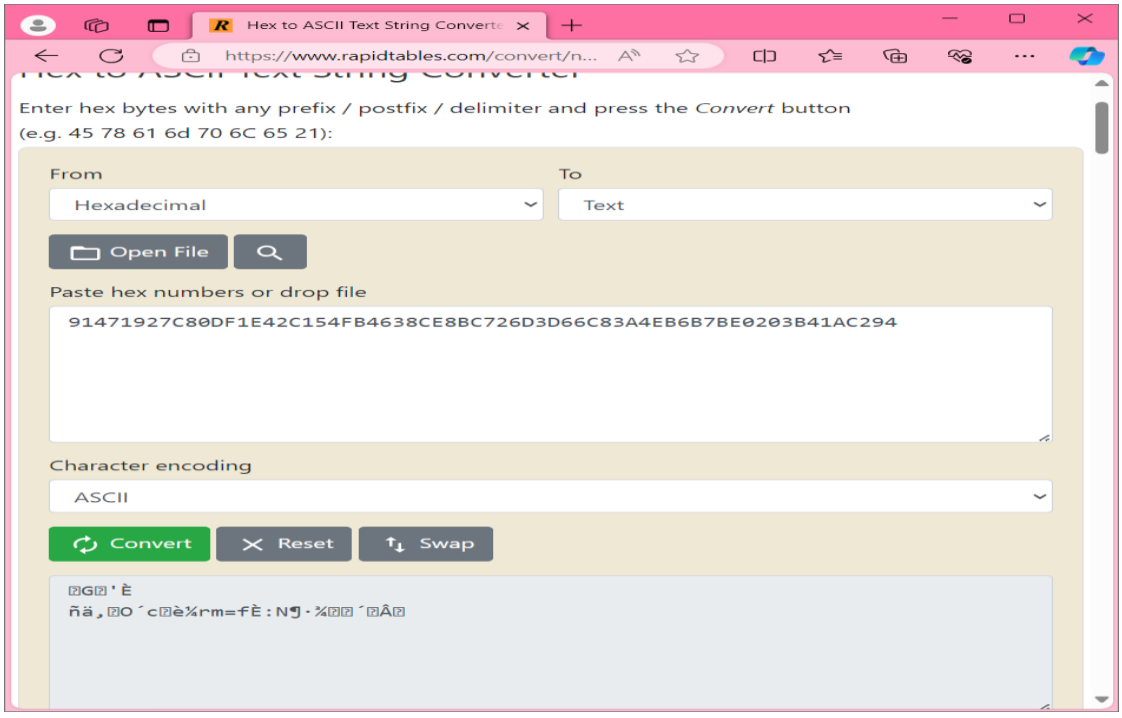
The verfication m = 4C61756E63682061206D697373696C652E



2f>>3f



The verfication m = 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294



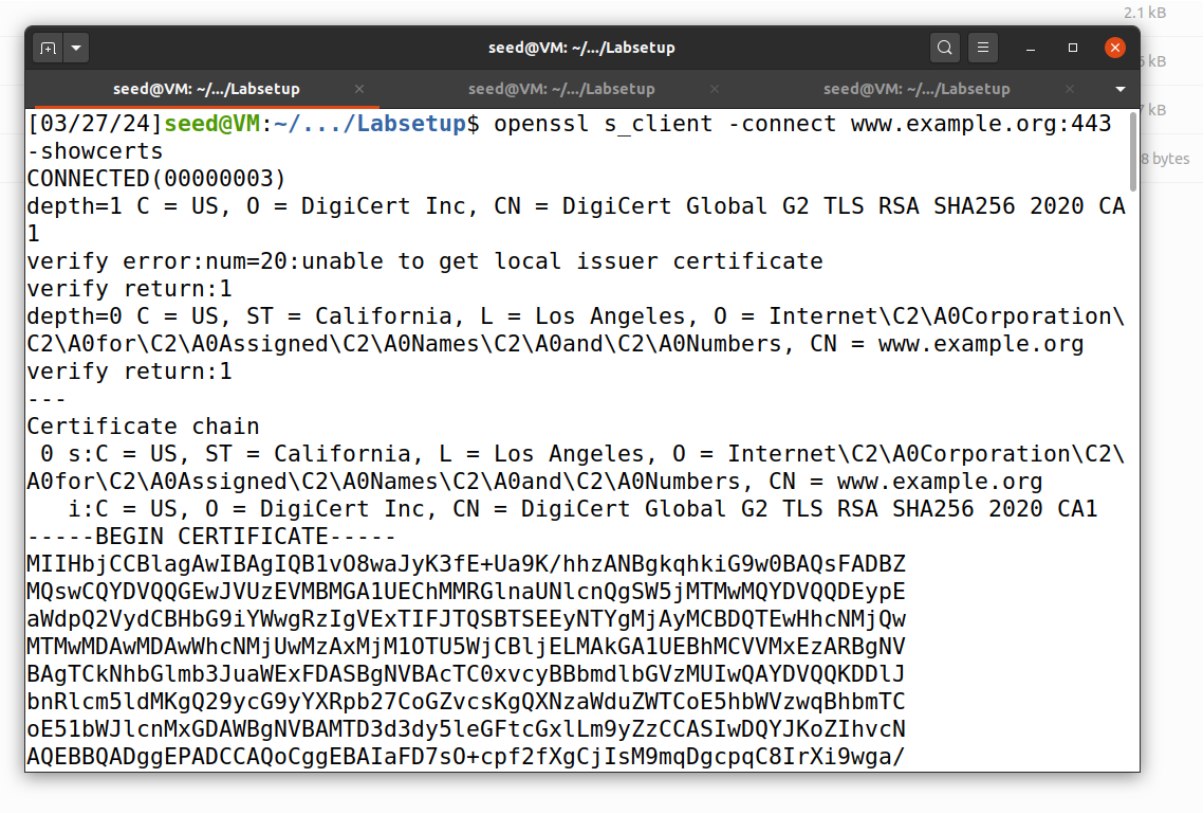
We can notice that changing just one byte of the signature gives us a totally different message.

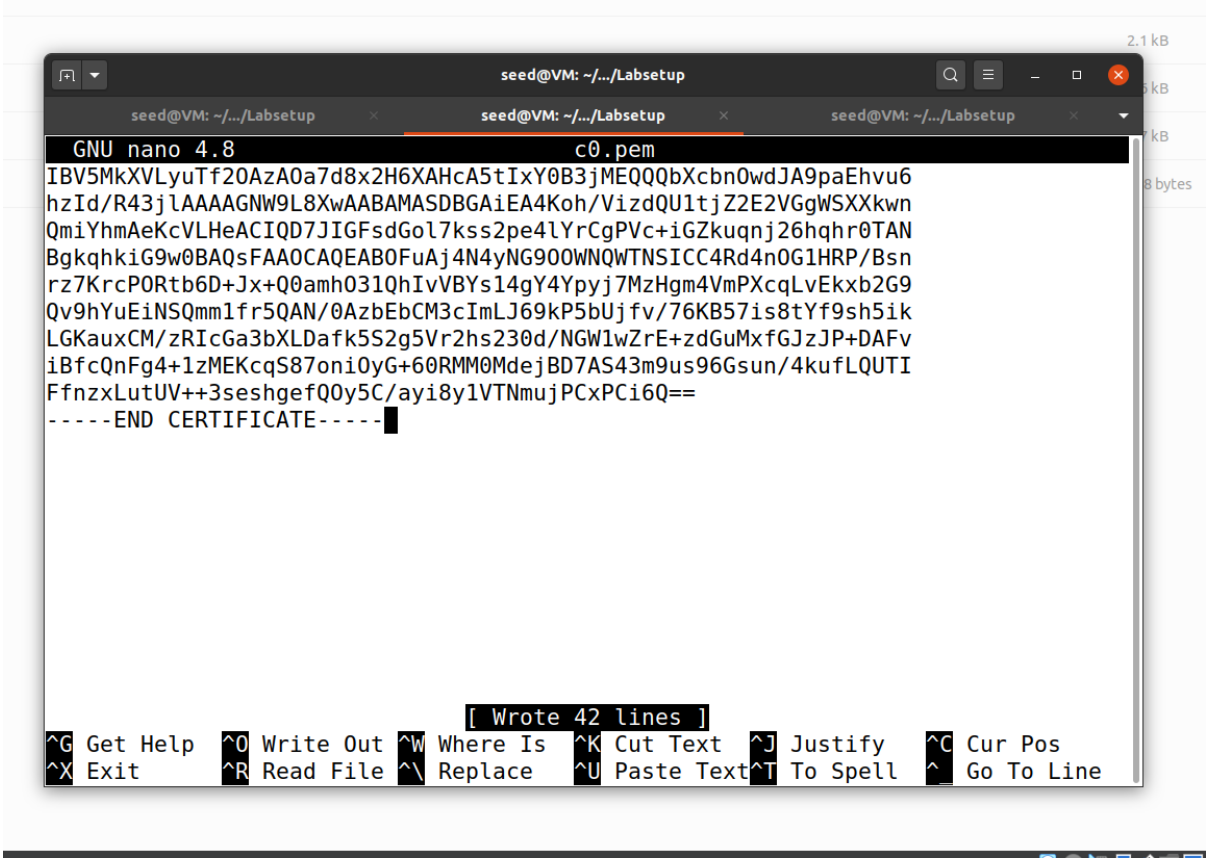
Task 6: Manually Verifying an X.509 Certificate

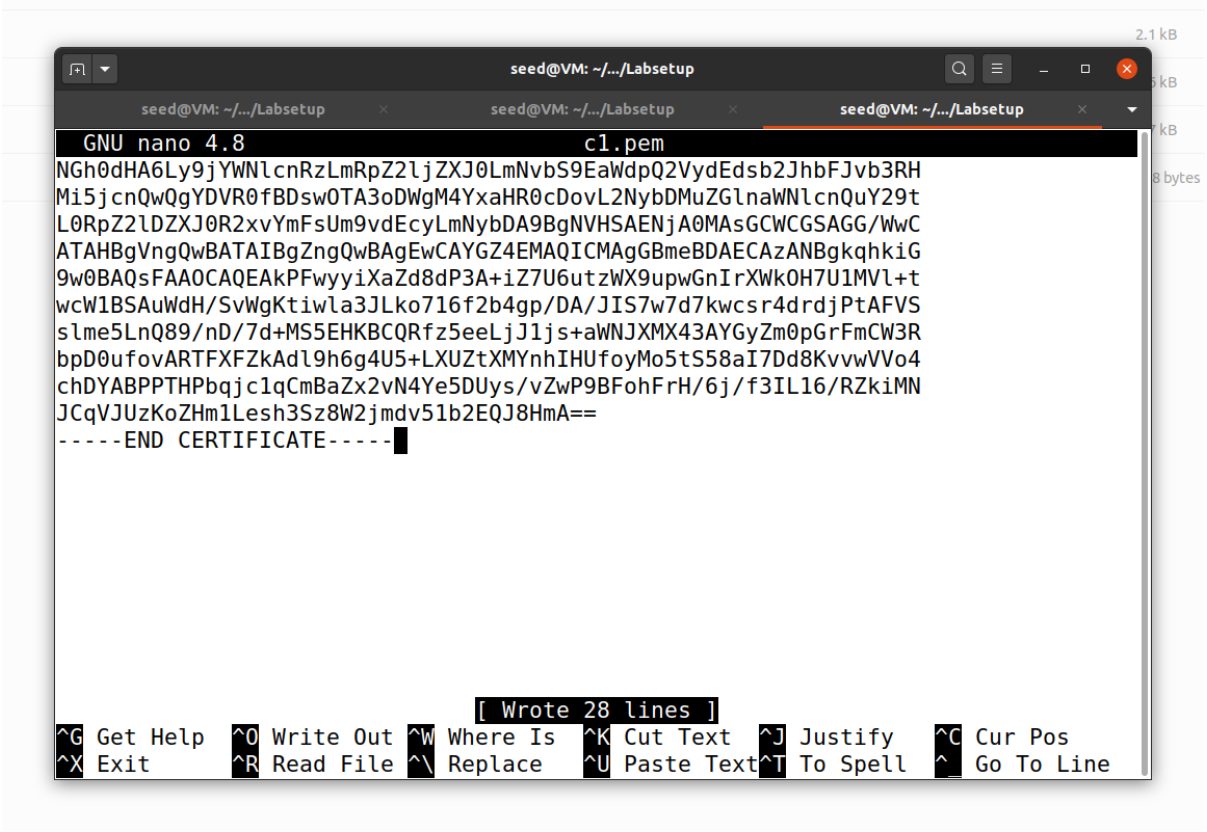
3.6 Task 6: Manually Verifying an X.509 Certificate

In this task, we will 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.

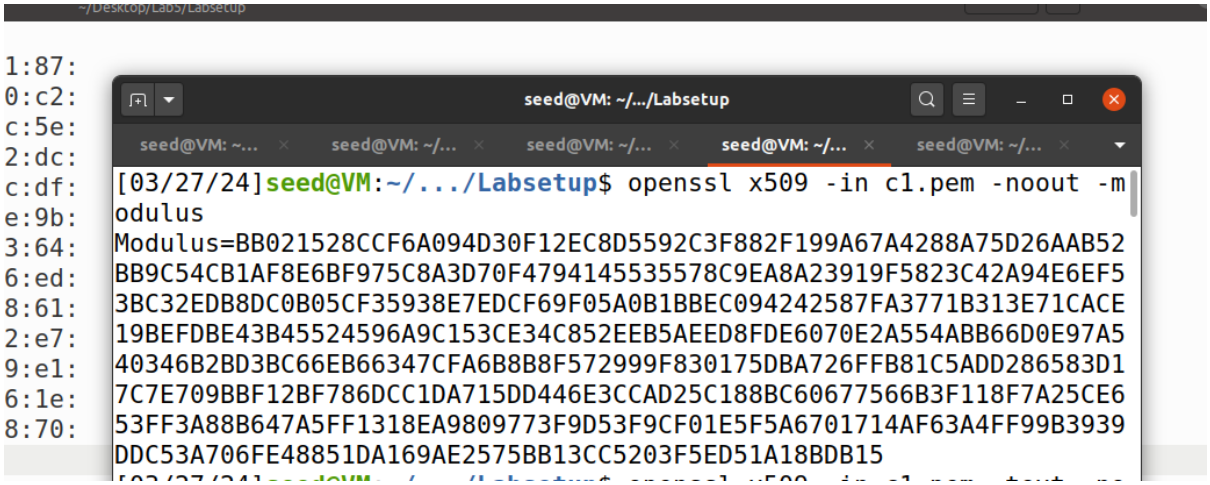
Step 1: Download a certificate from a real web server.



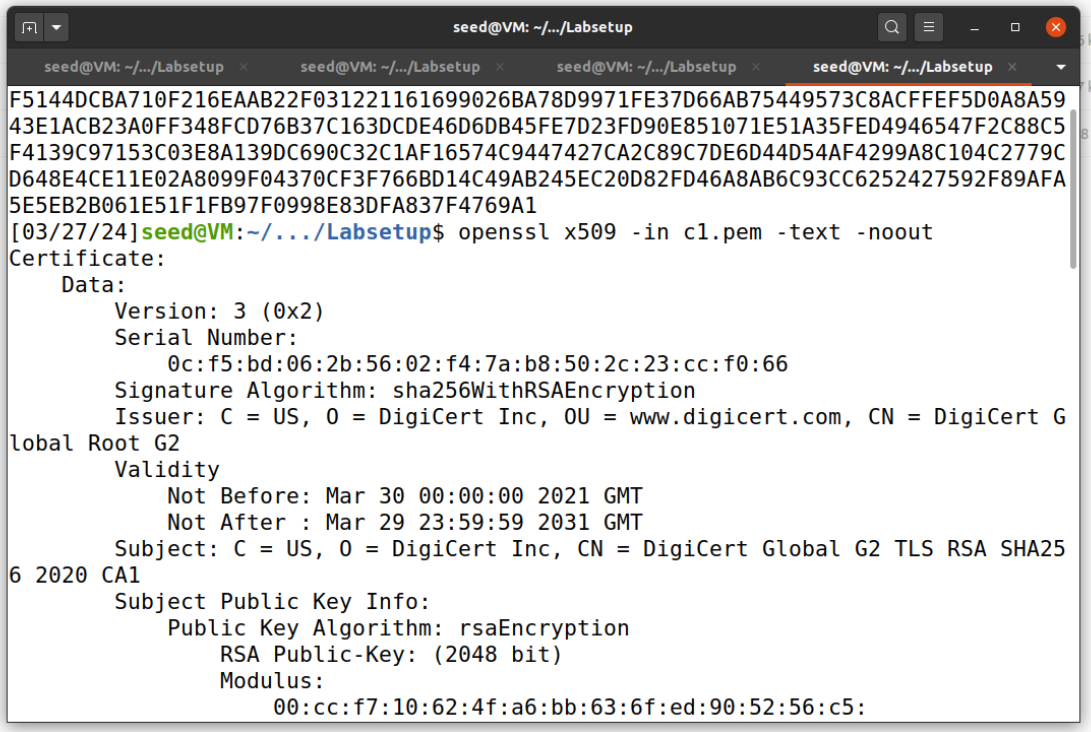


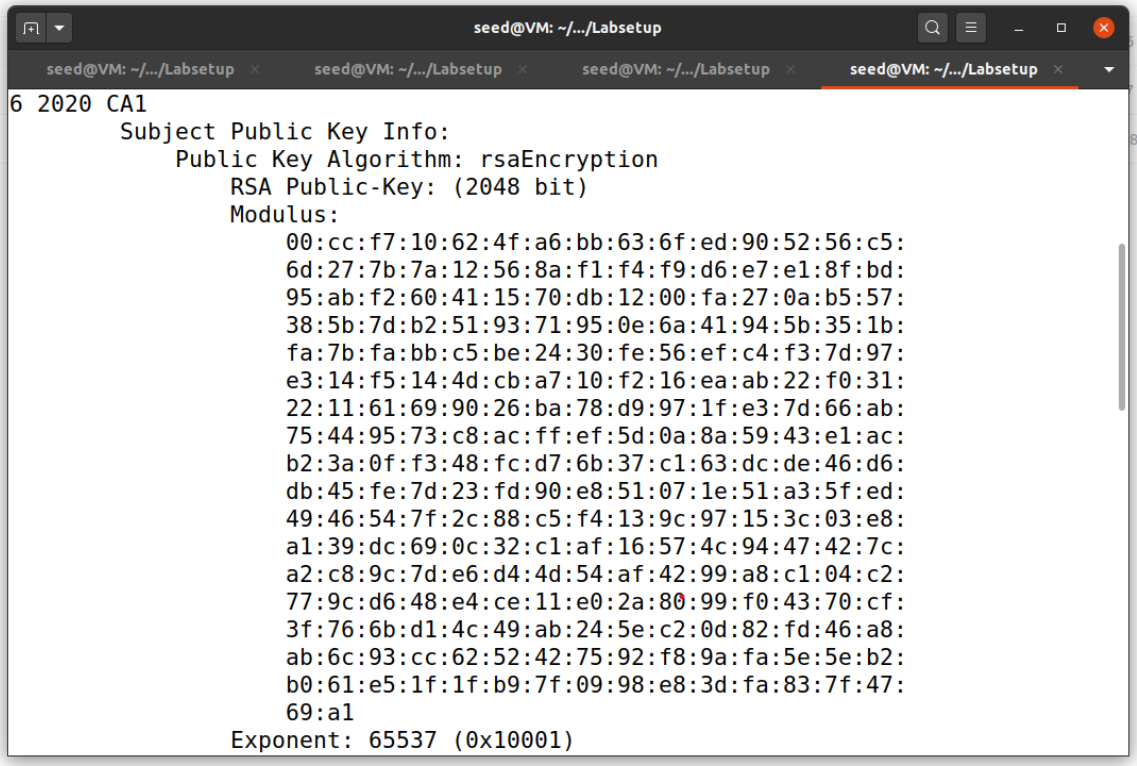


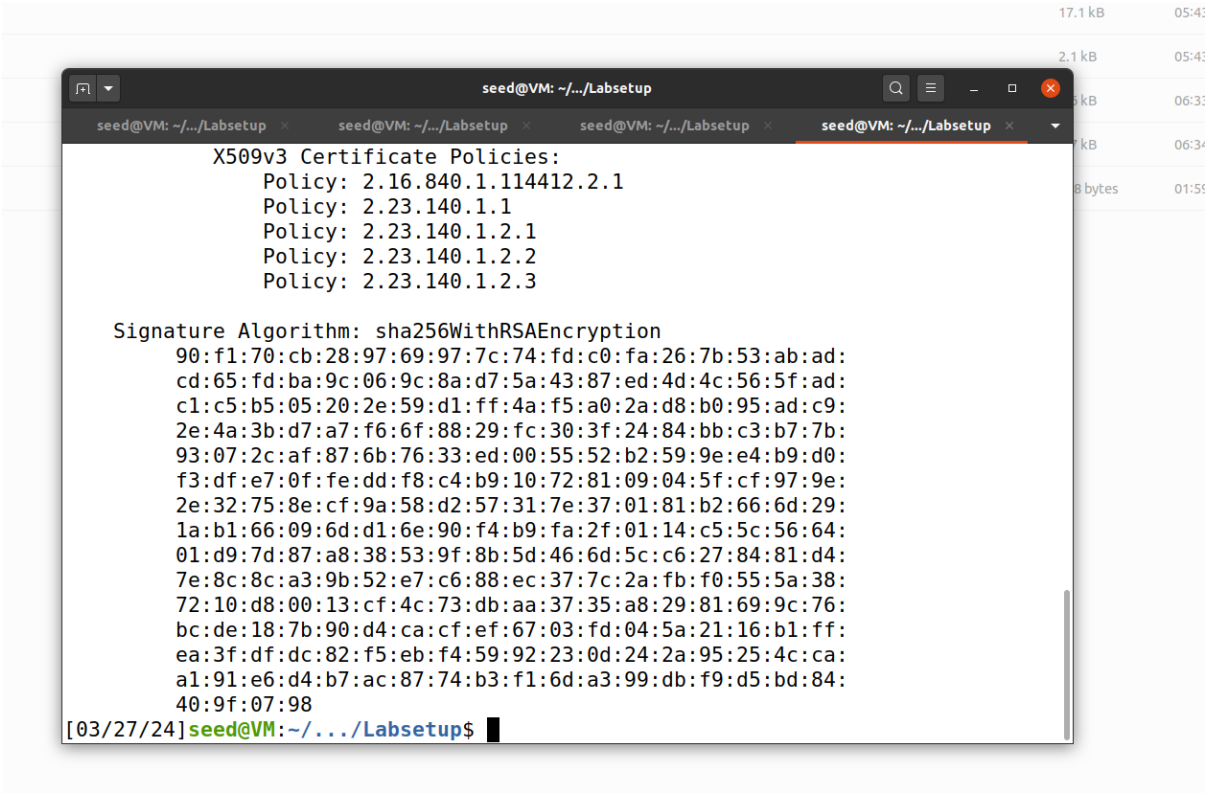
Step 2: Extract the public key (e, n) from the issuer's certificate.



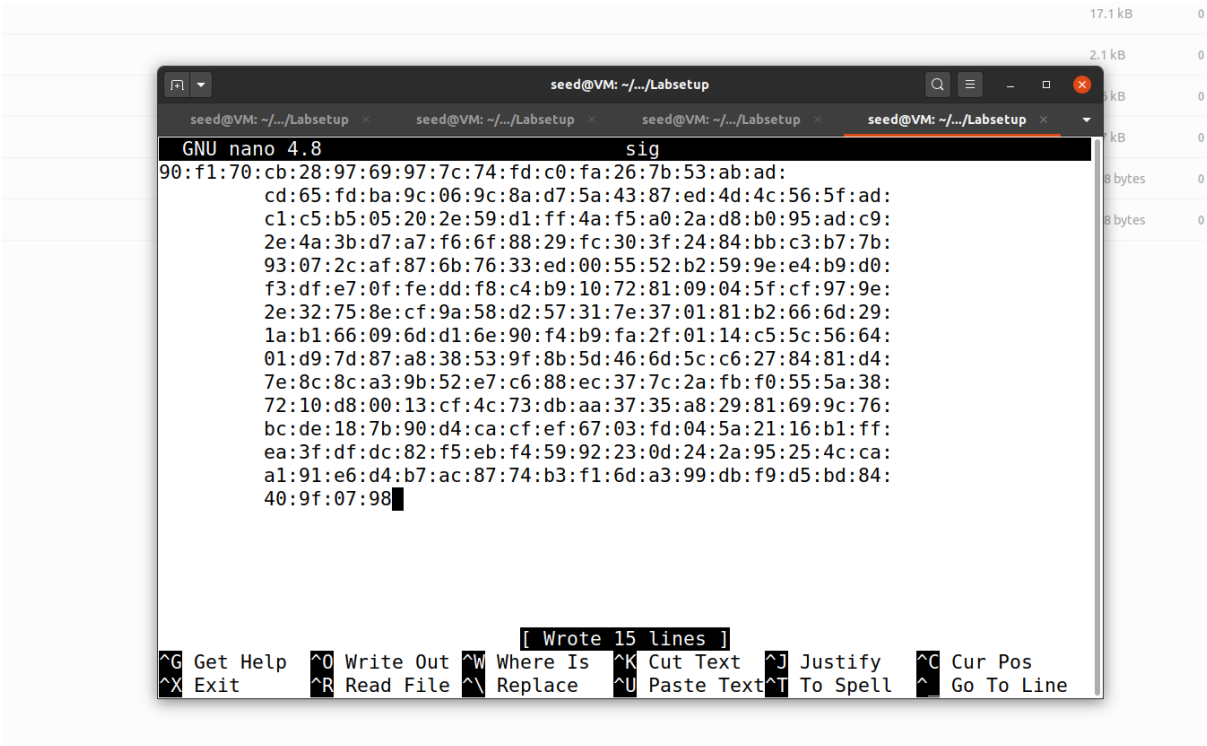
- openssl x509 -in c1.pem -text –noout


seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
F5144DCBA710F216EAAB22F031221161699026BA78D9971FE37D66AB75449573C8ACFFEF5D0A8A59
43E1ACB23A0FF348FCD76B37C163DCDE46D6DB45FE7D23FD90E851071E51A35FED4946547F2C88C5
F4139C97153C03E8A139DC690C32C1AF16574C9447427CA2C89C7DE6D44D54AF4299A8C104C2779C
D648E4CE11E02A8099F04370CF3F766BD14C49AB245EC20D82FD46A8AB6C93CC6252427592F89AFA
5E5EB2B061E51F1FB97F0998E83DFA837F4769A1
[03/27/24]seed@VM:~/.../Labsetup$ openssl x509 -in c1.pem -text -noout
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
0c:f5:bd:06:2b:56:02:f4:7a:b8:50:2c:23:cc:f0:66
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert G
lobal Root G2
Validity
Not Before: Mar 30 00:00:00 2021 GMT
Not After : Mar 29 23:59:59 2031 GMT
Subject: C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA25
6 2020 CA1
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public-Key: (2048 bit)
Modulus:
00:cc:f7:10:62:4f:a6:bb:63:6f:ed:90:52:56:c5:

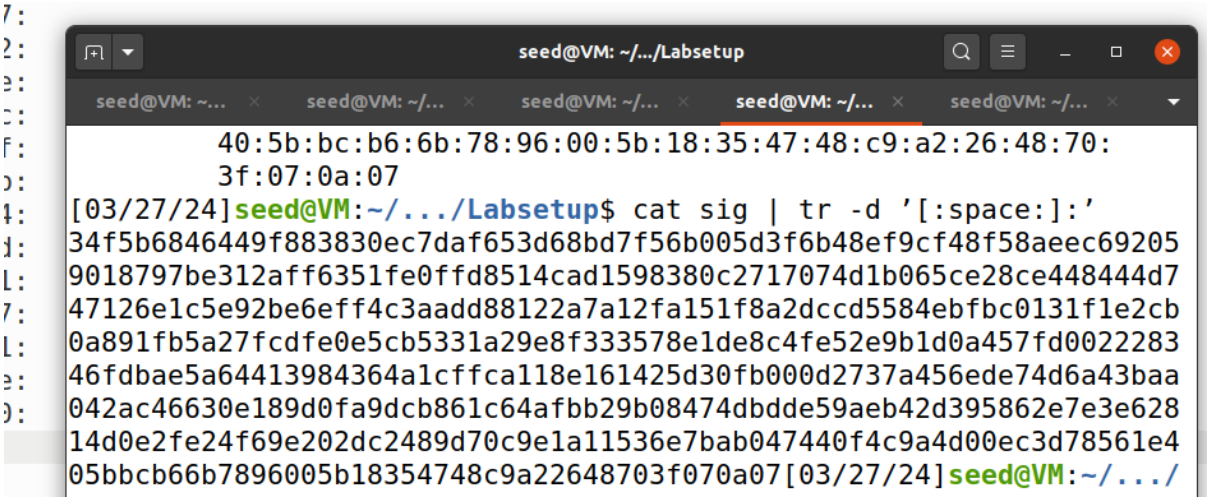

seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
6 2020 CA1
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public-Key: (2048 bit)
Modulus:
00:cc:f7:10:62:4f:a6:bb:63:6f:ed:90:52:56:c5:
6d:27:7b:7a:12:56:8a:f1:f4:f9:d6:e7:e1:8f:bd:
95:ab:f2:60:41:15:70:db:12:00:fa:27:0a:b5:57:
38:5b:7d:b2:51:93:71:95:0e:6a:41:94:5b:35:1b:
fa:7b:fa:bb:c5:be:24:30:fe:56:ef:c4:f3:7d:97:
e3:14:f5:14:4d:cb:a7:10:f2:16:ea:ab:22:f0:31:
22:11:61:69:90:26:ba:78:d9:97:1f:e3:7d:66:ab:
75:44:95:73:c8:ac:ff:ef:5d:0a:8a:59:43:e1:ac:
b2:3a:0f:f3:48:fc:d7:6b:37:c1:63:dc:de:46:d6:
db:45:fe:7d:23:fd:90:e8:51:07:1e:51:a3:5f:ed:
49:46:54:7f:2c:88:c5:f4:13:9c:97:15:3c:03:e8:
a1:39:dc:69:0c:32:c1:af:16:57:4c:94:47:42:7c:
a2:c8:9c:7d:e6:d4:4d:54:af:42:99:a8:c1:04:c2:
77:9c:d6:48:e4:ce:11:e0:2a:80:99:f0:43:70:cf:
3f:76:6b:d1:4c:49:ab:24:5e:c2:0d:82:fd:46:a8:
ab:6c:93:cc:62:52:42:75:92:f8:9a:fa:5e:5e:b2:
b0:61:e5:1f:1f:b9:7f:09:98:e8:3d:fa:83:7f:47:
69:a1
Exponent: 65537 (0x10001)



Step 3: Extract the signature from the server's certificate.

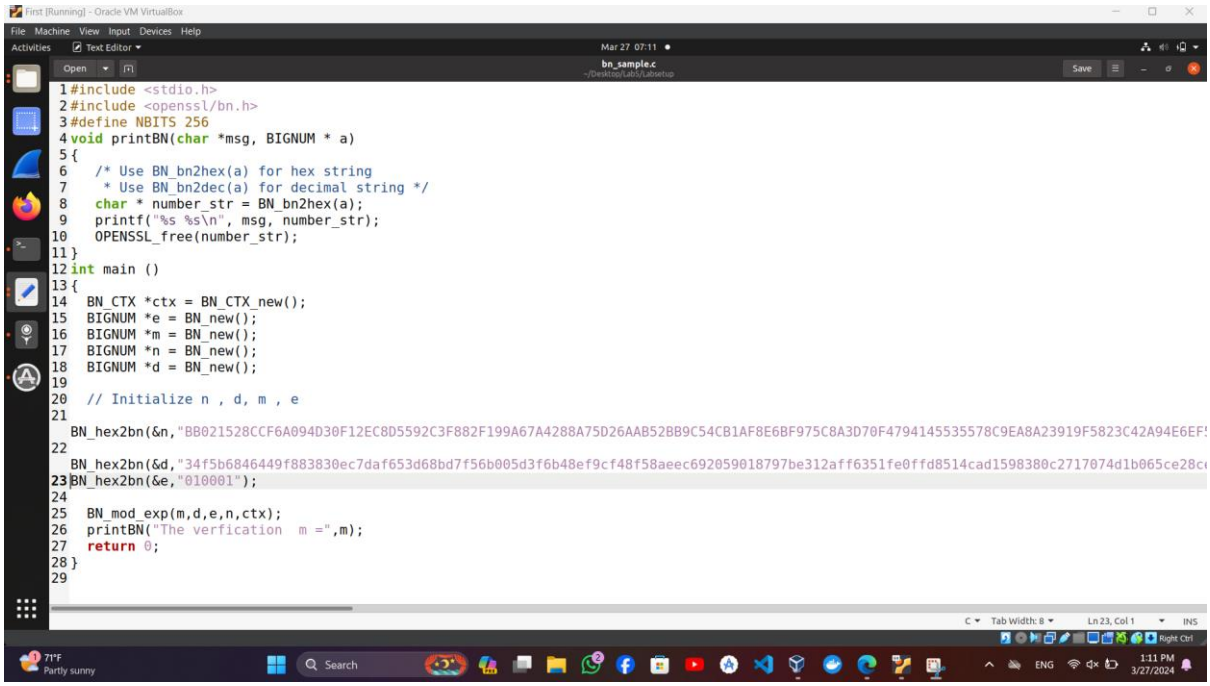


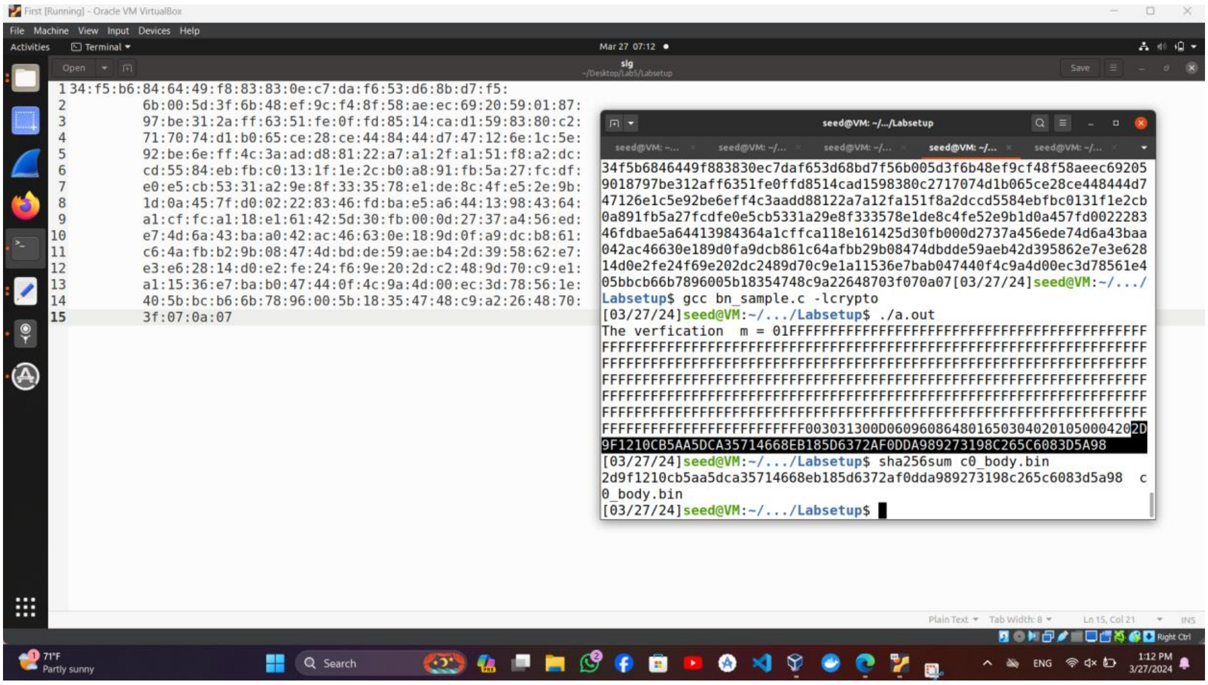
- **cat signature | tr -d '[:space:]:'**



Step 4: Extract the body of the server's certificate.

Step 5: Verify the signature.





We notice that the last part of the message as the hash of the signature.