



BIRZEIT UNIVERSITY

Faculty of Information Technology

Computer Systems Engineering Department

APPLIED CRYPTOGRAPHY

ENCS4320

RSA Public-Key Encryption and Signature Lab

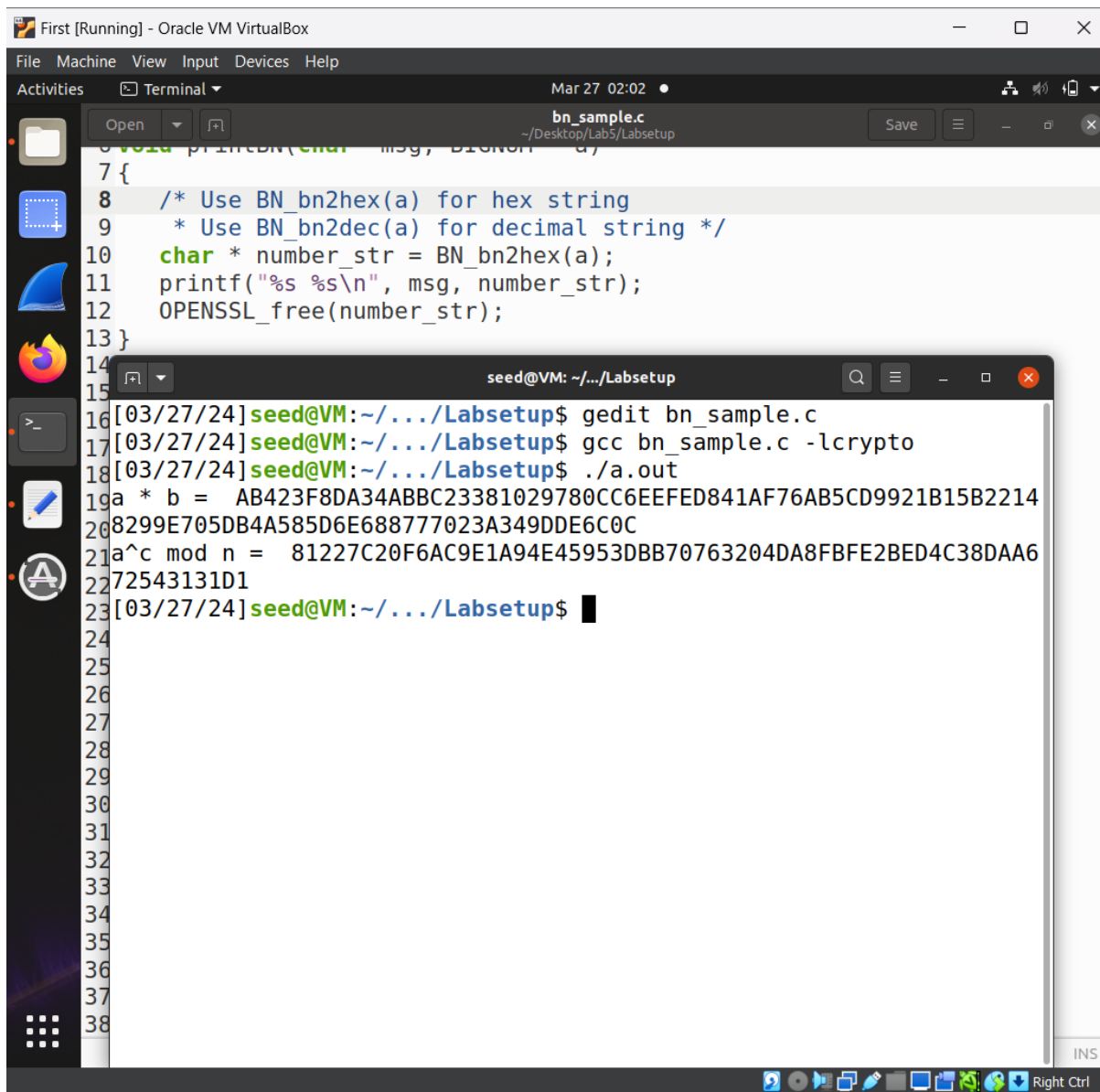
Prepared by: Dana Bornata – 1200284

Instructor: Dr. Ibrahim Nemer

Teacher Assistant: **Mohammed Balawi**

Section: 1

1.A Complete Example:



The screenshot shows an Oracle VM VirtualBox window titled "First [Running] - Oracle VM VirtualBox". Inside the VM, a terminal window is open with the following commands and output:

```
[03/27/24] seed@VM: ~/.../Labsetup$ gedit bn_sample.c
[03/27/24] seed@VM: ~/.../Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24] seed@VM: ~/.../Labsetup$ ./a.out
a * b = AB423F8DA34ABBC23381029780CC6EEFED841AF76AB5CD9921B15B2214
8299E705DB4A585D6E688777023A349DDE6C0C
a^c mod n = 81227C20F6AC9E1A94E45953DBB70763204DA8FBFE2BED4C38DAA6
72543131D1
[03/27/24] seed@VM: ~/.../Labsetup$
```

The C code in the background is as follows:

```
7 {
8     /* Use BN_bn2hex(a) for hex string
9     * Use BN_bn2dec(a) for decimal string */
10    char * number_str = BN_bn2hex(a);
11    printf("%s %s\n", msg, number_str);
12    OPENSSL_free(number_str);
13 }
```

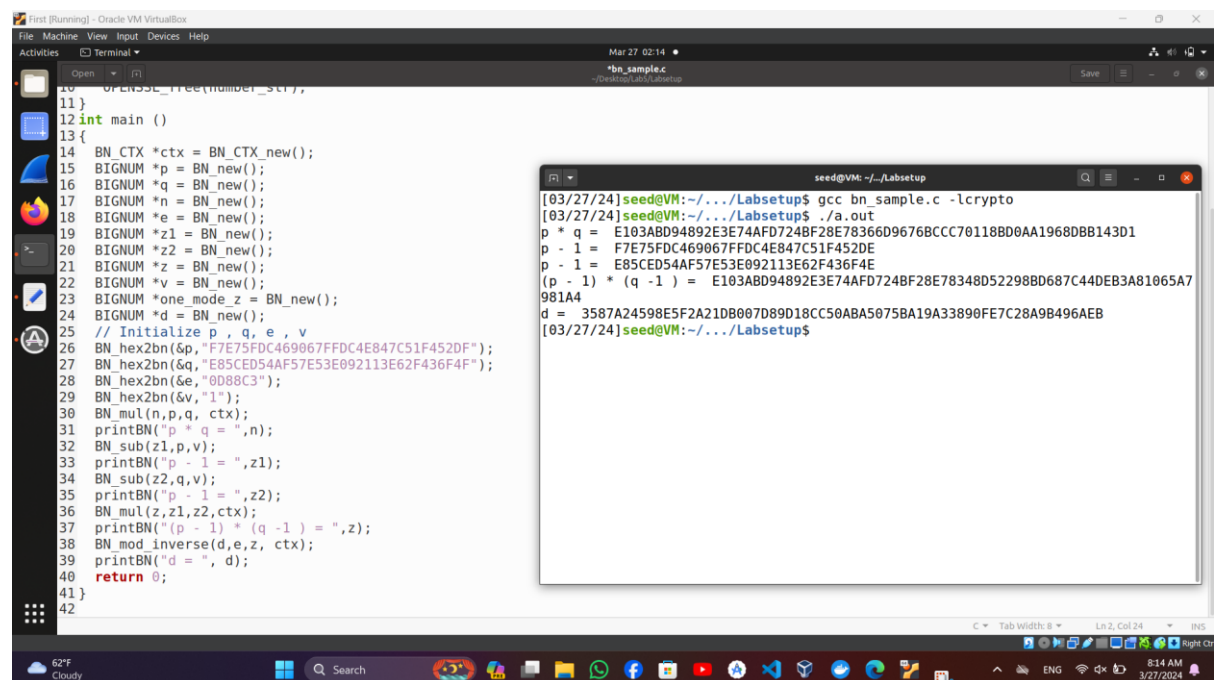
Task 1: Deriving the Private Key

3.1 Task 1: Deriving the Private Key

Let p , q , and e be three prime numbers. Let $n = p \cdot q$. 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
```

$$ed = 1 \bmod (p-1)(q-1)$$



The screenshot shows a terminal window with the following content:

```
bn_sample.c
11 }
12 int main ()
13 {
14     BN_CTX *ctx = BN_CTX_new();
15     BIGNUM *p = BN_new();
16     BIGNUM *q = BN_new();
17     BIGNUM *n = BN_new();
18     BIGNUM *e = BN_new();
19     BIGNUM *z1 = BN_new();
20     BIGNUM *z2 = BN_new();
21     BIGNUM *v = BN_new();
22     BIGNUM *one_mod_z = BN_new();
23     BIGNUM *d = BN_new();
24     // Initialize p, q, e, v
25     BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
26     BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
27     BN_hex2bn(&e, "0D88C3");
28     BN_hex2bn(&v, "1");
29     BN_mul(n, p, q, ctx);
30     printBN("p * q = ", n);
31     BN_sub(z1, p, v);
32     printBN("p - 1 = ", z1);
33     BN_sub(z2, q, v);
34     printBN("q - 1 = ", z2);
35     BN_mul(z1, z1, z2, ctx);
36     printBN("(p - 1) * (q - 1) = ", z1);
37     BN_mod_inverse(d, e, z1, ctx);
38     printBN("d = ", d);
39     return 0;
40 }
41 }
42 }
```

The terminal output shows the execution of the program:

```
[03/27/24]seed@VM: ~/Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24]seed@VM: ~/Labsetup$ ./a.out
p * q = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118B00AA1968DBB143D1
p - 1 = F7E75FDC469067FFDC4E847C51F452DE
q - 1 = E85CED54AF57E53E092113E62F436F4E
(p - 1) * (q - 1) = E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
d = 3587A24598E5F2A21D8007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[03/27/24]seed@VM: ~/Labsetup$
```

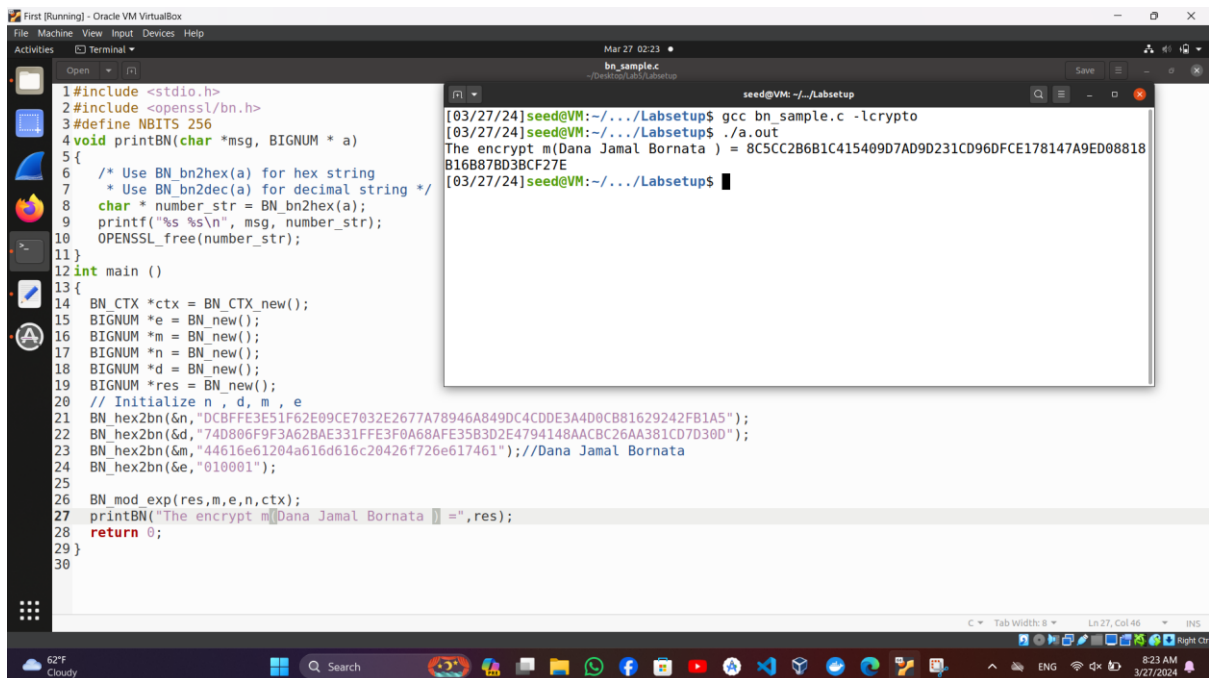
Task 2: Encrypting a message

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
```

$$C = m^e \bmod n$$



```
1#include <stdio.h>
2#include <openssl/bn.h>
3#define NBITS 256
4void printBN(char *msg, BIGNUM *a)
5{
6    /* Use BN_bn2hex(a) for hex string
7     * Use BN_bn2dec(a) for decimal string */
8    char *number_str = BN_bn2hex(a);
9    printf("%s %s\n", msg, number_str);
10   OPENSSL_free(number_str);
11}
12int main ()
13{
14   BN_CTX *ctx = BN_CTX_new();
15   BIGNUM *e = BN_new();
16   BIGNUM *m = BN_new();
17   BIGNUM *n = BN_new();
18   BIGNUM *d = BN_new();
19   BIGNUM *res = BN_new();
20   // Initialize n, d, m, e
21   BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDD3A4D0CB81629242FB1A5");
22   BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
23   BN_hex2bn(&m, "44616e61204a616d616c20426f726e617461"); //Dana Jamal Bornata
24   BN_hex2bn(&e, "010001");
25
26   BN_mod_exp(res, m, e, n, ctx);
27   printBN("The encrypt m[Dana Jamal Bornata] =", res);
28   return 0;
29}
30
```

```
seed@VM: ~/Labsetup
[03/27/24]seed@VM:~/Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24]seed@VM:~/Labsetup$ ./a.out
The encrypt m(Dana Jamal Bornata ) = 8C5CC2B6B1C415409D7AD9D231CD96DFCE178147A9ED08818
B16B87BD3BCF27E
[03/27/24]seed@VM:~/Labsetup$
```

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

```
First [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
bn_sample.c
~/Desktop/Lab5/Labsetup
seed@VM: ~/.../Labsetup
[03/27/24]seed@VM:~/.../Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24]seed@VM:~/.../Labsetup$ ./a.out
The encrypt m(Dana Jamal Bornata ) = 8C5CC2B6B1C415409D7AD9D231CD96DFCE178147A9ED08818B16B87BD3BCF27E
[03/27/24]seed@VM:~/.../Labsetup$
```

```
1#include <stdio.h>
2#include <openssl/bn.h>
3#define NBITS 256
4void printBN(char *msg, BIGNUM * a)
5{
6    /* Use BN_bn2hex(a) for hex string
7     * Use BN_bn2dec(a) for decimal string */
8    char * number_str = BN_bn2hex(a);
9    printf("%s %s\n", msg, number_str);
10    OPENSSL_free(number_str);
11}
12int main ()
13{
14    BN_CTX *ctx = BN_CTX_new();
15    BIGNUM *e = BN_new();
16    BIGNUM *m = BN_new();
17    BIGNUM *n = BN_new();
18    BIGNUM *d = BN_new();
19    BIGNUM *res = BN_new();
20    // Initialize n, d, m, e
21    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDD3A4D0CB81629242FB1A5");
22    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
23    BN_hex2bn(&m, "44616e61204a616d616c20426f726e617461"); //Dana Jamal Bornata
24    BN_hex2bn(&e, "010001");
25    BN_mod_exp(res,m,e,n,ctx);
26    printBN("The encrypt m(Dana Jamal Bornata ) =",res);
27    return 0;
28}
29
30
```

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 = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBDDFC7DCB67396567EA1E2493F
```

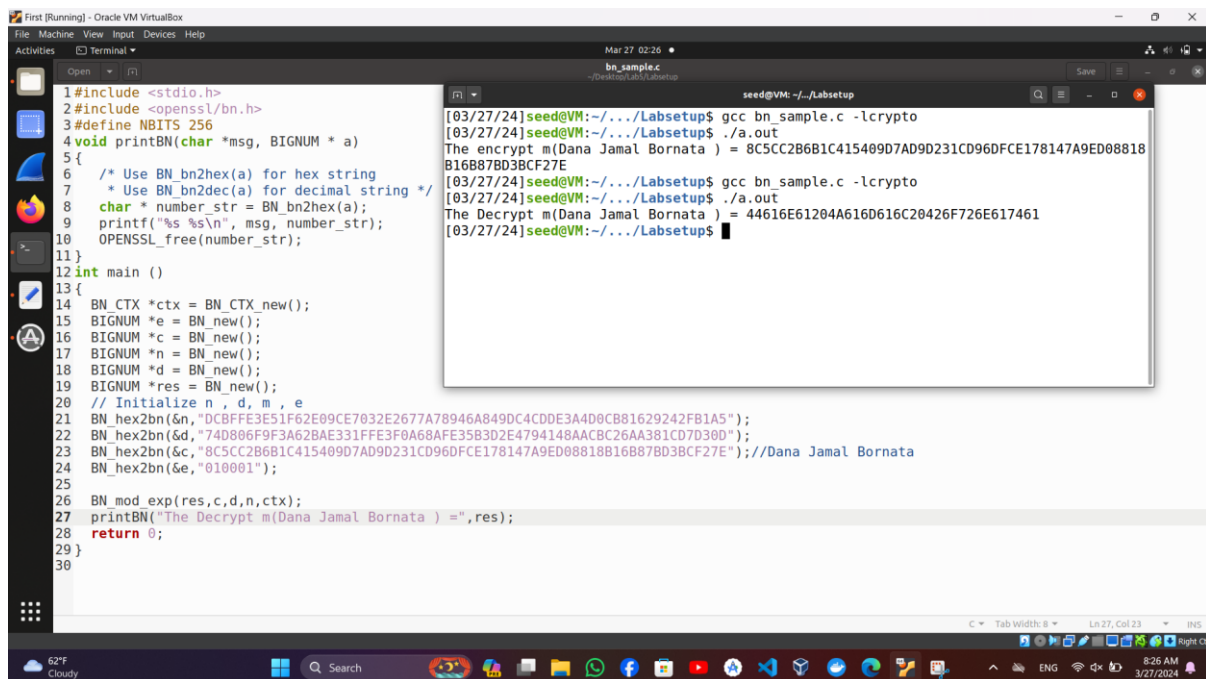
You can use the following python command to convert a hex string back to a plain ASCII string.

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

$m = c^e \bmod n$

C=

8C5CC2B6B1C415409D7AD9D231CD96DFCE178147A9ED08818B16B87BD3BCF27E

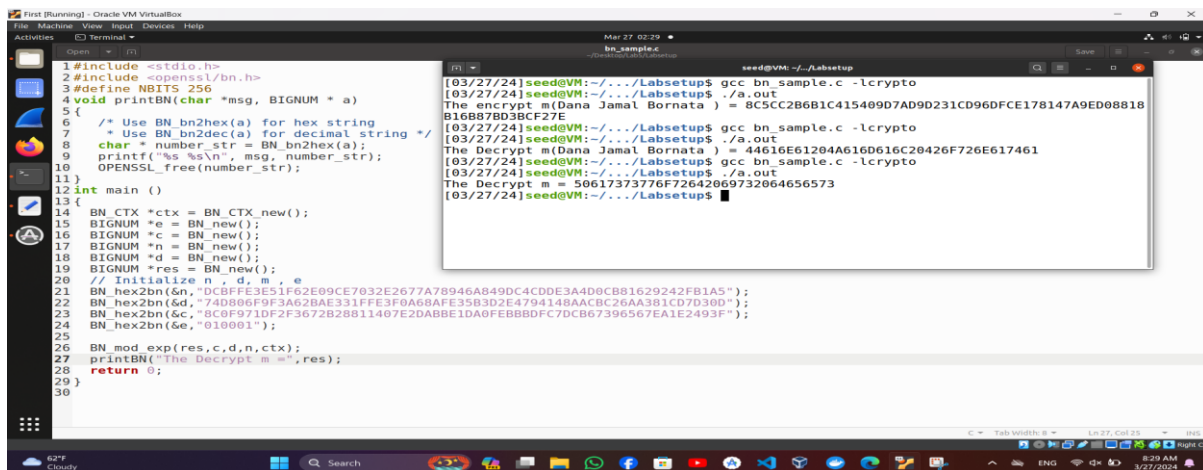


The Decrypt m(Dana Jamal Bornata) = 44616E61204A616D616C20426F726E617461

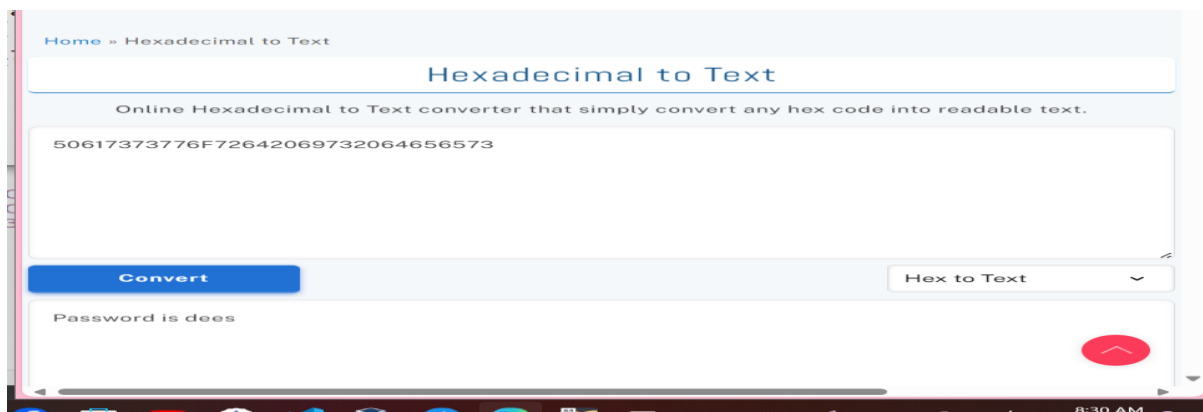


When the C is:

8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBD7C7DCB67396567EA1E2 493F



The Decrypt m = 50617373776F72642069732064656573



Task 4: Signing a Message

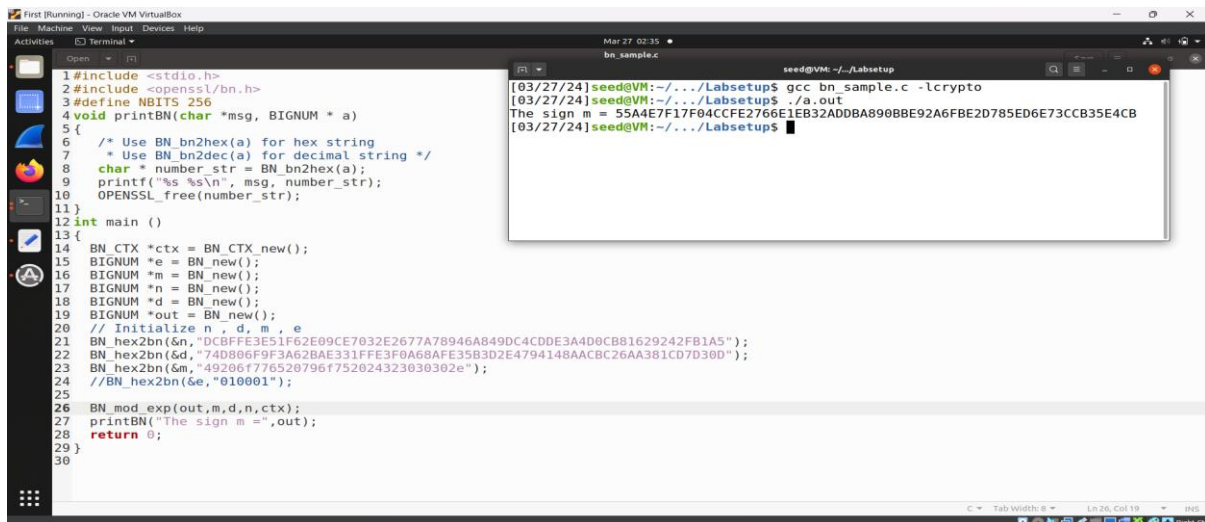
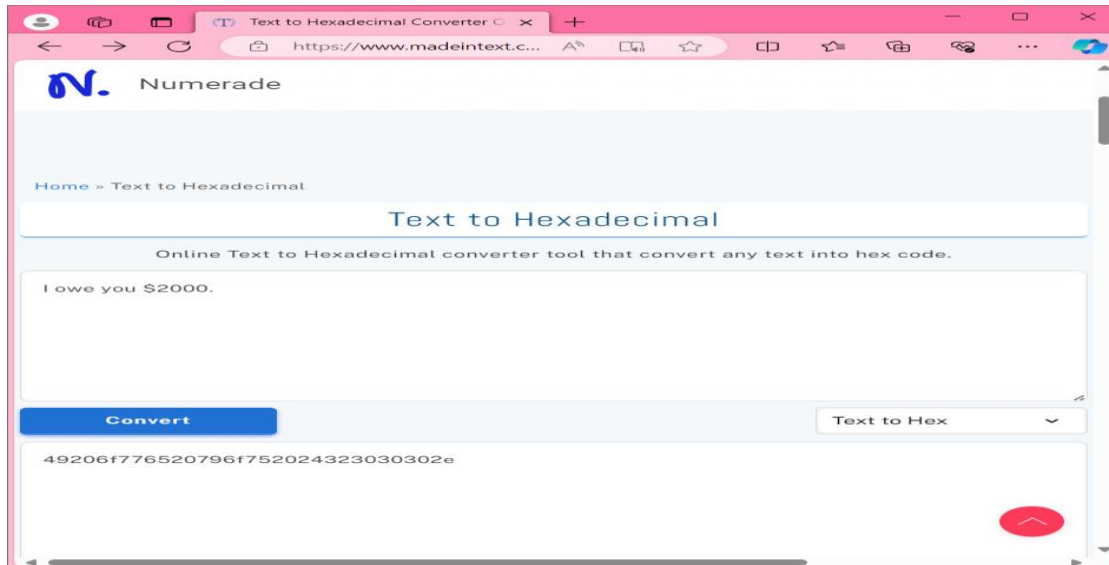
3.4 Task 4: Signing a Message

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):

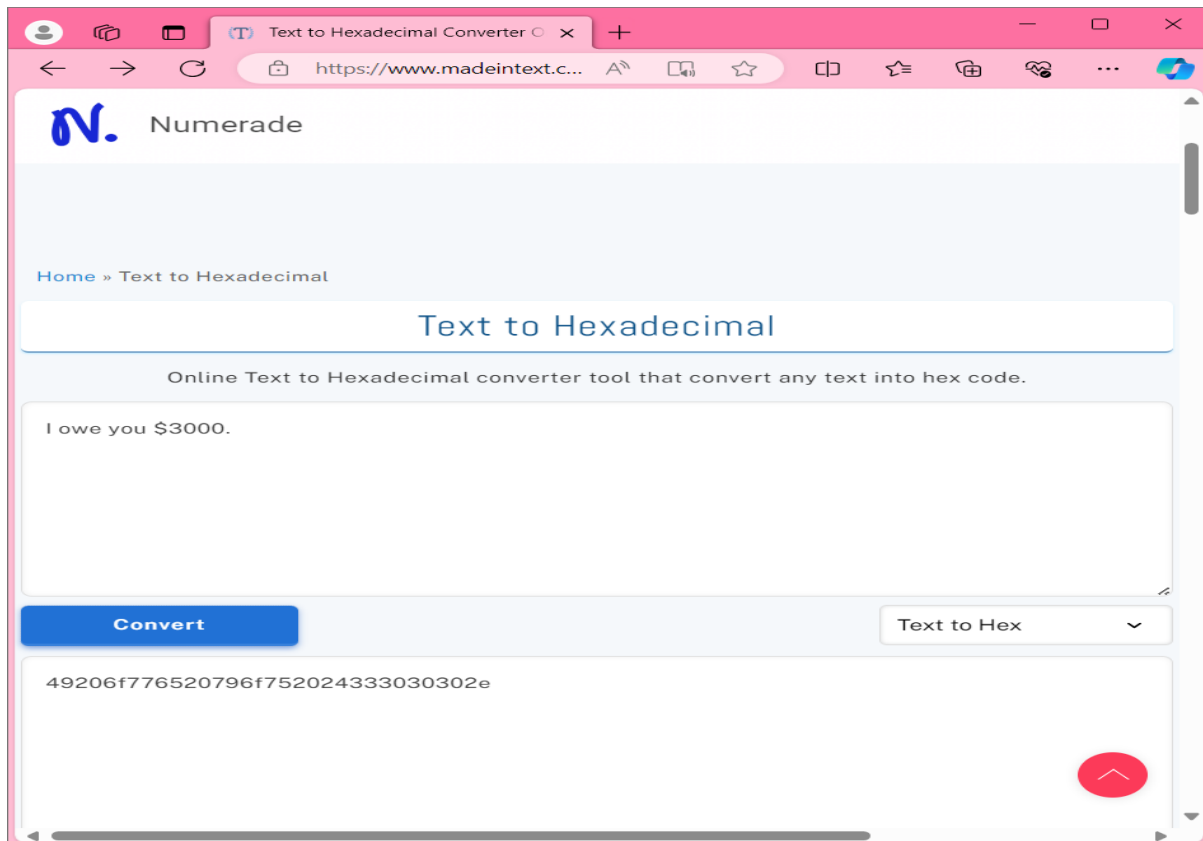
M = I owe you \$2000.

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.

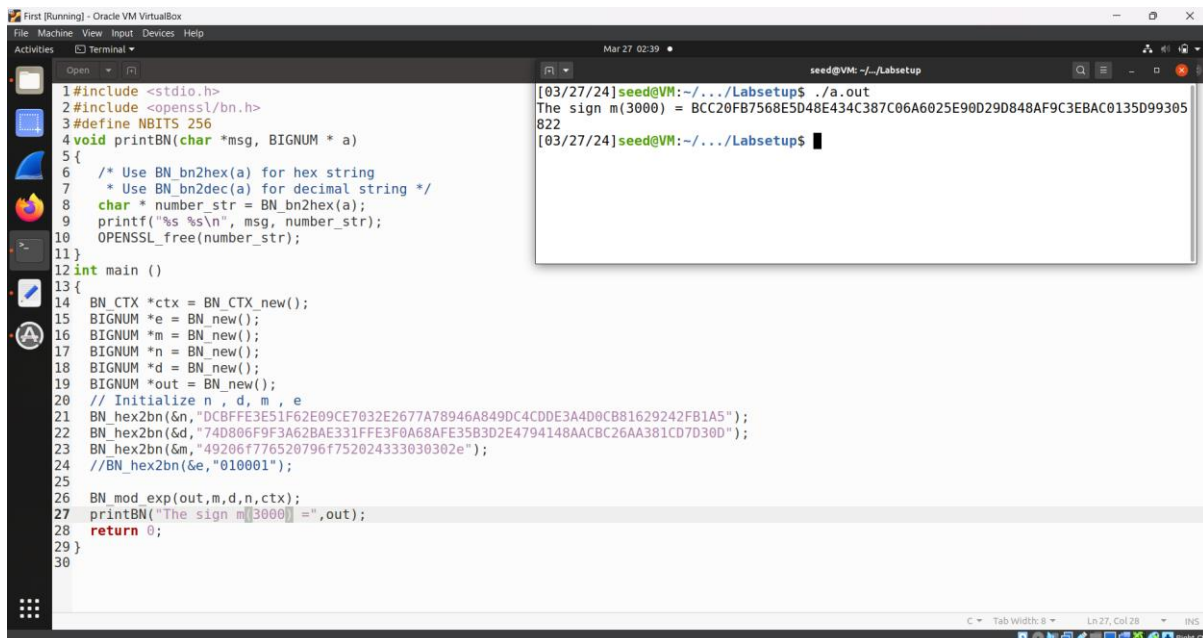
\$2000



\$3000



The sign m(3000) =
 BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822



Task 5: Verifying a Signature

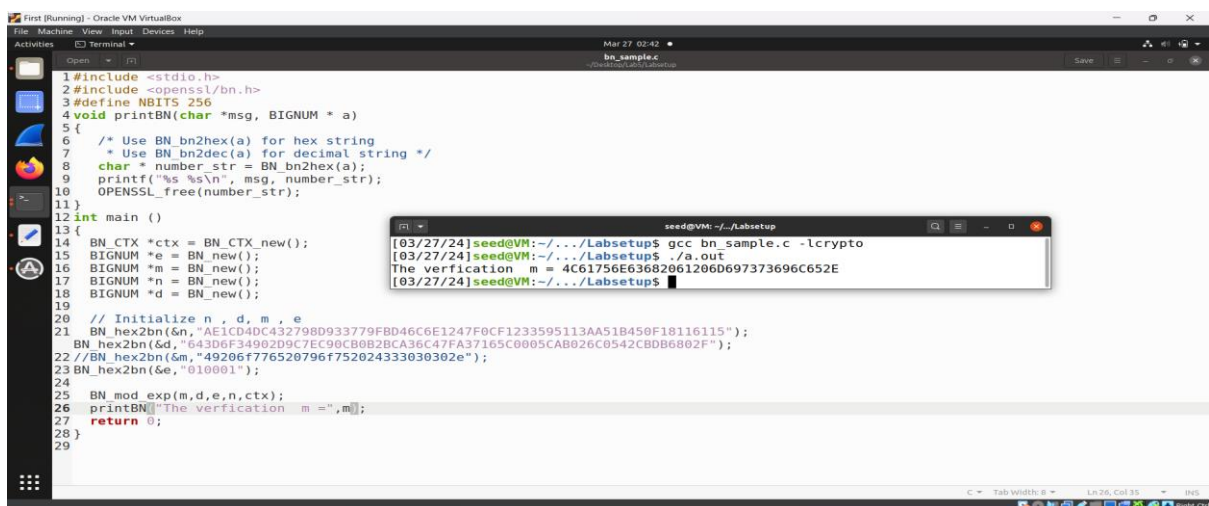
3.5 Task 5: Verifying a Signature

Bob receives a message $M = \text{"Launch a missile."}$ 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 = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CDBD6802F
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.

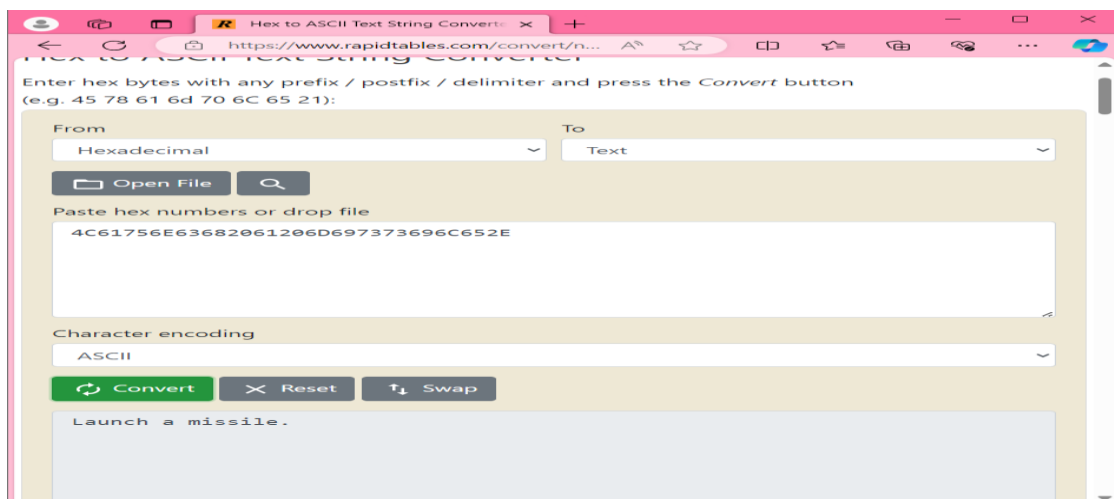
$M = S^e \bmod n$



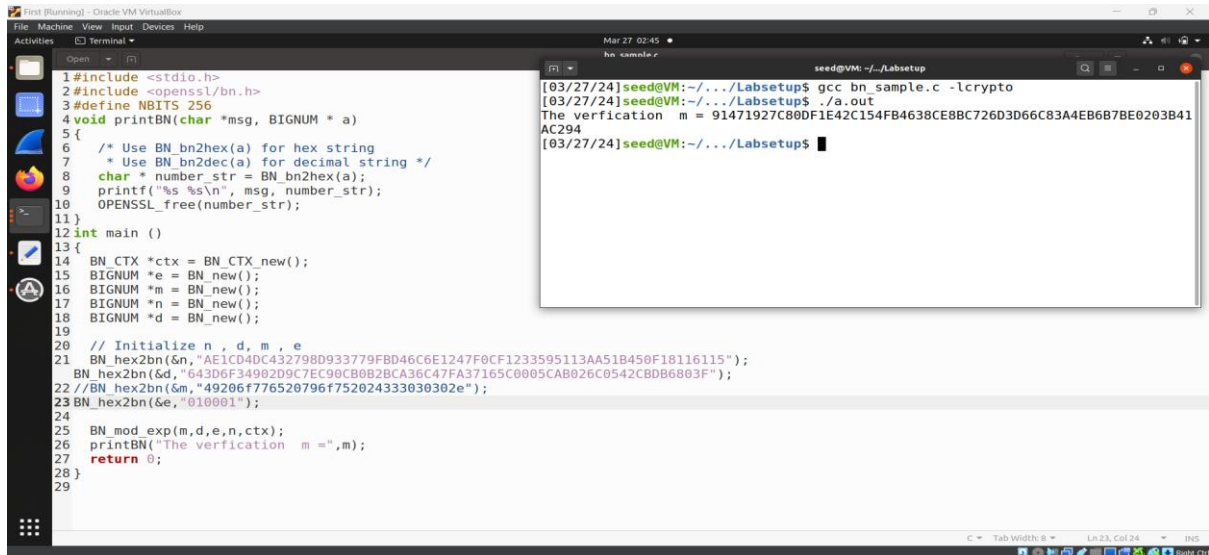
```
1#include <stdio.h>
2#include <openssl/bn.h>
3#define NBITS 256
4void printBN(char *msg, BIGNUM *a)
5{
6    /* Use BN_bn2hex(a) for hex string
7     * Use BN_bn2dec(a) for decimal string */
8    char *number_str = BN_bn2hex(a);
9    printf("%s %s\n", msg, number_str);
10   OPENSSL_free(number_str);
11}
12int main ()
13{
14   BN_CTX *ctx = BN_CTX_new();
15   BIGNUM *e = BN_new();
16   BIGNUM *m = BN_new();
17   BIGNUM *n = BN_new();
18   BIGNUM *d = BN_new();
19
20   // Initialize n, d, m, e
21   BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
22   BN_hex2bn(&d, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CDBD6802F");
23   BN_hex2bn(&m, "49206f776520796f752024333030302e");
24   BN_hex2bn(&e, "010001");
25   BN_mod_exp(m,d,e,n,ctx);
26   printBN("The verification m =",m);
27   return 0;
28}
29
```

```
seed@VM: ~/Labsetup
[03/27/24]seed@VM:~/Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24]seed@VM:~/Labsetup$ ./a.out
The verification m = 4C61756E63682061206D697373696C652E
[03/27/24]seed@VM:~/Labsetup$
```

The verification $m = 4C61756E63682061206D697373696C652E$



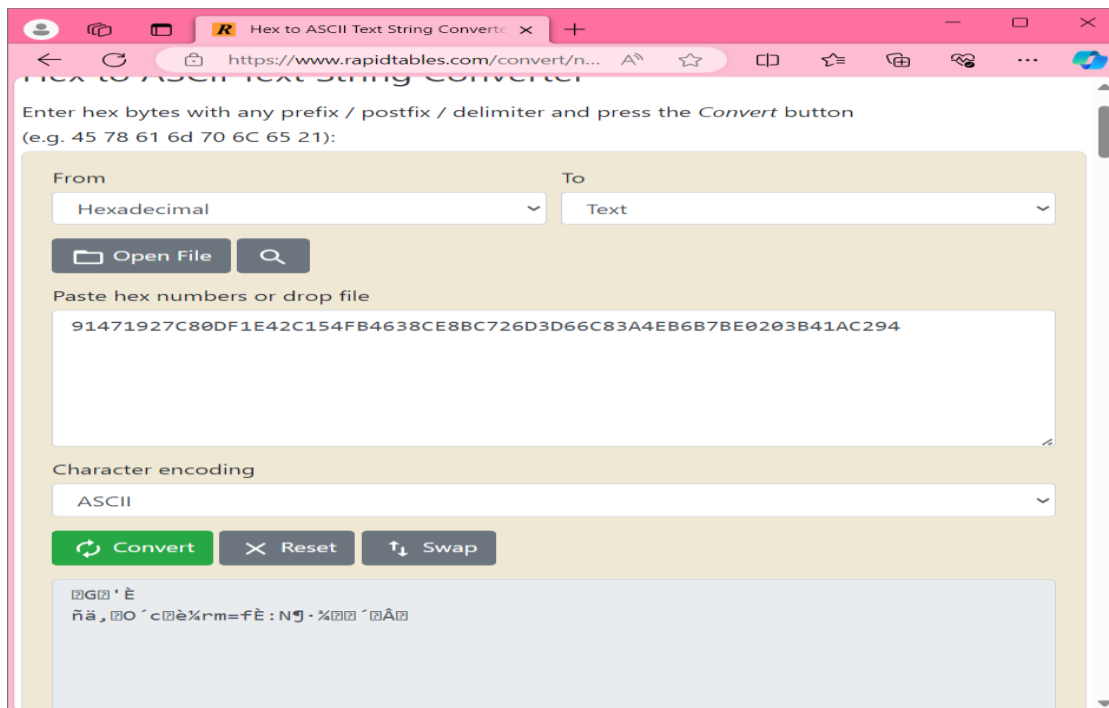
$2f \gg 3f$



```
1#include <stdio.h>
2#include <openssl/bn.h>
3#define NBITS 256
4void printBN(char *msg, BIGNUM * a)
5{
6    /* Use BN_bn2hex(a) for hex string
7     * Use BN_bn2dec(a) for decimal string */
8    char * number_str = BN_bn2hex(a);
9    printf("%s %s\n", msg, number_str);
10    OPENSSL_free(number_str);
11}
12int main ()
13{
14    BN_CTX *ctx = BN_CTX_new();
15    BIGNUM *e = BN_new();
16    BIGNUM *m = BN_new();
17    BIGNUM *n = BN_new();
18    BIGNUM *d = BN_new();
19
20    // Initialize n, d, m, e
21    BN_hex2bn(&n, "AE1CD40C432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
22    BN_hex2bn(&d, "643D6F34902D9C7EC90CB082BCA36C47FA37165C0005CAB026C0542CDBB6803F");
23    BN_hex2bn(&m, "49206f776520796f752024333030302e");
24
25    BN_mod_exp(m,d,e,n,ctx);
26    printBN("The verification m =",m);
27    return 0;
28}
29
```

```
[03/27/24]seed@VM:~/../Labsetup$ gcc bn_sample.c -lcrypto
[03/27/24]seed@VM:~/../Labsetup$ ./a.out
The verification m = 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
[03/27/24]seed@VM:~/../Labsetup$
```

The verification m =
91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294



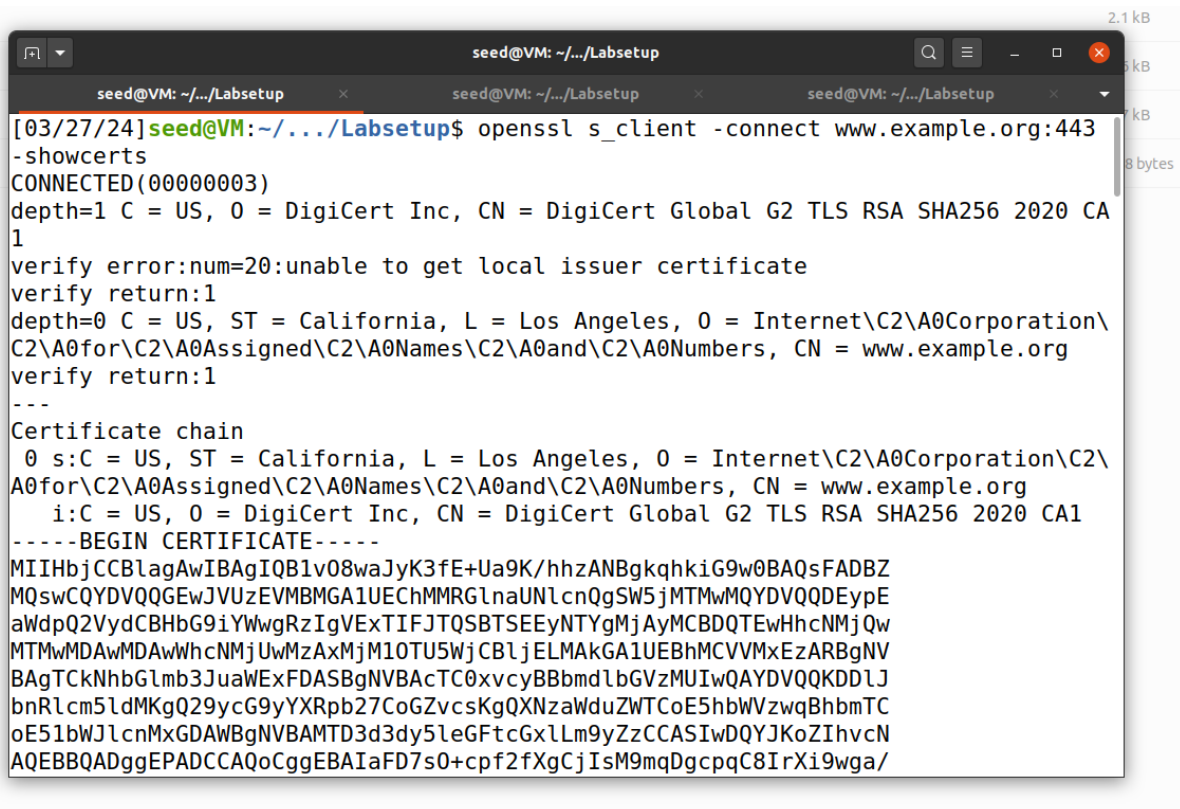
We can notice that changing just one byte of the signature gives us a totally different message.

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.



```
seed@VM: ~/.../Labsetup
[03/27/24] seed@VM: ~/.../Labsetup$ openssl s_client -connect www.example.org:443 -showcerts
CONNECTED(00000003)
depth=1 C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA256 2020 CA 1
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbers, CN = www.example.org
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbers, CN = www.example.org
  i:C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA256 2020 CA1
-----BEGIN CERTIFICATE-----
MIIHbjCCBlagAwIBAgIQB1v08waJyK3fE+Ua9K/hhzANBgkqhkiG9w0BAQsFADBZ
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMTMwMQYDVQQDEypE
aWdpQ2VydCBHbG9iYWwgRzIgVExTIFJTQSBTSEEyNTYgMjAyMCBDQTEwHhcNMjQw
MTMwMDAwMDAwWhcNMjUwMzAxMjM1OTU5WjCB1jELMAkGA1UEBhMCVVMxEzARBgNV
BAGTCkNhbgG1mb3JuaWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMUwQAYDVQQKDD1J
bnRlcm5ldMKgQ29ycG9yYXRpb27CoGZvczKgQXNzaWduZWtCoE5hbWVzwqBhbmTC
oE51bWJlcnMxGDAWBgNVBAMTD3d3dy5leGFtcGx1Lm9yZzCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAlaFD7s0+cpf2fXgCjIsM9mqDgcpcqC8IrXi9wga/
```

```
seed@VM: ~/.../Labsetup
GNU nano 4.8                                c0.pem
IBV5MkXVlyTf20AzA0a7d8x2H6XAHcA5tIxY0B3jMEQQ0bXcbn0wdJA9paEhvu6
hzId/R43jLAAAAGNW9L8XwAABAMASDBGAiEA4Koh/VizdQU1tjZ2E2VGgWSXXkwn
QmiYhmAeKcVLHeACIQD7JIGFsdGol7kss2pe4lYrCgPvc+iGZkuqnj26hqhr0TAN
BgkqhkiG9w0BAQsFAA0CAQEAB0FuAj4N4yNG900WNQWTNSICC4Rd4n0G1HRP/Bsn
rz7KrcPORTb6D+Jx+00amh031QhIvVBYs14gY4Ypyj7MzHgm4VmPXcqlvEkxb2G9
Qv9hYuEiNSQmm1fr5QAN/0AzbEbCM3cImLJ69kP5bUjfv/76KB57is8tYf9sh5ik
LGKauxCM/zRiCga3bXLDafk5S2g5Vr2hs230d/NGW1wZrE+zdGuMxfGJzJP+DAFv
iBfcQnFg4+1zMEKcqS87oni0yG+60RMM0MdejBD7AS43m9us96Gsun/4kufLQUTI
FfnzxLutUV++3seshgefQ0y5C/ayi8y1VTNmujPCxPCi6Q==
-----END CERTIFICATE-----

[ Wrote 42 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line
```

```
seed@VM: ~/.../Labsetup
GNU nano 4.8                                c1.pem
NGh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9EaWdpQ2VydEdsb2JhbFJvb3RH
Mi5jc nQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybmDMuZGlnaW NlcnQuY29t
L0RpZ2lDZXJ0R2xvYmFsUm9vdEcyLmNy bDA9BgNVHSAENjA0MA sGCWGSAGG/WwC
ATAHBgVngQwBATAIBgZngQwBAGewCAYGZ4EMAQICMAgGBmeBDAECAzANBgkqhkiG
9w0BAQsFAA0CAQEAKPFwyyiXaZd8dP3A+iZ7U6utzWX9upwGnIrXWk0H7U1MvL+t
wcW1BSAuWdH/SvWgKtiwLa3JLko716f2b4gp/DA/JIS7w7d7kwcsr4drdjPtAFVS
slme5LnQ89/nD/7d+MS5EHKBCQRfz5eelJj1js+aWNJXMX43AYGyZm0pGrFmCW3R
bpD0ufovARTFXFZkAdl9h6g4U5+LXUZtXMYnhIHUfoym05tS58aI7Dd8KvvwVVo4
chDYABPPTHpbqjclqCmBaZx2vN4Ye5DUys/vZwP9BFohFrH/6j/f3IL16/RZkiMN
JCqVJUzKoZhm1Lesh3Sz8W2jmdv51b2EQJ8HmA==
-----END CERTIFICATE-----

[ Wrote 28 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line
```

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

```
1:87:
0:c2:
c:5e:
2:dc:
c:df:
e:9b:
3:64:
6:ed:
8:61:
2:e7:
9:e1:
6:1e:
8:70:
[03/27/24] seed@VM: ~/.../Labsetup$ openssl x509 -in c1.pem -noout -m
odulus
Modulus=BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D26AAB52
BB9C54CB1AF8E6BF975C8A3D70F4794145535578C9EA8A23919F5823C42A94E6EF5
3BC32EDB8DC0B05CF35938E7EDCF69F05A0B1BBEC094242587FA3771B313E71CACE
19BEFDBE43B45524596A9C153CE34C852EEB5AEED8FDE6070E2A554ABB66D0E97A5
40346B2BD3BC66EB66347CFA6B8B8F572999F830175DBA726FFB81C5ADD286583D1
7C7E709BBF12BF786DCC1DA715DD446E3CCAD25C188BC60677566B3F118F7A25CE6
53FF3A88B647A5FF1318EA9809773F9D53F9CF01E5F5A6701714AF63A4FF99B3939
DDC53A706FE48851DA169AE2575BB13CC5203F5ED51A18BDB15
```

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


```

seed@VM: ~/.../Labsetup
F5144DCBA710F216EAB22F031221161699026BA78D9971FE37D66AB75449573C8ACFFEF5D0A8A59
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
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)

```

```
seed@VM: ~/.../Labsetup
X509v3 Certificate Policies:
  Policy: 2.16.840.1.114412.2.1
  Policy: 2.23.140.1.1
  Policy: 2.23.140.1.2.1
  Policy: 2.23.140.1.2.2
  Policy: 2.23.140.1.2.3

Signature Algorithm: sha256WithRSAEncryption
90:f1:70:cb:28:97:69:97:7c:74:fd:c0:fa:26:7b:53:ab:ad:
cd:65:fd:ba:9c:06:9c:8a:d7:5a:43:87:ed:4d:4c:56:5f:ad:
c1:c5:b5:05:20:2e:59:d1:ff:4a:f5:a0:2a:d8:b0:95:ad:c9:
2e:4a:3b:d7:a7:f6:6f:88:29:fc:30:3f:24:84:bb:c3:b7:7b:
93:07:2c:af:87:6b:76:33:ed:00:55:52:b2:59:9e:e4:b9:d0:
f3:df:e7:0f:fe:dd:f8:c4:b9:10:72:81:09:04:5f:cf:97:9e:
2e:32:75:8e:cf:9a:58:d2:57:31:7e:37:01:81:b2:66:6d:29:
1a:b1:66:09:6d:d1:6e:90:f4:b9:fa:2f:01:14:c5:5c:56:64:
01:d9:7d:87:a8:38:53:9f:8b:5d:46:6d:5c:c6:27:84:81:d4:
7e:8c:8c:a3:9b:52:e7:c6:88:ec:37:7c:2a:fb:f0:55:5a:38:
72:10:d8:00:13:cf:4c:73:db:aa:37:35:a8:29:81:69:9c:76:
bc:de:18:7b:90:d4:ca:cf:ef:67:03:fd:04:5a:21:16:b1:ff:
ea:3f:df:dc:82:f5:eb:f4:59:92:23:0d:24:2a:95:25:4c:ca:
a1:91:e6:d4:b7:ac:87:74:b3:f1:6d:a3:99:db:f9:d5:bd:84:
40:9f:07:98
[03/27/24] seed@VM: ~/.../Labsetup$
```

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

```
seed@VM: ~/.../Labsetup
GNU nano 4.8 sig
90:f1:70:cb:28:97:69:97:7c:74:fd:c0:fa:26:7b:53:ab:ad:
cd:65:fd:ba:9c:06:9c:8a:d7:5a:43:87:ed:4d:4c:56:5f:ad:
c1:c5:b5:05:20:2e:59:d1:ff:4a:f5:a0:2a:d8:b0:95:ad:c9:
2e:4a:3b:d7:a7:f6:6f:88:29:fc:30:3f:24:84:bb:c3:b7:7b:
93:07:2c:af:87:6b:76:33:ed:00:55:52:b2:59:9e:e4:b9:d0:
f3:df:e7:0f:fe:dd:f8:c4:b9:10:72:81:09:04:5f:cf:97:9e:
2e:32:75:8e:cf:9a:58:d2:57:31:7e:37:01:81:b2:66:6d:29:
1a:b1:66:09:6d:d1:6e:90:f4:b9:fa:2f:01:14:c5:5c:56:64:
01:d9:7d:87:a8:38:53:9f:8b:5d:46:6d:5c:c6:27:84:81:d4:
7e:8c:8c:a3:9b:52:e7:c6:88:ec:37:7c:2a:fb:f0:55:5a:38:
72:10:d8:00:13:cf:4c:73:db:aa:37:35:a8:29:81:69:9c:76:
bc:de:18:7b:90:d4:ca:cf:ef:67:03:fd:04:5a:21:16:b1:ff:
ea:3f:df:dc:82:f5:eb:f4:59:92:23:0d:24:2a:95:25:4c:ca:
a1:91:e6:d4:b7:ac:87:74:b3:f1:6d:a3:99:db:f9:d5:bd:84:
40:9f:07:98

[ Wrote 15 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^_ Replace   ^U Paste Text ^T To Spell   ^_ Go To Line
```

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


```

7:
2:
3:
4:
f:
5:
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
779:
780:
781:
782:
783:
784:
785:
786:
787:
788:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
979:
980:
981:
982:
983:
984:
985:
986:
987:
988:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
1000:

```

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

Step 5: Verify the signature.

```

1#include <stdio.h>
2#include <openssl/bn.h>
3#define NBITS 256
4void printBN(char *msg, BIGNUM *a)
5{
6    /* Use BN_bn2hex(a) for hex string
7     * Use BN_bn2dec(a) for decimal string */
8    char *number_str = BN_bn2hex(a);
9    printf("%s %s\n", msg, number_str);
10   OPENSSL_free(number_str);
11}
12int main ()
13{
14   BN_CTX *ctx = BN_CTX_new();
15   BIGNUM *e = BN_new();
16   BIGNUM *m = BN_new();
17   BIGNUM *n = BN_new();
18   BIGNUM *d = BN_new();
19
20   // Initialize n , d , m , e
21   BN_hex2bn(&n, "BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D26AAB52BB9C54CB1AF8E6BF975C8A3D70F4794145535578C9EA8A23919F5823C42A94E6EF!
22   BN_hex2bn(&d, "34f5b6846449f883830ec7daf653d68bd7f56b005d3f6b48ef9cf48f58aeec692059018797be312aff6351fe0ffd8514cad1598380c2717074d1b065ce28ce448444d7
23   BN_hex2bn(&e, "010001");
24
25   BN_mod_exp(m,d,e,n,ctx);
26   printBN("The verification m =",m);
27   return 0;
28}
29

```

