-Group 5
Computer Cryptography
Vlad Smirnov, Ryan Gumbayan, Christian Fecteau

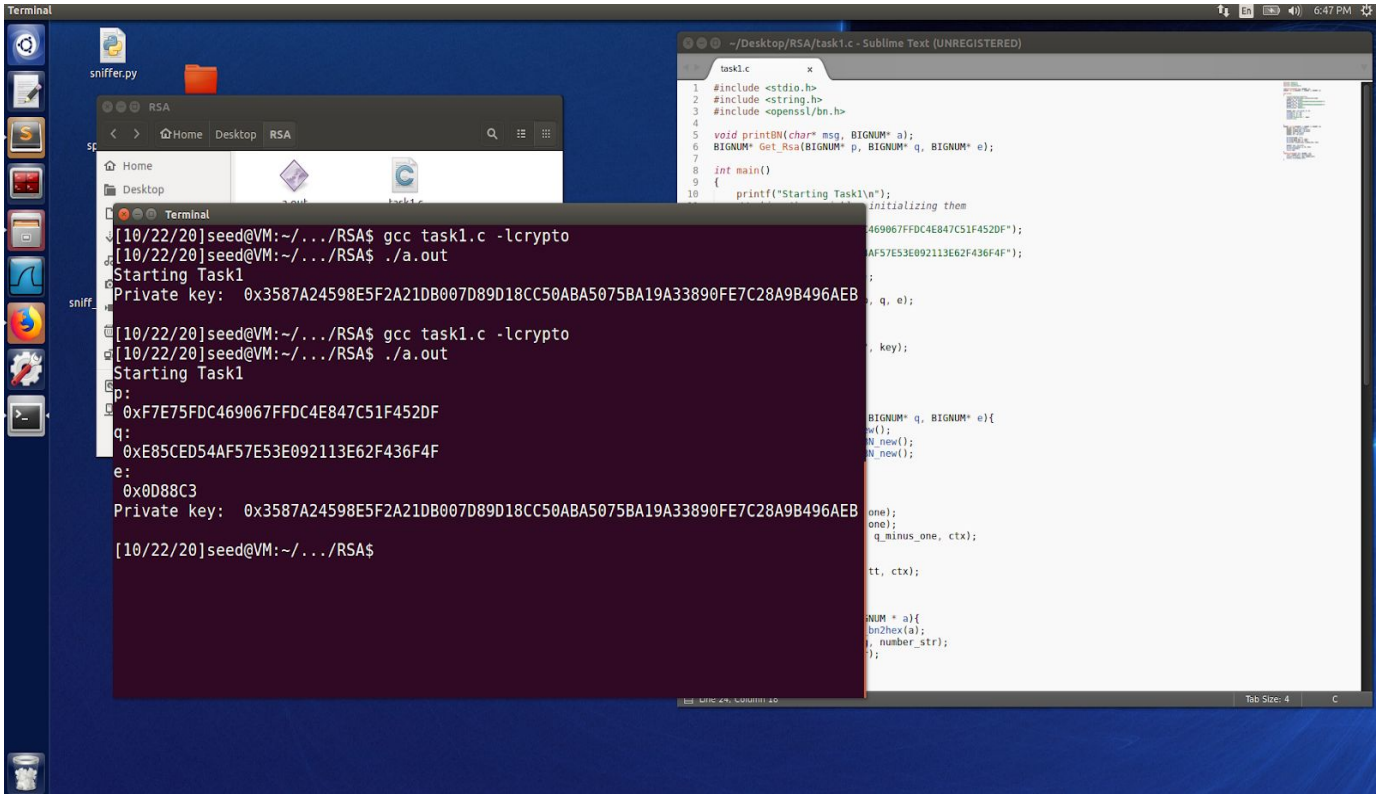
RSA LAB

Task 1:Deriving the Private Key

We are given

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

And we are supposed to make a program similar to the complete example in the lab guide to calculate what the private key is. The open ssl library and BIGNUM are used to do all of the math and algorithms behind the RSA encryption/decryption algorithm



Task 2:Encrypting a Message

In task two we are required to encrypt a message “a top secret” and the keys and private key are given to us to make sure we did it right.

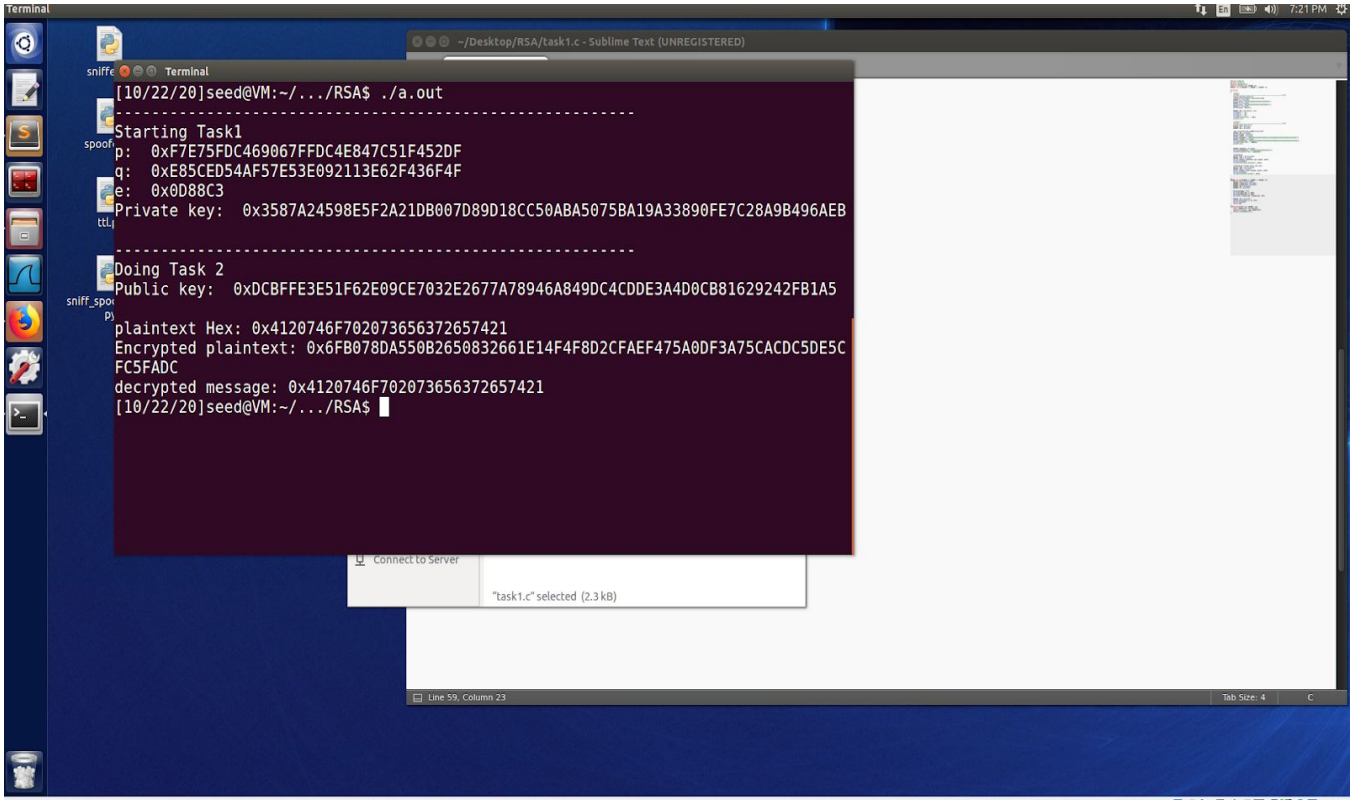
n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

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

M = A top secret!

d=74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

Basically we printed message in hex, encrypted the message and then printed the encrypted plaintext in hex and then decrypted and printed in hex again.You can see how the message before and after encryption/decryption is the same in hex.

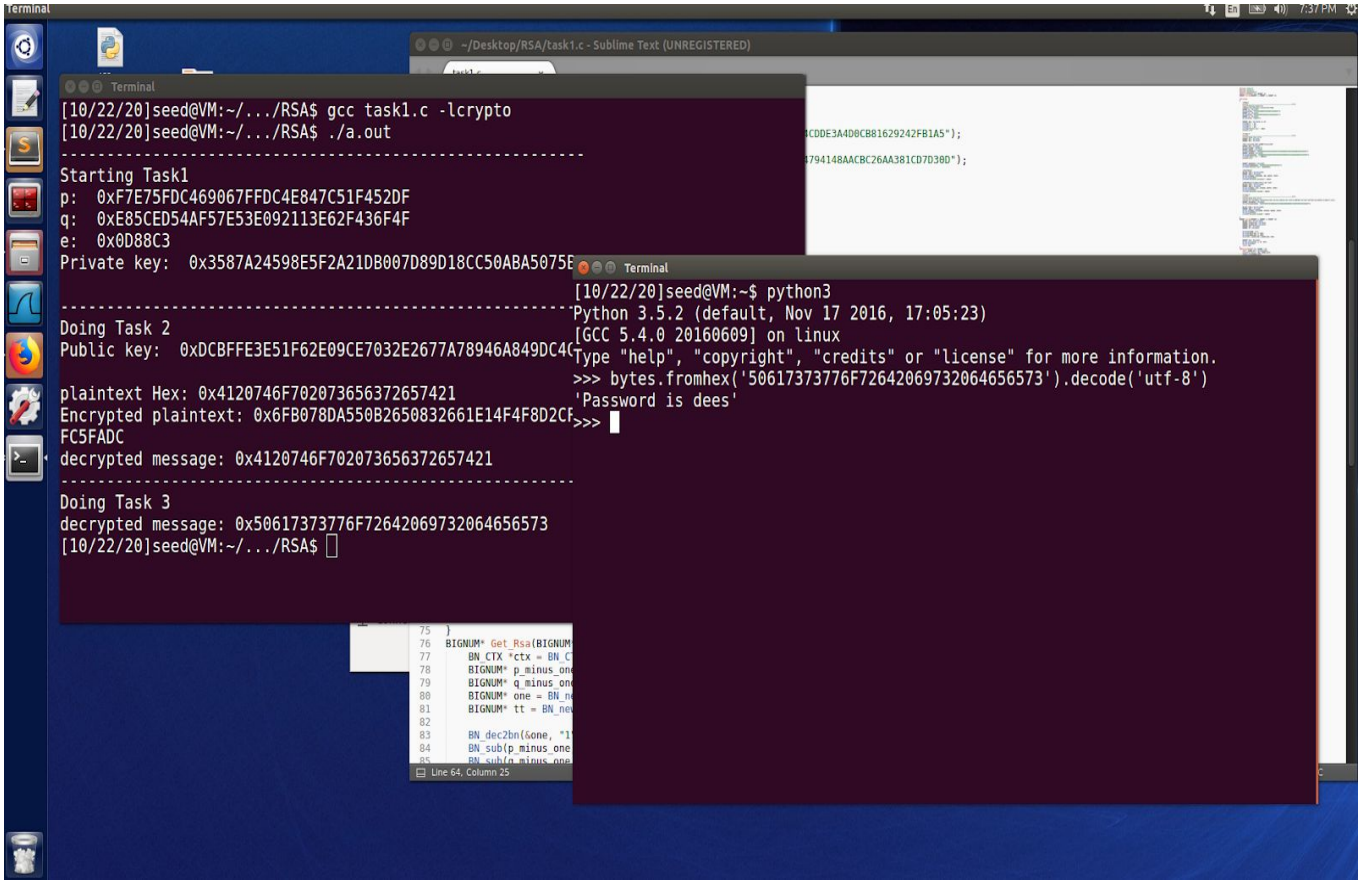


Task 3:Decrypting a Message

In task two we are required to use the same public and private keys as the previous tasks and only decrypt a ciphertext and then convert it from hex into ascii so we can see what it says. Basically using the same technique as step two to double check if we encrypted correctly.

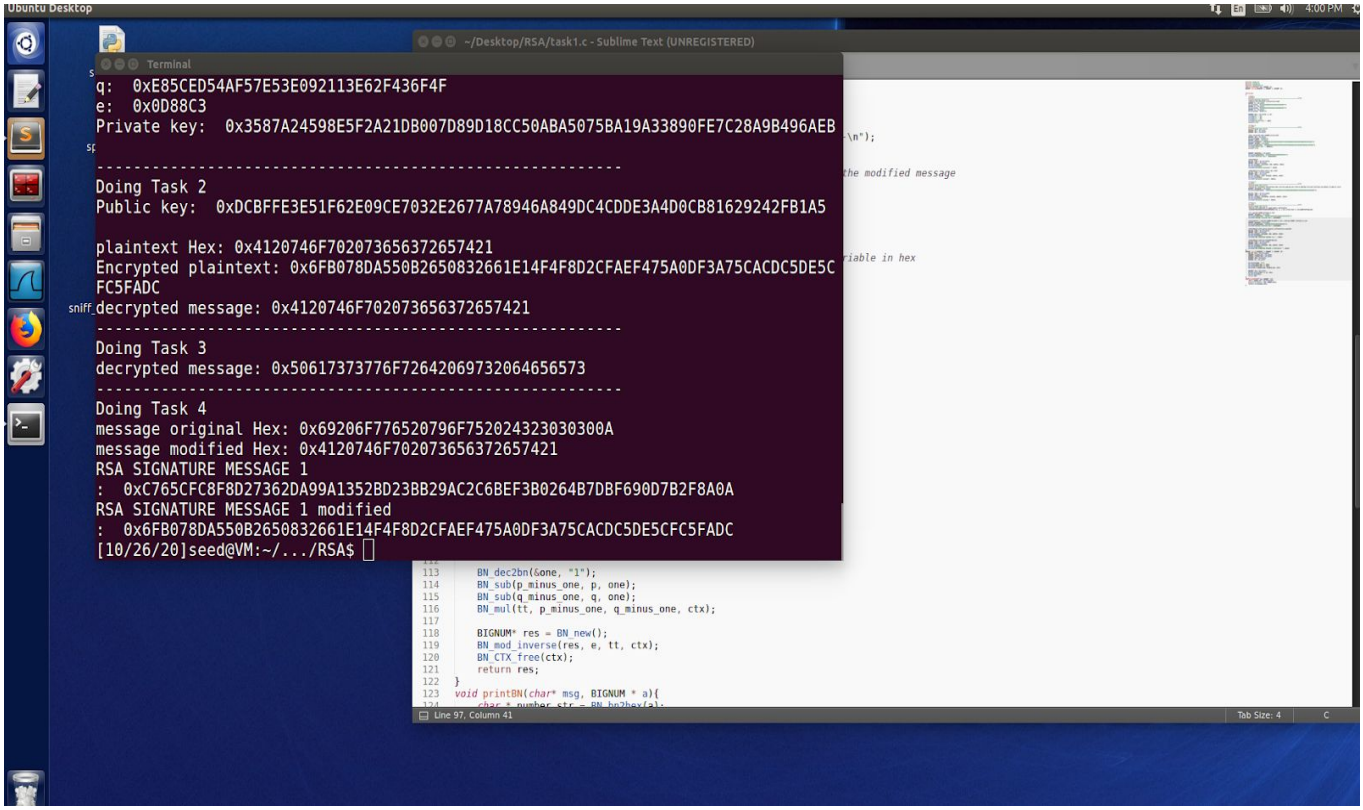
C=8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F

We got the decrypted ciphertext as “password is dees”



Task 4:Signing a Message

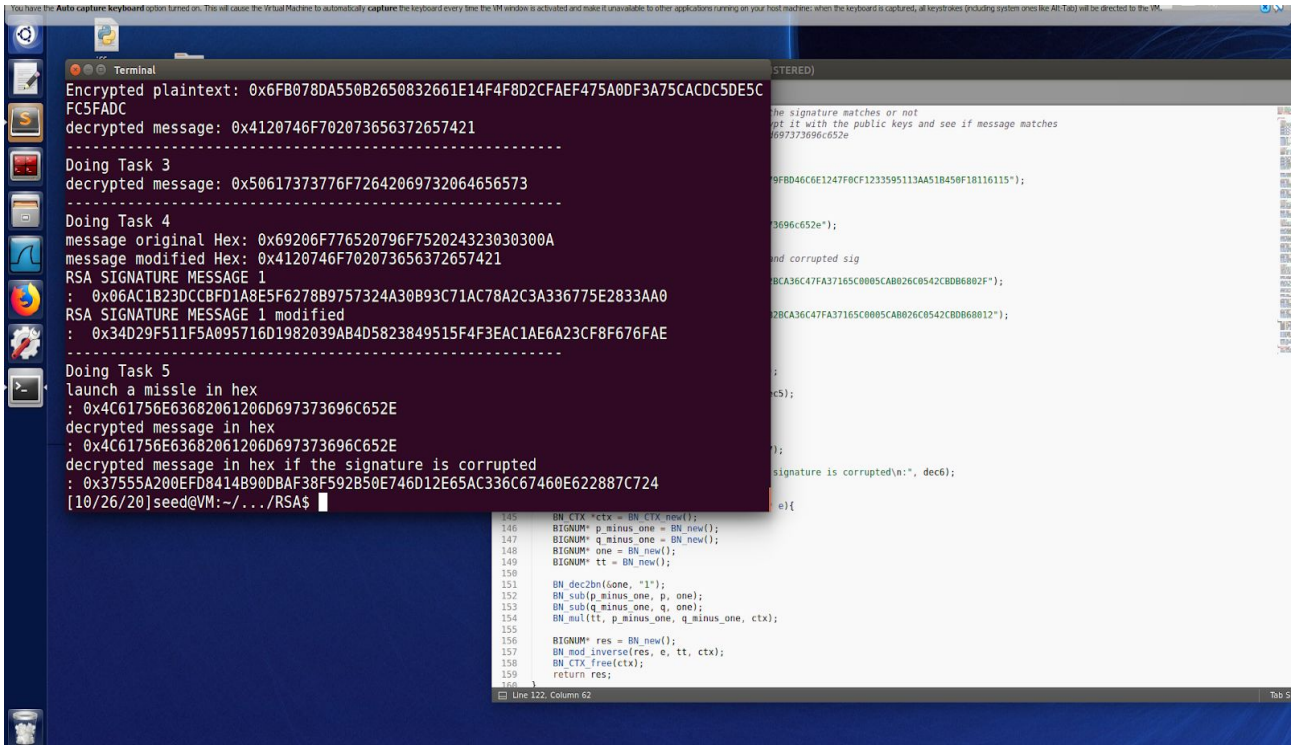
So we are given the keys remain the same and a message “i owe you $2000” and we are supposed to compare the signatures of the message original to the message modified and that's what we did, the signature is not the same. The signature is changed because the message was changed.



Task 5:Verifying a Signature

Here we are given the keys and everything we need to determine if the message sent was from Alice or not.

M = Launch a missile.
S=643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
e = 010001 (this hex value equals to decimal 65537)
n=AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115
This can be checked using the public key to decrypt the signature and see if the message is the same. We also corrupted the signature and saw if it matched the message to compare. The original message was sent by Alice but it i changed the signature (corrupted it) then we get something else which does not match the original message.



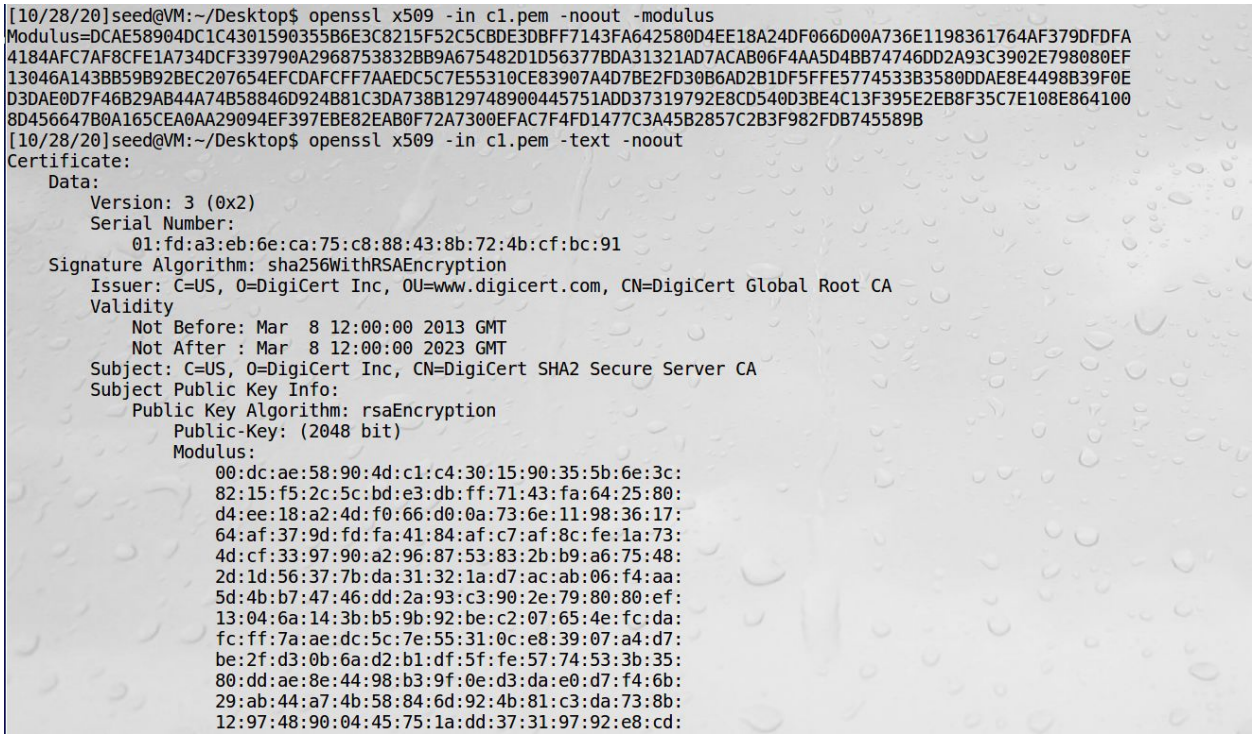
Task 6:Manually Verifying an X.509 Certificate

In this task we will be exploring X.509 certificates and its information to verify and conclude if its a legitimate signature through the issuers public key.

First, we pulled a certificate from www.example.org. The certificate is signed and the issuer information is indicated in the "i:" field.

```
[10/28/20]seed@VM:~/Desktop$ openssl s_client -connect www.example.org:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA
verify return:1
depth=0 C = US, ST = California, L = Los Angeles, O = Internet Corporation for Assigned Names and Numbers, O
U = Technology, CN = www.example.org
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Los Angeles/O=Internet Corporation for Assigned Names and Numbers/OU=Technology/C
N=www.example.org
   i:/C=US/O=DigiCert Inc/CN=DigiCert SHA2 Secure Server CA
-----BEGIN CERTIFICATE-----
MIIHQDCCBiigAwIBAgIQD9B43Ujxor1NDyupa2A4/jANBgkqhkiG9w0BAQsFADBN
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMScwJQYDVQQDEx5E
aWdpQ2VydCBTSEEyIFNlY3VyZSBTZXJ2ZXIgQ0EwHhcNMTgxMTI4MDAwMDAwWhcN
MjAxMjAyMTIwMDAwWjCBpTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3Ju
aWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMTwwOgYDVQQKEzNJbnRlcm5ldCBDb3Jw
b3JhdGlvbiBmb3IgQXNzaWduZWQgTmFtZXMgYW5kIE51bWJlcnMxEzARBgNVBAsT
ClRlY2hub2xvZ3kxGDAWBgNVBAMTD3d3dy5leGFtcGxlLm9yZzCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBANDwEnSgliByCGUZElpdStA6jGaPoCkrp9vV
rAzPpXGSFUIVsAeSdjF11yeOTVBqddF7U14nqu3rpGA68o5FGGtFM1yFEaogEv5g
rJ1MRY/d0w4+dw8JwoVlNMci+3QTuUKf9yH28JxEdG3J37Mfj2C3cREGkGNBnY80
eyRJRqzy8I0LSPTTkhr3okXuzOXXg38ugr1x3SgZWDNuEaE6oGpyYJIBWZ9jF3pJ
QnucP9vTBejMh374qvyd0QVQq3WxHrogy4nUbWw3gihMxT98wRD1oKVma1NTydvt
hcNtBfhkp8kO64/hxLHrLWgOFT/l4tz8IWQt7mkrBHjbd2XLVPkCAwEAAaOCA8Ew
ggO9MB8GA1UdIwQYMBaAFA+AYRyCMWHVLyjnjUY4tCzhxtniMB0GA1UdDgQWBBRm
mGIC4AmRp9njNvt2xrC/oW2nvjCBgQYDVR0RBHoweIIPd3d3LmV4YW1wbGUub3Jn
ggtleGFtcGxlLmNvbYILZXhhbXBsZS5lZHWCC2V4YW1wbGUubmV0ggtleGFtcGxl
Lm9yZ4IPd3d3LmV4YW1wbGUuY29tgg93d3cuZXhhbXBsZS5lZHWCD3d3dy5leGFt
cGxlLm5ldDAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsG
AQUFBwMCMGsGA1UdHwRkMGIwL6AtoCuGKWh0dHA6Ly9jcmwzLmRpZ2ljZXJ0LmNv
bS9zc2NhLXNoYTItZzYuY3JsMC+gLaArhilodHRwOi8vY3JsNC5kaWdpY2VydC5j
```

We need to extract the public key from the certificate. Using -modulus flag, we will find the value of n. And to find our e value, we print out the fields relevant to the certificate.



```
[10/28/20]seed@VM:~/Desktop$ openssl x509 -in c1.pem -noout -modulus
Modulus=DCAE58904DC1C4301590355B6E3C8215F52C5CBDE3DBFF7143FA642580D4EE18A24DF066D00A736E1198361764AF379DFDFA
4184AFC7AF8CFE1A734DCF339790A2968753832BB9A675482D1D56377BDA31321AD7ACAB06F4AA5D4BB74746DD2A93C3902E798080EF
13046A143BB59B92BEC207654EFCDAFCFF7AAEDC5C7E55310CE83907A4D7BE2FD30B6AD2B1DF5FFE5774533B3580DDAE8E4498B39F0E
D3DAE0D7F46B29AB44A74B58846D924B81C3DA738B129748900445751ADD37319792E8CD540D3BE4C13F395E2EB8F35C7E108E864100
8D456647B0A165CEA0AA29094EF397EBE82EAB0F72A7300EFAC7F4FD1477C3A45B2857C2B3F982FDB745589B
[10/28/20]seed@VM:~/Desktop$ openssl x509 -in c1.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:fd:a3:eb:6e:ca:75:c8:88:43:8b:72:4b:cf:bc:91
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert Global Root CA
        Validity
            Not Before: Mar  8 12:00:00 2013 GMT
            Not After : Mar  8 12:00:00 2023 GMT
        Subject: C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:dc:ae:58:90:4d:c1:c4:30:15:90:35:5b:6e:3c:
                    82:15:f5:2c:5c:bd:e3:db:ff:71:43:fa:64:25:80:
                    d4:ee:18:a2:4d:f0:66:d0:0a:73:6e:11:98:36:17:
                    64:af:37:9d:fd:fa:41:84:af:c7:af:8c:fe:1a:73:
                    4d:cf:33:97:90:a2:96:87:53:83:2b:b9:a6:75:48:
                    2d:1d:56:37:7b:da:31:32:1a:d7:ac:ab:06:f4:aa:
                    5d:4b:b7:47:46:dd:2a:93:c3:90:2e:79:80:80:ef:
                    13:04:6a:14:3b:b5:9b:92:be:c2:07:65:4e:fc:da:
                    fc:ff:7a:ae:dc:5c:7e:55:31:0c:e8:39:07:a4:d7:
                    be:2f:d3:0b:6a:d2:b1:df:5f:fe:57:74:53:3b:35:
                    80:dd:ae:8e:44:98:b3:9f:0e:d3:da:e0:d7:f4:6b:
                    29:ab:44:a7:4b:58:84:6d:92:4b:81:c3:da:73:8b:
                    12:97:48:90:04:45:75:1a:dd:37:31:97:92:e8:cd:
```

Extracting the signature from the certificate. We output the fields in the certificate and copy and paste the signature, and concatenate the hex-string by removing delimiters of ":" and "space".

```
[10/28/20]seed@VM:~/Desktop$ openssl x509 -in c0.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0f:d0:78:dd:48:f1:a2:bd:4d:0f:2b:a9:6b:60:38:fe
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
        Validity
            Not Before: Nov 28 00:00:00 2018 GMT
            Not After : Dec  2 12:00:00 2020 GMT
        Subject: C=US, ST=California, L=Los Angeles, O=Internet Corporation for Assigned Names and Numbers,
OU=Technology, CN=www.example.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d0:f0:12:74:a0:96:20:72:08:65:19:12:5a:5d:
                    4a:d0:3a:8c:66:8f:a0:29:2b:a7:db:d5:ac:0c:cf:
                    a5:71:92:15:42:15:b0:07:92:76:31:75:d7:27:8e:
                    4d:50:6a:75:d1:7b:53:5e:27:aa:ed:eb:a4:60:3a:
                    f2:8e:45:18:6b:45:33:5c:85:11:aa:20:12:fe:60:
                    ac:9d:4c:45:8f:dd:d3:0e:3e:77:0f:09:c2:85:65:
                    34:c7:22:fb:74:13:b9:42:9f:f7:21:f6:f0:9c:44:
                    74:6d:c9:df:b3:1f:8f:60:b7:71:11:06:90:63:41:
                    9d:8f:34:7b:24:49:46:ac:f2:f0:8d:0b:48:f4:d3:
                    92:1a:f7:a2:45:ee:cc:e5:d7:83:7f:2e:82:bd:71:
                    dd:28:19:58:33:6e:11:a1:3a:a0:6a:72:60:92:01:
                    59:9f:63:17:7a:49:42:7b:9c:3f:db:d3:05:e8:cc:
                    87:7e:f8:aa:fc:9d:d1:05:50:ab:75:b1:1e:ba:20:
                    cb:89:d4:6d:6c:37:82:28:4c:c5:3f:7c:c1:10:f5:
                    a0:a5:66:6b:53:53:c9:db:ed:85:c3:6d:05:f8:64:
                    a7:c9:0e:eb:8f:e1:c4:b1:eb:2d:68:0e:15:3f:e5:
                    e2:dc:fc:21:64:2d:ee:69:2b:04:78:db:77:65:cb:
                    54:f9
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:0F:80:61:1C:82:31:61:D5:2F:28:E7:8D:46:38:B4:2C:E1:C6:D9:E2

            X509v3 Subject Key Identifier:
                66:98:62:02:E0:09:91:A7:D9:E3:36:FB:76:C6:B0:BF:A1:6D:A7:BE
```

```
[10/28/20]seed@VM:~/Desktop$ cat signature | tr -d '[:space:]:'
737085ef4041a76a43d5789c7b5548e6bc6b9986bafb0d038b78fe11f029a00ccd69140bc60478b2cef087d5019dc4597a71fef06e9e
c1a0b0912d1fea3d55c533050ccdc13518b06a68664cbf5621da5bd948b98c3521915ddc75d77a462c2227a66fd33a17ebbebd13c512
2673c05da335896afb27d4ddaa74742e37e5013ba6d030b083d0a1c4752185b2e5fa670030a2bc53834dbfd6a883bbbcd6ed1cb31ef1
580382008e9cef90f21a5fa2a306da5dbe9fda5da6e62fde588018d3f1627ba6a39faea86972638165ae8283a3b5978a9b2051ff1a3f
61401e48d06b38f9e1fa17d8774a88e63d36244fef0ab99f70f38327f8cf2a057510a18a0a8088cd737085ef4041a76a43d5789c7b55
48e6bc6b9986bafb0d038b78fe11f029a00ccd69140bc60478b2cef087d5019dc4597a71fef06e9ec1a0b0912d1fea3d55c533050ccd
c13518b06a68664cbf5621da5bd948b98c3521915ddc75d77a462c2227a66fd33a17ebbebd13c5122673c05da335896afb27d4ddaa74
742e37e5013ba6d030b083d0a1c4752185b2e5fa670030a2bc53834dbfd6a883bbbcd6ed1cb31ef1580382008e9cef90f21a5fa2a306
da5dbe9fda5da6e62fde588018d3f1627ba6a39faea86972638165ae8283a3b5978a9b2051ff1a3f61401e48d06b38f9e1fa17d8774a
88e63d36244fef0ab99f70f38327f8cf2a057510a18a0a8088cd[10/28/20]seed@VM:~/Desktop$ ^C
```

Extracting the body of the server certificate was rather the more difficult part. Using the asn1parse command, we can focus on the part of the certificate that is used to generate the hash. Below we can find the body of the certificate that was used to generate the hash, which begins at “4:d=1” line. The signature block is at the “31:d=2” line.

```
[10/28/20]seed@VM:~$ openssl asn1parse -i -in c0.pem
c0.pem: No such file or directory
3070543552:error:02001002:system library:fopen:No such file or directory:bss_fil
e.c:398:fopen('c0.pem','r')
3070543552:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:400:
[10/28/20]seed@VM:~$ cd /home/seed/Desktop
[10/28/20]seed@VM:~/Desktop$ openssl asn1parse -i -in c0.pem
    0:d=0  hl=4 l=1856 cons: SEQUENCE
    4:d=1  hl=4 l=1576 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  16 prim:   INTEGER           :0FD078DD48F1A2BD4D0F2BA96B6038
FE
   31:d=2  hl=2 l=  13 cons:   SEQUENCE
   33:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   44:d=3  hl=2 l=   0 prim:    NULL
   46:d=2  hl=2 l=  77 cons:   SEQUENCE
   48:d=3  hl=2 l=  11 cons:    SET
   50:d=4  hl=2 l=   9 cons:     SEQUENCE
   52:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
   57:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   61:d=3  hl=2 l=  21 cons:    SET
   63:d=4  hl=2 l=  19 cons:     SEQUENCE
   65:d=5  hl=2 l=   3 prim:      OBJECT            :organizationName
```

Using the -strparse command will output the body of the signature without the signature block.. Once we obtain this body, we use it to calculate the hash by executing the command “sha256sum c0_body.bin”.

```
[10/28/20]seed@VM:~/Desktop$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
[10/28/20]seed@VM:~/Desktop$ sha256sum c0_body.bin
2c2a46bf245dab54ddb47298621e9629309f0e2c90c4d80d535c7d4e8ab07d29  c0_body.bin
[10/28/20]seed@VM:~/Desktop$ ^C
[10/28/20]seed@VM:~/Desktop$
```

In the end, we confirmed from our analysis of the certificate and the methods obtained for the body of the certificate, CA public key, and CA signature were found to be true and a **valid** signature.