Task 1: Deriving the Private Key

```
 GNU nano 4.8                                    task1.c
//K213881
#include <openssl/bn.h>
#include <stdio.h>

void printBN(char *msg, BIGNUM * a) {
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *one = BN_new();

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_dec2bn(&one, "1");

    BN_mul(n, p, q, ctx);
    //printBN("n = ", n);
```

```
 GNU nano 4.8                                    task1.c
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *one = BN_new();

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_dec2bn(&one, "1");

    BN_mul(n, p, q, ctx);
    //printBN("n = ", n);

    BIGNUM *p1 = BN_new();
    BIGNUM *q1 = BN_new();
    BN_sub(p1, p, one);
    BN_sub(q1, q, one);
BN_mul(phi, p1, q1, ctx);

BN_mod_inverse(d, e, phi, ctx);
printBN("private key d = ", d);

return 0;
}
```

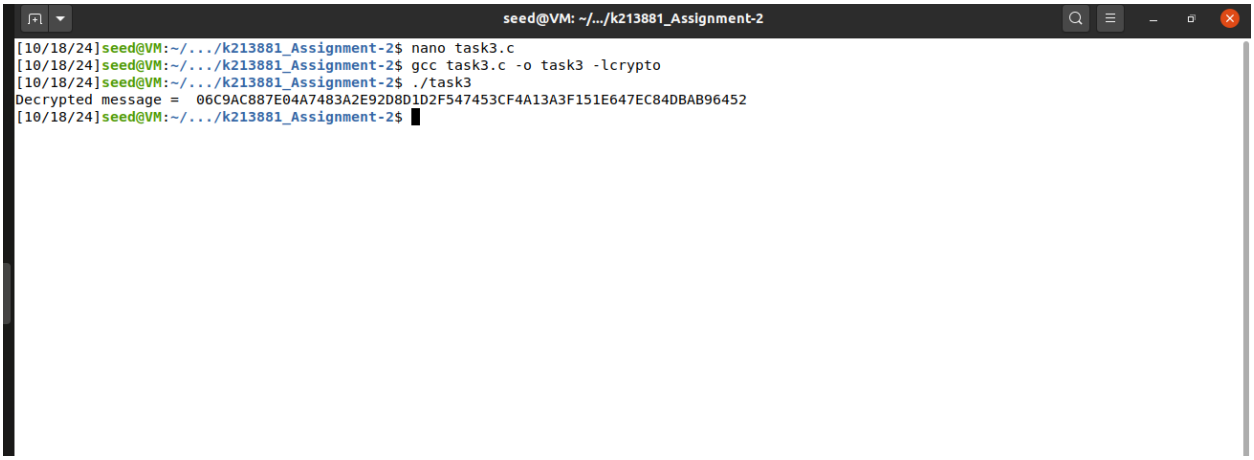
```
seed@VM: ~/.../k213881_Assignment-2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano task1.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc task1.c -o task1 -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./task1
private key d =  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
```

Task 2: Encrypting a Message

```
  GNU nano 4.8                                                task2.c
// k213881
#include <openssl/bn.h>
#include <stdio.h>

void printBN(char *msg, BIGNUM *a) {
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *c = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62209CE7032E2677A79864A89D4C4DDE3A4D0BC88162924F81A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&m, "412074f70207356372567421");

    BN_mod_exp(c, m, e, n, ctx);
    printBN("Ciphertext = ", c);

    return 0;
}
```

```
seed@VM: ~/.../k213881_Assignment-2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano task2.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc task2.c -o task2 -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./task2
Ciphertext =  042B5A5A00312EC413028808F16B5EE5D0710F1C027B409E4E8F3A473D859B1F
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
```

Task 3: Decrypting a Message



Task 4: Signing a Message

```
 GNU nano 4.8                                                    task4.c
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}

int main() {
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *s = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62209CE7032E2677A79864A89D4C4DDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D8069F5A3C182DE2E4794148AABC26AA381CD7D30D");

    BN_hex2bn(&m, "49206f776520796f75202432303030");  // "I owe you $2000." in hex
    BN_mod_exp(s, m, d, n, ctx);
    printBN("(I owe you $2000) Signature = ", s);

    // Now change the message to "I owe you $3000." and sign again
    BN_hex2bn(&m, "49206f776520796f75202433303030");  // "I owe you $3000." in hex
    BN_mod_exp(s, m, d, n, ctx);
    printBN("(I owe you $3000) Signature = ", s);

    return 0;
}
                                                    [ Wrote 32 lines ]
```

```
seed@VM: ~/.../k213881_Assignment-2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano task4.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc task4.c -o task4 -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./task4
(I owe you $2000) Signature =  04833D3AE18969995ACE09D294D234140D819885F65BC719040C61A144667ED3
(I owe you $3000) Signature =  0D8EB028EA42DA0B3B11D478218DC9886D167A416AF15BD431737E2D046C31B2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
```

Description:

In this task, the original message "I owe you $2000." is signed using RSA, and then a modified version of the message ("I owe you $3000.") is also signed. Upon comparing the

signatures of the two messages, the signatures are completely different. This is because RSA signatures are generatedbased on the exact content of the message, so even a small change, such as modifying "$2000" to"$3000," results in a different hash value and thus a different signature. This behaviour demonstrates the integrity protection offered by digital signatures, ensuring that even minor changes to the original message can be detected, making it tamper-evident.

Task 5: Verifying a Signature

```
 GNU nano 4.8                                                    task5.c
//k213881
#include <openssl/bn.h>
#include <stdio.h>

void printBN(char *msg, BIGNUM *a) {
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}

int main() {
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *s = BN_new();
    BIGNUM *m_verify = BN_new();
    BIGNUM *expected_m = BN_new();

    BN_hex2bn(&n, "AE1CD4DC432379B97FD846CE1C4720559F1233955113AA51B450F18116115");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&s, "643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6802F");
    BN_hex2bn(&expected_m, "4C61756E63682061206D697373696C652E");

    printf("original signature: 643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6802F\n");

    BN_mod_exp(m_verify, s, e, n, ctx);
    printBN("Recovered message (original) = ", m_verify);

                                                    [ Wrote 49 lines ]
```

```
 GNU nano 4.8                                                task5.c

    printf("original signature: 643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6802F\n");

    BN_mod_exp(m_verify, s, e, n, ctx);
    printBN("Recovered message (original) = ", m_verify);

if (BN_cmp(m_verify, expected_m) == 0) {
    printf("The signature is valid for the original message.\n");
} else {
    printf("The signature is NOT valid for the original message.\n");
}

BIGNUM *s_corrupted = BN_new();
BN_hex2bn(&s_corrupted, "643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6803F");
printf("Modified signature (2F -> 3F): 643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6803F\n");

BN_mod_exp(m_verify, s_corrupted, e, n, ctx);
printBN("Recovered message (corrupted) = ", m_verify);

if (BN_cmp(m_verify, expected_m) == 0) {
    printf("The corrupted signature is still valid for the message.\n");
} else {
    printf("The corrupted signature is NOT valid for the message.\n");
}

return 0;
}
```

```
seed@VM: ~/.../k213881_Assignment-2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano task5.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc task5.c -o task5 -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./task5
original signature: 643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6802F
Recovered message (original) =  0A3A7E027E690FBAAC4B2C6BD584E0C8106BE45190F9123E75F53812B2F258
The signature is NOT valid for the original message.
Modified signature (2F -> 3F): 643D6F3490209C7EC9C0B2BCA36C47FA37165C0005CAB026C0542CBD6803F
Recovered message (corrupted) =  049A2CFFFFB3BAC888F5A44CCD70A2AD4A4D039903BAFB36C14F43A2B2C866
The corrupted signature is NOT valid for the message.
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
```

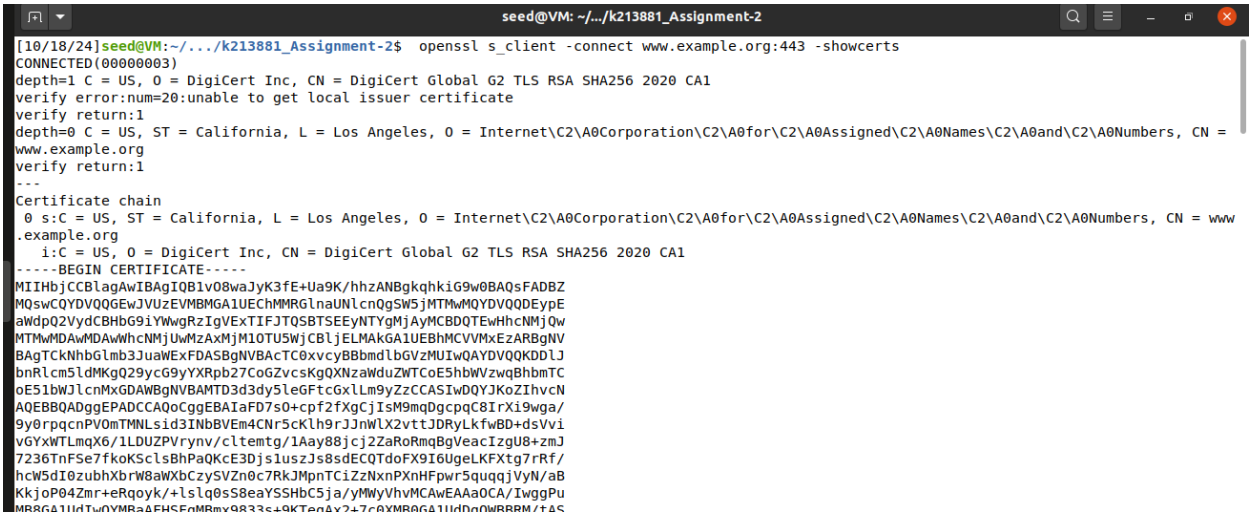
Description:

When Bob verifies Alice's signature on the message "Launch a missile." with the correct public key, the verification succeeds if the signature is unaltered. However, when the last byte of the signatureis changed from 2F to 3F, the verification fails. This is because even a tiny alteration in the signaturedisrupts the cryptographic process, causing the signature to no longer match the message. Thisdemonstrates how RSA ensures message integrity, as any modification to the signature or message will result in verification failure, providing strong tamper detection.

Task 6: Manually Verifying an X.509 Certificate

Major steps to follow in this tasks are:

- The objective of this task is to manually authenticate an X.509 certificate by obtaining the public key and signature from the certificate and confirming the signature's validity. This process includes downloading a certificate from a website, extracting necessary details, and validating the certificate using the issuer's public key.
- X.509 Certificates utilize public key infrastructure (PKI) to confirm the legitimacy of a server or individual.
- We will retrieve a certificate from the web server using the openssl s_client command, specifically from www.example.org.
- The initial certificate (labeled 0) is the server's certificate, while the subsequent one (labeled 1) is the issuer's certificate.
- Save these two certificates into separate files: the server certificate as server_cert.pem and the issuer's certificate as issuer_cert.pem.
- Extract the public key from the issuer's certificate.
- Extract the modulus component of the public key.
- Display the entire certificate to identify the public exponent e.
- Extract the signature from the server's certificate.
- The signature will be found under the Signature Algorithm section.
- Copy the entire signature and save it to a file named signature.txt.
- Extract the main body of the server's certificate. For verification, the hash is calculated based on the certificate's body.
- The main body of the certificate will be saved as server_body.bin, which will be used to generate the hash.
- The signature is created by hashing the certificate body first, then encrypting the hash with the CA's private key.
- Check the signature. If the signature is valid, the decrypted signature will match the hash computed from the certificate body. Otherwise, the signature may be corrupted or the certificate has been altered.


seed@VM: ~/.../k213881_Assignment-2
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ 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 CA1
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-----
MIIHbjCCBlagAwIBAgIQB1vO8waJyK3fE+Ua9K/hhzANBgkqhkiG9w0BAQsFADBZ
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMTMwMQYDVQQDEypE
aWdpQ2VydCBHbG9iYWwgRzIgVExTIFJTQSBTSEEyNTYgMjAyMCBDQTEwHhcNMjQw
MTMwMDAwMDAwWhcNMjUwMzAxMjM1OTU5WjCBljELMAkGA1UEBhMCVVMxEzARBgNV
BAgTCkNhbGlmb3JuaWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMUIwQAYDVQQKDDlJ
bnRlcm5ldMKgQ29ycG9yYXRpb27CoGZvcsKgQXNzaWduZWTCoE5hbWVzwqBhbmTC
oE51bWJlcnMxGDAWBgNVBAMTD3d3dy5leGFtcGxlLm9yZzCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAIaFD7sO+cpf2fXgCjIsM9mqDgcpqC8IrXi9wga/
9y0rpqcnPVOmTMNLsid3INbBVEm4CNr5cKlh9rJJnWlX2vttJDRyLkfwBD+dsVvi
vGYxWTLmqX6/1LDUZPVrynv/cltemtg/1Aay88jcj2ZaRoRmqBgVeacIzgU8+zmJ
7236TnFSe7fkoKSclsBhPaQKcE3Djs1uszJs8sdECQTdoFX9I6UgeLKFXtg7rRf/
hcW5dI0zubhXbrW8aWXbCzySVZn0c7RkJMpnTCiZzNxnPXnHFpwr5quqqjVyN/aB
KkjoP04Zmr+eRqoyk/+lslq0sS8eaYSSHbC5ja/yMWyVhvMCAwEAAaOCA/IwggPu
MB8GA1UdIwQYMBaAFHSFgMBmx9833s+9KTeqAx2+7c0XMB0GA1UdDgQWBBRM/tAS
TS4hz2v68vK4TEkCHTGRijCBgQYDVR0RBHoweIIPd3d3LmV4YW1wbGUub3Jnggtl
eGFtcGxlLm5ldIILZXhhbXBsZS5lZHWCC2V4YW1wbGUuY29tggtleGFtcGxlLm9y
Z4IPd3d3LmV4YW1wbGUuY29tgg93d3cuZXhhbXBsZS5lZHWCD3d3dy5leGFtcGxl
Lm5ldDA+BgNVHSAENzA1MDMGBmeBDAECAjApMCcGCCsGAQUFBwIBFhtodHRwOi8v
d3d3LmRpZ2ljZXJ0LmNvbS9DUFMwDgYDVR0PAQH/BAQDAgWgMB0GA1UdJQQWMBQG
CCsGAQUFBwMBBggrBgEFBQcDAjCBnwYDVR0fBIGXMIGUMEigRqBEhkJodHRwOi8v
Y3JsMy5kaWdpY2VydC5jb20vRGlnaUNlcnRHbG9iYWxHMlRMU1JTQVNIQTI1NjIw
MjBDQTEtMS5jcmwwSKBGoESGQmh0dHA6Ly9jcmw0LmRpZ2ljZXJ0LmNvbS9EaWdp
Q2VydEdsb2JhbEcyVExTUlNBU0hSMjU2MjAyMENBMS0xLmNybDCBhwYIKwYBBQUH
AQEEezB5MCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2VydC5jb20wUQYI
KwYBBQUHMAKGRWh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9EaWdpQ2VydEds
b2JhbEcyVExTUlNBU0hSMjU2MjAyMENBMS0xLmNydDAMBgNVHRMBAf8EAjAAMIIB
fQYKKwYBBAHWeQIEAgSCAW8EggFpAWcAdABOdaMnXJoQwzhbbNTfP1LrHfDgjhuN
acCx+mSxYpo53wAAAY1b0vxkAAAEAwBFMEMCH0BRCgxPbBBVxhcWZ26a8JCe83P1
JZ6wmv56GsVcyMACIDgpMbEo5HJITTRPnoyT4mG8cLrWjEvhchUdEcWUuk1TAHYA
fVkeEuF4KnscYWd8Xv340IdcFKBOlZ65Ay/ZDowuebgAAAGNW9L8MAAABAMARzBF
AiBdv5Z3pZFbfgoM3tGpCTM3ZxBMQsxBRSdTS6d8d2NAcwIhALLoCT9mTMN9OyFz
IBV5MkXVLyuTf2OAzAOa7d8x2H6XAHcA5tIxY0B3jMEQQQbXcbnOwdJA9paEhvu6
hzId/R43jlAAAAGNW9L8XwAABAMASDBGAiEA4Koh/VizdQUltjZ2E2VGgWSXXkwn
QmiYhmAeKcVLHeACIQD7JIGFsdGol7kss2pe4lYrCgPVc+iGZkuqnj26hqhr0TAN
BgkqhkiG9w0BAQsFAAOCAQEABOFuAj4N4yNG9OOWNOWTNSICC4Rd4nOGlHRP/Bsn
rz7KrcPORtb6D+Jx+Q0amhO3lQhIvVBYs14gY4Ypyj7MzHgm4VmPXcqLvEkxb2G9
Qv9hYuEiNSQmm1fr5QAN/0AzbEbCM3cImLJ69kP5bUjfv/76KB57is8tYf9sh5ik
LGKauxCM/zRIcGa3bXLDafk5S2g5Vr2hs230d/NGW1wZrE+zdGuMxfGJzJP+DAFv
iBfcQnFg4+1zMEKcqS87oniOyG+60RMM0MdejBD7AS43m9us96Gsun/4kufLQUTI
FfnzxLutUV++3seshgefQOy5C/ayi8y1VTNmujPCxPCi6Q==
-----END CERTIFICATE-----
1 s:C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA256 2020 CA1
i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root G2


seed@VM: ~/.../k213881_Assignment-2
i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root G2
-----BEGIN CERTIFICATE-----
MIIEyDCCA7CgAwIBAgIQDPW9BitWAvR6uFAsI8zwZjANBgkqhkiG9w0BAQsFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBH
MjAeFw0yMTAzMzAwMDAwMDBaFw0zMTAzMjkyMzU5NTlaMFkxCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxMzAxBgNVBAMTKkRpZ2lDZXJ0IEdsb2Jh
bCBHMiBUTFMgUlNBIFNIQTI1NiAyMDIwIENBMTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMz3EGJPprtjb+2QUlbFbSd7ehJWivH0+dbn4Y+9lavyYEEV
cNsSAPonCrVXOFt9slGTcZUOakGUWzUb+nv6u8W+JDD+Vu/E832X4xT1FE3LpxDy
FuqrIvAxIhFhaZAmunjZlx/jfWardUSVc8is/+9dCopZQ+GssjoP80j812s3wWPc
3kbW20X+fSP9kOhRBx5Ro1/tSUZUfyyIxfQTnJcVPAPooTncaQwywa8WV0yUR0J8
osicfebUTVSvQpmowQTCd5zWSOTOEeAqgJnwQ3DPP3Zr0UxJqyRewg2C/Uaoq2yT
zGJSQnWS+Jr6Xl6ysGHlHx+5fwmY6D36g39HaaECAwEAAaOCAYIwggF+MBIGA1Ud
EwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHSFgMBmx9833s+9KTeqAx2+7c0XMB8G
A1UdIwQYMBaAFE4iVCAYlebjbuYP+vq5Eu0GF485MA4GA1UdDwEB/wQEAwIBhjAd
BgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwdgYIKwYBBQUHAQEEajBoMCQG
CCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2VydC5jb20wQAYIKwYBBQUHMAKG
NGh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9EaWdpQ2VydEdsb2JhbFJvb3RH
Mi5jcnQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybDMuZGlnaWNlcnQuY29t
L0RpZ2lDZXJ0R2xvYmFsUm9vdEcyLmNybDA9BgNVHSAENjA0MAsGCWCGSAGG/WwC
ATAHBgVngQwBATAIBgZngQwBAgEwCAYGZ4EMAQICMAgGBmeBDAECAzANBgkqhkiG
9w0BAQsFAAOCAQEAkPFwyyiXaZd8dP3A+iZ7U6utzWX9upwGnIrXWkOH7U1MVl+t
wcW1BSAuWdH/SvWgKtiwla3JLko716f2b4gp/DA/JIS7w7d7kwcsr4drdjPtAFVS
slme5LnQ89/nD/7d+MS5EHKBCQRfz5eeLjJ1js+aWNJXMX43AYGyZm0pGrFmCW3R
bpD0ufovARTFXFZkAdl9h6g4U5+LXUZtXMYnhIHUfoyMo5tS58aI7Dd8KvvwVVo4
chDYABPPTHPbqjc1qCmBaZx2vN4Ye5DUys/vZwP9BFohFrH/6j/f3IL16/RZkiMN
JCqVJUzKoZHm1Lesh3Sz8W2jmdv51b2EQJ8HmA==
-----END CERTIFICATE-----

Server certificate
subject=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
Right Ctrl


seed@VM: ~/.../k213881_Assignment-2
-----END CERTIFICATE-----

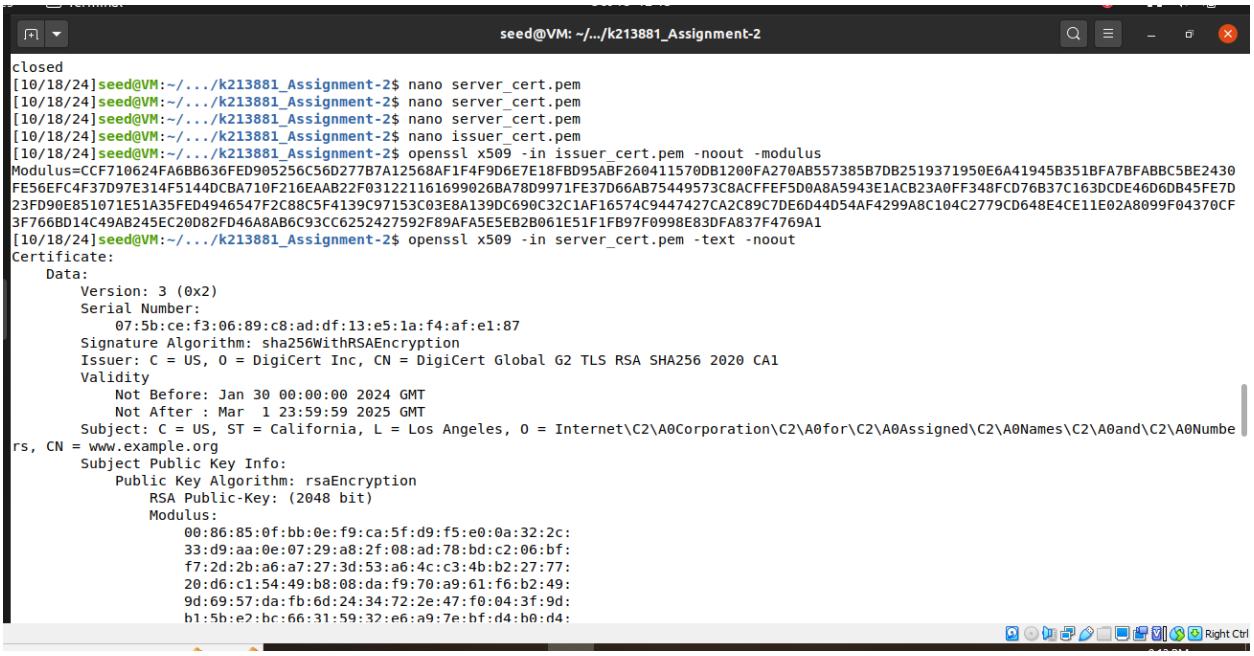
Server certificate
subject=C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbers, CN =
www.example.org
issuer=C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA256 2020 CA1

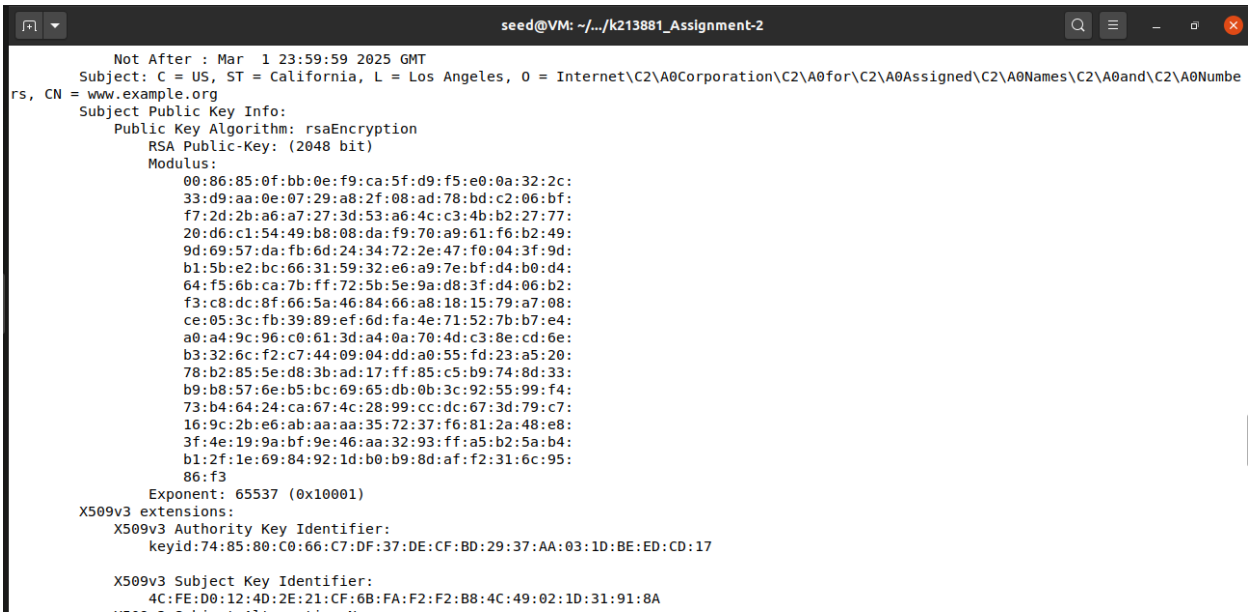
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: ECDH, P-256, 256 bits

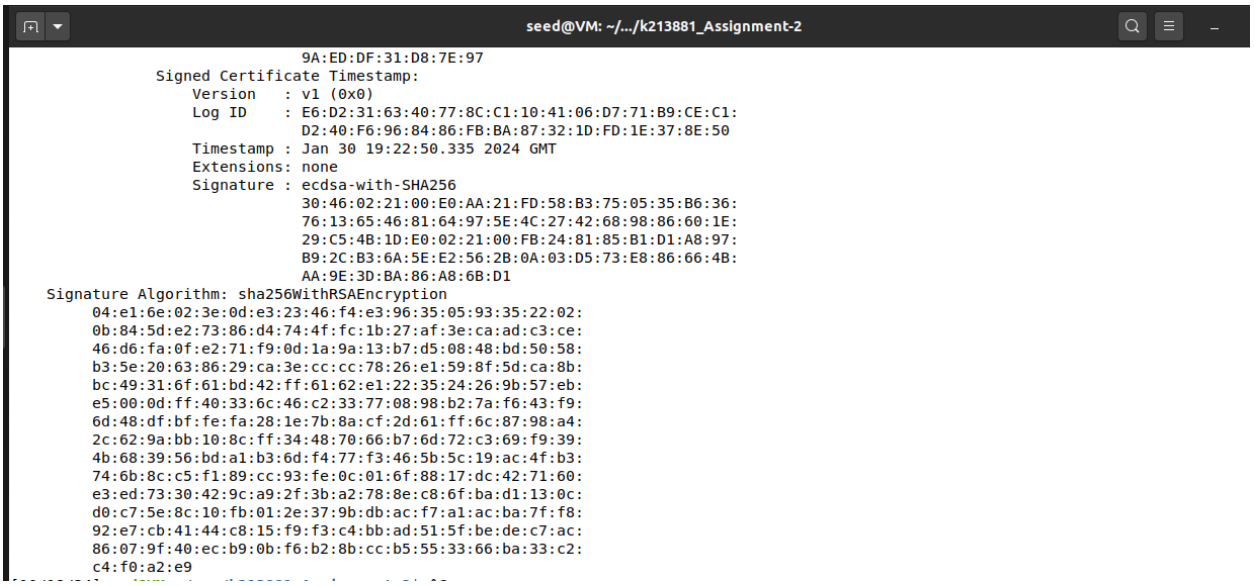
SSL handshake has read 3821 bytes and written 747 bytes
Verification error: unable to get local issuer certificate

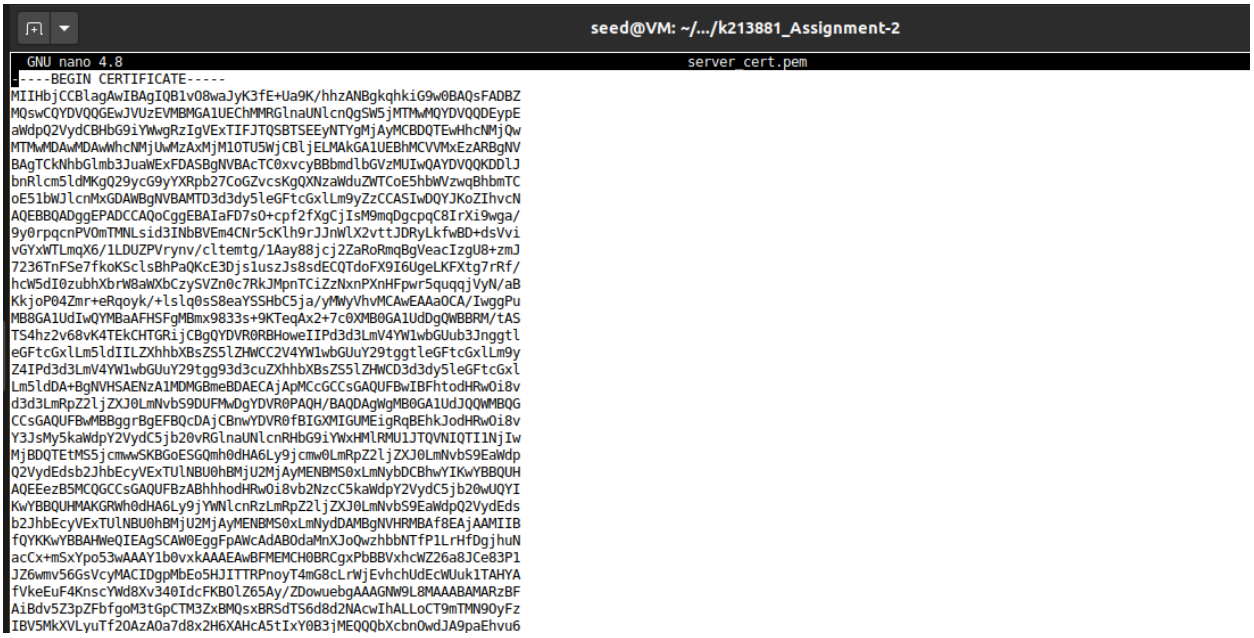
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 2048 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 20 (unable to get local issuer certificate)

Post-Handshake New Session Ticket arrived:
SSL-Session:
Protocol : TLSv1.3
Cipher : TLS_AES_256_GCM_SHA384
Session-ID: 5D871833981A20B5173B6E3DC9875858B050786ED54D302B4EBB5EC9E365DD57
Session-ID-ctx:


seed@VM: ~/.../k213881_Assignment-2
closed
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano server_cert.pem
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano server_cert.pem
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano server_cert.pem
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano issuer_cert.pem
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ openssl x509 -in issuer_cert.pem -noout -modulus
Modulus=CCF710624FA6BB636FED905256C56D277B7A12568AF1F4F9D6E7E18FBD95ABF260411570DB1200FA270AB557385B7DB2519371950E6A41945B351BFA7BFABBC5BE2430
FE56EFC4F37D97E314F5144DCBA710F216EAAB22F0312211616990268A78D9971FE37D66AB75449573C8ACFFEF5D0A8A5943E1ACB23A0FF348FCD76B37C163DCDE46D6DB45FE7D
23FD90E851071E51A35FED4946547F2C88C5F4139C97153C03E8A139DC690C32C1AF16574C9447427CA2C89C7DE6D44D54AF4299A8C104C2779CD648E4CE11E02A8099F04370CF
3F766BD14C49AB245EC20D82FD46A8AB6C93CC6252427592F89AFA5E5EB2B061E51F1FB97F0998E83DFA837F4769A1
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ openssl x509 -in server_cert.pem -text -noout
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
07:5b:ce:f3:06:89:c8:ad:df:13:e5:1a:f4:af:e1:87
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = DigiCert Inc, CN = DigiCert Global G2 TLS RSA SHA256 2020 CA1
Validity
Not Before: Jan 30 00:00:00 2024 GMT
Not After : Mar 1 23:59:59 2025 GMT
Subject: C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbe
rs, CN = www.example.org
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public-Key: (2048 bit)
Modulus:
00:86:85:0f:bb:0e:f9:ca:5f:d9:f5:e0:0a:32:2c:
33:d9:aa:0e:07:29:a8:2f:08:ad:78:bd:c2:06:bf:
f7:2d:2b:a6:a7:27:3d:53:a6:4c:c3:4b:b2:27:77:
20:d6:c1:54:49:b8:08:da:f9:70:a9:61:f6:b2:49:
9d:69:57:da:fb:6d:24:34:72:2e:47:f0:04:3f:9d:
Right Ctrl

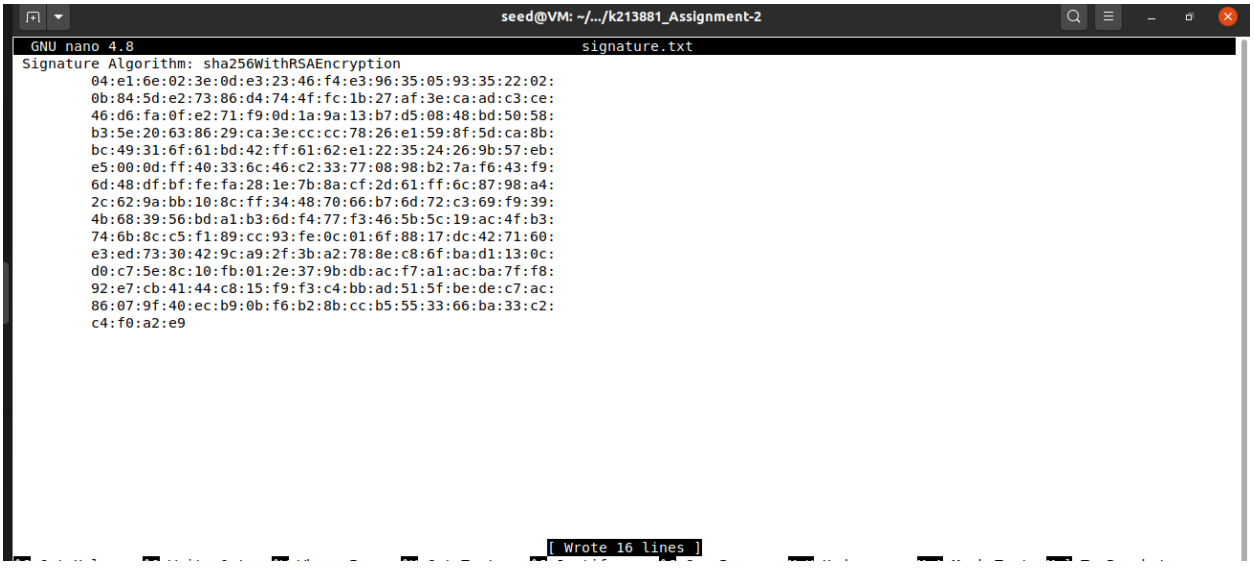

seed@VM: ~/.../k213881_Assignment-2
Not After : Mar 1 23:59:59 2025 GMT
Subject: C = US, ST = California, L = Los Angeles, O = Internet\C2\A0Corporation\C2\A0for\C2\A0Assigned\C2\A0Names\C2\A0and\C2\A0Numbe
rs, CN = www.example.org
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public-Key: (2048 bit)
Modulus:
00:86:85:0f:bb:0e:f9:ca:5f:d9:f5:e0:0a:32:2c:
33:d9:aa:0e:07:29:a8:2f:08:ad:78:bd:c2:06:bf:
f7:2d:2b:a6:a7:27:3d:53:a6:4c:c3:4b:b2:27:77:
20:d6:c1:54:49:b8:08:da:f9:70:a9:61:f6:b2:49:
9d:69:57:da:fb:6d:24:34:72:2e:47:f0:04:3f:9d:
b1:5b:e2:bc:66:31:59:32:e6:a9:7e:bf:d4:b0:d4:
64:f5:6b:ca:7b:ff:72:5b:5e:9a:d8:3f:d4:06:b2:
f3:c8:dc:8f:66:5a:46:84:66:a8:18:15:79:a7:08:
ce:05:3c:fb:39:89:ef:6d:fa:4e:71:52:7b:b7:e4:
a0:a4:9c:96:c0:61:3d:a4:0a:70:4d:c3:8e:cd:6e:
b3:32:6c:f2:c7:44:09:04:dd:a0:55:fd:23:a5:20:
78:b2:85:5e:d8:3b:ad:17:ff:85:c5:b9:74:8d:33:
b9:b8:57:6e:b5:bc:69:65:db:0b:3c:92:55:99:f4:
73:b4:64:24:ca:67:4c:28:99:cc:dc:67:3d:79:c7:
16:9c:2b:e6:ab:aa:aa:35:72:37:f6:81:2a:48:e8:
3f:4e:19:9a:bf:9e:46:aa:32:93:ff:a5:b2:5a:b4:
b1:2f:1e:69:84:92:1d:b0:b9:8d:af:f2:31:6c:95:
86:f3
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Authority Key Identifier:
keyid:74:85:80:C0:66:C7:DF:37:DE:CF:BD:29:37:AA:03:1D:BE:ED:CD:17
X509v3 Subject Key Identifier:
4C:FE:D0:12:4D:2E:21:CF:6B:FA:F2:F2:B8:4C:49:02:1D:31:91:8A


seed@VM: ~/.../k213881_Assignment-2
9A:ED:DF:31:D8:7E:97
Signed Certificate Timestamp:
Version : v1 (0x0)
Log ID : E6:D2:31:63:40:77:8C:C1:10:41:06:D7:71:B9:CE:C1:
D2:40:F6:96:84:86:FB:BA:87:32:1D:FD:1E:37:8E:50
Timestamp : Jan 30 19:22:50.335 2024 GMT
Extensions: none
Signature : ecdsa-with-SHA256
30:46:02:21:00:E0:AA:21:FD:58:B3:75:05:35:B6:36:
76:13:65:46:81:64:97:5E:4C:27:42:68:98:86:60:1E:
29:C5:4B:1D:E0:02:21:00:FB:24:81:85:B1:D1:A8:97:
B9:2C:B3:6A:5E:E2:56:2B:0A:03:D5:73:E8:86:66:4B:
AA:9E:3D:BA:86:A8:6B:D1
Signature Algorithm: sha256WithRSAEncryption
04:e1:6e:02:3e:0d:e3:23:46:f4:e3:96:35:05:93:35:22:02:
0b:84:5d:e2:73:86:d4:74:4f:fc:1b:27:af:3e:ca:ad:c3:ce:
46:d6:fa:0f:e2:71:f9:0d:1a:9a:13:b7:d5:08:48:bd:50:58:
b3:5e:20:63:86:29:ca:3e:cc:cc:78:26:e1:59:8f:5d:ca:8b:
bc:49:31:6f:61:bd:42:ff:61:62:e1:22:35:24:26:9b:57:eb:
e5:00:0d:ff:40:33:6c:46:c2:33:77:08:98:b2:7a:f6:43:f9:
6d:48:df:bf:fe:fa:28:1e:7b:8a:cf:2d:61:ff:6c:87:98:a4:
2c:62:9a:bb:10:8c:ff:34:48:70:66:b7:6d:72:c3:69:f9:39:
4b:68:39:56:bd:a1:b3:6d:f4:77:f3:46:5b:5c:19:ac:4f:b3:
74:6b:8c:c5:f1:89:cc:93:fe:0c:01:6f:88:17:dc:42:71:60:
e3:ed:73:30:42:9c:a9:2f:3b:a2:78:8e:c8:6f:ba:d1:13:0c:
d0:c7:5e:8c:10:fb:01:2e:37:9b:db:ac:f7:a1:ac:ba:7f:f8:
92:e7:cb:41:44:c8:15:f9:f3:c4:bb:ad:51:5f:be:de:c7:ac:
86:07:9f:40:ec:b9:0b:f6:b2:8b:cc:b5:55:33:66:ba:33:c2:
c4:f0:a2:e9


seed@VM: ~/.../k213881_Assignment-2
GNU nano 4.8
server cert.pem
-----BEGIN CERTIFICATE-----
MIIHbjCCBlagAwIBAgIQB1vO8waJyK3fE+Ua9K/hhzANBgkqhkiG9w0BAQsFADBZ
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMTMwMQYDVQQDEypE
aWdpQ2VydCBHbG9iYWwgRzIgVExTIFJTQSBTSEEyNTYgMjAyMCBDQTEwHhcNMjQw
MTMwMDAwMDAwWhcNMjUwMzAxMjM1OTU5WjCBljELMAkGA1UEBhMCVVMxEzARBgNV
BAgTCkNhbGlmb3JuaWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMUIwQAYDVQQKDDlJ
bnRlcm5ldMKgQ29ycG9yYXRpb27CoGZvcsKgQXNzaWduZWTCoE5hbWVzwqBhbmTC
oE51bWJlcnMxGDAWBgNVBAMTD3d3dy5leGFtcGxlLm9yZzCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAIaFD7sO+cpf2fXgCjIsM9mqDgcpqC8IrXi9wga/
9y0rpqcnPVOmTMNLsid3INbBVEm4CNr5cKlh9rJJnWlX2vttJDRyLkfwBD+dsVvi
vGYxWTLmqX6/1LDUZPVrynv/cltemtg/1Aay88jcj2ZaRoRmqBgVeacIzgU8+zmJ
7236TnFSe7fkoKSclsBhPaQKcE3Djs1uszJs8sdECQTdoFX9I6UgeLKFXtg7rRf/
hcW5dI0zubhXbrW8aWXbCzySVZn0c7RkJMpnTCiZzNxnPXnHFpwr5squqqjVyN/aB
KkjoP04Zmr+eRqoyk/+lslq0sS8eaYSSHbC5ja/yMWyVhvMCAwEAAaOCA/IwggPu
MB8GA1UdIwQYMBaAFHSFgMBmx9833s+9KTeqAx2+7c0XMB0GA1UdDgQWBBRM/tAS
TS4hz2v68vK4TEkCHTGRijCBgQYDVR0RBHoweIIPd3d3LmV4YW1wbGUub3Jnggtl
eGFtcGxlLm5ldIILZXhhbXBsZS5lZHWCC2V4YW1wbGUuY29tggtleGFtcGxlLm9y
Z4IPd3d3LmV4YW1wbGUuY29tgg93d3cuZXhhbXBsZS5lZHWCD3d3dy5leGFtcGxl
Lm5ldDA+BgNVHSAENzA1MDMGBmeBDAECAjApMCcGCCsGAQUFBwIBFhtodHRwOi8v
d3d3LmRpZ2ljZXJ0LmNvbS9DUFMwDgYDVR0PAQH/BAQDAgWgMB0GA1UdJQQWMBQG
CCsGAQUFBwMBBggrBgEFBQcDAjCBnwYDVR0fBIGXMIGUMEigRqBEhkJodHRwOi8v
Y3JsMy5kaWdpY2VydC5jb20vRGlnaUNlcnRHbG9iYWxHMlRMU1JTQVNIQTI1NjIw
MjBDQTEtMS5jcmwwSKBGoESGQmh0dHA6Ly9jcmw0LmRpZ2ljZXJ0LmNvbS9EaWdp
Q2VydEdsb2JhbEcyVExTUlNBU0hBMjU2MjAyMENBMS0xLmNybDCBhwYIKwYBBQUH
AQEEezB5MCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2VydC5jb20wUQYI
KwYBBQUHMAKGRWh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9EaWdpQ2VydEds
b2JhbEcyVExTUlNBU0hBMjU2MjAyMENBMS0xLmNydDAMBgNVHRMBAf8EAjAAMIIB
fQYKKwYBBAHWeQIEAgSCAW0EggFpAWcAdABOdaMnXJoQwzhbbNTfP1LrHfDgjhuN
acCx+mSxYpo53wAAAY1b0vxkAAAEAwBFMEMCH0BRCgxPbBBVxhcWZ26a8JCe83P1
JZ6wmv56GsVcyMACIDgpMbEo5HJITTRPnoyT4mG8cLrWjEvhchUdEcWUuk1TAHYA
fVkeEuF4KnscYWd8Xv340IdcFKBOlZ65Ay/ZDowuebgAAAGNW9L8MAAABAMARzBF
JZ6wmv56GsVcyMACIDgpMbEo5HJITTRPnoyT4mG8cLrWjEvhchUdEcWUuk1TAHYA
AiBdv5Z3pZFbfgoM3tGpCTM3ZxBMQsxBRSdTS6d8d2NAcwIhALLoCT9mTMN9OyFz
IBV5MkXVLyuTf2OAzAOa7d8x2H6XAHcA5tIxY0B3jMEQQQbXcbnOwdJA9paEhvu6

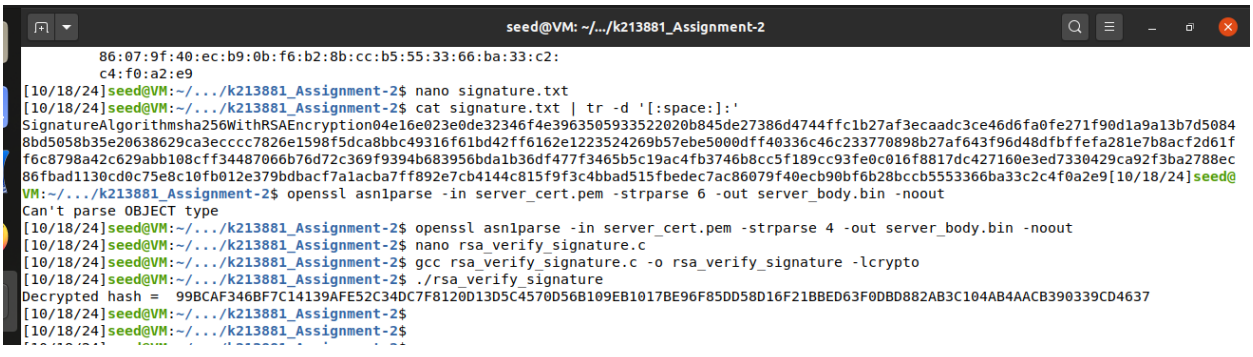

seed@VM: ~/.../k213881_Assignment-2
GNU nano 4.8 issuer_cert.pem
-----BEGIN CERTIFICATE-----
MIIEyDCCA7CgAwIBAgIQDPW9BitWAvR6uFAsI8zwZjANBgkqhkiG9w0BAQsFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBH
MjAeFw0yMTAzMzAwMDAwMDBaFw0zMTAzMjkyMzU5NTlaMFkxCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxMzAxBgNVBAMTKkRpZ2lDZXJ0IEdsb2Jh
bCBHMiBUTFMgUlNBIFNIQTI1NiAyMDIwIENBMTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMz3EGJPprtjb+2QUlbFbSd7ehJWivH0+dbn4Y+9lavyYEEV
cNsSAPonCrVXOFt9slGTcZUOakGUWzUb+nv6u8W+JDD+Vu/E832X4xT1FE3LpxDy
FuqrIvAxIhFhaZAmunjZlx/jfWardUSVc8is/+9dCopZQ+GssjoP80j812s3wWPc
3kbW20X+fSP9kOhRBx5Ro1/tSUZUfyyIxfQTnJcVPAPooTncaQwywa8WV0yUR0J8
osicfebUTVSvQpmowQTCd5zWSOTOEeAqgJnwQ3DPP3Zr0UxJqyRewg2C/Uaoq2yT
zGJSQnWS+Jr6Xl6ysGHlHx+5fwmY6D36g39HaaECAwEAAaOCAYIwggF+MBIGA1Ud
EwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHSFgMBmx9833s+9KTeqAx2+7c0XMB8G
A1UdIwQYMBaAFE4iVCAYlebjbuYP+vq5Eu0GF485MA4GA1UdDwEB/wQEAwIBhjAd
BgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwdgYIKwYBBQUHAQEEajBoMCQG
CCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2VydC5jb20wQAYIKwYBBQUHMAKG
NGh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9EaWdpQ2VydEdsb2JhbFJvb3RH
Mi5jcnQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybDMuZGlnaWNlcnQuY29t
L0RpZ2lDZXJ0R2xvYmFsUm9vdEcyLmNybDA9BgNVHSAENjA0MAsGCWCGSAGG/WwC
ATAHBgVngQwBATAIBgZngQwBAgEwCAYGZ4EMAQICMAgGBmeBDAECAzANBgkqhkiG
9w0BAQsFAAOCAQEAkPFwyyiXaZd8dP3A+iZ7U6utzWX9upwGnIrXWkOH7U1MVl+t
wcW1BSAuWdH/SvWgKtiwla3JLko716f2b4gp/DA/JIS7w7d7kwcsr4drdjPtAFVS
slme5LnQ89/nD/7d+MS5EHKBCQRfz5eeLjJ1js+aWNJXMX43AYGyZm0pGrFmCW3R
bpD0ufovARTFXFZkAdl9h6g4U5+LXUZtXMYnhIHUfoyMo5tS58aI7Dd8KvvwVVo4
chDYABPPTHPbqjc1qCmBaZx2vN4Ye5DUys/vZwP9BFohFrH/6j/f3IL16/RZkiMN
JCqVJUzKoZHm1Lesh3Sz8W2jmdv51b2EQJ8HmA==
-----END CERTIFICATE-----
[Wrote 28 lines]


seed@VM: ~/.../k213881_Assignment-2
86:07:9f:40:ec:b9:0b:f6:b2:8b:cc:b5:55:33:66:ba:33:c2:
c4:f0:a2:e9
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano signature.txt
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ cat signature.txt | tr -d '[:space:]:'
SignatureAlgorithmsha256WithRSAEncryption04e16e023e0de32346f4e3963505933522020b845de27386d4744ffc1b27af3ecaadc3ce46d6fa0fe271f90d1a9a13b7d5084
8bd5058b35e20638629ca3ecccc7826e1598f5dca8bbc49316f61bd42ff6162e1223524269b57ebe5000dff40336c46c233770898b27af643f96d48dfbffefa281e7b8acf2d61f
f6c8798a42c629abb108cff34487066b76d72c369f9394b683956bda1b36df477f3465b5c19ac4fb3746b8cc5f189cc93fe0c016f8817dc427160e3ed7330429ca92f3ba2788ec
86fbad1130cd0c75e8c10fb012e379bdbacf7a1acba7ff892e7cb4144c815f9f3c4bbad515fbedec7ac86079f40ecb90bf6b28bccb5553366ba33c2c4f0a2e9[10/18/24]seed@
VM:~/.../k213881_Assignment-2$ openssl asn1parse -in server_cert.pem -strparse 6 -out server_body.bin -noout
Can't parse OBJECT type
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ openssl asn1parse -in server_cert.pem -strparse 4 -out server_body.bin -noout
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano rsa_verify_signature.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc rsa_verify_signature.c -o rsa_verify_signature -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./rsa_verify_signature
Decrypted hash = 99BCAF346BF7C14139AFE52C34DC7F8120D13D5C4570D56B109EB1017BE96F85DD58D16F21BBED63F0DBD882AB3C104AB4AACB390339CD4637
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
[10/18/24]seed@VM:~/.../k213881_Assignment-2$


seed@VM: ~/.../k213881_Assignment-2
GNU nano 4.8 signature.txt
Signature Algorithm: sha256WithRSAEncryption
04:e1:6e:02:3e:0d:e3:23:46:f4:e3:96:35:05:93:35:22:02:
0b:84:5d:e2:73:86:d4:74:4f:fc:1b:27:af:3e:ca:ad:c3:ce:
46:d6:fa:0f:e2:71:f9:0d:1a:9a:13:b7:d5:08:48:bd:50:58:
b3:5e:20:63:86:29:ca:3e:cc:cc:78:26:e1:59:8f:5d:ca:8b:
bc:49:31:6f:61:bd:42:ff:61:62:e1:22:35:24:26:9b:57:eb:
e5:00:0d:ff:40:33:6c:46:c2:33:77:08:98:b2:7a:f6:43:f9:
6d:48:df:bf:fe:fa:28:1e:7b:8a:cf:2d:61:ff:6c:87:98:a4:
2c:62:9a:bb:10:8c:ff:34:48:70:66:b7:6d:72:c3:69:f9:39:
4b:68:39:56:bd:a1:b3:6d:f4:77:f3:46:5b:5c:19:ac:4f:b3:
74:6b:8c:c5:f1:89:cc:93:fe:0c:01:6f:88:17:dc:42:71:60:
e3:ed:73:30:42:9c:a9:2f:3b:a2:78:8e:c8:6f:ba:d1:13:0c:
d0:c7:5e:8c:10:fb:01:2e:37:9b:db:ac:f7:a1:ac:ba:7f:f8:
92:e7:cb:41:44:c8:15:f9:f3:c4:bb:ad:51:5f:be:de:c7:ac:
86:07:9f:40:ec:b9:0b:f6:b2:8b:cc:b5:55:33:66:ba:33:c2:
c4:f0:a2:e9
[Wrote 16 lines]


seed@VM: ~/.../k213881_Assignment-2
GNU nano 4.8 rsa_verify_signature.c Modified
//k213881
#include <openssl/bn.h>
#include <stdio.h>
void printBN(char *msg, BIGNUM *a) {
char * number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}
int main() {
BN_CTX *ctx = BN_CTX_new();
BIGNUM *n = BN_new(); // Modulus
BIGNUM *e = BN_new(); // Public exponent
BIGNUM *sig = BN_new(); // Signature
BIGNUM *decrypted_hash = BN_new(); // Decrypted signature (hash)
// Initialize the modulus and exponent (public key from the issuer certificate)
BN_hex2bn(&n, "CCF710624FA6B63C6FE9D95256C65DD277BA71256BAF49D6E7E18FBD95ABF260411570DB1200FA270AB557385B7D82519371950E6A41945B35 1FBE6A8
BN_set_word(e, 65537); // Exponent = 65537
// Initialize the signature (from server certificate)
BN_hex2bn(&sig, "04e16e023e0de32346f4e936503593352020b845de27386d4744ffc1b27af3ecaadc3ce46d6fa0fe271f90d1a9a13b7d5084bd5058b35e20");
// Decrypt the signature: decrypted_hash = sig^e mod n
BN_mod_exp(decrypted_hash, sig, e, n, ctx);
printBN("Decrypted hash = ", decrypted_hash);
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket


seed@VM: ~/.../k213881_Assignment-2
GNU nano 4.8 rsa_verify_signature.c
void printBN(char *msg, BIGNUM *a) {
char * number_str = BN_bn2hex(a);
printf("%s %s\n", msg, number_str);
OPENSSL_free(number_str);
}
int main() {
BN_CTX *ctx = BN_CTX_new();
BIGNUM *n = BN_new(); // Modulus
BIGNUM *e = BN_new(); // Public exponent
BIGNUM *sig = BN_new(); // Signature
BIGNUM *decrypted_hash = BN_new(); // Decrypted signature (hash)
// Initialize the modulus and exponent (public key from the issuer certificate)
BN_hex2bn(&n, "CCF710624FA6B63C6FE9D95256C65DD277BA71256BAF49D6E7E18FBD95ABF260411570DB1200FA270AB557385B7D82519371950E6A41945B35 1FBE6A87F0");
BN_set_word(e, 65537); // Exponent = 65537
// Initialize the signature (from server certificate)
BN_hex2bn(&sig, "04e16e023e0de32346f4e936503593352020b845de27386d4744ffc1b27af3ecaadc3ce46d6fa0fe271f90d1a9a13b7d5084bd5058b35e20");
// Decrypt the signature: decrypted_hash = sig^e mod n
BN_mod_exp(decrypted_hash, sig, e, n, ctx);
printBN("Decrypted hash = ", decrypted_hash);
// Free allocated memory
BN_free(n);
BN_free(e);
BN_free(sig);
BN_free(decrypted_hash);
BN_CTX_free(ctx);
return 0;
}
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-Q Previous ^B Back
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text ^Q Where Was M-W Next ^F Forward


seed@VM: ~/.../k213881_Assignment-2
86:07:9f:40:ec:b9:0b:f6:b2:8b:cc:b5:55:33:66:ba:33:c2:
c4:f0:a2:e9
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano signature.txt
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ cat signature.txt | tr -d '[:space:]:'
SignatureAlgorithmsha256WithRSAEncryption04e16e023e0de32346f4e3963505933522020b845de27386d4744ffc1b27af3ecaadc3ce46d6fa0fe271f90d1a9a13b7d5084
8bd5058b35e20638629ca3ecccc7826e1598f5dca8bbc49316f61bd42ff6162e1223524269b57ebe5000dff40336c46c233770898b27af643f96d48dfbffefa281e7b8acf2d61f
f6c8798a42c629abb108cff34487066b76d72c369f9394b683956bda1b36df477f3465b5c19ac4fb3746b8cc5f189cc93fe0c016f8817dc427160e3ed7330429ca92f3ba2788ec
86fbad1130cd0c75e8c10fb012e379bdbacf7a1acba7ff892e7cb4144c815f9f3c4bbad515fbedec7ac86079f40ecb90bf6b28bccb5553366ba33c2c4f0a2e9[10/18/24]seed@
VM:~/.../k213881_Assignment-2$ openssl asn1parse -in server_cert.pem -strparse 6 -out server_body.bin -noout
Can't parse OBJECT type
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ openssl asn1parse -in server_cert.pem -strparse 4 -out server_body.bin -noout
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ nano rsa_verify_signature.c
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ gcc rsa_verify_signature.c -o rsa_verify_signature -lcrypto
[10/18/24]seed@VM:~/.../k213881_Assignment-2$ ./rsa_verify_signature
Decrypted hash = 99BCAF346BF7C14139AFE52C34DC7F8120D13D5C4570D56B109EB1017BE96F85DD58D16F21BBED63F0DBD882AB3C104AB4AACB390339CD4637
[10/18/24]seed@VM:~/.../k213881_Assignment-2$
[10/18/24]seed@VM:~/.../k213881_Assignment-2$


seed@VM: ~/.../k213881_Assignment-2
Files
GNU nano 4.8 server_body.bin
[Read 17 lines (Converted from Mac format)]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket