Task 1

CryptoRSA1.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char *msg, BIGNUM * a)
{
    char * number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
void computer_privatekey(){
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BIGNUM *n = BN_new();

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

    BIGNUM *totient = BN_new();
    BIGNUM *p_minus_one = BN_new();
    BIGNUM *q_minus_one = BN_new();

    BIGNUM *one = BN_new();
    BN_hex2bn(&one, "1");

    BN_sub(p_minus_one, p, one);
    BN_sub(q_minus_one, q, one);

    BN_mul(totient,p_minus_one,q_minus_one,ctx);
    printBN("totient = (p-1)*(q-1) = ", totient);

    BIGNUM *d = BN_new();

    BN_mod_inverse(d,e,totient, ctx);
    printBN("private key d = ", d);
}
int main ()
{
    computer_privatekey();
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA1.c -o Task1 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task1
n = p * q = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
totient = (p-1)*(q-1) =  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
private key d =  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
```

Task 2

CryptoRSA2.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *m = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *enc = BN_new();
    BIGNUM *dec = BN_new();

    BN_hex2bn(&m, "4120746f702073656372657421");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    BN_mod_exp(enc,m,e,n,ctx);
    printBN("Encrypted Message = ", enc);

    BN_mod_exp(dec,enc,d,n,ctx);
    printBN("Decrypted Message = ", dec);

    return 0;
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("4120746f702073656372657421".decode("hex"))'
A top secret!
```

Task 3

CryptoRSA3.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *c = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *enc = BN_new();
    BIGNUM *dec = BN_new();

    BN_hex2bn(&c, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");


    BN_mod_exp(dec,c,d,n,ctx);
    printBN("Decrypted Message = ", dec);

    return 0;
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ subl CryptoRSA3.c

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA3.c -o Task3 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task3
Decrypted Message =  50617373776F72642069732064656573

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("50617373776F72642069732064656573".decode("hex"))'
Password is dees
```

Task 4

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("I owe you $2000.".encode("hex"))'
49206f776520796f752024323030302e
```

CryptoRSA4.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *m = BN_new();
    BIGNUM *sign = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    BN_hex2bn(&m, "49206f776520796f752024323030302e");

    BN_mod_exp(sign,m,d,n,ctx);
    printBN("Signed Message = ", sign);

    return 0;
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA4.c -o Task4 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task4
Signed Message =  55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
```

Changing from $2000 to $3000

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("I owe you $3000.".encode("hex"))'
49206f776520796f752024333030302e

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA4.c -o Task4 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task4
Signed Message =  BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
```

CryptoRSA4.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *m = BN_new();
    BIGNUM *sign = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    // BN_hex2bn(&m, "49206f776520796f752024323030302e"); // $2000
    BN_hex2bn(&m, "49206f776520796f752024333030302e"); // $3000

    BN_mod_exp(sign,m,d,n,ctx);
    printBN("Signed Message = ", sign);

    return 0;
}
```

Task 5

CryptoRSA5.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *message = BN_new();

    BN_hex2bn(&s, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
    BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
    BN_hex2bn(&e, "10001");

    BN_mod_exp(message,s,e,n,ctx);
    printBN("Message = ", message);
    return 0;
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA5.c -o Task5 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task5
Message =  4C61756E63682061206D697373696C652E

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("4C61756E63682061206D697373696C652E".decode("hex"))'
Launch a missile.
```

Changing from 2F to 3F

CryptoRSA5.c

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *message = BN_new();

    BN_hex2bn(&s, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");
    BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
    BN_hex2bn(&e, "10001");

    BN_mod_exp(message,s,e,n,ctx);
    printBN("Message = ", message);
    return 0;
}
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA5.c -o Task5 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task5
Message =  91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ python2 -c 'print("91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294".decode("hex"))'
◆◆,O◆c◆◆◆rm=f◆:N◆◆◆◆
```

Task 6

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl s_client -connect www.tryhackme.com:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X2
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = E1
verify return:1
depth=0 CN = tryhackme.com
verify return:1
---
Certificate chain
 0 s:CN = tryhackme.com
   i:C = US, O = Let's Encrypt, CN = E1
   a:PKEY: id-ecPublicKey, 256 (bit); sigalg: ecdsa-with-SHA384
   v:NotBefore: Sep 21 02:09:33 2023 GMT; NotAfter: Dec 20 02:09:32 2023 GMT
-----BEGIN CERTIFICATE-----
MIIDjDCCAxKgAwIBAgISAwg749s8VcXcTsesY0RAdCNjMAoGCCqGSM49BAMDMDIx
CzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQDEwJF
MTAeFw0yMzA5MjEwMjA5MzNaFw0yMzEyMjAwMjA5MzJaMBgxFjAUBgNVBAMTDXRy
eWhhY2ttZS5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARdNFNKlRwgkXfP
11P3xmprt18VPWq29eZs6llCrdsN7Ixw0kAUQXYzrxjjLKk+lg9epwmrK1XiUfpQ
i+aWJSYWo4ICIDCCAhwwDgYDVR0PAQH/BAQDAgeAMB0GA1UdJQQWMBQGCCsGAQUF
BwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBT+mkTSBJh3O5oi
OaIn2AvY3dR7kzAfBgNVHSMEGDAWgBRa8+0r/DbCN3m5UjDqVG/PVcsurDBVBggr
BgEFBQcBAQRJMEcwIQYIKwYBBQUHMAGGFWh0dHA6Ly9lMS5vLmxlbmNyLm9yZzAi
BggrBgEFBQcwAoYWaHR0cDovL2UxLmkubGVuY3Iub3JnLzApBgNVHREEIjAggg8q
LnRyeWhhY2ttZS5jb22CDXRyeWhhY2ttZS5jb20wEwYDVR0gBAwwCjAIBgZngQwB
AgEwggEEBgorBgEEAdZ5AgQCBIH1BIHyAPAAdQB6MoxU2LcttiDqOOBSHumEFnAy
E4VNO9IrwTpXo1LrUgAAAYq1ttcMAAAEAwBGMEQCIB72SVnMWiiLUXCfcoatddkG
EDgVn/A+ASC5ZpsqpytDAiBC4aTOYUUFI/21loxf/ZtNmzK4u4MICQvwqDskge1Z
VwB3AOg+0No+9QY1MudXKLyJa8kD08vREWvs62nhd31tBr1uAAABirW21vAAAAQD
AEgwRgIhAOF8+NTH+xApb8Dtq8XSX1PgTDiN7+16CpYTTyrpjHXFAiEA1p/j70xJ
```

CryptoRSA6.c | Signature | Certificate1.pem | Certificate2.pem | Certificate3.pem | Certificate4.pem

```
-----BEGIN CERTIFICATE-----
MIIDjDCCAxKgAwIBAgISAwg749s8VcXcTsesY0RAdCNjMAoGCCqGSM49BAMDMDIx
CzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQDEwJF
MTAeFw0yMzA5MjEwMjA5MzNaFw0yMzEyMjAwMjA5MzJaMBgxFjAUBgNVBAMTDXRy
eWhhY2ttZS5jb20wWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARdNFNKlRwgkXfP
11P3xmprt18VPWq29eZs6llCrdsN7Ixw0kAUQXYzrxjjLKk+lg9epwmrK1XiUfpQ
i+aWJSYWo4ICIDCCAhwwDgYDVR0PAQH/BAQDAgeAMB0GA1UdJQQWMBQGCCsGAQUF
BwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBT+mkTSBJh3O5oi
OaIn2AvY3dR7kzAfBgNVHSMEGDAWgBRa8+0r/DbCN3m5UjDqVG/PVcsurDBVBggr
BgEFBQcBAQRJMEcwIQYIKwYBBQUHMAGGFWh0dHA6Ly9lMS5vLmxlbmNyLm9yZzAi
BggrBgEFBQcwAoYWaHR0cDovL2UxLmkubGVuY3Iub3JnLzApBgNVHREEIjAggg8q
LnRyeWhhY2ttZS5jb22CDXRyeWhhY2ttZS5jb20wEwYDVR0gBAwwCjAIBgZngQwB
AgEwggEEBgorBgEEAdZ5AgQCBIH1BIHyAPAAdQB6MoxU2LcttiDqOOBSHumEFnAy
E4VNO9IrwTpXo1LrUgAAAYq1ttcMAAAEAwBGMEQCIB72SVnMWiiLUXCfcoatddkG
EDgVn/A+ASC5ZpsqpytDAiBC4aTOYUUFI/21loxf/ZtNmzK4u4MICQvwqDskge1Z
VwB3AOg+0No+9QY1MudXKLyJa8kD08vREWvs62nhd31tBr1uAAABirW21vAAAAQD
AEgwRgIhAOF8+NTH+xApb8Dtq8XSX1PgTDiN7+16CpYTTyrpjHXFAiEA1p/j70xJ
thW0cs9H2ETjwqFnls4SC2y0QlVsO6Jbji0wCgYIKoZIzj0EAwMDaAAwZQIwQJRg
MSDaHAFKRIqMFmTf0veIy8z90bGR20jFa0+7rvLo7OdJar/fvGwA4h++tShdAjEA
t8i+Q5B5XtUVZoFtEzKeJEberp4TMW1sDHBYU1IFpkxg35oTNkL5uUe/PHe39NID
-----END CERTIFICATE-----
```

CryptoRSA6.c | Signature | Certificate1.pem | Certificate2.pem | Certificate3.pem | Certificate4.pem

```
-----BEGIN CERTIFICATE-----
MIICxjCCAk2gAwIBAgIRALO93/inhFu86QOgQTWzSkUwCgYIKoZIzj0EAwMwTzEL
MAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2VhcmNo
IEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDIwHhcNMjAwOTA0MDAwMDAwWhcN
MjUwOTE1MTYwMDAwWjAyMQswCQYDVQQGEwJVUzEWMBQGA1UEChMNTGV0J3MgRW5j
cnlwdDELMAkGA1UEAxMCRTEwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQkXC2iKv0c
S6Zdl3MnMayyoGli72XoprDwrEuf/xwLcA/TmC9N/A8AmzfwdAVXMpcuBe8qQyWj
+240JxP2T35p0wKZXuskR5LBJJvmsSGPwSSB/GjMH2m6WPUZIvd0xhajggEIMIIB
BDAOBgNVHQ8BAf8EBAMCAYYwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMB
MBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFFrz7Sv8NsI3eblSMOpUb89V
yy6sMB8GA1UdIwQYMBaAFHxClq7eS0g7+pL4nozPbYupcjeVMDIGCCsGAQUFBwEB
BCYwJDAiBggrBgEFBQcwAoYWaHR0cDovL3gyLmkubGVuY3Iub3JnLzAnBgNVHR8E
IDAeMBygGqAYhhZodHRwOi8veDIuYy5sZW5jci5vcmcvMCIGA1UdIAQbMBkwCAYG
Z4EMAQIBMA0GCysGAQQBgt8TAQEBMAoGCCqGSM49BAMDA2cAMGQCMHt01VITjWH+
Dbo/AwCd89eYhNlXLr3pD5xcSAQh8suzYHKOl9YST8pE9kLJ03uGqQIwWrGxtO3q
YJkgsTgDyj2gJrjubi1K9sZmHzOa25JK1fUpE8ZwYii6I4zPPS/Lgul/
-----END CERTIFICATE-----
```

CryptoRSA6.c | Signature | Certificate1.pem | Certificate2.pem | Certificate3.pem | Certificate4.pem

```
-----BEGIN CERTIFICATE-----
MIIEYDCCAkigAwIBAgIQB55JKIY3b9QISMI/xjHkYzANBgkqhkiG9w0BAQsFADBP
MQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJuZXQgU2VjdXJpdHkgUmVzZWFy
Y2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBYMTAeFw0yMDA5MDQwMDAwMDBa
Fw0yNTA5MTUxNjAwMDBaME8xCzAJBgNVBAYTAlVTMSkwJwYDVQQKEyBJbnRlcm5l
dCBTZWN1cml0eSBSZXNlYXJjaCBHcm91cDEVMBMGA1UEAxMMSVNSRyBSb290IFgy
MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEzZvVn4CDCuwJSvMWSj5cz3es3mcFDR0H
ttwW+1qLFNvicWDEukWVEYmO6gbf9yoWHKS5xcUy4APgHoIYOIvXRdgKam7mAHf7
AlF9ItgKbppbd9/w+kHsOdx1ymgHDB/qo4HlMIHiMA4GA1UdDwEB/wQEAwIBBjAP
BgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBR8Qpau3ktIO/qS+J6Mz22LqXI3lTAf
BgNVHSMEGDAWgBR5tFnme7bl5AFzgAiIyBpY9umbbjAyBggrBgEFBQcBAQQmMCQw
IgYIKwYBBQUHMAKGFmh0dHA6Ly94MS5pLmxlbmNyLm9yZy8wJwYDVR0fBCAwHjAc
oBqgGIYWaHR0cDovL3gxLmMubGVuY3Iub3JnLzAiBgNVHSAEGzAZMAgGBmeBDAEC
ATANBgsrBgEEAYLfEwEBATANBgkqhkiG9w0BAQsFAAOCAgEAG38lK5B6CHYAdxjh
wy6KNkxBfr8XS+Mw11sMfpyWmG97sGjAJETM4vL80erb0p8B+RdNDJ1V/aWtbdIv
P0tywC6uc8clFlfCPhWt4DHRCoSEbGJ4QjEiRhrtekC/lxaBRHfKbHtdIVwH8hGR
Ib/hL8Lvbv0FIOS093nzLbs3KvDGsaysUfUfs1oeZs5YBxg4f3GpPIO617yCnpp2
D56wKf3L84kHSBv+q5MuFCENX6+Ot1SrXQ7UW0xx0JLqPaM2m3wf4DtVudhTU8yD
ZrtK3IEGABiL9LPXSLETQbnEtp7PLHeOQiALgH6fxatI27xvBI1sRikCDXCKHfES
c7ZGJEKeKhcY46zHmMJyzG0tdm3dLCsmlqXPIQgb5dovy++fc5Ou+DZfR4+XKM6r
4pgmmIv97igyIintTJUJxCD6B+GGLET2gUfA5GIy7R3YPEiIlsNekbave1mk7uOG
nMeIWMooKmZVm4WAuR3YQCvJHBM8qevemcIWQPb1pK4qJWxSuscETLQyu/w4XKAM
YXtX7HdOUM+vBqIPN4zhDtLTLxq9nHE+zOH40aijvQT2GcD5hq/1DhqqlWvvykdx
S2McTZbbVSMKnQ+BdaDmQPVkRgNuzvpqfQbspDQGdNpT2Lm4xiN9qfgqLaSCpi4t
EcrmzTFYeYXmchynn9NM0GbQp7s=
-----END CERTIFICATE-----
```

CryptoRSA6.c | Signature | Certificate1.pem | Certificate2.pem | Certificate3.pem | Certificate4.pem

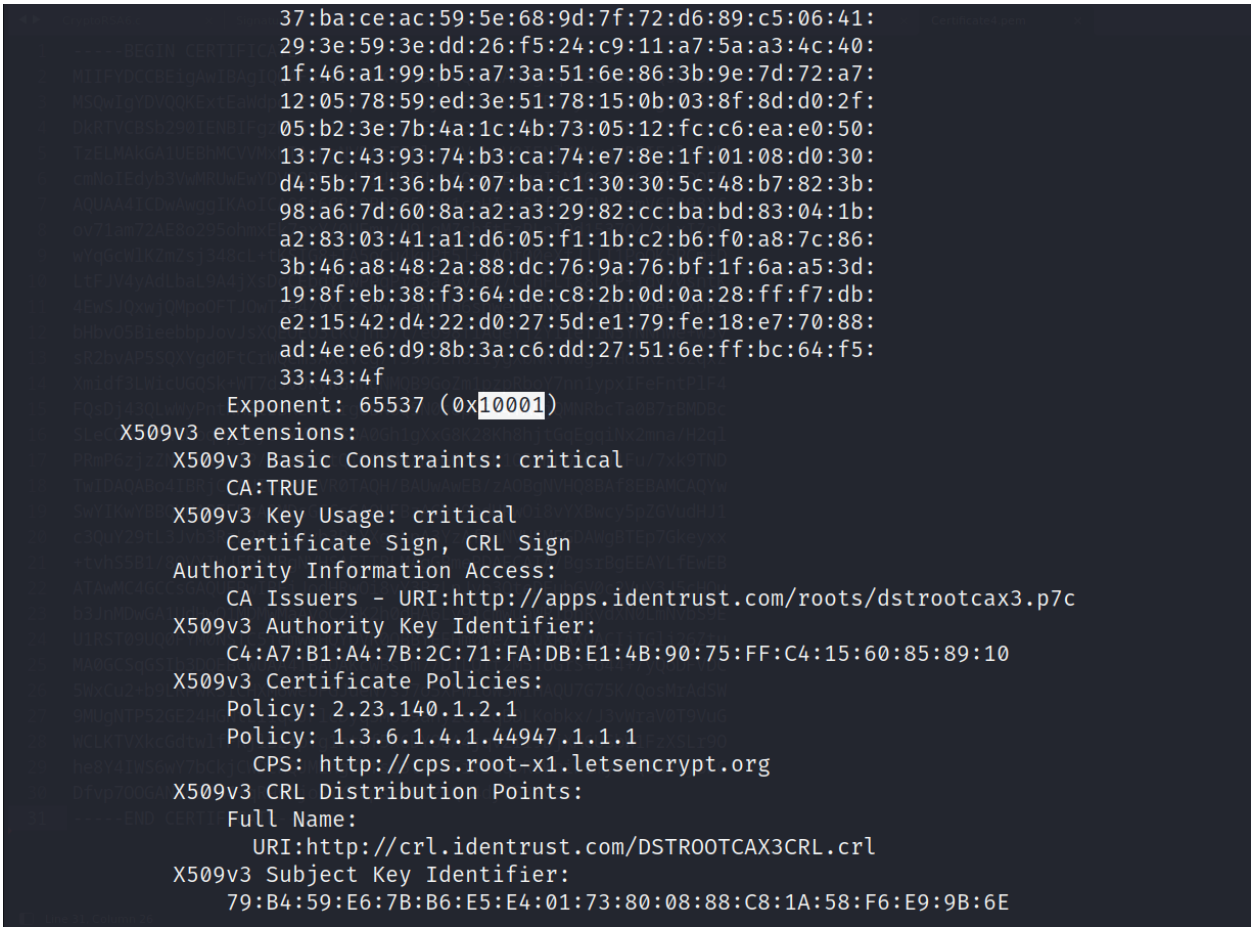
```
-----BEGIN CERTIFICATE-----
MIIFYDCCBEigAwIBAgIQQAF3ITfU6UK47naqPGQKtzANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTIxMDEyMDE5MTQwM1oXDTI0MDkzMDE4MTQwM1ow
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwggIiMA0GCSqGSIb3DQEB
AQUAA4ICDwAwggIKAoICAQCt6CRz9BQ385ueK1coHIe+3LffOJCMbjzmV6B493XC
ov71am72AE8o295ohmxEk7axY/0UEmu/H9LqMZshftEzPLpI9d1537O4/xLxIZpL
wYqGcWlKZmZsj348cL+tKSIG8+TA5oCu4kuPt5l+lAOf00eXfJlII1PoOK5PCm+D
LtFJV4yAdLbaL9A4jXsDcCEbdfIwPPqPrt3aY6vrFk/CjhFLfs8L6P+1dy70sntK
4EwSJQxwjQMpoOFTJOwT2e4ZvxCzSow/iaNhUd6shweU9GNx7C7ib1uYgeGJXDR5
bHbvO5BieebbpJovJsXQEOEO3tkQjhb7t/eo98flAgeYjzYIlefiN5YNNnWe+w5y
sR2bvAP5SQXYgd0FtCrWQemsAXaVCg/Y39W9Eh81LygXbNKYwagJZHduRze6zqxZ
Xmidf3LWicUGQSk+WT7dJvUkyRGnWqNMQB9GoZm1pzpRboY7nn1ypxIFeFntPlF4
FQsDj43QLwWyPntKHEtzBRL8xurgUBN8Q5N0s8p0544fAQjQMNRbcTa0B7rBMDBc
SLeCO5imfWCKoqMpgsy6vYMEG6KDA0Gh1gXxG8K28Kh8hjtGqEgqiNx2mna/H2ql
PRmP6zjzZN7IKw0KKP/32+IVQtQi0Cdd4Xn+GOdwiK1O5tmLOsbdJ1Fu/7xk9TND
TwIDAQABo4IBRjCCAUIwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYw
SwYIKwYBBQUHAQEEPzA9MDsGCCsGAQUFBzAChi9odHRwOi8vYXBwcy5pZGVudHJ1
c3QuY29tL3Jvb3RzL2RzdHJvb3RjYXgzLnA3YzAfBgNVHSMEGDAWgBTEp7Gkeyxx
+tvhS5B1/8QVYIWJEDBUBgNVHSAETTBLMAgGBmeBDAECATA/BgsrBgEEAYLfEwEB
ATAwMC4GCCsGAQUFBwIBFiJodHRwOi8vY3BzLnJvb3QteDEubGV0c2VuY3J5cHQu
b3JnMDwGA1UdHwQ1MDMwMaAvoC2GK2h0dHA6Ly9jcmwuaWRlbnRydXN0LmNvbS9E
U1RST09UQ0FYM0NSTC5jcmwwHQYDVR0OBBYEFHm0WeZ7tuXkAXOACIjIGlj26Ztu
MA0GCSqGSIb3DQEBCwUAA4IBAQAKcwBslm7/DlLQrt2M51oGrS+o44+/yQoDFVDC
5WxCu2+b9LRPwkSICHXM6webFGJueN7sJ7o5XPWioW5WlHAQU7G75K/QosMrAdSW
9MUgNTP52GE24HGNtLi1qoJFlcDyqSMo59ahy2cI2qBDLKobkx/J3vWraV0T9VuG
WCLKTVXkcGdtwlfFRjlBz4pYg1htmf5X6DYO8A4jqv2Il9DjXA6USbW1FzXSLr9O
he8Y4IWS6wY7bCkjCWDcRQJMEhg76fsO3txE+FiYruq9RUWhiF1myv4Q6W+CyBFC
Dfvp7OOGAN6dEOM4+qR9sdjoSYKEBpsr6GtPAQw4dy753ec5
-----END CERTIFICATE-----
```

n value to be copied in CryptoRSA6.c

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate1.pem -noout -modulus
Modulus=No modulus for this public key type

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate2.pem -noout -modulus
Modulus=No modulus for this public key type

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate3.pem -noout -modulus
Modulus=No modulus for this public key type

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate4.pem -noout -modulus
Modulus=ADE82473F41437F39B9E2B57281C87BEDCB7DF38908C6E3CE657A078F775C2A2FEF56A6EF6004F28DBDE68866C4493B6B163FD14126BBF1FD2EA319B217ED1333CBA48F5D
D79DFB3B8FF12F1219A4BC18A8671694A66666C8F7E3C70BFAD292206F3E4C0E680AEE24B8FB7997E94039FD347977C99482353E838AE4F0A6F832ED149578C8074B6DA2FD0388D7B
0370211B75F2303CFA8FAEDDDA63ABEB164FC28E114B7ECF0BE8FFB5772EF4B27B4AE04C12250C708D0329A0E15324EC13D9EE19BF10B34A8C3F89A36151DEAC870794F46371EC2EE
26F5B9881E1895C34796C76EF3B906279E6DBA49A2F26C5D010E10EDED9108E16FBB7F7A8F7C7E50207988F360895E7E237960D36759EFB0E72B11D9BBC03F94905D881DD05B42AD6
41E9AC0176950A0FD8DFD5BD121F352F28176CD298C1A80964776E4737BACEAC595E689D7F72D689C50641293E593EDD26F524C911A75AA34C401F46A199B5A73A516E863B9E7D72A
712057859ED3E5178150B038F8DD02F05B23E7B4A1C4B730512FCC6EAE050137C439374B3CA74E78E1F0108D030D45B7136B407BAC130305C48B7823B98A67D608AA2A32982CCBABD
83041BA2830341A1D605F11BC2B6F0A87C863B46A8482A88DC769A76BF1F6AA53D198FEB38F364DEC82B0D0A28FFF7DBE21542D422D0275DE179FE18E77088AD4EE6D98B3AC6DD275
16EFFBC64F533434F
```

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate4.pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            40:01:77:21:37:d4:e9:42:b8:ee:76:aa:3c:64:0a:b7
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O = Digital Signature Trust Co., CN = DST Root CA X3
        Validity
            Not Before: Jan 20 19:14:03 2021 GMT
            Not After : Sep 30 18:14:03 2024 GMT
        Subject: C = US, O = Internet Security Research Group, CN = ISRG Root X1
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:ad:e8:24:73:f4:14:37:f3:9b:9e:2b:57:28:1c:
                    87:be:dc:b7:df:38:90:8c:6e:3c:e6:57:a0:78:f7:
                    75:c2:a2:fe:f5:6a:6e:f6:00:4f:28:db:de:68:86:
                    6c:44:93:b6:b1:63:fd:14:12:6b:bf:1f:d2:ea:31:
                    9b:21:7e:d1:33:3c:ba:48:f5:dd:79:df:b3:b8:ff:
                    12:f1:21:9a:4b:c1:8a:86:71:69:4a:66:66:6c:8f:
                    7e:3c:70:bf:ad:29:22:06:f3:e4:c0:e6:80:ae:e2:
                    4b:8f:b7:99:7e:94:03:9f:d3:47:97:7c:99:48:23:
                    53:e8:38:ae:4f:0a:6f:83:2e:d1:49:57:8c:80:74:
                    b6:da:2f:d0:38:8d:7b:03:70:21:1b:75:f2:30:3c:
                    fa:8f:ae:dd:da:63:ab:eb:16:4f:c2:8e:11:4b:7e:
                    cf:0b:e8:ff:b5:77:2e:f4:b2:7b:4a:e0:4c:12:25:
                    0c:70:8d:03:29:a0:e1:53:24:ec:13:d9:ee:19:bf:
                    10:b3:4a:8c:3f:89:a3:61:51:de:ac:87:07:94:f4:
                    63:71:ec:2e:e2:6f:5b:98:81:e1:89:5c:34:79:6c:
```

e value to be copied in CryptoRSA6.c

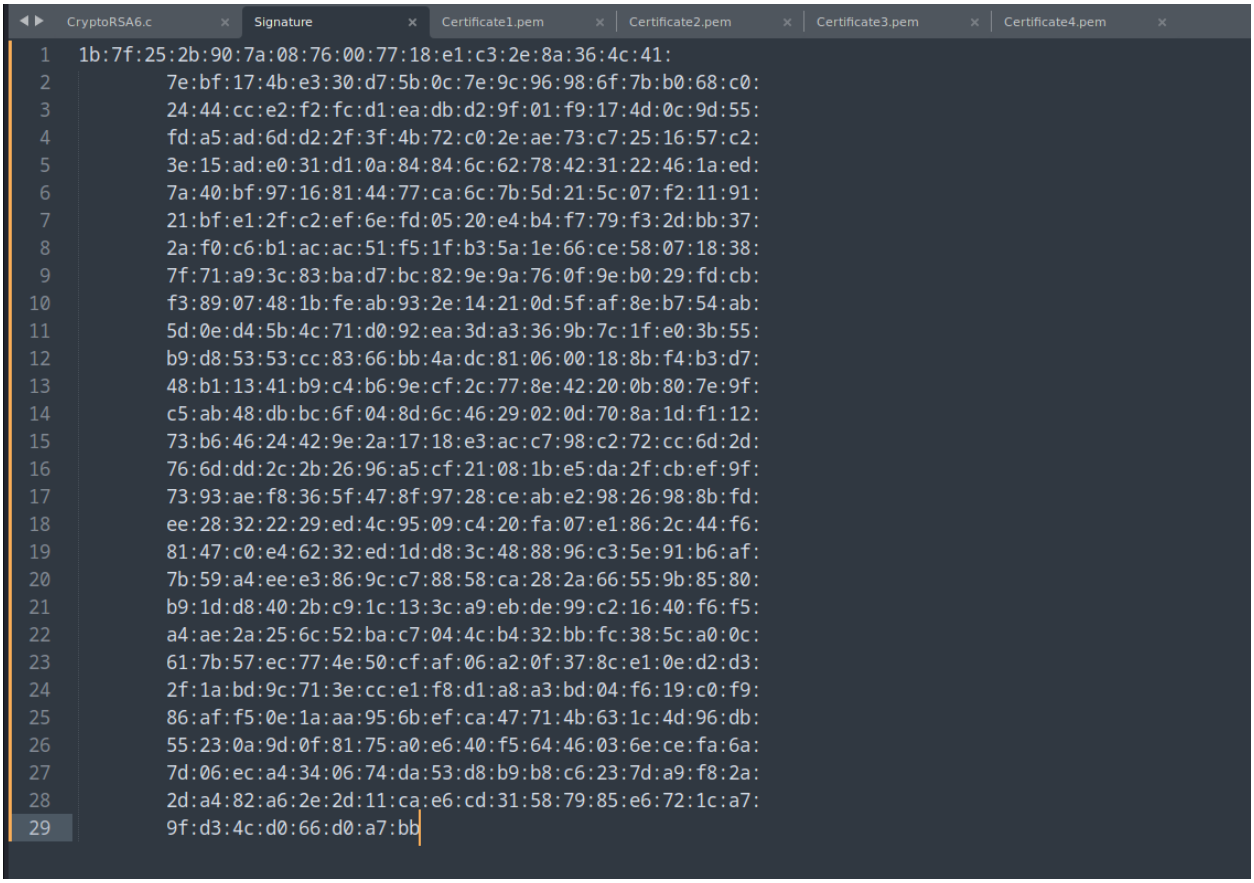


```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl x509 -in Certificate3.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            07:9e:49:28:86:37:6f:d4:08:48:c2:3f:c6:31:e4:63
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Internet Security Research Group, CN = ISRG Root X1
        Validity
            Not Before: Sep  4 00:00:00 2020 GMT
            Not After : Sep 15 16:00:00 2025 GMT
        Subject: C = US, O = Internet Security Research Group, CN = ISRG Root X2
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:cd:9b:d5:9f:80:83:0a:ec:09:4a:f3:16:4a:3e:
                    5c:cf:77:ac:de:67:05:0d:1d:07:b6:dc:16:fb:5a:
                    8b:14:db:e2:71:60:c4:ba:45:95:11:89:8e:ea:06:
                    df:f7:2a:16:1c:a4:b9:c5:c5:32:e0:03:e0:1e:82:
                    18:38:8b:d7:45:d8:0a:6a:6e:e6:00:77:fb:02:51:
                    7d:22:d8:0a:6e:9a:5b:77:df:f0:fa:41:ec:39:dc:
                    75:ca:68:07:0c:1f:ea
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
```

Signature value taken from Certificate3.pem

```
            Policy: 2.23.140.1.2.1
            Policy: 1.3.6.1.4.1.44947.1.1.1
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        1b:7f:25:2b:90:7a:08:76:00:77:18:e1:c3:2e:8a:36:4c:41:
        7e:bf:17:4b:e3:30:d7:5b:0c:7e:9c:96:98:6f:7b:b0:68:c0:
        24:44:cc:e2:f2:fc:d1:ea:db:d2:9f:01:f9:17:4d:0c:9d:55:
        fd:a5:ad:6d:d2:2f:3f:4b:72:c0:2e:ae:73:c7:25:16:57:c2:
        3e:15:ad:e0:31:d1:0a:84:84:6c:62:78:42:31:22:46:1a:ed:
        7a:40:bf:97:16:81:44:77:ca:6c:7b:5d:21:5c:07:f2:11:91:
        21:bf:e1:2f:c2:ef:6e:fd:05:20:e4:b4:f7:79:f3:2d:bb:37:
        2a:f0:c6:b1:ac:ac:51:f5:1f:b3:5a:1e:66:ce:58:07:18:38:
        7f:71:a9:3c:83:ba:d7:bc:82:9e:9a:76:0f:9e:b0:29:fd:cb:
        f3:89:07:48:1b:fe:ab:93:2e:14:21:0d:5f:af:8e:b7:54:ab:
        5d:0e:d4:5b:4c:71:d0:92:ea:3d:a3:36:9b:7c:1f:e0:3b:55:
        b9:d8:53:53:cc:83:66:bb:4a:dc:81:06:00:18:8b:f4:b3:d7:
        48:b1:13:41:b9:c4:b6:9e:cf:2c:77:8e:42:20:0b:80:7e:9f:
        c5:ab:48:db:bc:6f:04:8d:6c:46:29:02:0d:70:8a:1d:f1:12:
        73:b6:46:24:42:9e:2a:17:18:e3:ac:c7:98:c2:72:cc:6d:2d:
        76:6d:dd:2c:2b:26:96:a5:cf:21:08:1b:e5:da:2f:cb:ef:9f:
        73:93:ae:f8:36:5f:47:8f:97:28:ce:ab:e2:98:26:98:8b:fd:
        ee:28:32:22:29:ed:4c:95:09:c4:20:fa:07:e1:86:2c:44:f6:
        81:47:c0:e4:62:32:ed:1d:d8:3c:48:88:96:c3:5e:91:b6:af:
        7b:59:a4:ee:e3:86:9c:c7:88:58:ca:28:2a:66:55:9b:85:80:
        b9:1d:d8:40:2b:c9:1c:13:3c:a9:eb:de:99:c2:16:40:f6:f5:
        a4:ae:2a:25:6c:52:ba:c7:04:4c:b4:32:bb:fc:38:5c:a0:0c:
        61:7b:57:ec:77:4e:50:cf:af:06:a2:0f:37:8c:e1:0e:d2:d3:
        2f:1a:bd:9c:71:3e:cc:e1:f8:d1:a8:a3:bd:04:f6:19:c0:f9:
        86:af:f5:0e:1a:aa:95:6b:ef:ca:47:71:4b:63:1c:4d:96:db:
        55:23:0a:9d:0f:81:75:a0:e6:40:f5:64:46:03:6e:ce:fa:6a:
        7d:06:ec:a4:34:06:74:da:53:d8:b9:b8:c6:23:7d:a9:f8:2a:
        2d:a4:82:a6:2e:2d:11:ca:e6:cd:31:58:79:85:e6:72:1c:a7:
        9f:d3:4c:d0:66:d0:a7:bb
```

Copy in Sublime text and save the file as Signature



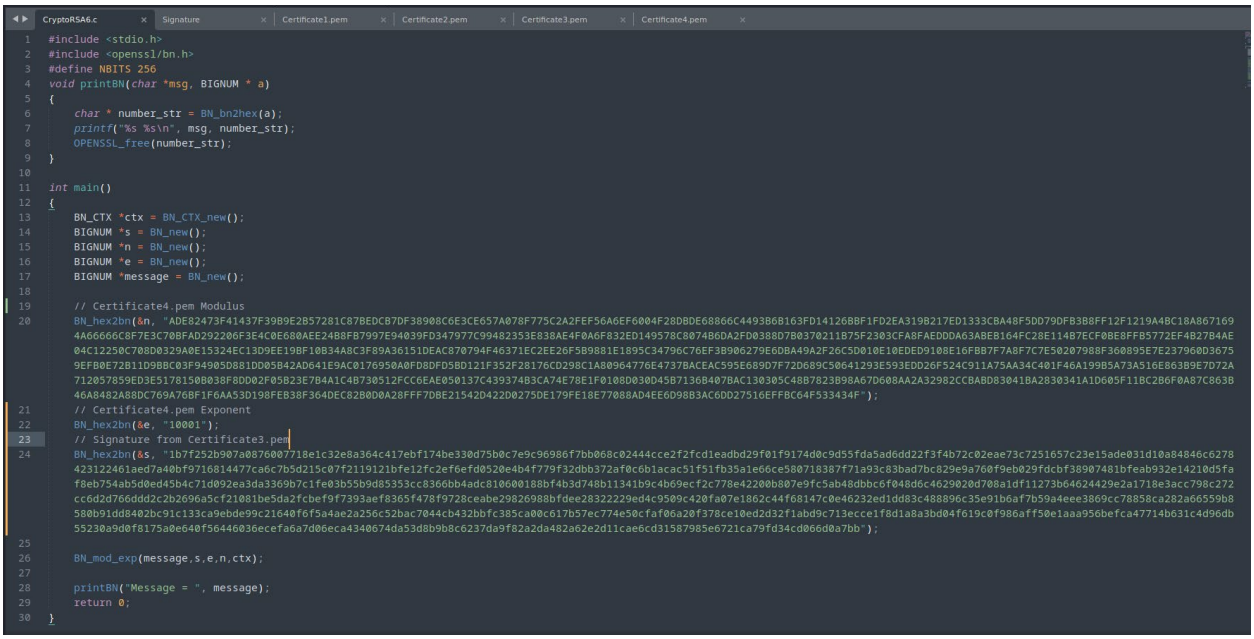
Sha256Sum to Certificate3_body.bin, found that is same as compiling Task6

```
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ cat Signature | tr -d '[:space:]:'
1b7f252b907a0876007718e1c32e8a364c417ebf174be330d75b0c7e9c96986f7bb068c02444cce2f2fcd1eadbd29f01f9174d0c9d55fda5ad6dd22f3f4b72c02eae73c7251657c23
e15ade031d10a84846c6278423122461aed7a40bf9716814477ca6c7b5d215c07f2119121bfe12fc2ef6efd0520e4b4f779f32dbb372af0c6b1acac51f51fb35a1e66ce580718387f
71a93c83bad7bc829e9a760f9eb029fdcbf38907481bfeab932e14210d5faf8eb754ab5d0ed45b4c71d092ea3da3369b7c1fe03b55b9d85353cc8366bb4adc810600188bf4b3d748b
11341b9c4b69ecf2c778e42200b807e9fc5ab48dbbc6f048d6c4629020d708a1df11273b64624429e2a1718e3acc798c272cc6d2d766ddd2c2b2696a5cf21081be5da2fcbef9f7393
aef8365f478f9728ceabe29826988bfdee28322229ed4c9509c420fa07e1862c44f68147c0e46232ed1dd83c488896c35e91b6af7b59a4eee3869cc78858ca282a66559b8580b91dd
8402bc91c133ca9ebde99c21640f6f5a4ae2a256c52bac7044cb432bbfc385ca00c617b57ec774e50cfaf06a20f378ce10ed2d32f1abd9c713ecce1f8d1a8a3bd04f619c0f986aff5
0e1aaa956befca47714b631c4d96db55230a9d0f8175a0e640f56446036ecefa6a7d06eca4340674da53d8b9b8c6237da9f82a2da482a62e2d11cae6cd31587985e6721ca79fd34cd
066d0a7bb
┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ openssl asn1parse -i -in Certificate3.pem -strparse 4 -out Certificate3_body.bin -noout

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ sha256sum Certificate3_body.bin
5aef843ffcf2ec7055f504a162f229f8391c370ff3a6163d2db3f3d604d622be  Certificate3_body.bin

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ gcc CryptoRSA6.c -o Task6 -lcrypto

┌──(UWLGroup2㉿kali)-[~/Desktop/CryptoRSA Lab UWL]
└─$ ./Task6
Message =  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D0609608648016503040201050004205AEF843FFCF2EC7055F504A162F229F8391C370FF3A616
3D2DB3F3D604D622BE
```

n copied from openssl x509 -in Certificate4.pem -noout -modulus

e copied from Exponent of openssl x509 -in Certificate4.pem -noout -text

s copied from cat Signature | tr -d '[:space:]:' of Certificate3.pem



```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
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 *s = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *message = BN_new();

    // Certificate4.pem Modulus
    BN_hex2bn(&n, "ADE82473F41437F39B9E2B57281C87BEDCB7DF38908C6E3CE657A078F775C2A2FEF56A6EF6004F28DBDE68866C4493B6B163FD14126BBF1FD2EA319B217ED1333CBA48F5DD79DFB3B8FF12F1219A4BC18A8671694A66666C8F7E3C70BFAD292206F3E4C0E680AEE24B8FB7997E94039FD347977C99482353E838AE4F0A6F832ED149578C8074B6DA2FD0388D7B0370211B75F2303CFA8FAEDDDA63ABEB164FC28E114B7ECF0BE8FFB5772EF4B2784AE04C12250C708D0329A0E15324EC13D9EE19BF10B34A8C3F89A36151DEAC870794F46371EC2EE26F5B9881E1895C34796C76EF3B906279E6DBA49A2F26C5D010E10EDED9108E16FBB7F7A8F7C7E50207988F360895E7E237960D36759EFB0E72B11D9BBC03F94905D881DD05B42AD641E9AC0176950A0FD8DFD5BD121F352F28176CD298C1A80964776E4737BACEAC595E689D7F72D689C50641293E593EDD26F524C911A75AA34C401F46A199B5A73A516E863B9E7D72A712057859ED3E5178150B038F8DD02F05B23E7B4A1C4B730512FCC6EAE050137C439374B3CA74E78E1F0108D030D45B7136B407BAC130305C48B7823B98A67D608AA2A32982CCBABD83041BA2830341A1D605F11BC2B6F0A87C863B46A8482A88DC769A76BF1F6AA53D198FEB38F364DEC82B0D0A28FFF7DBE21542D422D0275DE179FE18E77088AD4EE6D98B3AC6DD27516EFFBC64F533434F");
    // Certificate4.pem Exponent
    BN_hex2bn(&e, "10001");
    // Signature from Certificate3.pem
    BN_hex2bn(&s, "1b7f252b907a087607718e1c32e8a364c417ebf174be330d75b0c7e9c96986f7bb068c02444cce2f2fcd1eadbd29f01f9174d0c9d55fda5ad6dd22f3f4b72c02eae73c7251657c23e15ade031d10a84846c6278423122461aed7a40bf9716814477ca6c7b5d215c07f2119121bfe12fc2ef6efd0520e4b4f779f32dbb372af0c6b1acac51f51fb35a1e66ce580718387f71a93c83bad7bc829e9a760f9eb029fdcbf38907481bfeab932e14210d5faf8eb754ab5d0ed45b4c71d092ea3da3369b7c1fe03b55b9d85353cc8366bb4adc81060018 8bf4b3d748b11341b9c4b69ecf2c778e42200b807e9fc5ab48dbbc6f048d6c4629020d708a1df11273b64624429e2a1718e3acc798c272cc6d2d766ddd2c2b2696a5cf21081be5da2fcbef9f7393aef8365f478f9728ceabe29826988bfdee28322229ed4c9509c420fa07e1862c44f68147c0e46232ed1dd83c488896c35e91b6af7b59a4eee3869cc78858ca282a66559b8580b91dd8402bc91c133ca9ebde99c21640f6f5a4ae2a256c52bac7044cb432bbfc385ca00c617b57ec774e50cfaf06a20f378ce10ed2d32f1abd9c713ecce1f8d1a8a3bd04f619c0f986aff50e1aaa956befca47714b631c4d96db55230a9d0f8175a0e640f56446036ecefa6a7d06eca4340674da53d8b9b8c6237da9f82a2da482a62e2d11cae6cd31587985e6721ca79fd34cd066d0a7bb");

    BN_mod_exp(message,s,e,n,ctx);

    printBN("Message = ", message);
    return 0;
}
```