Lab 10

Task 1: get private key

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

```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char*msg, BIGNUM*a)
{
/*Use BN_bn2hex(a) for hex string
Use BN_bn2dec(a) for decimal string*/

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* d = BN_new();
    BIGNUM* res1 = BN_new();
    BIGNUM* res2 = BN_new();
    BIGNUM* res = BN_new();
    BIGNUM* constant = BN_new();
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_hex2bn(&constant, "1");

    //ed mod (p-1)(q-1) = 1
    BN_sub(res1, p, constant);
    BN_sub(res2, q, constant);
    BN_mul(res,res1,res2,ctx);
    BN_mod_inverse(d, e, res, ctx);
    printBN("private key: ",d);

    /*
    private key:
3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
    */

}
```

```
[11/22/21]seed@VM:~/Desktop$ gcc bn.c -lcrypto
[11/22/21]seed@VM:~/Desktop$ ./a.out
secret key =  3587A24598E5F2A21DB007D89D18CC50ABA5075BA
19A33890FE7C28A9B496AEB
```

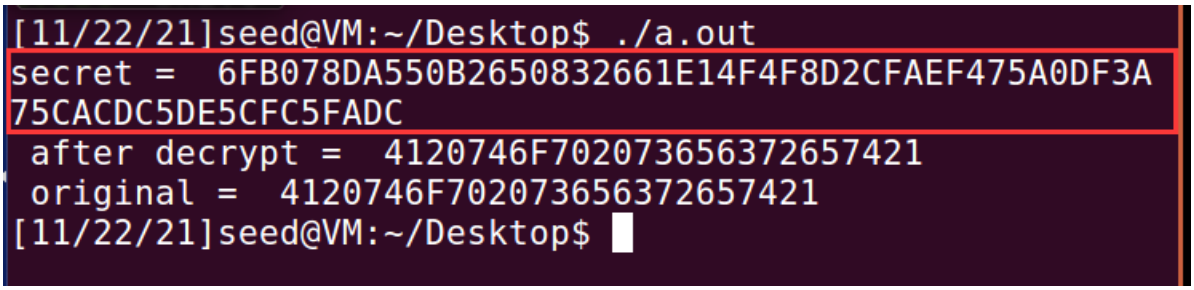
Task 2: Encrypting a Message

use the public key to encrypt the message.

secret= m^e mod n

```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char*msg, BIGNUM*a)
{
/*Use BN_bn2hex(a) for hex string
Use BN_bn2dec(a) for decimal string*/

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

int main()
{
/*
n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5
e = 010001 (this hex value equals to decimal 65537)
M = A top secret!
d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D
*/

/*4120746f702073656372657421*/
    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* M = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* d = BN_new();
    BIGNUM* res = BN_new();
    BN_hex2bn(&M, "4120746f702073656372657421");
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //s= m^e mod n
    BN_mod_exp(res, M, e, n,ctx);
    printBN("secret: ",res);

/*
secret:  6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
*/
}
```

The result is as following.



Task 3: Decrypting a Message

The message is '**Password is dees**'

```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char*msg, BIGNUM*a)
{
/*Use BN_bn2hex(a) for hex string
Use BN_bn2dec(a) for decimal string*/

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

int main()
{
/*
n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5
e = 010001 (this hex value equals to decimal 65537)
M = A top secret!
d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D
*/

    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* M = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* d = BN_new();
    BIGNUM* c = BN_new();
    BIGNUM* res = BN_new();
    BN_hex2bn(&c,
"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //s= c^d mod n
    BN_mod_exp(res, c, d, n,ctx);
    printBN("message: ",res);
/*
50617373776F726420697320646565573
Password is dees
```

```
*
}
```

```
[11/22/21]seed@VM:~/Desktop$ ./a.out
 after decrypt =  50617373776F726420697320646565 73
[11/22/21]seed@VM:~/Desktop$ python -c 'print("50617373
776F726420697320646565 73".decode("hex"))'
Password is dees
```

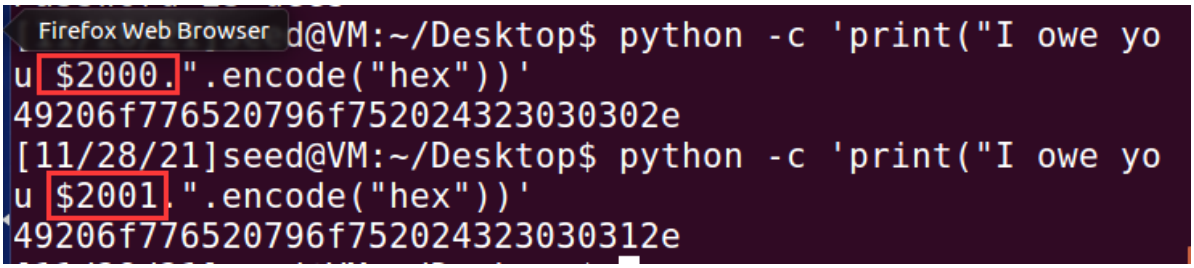
Task 4: Signing a Message

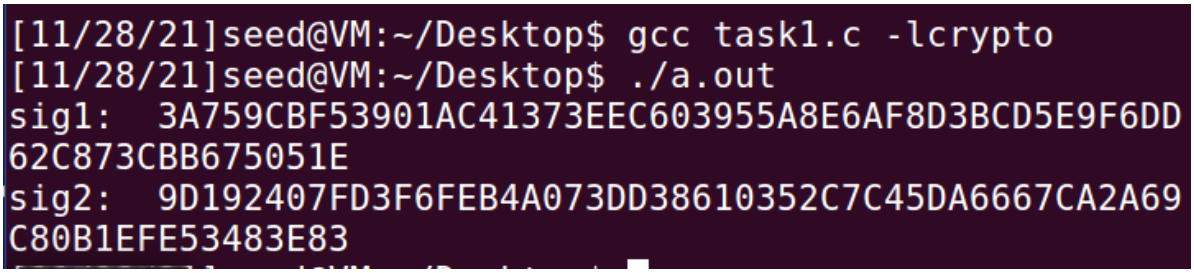
use private key to sign the message.

change a little bit about the message will make the signature totally different.

```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char*msg, BIGNUM*a)
{
/*Use BN_bn2hex(a) for hex string
Use BN_bn2dec(a) for decimal string*/

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

int main()
{
/*
n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5
e = 010001 (this hex value equals to decimal 65537)
M1 =49206f776520796f752024323030302e
M2 =49206f776520796f752024323030312e
d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D
*/

    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* m1 = BN_new();
    BIGNUM* m2 = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* d = BN_new();
    BIGNUM* res1 = BN_new();
    BIGNUM* res2 = BN_new();
    BN_hex2bn(&m1, "49206f776520796f752024323030302e");
    BN_hex2bn(&m2, "49206f776520796f752024323030312e");
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //s= m^e mod n
    BN_mod_exp(res1, m1, e, n,ctx);
```

```
    printBN("sig1: ",res1);
    BN_mod_exp(res2, m2, e, n,ctx);
    printBN("sig2: ",res2);
}
```





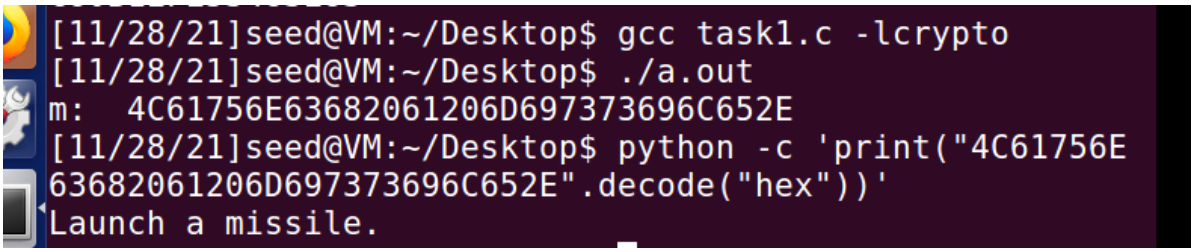
Task 5: Verifying a Signature

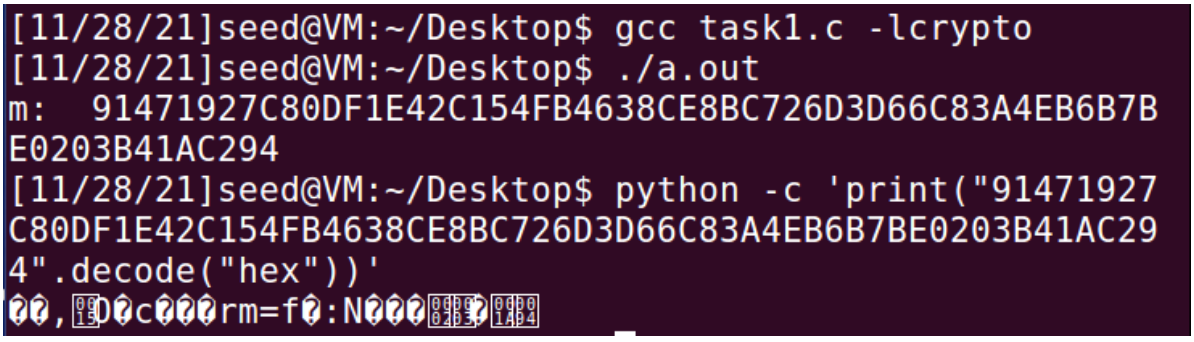
use the public key to verify it.

change a little bit about the signature will generate a wrong message.

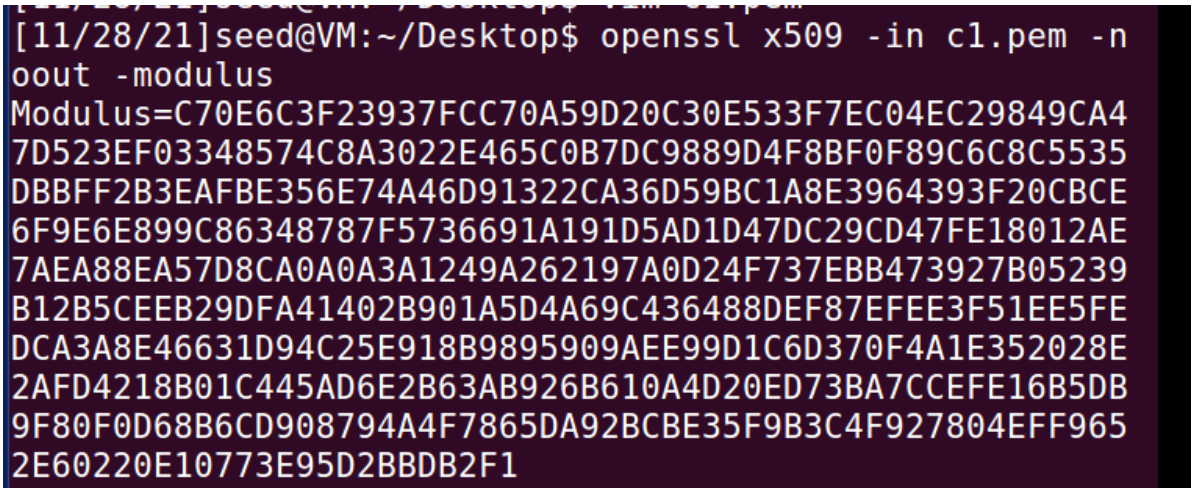
```
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 256
void printBN(char*msg, BIGNUM*a)
{
/*Use BN_bn2hex(a) for hex string
Use BN_bn2dec(a) for decimal string*/

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

int main()
{
/*
S = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
e = 010001 (this hex value equals to decimal 65537)
n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115
*/

    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* e = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* s = BN_new();
    BIGNUM* res = BN_new();
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n,
"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
```

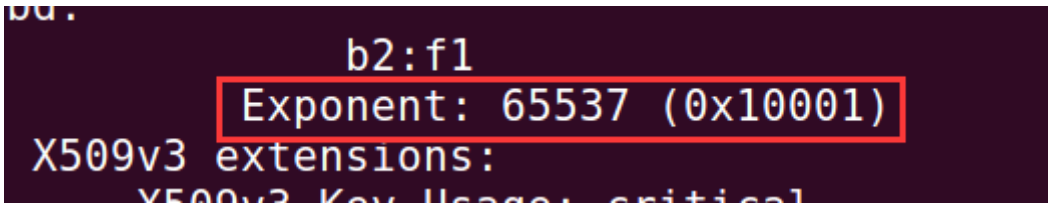
```
    BN_hex2bn(&s,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");

    //s= s^e mod n
    BN_mod_exp(res, s, e, n,ctx);
    printBN("m: ",res);

/*m:  4C61756E63682061206D697373696C652E
*/
}
```

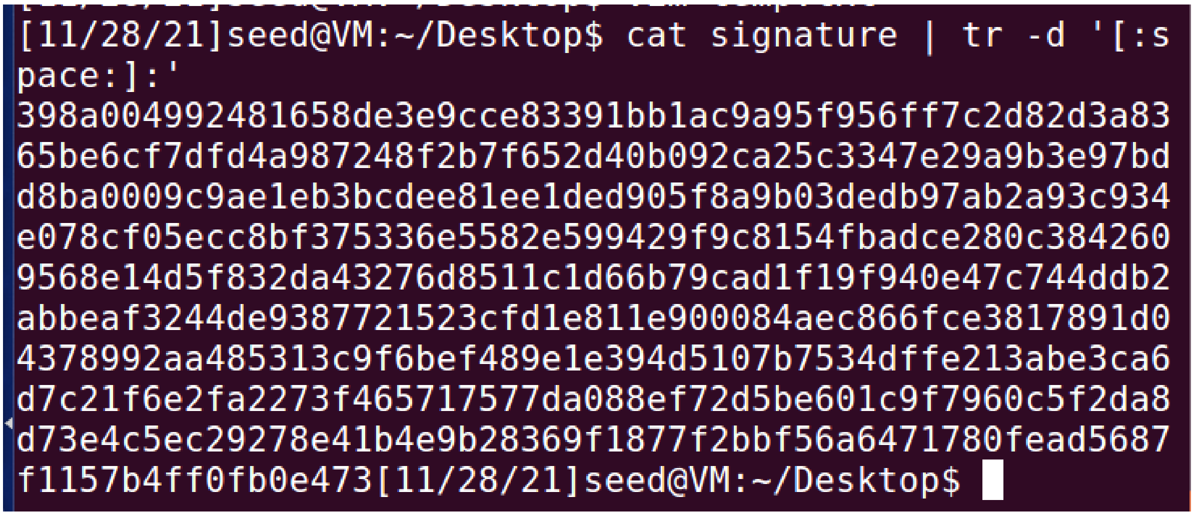




Task 6:



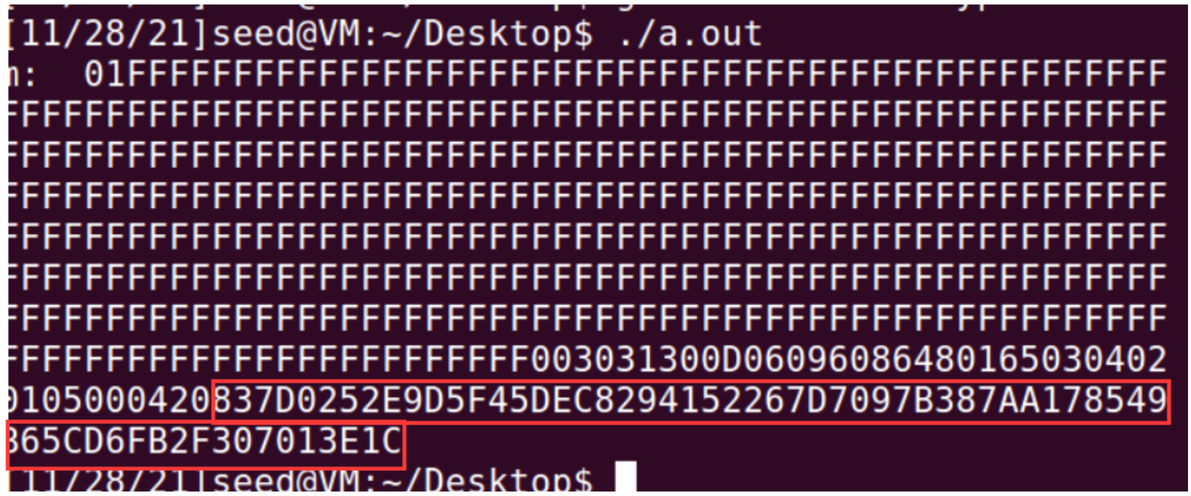




```
[11/28/21]seed@VM:~/Desktop$ cat signature | tr -d '[:s
pace:]:'
398a004992481658de3e9cce83391bb1ac9a95f956ff7c2d82d3a83
65be6cf7dfd4a987248f2b7f652d40b092ca25c3347e29a9b3e97bd
d8ba0009c9ae1eb3bcdee81ee1ded905f8a9b03dedb97ab2a93c934
e078cf05ecc8bf375336e5582e599429f9c8154fbadce280c384260
9568e14d5f832da43276d8511c1d66b79cad1f19f940e47c744ddb2
abbeaf3244de9387721523cfd1e811e900084aec866fce3817891d0
4378992aa485313c9f6bef489e1e394d5107b7534dffe213abe3ca6
d7c21f6e2fa2273f465717577da088ef72d5be601c9f7960c5f2da8
d73e4c5ec29278e41b4e9b28369f1877f2bbf56a6471780fead5687
f1157b4ff0fb0e473[11/28/21]seed@VM:~/Desktop$
```

```
[11/28/21]seed@VM:~/Desktop$ sha256sum c0_body.bin
837d0252e9d5f45dec8294152267d7097b387aa178549b65cd6fb2f
307013e1c  c0_body.bin
```

use m=s^e mod n to get the signature.



```
[11/28/21]seed@VM:~/Desktop$ ./a.out
n:  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFF003031300D060960864801650304 02
0105000420837D0252E9D5F45DEC8294152267D7097B387AA178549
B65CD6FB2F307013E1C
[11/28/21]seed@VM:~/Desktop$
```

We can see that the last part of it is the same as the hashed certificate.