Lab6

Yue Zhang

Task1

```c
#include <stdio.h>
#include <openssl/bn.h>
#define NBITS 128

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

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 *res = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *p_minus_one = BN_new();
    BIGNUM *q_minus_one = BN_new();
    BIGNUM *phi = BN_new();

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

    BN_mul(n, p, q, ctx);
    printBN("public key: ", e, n);
```

```
    BN_sub(p_minus_one, p, BN_value_one());
    BN_sub(q_minus_one, q, BN_value_one());
    BN_mul(phi, p_minus_one, q_minus_one, ctx);
    BN_mod_inverse(d, e, phi, ctx);
    printBN("private key: ", d, n);
    BN_clear_free(p);
    BN_clear_free(q);
    BN_clear_free(n);
    BN_clear_free(res);
    BN_clear_free(phi);
    BN_clear_free(e);
    BN_clear_free(d);
    BN_clear_free(p_minus_one);
    BN_clear_free(q_minus_one);

    return 0;

}
```

This code is used for calculate private key(d).

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task1 task1.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task1
public key:  (0D88C3,E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1)
private key:  (3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB,E103ABD9489
2E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1)
```

Here we can see private key is:

3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB

Task 2

```c
#include <openssl/bn.h>

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

int main()
{
    // init
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    // BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();

    // assign values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB8162924
2FB1A5");
    BN_dec2bn(&e, "65537");
    BN_hex2bn(&M, "4120746f702073656372657421"); //hex encode for " A top
 secret!"

    // encrypt M: M^e mod n
    BN_mod_exp(C, M, e, n, ctx);
    printBN("Encryption result:", C);
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task2 task2.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task2
Encryption result: 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
```

The result is: 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC

Task3

```c
//task3.c
#include <stdio.h>
#include <openssl/bn.h>

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

int main()
{
    // init
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    // BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();

    // assign values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    BN_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");

    // decrypt C: C^d mod n
    BN_mod_exp(M, C, d, n, ctx);
    printBN("Decryption result:", M);
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task3 task3.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task3
Decryption result: 50617373776F726420697320646565573
```

I get the decryption result.

```
print(binascii.a2b_hex("50617373776F726420697320646565573"))
```

```
b'Password is dees'
```

Then I convert it to ASCII string, we can see the message is “Password is dees”

Task4

```
m1 = bytes("I owe you $2000", 'utf-8')
m2 = bytes("I owe you $3000", 'utf-8')
print(binascii.b2a_hex(m1))
print(binascii.b2a_hex(m2))
```

```
b'49206f776520796f75202432303030'
b'49206f776520796f75202433303030'
```

I convert them to hex string, we can find they are very similar.

```
//task4.c
#include <stdio.h>
#include <openssl/bn.h>

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

int main()
{
    // init
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *M1 = BN_new();
    BIGNUM *M2 = BN_new();
    BIGNUM *C1 = BN_new();
    BIGNUM *C2 = BN_new();

    // assign values
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB8162924
2FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381C
D7D30D");
    BN_hex2bn(&M1, "49206f776520796f75202432303030"); // hex encode for "I
 owe you $2000"
    BN_hex2bn(&M2, "49206f776520796f75202433303030"); // hex encode for "I
 owe you $3000"
```

```c
    // encrypt M: M^d mod n
    BN_mod_exp(C1, M1, d, n, ctx);
    BN_mod_exp(C2, M2, d, n, ctx);
    printBN("Signature of M1:", C1);
    printBN("Signature of M2:", C2);

    // clear sensitive data
    BN_clear_free(n);
    BN_clear_free(d);
    BN_clear_free(M1);
    BN_clear_free(M2);
    BN_clear_free(C1);
    BN_clear_free(C2);

    return 0;
}
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ vi task4.c
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task4 task4.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task4
Signature of M1: 80A55421D72345AC199836F60D51DC9594E2BDB4AE20C804823FB71660DE7B82
Signature of M2: 04FC9C53ED7BBE4ED4BE2C24B0BDF7184B96290B4ED4E3959F58E94B1ECEA2EB
```

Although two message are very similar, their signatures are very different.

Task5

```c
//task5.c
#include <stdio.h>
#include <openssl/bn.h>

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

int main()
{
    // init
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    // BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *S = BN_new();

    // assign values
    BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18
116115");
    BN_dec2bn(&e, "65537");
    BN_hex2bn(&M, "4c61756e63682061206d697373696c652e"); //hex encode for "
 Launch a missile."
    BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBD
B6802F");
    //BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542C
BDB6803F");
```

```c
    // Get S^e mod: if S=M^d mod n, C=M
    BN_mod_exp(C, S, e, n, ctx);

    // verify the signature
    if (BN_cmp(C, M) == 0)
    {
        printf("Valid Signature! \n");
    }
    else
    {
        printf("Verification fails! \n");
    }

    // clear sensitive data
    BN_clear_free(n);
    BN_clear_free(e);
    BN_clear_free(M);
    BN_clear_free(C);
    BN_clear_free(S);

    return 0;
}
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ vi task5.c
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task5 task5.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task5
Valid Signature!
```

The signature is Alice's.

```
BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ vi task5.c
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task5 task5.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task5
Verification fails!
```

Then I change the last bytes "2F" to "3F"of signature. Repeat the task, we can find the verification failed.

Task6

Step1

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ openssl s_client -connect seedsecuritylabs.org:443
 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = seedsecuritylabs.org
verify return:1
---
Certificate chain
 0 s:CN = seedsecuritylabs.org
   i:C = US, O = Let's Encrypt, CN = R3
-----BEGIN CERTIFICATE-----
MIIFLjCCBBagAwIBAgISAw+FC14Mj71ABAv+HhZAEmTeMA0GCSqGSIb3DQEBCwUA
MDIxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQD
EwJSMzAeFw0yMTExMDMyMjQ3MDVaFw0yMjAyMDEyMjQ3MDRaMB8xHTAbBgNVBAMT
FHNlZWRzZWN1cml0eWxhYnMub3JnMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAq8cbDO3GAfjqqbPPCBdPost8NMRmEubv85gXecll7mZMH5qSfTPuB/ou
FWL3tPMf1U8usWeoSUK/48yatzBGwmj1KKlkaW9MS2QkydztRp+kH8LvbzbQvGkn
```

I use commend to see certificates. Then save the first certificate as c0.pem and save the second certificate as c1.pem.

Step2

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$  openssl x509 -in c1.pem -noout -modulus
Modulus=BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D26AAB52BB9C54CB1AF8E6BF975C8A3D
70F4794145535578C9EA8A23919F5823C42A94E6EF53BC32EDB8DC0B05CF35938E7EDCF69F05A0B1BBEC0942425
87FA3771B313E71CACE19BEFDBE43B45524596A9C153CE34C852EEB5AEED8FDE6070E2A554ABB66D0E97A540346
B2BD3BC66EB66347CFA6B8B8F572999F830175DBA726FFB81C5ADD286583D17C7E709BBF12BF786DCC1DA715DD4
46E3CCAD25C188BC60677566B3F118F7A25CE653FF3A88B647A5FF1318EA9809773F9D53F9CF01E5F5A6701714A
F63A4FF99B3939DDC53A706FE48851DA169AE2575BB13CC5203F5ED51A18BDB15
```

We get the modulus n.

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

```
        af:27
    Exponent: 65537 (0x10001)
v3 extensions:
```

65537 is e.

Step3

```
                                7D:0F:4E:B9:4B:78
    Signature Algorithm: sha256WithRSAEncryption
         10:4e:60:55:78:e7:57:4c:71:b6:12:3e:6c:a7:01:61:44:75:
         82:71:1b:84:fd:eb:c7:44:34:2c:33:54:28:fe:80:67:2b:f6:
         29:68:b4:ad:f8:14:bf:67:b2:bc:c2:6b:1e:01:54:c5:a0:ee:
         ea:b9:f6:cd:c2:0f:76:46:8c:51:98:db:fa:71:e4:59:b7:1c:
         13:4a:7f:56:c1:5c:b3:71:36:6c:19:60:5d:60:8e:42:31:f6:
         37:03:99:f0:17:5e:81:b0:a3:da:f8:9e:fc:fe:4f:ad:e4:e1:
         60:65:97:b5:17:18:1f:05:40:24:5f:3e:84:c5:b6:30:24:78:
         46:ea:2e:0a:ec:f9:55:e7:fb:73:4a:98:99:30:35:1c:4b:a3:
         da:78:50:c6:44:62:d6:a6:e9:64:cd:0f:46:40:03:79:de:b7:
         79:c9:40:bc:10:e1:a9:6a:3c:e9:bc:1c:43:4b:fc:76:9c:cb:
         1f:c1:38:43:8b:85:bf:51:a9:aa:e4:3d:07:e9:4a:2c:61:53:
         56:e0:5c:ee:43:52:40:6e:7f:3c:df:0a:b0:36:46:00:4f:d7:
         16:48:d2:f7:06:31:d2:20:f6:ab:86:b9:d4:a7:71:dc:e8:2e:
         f7:4f:36:a7:43:ae:99:e8:c2:05:99:6e:7c:3f:f1:11:4e:c9:
         dc:30:d1:34
```

This is signature block.

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$  cat signature | tr -d '[:space:]:'
104e605578e7574c71b6123e6ca70161447582711b84fdebc744342c335428fe80672bf62968b4adf814bf67b2b
cc26b1e0154c5a0eeeab9f6cdc20f76468c5198dbfa71e459b71c134a7f56c15cb371366c19605d608e4231f637
0399f0175e81b0a3daf89efcfe4fade4e1606597b517181f0540245f3e84c5b630247846ea2e0aecf955e7fb734
a989930351c4ba3da7850c64462d6a6e964cd0f46400379deb779c940bc10e1a96a3ce9bc1c434bfc769ccb1fc1
38438b85bf51a9aae43d07e94a2c615356e05cee4352406e7f3cdf0ab03646004fd71648d2f70631d220f6ab86b
9d4a771dce82ef74f36a743ae99e8c205996e7c3ff1114ec9dc30d134seed@ip-172-31-30-228:/home/ubuntu
/lab6$
```

Remove the “space” and “ : “ from data.

Step 4

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ openssl asn1parse -i -in c0.pem
    0:d=0  hl=4 l=1326 cons: SEQUENCE
    4:d=1  hl=4 l=1046 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
```

```
 1054:d=1  hl=2 l=  13 cons:  SEQUENCE
 1056:d=2  hl=2 l=   9 prim:   OBJECT            :sha256WithRSAEncryption
 1067:d=2  hl=2 l=   0 prim:   NULL
 1069:d=1  hl=4 l= 257 prim:  BIT STRING
```

The certificate body is from 4 to 1053.

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$  openssl asn1parse -i -in c0.pem -strparse 4 -out
c0_body.bin -noout
seed@ip-172-31-30-228:/home/ubuntu/lab6$  sha256sum c0_body.bin
976533f9a4d372cbbee6b5b85398f8484d3228e192eb59f6abc67873749fcde9  c0_body.bin
seed@ip-172-31-30-228:/home/ubuntu/lab6$
```

Then we use commend to get hash value.

976533f9a4d372cbbee6b5b85398f8484d3228e192eb59f6abc67873749fcde9

Step5

```
prefix = "0001"
hash = "976533f9a4d372cbbee6b5b85398f8484d3228e192eb59f6abc67873749fcde9"
A = "30 31 30 0D 06 09 60 86 48 01 65 03 04 02 01 05 00 04 20".replace(' ','')
total_len = 256
pad_len = total_len - 1 - (len(A) + len(prefix) + len(hash))//2
print(prefix + "FF" * pad_len + "00" + A + hash)
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ python3 message.py
0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D060960864801650304020105000420976533f
9a4d372cbbee6b5b85398f8484d3228e192eb59f6abc67873749fcde9
```

Getting signed certificate body.

```c
//task6.c
#include <stdio.h>
#include <openssl/bn.h>

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

int main()
{
    // init
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    // BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *S = BN_new();

    // assign values
    BN_hex2bn(&n, "BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D26AAB5
2BB9C54CB1AF8E6BF975C8A3D70F4794145535578C9EA8A23919F5823C42A94E6EF53BC32EDB8
DC0B05CF35938E7EDCF69F05A0B1BBEC094242587FA3771B313E71CACE19BEFDBE43B45524596
A9C153CE34C852EEB5AEED8FDE6070E2A554ABB66D0E97A540346B2BD3BC66EB66347CFA6B8B8
F572999F830175DBA726FFB81C5ADD286583D17C7E709BBF12BF786DCC1DA715DD446E3CCAD25
C188BC60677566B3F118F7A25CE653FF3A88B647A5FF1318EA9809773F9D53F9CF01E5F5A6701
714AF63A4FF99B3939DDC53A706FE48851DA169AE2575BB13CC5203F5ED51A18BDB15");
    BN_dec2bn(&e, "65537");
    BN_hex2bn(&M, "0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

```c
    BN_hex2bn(&M, "0001FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D0609608648016503040201050
00420976533f9a4d372cbbee6b5b85398f8484d3228e192eb59f6abc67873749fcde9"); /
/hash of the certificate body
    BN_hex2bn(&S, "104e605578e7574c71b6123e6ca70161447582711b84fdebc744342c33
5428fe80672bf62968b4adf814bf67b2bcc26b1e0154c5a0eeeab9f6cdc20f76468c5198dbfa7
1e459b71c134a7f56c15cb371366c19605d608e4231f6370399f0175e81b0a3daf89efcfe4fad
e4e1606597b517181f0540245f3e84c5b630247846ea2e0aecf955e7fb734a989930351c4ba3d
a7850c64462d6a6e964cd0f46400379deb779c940bc10e1a96a3ce9bc1c434bfc769ccb1fc138
438b85bf51a9aae43d07e94a2c615356e05cee4352406e7f3cdf0ab03646004fd71648d2f7063
1d220f6ab86b9d4a771dce82ef74f36a743ae99e8c205996e7c3ff1114ec9dc30d134");

    // decrypt S: S^e mod n = (M^d mod)^e mod n = M
    BN_mod_exp(C, S, e, n, ctx);

    if (BN_cmp(C, M) == 0)
    {
        printf("Valid Signature! \n");
    }
    else
    {
        printf("Verification fails! \n");
    }
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ vi task6.c
seed@ip-172-31-30-228:/home/ubuntu/lab6$ gcc -o task6 task6.c -lcrypto
seed@ip-172-31-30-228:/home/ubuntu/lab6$ ./task6
Valid Signature!
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab6$ openssl verify -untrusted c0.pem c1.pem
c1.pem: OK
```

The signature is valid.