CS 766 - RSA Cryptosystem Lab

Task 1: Deriving the Private Key –

Program code:

```c
#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(); //To store BIGNUM temporary variable

   BIGNUM *p=BN_new(); //Initialize p BIGNUM Varibale
BIGNUM *q=BN_new(); //Initialize q BIGNUM Varibale
BIGNUM *e=BN_new(); //Initialize e BIGNUM Varibale
BIGNUM *res1=BN_new(); //Initialize res1  BIGNUM Varibale
BIGNUM *res2=BN_new(); //Initialize res2 BIGNUM Varibale
BIGNUM *n=BN_new(); //Initialize n  BIGNUM Varibale
BIGNUM *d=BN_new(); //Initialize d BIGNUM Varibale
BIGNUM *one=BN_new(); //Initialize one BIGNUM Varibale


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

BN_sub(res1,p,one); //res1=p-1
BN_sub(res2, q,one); //res2=q-1
BN_mul(n,res1,res2,ctx); //n=res1*res2
BN_mod_inverse(d,e,n,ctx); //e.d modn=1
printBN("The private key is d= ",d);
return 0;
}
```

Output:

```
[03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  privatekey
[03/26/21]seed@SEED:~$ ./privatekey
The private key is d=  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[03/26/21]seed@SEED:~$
```

Task 2: Encrypting a Message –

Program code:

```
/bin/bash 118x42
#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(); //To store BIGNUM temporary variable

   BIGNUM *c = BN_new(); //Initialize c BIGNUM Varibale
   BIGNUM *p = BN_new(); //Initialize p BIGNUM Varibale
   BIGNUM *e = BN_new(); //Initialize e BIGNUM Varibale
   BIGNUM *M = BN_new(); //Initialize M  BIGNUM Varibale
   //BIGNUM *res2 = BN_new(); //Initialize res2 BIGNUM Varibale
   BIGNUM *n = BN_new(); //Initialize n  BIGNUM Varibale
   BIGNUM *d = BN_new(); //Initialize d BIGNUM Varibale
   //BIGNUM *one = BN_new(); //Initialize one BIGNUM Varibale


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

  BN_mod_exp(c,M,e,n,ctx); // c=M^e modn
  printBN("The hex ciphertext is =",c);
 BN_mod_exp(p,c,d,n,ctx); //p=c^d mod n
 printBN("The hex plaintext is =", p);
  return 0;
}
```

Output:

```
/bin/bash 118x42
[03/26/21]seed@SEED:~$ python -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421
[03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  encanddec
[03/26/21]seed@SEED:~$ ./encanddec
The hex ciphertext is = 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
The hex plaintext is = 4120746F702073656372657421
[03/26/21]seed@SEED:~$
```

Task 3: Decrypting a Message -

Program code:

```
#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(); //To store BIGNUM temporary variable

   BIGNUM *c = BN_new(); //Initialize c BIGNUM Varibale
   BIGNUM *p = BN_new(); //Initialize p BIGNUM Varibale
   BIGNUM *e = BN_new(); //Initialize e BIGNUM Varibale
   BIGNUM *M = BN_new(); //Initialize M  BIGNUM Varibale
   //BIGNUM *res2 = BN_new(); //Initialize res2 BIGNUM Varibale
   BIGNUM *n = BN_new(); //Initialize n  BIGNUM Varibale
   BIGNUM *d = BN_new(); //Initialize d BIGNUM Varibale
   //BIGNUM *one = BN_new(); //Initialize one BIGNUM Varibale


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

  //BN_mod_exp(c,M,e,n,ctx); // c=M^e modn
  //printBN("The hex ciphertext is =",c);
 BN_mod_exp(p,c,d,n,ctx); //p=c^d mod n
 printBN("The plaintext of ciphertext is ", p);
  return 0;
}
```

Output:

```
[03/26/21]seed@SEED:~$ vim rsa-lab_benakahallisiddeshappa.c
[03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  plaintext
[03/26/21]seed@SEED:~$ ./plaintext
The plaintext of ciphertext is  = 50617373776F726420697320064656573
[03/26/21]seed@SEED:~$ python -c 'print("50617373776F726420697320064656573".decode("hex"))'
Password is dees
[03/26/21]seed@SEED:~$
```

Task 4: Signing a Message-

Program code:

```
/*task 4 SIgning a message*/
  BN_hex2bn(&m,"49206f776520796f752024323030302e"); // Initialize m
   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5"); //Initialize n
  BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");//Initialize d

 BN_mod_exp(s,m,d,n,ctx); //s=m^d mod n
printf("\n TASK 4 OUTPUT \n");
 printBN(" The signature for the message in hex is", s);
```

Output:

```
/bin/bash 111x32
[03/26/21]seed@SEED:~$ python -c 'print("I owe you $2000.".encode("hex"))'
49206f776520796f752024323030302e
[03/26/21]seed@SEED:~$ vim rsa-lab_benakahallisiddeshappa.c
[03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  RSA
[03/26/21]seed@SEED:~$ ./RSA

 TASK 4 OUTPUT
 The signature for the message in hex is 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB

[03/26/21]seed@SEED:~$
```

Task 5: Verifying a Signature-

Program code:

```
/*task 5 Verifying a signature*/
    BN_hex2bn(&m,"4c61756e63682061206d697373696c652e"); // Initialize m
   BN_hex2bn(&n,"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115"); //Initialize n
  BN_hex2bn(&e,"010001"); //Initialize e
  BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");//Initialize s

  BN_mod_exp(p,s,e,n,ctx); // p=s^e modn
  printf("\n TASK 5 OUTPUT \n");
  printBN("\n The provided message is ", m);
  printBN("\n The provided signature is ", s);
  printBN("\n The computed msg is", p);

 if(BN_cmp(m, p) == 0)
{

  printf("The Signature Matches!");
}
else
{
printf("Verification failed");
}


 BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F"); //Corrupted s from 2F to 3F

BN_mod_exp(c,s,e,n,ctx); // p=s^e modn
 printBN("\n The provided message is ", m);
  printBN("\n The provided signature is ", s);
  printBN("\n The computed msg is",c);
   if(BN_cmp(c, m) == 0)
{

  printf("The Signature Matches!");
}
else
{
printf("Verification failed");
}


 printf("\n");
  return 0;
}
```

Output:

```
/bin/bash 132x48
[03/26/21]seed@SEED:~$ python -c 'print("Launch a missile.".encode("hex"))'
4c61756e63682061206d697373696c652e
[03/26/21]seed@SEED:~$ vim rsa-lab_benakahallisiddeshappa.c
[03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  RSA
[03/26/21]seed@SEED:~$ ./RSA

 TASK 5 OUTPUT

 The provided message is  4C61756E63682061206D697373696C652E

 The provided signature is  643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F

 The computed msg is 4C61756E63682061206D697373696C652E
The Signature Matches!
 The provided message is  4C61756E63682061206D697373696C652E

 The provided signature is  643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F

 The computed msg is 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
Verification failed
[03/26/21]seed@SEED:~$
```

Tasks 1-5:

Provided file with the code from lab tasks 1-5 in a single file called rsa-lab_benakahallisiddeshappa.c.

Program code and Output:

```c
#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(); //To store BIGNUM temporary variable

BIGNUM *q=BN_new(); //Initialize q BIGNUM Varibale

BIGNUM *res1=BN_new(); //Initialize res1  BIGNUM Varibale
BIGNUM *res2=BN_new(); //Initialize res2 BIGNUM Varibale

   BIGNUM *s = BN_new(); //Initialize s BIGNUM Varibale
   BIGNUM *p = BN_new(); //Initialize p BIGNUM Varibale
   BIGNUM *e = BN_new(); //Initialize e BIGNUM Varibale
   BIGNUM *m = BN_new(); //Initialize m  BIGNUM Varibale
   BIGNUM *c = BN_new(); //Initialize res2 BIGNUM Varibale
   BIGNUM *n = BN_new(); //Initialize n  BIGNUM Varibale
   BIGNUM *d = BN_new(); //Initialize d BIGNUM Varibale
   BIGNUM *one = BN_new(); //Initialize one BIGNUM Varibale
   BIGNUM *M = BN_new();


/* Task1 code to find the private key d */

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

BN_sub(res1,p,one); //res1=p-1
BN_sub(res2, q,one); //res2=q-1
BN_mul(n,res1,res2,ctx); //n=res1*res2
BN_mod_inverse(d,e,n,ctx); //e.d modn=1
printf("\n TASK1 OUTPUT \n");
printBN(" The private key is d= ",d);

/* Task2 code encrypting a Message */
```

```c
/* Task2 code encrypting a Message */

   BN_hex2bn(&M,"4120746f702073656372657421"); //Intialize M
   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5"); //Initialize n
  BN_hex2bn(&e,"010001"); //Initialize e
  BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");//Initialize d

  BN_mod_exp(c,M,e,n,ctx); // c=M^e modn
 printf("\n TASK 2 OUTPUT \n");
  printBN(" The hex ciphertext is = ",c);
  BN_mod_exp(p,c,d,n,ctx); //p=c^d mod n
  printBN(" The hex plaintext is ", p);

/*Task 3 decrypting a Message */
    BN_hex2bn(&c,"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F"); // Initialize c
   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5"); //Initialize n
   BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");//Initialize d

 BN_mod_exp(p,c,d,n,ctx); //p=c^d mod n
 printf("\n TASK 3 OUTPUT \n");
 printBN(" The plaintext of ciphertext is", p);

/*task 4 SIgning a message*/
  BN_hex2bn(&m,"49206f776520796f752024323030302e"); // Initialize m
   BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5"); //Initialize n
  BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");//Initialize d

 BN_mod_exp(s,m,d,n,ctx); //s=m^d mod n
printf("\n TASK 4 OUTPUT \n");
 printBN(" The signature for the message in hex is", s);

/*task 5 Verifying a signature*/
```

```c
/*task 5 Verifying a signature*/
    BN_hex2bn(&m,"4c61756e63682061206d697373696c652e"); // Initialize m
   BN_hex2bn(&n,"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115"); //Initialize n
  BN_hex2bn(&e,"010001"); //Initialize e
  BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");//Initialize s

  BN_mod_exp(p,s,e,n,ctx); // p=s^e modn
  printf("\n TASK 5 OUTPUT \n");
  printBN("\n The provided message is ", m);
  printBN("\n The provided signature is ", s);
  printBN("\n The computed msg is", p);

 if(BN_cmp(m, p) == 0) //Comparing m and p
{

  printf("The Signature Matches!\n"); //If m and p are equal
}
else
{
printf("Verification failed");     // If m and p are not equal
}

printf("=========================");
 BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F"); //Corrupted s from 2F to 3F

BN_mod_exp(c,s,e,n,ctx); // c=s^e modn
 printBN("\n The provided message is ", m);
  printBN("\n The provided signature is ", s);
  printBN("\n The computed msg is",c);
   if(BN_cmp(c, m) == 0) //Comparing m and c
{

  printf("The Signature Matches!"); //If c and m are equal
}
else
{
printf("Verification failed"); //If c and m are not equal
}


 printf("\n");
  return 0;
}
```

```
03/26/21]seed@SEED:~$ vim rsa-lab_benakahallisiddeshappa.c
03/26/21]seed@SEED:~$ gcc rsa-lab_benakahallisiddeshappa.c -lcrypto -o  RSA
03/26/21]seed@SEED:~$ ./RSA

TASK1 OUTPUT
The private key is d=  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB

TASK 2 OUTPUT
The hex ciphertext is =  6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
The hex plaintext is  4120746F702073656372657421

TASK 3 OUTPUT
The plaintext of ciphertext is 50617373776F726420697320646565573

TASK 4 OUTPUT
The signature for the message in hex is 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB

TASK 5 OUTPUT

The provided message is  4C61756E63682061206D697373696C652E

The provided signature is  643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F

The computed msg is 4C61756E63682061206D697373696C652E
he Signature Matches!
=========================
The provided message is  4C61756E63682061206D697373696C652E

The provided signature is  643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F

The computed msg is 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
erification failed
03/26/21]seed@SEED:~$ python -c 'print("Task 3","50617373776F726420697320646565573".decode("hex"))'
'Task 3', 'Password is dees')
03/26/21]seed@SEED:~$
```