RSA Encryption and Signature Lab

Task 2: Encrypting a Message

```
[04/15/22]seed@VM:~/.../Task2-5$ python -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421
[04/15/22]seed@VM:~/.../Task2-5$ gcc CS234_Security_Lab03-Cryptography-RSA_S40_task-2-5.c -
lcrypto
[04/15/22]seed@VM:~/.../Task2-5$ ./a.out
Encrypting a Message 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
```

```
// Task 2 Encrypting a message

// assign values
BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
BN_hex2bn(&e, "010001"); //hex value equals to decimal 65537
BN_hex2bn(&M, "4120746f702073656372657421"); //hex encode for " A top secret!"

// Solution: C = M^e mod n
BN_mod_exp(C, M, e, n, ctx);
printBN("Encrypting a Message", C);
```

โดยใน Task นี้จะเป็นการ Encrypting Message โดยการที่เราต้องการจะให้ข้อความที่เราจะส่งไปเป็นความลับ โดยเราจะนำข้อความไปทำการ encode ให้เป็น hex(เลขฐาน16) เพื่อจะนำไป Encrypting Message ใช้ฟังก์ชั่น BN_mod_exp โดยใช้สูตร $C = M^e \bmod n$ โดยจะเก็บที่ตัวแปร C โดย Encrypt ที่ได้ จะเป็น hex(เลขฐาน16)

Task 3: Decrypting a Message

```
[04/15/22]seed@VM:~/.../Task2-5$ python -c 'print("4120746f702073656372657421".decode("hex"
))'
A top secret!
[04/15/22]seed@VM:~/.../Task2-5$ gcc CS234_Security_Lab03-Cryptography-RSA_S40_task-2-5.c -
lcrypto
[04/15/22]seed@VM:~/.../Task2-5$ ./a.out
Decrypting a Message 50617373776F726420697320064656573
```

```
[04/15/22]seed@VM:~/.../Task2-5$ python -c 'print("50617373776F726420697320064656573".decode
("hex"))'
Password is dees
```

```
// Task 3 Decrypting a Message

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

// Solution: M = C^d mod n
BN_mod_exp(M, C, d, n ,ctx);
printBN("Decrypting a Message", M);
```

โดยใน Task นี้จะเป็นการ Decrypting Message โดยเราต้องการถอดรหัสจากข้อความที่ถูกเข้ารหัส โดยเราจะทำการ decrypting Message จากรหัสมาเป็นข้อความ โดยใช้ฟังก์ชั่น BN_mod_exp โดยใช้สูตร $M = C^d \bmod n$ โดยจะเก็บที่ตัวแปร M หลังจากที่ Decrypt แล้วจะนำ hex(เลขฐาน16) ที่ได้ไป decode จะได้เป็น ASCII String

Task 4: Signing a Message

```
[04/24/22]seed@VM:~/.../Task2-5$ python -c 'print("I owe you $2000".encode("hex"))'
49206f776520796f75202432303030
[04/24/22]seed@VM:~/.../Task2-5$ python -c 'print("I owe you $3000".encode("hex"))'
49206f776520796f75202433303030

[04/24/22]seed@VM:~/.../Task2-5$ gcc CS234_Security_Lab03-Cryptography-RSA_S40_task-2-5.c -lcrypto
[04/24/22]seed@VM:~/.../Task2-5$ ./a.out
Signature of M1 80A55421D72345AC199836F60D51DC9594E2BDB4AE20C804823FB71660DE7B82
Signature of M2 04FC9C53ED7BBE4ED4BE2C24B0BDF7184B96290B4ED4E3959F58E94B1ECEA2EB
```

```
// Task 4 Signing a Message

//new variable
BIGNUM *M1 = BN_new();
BIGNUM *M2 = BN_new();
BIGNUM *S1 = BN_new();
BIGNUM *S2 = BN_new();

// assign values
BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
BN_hex2bn(&M1, "49206f776520796f75202432303030"); // hex encode for "I owe you $2000"
BN_hex2bn(&M2, "49206f776520796f75202433303030"); // hex encode for "I owe you $3000"

// Solution: C = M^d mod n
BN_mod_exp(S1, M1, d, n, ctx);
BN_mod_exp(S2, M2, d, n, ctx);
printBN("Signature of M1", S1);
printBN("Signature of M2", S2);
```

โดยใน Task นี้จะเป็นการ Signing a Message โดยการไป encode ข้อความ 2 ข้อความที่โจทย์ให้มา เพื่อนำข้อความสองข้อความไปหาข้อแตกต่าง โดยใช้คำสั่ง BN_mod_exp โดยใช้สูตร $C = M^d$ **mod n** โดยข้อสังเกตุของข้อนี้คือ ถ้า 2 ข้อความ ที่นำไป **encode** มีข้อความที่ต่างกันไม่มากเกินไป ก็จะได้เลขฐาน 16 ที่ไม่ต่างกันมาก ถึงแม้ว่าเลขฐาน 16 ได้จะไม่ต่างกันมากแต่ถ้านำไป **Encrypt** แล้วจะแตกต่างกันมาก

Task 5: Verifying a Signature

```
[04/24/22]seed@VM:~/.../Task2-5$ python -c 'print("Launch a missile.".encode("hex"))'
4c61756e63682061206d697373696c652e
[04/24/22]seed@VM:~/.../Task2-5$ gcc CS234_Security_Lab03-Cryptography-RSA_S40_task-2-5.c -lcrypto
[04/24/22]seed@VM:~/.../Task2-5$ ./a.out
Valid Signature
```

```
// Task 5 Verifying a Signature

//new variable
BIGNUM *S = BN_new();

// assign values
BN_hex2bn(&M, "4c61756e63682061206d697373696c652e"); //hex encode for "Launch a missile."
BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
//BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F"); //change 2F to 3F for fails case
BN_hex2bn(&e, "010001");
BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

// Solution: C = S^e mod n for get Message to check C=M
BN_mod_exp(C, S, e, n ,ctx);

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

โดยใน Task นี้จะเป็นการ Verifying a Signature โดยจะนำข้อความที่ encode ไปใช้เป็นข้อความเพื่อใช้ เช็คว่าข้อความนี้เป็น Signature ที่ตรงหรือกับผู้ใช้นั้นหรือไม่ โดยใช้สั่ง BN_mod_exp โดยใช้สูตร $C = S^e \bmod n$ เพื่อ Encrypt ให้อยู่ในตัวแปร C เพื่อจะนำไปเช็คเงื่อนไขว่าเลขฐาน 16 ที่ได้มาตรงกับข้อความ ที่ทำการ encode หรือไม่ ถ้าถูกก็จะพิมพ์ Valid Signature มา

```
BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
//BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F"); //change 2F to 3F for fails case
```

```
[04/25/22]seed@VM:~/.../task5$ gcc task5.c -lcrypto
[04/25/22]seed@VM:~/.../task5$ a.out
Verification fails!
```

กรณีที่ เลขฐาน 16 ไม่ตรงแล้วนำไป Encrypt แล้วนำไปเช็คเงื่อนไขแล้วไม่ตรงจะขึ้นข้อความ Verification fails