**กลุ่ม : G01 ข้อที่ 2-5**

6509611809 ธีรภัทร ศิริธรรม

6509611544 กิตติธรา สุทธาภิรมย์

6509611858 ประพล ขาวสอาด

## Header

```
#include <stdio.h>
#include <string.h>
#include <openssl/bn.h>
```

ติดตั้ง เตรียมพรอมสภาพแวดลอม include openssl library สำหรับจัดการตัวเลขขนาดใหญ

## Utility Functions

```
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);
}

void hexToString(const char *hex, char *output) {

    // Return if string is null
    if (hex == NULL || output == NULL) return;

    // Return if character is not 8 bits
    size_t len = strlen(hex);
    if (len % 2 != 0) {
        output[0] = '\0';
        return;
    }

    char temp[3] = {0};
    char *ptr = output;

    // Loop each pair of character in hex
    for (size_t i = 0; i < len; i += 2) {
```

```
        // Pick 2 character and convert to long
        temp[0] = hex[i];
        temp[1] = hex[i + 1];
        // Convert back to char
        *ptr = (char)strtol(temp, NULL, 16);
        ptr++;
    }

    // End string
    *ptr = '\0';
}

void stringToHex(const char *input, char *output) {

    // Loop each character in string
    while (*input) {
        // print to character to hex using sprintf
        sprintf(output, "%02X", (unsigned char)*input);
        input++;
        output += 2;
    }
    *output = '\0';
}
```

printBN มีไว้เพื่อ print BIGNUM

hexToString มีไว้เพื่อแปลง string ที่ encode hexadecimal เป็น decadecimal

stringToHex มีไว้เพื่อแปลง string decadecimal เป็น hexadecimal

Task 2

คำสั่ง

```
void task2(){
    //TASK 2

    // Convert string to hex using stringToHex function (plaintext ->
hex_plainText)
    const char* plainText = "A top secret!";
    char hex_plainText[strlen(plainText)*2 + 1];
    stringToHex(plainText,hex_plainText);

    // printf("%s\n",hex_plainText);


    // Create BIGNUM
    BN_CTX *ctx = BN_CTX_new();

    BIGNUM* M = BN_new(); // message
    BIGNUM* n = BN_new(); // n
    BIGNUM* e = BN_new(); // e (public key)
    BIGNUM* d = BN_new(); // d (private key)
    BIGNUM* encrypted_m = BN_new(); // encrypted message
    BIGNUM* check_message = BN_new(); // confirmation message

    // Assign value to variables
    BN_hex2bn(&M, hex_plainText);
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    printf("Task 2:\n");

    //Encrypting Message
    BN_mod_exp(encrypted_m , M , e , n , ctx); // encrypted_m = M^e (mod n)

    printBN("Encrypted message:" , encrypted_m);


    //Validating by decrypt into plain text
    BN_mod_exp(check_message , encrypted_m , d , n , ctx); // check_message =
C^d (mod n)


    char* check_message_string = BN_bn2hex(check_message);
    char check_s[strlen(plainText)+1];
```

```
    hexToString(check_message_string,check_s);
    printf("checking by decrypting to plain message: %s\n",check_s);

    //Free memory
    OPENSSL_free(check_message_string);
    BN_CTX_free(ctx);
    BN_free(M);
    BN_free(e);
    BN_free(n);
    BN_free(d);
    BN_free(encrypted_m);
    BN_free(check_message);
}
```

ผลลัพธ์ที่ได้

```
Task 2:
Encrypted message: 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
checking by decrypting to plain message: A top secret!
```

อธิบายผลลัพธ์

สิ่งที่แสดงออกมาคือ ข้อความที่เข้ารหัส กับ ข้อความที่ถอดรหัสจากข้อความที่เข้ารหัส

สิ่งที่โค้ดทำคือการนำ string ในตัวแปร plainText แปลงเป็น hexadecimal ด้วยฟังก์ชัน stringToHex เก็บไว้ในตัวแปร hex_plainText จากนั้นนำ hex_plainText มา encrypt ด้วย public key e, n ด้วยสูตร

$$C = M^e \ (mod\ n)$$

โดย C คือ ข้อความที่เข้ารหัส

M คือ ข้อความที่ต้องการเข้ารหัสในรูปแบบฐาน 16 (hex_plainText)

e, n คือ public key ที่โจทย์กำหนด

e = 010001 (65537 ในฐาน 16)

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

โดยใช้ฟังก์ชัน BN_mod_exp(ans, base, exp, n , ctx); ซึ่งมีค่าเท่ากับ

$$ans = base^{exp} \ (mod\ n)$$

นำผลลัพธ์ที่ได้ใส่ในตัวแปร encrypted_m จากนั้นแสดงผลลัพธ์ด้วยฟังก์ชัน printBN

โดยผลลัพธ์ที่ได้คือ

6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC

เมื่อได้ผลลัพธ์จะทำการตรวจสอบผลลัพธ์โดยการ decrypt ด้วยสูตร

$$M = C^d (mod\ n)$$

โดย M คือ ข้อความที่ถอดรหัสในรูปแบบฐาน 16

C คือ ข้อความที่เข้ารหัส

d, n คือ private key ที่โจทย์กำหนด

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

เมื่อได้ข้อความที่ถอดรหัสแล้ว ต้องแปลงเป็นรูปแบบฐาน 10 จึงจะสามารถแปลงกลับมาเป็น ascii character ได้ด้วยฟังก์ชัน hexToString

โดยผลลัพธ์ที่ได้คือ A top secret!

สุดท้ายเมื่อทำงานเสร็จแล้ว ทำการ free memory ของตัวแปรต่าง ๆ

Task 3

คำสั่ง

```
void task3(){
    //TASK 3

    // Create and assign variables
    const char *cipher_string =
"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F";
    BN_CTX* ctx = BN_CTX_new();
    BIGNUM* C = BN_new();
    BIGNUM* plainText_3 = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* d = BN_new();
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    BN_hex2bn(&C , cipher_string);

    printf("Task 3:\n");

    //Decrypting Message
    BN_mod_exp(plainText_3 , C , d , n , ctx); // plainText_3 = C^d (mod n)
    printBN("Decrypted Message (Hex):",plainText_3);

    //Convert Hex to string form
    char* decrypt_hex = BN_bn2hex(plainText_3);
    char decrypt_message[strlen(decrypt_hex) + 1];
    hexToString(decrypt_hex,decrypt_message);
    printf("Decrypted message (string): %s\n",decrypt_message);

    //Free memory
    BN_CTX_free(ctx);
    BN_free(C);
    BN_free(plainText_3);
    BN_free(n);
    BN_free(d);
    OPENSSL_free(decrypt_hex);
}
```

ผลลัพธ์ที่ได้

```
Task 3:
Decrypted Message (Hex): 50617373776F726420697320646565573
Decrypted message (string): Password is dees
```

อธิบายผลลัพธ์

สิ่งที่แสดงออกมาคือ ข้อความที่ถอดรหัสในรูปเลขฐาน 16 และ ข้อความที่ถอดรหัสในรูปเลขฐาน 10 เป็นรหัส ascii

สิ่งที่โค้ดทำคือการนำ string ในตัวแปร cipher_string ที่อยู่ในรูป hexadecimal แปลงเป็นตัวแปร BigNum ด้วยฟังก์ชัน BN_hex2bn เก็บไว้ในตัวแปร C จากนั้นนำตัวแปร C มา decrypt ด้วยสูตร

$$M = C^d (mod\ n)$$

โดย M คือ ข้อความที่ถอดรหัสในรูปแบบฐาน 16

C คือ ข้อความที่เข้ารหัส

d, n คือ private key ที่โจทย์กำหนด

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

โดยใช้ฟังก์ชัน BN_mod_exp(plainText_3, C, d, n, ctx);

ผลลัพธ์ที่ได้คือ 50617373776F726420697320646565573

เมื่อได้ข้อความที่ถอดรหัสแล้ว ต้องแปลงเป็นรูปแบบฐาน 10 จึงจะสามารถแปลงกลับมาเป็น ascii character ได้ด้วยฟังก์ชัน hexToString

ได้ผลลัพธ์คือ Password is dees

สุดท้ายเมื่อทำงานเสร็จแล้ว ทำการ free memory ของตัวแปรต่าง ๆ

Task 4

คำสั่ง

```
void task4(){

    // Create variables
    const char* message = "I owe you $2000."; // Plaintext
    char hex_message[strlen(message)*2 + 1];
    stringToHex(message,hex_message); // Plaintext to hexadecimal
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM* M = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* d = BN_new();
    BIGNUM* signature_2000 = BN_new();
    BIGNUM* signature_3000 = BN_new();

    // Assign variables
    BN_hex2bn(&M , hex_message);    // M
    BN_hex2bn(&e, "010001");        // E
    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5"); // n
    BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D"); // d


    //Signing
    BN_mod_exp(signature_2000, M , d , n , ctx); // signature_2000 = M^d (mod
n)

    //Create alternative Message and sign
    const char* message_modify = "I owe you $3000."; // new plaintext
    stringToHex(message_modify,hex_message); // new plaintext to hex
    BN_hex2bn(&M , hex_message); // M = new plaintext
    BN_mod_exp(signature_3000, M , d , n , ctx); // signature_3000 = M^d (mod
n)

    printf("Task 4:\n");
    printf("Signature (I owe you 2000$): ");
    BN_print_fp(stdout , signature_2000);
    printf("\nModified Signature (I owe you 3000$): ");
    BN_print_fp(stdout , signature_3000);
    printf("\n");

    //Free memory
    BN_CTX_free(ctx);
    BN_free(M);
```

```
    BN_free(e);
    BN_free(n);
    BN_free(d);
    BN_free(signature_2000);
    BN_free(signature_3000);
}
```

ผลลัพธ์ที่ได้

```
Task 4:
Signature (I owe you 2000$): 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
Modified Signature (I owe you 3000$): BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
```

อธิบายผลลัพธ์

ผลลัพธ์ที่ได้คือ signature ของข้อความ “I owe you 2000$” และ signature ของข้อความ “I owe you 3000$” โดยหลักการทำงานข้อการ encrypt ข้อความด้วย private key d, n ด้วยสูตร

$$S = M^d \ (mod\ n)$$

โดย S คือ ข้อความที่ถูก sign

M คือ ข้อความที่ต้องการ sign ในรูปแบบฐาน 16 (signature_2000/signature_3000)

d, n คือ private key ที่โจทย์กำหนด

d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

โดยใช้ฟังก์ชัน BN_mod_exp(signature_2000, M , d , n , ctx); และ BN_mod_exp(signature_3000, M , d , n , ctx);

โดยสิ่งที่สังเกตเห็นได้จากการเปลี่ยนข้อความ 1 ตัวอักษรคือได้ผลลัพธ์ที่ต่างกันเกือบทั้ง string ดังนั้นจึงทราบว่าการเปลี่ยนข้อความที่จะ sign มีผลต่อผลลัพธ์เป็นอย่างมาก

Task 5

คำสั่ง

```
void task5(){
    const char* message = "Launch a missle."; //  message M
    const char* signature_message =
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F";
//  signature S

    char message_hex[strlen(message)*2 + 1];

    //convert plaintext to hex
    stringToHex(message,message_hex);
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM* S = BN_new();
    BIGNUM* n = BN_new();
    BIGNUM* e = BN_new();
    BIGNUM* result = BN_new();
    BIGNUM* corrupt_S = BN_new();
    BIGNUM* corrupt_res = BN_new();


    BN_hex2bn(&S,signature_message);
    BN_hex2bn(&n,
"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115"); // given
n
    BN_hex2bn(&e, "010001"); // given e

    //decrpyting signature
    BN_mod_exp(result, S , e , n , ctx); //result = S^e (mod n)

    printf("Task 5:\n");
    printf("Message: %s\n",message); // given message M
    printf("Message (hex): %s\n\n",message_hex); // M in hex
    printf("Signature : %s\n" , signature_message); // given signature
    printf("Decrypted Signature: ");
    BN_print_fp(stdout ,result); // signature after decrypting
    printf("\n\n");

    //create alternative corrupt signature
    const char* corrupt_signature =
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F"; //
corrupted signature
    BN_hex2bn(&corrupt_S,corrupt_signature);
    BN_mod_exp(corrupt_res , corrupt_S , e , n ,ctx); // corrupt_res =
corrupt_S^e (mod n)
```

```
    printf("Corrupt Signature: %s\n",corrupt_signature); // corrupted
signature
    printf("Decrypted Corrupt Signature: ");
    BN_print_fp(stdout ,corrupt_res); // decrypt corrupt signature
    printf("\n");

    //Free memory
    BN_CTX_free(ctx);
    BN_free(S);
    BN_free(n);
    BN_free(e);
    BN_free(result);
    BN_free(corrupt_res);
    BN_free(corrupt_S);
}
```

ผลลัพธ์ที่ได้

```
Task 5:
Message: Launch a missle.
Message (hex): 4C61756E63682061206D6973736C652E

Signature : 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
Decrypted Signature: 4C61756E63682061206D697373696C652E

Corrupt Signature: 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F
Decrypted Corrupt Signature: 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
```

อธิบายผลลัพธ์

สิ่งที่แสดงผลบรรทัดแรกคือ ข้อความ M “Launch a missle.”

บรรทัดที่ 2 คือ ข้อความ M ที่ถูกแปลงเป็นเลขฐาน 16 “4C61756E63682061206D6973736C652E”

บรรทัดที่ 3 คือ signature S ที่ได้รับมาจาก Alice

“643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F”

บรรทัดที่ 4 คือ signature S ที่ decrypt ด้วย public key e, n (e = e = 010001, n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115)

“4C61756E63682061206D697373696C652E”

โดยการตรวจสอบว่าข้อความที่ได้รับถูกต้องไหม ทำโดยการนำ signature S ของ Alice มา decrypt ด้วย public key e, n ซึ่งจะต้องได้ผลลัพธ์เท่ากับข้อความ M ในรูปฐาน 16

$$M\ (mod\ n) = S^e (mod\ n) = (M^d)^e\ (mod\ n)$$

ซึ่ง $M\ (mod\ n) = (M^d)^e\ (mod\ n)$ เสมอ ดังนั้นหาก signature ที่ decrypt ด้วย public key จะต้องมีค่าเท่ากับ message ซึ่งจะเห็นได้ว่าผลลัพธ์ในบรรทัดที่ 2 เท่ากับผลลัพธ์ของบรรทัดที่ 4 ดังนั้น signature นี้จึงเป็นของ Alice

เมื่อเราลองแก้ไข signature 1 bit จาก “643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F” เป็น “643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F” แล้วทำการ decrypt ออกมา จะเห็นผลลัพธ์ในบรรทัดที่ 6 ได้ว่าผลลัพธ์แตกต่างกับข้อความ M โดยสิ้นเชิงถึงแม้ว่าจะเป็นข้อมูลเพียงแค่ 1 bit ทำให้ได้ทราบว่าการใช้ signature ทำให้ตรวจสอบได้ว่าการส่งข้อมูลมีความผิดพลาดไหม และช่วยในการยืนยันตัวตนได้