**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 เตรียมพรอมสภาพแวดลอม 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

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

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

นำผลลัพธ์ที่ได้ใส่ในตัวแปร encrypted\_m จากนั้นแสดงผลลัพธ์ด้วยฟังก์ชัน printBN

โดยผลลัพธ์ที่ได้คือ 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC

เมื่อได้ผลลัพธ์จะทำการตรวจสอบผลลัพธ์โดยการ decrypt ด้วยสูตร

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);

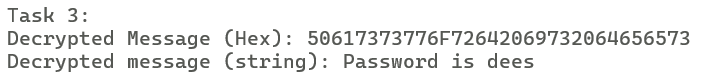
    BN\_free(n);

    BN\_free(d);

    OPENSSL\_free(decrypt\_hex);

}

ผลลัพธ์ที่ได้



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

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 50617373776F72642069732064656573

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);

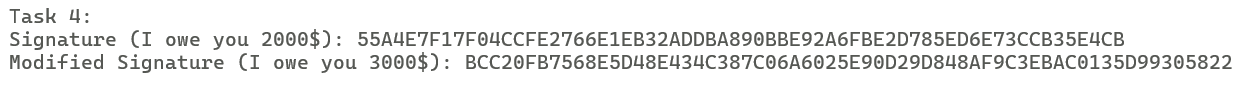
    BN\_free(d);

    BN\_free(signature\_2000);

    BN\_free(signature\_3000);

}

ผลลัพธ์ที่ได้



อธิบายผลลัพธ์

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

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

A number and text on a white background

AI-generated content may be incorrect.

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

ซึ่ง มีค่าเท่ากับ เสมอ ดังนั้นหาก signature ที่ decrypt ด้วย public key จะต้องมีค่าเท่ากับ message ซึ่งจะเห็นได้ว่าผลลัพธ์ในบรรทัดที่ 2 เท่ากับผลลัพธ์ของบรรทัดที่ 4 ดังนั้นsignature นี้จึงเป็นของ Alice

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