Cryptography Program #3

소프트웨어학부

2018044720 석예림

1. mod.c 소스코드

```c
/*
 * Copyright 2020. Heekuck Oh, all rights reserved
 * 이 프로그램은 한양대학교 ERICA 소프트웨어학부 재학생을 위한 교육용으로 제작되었습니다.
 */
#include <stdio.h>
#include <stdint.h>
/*
 * mod_add() - computes a+b mod m
 * 만일 a+b에서 오버플로가 발생하면 값이 틀리게 된다.
 * 이런 경우에는 m을 빼서 오버플로가 생기지 않게 한다. 즉, a+(b-m)을 계산하면 된다.
 * 오버플로는 a+b >= b가 성립하지 않을 때 발생한다.
 */
uint64_t mod_add(uint64_t a, uint64_t b, uint64_t m)
{
    if(a+b < b) b = b - m;
    return ((a % m) + (b % m)) % m;
}

/*
 * mod_sub() - computes a-b mod m
 * 만일 a < b이면 결과가 음수가 되므로 m을 더해서 양수로 만든다.
 */
uint64_t mod_sub(uint64_t a, uint64_t b, uint64_t m)
{
    if(a < b) return ((a % m) - (b % m) + m) % m;
    return ((a % m) - (b % m)) % m;
}

/*
 * mod_mul() - computes a*b mod m
 * a*b에서 오버플로가 발생할 수 있기 때문에 덧셈을 사용하여 빠르게 계산할 수 있는
 * "double addition" 알고리즘을 사용한다. 그 알고리즘은 다음과 같다.
 *     r = 0;
 *     while (b > 0) {
 *         if (b & 1)
 *             r = mod_add(r, a, m);
 *         b = b >> 1;
 *         a = mod_add(a, a, m);
 *     }
 */
uint64_t mod_mul(uint64_t a, uint64_t b, uint64_t m)
{
    uint64_t r = 0;
    while (b > 0) {
        if (b & 1)
            r = mod_add(r, a, m);
        b = b >> 1;
        a = mod_add(a, a, m);
    }
    return r;
}
```

```c
/*
 * mod_pow() - computes a^b mod m
 * a^b에서 오버플로가 발생할 수 있기 때문에 곱셈을 사용하여 빠르게 계산할 수 있는
 * "square multiplication" 알고리즘을 사용한다. 그 알고리즘은 다음과 같다.
 *     r = 1;
 *     while (b > 0) {
 *         if (b & 1)
 *             r = mod_mul(r, a, m);
 *         b = b >> 1;
 *         a = mod_mul(a, a, m);
 *     }
 */
uint64_t mod_pow(uint64_t a, uint64_t b, uint64_t m)
{
    uint64_t r = 1;
    while (b > 0) {
        if (b & 1)
            r = mod_mul(r, a, m);
        b = b >> 1;
        a = mod_mul(a, a, m);
    }
    return r;
}

int main(void)
{
    uint64_t a, b, m;

    a = 1234; b = 5678; m = 3456;
    printf("<덧셈> ");
    printf("%llu + %llu mod %llu = %llu\n", a, b, m, mod_add(a,b,m));
    printf("<뺄셈> ");
    printf("%llu - %llu mod %llu = %llu\n", a, b, m, mod_sub(a,b,m));
    printf("<곱셈> ");
    printf("%llu * %llu mod %llu = %llu\n", a, b, m, mod_mul(a,b,m));
    printf("<지수> ");
    printf("%llu ^ %llu mod %llu = %llu\n", a, b, m, mod_pow(a,b,m));
    printf("---\n");
    a = 3684901700; b = 3904801120; m = 4294901760;
    printf("<덧셈> ");
    printf("%llu + %llu mod %llu = %llu\n", a, b, m, mod_add(a,b,m));
    printf("<뺄셈> ");
    printf("%llu - %llu mod %llu = %llu\n", a, b, m, mod_sub(a,b,m));
    printf("<곱셈> ");
    printf("%llu * %llu mod %llu = %llu\n", a, b, m, mod_mul(a,b,m));
    printf("<지수> ");
    printf("%llu ^ %llu mod %llu = %llu\n", a, b, m, mod_pow(a,b,m));
    printf("---\n");
    a = 18446744073709551360u;
    b = 18446744073709551598u;
    m = 18441921395520346504u;
    printf("<덧셈> ");
    printf("%llu + %llu mod %llu = %llu\n", a, b, m, mod_add(a,b,m));
    printf("<뺄셈> ");
    printf("%llu - %llu mod %llu = %llu\n", a, b, m, mod_sub(a,b,m));
    printf("<곱셈> ");
    printf("%llu * %llu mod %llu = %llu\n", a, b, m, mod_mul(a,b,m));
    printf("<지수> ");
    printf("%llu ^ %llu mod %llu = %llu\n", a, b, m, mod_pow(a,b,m));
}
```

2. 코드 내 함수 설명

- uint64_t mod_add(uint64_t a, uint64_t b, uint64_t m) : a + b mod m 계산하는 함수. a+b mod m = ((a % m) + (b % m)) % m 으로 계산해도 가능하다. a+b<b 일때 오버플로가 발생하기 때문에 b 대신 b-m 을 넣어 계산해준다.
- uint64_t mod_sub(uint64_t a, uint64_t b, uint64_t m) : a – b mod m 계산하는 함수. a-b mod m = ((a % m) - (b % m)) % m 으로 가능하다. a < b 일때 결과가 음수가 되므로 (a % m) - (b % m) 빼기 해준 상태에서 m 을 더해 양수로 만들어준 후 mod m 해 준다.
- uint64_t mod_mul(uint64_t a, uint64_t b, uint64_t m) : a * b mod m 계산하는 함수. 오버플로 발생을 막기 위해 덧셈을 사용하여 계산하는 Double addition 알고리즘을 사용한다. Double addition 알고리즘은 b 가 0 이상이고 b 의 해당 비트가 1 이면 나머지 r 에다가 r+a mod m 해주고 b 비트를 하나 옮기고 a 에 a 를 더해 mod m 해주며 b 의 값이 0 보다 작아질 때까지 계산해 준다.
- uint64_t mod_pow(uint64_t a, uint64_t b, uint64_t m) : a ^ b mod m 계산하는 함수. Square multiplication 알고리즘을 사용한다. a^b mod m = (a mod m * a mod m * ... * a mod m) mod m 로 계산되기 때문에 mod_mul 함수를 사용하여 a 의 지수제곱한 결과에 mod m 해준 값을 b 의 비트가 1 일때 나머지와 a 를 곱해 계산한다.

3. 실행 결과

```
yerim ~
❯ cd Downloads/2020\ 암호학/프로그램\ \#3
yerim ~/Downloads/2020 암호학/프로그램 #3
❯ gcc mod.c
yerim ~/Downloads/2020 암호학/프로그램 #3
❯ ./a.out
<덧셈> 1234 + 5678 mod 3456 = 0
<뺄셈> 1234 - 5678 mod 3456 = 2468
<곱셈> 1234 * 5678 mod 3456 = 1340
<지수> 1234 ^ 5678 mod 3456 = 1792
---
<덧셈> 3684901700 + 3904801120 mod 4294901760 = 3294801060
<뺄셈> 3684901700 - 3904801120 mod 4294901760 = 4075002340
<곱셈> 3684901700 * 3904801120 mod 4294901760 = 2417663360
<지수> 3684901700 ^ 3904801120 mod 4294901760 = 1734737920
---
<덧셈> 18446744073709551360 + 18446744073709551598 mod 18441921395520346504 = 9645356378409950
<뺄셈> 18446744073709551360 - 18446744073709551598 mod 18441921395520346504 = 18441921395520346266
<곱셈> 18446744073709551360 * 18446744073709551598 mod 18441921395520346504 = 14923616227936587640
<지수> 18446744073709551360 ^ 18446744073709551598 mod 18441921395520346504 = 6550219153064247488
yerim ~/Downloads/2020 암호학/프로그램 #3
❯
```