Cryptography Project #2

소프트웨어학부

2018044720 석예림

1. mRSA.c 소스코드

```c
/*
 * Copyright 2020. Heekuck Oh, all rights reserved
 * 이 프로그램은 한양대학교 ERICA 소프트웨어학부 재학생을 위한 교육용으로 제작되었습니다.
 */
#include <stdlib.h>
#include <stdint.h>
#include "mRSA.h"

static uint64_t gcd(uint64_t a, uint64_t b)
{
    uint64_t tmp;
    while(b != 0){
        tmp = a % b;
        a = b;
        b = tmp;
    }
    return a;
}

static uint64_t mod_add(uint64_t a, uint64_t b, uint64_t m)
{
    a = a % m;
    b = b % m;
    if(a + b < b) return ((a % m) + (b % m) - m) % m;
    return ((a % m) + (b % m)) % m;
}

static uint64_t mod_sub(uint64_t a, uint64_t b, uint64_t m)
{
    a = a % m;
    b = b % m;
    if(a < b) return ((a % m) - (b % m) + m) % m;
    return ((a % m) - (b % m)) % m;
}

static 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;
}

static 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;
}

static uint64_t mul_inv(uint64_t a, uint64_t m)
{
    uint64_t d0 = a, d1 = m;
    uint64_t x0 = 1, x1 = 0;
    uint64_t q = 0, tmp;

    while(d1 > 1){
        q = d0 / d1;

        tmp = mod_sub(d0 ,mod_mul(q, d1, m), m);
        d0 = d1;
        d1 = tmp;

        tmp = mod_sub(x0, mod_mul(q, x1, m), m);
        x0 = x1;
        x1 = tmp;
    }

    if(d1 == 1) return (x1 > 0 ? x1 : x1 + m);

    return 0;
}

static int miller_rabin(uint64_t n)
{
    uint64_t a[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
    uint64_t i, j, q, k, l, flg;

    for(i = 0; i < 12; i++){
        if(n == a[i]) return PRIME;
    }

    q = (n - 1);
```

```
    k = 0;

    while(k % 2 == 0){
        q = q / 2;
        k = k + 1;
    }

    for(i = 0; i < 12; i++){
        flg = COMPOSITE;
        if(mod_pow(a[i], q, n) == 1) flg = PRIME;
        l = 1;
        for(j = 0; j < k; j++){
            if(mod_pow(mod_pow(a[i], q, n),l, n) == n-1){
                flg = PRIME;
            }
            l *= 2;
        }
        if(flg == COMPOSITE) return COMPOSITE;
    }
    return flg;
}

/*
 * mRSA_generate_key() - generates mini RSA keys e, d and n
 * Carmichael's totient function Lambda(n) is used.
 */
void mRSA_generate_key(uint64_t *e, uint64_t *d, uint64_t *n)
{
    uint64_t p = 0;
    uint64_t q = 0;
    uint64_t Lambda = 0;

    while (p * q < MINIMUM_N) // p, q 랜덤 선택
    {
        while (1)
        {
            arc4random_buf(&p, sizeof(uint32_t));
            if (miller_rabin(p)) break;
        }

        while (1)
        {
            arc4random_buf(&q, sizeof(uint32_t));
            if (miller_rabin(q)) break;
        }
    }
    *n = p * q; // n 생성

    Lambda = ((p - 1)*(q - 1))/gcd(p - 1, q - 1);

    while (1) // e, d 생성
```

```
    {
        arc4random_buf(e,sizeof(uint64_t));
        if ((1 < *e) && (*e < Lambda) && (gcd(*e,Lambda) == 1))
        {
            *d = mul_inv(*e,Lambda);
            if ((1<*d) && (*d<Lambda)) break;
        }
    }
}

/*
 * mRSA_cipher() - compute m^k mod n
 * If data >= n then returns 1 (error), otherwise 0 (success).
 */
int mRSA_cipher(uint64_t *m, uint64_t k, uint64_t n)
{
    if(*m >= n) return 1;
    *m = mod_pow(*m, k, n);
    return 0;
}
```

2. 코드 내 함수 설명

- gcd, mod_add, mod_sub, mod_mul, mod_pow, mod_inv, miller_rabin 함수는 앞에서 한 프로그램과 프로젝트와 동일
- mRSA_generate_key : arc4random_buf()를 사용해서 랜덤한 수를 선택해 주고 그 수가 prime 인지 miller_rabin()함수로 확인해 주고, p*q>= MNIMUM_N 일때 까지 p 와 q 를 구한다. 이렇게 구한 p, q 를 곱하여 n 을 생성해준다.

  Euler totient function 대신 Carmichael's totient function 인 Lambda 를 사용하여 키를 생성한다. 그다음 1 < e < $\emptyset(n)$, gcd(e, $\emptyset(n)$) = 1 인 랜덤한 e 를 선택하고, ed ≡ 1 mod $\emptyset(n)$를 만족하는 d 를 구하면 암호화에 필요한 키가 생성된다.
- mRSA_cipher : m^k mod n 을 계산하는데 m >= n 이면 오류처리 해주고 오류가 발생하지 않으면 0 을 반환한다.

3. 실행 결과

```
yerim ~
❯ cd Downloads/2020-2\ 암호학/프로젝트\ \#2
yerim ~/Downloads/2020-2 암호학/프로젝트 #2
❯ make
gcc -Wall -c test.c
gcc -Wall -c mRSA.c
gcc -Wall -o test test.o mRSA.o
yerim ~/Downloads/2020-2 암호학/프로젝트 #2
❯ ./test
e = 6ab77637e2f401c3
d = 1353fc2740d55f07
n = da3c4499608f20e3
m = 0, c = 0, v = 0
m = 1, c = 1, v = 1
m = 2, c = 570387526056030050, v = 2
m = 3, c = 11336044852648805110, v = 3
m = 4, c = 7045631392743041378, v = 4
m = 5, c = 11412037967413525565, v = 5
m = 6, c = 16932216917983658 37, v = 6
m = 7, c = 11269861752836292805, v = 7
m = 8, c = 15245566397395875090, v = 8
m = 9, c = 10260739228264497523, v = 9
m = 10, c = 7084242121789812888, v = 10
m = 11, c = 10922383689966538702, v = 11
m = 12, c = 6792933971527304525, v = 12
m = 13, c = 1976085408432841918, v = 13
m = 14, c = 13634396414391401995, v = 14
m = 15, c = 3776736997957487282, v = 15
m = 16, c = 13440110267894156085, v = 16
m = 17, c = 6694208882839649624, v = 17
m = 18, c = 7085472810863917363, v = 18
m = 19, c = 12567535397375147829, v = 19
e = 03d237c1c13484db
d = 0248524b4efe88e5
n = 9b87be7e63de57ad
m = ba2de19494701973, m may be too big
m = 45fe236571735e3b, c = 6872ce89e61df5d6, v = 45fe236571735e3b
m = aac6a94848179b25, m may be too big
m = b747ac54ba703cc4, m may be too big
m = 5b85ecb044c2961f, c = 21c54bbe494f504d, v = 5b85ecb044c2961f
m = b094a4db3b54c6a3, m may be too big
m = 2de9206bdfb41eb0, c = 0ac6eb1989ed82ea, v = 2de9206bdfb41eb0
m = 77c1c633729a5f59, c = 4d189938c7ebedf2, v = 77c1c633729a5f59
m = 7c6fb229f9fdeb52, c = 2a147587b1204f02, v = 7c6fb229f9fdeb52
m = 54fcaaea101b5435, c = 54a76e67f560551e, v = 54fcaaea101b5435
m = e97d1055151779e4, m may be too big
m = 1ba60d6b891810c0, c = 0a02174dc290d323, v = 1ba60d6b891810c0
m = 0d658582e1c2e0e3, c = 0ecbb276d902332f, v = 0d658582e1c2e0e3
m = eca171289aab22e5, m may be too big
m = 74a34c1895f88929, c = 866827d1164a2fbc, v = 74a34c1895f88929
m = 37683aa2c7cc630a, c = 05f5feed23a933e0, v = 37683aa2c7cc630a
m = 088ba572a2270cc4, c = 4cb209fb0570c3c8, v = 088ba572a2270cc4
m = 65bcef4f01afc80e, c = 15a84b698233e5e1, v = 65bcef4f01afc80e
m = 5cc80d026a3e0f20, c = 2d2b178696572ba1, v = 5cc80d026a3e0f20
m = f98069dc21254210, m may be too big
Random testing.................No error found
yerim ~/Downloads/2020-2 암호학/프로젝트 #2
❯
```