



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Криптографія

Комп'ютерний практикум №5

“ Вивчення криптосистеми RSA та алгоритму електронного підпису; ”

Перевірив:

Чорний О.М.

Завадська Л.О.

Савчук М.М.

Виконали:

Студенти групи ФБ-71


Новик Л.А.

Равкін Д.Б.

Мета роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Порядок виконання роботи

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.
2. За допомогою цієї функції згенерувати дві пари простих чисел і довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб p і q – прості числа для побудови ключів абонента A , і – абонента B .
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ та відкритий ключ. За допомогою цієї функції побудувати схеми RSA для абонентів A і B .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B . Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання.
За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів A і B , перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.
5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа. nk  0

ЧИСЛА p і q :

p :237182050498247112577768247047601284093634261176061653795972235802586212220705912849
 q ::1173056682342418460855248456434410842083088198036862944870263840768654170364141670517
 $p1$:295666785430523279005606194866148572871051832408048359462455722216871093413099838209
 $q1$:1543463312197651954139470435111904943634986886034917598963586401126387053933497774961

Generate Key Pair A:

e : 65537

n :

278227989268645717370658878417515819791288712224929275944514181601407091285757219254933358283
267884336022901670814820981423168704200255721931819961999761731814274772933

d :

819226621437822361124495228164670762243083796909297074614536851299625042425081566128329895778
65847390965363233205993751311330594771206016229854467508646664877553999681

Generate Key Pair B:

e :65537

$n1$:456350835947427924021571436645901633296304776872924127739512595602546688981685028226881408
15013367782222927265512701363425810701976272419293455951500190254850591284849
 $d1$:2059313002168123872776287237341638296398205207934427598725658101888965894066611688775409361
29587831813622661316436542341346191260524802863629596968947512247239316856833

VT:

7777777

Шифруєм

C:

233217038661022196140269814700830361507923520797156086875735798875687699791819049683988347585
826239768917152345537372390356974972624716830287626909680146595453963062833

Расшифровываем
M: 7777777

k:123456789

Send: k, n1, n, d, e

S:2902464745215987151432767769901929843787947183703130679060873863716738419418296402109154815
1924045411639343216613691816492192090774466667480101719386220766023090963209

S1:414533015285653432543973241847731281746007634574555261018576428837465149927081635859783179
921970422352657238757674434166239086721532120340517010538258126302651583215240

k1:379279967573977596554267632746295274603153340778154058204085936265178754438131892758233446
29738619865115207680463959669928365443849651945121073012636573148350422381418

Receive:k1, S1, d1, n1, e, n

k:123456789

S:2902464745215987151432767769901929843787947183703130679060873863716738419418296402109154815
1924045411639343216613691816492192090774466667480101719386220766023090963209

k:123456789

digital true

КОД

```
#include <iostream>
```

```
#include <NTL/ZZ.h>
```

```
NTL_CLIENT
```

```
using namespace std;
```

```
ZZ gcd(ZZ a, ZZ b, ZZ& x, ZZ& y) {
```

```
    if (a == 0) {
```

```
        x = 0; y = 1;
```

```
        return b;
```

```
    }
```

```
    ZZ x1, y1;
```

```
    ZZ d = gcd(b % a, a, x1, y1);
```

```
    x = y1 - (b / a) * x1;
```

```
    y = x1;
```

```
    return d;
```

```
}
```

```
long simplicity_test(const ZZ& n, long t)
```

```
{
```

```
    if (n <= 1) return 0;
```

```
    long p;
```

```

PrimeSeq s;

p = s.next();

while (p && p < 2000) {
    if ((n % p) == 0) return (n == p);
    p = s.next();
}

long i;

ZZ x;

for (i = 0; i < t; i++) {
    x = RandomBnd(n);

    ZZ m, y, z;

    long j, k;

    if (x == 0) return 0;

    m = n / 2; k = 1;

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

    z = PowerMod(x, m, n); // z = x^m % n

    if (z == 1) return 0;

    j = 0;

    do {
        y = z;
        z = (y * y) % n;
        j++;
    } while (j < k && z != 1);

    if(z != 1 || y != n - 1)return 0;
}

return 1;
}

void GenerateKeyPair(ZZ p,ZZ q,ZZ& e,ZZ& n,ZZ& d, ZZ& fi)
{

```

```

ZZ x,y,pq;

e = 65537;

pq = p * q;

fi = (p - 1) * (q - 1);

ZZ g = gcd(e, fi, x, y);

//cout << "d=" << g << endl;

if (g == 1){
    x = (x % fi + fi) % fi;

    // cout << "obratnoe:" << x << endl;
}

if (x > 0)
    while (x > fi) x = x - fi;

if (x < 0)
    while (x < 0) x = x + fi;


d = x;

n = pq;
}

void Encrypt(ZZ& C, ZZ M, ZZ e, ZZ n)
{
    C = PowerMod(M, e, n);
}

void Decrypt(ZZ& M, ZZ C, ZZ d, ZZ n)
{
    M = PowerMod(C, d, n);
}

void Sign(ZZ& S, ZZ M, ZZ d, ZZ n)
{
    S = PowerMod(M, d, n);
}

ZZ Verify(ZZ M, ZZ S, ZZ e, ZZ n)
{
    M = PowerMod(S, e, n);
}

```

```

    return M;
}

void Send(ZZ& k1, ZZ& S1, ZZ k, ZZ& S, ZZ n1, ZZ n, ZZ d, ZZ e)
{
    S= PowerMod(k, d, n);
    std::cout << "S:" << S << endl;
    S1= PowerMod(S, e, n1);
    std::cout << "S1:" << hex << S1 << endl;
    k1 = PowerMod(k, e, n1);
    std::cout << "k1:" << hex << k1 << endl;
}

void Receive(ZZ k, ZZ S, ZZ k1, ZZ S1, ZZ d1, ZZ n1, ZZ e, ZZ n)
{
    k = PowerMod(k1, d1, n1);
    std::cout << "k:" << hex << k << endl;
    ZZ buff;
    buff = k;
    S = PowerMod(S1, d1, n1);
    std::cout << "S:" << hex << S << endl;
    k = PowerMod(S, e, n);
    std::cout << "k:" << hex << k << endl;
    if (buff == k) { std::cout << "digital true\n"; }
    else { std::cout << "digital false\n"; }
}

ZZ random(ZZ a, ZZ b) {
    return RandomBnd(b - a) + a ;
}

int main()
{
    cout.setf(ios::showbase);
    const std::string hex("0x31c3");
    //std::istream stream(hex);

```

```

setlocale(LC_ALL, "Russian");

ZZ c_256;

c_256 = 2;

for (int i = 2; i <= 256; i++)
    c_256 = c_256 * 2;

ZZ c_260 = c_256 * 2*2*2*2*1024*1024;

//cout << c_256 << "\n";

ZZ n1, n2, n3, n4;

//n1 = RandomBnd(c_256);

//cout << simplicity_test(n1, 2)<<endl;

for(;;){
    //ZZ n1;

    while ((simplicity_test(n1,4)) == 0)
    { //while(n1<c_256)

        //n1 = RandomBnd(c_260);

        n1 = random(c_260, c_256);

    }

    std::cout << "n1\n";

    //ZZ n2;

    while ((simplicity_test(n2, 4)) == 0)
    {

        //while (n2 < c_256)

        n2 = random(c_260, c_256);

    }

    std::cout << "n2\n";

    //ZZ n3;


    while ((simplicity_test(n3, 4)) == 0)
    {

        //while (n3 < c_256)

        n3 = random(c_260, c_256);

    }

    std::cout << "n3\n";

```

```

//ZZ n4;

while ((simplicity_test(n4, 4)) == 0)
{
    //while (n4 < c_256)
    n4 = random(c_260, c_256);
}

std::cout << "n4\n";


if (n1 * n2 <= n3 * n4)break;
}

std::cout << "n1:" << n1 << endl;
std::cout << "n2:" << n2 << endl;
std::cout << "n3:" << n3 << endl;
std::cout << "n4:" << n4 << endl;


ZZ p, q, p1, q1;

if (n1 * n2 <= n3 * n4) {
    std::cout << "OK\n\n";

    //cout << n1 * n2 << endl<< n3* n4 << endl;

    p = n1; q = n2; p1 = n3; q1 = n4;
}

ZZ e,n,d,fi;

std::cout << "Generate Key Pair A:" << endl;

GenerateKeyPair(p, q, e, n, d,fi);

std::cout << "e: " << e << endl;

std::cout << "n: " << n << endl;

std::cout << "d: " << d << endl<<endl;

ZZ e1, d1, fi1;

std::cout << "Generate Key Pair B:" << endl;

GenerateKeyPair(p1, q1, e1, n1, d1,fi1);

std::cout << "e:" << e1 << endl;

std::cout << "n1:" << n1 << endl;

std::cout << "d1:" << d1 << endl<<endl;

```


ZZ M,C;

```
std::cout << "VT:" << endl;
```

```
std::cin >> M;
```

```
std::cout<<"Шифруєм"<<endl;
```

```
Encrypt(C, M, e, n);
```

```
std::cout << "C: " << C << endl;
```

```
Decrypt(M, C, d, n);
```

```
std::cout<<"Расшифровываем"<<endl;
```

```
std::cout << "M: " << M << endl<<endl;
```

```
//cout << "5)-----" << endl;
```

ZZ k1,S1,k,S;

```
/*  
Sign(S, M, d, n);  
if (M == Verify(M, S, e, n)) {cout << "digital true\n"; cout << Verify(M, S, e, n) << endl;}  
else { cout << "digital false\n"; cout << Verify(M, S, e, n) << endl;}  
*/
```

```
std::cout << "vvedite k:";
```

```
std::cin >> k;
```

```
std::cout << "Send: k, n1, n, d, e" << endl;
```

```
Send(k1, S1, k, S, n1, n, d, e);
```

```
std::cout << "Receive:k1, S1, d1, n1, e, n" << endl;
```

```
Receive(k, S, k1, S1, d1, n1, e, n);
```

```
}
```

ВИСНОВОК

В ході практикума ми ознайомились з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомились з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи засекреченого зв'язку й електронного підпису.