



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №5

з дисципліни

«Криптографія»

**на тему: «Вивчення криптосистеми RSA та алгоритму електронного підпису;
ознайомлення з методами генерації параметрів для асиметричних криптосистем»**

Виконали:

студенти 3 курсу ФТІ

групи ФБ-72

Макоїд Ігор, Оліферук Артур

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

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 зашкереженого 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 p, q і p_1, q_1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \nmid p_1q_1$; $p \nmid q$ – прості числа для побудови ключів абонента А, p_1, q_1 – абонента В.
3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p, q) та відкритий ключ (n, e) . За допомогою цієї функції побудувати схеми RSA для абонентів А і В – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , (e_1, n_1) та секретні d і d_1 .
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і В. 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 А и В, перевірити правильність розшифрування. Скласти для А і В 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.

Результати роботи:

Вибрані числа:

$p=7896453798458945128465128465128645132846512384651328465123846513264153264157423$

$q=1238701616274046826120764261460210120412010318074132037138214268370426170912671604163720916239193$

$p_1=983985712398023709231795784598712357812530091235721357123385489802357981325676787389035792315789213579823519786321$

$q_1=87435398093851987938251097230157071057982179347159893215789437169268571985679123795632487500234591867571405198657912985601769167$

Відкриті ключі:

n=978135008298443174136200924429204848219357514833498461520184156270967569815
30668736025332847384659725478096638840833587167894823133676570841958696892401
043682499335927873473074479639

e=65537

n1=86035182482183752641509072981868478602842711672700988392278848289114618025
17199408216528002036086204414010764469285111056776518576420890196478762688237
60119939329622875558770629162322108209096029736410891849493920283723758984987
87007406164607

e1=65537

Секретні ключі:

d=156711744314412520079193580824673862189347298560381675175274022931247379084
49761535136685971749808817892135869720080742489124174539952648483302968980323
83966290583314783525013943073

d1=39199391930024365684658451244924130965422331973493622127858254267253040179
31299484708569604052634909498487288509583782200232519030742371885652803898415
45118352378206999534078796125030219773449594861835979919238368584579464368422
57118280947073

Повідомлення:

k=19

Сигнатура:

S=101678981058811329075630621315958213250370652555528039481452055026798504796
52026654968669305112830313565838652825337957153558188048163111790996441129091
612939543059324893169799537146

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Receive key

Clear

Key8843D32748A35322D14D966B6F492D38412F42DC16FEC53DE95044A7B3F73F94D77D63B17C27DDFA83DF/

Signature6D0E387065C60EA1CB0BA55369B0794D95D75AE2D32CFBE01BDA818D12B3FEA053060045CC0B7B059DCI

Modulus1792804742EA8D87E8A3B6C963B4D4770444F1A5FBCDED25D01DE7C661F49027124CD78777C110B24D498

Public exponent10001

Receive

Key19

Verificationtrue

Код:

```
#include <fstream>
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
#include <math.h>
#include "biginteger.h"
```

```

using cus_int = wide_integer::generic_template::uint8192_t;
using namespace std;
cus_int el1[5],el2[5],sm1[2],sm2[2];
cus_int cus_pow(cus_int x,cus_int y)
{
    cus_int temp("1");
    for(cus_int i=0;i<y;i++)
    {
        temp*=x;
    }
    return temp;
}
cus_int getMes()
{
    srand(static_cast<unsigned int>(time(0)));
    cus_int randx = rand() % (50 - 4 + 1) + 4;
    cout<<"x = "<<randx<<endl;
    return randx;
}
string bin(cus_int a)
{
    string bin;
    while (1)
    {
        if(a%2==0)
        {
            bin = "0"+bin;
        }
        if(a%2==1)
        {
            bin = "1"+bin;
        }
        if(a==1)
            break;
        a/=2;
    }
    return bin;
}
cus_int nsd(cus_int a, cus_int b) {
    cus_int t;
    while (b != 0) {
        t = b;
        b = a % b;
        a = t;
    }
    return a;
}
cus_int converse(cus_int X, cus_int m)
{
    cus_int u, p,k=0,i=1, d;
    d = nsd(X, m);
    while(m%X!=0){
        u = m%X;
        p = m/X;
        cus_int pere_i = i;

```

```

        i = (-p)*i+k;
        k = pere_i;
        m = X;
        X = u;
    }
    return i;
}

cus_int gorn (cus_int x, cus_int a, cus_int m)
{
    string bi=bin(a);
    cus_int y=1;
    cus_int step[bi.length()];
    for(int i =0;i<bi.length();i++)
    {
        if((int)bi[i]==49)
            step[i]=1;
        else
            step[i]=0;
    }
    for(int i=0;i<bi.length();i++)
    {
        y=pow(y,2)%m;
        y=(y*pow(x,step[i]))%m;
    }
    return y;
}

cus_int test(cus_int p,cus_int k)
{
    cus_int temp;
    cus_int x= rand()%p+1;
    cus_int d,s=0,Xr,c=0;
    temp=p-1;
    while(1)
    {
        d=temp;
        if(temp%2!=0)
            break;
        temp=temp/2;
        s++;
    }
    while(c<k)
    {
        if(nsd(x,p)>1)
            return 0;
        if(gorn(x,d,p)==1 )
            return 1;
        for(cus_int r=0;r<s;r++)
        {
            temp = cus_pow(2,r);
            Xr=gorn(x,temp*d,p);
            if(Xr==p-1)
                return 1;
            if(Xr==1)
                return 0;
        }
    }
}

```

```

        c++;
    }
}

cus_int getRandomNumber(cus_int n, cus_int n1, cus_int k)
{
    srand(static_cast<unsigned int>(time(0)));
    cus_int randx = rand() % (n1 - n + 1) + n;
    if(randx%2==0 and randx!=n1)
        randx+=1;
    cus_int i = 0;
    while(i<=(n1-randx)/2){
        cus_int simple = randx+2*i;
        if(simple%3==0 or simple%5==0 or simple%7==0 or simple%11==0 or simple%13==0 or simple%17==0){
            i++;
            continue;
        }
        test(simple,k);
        if(test(simple,k)==1)
            return simple;
    }
    i++;
}

return 0;
}

cus_int gener(cus_int p, cus_int q, cus_int el1[])
{
    cus_int n,Fn,d;
    n=p*q;
    Fn=(p-1)*(q-1);
    cus_int e=65537;
    d=converse(e,Fn);
    el1[0]=p;el1[1]=q;el1[2]=n;el1[3]=e;el1[4]=d;
}

cus_int encrypt(cus_int m, cus_int n, cus_int e)
{
    return gorn(m,e,n);
}

cus_int decrypt(cus_int c, cus_int n, cus_int d)
{
    return gorn(c,d,n);
}

cus_int sign(cus_int m, cus_int d, cus_int n)
{
    return gorn(m, d, n);
}

cus_int verify(cus_int s, cus_int e, cus_int n)
{
    return gorn(s, e, n);
}

void send(cus_int k, cus_int e1, cus_int d, cus_int n, cus_int n1)
{
    cus_int S=sign(k,d,n);
    cout<<"signature = "<<S<<endl;
    cus_int S1=encrypt(S,n1,e1);
    cus_int k1=encrypt(k,n1,e1);
    sm1[0]=S1;
}

```

```

        sm1[1]=k1;
    }
void receive(cus_int d1,cus_int k1, cus_int S1,cus_int n1,cus_int e,cus_int n)
{
    cus_int k=decrypt(k1,n1,d1);
    cus_int S=decrypt(S1,n1,d1);
    cus_int K=verify(S,e,n);
    cout<<"K "<<K<<endl<<"Message "<<k<<endl;
}
int main()
{
    cus_int p,k,l;
    cus_int r,q;
    cus_int p1,q1,n1,e1,d1,p2,q2,n2,e2,d2;
    cin>>p>>k>>l;
    r=getRandomNumber(p,k,l);
    cout<<"          "<<endl;
    cin>>p>>k>>l;
    q=getRandomNumber(p,k,l);
    gener(r,q,e1);
    p1=e1[0];q1=e1[1];n1=e1[2];e1=e1[3];d1=e1[4];
    cout<<"p "<<p1<<endl;
    cout<<"q "<<q1<<endl;
    cout<<"n "<<n1<<endl;
    cout<<"e "<<e1<<endl;
    cout<<"d "<<d1<<endl;
    cout<<"          "<<endl;
    cin>>p>>k>>l;
    r=getRandomNumber(p,k,l);
    cout<<"          "<<endl;
    cin>>p>>k>>l;
    q=getRandomNumber(p,k,l);
    gener(r,q,e2);
    p2=e2[0];q2=e2[1];n2=e2[2];e2=e2[3];d2=e2[4];
    cout<<"p "<<p2<<endl;
    cout<<"q "<<q2<<endl;
    cout<<"n "<<n2<<endl;
    cout<<"e "<<e2<<endl;
    cout<<"d "<<d2<<endl;
    cus_int message = getMes();
    send(message,e2,d1,n1,n2);
    cus_int k2=sm1[1],S2=sm1[0];
    receive(d2,k2,S2,n2,e1,n1);
}

```

Висновок: у ході комп'ютерного практикуму було набуто навичок роботи з числами великої розрядності, написання тестів перевірки чисел на простоту та методів генерації ключів для асиметричної криптосистеми RSA. Набуто навичок побудови цифрового підпису на основі криптосистеми RSA.