

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Фізико-технічний інститут

**Лабораторна робота № 2**

З курсу

***Асиметричні криптографічні системи та протоколи***

«Вивчення криптосистеми RSA та алгоритму електронного підпису.

Ознайомлення з методами генерації параметрів для асиметричних криптосистем»

***Перевірив: Виконала:***

Яковлєв Сергій Володимирович студентка 4 курсу

ФТІ, групи ФІ-12

Молоченко Дар'я Романівна

**КИЇВ – 2014**

**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.

**Постановка задачі**

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 треба використовувати один з генераторів практикуму №1, що показав гарні статистичні властивості. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями.

2. За допомогою цієї функції згенерувати дві пари простих чисел і p, q і p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq < p1 q1; *p* і *q* – прості числа для побудови ключів абонента *А*, і – абонента *B*.

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 *B* – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n), (e1, n1), та секретні d і d1.

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів *А* і *B*. За допомогою датчика випадкових чисел вибрати відкрите повідомлення *M* і знайти криптограму для абонентів *А* і *B*, перевірити правильність розшифрування. Скласти для *А* 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. Перевірити роботу програм для випадково обраного ключа 0<k<n.

**Параметри криптосистеми RSA для абонентів А і B**

|  |  |  |
| --- | --- | --- |
|  | **A** | **B** |
| **p** | 1be6c78938eb67ccd16d3bae8d2d0d07a69654bd4ab91154bd0b40000926695bf | 1397336b3ab7dff98e9146452b8368c185cb209596192834c456e40049334a8e5 |
| **q** | 1e90d9c38f999764d3fd0000014ee4aca6d56afae6a00493300000000000001e7 | 1797cc425802e19bc0000000014ee4aca6d56afae000049334a8ba4ce00000021 |
| **n** | 354d4ee416f903d4ece785bf9c56ddf7240fe6a08645ec0850e15ede4fbf3bdcb91c316b065f936c60550bb87714d9ce2e76f3e41e222cf47066c01168126de59 | 1ce336f87370e23445932794d703ddf91b83aafa1d2178583e294fedd34bec66bbfefe142bb14232f2733de8fd1d50e1b39d6af47804d77da0a1fdcdcf9c9c585 |
| **fi(n)** | 354d4ee416f903d4ece785bf9c56ddf7240fe6a08645ec0850e15ede4fbf3bdc7ea4901e3dda943abaead009e898e819e10b342becc9170c835b80115eec046b4 | 1ce336f87370e23445932794d703ddf91b83aafa1d2178583e294fedd34bec6690cffe6698f6809da3e1f7a3d04b037386fcdf6401ebaab5a7a25f80a66951c80 |
| **e** | 10001 | 10001 |
| **d** | 25f78d4c5738c50bf43bee3f504368145471a71b8cc95d9986d321da44ffb8ca14c36f8cbb0e692b745d3216fcc17e02811493f2fdd5373fc88972adece93d349 | d19d6e252a88d4e1b87a359d331c3c7a7fcccdbe5be3a2ffaf10d78d46d086773a180c4d2f6685a15eaf33e6dc6b767fb7e4f69b19d49ac5cbc3d07c170de501 |

**Чисельні значення ВТ, ШТ, цифрового підпису для А і B**

Відкритий текст : M= cccdbe5bec3d07c17

Шифр текст і цифровий підпис абонента А :

CА= 12ff0fc09d60f3725580e67c64dc71df75b1e7920b4b90cd92fd65cabfd1889525b877  
7c0d293f4d8178399957c07e04c65df39185f980c97c798b2ef2ca45415

SА= 2952b75db762f97275d86ba029b2591598642047be274c3fb14028fd193de651a69eb  
0b3da0f92461095d53a01fbc95af7bbd2ba29999eaefee984bba75286e8e

Шифр текст і цифровий підпис абонента В :

Cв= 19c39f9e45f0ef6c579755b33c7b2d34e52686fbe4719d867cd9891cfc1138e9eb1f064  
e5de8140dd4dbee6d31b007f5d629b2dae32c5a993ea696a04d5b7e556

Sв= 13d3c97c749ee0453a7f6735a3d2933830df361db3f014f42df0161c99f9514df226069  
4715aefb062753b7fdc62ac0d995117d85c71a98394be6afbac5d12193

**Протокол конфіденційного розсилання ключів з підтвердженням**

**справжності**

*Абонент А має намір передати ключ K абоненту В:*

**K**= 4427467c611cd0ed7675f0ce29713

*А формує повідомлення (K1,S1) і відправляє його B:*

**K1**= 33c7365c0dcd8c291cdd2717aef7101d4c7f0dba34332ce8c65469623612d476382  
19475413fadd93d333ba951c32a7526587698ebc0ee147e647ab9f178f56e

**S**= c0486afae4fbca12cca0ac4ed96c8cffcf4052da87ca8d779538afc98a868194d33a706  
ca6f799bf633b393943ff7509a5486aa10abae7d0007e5ccb583dd005

**S1**= 1c41120cc7969c6d6fb53d22c6490920f35ab17fc7284998af31d12cf245df557f0fc4  
3c8588bf9307835d095594d81cb85be59c4c478c4af14fab9e3d17dbc92

*Абонент B за допомогою свого секретного ключа знаходить K i S:*

**K’**= 4427467c611cd0ed7675f0ce29713

**S’**= c0486afae4fbca12cca0ac4ed96c8cffcf4052da87ca8d779538afc98a868194d33a706  
ca6f799bf633b393943ff7509a5486aa10abae7d0007e5ccb583dd005

*Абонент В перевіряє підпис:*

**K**= 4427467c611cd0ed7675f0ce29713

**Код програми**

**Файл «ascrypt2.h»**

#include <cstdlib>

#include <iostream>

#include <fstream>

#include <string>

#include "gmp.h"

#include "gmpxx.h"

using namespace std;

void Rand\_x\_modp (mpz\_t x, mpz\_t p, mpz\_t n, unsigned long c, int m2exp); //rand() 0<x<p

int ZeroStep(mpz\_t d, mpz\_t p);

bool Miller\_Rabin(mpz\_t p, int k);

bool Step\_2\_point\_2(mpz\_t p, mpz\_t x, mpz\_t d, int s);

void Horner(mpz\_t result, mpz\_t x, mpz\_t exp, mpz\_t mod);

void Rand\_x (mpz\_t x, mpz\_t n, unsigned long c, int m2exp, int min); //rand() from interval [2^min;2^(min+1)]

void decrypt(mpz\_t M, mpz\_t C, mpz\_t d, mpz\_t n);

void encrypt(mpz\_t C, mpz\_t M, mpz\_t e, mpz\_t n);

void signat(mpz\_t S, mpz\_t M, mpz\_t d, mpz\_t n);

void RSA(mpz\_t e, mpz\_t d, mpz\_t n, mpz\_t phi, mpz\_t e1, mpz\_t d1, mpz\_t n1, mpz\_t phi1);

**Файл «func.cpp»**

#include "ascrypt2.h"

void Rand\_x\_modp (mpz\_t x, mpz\_t p, mpz\_t n, unsigned long c, int m2exp)

{

mpz\_t temp; mpz\_init(temp);

gmp\_randstate\_t state;

gmp\_randinit\_lc\_2exp(state, n, c, m2exp);

mpz\_urandomb(n,state,257);

mpz\_add\_ui(temp, x, 2);

mpz\_mod(x, temp,p);

}

int ZeroStep(mpz\_t d, mpz\_t p)

{

mpz\_t temp, p\_1;

mpz\_init\_set(p\_1,p);

int s=0;

mpz\_init(temp);

while (mpz\_divisible\_ui\_p(p\_1,2)!=0)

{

mpz\_cdiv\_q\_ui(temp,p\_1,2);

mpz\_set(p\_1,temp);

s++;

}

mpz\_set(d,p\_1);

return s;

}

bool Step\_2\_point\_2(mpz\_t p, mpz\_t x, mpz\_t d, int s)

{

mpz\_t temp, exp, temp1, temp2;

mpz\_init(temp); mpz\_init(exp); mpz\_init(temp1); mpz\_init(temp2);

mpz\_sub\_ui(temp2, p, 1);

int r=1;

Horner(temp,x,d,p);

if (mpz\_cmp\_ui(temp,1)==0) {return 1;}

else if (mpz\_cmp(temp,temp2)==0) {return 1;}

else

{

while(r!=s)

{

if (r==1)

{

mpz\_mul\_2exp(exp,d,r);

Horner(temp,x,exp,p);

mpz\_set(temp1,temp);

}

else

{

Horner(temp,temp1,exp,p);

mpz\_set(temp1,temp);

}

if (mpz\_cmp\_ui(temp,1)==0) {return 0;}

else if (mpz\_cmp(temp,temp2)==0) {return 1;}

else r++;

} }

return 0;

}

bool Miller\_Rabin(mpz\_t p, int k)

{

int i=0, m2exp=0, s=0, check=0;

unsigned long c = 0;

mpz\_t temp, p\_1, d, x, n, temp1; mpz\_init(temp1); mpz\_init(temp); mpz\_init(p\_1); mpz\_init(d); mpz\_init(x); mpz\_init\_set\_str (n, "34", 0);

mpz\_sub\_ui(p\_1,p,1);

s=ZeroStep(d,p\_1);

while (i<k)

{

m2exp=rand()%100+2;

c = rand()%10000+1;

Rand\_x\_modp (x, p, n, c, m2exp);

mpz\_gcd(temp1,x,p);

if (mpz\_cmp\_ui(temp1,1)==0)

{

bool t = Step\_2\_point\_2(p,x,d,s);

if (t==false) return false;

else i++;

}

else {return false;}

}

if (i==k) return true;

else return false;

}

void Horner(mpz\_t result, mpz\_t x, mpz\_t exp, mpz\_t mod)

{

mpz\_t temp, temp1;

mpz\_init\_set\_str(temp, "0", 0);

mpz\_init\_set\_str(temp1, "0", 0);

int size=(mpz\_sizeinbase (exp, 2)-1);

mpz\_set\_str(result, "1", 0);

for (int i=size; i>=0; i--)

{

mpz\_powm\_ui(temp,result,2,mod);

if (mpz\_tstbit(exp, i)==1)

{

mpz\_mul(temp1, temp, x);

mpz\_mod(result, temp1, mod);

}

else mpz\_set(result,temp);

}

}

void Rand\_x (mpz\_t x, mpz\_t n, unsigned long c, int m2exp, int min)

{

int k=0;

mpz\_t temp; mpz\_init(temp);

mpz\_t two; mpz\_init\_set\_str(two,"2",0);

mpz\_t r\_min; mpz\_init(r\_min);

mpz\_pow\_ui(r\_min, two, min);

gmp\_randstate\_t state;

gmp\_randinit\_lc\_2exp(state, n, c, m2exp);

mpz\_urandomb(temp,state,min);

mpz\_add(x, temp, r\_min);

if (mpz\_divisible\_ui\_p(x,2)==1) {mpz\_add\_ui(temp, x, 1); mpz\_init\_set(x,temp);}

k=5+rand()%10;

bool h=Miller\_Rabin(x,k);

while (h!=1) {mpz\_add\_ui(temp,x,2); mpz\_set(x,temp); h=Miller\_Rabin(x,k);}

}

void RSA(mpz\_t e, mpz\_t d, mpz\_t n, mpz\_t phi, mpz\_t e1, mpz\_t d1, mpz\_t n1, mpz\_t phi1)

{

FILE\* rsa;

string s;

rsa = fopen("RSA.txt", "w");

mpz\_t p, q, p1, q1, temp1, temp2; mpz\_init(p); mpz\_init(q); mpz\_init(p1); mpz\_init(q1); mpz\_init(temp1); mpz\_init(temp2);

unsigned long c=0;

int mp2exp=0;

mpz\_set\_str(e,"65537",0);

mpz\_set\_str(e1,"65537",0);

mpz\_set\_str(n,"345678908765432345678654",0);

cout<<"Input randomize constants\n C and mp2exp\n";

cin>>c;

cin>>mp2exp;

Rand\_x(p,n,c,mp2exp,256);

Rand\_x(q,n,c+123,mp2exp+105,256);

Rand\_x(p1,n,c-5,mp2exp+5,256);

Rand\_x(q1,n,c-8,mp2exp+89,256);

mpz\_sub\_ui(temp1,p,1);

mpz\_sub\_ui(temp2,q,1);

mpz\_mul(phi,temp1, temp2);

mpz\_gcdext(n,temp1, temp2, e, phi);

mpz\_mod(temp2,temp1,phi);

mpz\_set(d,temp2);

mpz\_mul(n,p,q);

mpz\_sub\_ui(temp1,p1,1);

mpz\_sub\_ui(temp2,q1,1);

mpz\_mul(phi1,temp1, temp2);

mpz\_gcdext(n1,temp1, temp2, e1, phi1);

mpz\_mod(temp2,temp1,phi1);

mpz\_set(d1,temp2);

mpz\_mul(n1,p1,q1);

mpz\_t C,M;

mpz\_init(C); mpz\_init(M);

fputs("A:\n public key\n e= ", rsa);

mpz\_out\_str(rsa, 16,e);

fputs("\n n= ", rsa);

mpz\_out\_str(rsa,16,n);

fputs("\n private key\n d= ", rsa);

mpz\_out\_str(rsa,16,d);

fputs("\n phi(n)= ", rsa);

mpz\_out\_str(rsa,16,phi);

fputs("\np= ", rsa);

mpz\_out\_str(rsa,16,p);

fputs("\nq= ", rsa);

mpz\_out\_str(rsa,16,q);

fputs("\n decrypt/encrypt for A", rsa);

Rand\_x\_modp(M,n,temp1,c,mp2exp);

fputs("\n M= ", rsa);

mpz\_out\_str(rsa,16,M);

encrypt(C,M,e,n);

fputs("\n C= ", rsa);

mpz\_out\_str(rsa,16,C);

decrypt(M,C,d,n);

fputs("\n M= ", rsa);

mpz\_out\_str(rsa,16,M);

fputs("\n B:\n public key\n e1= ", rsa);

mpz\_out\_str(rsa, 16,e1);

fputs("\n n1= ", rsa);

mpz\_out\_str(rsa,16,n1);

fputs("\n private key\n d1= ", rsa);

mpz\_out\_str(rsa,16,d1);

fputs("\n phi(n1)= ", rsa);

mpz\_out\_str(rsa,16,phi1);

fputs("\np1= ", rsa);

mpz\_out\_str(rsa,16,p1);

fputs("\nq1= ", rsa);

mpz\_out\_str(rsa,16,q1);

fputs("\nSignature. B is a sender", rsa);

Rand\_x\_modp(M,n,temp1,c,mp2exp);

fputs("\n M= ", rsa);

mpz\_out\_str(rsa,16,M);

signat(C,M,d1,n1);

fputs("\n S= ", rsa);

mpz\_out\_str(rsa,16,C);

encrypt(M,C,e1,n1);

fputs("\n M= ", rsa);

mpz\_out\_str(rsa,16,M);

}

void encrypt(mpz\_t C, mpz\_t M, mpz\_t e, mpz\_t n)

{ Horner(C,M,e,n); }

void decrypt(mpz\_t M, mpz\_t C, mpz\_t d, mpz\_t n)

{ Horner(M,C,d,n); }

void signat(mpz\_t S, mpz\_t M, mpz\_t d, mpz\_t n)

{ Horner(S,M,d,n); }

**Файл «main.cpp»**

#include "ascrypt2.h"

int main(int argc, char \*argv[])

{

mpz\_t e, n, d, phi, e1, n1, d1, phi1, k,k1,S,S1,temp;

mpz\_init\_set\_str (temp, "157863", 0); //for Random func

mpz\_init\_set\_str(e1, "10001", 16); mpz\_init\_set\_str(e, "10001", 16);

mpz\_init(n); mpz\_init(d); mpz\_init(phi);

mpz\_init(n1); mpz\_init(d1); mpz\_init(phi1);

mpz\_init(k); mpz\_init(k1);

mpz\_init(S); mpz\_init(S1);

RSA(e,d,n,phi,e1,d1,n1,phi1);

FILE\* key;

key=fopen("KEY\_EXCHANGE.txt", "w");

fputs("A:\n public key\n e= ", key);

mpz\_out\_str(key, 16,e);

fputs("\n n= ", key);

mpz\_out\_str(key,16,n);

fputs("\n private key\n d= ", key);

mpz\_out\_str(key,16,d);

fputs("\n B:\n public key\n e1= ", key);

mpz\_out\_str(key, 16,e1);

fputs("\n n1= ", key);

mpz\_out\_str(key,16,n1);

fputs("\n private key\n d1= ", key);

mpz\_out\_str(key,16,d1);

Rand\_x\_modp(k,n,temp,2345,78);

fputs("\n\n k= ", key);

mpz\_out\_str(key,16,k);

Horner(k1,k,e1,n1);

fputs("\n k1= ", key);

mpz\_out\_str(key,16,k1);

Horner(S,k,d,n);

fputs("\n S= ", key);

mpz\_out\_str(key,16,S);

Horner(S1,S,e1,n1);

fputs("\n S1= ", key);

mpz\_out\_str(key,16,S1);

Horner(k,k1,d1,n1);

fputs("\n k'= ", key);

mpz\_out\_str(key,16,k);

Horner(S,S1,d1,n1);

fputs("\n S' = ", key);

mpz\_out\_str(key,16,S);

Horner(k,S,e,n);

fputs("\n Check signature of A:\n S^emodn= ", key);

mpz\_out\_str(key,16,k);

system("PAUSE");

return EXIT\_SUCCESS;

}

**Висновок**

В ході виконання лабораторної роботи ми ознайомились з тестами перевірки чисел на простоту, а також методами генерації ключів для асиметричної криптосистеми типу RSA. Ознайомились з системою захисту інформації на основі криптосхеми RSA. Організували з використанням цієї системи засекречений зв'язок й систему електронного підпису. Розібрались та вивчили протокол розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA.