Міністерство освіти і науки України Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

Лабораторна робота №5

З предмету «Криптографія»

Виконали:

Студенти 3 курсу,

ФТІ, групи ФБ-72

Солдатова Катерина, Яшкова Вікторія

Київ 2019

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 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб p1q1≤pq ; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – 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 A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , (,) 1 n1 e та секретні d і d1 .

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 и 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 n.< k <0

Етапи розробки програмного коду:

Перш за все було розроблено функцію перевірки числа на простоту. Для цього був використаний тест Мілера-Рабіна.

Далі ми отримали пари простих чисел p1, q1, p, q та написали функції за шифрування, розшифрування, підписання ЦП, його перевірки, та функції відправки та отримання ключів. На цих етапах проблем не було.

Результати:

Абонент A

p: 10341009355767392870487011998918102816560737849329674826251046880882881107339 7

q: 73167021658681660926453177319852259258703707762300933705789034099144316593859

n:
75662085550606252338303137130166248127418737540253103926833075650830601916773323312873643524774467662762142043599453728421843175792434598054402766884690 23

fn:
75662085550606252338303137130166248127418737540253103926833075650830601916771557541721479968878154429789051710725210617559287198972751603025323035608017 68

e:
12037319518568821445852606203141566109840443672311917605993691865987536958307718445169512598636561844139850479524398097952865337839596268295383635965257 7

d:
40220050032013345814535283948644715435250850989531515700275342299351613617048822096499674974946447831601502577924368387571720784147927596967841310855882 49

C:
20205849103684424864458213012804336471361799435357951272667522395378104819569456913358170550688458730598662371779756903729309110107169831286753601498460 91

M:

18790943946673214821216224343838117599696441702906036456886672139261840175268656165015480383032
37841683196849053943005472439332923069915783592350669459467

S:
56329846487811486164703645739955060802697049146857969108404482565026952957959514970279384750697
82537220850139621539638324556399458780233434175626223657137

K:
66615351003145489247713946118926960556289859488093157326354545011668723248159073382558529330537
51104260460865879368981601858480521620906267655483705102958

Абонент Б

p: 10119060205524373796459506561516714086365154026691580875204350511731297917540 1

q: 77528778831008641771982708040628564569197796793044371388083189959024831030363

n:
78451838065176002705645633145606336324266196182050481796948002698570706764359006100809824344525
24331312883023809733225933832888168222407461437368933700563

fn:
78451838065176002705645633145606336324266196182050481796948002698570706764357218907000961820727
87753539227228104300376596772927988082280766361031123494800

e:
14083238867540448581184469242983463711351104679782710873442570011791756139971565025194921297577
45371359470695012848310559301908330169225967545462850208263

d:
23428565892867497621304173142102445722872035120824158023470780874643386366996360936787408962283
67932368105379573191948730959105619777167921169550189005527

C:
19049644202139721562993882249155379799911101465315063451461702542148957841680596261498980336755
64429156506231041619609215068016188433005070646729561749615

M:
71583878139592777927481763532353374133978254115917966441496093228367458372828129103649150364608
35842032728215117818459748405304846083058500661929323266766

S:

36312912284799679581106716247809083480106939779628108552330636809882193721934008968118517994387
78063153868054572351137064602183011700140187576630561878834

K:
57403899604234695043321769770923135243101898408741932488076045855882540407177294710820124612110
45365955315746531180818320212559426514486472086597402021011

Відправка ключа
A:

к1 = 15403479317446713232930234899060839475880285567089824953078197436006291304009

s1 = 51847077957643887416334547237039794446263914992774670656952368961647944133505

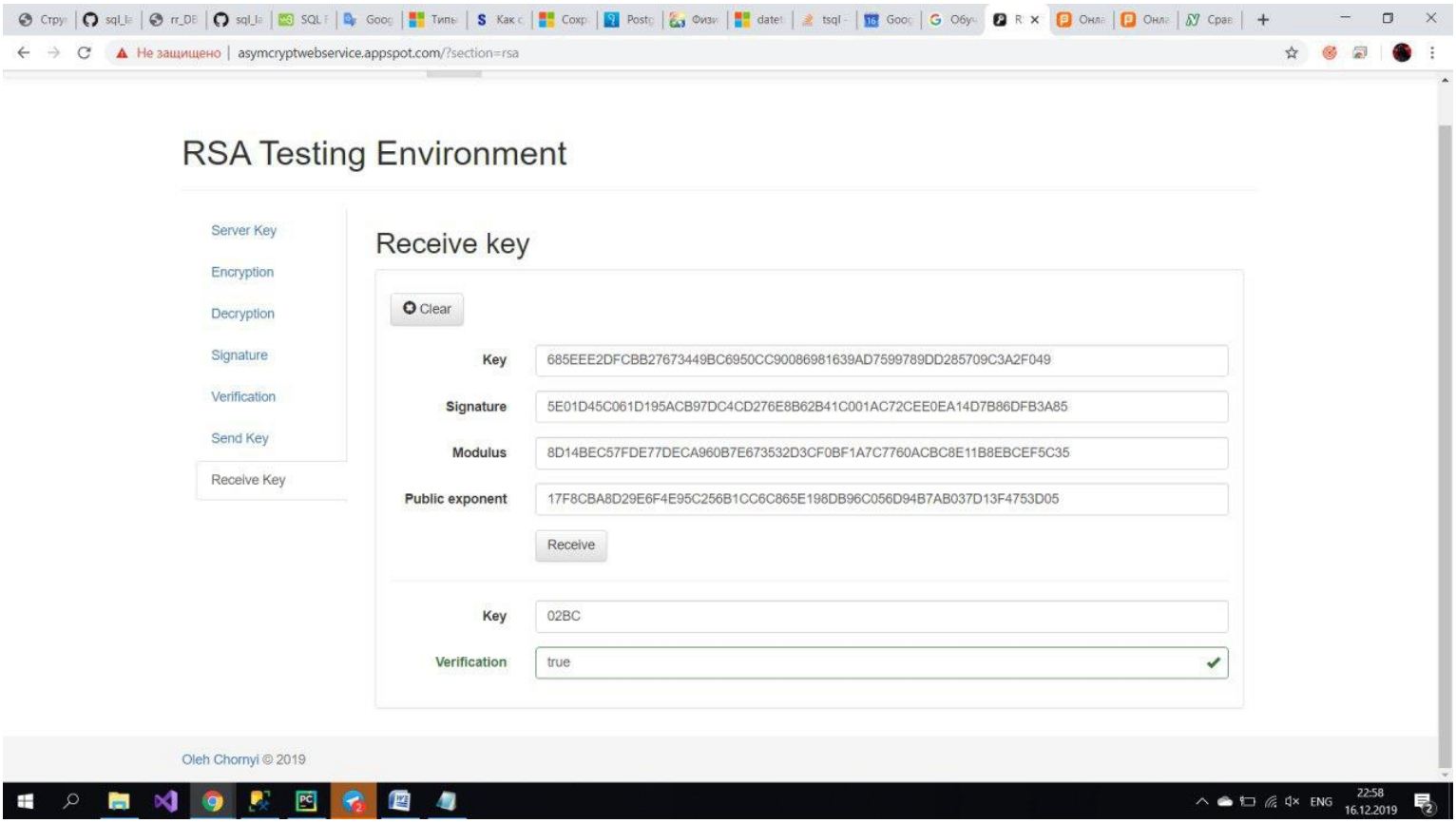
B:

k1:
78610176756665993723705546618015615013896239147061410072716834813549774521618091517303092715368

724696072860006204862115865664702367883435414204755150407 00

s1:
829451176232031993549173884056822587042513427618604014549589149271696723839884824599358576822491488797560193808556054300253088336741975335270005953661601 5

Результати з сайта:



Програмний код:

```
import math
from collections import Counter
from collections import OrderedDict
from collections import ChainMap
import re
import random

length=0

def CreateS(M,d,n):
    s=pow(M,d,n)
    #print("Ms: ",M,"\nS: ",s)
    return s


def SendKey(m,e1,n1,d,n):
    k1=pow(m,e1,n1)
    s=CreateS(m,d,n)
    print("S: ", s)
    s1=pow(s,e1,n1)

    return k1,s1

def CheckerForS(m, s, e, n):
    Check=pow(s,e,n)
    if m==Check:
        return 1
    else: return 0

def ReceiveKey(k1,s1,d1,n1,e,n):
    k=pow(k1,d1,n1)
    s=pow(s1,d1,n1)

    if CheckerForS(k, s, e, n):
        return k
    else : return 0

def KeyGenerator(length):
```

```
    temp=''
    temp+='1'
    for i in range(length - 2):
        temp+=str(random.randint(0,1))

    temp+='1'
    temp=int(temp,2)
    return temp

def CheckerPrimeNum(key):
    if key%2==0:
        #print("error"," 2")
        return 0
    elif key%3==0:
        #print("error"," 3")
        return 0
    elif key%5==0:
        #print("error"," 5")
        return 0
    elif key%7==0:
        #print("error","7")
        return 0
    elif key%11==0:
        #print("error","11")
        return 0
    else:
        #print("!!!!!!")
        return 1

    return 0


def CheckerPsevdoprime(num, x, p):

    for r in range(1, num[1]):
        xr=pow(x, (num[0] * (pow(2, r))), p)
        if xr==(-1)%p:
            return 1
        elif xr==1:
            return 0
        else: continue
    return 0



def Decompositor(p):
    pwithout1=p-1
    d=0
    s=0
    i=pwithout1
    while i%2==0:
        i=i//2
        s+=1
    d=pwithout1//(2**s)
    return d,s



def MillerTester(p):
    decomp=Decompositor(p)
    k=20
    for i in range(k):
        x=random.randint(2,p-1)
        evk=AlgorEvclid(x, p)[0]
        if evk>1:

            return 0
        else:

            if pow(x,decomp[0],p)==1 or pow(x,decomp[0],p)==(-1)%p:
                continue
            else:

                if CheckerPsevdoprime(decomp, x, p):
                    return 1
                else:
                    return 0
                    print("no","\n")

    return 1

def Prime(length):

    temp=KeyGenerator(length)
    test=MillerTester(temp)
```

```
    while test==0:
        if test ==0:
            print ("не пройшли перевірку: ",  temp)
        temp+=2
        test=MillerTester(temp)
    return temp

def AlgorEvclid(a, b):
    if b == 0:
        return a, 1, 0
    else:
        d, x, y = AlgorEvclid(b, a % b)
        return d, y, x - y * (a // b)


def E(fn):
    e=random.randint(2,fn-1)
    ev=AlgorEvclid(e, fn)
    while ev[0]!=1:
        e=random.randint(2,fn-1)
        ev=AlgorEvclid(e, fn)

    return e,ev[1]%fn

def RSA(length):
    p=Prime(length)
    q=Prime(length)
    n=p*q
    print("p: ",p,"\nq: ",q,"\nn: ",n)
    fn=(p-1)*(q-1)
    print("fn: ",fn)
    e,d=E(fn)
    d=AlgorEvclid(e,fn)[1]%fn
    print("e: ",e,"\nd: ",d)
    M=random.randint(0,n-1)

    C=pow(M,e,n)
    print("C: ",C)
    dM=pow(C,d,n)
    print("M: ",M)
    print("Decrypt: ",dM)
    return e,n,d,p,q


def Crypter(M,e,n):
    C=pow(M,e,n)
    return C



def Decrypter(C,d,n):
    dM=pow(C,d,n)
    print("Decrypt: ",dM)
    return dM




#A = RSA(256)
#B = RSA(256)


#print("A_e: ",A[0],"\nA_n: ",A[1],"\nA_d: ",A[2])
#print("B_e: ",B[0],"\nB_n: ",B[1],"\nB_d: ",B[2])
#k=random.randint(2,B[1]-1)

#k1,s1=SendKey(k,B[0],B[1],A[2],A[1])
#print("K: ",k)
#print("k1: ",k1,"\ns1: ",s1)

k1,s1=SendKey(k,B[0],B[1],A[2],A[1])
```

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.