

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-74
Пудім Єлизавета,
Горобець Ангеліна

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 $1 < p, q$ довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $p-1 \leq pq$; p і q – прості числа для побудови ключів абонента А, $1 < p < q-1$ – 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) , (d_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 і 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. Перевірити роботу програм для випадково обраного ключа $n < k < 2^n$

Результати:

Абонент А

$p = 81260755033274521310356298409470587420002056443125793164516156066888920807813$

$q = 109470439663434889365443423467274495778624041135279792342760812886051396768467$

$n =$

$88956505808752414806124410982595106493001868652277198115282093461103831463355697866032035594273$
 $36013562419840714344912630334068162714387106084663365632671$

$e =$

$36113831167004717508791375169533628954225910923202840952193687124201808506880234325925687177465$
 $91937262332949926303330700912245484225205561434978407353117$

$d =$

$56319635958804572633369692768927238992809780022175130544223474269268736668038941107861177226927$
 $1584346963121539467605935959716747436760301645193128046565$

Відкритий текст

$m =$

$84669000144389915709947018393780730516023192593867733801474135117414214929442131032294520800017$
 $98850825590292218230188152329122114327761464086529951329900$

Шифрований текст

$c =$

$15584764467384092057623179234536270899923044833077997476618859977936723260171613534807909306903$
 $43780758741575992408125183264796206131841809747538411720017$

Підпис...

$s =$

$15890217103962333437018150022891068911650910527755574061905585989204497768982671031328906206278$

0231968390186009169690536275418853407274932418848302871836

Абонент Б

p = 108522054899911941072063993274743563089709021591933769395868762172352049623761

q = 81128713671651298646760006994423119325693955431914772019741978616306754529527

n =
88042547190341786978492323208881100569915785335256669997819169308031893105368548688094707681356
24963731549067386919751292555400745676214964126962915291047

e =
76189370995635589797499718493686581851807510014392488398684423003201512160104872350844252368292
12597892860910367209737063057243998140040982411841009509467

d =
72992802015319506493080862211870924195885237357955366984289668186306035829063665246300757886862
07449003740529861908788599201048896432239983002184754690803

Відкритий текст

m =
81779047566774671453503320850917783281277036067722219193876388524118369938479732356259610508041
81829005817530547697261766428358529775944112268039191919096

Шифрований текст

c =
90787983738523930298492661277035496077576519329587708662840995907745351723356788614916438630647
6729925176861841242062187712325246816678374790694262500162

Підпис...

s =
80891197845920633319937957416195794845380329849889570450775728512671266691804659798155551361255
10129380331581013136460597251081192545974472029077004454412

Відправка ключа

$k_1 = 15403479317446713232930234899060839475880285567089824953078197436006291304009$

$s_1 = 51847077957643887416334547237039794446263914992774670656952368961647944133505$

Результати з сайта:

Вхідні (288) - elizav... Главная страница Lviv Data Science S... (95) YouTube 24 Ultimate Data Sc... (98) Machine Learn... RPA InteticsLab | Int...

Receive key

Clear

Key

5c1e5b8c2a6ad031020432b9ff52f4fcdcf116a35f869d02f412dc8950001dce6ab5a7fbd7c77d80ba6d0f7015938e27f

Signature

2163fc383643e36bd57d940c7fa08285f17194c832649b3b57f828b3b7bfff0ee54b062e4d36439a5bcd2376892d72

Modulus

84a5838b9c4c043588afe79ae3506e8ce499721decde5e12e9e87feca69b5be840867e46c10d756f9de3502d00431

Public exponent

10001

Receive

Key

63ECA8

Verification

true

RSA Testing Environment

Server Key

Encryption

Decryption

Signature

Verification

Send Key

Receive Key

Get server key

Clear

Key size

512

Get key

Modulus

80004DA1EB869CB642BB49266ACD89C0D9CFEDD265CFDF3503E7FD92761BF877076EBD58D7915E94EAD

Public exponent

10001

Oleh Chornyi © 2020

Активация Windows

Програмний код:

```
import random
import math

def power(x, y, p):
    res = 1;
    while (y > 0):
        x = x % p;
```

```

    if (y & 1):
        res = (res * x) % p;

    y = y >> 1;
    x = (x * x) % p;

return res;

def miillerTest(d, n):

    a = 2 + random.randint(1, n - 4);

    # a^d % n
    x = power(a, d, n);

    if (x == 1 or x == n - 1):
        return True;

    while (d != n - 1):
        x = (x * x) % n;
        d *= 2;

        if (x == 1):
            return False;
        if (x == n - 1):
            return True;

    return False;

def isPrime( n, k):

    if (n <= 1 or n == 4):
        return False;
    if (n <= 3):
        return True;

    d = n - 1;
    while (d % 2 == 0):
        d //= 2;

    for i in range(k):
        if (miillerTest(d, n) == False):
            return False;

    return True;

def generate_ned(bit):
    p=q=0
    while(not (isPrime(p,20))):
        p = random.randint(2**(bit-1), 2**bit) #75842885601676576300500163132150530865942493374981881120300052634072283878357

    while(not (isPrime(q,20))):
        q = random.randint(2**(bit-1), 2**bit) #63738451939031325961840482847016348013045118738110702785965932149633377821763
    print('p: ', p)
    print('q: ', q)
    n = p*q
    phi_n = (p-1)*(q-1)
    e=0
    while(math.gcd(e,phi_n )!=1):
        e = random.randint(2,phi_n-1)
    def ext_euc(a, b):
        u, uu, v, vv = 1, 0, 0, 1
        while b:
            q = a // b
            a, b = b, a % b
            u, uu = uu, u - uu*q
            v, vv = vv, v - vv*q
        return (u, v, a)

    def inverse(a, n):
        """
        a - число
        n - модуль
        """
        u, v, a = ext_euc(a, n)
        if a == 1:
            return (u%n)

```

```

        else:
            return False
    d = inverse(e, phi_n)
    return n, e, d

#M = random.randint(1,n-1)

def Encrypt(M,e,n):
    C = power(M, e, n)
    return C

def Decrypt(C,d,n):
    M1 = power(C, d, n)
    return M1

def Sign(M,d,n):
    S = power(M, d, n)
    return S

def Verify(M, S, e, n):
    return M == power(S, e, n)

#k = random.randint(1, n-1)
def Sendkey (n,n1,e1,d, k):
    S = power(k,d,n)
    S1 = power(S,e1,n1)
    k1 = power(k, e1, n1)
    return k1, S1

def Receivekey(k1, d1, n1, s1, e, n):
    k = power(k1,d1,n1)
    S = power(S1, d1, n1)
    if k == power(S,e,n):
        return k

n, e, d = generate_ned(256)
print(n, e, d)
M = random.randint(1,n-1)
print('Повідомлення',M)
C = Encrypt(M,e,n)
print('Шифртекст', C)
S = Sign(M,d,n)
print('Підпис' ,S)

ver = Verify(M, S, e, n)
print(ver)

key = 6548648
n1 =
int('80004DA1EB869CB642BB49266ACD89C0D9CFEDD265CFDF3503E7FD92761BF877076EBD58D7915E94EAD526AE3B62CF61C0FA911A5C1D2C783B37E059517B57AF',
16)
e1 = int('10001', 16)
k1,s1 = Sendkey (n,n1,e1,d, key)
print(hex(k1),hex(s1), hex(n))

```

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.

