Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Лабораторна робота із Криптографії №5

Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем

Виконали:

Студенти ФТІ, групи ФБ-74
Опанасюк О. та Панчук. О.

Перевірено:

_____

Київ 2019

КРИПТОГРАФІЯ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ No5

Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем.

Мета та основні завдання 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 і рекомендації щодо 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 довжини щонайменше 256 біт.
3. Написати функцію генерації ключових пар для RSA.
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B.

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:

p = 171697676811165117816141924070179419629

q = 319244783954675094753657278877920567147

p1 = 299784024989492940442647905756645568671

q1 = 212714641815858669792298191937186751773

e = 65537 (const), E = 2^16 + 1

Не підішли:

p = 288579977506719825311000696956362473797

q = 278931316550957101736719848923801414047

p1 = 19256643008675659132432123087756768623 7

q1 = 33270903444269943327347835687295231 5731

—-

p = 24555025731822367860773106828046199 5733

q = 25162200725058583482622172299806231 8323

p1 = 22065384901140744067233764266281478 5581

q1 = 23303423746439303268744856727028819 1523

Відкритий текст:

48219111078788597015034770286345555109514912923926030671826757023133787068025

Зашифрований текст:

61489738809790241104423245215779860161498027128899873251244461975643637894957

Цифровой підпис:

47964518666421470184831041104012962980581353323090448481172299507103778778061

Send Key:

K1 =

18958168732268735211291063685116085616452604267408554841651622399454046374578

S1 =

22520517279785500483159211563538710166398374303330273444100335914571484096296

Труднощі:

При збільшені довжин ключа - геометрично збільшується час роботи програми.

Код програми:

```
import  math, os, sys, ast, requests, json, funcs, random

from random import randint

#----Functions----


def site_Decrypt(e1, n1, headers_data):

    #ast.literal_eval(b)    from 16 to 10

    #hex(a)                  from 10 to 16

    n, e, d = GenerateKeyPair(128)

    k1, s1 = SendKey(e, n, n1, d, e1)

    print('\n\n----Results----\n')

    print('n= ' + hex(n))

    print('e= ' + hex(e))

    print('k= ' + hex(k1))

    print('s= ' + hex(s1))

    url = 'http://asymcryptwebservice.appspot.com/rsa/receiveKey?key=' + hex(k1)[2:] + '&Signature='
+ hex(s1)[2:] + '&modulus=' + hex(n)[2:] + '&publicExponent=' + hex(e)[2:]

    r = requests.get(url, headers=headers_data)

    print('\n\n----Responce----\n' + r.text)




def report(text, init):

    if init==1:

        f = open('report.txt', 'a')

        f.write(text + '\n')

        f.close


def power(b, a, m):
```

```
    w = str(bin(a))[2:]
    i = 0; y = 1
    for i in w:
        y = ((y ** 2) % m * (b ** int(i)) % m)
    return y


def rev(a, b, j):
    c, x = b, 1
    while b != 0:
        q = a // b
        j = a
        b = j % b
        j = x
    if j == 0:
        return a
    else:
        if x < 0:
            x = c + x
        if a == 1:
            return x
        return -a


def GenerateKeyPair(bit):
    tmp_p = '1'
    tmp_q = '1'
    for i in range(bit-2):
        tmp1 += str(randint(0, 1))
    for i in range(bit-2):
        tmp2 += str(randint(0, 1))
    p = -1 * int(tmp1, 2); q = -1 * int(tmp2, 2)
    while p < 0 and q < 0:
        if (p<0):
            p = -1 * p + 2
            p = prm(p)
        if (q<0):
```

```
            q = -1 * q + 2
            q = prm(q)
        if p > 0 and q > 0:
            break
    report('p = ' + str(p), 1)
    tmp = '1'
    i = 0
    report('q = ' + str(q), 1)
    n = p * q
    report('n = ' + str(n), 1)
    fi = (p - 1) * (q - 1)
    e = 65537
    report('e = ' + str(e), 1)
    d = rev(e, fi, 1)
    report('d = ' + str(d), 1)
    return n, e, d


def Encrypt_one():
    report('Init Encrypt\n', 1)
    n, e, d = GenerateKeyPair(128)
    m = randint(0, n - 1)
    report('m = ' + str(m), 1)
    c = power(m, e, n)
    report('c = ' + str(c) + '\n', 1)
    return d, n, e, m, c


def Encrypt(e, n):
    report('Encrypt\n', 1)
    m = randint(0, n-1)
    report('m = ' + str(m), 1)
    c = power(m, e, n)
    report('c = ' + str(c) + '\n', 1)
    return 0, n, e, m, c
```

```
def Decrypt(d, c, n):
    report('Decrypt', 1)
    m = power(c, d, n)
    report('m = ' + str(m), 1)
    return 1


def Sign(m, d, n):
    report('Подпись...', 1)
    s = power(m, d, n)
    report('s = ' + str(s), 1)
    return s


def Verify(s, e, n):
    report('Проверка...', 1)
    m = power(s, e, n)
    report('m = ' + str(m), 1)
    return m


def ReceiveKey(k1, d1, n1, s1, e, n, e1):
    report('Получаем ключ: ', 1)
    k = power(k1, d1, n1)
    report('k = ' + str(k), 1)
    s = power(s1, d1, n1)
    report('s = ' + str(s), 1)
    report('Проверка...', 1)
    k = power(s, e, n)
    report('k = ' + str(k), 1)
    return s, k


def SendKey(e1, n, n1, d, e):
    report('Отслыаем ключ: ', 1)
```

```
    k = randint(1, n-1)
    report('k = ' + str(k), 1)
    s = power(k, d, n)
    s1 = power(s, e1, n1)
    report('s1 = ' + str(s1), 1)
    k1 = power(k, e1, n1)
    return k1, s1


def prm(num):
    s = num - 1
    t = 0
    while s % 2 == 0:
       s = s // 2
       t += 1
    for trials in range(len(num)):
       a = random.randrange(2, num - 1)
       v = pow(a, s, num)
       if v != 1:
          i = 0
          while v != (num - 1):
             if i == t - 1:
                return num
             else:
                i = i + 1
                v = (v ** 2) % num
       return -num


def init():
    f = open('report.txt', 'w')
    f.write('')
    f.close
    mas1 = Encrypt_one()
    mas2 = Encrypt_one()
    if mas2[1] < mas1[1]:
        report('\nTry again\n---------\n\n\n', 1)
```

```
        return False
    try:
        s1 = Sign(mas1[3], mas1[0], mas1[1])
        s2 = Sign(mas2[3], mas2[0], mas2[1])
        for i in s1:
            print(s1)
        try:
            Verify(s1, mas1[2], mas1[1])
            Verify(s2, mas2[2], mas2[1])
        except:
            print('verify error')
        try:
            Decrypt(mas1[0], mas1[4], mas1[1])
            Decrypt(mas2[0], mas2[4], mas2[1])
        except:
            print('decrypt error')
        report('\n\n', 1)
        serverkey1 = SendKey(mas2[2], mas1[1], mas2[1], mas1[0], mas1[1])
        report('\n\n', 1)
        ReceiveKey(serverkey1[0], mas2[0], mas2[1], serverkey1[1], mas1[2], mas1[1], mas1[2])
    except Exception as err:
        print(str(err))


    os.system('open report.txt')
    sys.exit()
    return True


#---main----


try:
    mode = sys.argv[2]
    if (mode == '--site'):
        cookie = input('Enter Cookie for site: ') #JSESSIONID=S91RGJz6uNvvZBDgG91qPA
        headers_data = {'Cookie':cookie}
```

```
        r = requests.get('http://asymcryptwebservice.appspot.com/rsa/serverKey?keySize=256',
headers=headers_data)

        hs = r.text

        tmp1 = hs.find(':')

        tmp1 = hs.find('"', tmp1)

        tmp2 = hs.find('"', tmp1 + 1)

        n = hs[tmp1+1:tmp2]

        os.system('clear')

        print('----Input----\n' + hs)

        n = ast.literal_eval('0X' + n.lower())

        e = 10001

        site_Decrypt(e, n, headers_data)

    elif (mode == '--self'):

        while (init() != True):

            pass

    else:

        print('Enter mode.\n--self\n--site')

except Exception as err:

    print('Enter mode.\n--self\n--site')

    print(str(err))
```