«Київський Політехнічний Інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №5

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

Виконали

студенти групи ФБ-73:

Деркач Вячеслав

Михалко Дмитро

Перевірили:

Чорний О.М., Завадська Л.А.

Київ 2019

Мета комп’ютерного практикуму:

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 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 $pq \le p_1 q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 – 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), (e_1, n_1) та секретні d і d_1.

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 $0 < k < n$.

Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція Encrypt(), яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().

Хід роботи:

1) Ми прочитали завдання та методичні вказівки;
2) Проаналізували завдання та виписали всі нюанси та деталі лабораторної;
3) Написали функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту

4)

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 $pq \le p_1 q_1$; p і q – прості числа для побудови ключів абонента A, p_1 і q_1 – абонента B.

5)

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), (e_1, n_1) та секретні d і d_1.

6) Перевірити роботу програм для випадково обраного ключа . n>k>0
7) Створили протокол, що описує нашу роботу;
8) Відправили все на Github;

значення вибраних чисел *p, q, p1 , q1*
із зазначенням кандидатів, що не пройшли тест перевірки простоти, і параметрів криптосистеми RSA для абонентів *A* і *B*;

p = 31692636804681271223692616919556384426 7
****
q = 31964312729834980766047553946852422422 3
****
p1 = 29064982738913469129498018277152954496 7
****
q1 = 31847992512878786688895607498114610024 7

чисельні значення прикладів ВТ, ШТ, цифрового підпису для *A* і *B*;

M = 207595164370055065609006772617996411389
C = 175519734061121376172163096088936433577828891576349129105884911380620016879 16
k = 191457855455408794777208312020376767503385599972489669835082358033148197621 55
S = 132929318272008594961542417144785000002459205621635392691964236678781580735 35
Secret =
[577029086987727831211082205369430170134411934080925178436609270963696125600 81
,
751038067403055879810525181102426180042653883324515541997741007593694062194 49]

Програмний код:

```
import random as rnd

def Random():
   k = rnd.randint(2**256, 2**256 + 10**6)
   if k % 2 == 0:
      k += 1
   return k

def RandomBin(n):
   a = ''
   a += '1'
   for i in range(1,n-1):
      a += str(rnd.randint(0,1))
   a += '1'
```

```
    return(a)

def Decomposition(a):
    s = 0
    d = a - 1
    while 1:
        if  d % 2 == 0:
            s +=1
            d //= 2
        else :
            return d, s
        if d == 0:
            return d, s

def gsd(a, b):
    if b > a:
        a, b = b, a
    while 1:
        q = a // b
        r = a % b
        if r == 0:
            break
        a, b = b, r
    return b

def Inverse(a, m):
    if gsd(a,m) == 1:
        r0, r1 , u0, u1, y0, y1 =a, m, 1, 0, 0, 1
        while r1:
            q = r0 // r1
            r0, r1= r1, r0 % r1
            u0, u1 = u1, u0 - u1*q
            y0, y1 = y1, y0 - y1*q
            if a*u0+m*y0 == 1:
                return u0
        else:
            return 'error'

def Equation(a, m, b):
    r0, r1 , u0, u1, y0, y1 =a, m, 1, 0, 0, 1
    while r1:
        q = r0 // r1
        r0, r1= r1, r0 % r1
        u0, u1 = u1, u0 - u1*q
        y0, y1 = y1, y0 - y1*q
        if a*u0+m*y0 == 1:
            if u0 < 0:
                return ((u0 + m) * b) % m
            else:
                return (u0 * b) % m
    if b % r0 !=0:
        return 'error'
    else:
        u0 = Ober(a//r0, m//r0, b//r0)
        listO = []
        while 1:
            listO.append(u0)
            u0 += m//r0
            if u0 > m:
```

```
            return listO


def MR_Test(p):
    d, s = Decomposition(p)
    counter = 0
    k = 10
    while 1:
        x = rnd.randint(1,p)
        if gsd(x, p) > 1:
            return 'Skladne'
        else:
            if (pow(x,d,p) == 1) or (pow(x,d,p) == p-1):
                counter += 1
            else:
                for r in range(1, s+1):
                    xr = pow(x,d * (2**r), p)
                    if xr == 1:
                        break
                    if xr == p-1:
                        return "Skladne"
                    if r == s:
                        return "Skladne"
                counter += 1
        if counter >= k:
            return 'Proste'
    return

def Simple(k):
    while 1:
        if MR_Test(k) == 'Proste':
            print('***************************************')
            print('Proste', k)
            print('***************************************')
            return k
        else:
            print('Skladne', '  ', k)
            k += 2

def GenerateKeyPairs(p, q):
    n = p * q
    fi = (p-1) * (q-1)
    while 1:
        e = rnd.randint(2,fi-1)
        if gsd(e, fi) == 1:
            break
    d = Inverse(e, fi)
    if d < 0:
        d += fi
    return [n,e], [d,p,q]

def Encrypt(M, publickKey):
    n = publickKey[0]
    e = publickKey[1]
    C = pow(M, e, n)
    return C

def Decrypt(C, publickKey, privateKey):
    n = publickKey[0]
```

```
    d = privateKey[0]
    M = pow(C, d, n)
    return M

def Sign(M, privateKey, publickKey):
    d = privateKey[0]
    n = publickKey[0]
    S = pow(M, d, n)
    return S

def Verify(M, S, publickKey):
    e = publickKey[1]
    n = publickKey[0]
    if M == pow(S, e, n):
        return True
    else:
        return False

def SendKey(privateKey, publickKey, publickKey1):
    n = publickKey[0]
    n1 = publickKey1[0]
    e1 = publickKey1[1]
    d = privateKey[0]
    k = rnd.randint(1, n)
    print('k = ', k)
    print('**************************************')
    k1 = pow(k, e1, n1)
    S = Sign(k, privateKey, publickKey)
    S1 = pow(S, e1, n1)
    return [k1, S1]

def ReceiveKey(privateKey1, publickKey1, publickKey, Secret):
    d1 = privateKey1[0]
    n1 = publickKey1[0]
    n = publickKey[0]
    e = publickKey[1]
    k1 = Secret[0]
    S1 = Secret[1]
    k = pow(k1, d1, n1)
    S = pow(S1, d1, n1)
    if Verify(k, S, publickKey) == 1:
        return k
    else:
        return False

def RSA():
    p = int(RandomBin(128),2)
    q = int(RandomBin(128),2)
    p1 = int(RandomBin(128),2)
    q1 = int(RandomBin(128),2)
    p = Simple(p)
    q = Simple(q)
    p1 = Simple(p1)
    q1 = Simple(q1)
    print('**************************************')
    print('p',' = ',p)
    print('**************************************')
    print('q',' = ',q)
    print('**************************************')
```

```
    print('p1',' = ',p1)
    print('**************************************')
    print('q1',' = ',q1)
    print('**************************************')
    if p*q > p1*q1:
        p, q = p1, q1
    publickKey, privateKey = GenerateKeyPairs(p,q)
    print('publickKey = ', publickKey)
    print('**************************************')
    print('privateKey = ', privateKey)
    print('**************************************')
    publickKey1, privateKey1 = GenerateKeyPairs(p1,q1)
    print('publickKey1 = ', publickKey1)
    print('**************************************')
    print('privateKey1 = ', privateKey1)
    print('**************************************')
    M = int(RandomBin(128),2)
    print('M = ',M)
    print('**************************************')
    C = Encrypt(M, publickKey)
    print('C = ',C)
    print('**************************************')
    print('Decrypted M = ', Decrypt(C, publickKey, privateKey))
    print('**************************************')
    S = Sign(M, privateKey, publickKey)
    print('S = ',S)
    print('**************************************')
    print('Verify signanure = ', Verify(M, S, publickKey))
    print('**************************************')
    Secret = SendKey(privateKey, publickKey, publickKey1)
    print('Secret = ', Secret)
    print('**************************************')
    print('k = ', ReceiveKey(privateKey1, publickKey1, publickKey, Secret))
RSA()
```

```
Skladne    316926368046812712236926169195563844169
Skladne    316926368046812712236926169195563844171
Skladne    316926368046812712236926169195563844173
Skladne    316926368046812712236926169195563844175
Skladne    316926368046812712236926169195563844177
Skladne    316926368046812712236926169195563844179
Skladne    316926368046812712236926169195563844181
Skladne    316926368046812712236926169195563844183
Skladne    316926368046812712236926169195563844185
Skladne    316926368046812712236926169195563844187
Skladne    316926368046812712236926169195563844189
Skladne    316926368046812712236926169195563844191
Skladne    316926368046812712236926169195563844193
Skladne    316926368046812712236926169195563844195
Skladne    316926368046812712236926169195563844197
Skladne    316926368046812712236926169195563844199
Skladne    316926368046812712236926169195563844201
Skladne    316926368046812712236926169195563844203
Skladne    316926368046812712236926169195563844205
Skladne    316926368046812712236926169195563844207
Skladne    316926368046812712236926169195563844209
Skladne    316926368046812712236926169195563844211
Skladne    316926368046812712236926169195563844213
```

Skladne 31692636804681271223692616919556384421 5
Skladne 3169263680468127122369261691955638442 17
Skladne 3169263680468127122369261691955638442 19
Skladne 3169263680468127122369261691955638442 21
Skladne 3169263680468127122369261691955638442 23
Skladne 3169263680468127122369261691955638442 25
Skladne 3169263680468127122369261691955638442 27
Skladne 3169263680468127122369261691955638442 29
Skladne 3169263680468127122369261691955638442 31
Skladne 3169263680468127122369261691955638442 33
Skladne 3169263680468127122369261691955638442 35
Skladne 3169263680468127122369261691955638442 37
Skladne 3169263680468127122369261691955638442 39
Skladne 3169263680468127122369261691955638442 41
Skladne 3169263680468127122369261691955638442 43
Skladne 3169263680468127122369261691955638442 45
Skladne 3169263680468127122369261691955638442 47
Skladne 3169263680468127122369261691955638442 49
Skladne 3169263680468127122369261691955638442 51
Skladne 3169263680468127122369261691955638442 53
Skladne 3169263680468127122369261691955638442 55
Skladne 3169263680468127122369261691955638442 57
Skladne 3169263680468127122369261691955638442 59
Skladne 3169263680468127122369261691955638442 61
Skladne 3169263680468127122369261691955638442 63
Skladne 3169263680468127122369261691955638442 65
****

Proste 31692636804681271223692616919556384426 7
****

Skladne 3196431272983498076604755394685242241 93
Skladne 3196431272983498076604755394685242241 95
Skladne 3196431272983498076604755394685242241 97
Skladne 3196431272983498076604755394685242241 99
Skladne 3196431272983498076604755394685242242 01
Skladne 3196431272983498076604755394685242242 03
Skladne 3196431272983498076604755394685242242 05
Skladne 3196431272983498076604755394685242242 07
Skladne 3196431272983498076604755394685242242 09
Skladne 3196431272983498076604755394685242242 11
Skladne 3196431272983498076604755394685242242 13
Skladne 3196431272983498076604755394685242242 15
Skladne 3196431272983498076604755394685242242 17
Skladne 3196431272983498076604755394685242242 19
Skladne 3196431272983498076604755394685242242 21
****

Proste 31964312729834980766047555394685242242 23
****

Skladne 2906498273891346912949801827715295446 05
Skladne 2906498273891346912949801827715295446 07
Skladne 2906498273891346912949801827715295446 09
Skladne 2906498273891346912949801827715295446 11
Skladne 2906498273891346912949801827715295446 13
Skladne 2906498273891346912949801827715295446 15
Skladne 2906498273891346912949801827715295446 17
Skladne 2906498273891346912949801827715295446 19
Skladne 2906498273891346912949801827715295446 21
Skladne 2906498273891346912949801827715295446 23
Skladne 2906498273891346912949801827715295446 25
Skladne 2906498273891346912949801827715295446 27
Skladne 2906498273891346912949801827715295446 29
Skladne 2906498273891346912949801827715295446 31
Skladne 2906498273891346912949801827715295446 33
Skladne 2906498273891346912949801827715295446 35
Skladne 2906498273891346912949801827715295446 37
Skladne 2906498273891346912949801827715295446 39
Skladne 2906498273891346912949801827715295446 41
Skladne 2906498273891346912949801827715295446 43
Skladne 2906498273891346912949801827715295446 45
Skladne 2906498273891346912949801827715295446 47
Skladne 2906498273891346912949801827715295446 49

Skladne 290649827389134691294980182771529544651
Skladne 290649827389134691294980182771529544653
Skladne 290649827389134691294980182771529544655
Skladne 290649827389134691294980182771529544657
Skladne 290649827389134691294980182771529544659
Skladne 290649827389134691294980182771529544661
Skladne 290649827389134691294980182771529544663
Skladne 290649827389134691294980182771529544665
Skladne 290649827389134691294980182771529544667
Skladne 290649827389134691294980182771529544669
Skladne 290649827389134691294980182771529544671
Skladne 290649827389134691294980182771529544673
Skladne 290649827389134691294980182771529544675
Skladne 290649827389134691294980182771529544677
Skladne 290649827389134691294980182771529544679
Skladne 290649827389134691294980182771529544681
Skladne 290649827389134691294980182771529544683
Skladne 290649827389134691294980182771529544685
Skladne 290649827389134691294980182771529544687
Skladne 290649827389134691294980182771529544689
Skladne 290649827389134691294980182771529544691
Skladne 290649827389134691294980182771529544693
Skladne 290649827389134691294980182771529544695
Skladne 290649827389134691294980182771529544697
Skladne 290649827389134691294980182771529544699
Skladne 290649827389134691294980182771529544701
Skladne 290649827389134691294980182771529544703
Skladne 290649827389134691294980182771529544705
Skladne 290649827389134691294980182771529544707
Skladne 290649827389134691294980182771529544709
Skladne 290649827389134691294980182771529544711
Skladne 290649827389134691294980182771529544713
Skladne 290649827389134691294980182771529544715
Skladne 290649827389134691294980182771529544717
Skladne 290649827389134691294980182771529544719
Skladne 290649827389134691294980182771529544721
Skladne 290649827389134691294980182771529544723
Skladne 290649827389134691294980182771529544725
Skladne 290649827389134691294980182771529544727
Skladne 290649827389134691294980182771529544729
Skladne 290649827389134691294980182771529544731
Skladne 290649827389134691294980182771529544733
Skladne 290649827389134691294980182771529544735
Skladne 290649827389134691294980182771529544737
Skladne 290649827389134691294980182771529544739
Skladne 290649827389134691294980182771529544741
Skladne 290649827389134691294980182771529544743
Skladne 290649827389134691294980182771529544745
Skladne 290649827389134691294980182771529544747
Skladne 290649827389134691294980182771529544749
Skladne 290649827389134691294980182771529544751
Skladne 290649827389134691294980182771529544753
Skladne 290649827389134691294980182771529544755
Skladne 290649827389134691294980182771529544757
Skladne 290649827389134691294980182771529544759
Skladne 290649827389134691294980182771529544761
Skladne 290649827389134691294980182771529544763
Skladne 290649827389134691294980182771529544765
Skladne 290649827389134691294980182771529544767
Skladne 290649827389134691294980182771529544769
Skladne 290649827389134691294980182771529544771
Skladne 290649827389134691294980182771529544773
Skladne 290649827389134691294980182771529544775
Skladne 290649827389134691294980182771529544777
Skladne 290649827389134691294980182771529544779
Skladne 290649827389134691294980182771529544781
Skladne 290649827389134691294980182771529544783
Skladne 290649827389134691294980182771529544785
Skladne 290649827389134691294980182771529544787
Skladne 290649827389134691294980182771529544789

Skladne 290649827389134691294980182771529544791
Skladne 290649827389134691294980182771529544793
Skladne 290649827389134691294980182771529544795
Skladne 290649827389134691294980182771529544797
Skladne 290649827389134691294980182771529544799
Skladne 290649827389134691294980182771529544801
Skladne 290649827389134691294980182771529544803
Skladne 290649827389134691294980182771529544805
Skladne 290649827389134691294980182771529544807
Skladne 290649827389134691294980182771529544809
Skladne 290649827389134691294980182771529544811
Skladne 290649827389134691294980182771529544813
Skladne 290649827389134691294980182771529544815
Skladne 290649827389134691294980182771529544817
Skladne 290649827389134691294980182771529544819
Skladne 290649827389134691294980182771529544821
Skladne 290649827389134691294980182771529544823
Skladne 290649827389134691294980182771529544825
Skladne 290649827389134691294980182771529544827
Skladne 290649827389134691294980182771529544829
Skladne 290649827389134691294980182771529544831
Skladne 290649827389134691294980182771529544833
Skladne 290649827389134691294980182771529544835
Skladne 290649827389134691294980182771529544837
Skladne 290649827389134691294980182771529544839
Skladne 290649827389134691294980182771529544841
Skladne 290649827389134691294980182771529544843
Skladne 290649827389134691294980182771529544845
Skladne 290649827389134691294980182771529544847
Skladne 290649827389134691294980182771529544849
Skladne 290649827389134691294980182771529544851
Skladne 290649827389134691294980182771529544853
Skladne 290649827389134691294980182771529544855
Skladne 290649827389134691294980182771529544857
Skladne 290649827389134691294980182771529544859
Skladne 290649827389134691294980182771529544861
Skladne 290649827389134691294980182771529544863
Skladne 290649827389134691294980182771529544865
Skladne 290649827389134691294980182771529544867
Skladne 290649827389134691294980182771529544869
Skladne 290649827389134691294980182771529544871
Skladne 290649827389134691294980182771529544873
Skladne 290649827389134691294980182771529544875
Skladne 290649827389134691294980182771529544877
Skladne 290649827389134691294980182771529544879
Skladne 290649827389134691294980182771529544881
Skladne 290649827389134691294980182771529544883
Skladne 290649827389134691294980182771529544885
Skladne 290649827389134691294980182771529544887
Skladne 290649827389134691294980182771529544889
Skladne 290649827389134691294980182771529544891
Skladne 290649827389134691294980182771529544893
Skladne 290649827389134691294980182771529544895
Skladne 290649827389134691294980182771529544897
Skladne 290649827389134691294980182771529544899
Skladne 290649827389134691294980182771529544901
Skladne 290649827389134691294980182771529544903
Skladne 290649827389134691294980182771529544905
Skladne 290649827389134691294980182771529544907
Skladne 290649827389134691294980182771529544909
Skladne 290649827389134691294980182771529544911
Skladne 290649827389134691294980182771529544913
Skladne 290649827389134691294980182771529544915
Skladne 290649827389134691294980182771529544917
Skladne 290649827389134691294980182771529544919
Skladne 290649827389134691294980182771529544921
Skladne 290649827389134691294980182771529544923
Skladne 290649827389134691294980182771529544925
Skladne 290649827389134691294980182771529544927
Skladne 290649827389134691294980182771529544929

Skladne 290649827389134691294980182771529544931
Skladne 29064982738913469129498018277152954493 3
Skladne 29064982738913469129498018277152954493 5
Skladne 29064982738913469129498018277152954493 7
Skladne 29064982738913469129498018277152954493 9
Skladne 29064982738913469129498018277152954494 1
Skladne 29064982738913469129498018277152954494 3
Skladne 29064982738913469129498018277152954494 5
Skladne 29064982738913469129498018277152954494 7
Skladne 29064982738913469129498018277152954494 9
Skladne 29064982738913469129498018277152954495 1
Skladne 29064982738913469129498018277152954495 3
Skladne 29064982738913469129498018277152954495 5
Skladne 29064982738913469129498018277152954495 7
Skladne 29064982738913469129498018277152954495 9
Skladne 29064982738913469129498018277152954496 1
Skladne 29064982738913469129498018277152954496 3
Skladne 29064982738913469129498018277152954496 5
****

Proste 29064982738913469129498018277152954496 7
****

Skladne 31847992512878786688895607498114610022 7
Skladne 31847992512878786688895607498114610022 9
Skladne 31847992512878786688895607498114610023 1
Skladne 31847992512878786688895607498114610023 3
Skladne 31847992512878786688895607498114610023 5
Skladne 31847992512878786688895607498114610023 7
Skladne 31847992512878786688895607498114610023 9
Skladne 31847992512878786688895607498114610024 1
Skladne 31847992512878786688895607498114610024 3
Skladne 31847992512878786688895607498114610024 5
****

Proste 318479925128787866888956074981146100247

Висновки:

Ми провели 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.