



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»  
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

## Комп'ютерний практикум №4

з дисципліни КRYPTOграфія:

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

Виконав:

студент III курсу ФТІ

групи ФБ-95

Колесник Вікторія  
студент III курсу ФТІ

групи ФБ-96

Ліпатова Софія

Перевірила:

Селюх П.В.

## 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  $p, q$  і  $1 < p, q$  довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб  $p \cdot q \leq 2^{256}$ ;  $p$  і  $q$  – прості числа для побудови ключів абонента А,  $1 < p < q$  – 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  $(e, n)$ ,  $(d, p, q)$  і  $(e, n)$  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  $0 < k < n$ .

## Порядок виконання роботи

1. Реалізовуємо функцію генерації випадкового великого числа. У цій же функції перевіряємо це число на простоту за допомогою теста Міллера-Рабіна. Якщо тест не проходить, генерується нове число

```

def testmilrab(number):
    t, s = number - 1, 0
    while t % 2 == 0:
        t = t // 2
        s += 1

    for i in range(300):
        a = random.randint(2, number - 2)
        x = pow(a, t, number)
        if (x == number - 1) or (x == 1):
            continue
        x = pow(x, 2, number)
        if x == 1 or x != number - 1:
            return False
    return True

def generate_random_number(min = min, max = max):
    number = random.randrange(min, max)

    while not testmilrab(number):
        number = random.randrange(min, max)

    return number

```

2. Реалізована функція генерацію ключових пар. За допомогою функції генерації випадкового великого числа генеруємо числа **p** та **q**. Знаходимо модуль **n = q\*p** та застосовуємо функцію Ойлера до **n** (**fi\_n = (p-1)\*(q-1)**). Знаходимо секретний ключ **d = e<sup>-1</sup>mod(fi\_n)**. Та генеруємо ключові пари: відкритий ключ(**e, n**), секретний ключ(**d,n**), де **e** - **2<sup>16</sup>+1**, відкритий ключ.

В нашому випадку генеруємо ключі для абонентів А та В

```

def generate_rsa(pair):
    n = pair[0] * pair[1]
    phi = (pair[0] - 1) * (pair[1] - 1)
    flag = False
    while flag == False:
        e = random.randint(2, phi)

```

```

    g, x, y = gcdExtended(e, phi)
    if g == 1:
        flag = True
    d = inverse(e, phi)
    open_key = (e, n)
    private_key = (d, pair[0], pair[1])
    return open_key, private_key

```

3. Реалізували функції шифрування, розшифрування, та створення повідомлення з цифровим підписом для абонентів А і В.

```

def __init__(self):
    pair = create_pair()
    self.open_key, self.private_key = generate_rsa(pair)

def encrypt(self, text, open_key):
    return pow(text, open_key[0], open_key[1])

def decrypt(self, text, private_key):
    return pow(text, private_key[0], private_key[1] * private_key[2])

def sign(self, text, private_key):
    return pow(text, private_key[0], private_key[1] * private_key[2])

def verify(self, sign, text, open_key):
    return text == pow(sign, open_key[0], open_key[1])

def encrypt_message(self, text, open_key):
    enc_text = self.encrypt(text, open_key)
    sign = self.sign(text, self.private_key)
    return enc_text, sign

def decrypt_verify_message(self, packet, open_key):
    text, sign = packet
    text = self.decrypt(text, self.private_key)

    if self.verify(sign, text, open_key):
        print(text)

a = Abonent()
b = Abonent()

```

4. Вибираємо випадкове повідомлення(число). За допомогою згенерованих ключів шифруємо, дешифруємо повідомлення та знаходимо підпис. Перевіряємо підпис функцією верифікації.

Абонент А маючи свій ключ та відкритий ключ абонента В формує повідомлення (k1, s1) та відправляє його до В.

За допомогою свого приватного ключа, В може розшифрувати повідомлення та перевірити підпис А за його публічним ключем

A open key:

33144382631289908050441228261240138806700141951719765299955266016895709282857004  
26628305678355571546494338296714278278055583661225201895876832845617058759  
55971963223682829823259413030018140344418239530290518764695119741527662495461871  
06279929556411545784668511950330874399300807260667145367628859542907591003

A priavte key:

41100044588588065235814112385774078819453472213532410189651958279051650251292549  
50536460187605330280288345242551400546401920373888994378925431049074509967  
75094027628977593738770472256727573443202710555968401307752469020276536375669  
74535838589225086825797127399798256286099242692102046120460084324955723046287

B open key:

14988055732484852848717145494983001725877462177299577265919168287301538913986762  
05351550919964223616722713889010497238193621224422455908420452517949686633  
23735462005161023139491955328940879527219209968447702013523317305164645393980516  
99376690372486299938648313972633621884879374897138778329497841022425523053

B priavte key:

20643283656308678361782383733342299077697295223912171467318553380212055879735228  
93225037823633677601187011470726136325301063749812852824753077490677432285  
51777908764847769102971289093237551023794788140504340227031863784414579737579  
45840905072000752443848032087950981362006083447119526379689079145609219840007

## Перевірка за допомогою сайта

### Get server key

Key size

309


Modulus

145A7E0C056D97960B7F598600B1B36FA496E82B164BEF32DB0750DB4CD20C346FF35CBBF325C5

Public exponent

10001

## Encryption



Modulus

145A7E0C056D97960B7F598600B1B36FA496E82B164BEF32DB0750DB4CD20C346FF35CBBF325C5

Public exponent

10001

Message

312


Bytes

Encrypt

Ciphertext

022B3DE44566E5178B523727043299D0F58B81B24DDAA9D5D12F55E72383084B6BF07C8BB5B3CB

## Decryption



Ciphertext

022B3DE44566E5178B523727043299D0F58B81B24DDAA9D5D12F55E72383084


Bytes

Decrypt

Message

0312

## Sign



Message

0312

Bytes

Sign

Signature

F8A729F63D4B1193C3049A998C5058B00FEB3D398611E2F696C04C937C81BA453093AA8E714B

**Висновок:** Ознайомилися з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практично ознайомилися з системою захисту інформації на основі криптосхеми RSA, організували з використанням цієї системи засекречений зв'язок й електронний підпис, вивчили протокол розсилання ключів.