

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №4
з дисципліни «Криптографія»

Виконав:

студент 3 курсу

НН ФТІ групи ФБ-25

Черняк Денис

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

Мета 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.

Хід 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

```
def is_prime(p, primes=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47], k=10):
    for prime in primes:
        if p % prime == 0:
            return False

    s, d = 0, p - 1
    while d % 2 == 0:
        s += 1
        d //= 2

    for _ in range(k):
        x = random.randrange(2, p)
        y = pow(x, d, p)

        if gcd(x, p) != 1:
            return False

        if y in (1, p - 1):
            continue

        for _ in range(s - 1):
            y = pow(y, 2, p)
            if y == 1:
                return False
            elif y == p - 1:
                break
        else:
            return False
    return True
```

```
def GeneratePrime(bits):
    while True:
        x = random.getrandbits(bits) | (1 << (bits - 1)) | 1

        if is_prime(x):
            return x
        else:
            print(f"Кандидат {x} не є простим числом")
```

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 $p \cdot q \leq p_1 \cdot q_1$; p і q – прості числа для побудови ключів абонента А, p_1 і q_1 – абонента В.

```
bits = 256

while True:
    p = GeneratePrime(bits)
    q = GeneratePrime(bits)
    p1 = GeneratePrime(bits)
    q1 = GeneratePrime(bits)
    print(f"Перевірка потенційних пар простих чисел: p={p}, q={q}, p1={p1}, q1={q1}")

    if p * q <= p1 * q1:
        break
```

```
p=65591245601768441168887670728635403274471641392145849430900866528638408340081, q=64893971463171984165683898055064854954567912170594943194711736657171321576381
p1=107740228391240460592605454953777649222984130858105416316492715152871651856499, q1=115742475952286724694221719023248338316218497255350632599297569798007447167859
```

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), (e_1, n_1) та секретні d і d_1 .

```
def GenerateKeyPair(p, q):
    n = p * q
    f = (p - 1) * (q - 1)
    e = 65537
    d = pow(e, -1, f)
    return ((n, e), (d, p, q))

Ключі абонента А:
Відкритий ключ (n, e): (4256476240315066139198778668898719705783458420083461054848786442518375056635448547924057475355743580889143662815509017372958532899109381447582454465224861, 65537)
Секретний ключ (d, p, q): (1597972837411414731965842797695275457239364190727886167719247788075317757807888997194605250918226889775774547468583801124073438529694167396242352364870273, 6559124560176844116888767072863540327447164)

Ключі абонента В:
Відкритий ключ (n, e): (12470120793667028441029897130558921418069024403965177169274498922098986509219075035916487336045374484910664423031660929076649974693659840454459957033065641, 65537)
Секретний ключ (d, p, q): (110645516906745457357031679530952176783345250635066118976016772693733131111619057827816727249412236428844985452801759284676685218832899517757727489053, 1077402283912404605926054549537776492229841)
```

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.

```
def Encrypt(message, public_key):
    n, e = public_key
    return pow(message, e, n)

def Decrypt(ciphertext, private_key):
    d, p, q = private_key
    n = p * q
    return pow(ciphertext, d, n)

def Sign(message, private_key):
    d, p, q = private_key
    n = p * q
    return pow(message, d, n)

def Verify(message, signature, public_key):
    n, e = public_key
    return pow(signature, e, n) == message

Перевірка шифрування, розшифрування та підпису для абонента А
Повідомлення: 282568247009273523827671013217347211027099061530577220233265849221936930568409527261738771016642214888276548662868283638583672822867585130570768107651359
Зашифроване повідомлення: 1593543118498211626589901691912754244389586244458059829488054079978345829287961433817795871865749497993630132817014135992052605853910096120111541763773832
Розшифроване повідомлення: 282568247009273523827671013217347211027099061530577220233265849221936930568409527261738771016642214888276548662868283638583672822867585130570768107651359
Повідомлення розшифровано правильно
Цифровий підпис: 3207991420139171490268539864398220108119969588542691956282647159637305705958835115229415919146644229496743804551730943123873582214410095228751761389474573
Підпис дійсний

Перевірка шифрування, розшифрування та підпису для абонента В
Повідомлення: 6682677235237046817337297491103165346768877865648950896157015596105979161679418590476008536550957789622806965785720927837588146066978468745340883495955674
Зашифроване повідомлення: 843025019493575827662239879461530254506262565226971939518483291157239664812617656420004701833512696303681829434026493423236750424776833908724486995952524
Розшифроване повідомлення: 6682677235237046817337297491103165346768877865648950896157015596105979161679418590476008536550957789622806965785720927837588146066978468745340883495955674
Повідомлення розшифровано правильно
Цифровий підпис: 265937468176076443959605582551918659010797784622539778155891296680226350825725736664733402768096996775957411886095806811392519558933102557700682720579311
Підпис дійсний
```

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 $0 < k < n$.

```
def SendKey(key, sender_private_key, receiver_public_key):
    k1 = Encrypt(key, receiver_public_key)
    signature = Sign(key, sender_private_key)
    signature1 = Encrypt(signature, receiver_public_key)
    return k1, signature1

def ReceiveKey(encrypted_key, encrypted_signature, receiver_private_key, sender_public_key):
    k = Decrypt(encrypted_key, receiver_private_key)
    signature = Decrypt(encrypted_signature, receiver_private_key)
    if Verify(k, signature, sender_public_key):
        return k
    return None
```

```
Перевірка протоколу розсилання ключів
Ключ: 3687496460900708551303365906863854532611370407050372276392565631661501205736694780946611667201947304173228020105462327198305273766424014359938540819487814
Зашифрований ключ: 33055142619200471184212394814994479510612952760010438002662461255530132983438264905269181287329236189447707926076470864538069669737811964186523992423406928
Зашифрований підпис: 59405110169035051544548892287367785336208070963232823896918779037323460522384779291184457367300734509429096837793504042779347209462853208660145271302444
Отриманий ключ: 3687496460900708551303365906863854532611370407050372276392565631661501205736694780946611667201947304173228020105462327198305273766424014359938540819487814
Ключ передано успішно
```

Перевіряю на сайті: <https://www.dcode.fr/rsa-cipher>

Як можна побачити, повідомлення зашифроване абонентом А, правильно розшифровується його приватним ключем, що свідчить про правильну роботу функцій:

```
Перевірка шифрування, розшифрування та підпису для абонента А
Повідомлення: 1407541347269286493208747487006736781464243644397479813410553181496703504542095408746883575282856267764916224455254282396533656524733436313636728773140421
Зашифроване повідомлення: 2303408140728774065260419260827744812829657888777932796354154890008563784317995541322910736086178117655646620495115754118227285056588058606090103789435738
Розшифроване повідомлення: 1407541347269286493208747487006736781464243644397479813410553181496703504542095408746883575282856267764916224455254282396533656524733436313636728773140421
Повідомлення розшифровано правильно
Ключі абонента А:
Відкритий ключ (n, e): (562899187787766086914409002552059471438556061300788960746531246478721191356364408461622369563852593220883175122402904015520357421503266048351390343655761, 65537)
Секретний ключ (d, p, q): (342101021554033953977156576788062754587449653197589518997426484969899018820470429863408210644934383316954637289869728002242787651585523981164525458957993, 580889475676958241536167317195226170874797)
```

Search for a tool

SEARCH A TOOL ON dCODE BY KEYWORDS:

BROWSE THE FULL dCODE TOOLS' LIST

Results

Decryption using C,D,N

1407541347269286493208747487006736781464243644397479813410553181496703504542095408746883575282856267764916224455254282396533656524733436313636728773140421

ONLY 1% CAN SOLVE!

IQ: 155

TRY

RSA Cipher - dCode

Tag(s) : Modern Cryptography, Arithmetics

RSA DECODER

Indicate known numbers, leave remaining cells empty.

VALUE OF THE CIPHER MESSAGE (INTEGER) C=

2303408140728774065260419260827744812829657888777...

PUBLIC KEY E (USUALLY E=65537) E=

65537

PUBLIC KEY VALUE (INTEGER) N=

5628991877877660869144090025520594714385560613007...

PRIVATE KEY VALUE (INTEGER) D=

3421010215540339539771565767680627545874496531975...

FACTOR 1 (PRIME NUMBER) P=

FACTOR 2 (PRIME NUMBER) Q=

INTERMEDIATE VALUE PHI (INTEGER) φ=

DISPLAY

☐ PLAINTEXT AS CHARACTER STRING

☐ COMPUTED VALUES (C,D,E,N,P,Q,...)

☒ PLAINTEXT AS INTEGER NUMBER

☐ PLAINTEXT AS HEXADECMAL FORMAT

CALCULATE/DECRYPT

RSA CERTIFICATE READER

Висновок:

У ході виконання лабораторної роботи я ознайомився з принципами функціонування криптосистеми RSA та алгоритму цифрового підпису. Реалізував функції для генерації простих чисел за допомогою тесту Міллера–Рабіна, створення ключових пар RSA (відкритого та закритого ключів), а також функції шифрування, розшифрування, підписання та перевірки цифрового підпису. Також було реалізовано протокол конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу.