Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №4

Вивчення криптосистеми RSA та алгоритму електронного

підпису; ознайомлення з методами генерації параметрів для

асиметричних криптосистем

Виконали:  
ФБ-21 Худоба Арсен,

ФБ-21 Шабанов Кирило

**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.**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.

# Тест Міллера-Рабіна

def miller\_rabin(p, k=5):

if p == 2 or p == 3:

return True

if p < 2 or p % 2 == 0:

return False

# Крок 0: розклад p-1 = d \* 2^s

s, d = 0, p - 1

while d % 2 == 0:

s += 1

d //= 2

# Крок 1: k раундів перевірок

for \_ in range(k):

x = random.randint(2, p - 2) # вибір випадкової основи

g = gcd(x, p)

# Якщо x і p не взаємно прості, то p складене

if g != 1:

return False

# Крок 2: перевірка сильної псевдопростоти

x\_power = pow(x, d, p) # x^d mod p

if x\_power == 1 or x\_power == p - 1:

continue

for r in range(s - 1):

x\_power = pow(x\_power, 2, p) # x^(2^r \* d) mod p

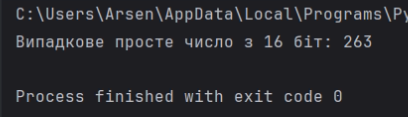
if x\_power == p - 1:

break

else:

return False # Якщо не знайдено псевдопростоти, p складене

return True # Якщо p пройшло всі k раундів, воно, ймовірно, просте



2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p1 , q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq ≤ p1q1 ; p і q – прості числа для побудови ключів абонента А, p1 і q1 – абонента B.

**p**=36646164541686624240937833154972327964008617161515222144838114780214993351409

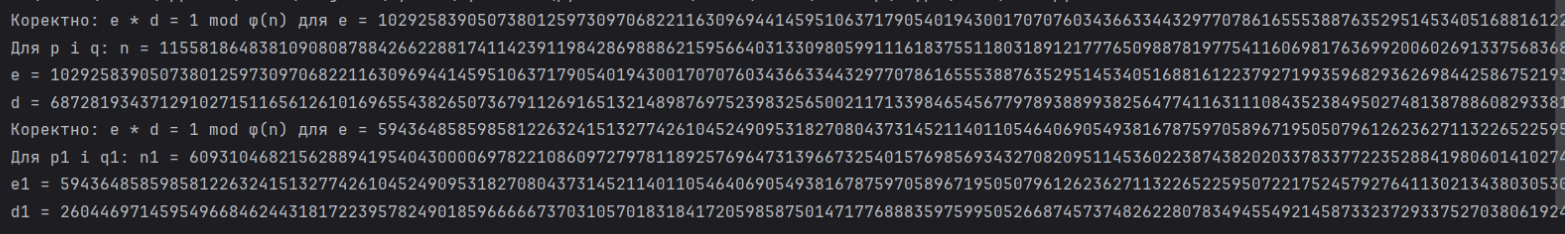
**q**=31539962308096287988814609326825756936315439947047156766760061841746726548927

**p1**=73431616661604875049686928558414853172421655489331888049826565905174368641603

**q1**=82976583645640818895092178503362825123374069319439494124111219775406714256373

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 B – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n), (e1, n1) та секретні d і d1 .

|  |  |  |
| --- | --- | --- |
| USER A  Public key | e | 1029258390507380125973097068221163096944145951063717905401943001707076034366334432977078616555388763529514534051688161223792719935968293626984425867521933 |
| n | 1155818648381090808788426622881741142391198428698886215956640313309805991116183755118031891217776509887819775411606981763699200602691337568368892342888143 |
| USER A Private key | d | 687281934371291027151165612610169655438265073679112691651321489876975239832565002117133984654567797893889938256477411631110843523849502748138788608293381 |
| p | 36646164541686624240937833154972327964008617161515222144838114780214993351409 |
| q | 31539962308096287988814609326825756936315439947047156766760061841746726548927 |
| USER B  Public key | e1 | 5943648585985812263241513277426104524909531827080437314521140110546406905493816787597058967195050796126236271132265225950722175245792764113021343803053007 |
| n1 | 6093104682156288941954043000069782210860972797811892576964731396673254015769856934327082095114536022387438202033783377223528841980601410274294334295685919 |
| USER B  Private key | d1 | 2604469714595496684624431817223957824901859666667370310570183184172059858750147177688835975995052668745737482622807834945549214587332372933752703806192447 |
| p1 | 73431616661604875049686928558414853172421655489331888049826565905174368641603 |
| q1 | 82976583645640818895092178503362825123374069319439494124111219775406714256373 |



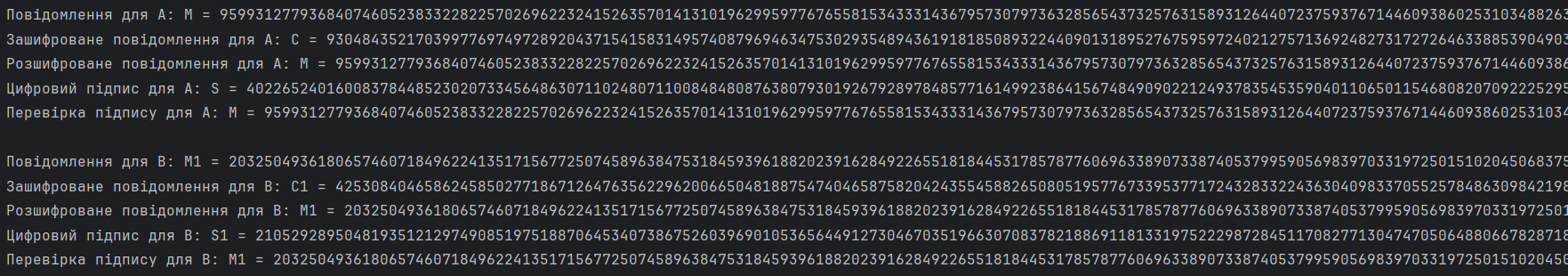
4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів А і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів А и B, перевірити правильність розшифрування. Скласти для А і B повідомлення з цифровим підписом і перевірити його.

**Зашифрувати C = Memodn**

**Розшифрувати M = Cdmodn**

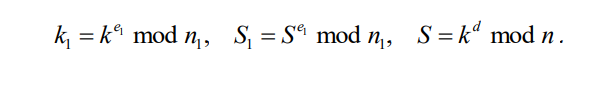
**Цифровий підпис створити S=Mdmodn**

**Перевірити цифровий підпис M=Semodn**

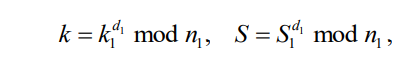


5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді 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.

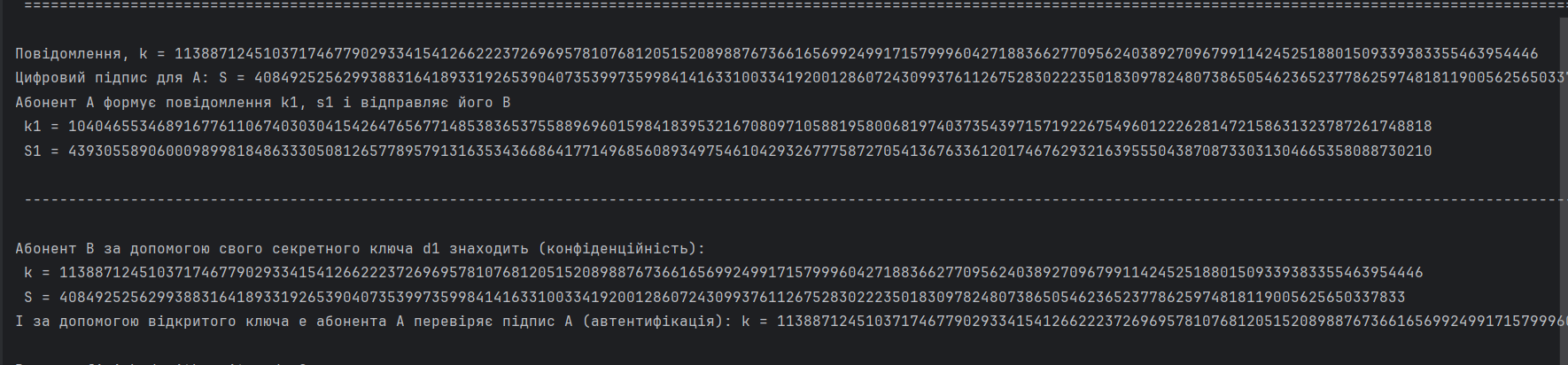
Абонент А формує повідомлення k1, s1 і відправляє його B, де



Абонент B за допомогою свого секретного ключа d1 знаходить (конфіденційність):



і за допомогою відкритого ключа e абонента А перевіряє підпис А (автентифікація):

**Висновки:**  
У роботі була розроблена функція генерації випадкових простих чисел заданої довжини, яка використовує тест Міллера-Рабіна в комбінації з попередніми пробними діленнями для перевірки чисел на простоту. Такий підхід гарантує отримання простих чисел високої якості, необхідних для криптографічних цілей. Генерація двох пар простих чисел довжиною понад 256 біт дозволила забезпечити належну стійкість системи до атак​. Використання власної реалізації тесту забезпечило розуміння алгоритму його роботи та дозволило адаптувати його до специфічних потреб завдання.  
Було реалізовано функції для роботи з RSA, зокрема генерацію ключових пар, шифрування, розшифрування, створення та перевірку цифрового підпису. Особливу увагу приділено правильній реалізації операцій піднесення до степеня за модулем, яка виконувалася за допомогою схеми Горнера, що забезпечує ефективність навіть для великих чисел. Ці функції дозволяють виконувати всі необхідні криптографічні операції в системі RSA, забезпечуючи конфіденційність, автентифікацію та цілісність повідомлень між абонентами.  
На завершальному етапі було розроблено та протестовано протокол конфіденційної передачі ключів за допомогою алгоритму RSA. Протокол передбачає використання відкритих та закритих ключів абонентів для шифрування та підпису повідомлень, що гарантує підтвердження автентичності відправника і захист переданих даних у відкритому каналі. Проведене тестування для випадково згенерованих ключів та повідомлень підтвердило коректність реалізації та забезпечення безпеки обміну. Це демонструє повну функціональність розробленої системи, яка може бути використана у реальних сценаріях.