НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені Ігоря Сікорського»

Фізико-Технічний Інститут

Звіт

із лабораторної роботи №4

із дисципліни «Криптографія»

на тему

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

Виконав:  
студент групи ФБ-13

Берчук В.В.

Київ – 2023

**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.
2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб pq ≤ p1q1 ; p і q – прості числа для побудови ключів абонента А, 1 p і q1 – 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 А і B
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.
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.
6. Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою: http://asymcryptwebservice.appspot.com/?section=rsa

**Хід роботи**

1. Реалізація функції пошуку випадкового простого числа заданої довжини, з використанням модулю random у python, та реалізація тесту Міллера – Рабіна.

Тест Міллера – Рабіна (з пробними діленнями):

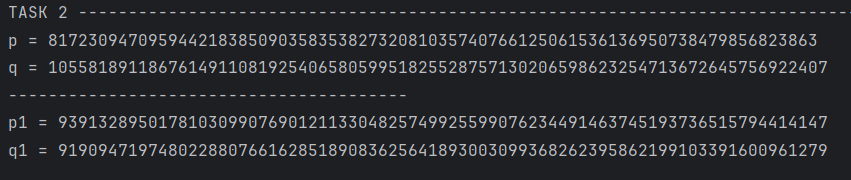
def MillerRabin(p, k = 100):  
 primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]  
 for prime in primes:  
 if p % prime == 0:  
 return False  
  
 d = p - 1  
 s = 0  
 while d % 2 == 0:  
 d //= 2  
 s += 1  
  
 for i in range(k):  
 x = random.randint(2, p - 1)  
 if gcd(x, p) > 1:  
 return False  
  
 if pow(x, d, p) != 1 and pow(x, d, p) != p - 1:  
 for r in range(1, s - 1):  
 xr = pow(x, d \* (2 \*\* r), p)  
 if xr == - 1:  
 break  
 else:  
 return False  
 return True

Генератор:

def PrimeGen(bits):  
 while True:  
 p = random.randint(2 \*\* (bits - 1), 2 \*\* bits)  
 if MillerRabin(p):  
 return p

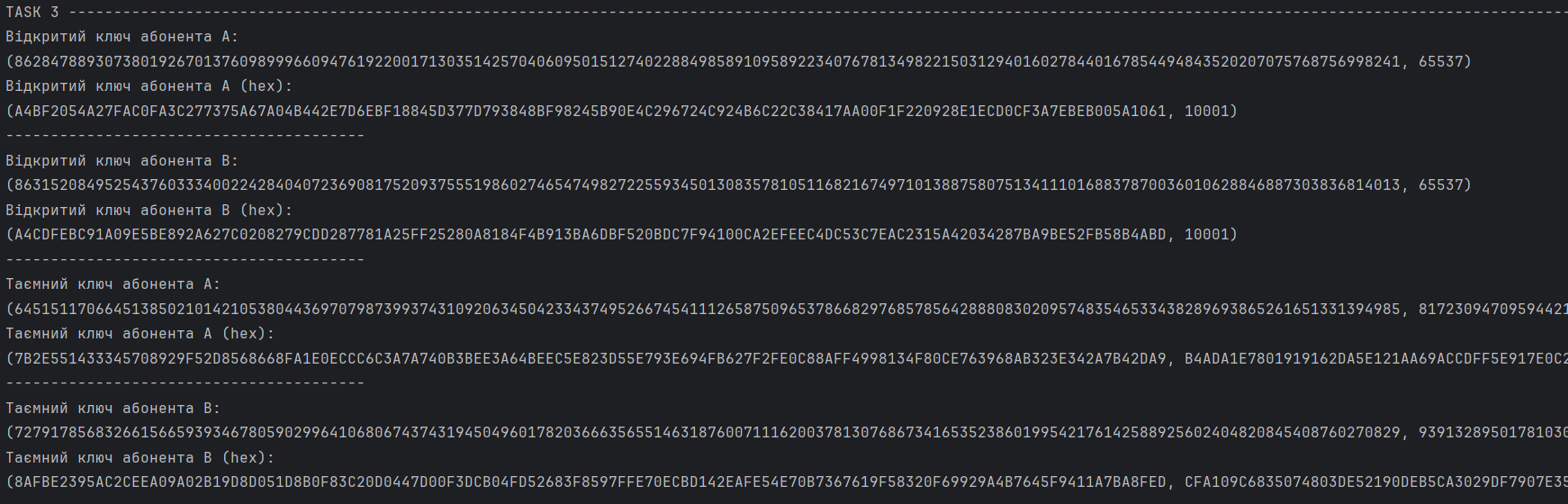
1. Результат функції генерування простих p,q,p1,q1 (pq ≤ p1q1):

p, q = PrimeGen(256), PrimeGen(256)  
p1, q1 = PrimeGen(256), PrimeGen(256)  
while p \* q > p1 \* q1:  
 p1, q1 = PrimeGen(256), PrimeGen(256)



1. Функція генерації ключових пар для RSA:

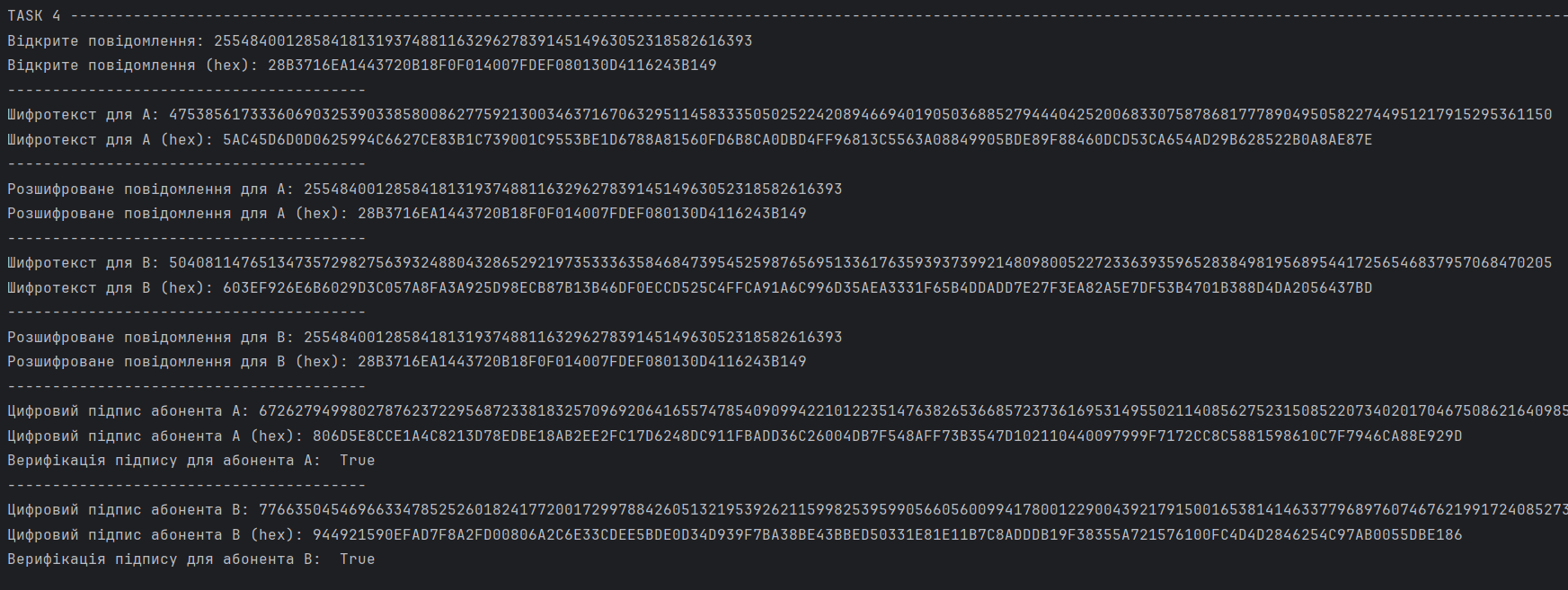
def GenerateKeyPair(p, q):  
 n = p \* q  
 phi = (p - 1) \* (q - 1)  
 e = 65537 #2^16 + 1  
 d = pow(e, -1, phi)  
 return (d, p, q), (n, e)  
  
A\_private, A\_public = GenerateKeyPair(p, q)  
B\_private, B\_public = GenerateKeyPair(p1, q1)



1. Функції шифрування, дешифрування, підписування та верифікації:

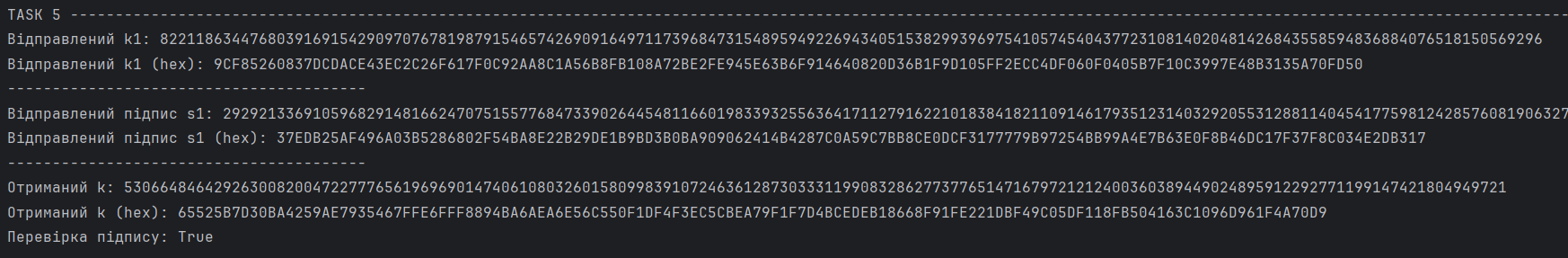
def Encrypt(m, pubkey):  
 n, e = pubkey  
 c = pow(m, e, n)  
 return c  
  
def Decrypt(c, privkey):  
 d, p, q = privkey  
 m = pow(c, d, p \* q)  
 return m  
  
def Sign(m, privkey):  
 d, p, q = privkey  
 s = pow(m, d, p \* q)  
 return s  
  
def Verify(m, s, pubkey):  
 n, e = pubkey  
 v = pow(s, e, n)  
 return v == m

m = random.randint(10\*\*50, 10\*\*60)  
c1 = Encrypt(m, A\_public)  
m1 = Decrypt(c1, A\_private)  
c2 = Encrypt(m, B\_public)  
m2 = Decrypt(c2, B\_private)  
s1 = Sign(m, A\_private)  
s2 = Sign(m, B\_private)



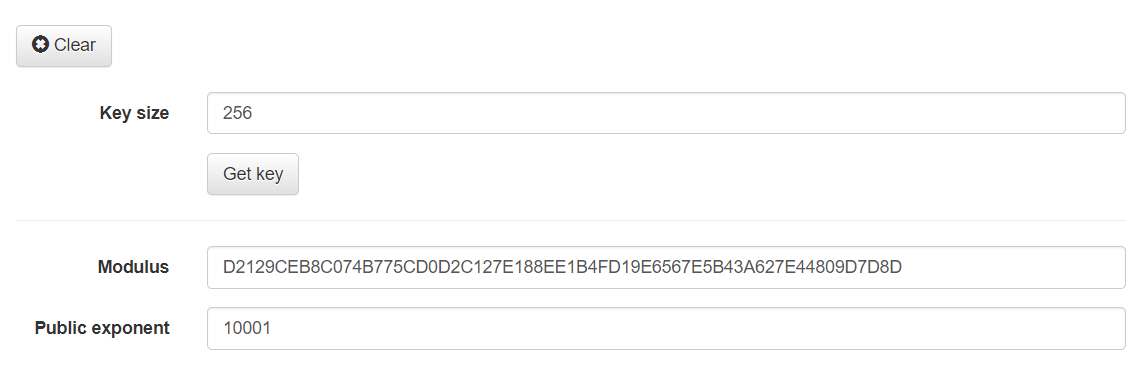
1. Функції надсилання та отримання ключа:

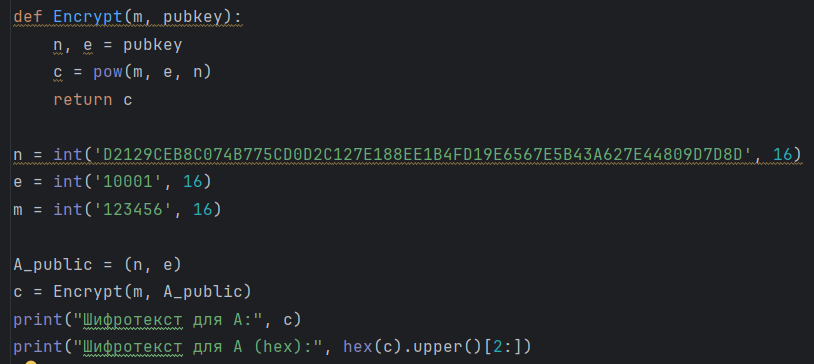
def SendKey(k, B\_public, A\_private):  
 k1 = Encrypt(k, B\_public)  
 s = Sign(k, A\_private)  
 s1 = Encrypt(s, B\_public)  
 return k1, s1  
  
k = random.randint(1, A\_public[0] - 1)  
k1, s1 = SendKey(k, B\_public, A\_private)  
  
def ReceiveKey(k1, s1, A\_public, B\_private):  
 k = Decrypt(k1, B\_private)  
 s = Decrypt(s1, B\_private)  
 check = Verify(k, s, A\_public)  
 return k, check  
  
k\_res, check = ReceiveKey(k1, s1, A\_public, B\_private)

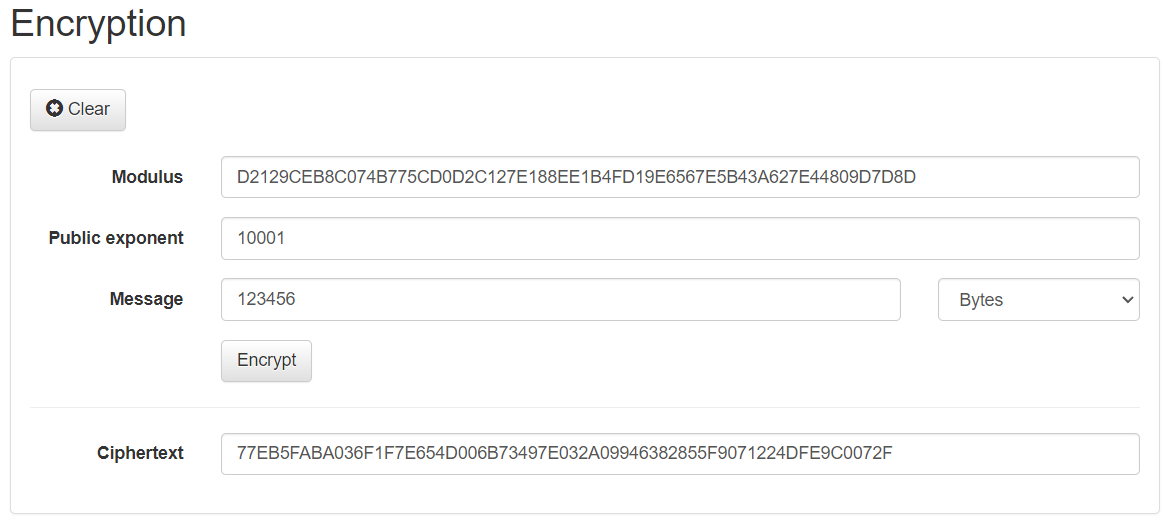


1. **Перевірка шифрування за допомогою http://asymcryptwebservice.appspot.com/?section=rsa**

Генеруємо ключ

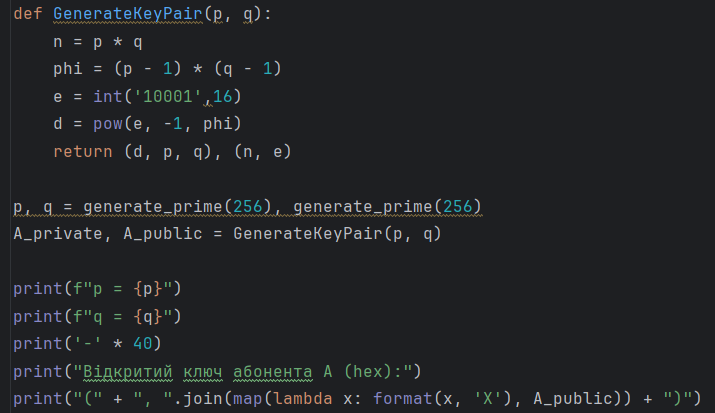


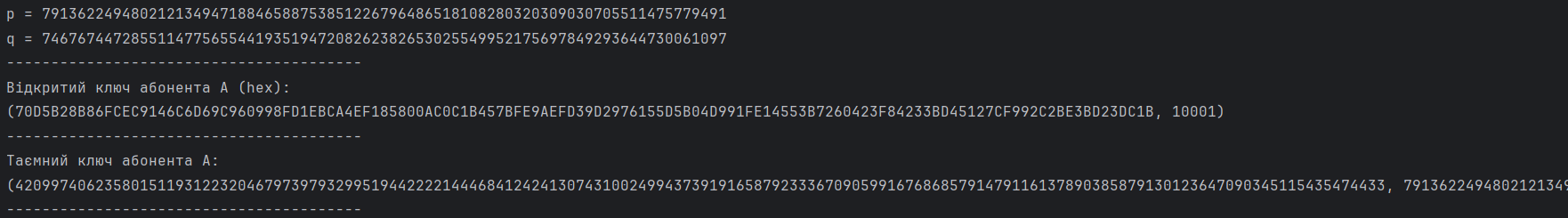
Записуємо його в код і шифруємо:  


Результат:  
  


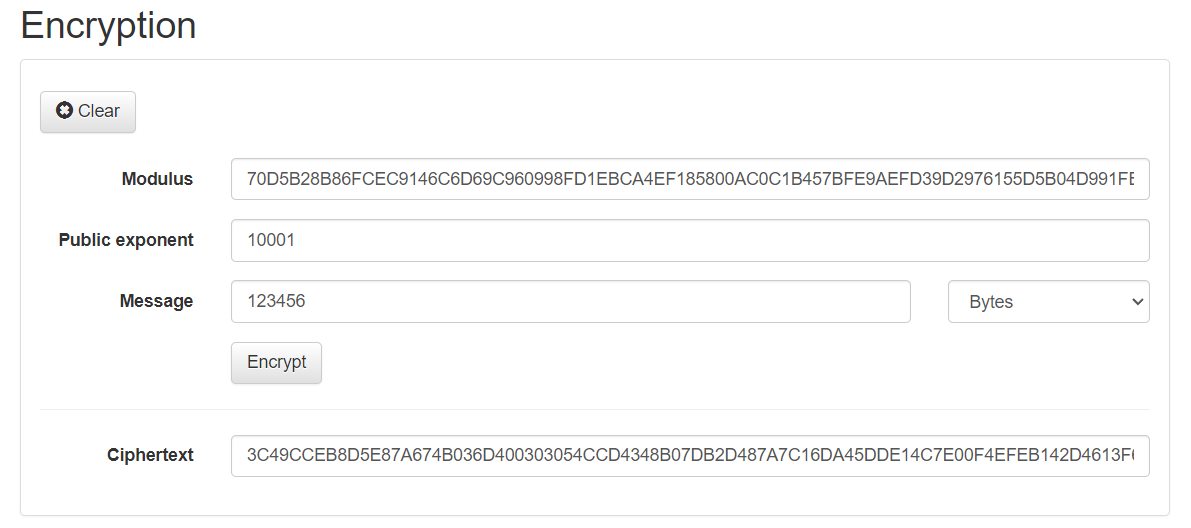
**Перевірка дешифрування.**

Згенеруємо ключ:

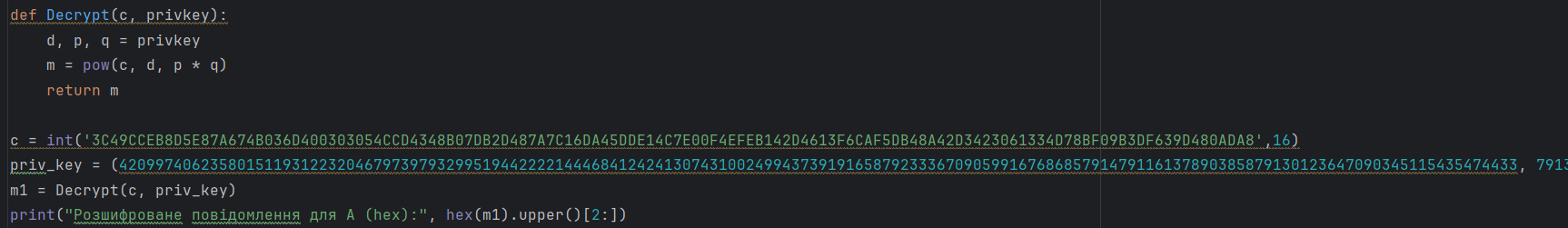




Введемо його на сервер:

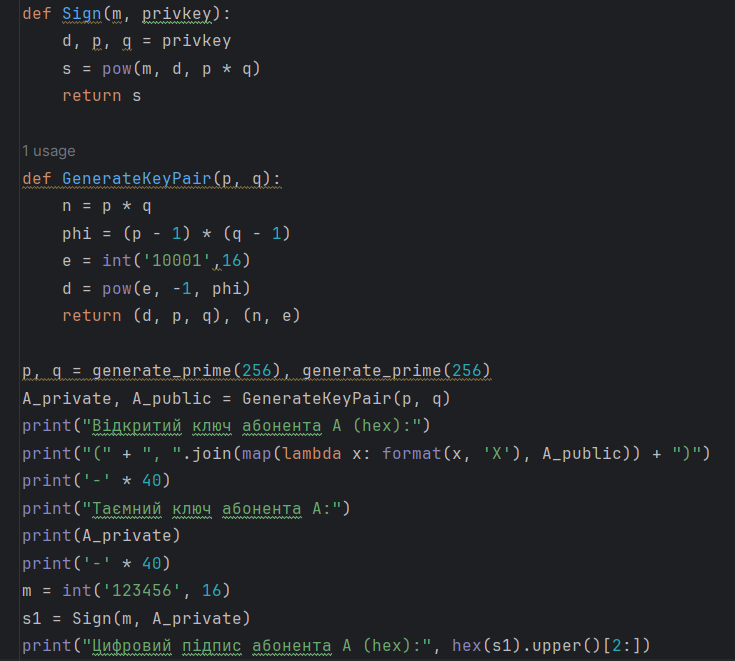
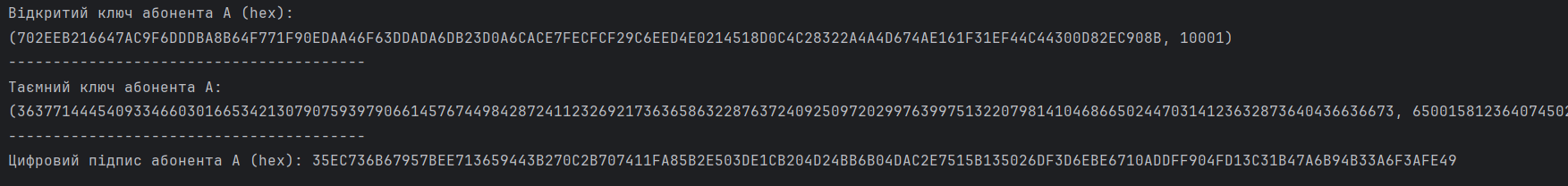


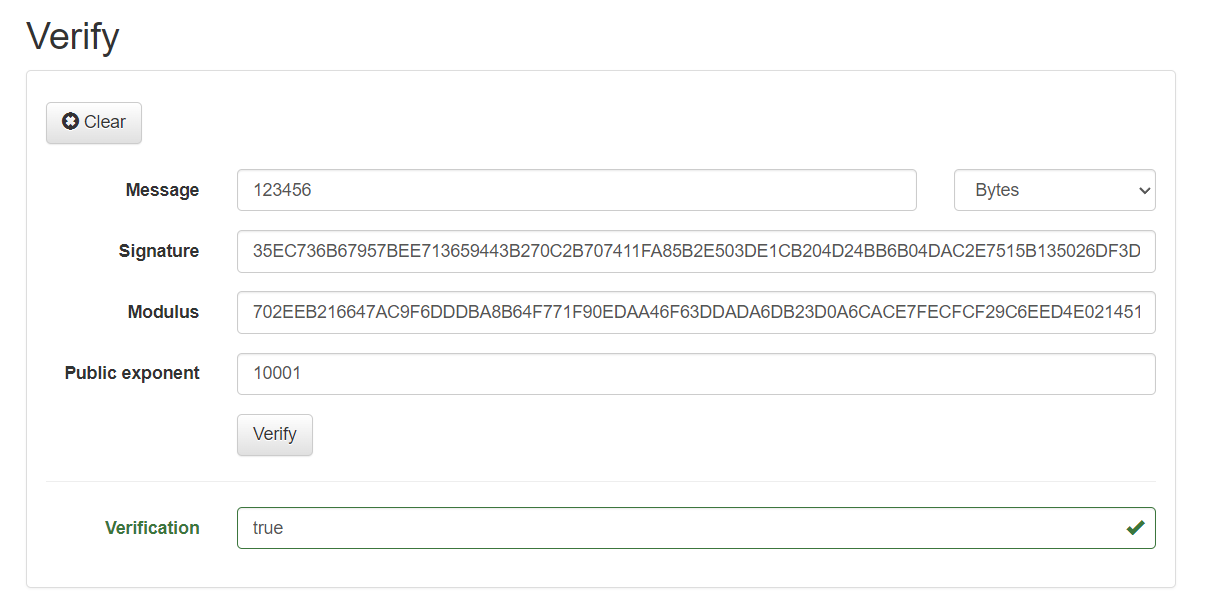
Дешифруємо результат:



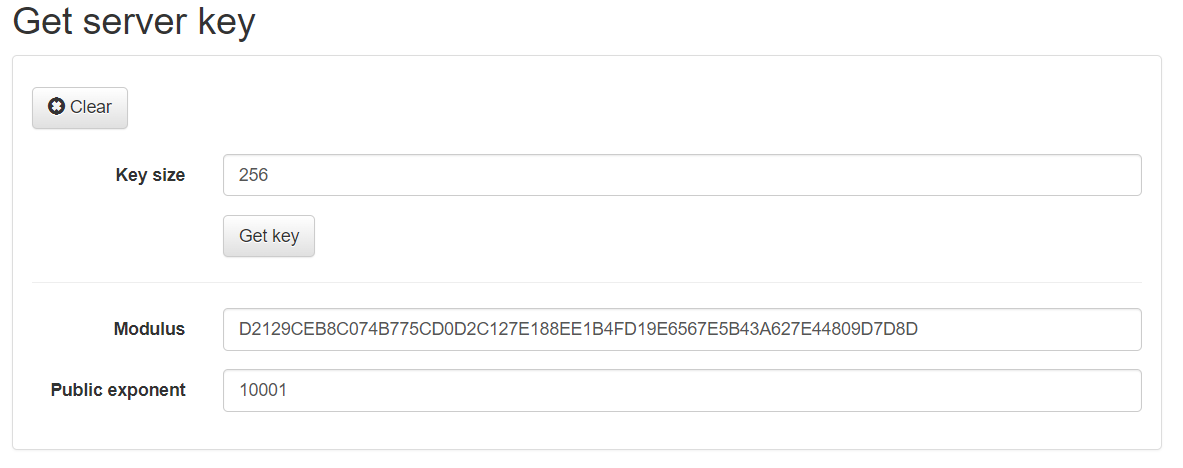


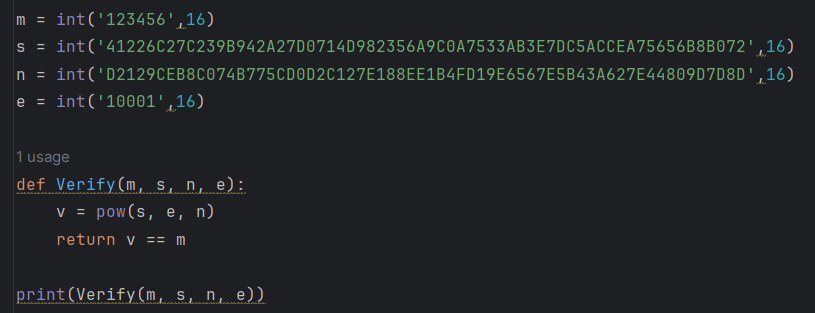
**Перевірка підпису.**

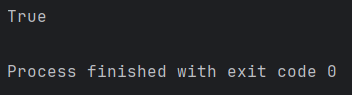
Підпишемо повідомлення:  
  




**Перевірка верифікації.**

Генеруємо ключ та підписуємо повідомлення:  
  



**Висновок**

У ході виконання даного практикуму було реалізовано основні алгоритми та процедури криптосистеми RSA - генерацію ключів, шифрування, розшифрування, підпис та верифікацію. Також було продемонстровано застосування RSA для безпечного обміну ключами між абонентами. Робота дозволила отримати практичні навички використання асиметричних криптосистем.