Міністерство освіти і науки України Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського» Фізико-технічний інститут

Лабораторна робота №4

З предмету «Криптографія»

На тему

«криптосистеми RSA та алгоритму електронного підпису»

Виконали: Студенти групи ФБ-83,84

Мельниченко А.

Іванченков М.

Перевірив:

Чорний О.М

**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.*

Генерацію числа з перевіркою на простоту з використанням тесту Міллера – Рабіна реалізовано методом *random\_s().* Допоміжні *is\_Prime()* та *trial\_composite()*.



Метод генерує прості та використовує тест Мілера-Рабіна. Повертається перше число, яке пройшло тест. Метод *is\_Prime()* виконує повторні перевірки не більше аніж разів, де – число, що перевіряється.

**2.** *За допомогою цієї функції згенерувати дві пари простих чисел*

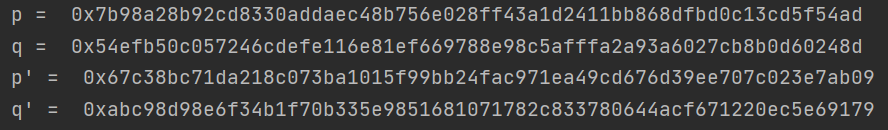
*довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб*

*- прості числа для побудови ключів абонента А та абонента B.*

Реалізовано функцією *pair\_generator()*.

Для генераціє чисел функція використовує *random\_s(),* описаний в попередньому пункті. Додатково перевіряється умова (\*), а *check\_pairs()* забезпечує відсутність повторів в парах.

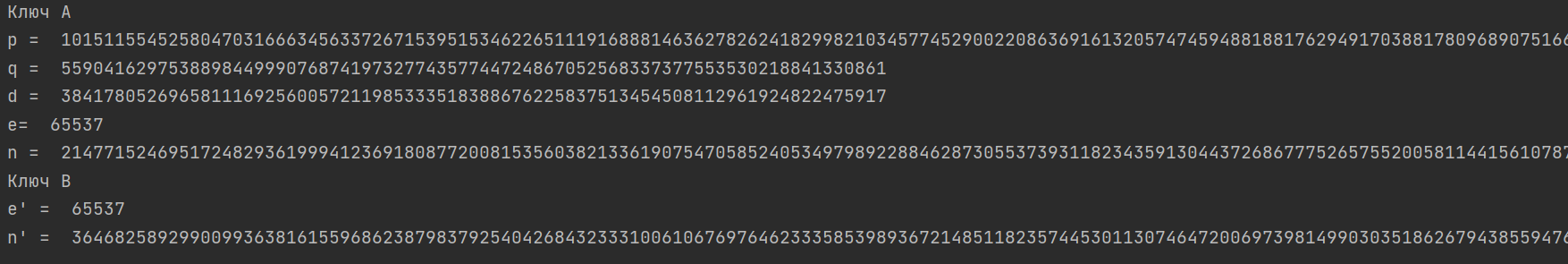
В результаті роботи методу згенеровано наступні пари ключів



Разом з консольним виводом ключі записано у файл для можливості використати у подальших операціях генерації ключів та шифрування.

**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 та секретні .*

Реалізовано функцією *gen\_key(p, q)*. Сгенеровані ключі для А та відкритий ключ B показано на рисунку

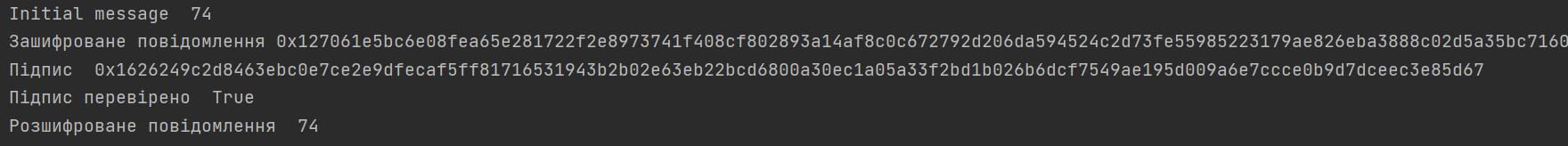


В якості використовується рекомендоване

**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.*

Згідно вказівок до завдання шифрування та дешифрування реалізовано функціями *encrypt(M, key)* та *decrypt(M, key).* За генерування цифрового підпису відповідає функція *sign()*. Перевірка правильності підпису здійснюється методом *verify()*. У якості параметрів методу передаються повідомлення, згенерований підпис та відкриті ключі абонента A.

В якості повідомлення генерується випадкове число M з діапазону (0, 255). Такий діапазон дозволяє зашифрувати усі символи базового алфавіту.



**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*

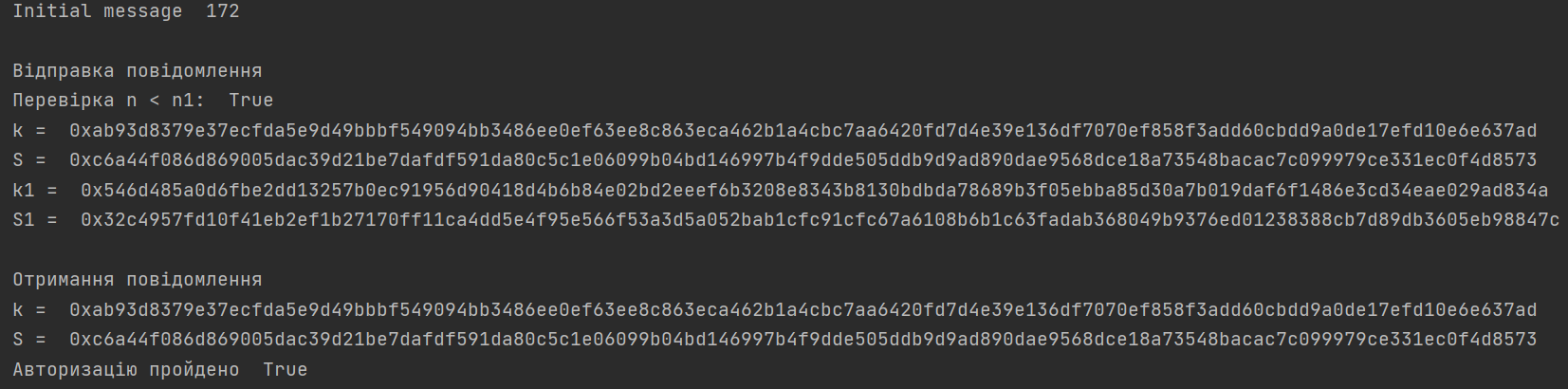
*sign(M, key)* – створення підпису;

*verify(M, S, key)* – верифікація підпису;

*sendKey(k, A\_keys, B\_key)* – відправка ключа: k – сгенероване повідомлення, A\_keys – ключі абонента А, B\_key – відкритий ключ абонента B;

*receiveKey(k, A\_key, B\_keys)* –приймання ключа: k – сгенероване повідомлення, A\_keys – ключі абонента А, B\_key – відкритий ключ абонента B.

Знимок екрану виконання програми для відправлення та отримання повідомлення



*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().*