МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електроних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

**ЗВІТ**

про виконання лабораторної роботи

з дисципліни «Основи криптографічного захисту інформації»

Виконав: здобувач 3 курсу, групи КБ-221 Регент Анастасія

Перевірив: Шелест Михайло Євгенович

НУ «Чернігівська політехніка» 2025

Лабораторна робота № 6

**Асиметрична криптографія: RSA**

**Мета роботи:** ознайомитися з алгоритмом RSA, його застосуванням для цифрового підпису та шифрування. Навчитися перевіряти автентичність підпису та шифрувати повідомлення з використанням відкритого ключа.

Для виконання роботи додані наступні файли:

lector\_pub.pem – відкритий ключ RSA лектора.

lector\_message.txt – файл з повідомленням.

lector\_signature.txt – файл з підписом RSA, згенерованим приватним ключем лектора

**Завдання № 1 Але спочатку находження оберненого елемента по модулю**

**Дано:**

число a

модуль n

**Завдання:** у відповідності з варіантом найти обернений елемент за модулем **n** (якщо він існує) або вказати, що оберненого елемента немає.

|  |  |  |
| --- | --- | --- |
| № | a | n |
| 17 | 176 | 2027 |

**Інструкція:**

Використати розширений алгоритм Евкліда для знаходження оберненого елемента за модулем n.

Якщо gcd(a, n) ≠ 1, то оберненого елемента не існує.

Перевірити відповідь: має виконуватися a⋅ ≡ 1(mod n).

Використовувати Python або ручний розрахунок.

**Ручний розрахунок**

Потрібно знайти обернений елемент тобто таке число , для якого:

|  |  |
| --- | --- |
|  | (1.1) |

Перевіримо, чи існує обернений елемент.

**Обернений елемент існує лише тоді, коли , використаємо алгоритм Евкліда:**

|  |  |
| --- | --- |
|  | (2.1) |
|  | (2.2) |
|  | (2.3) |
|  | (2.4) |
|  | (2.5) |

**Найбільший спільний дільник:** — отже, обернений елемент існує.

**Розширений алгоритм Евкліда: рухаємося вгору**

|  |  |
| --- | --- |
| Розв'язуємо: | (3.1) |
| Підставляємо : | (3.2) |
| Підставляємо 85: | (3.3) |
| Але : | (3.4) |
| Отже, | (3.5) |
| Оскільки: | (3.6) |
| то обернений елемент: | (3.7) |

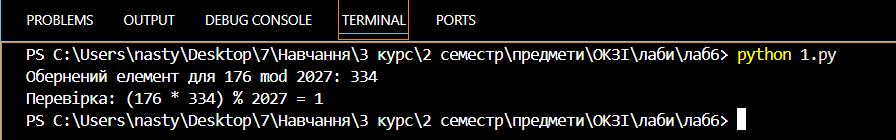


Рисунок 1 – Результат в терміналі

**Завдання № 2 Перевірка цифрового підпису**

**Мета:** перевірити, чи є підпис дійсним для наданого повідомлення, використовуючи відкритий ключ.

**Умова:**

Схема підпису: RSA-PSS

Алгоритм гешування: SHA-256

**Що зробити**:

Отримати відкритий ключ підписувача (формат PEM): task\_pub.pem.

Отримати повідомлення та відповідний підпис: task\_message.txt, task\_signature.txt.

Використати бібліотеку криптографії (наприклад, cryptography або PyCryptodome) для перевірки підпису RSA-PSS.

Визначити, чи є підпис дійсним.

**Додаткова інформація**

Для криптографічних операцій користуйтесь бібліотекою [cryptography](https://cryptography.io/).

pip install cryptography

За необхідності звертайтеся до [документації](https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/#verification) по верифікації підпису RSA бібліотеки cryptography.

Вичитати значення у збережених у файлі в форматі hex можна наступним чином:

with open("task\_message.txt", "r") as msg\_file:

message = bytes.fromhex(msg\_file.read())

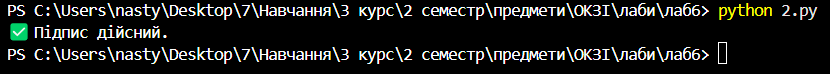


Рисунок 2 – Результат в терміналі

**Завдання № 3 Шифрування повідомлення лектору**

**Мета:** зашифрувати довільне (текстове) повідомлення, яке буде надіслано лектору.

**Умова:**

Використовуйте наданий відкритий ключ лектора task\_pub.pem.

Рекомендовано використати RSA-OAEP (Optimal Asymmetric Encryption Padding).

**Що зробити:**

Отримайте відкритий ключ лектора (public.pem).

Введіть текст повідомлення.

**Приклади повідомлень (матюками на треба** - **RSA не переварить):**

"Дякую за цікаву лекцію!"

"Чи можна зробити додаткову задачу для підвищення оцінки?"

"Я зашифрував це повідомлення власноруч"

Зашифруйте його та збережіть у файл encrypted\_message.bin.

Передати файл лектору будь-яким каналом (наприклад, через систему курсу).

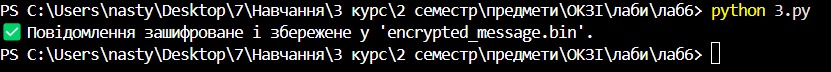


Рисунок 3 – Результат в терміналі

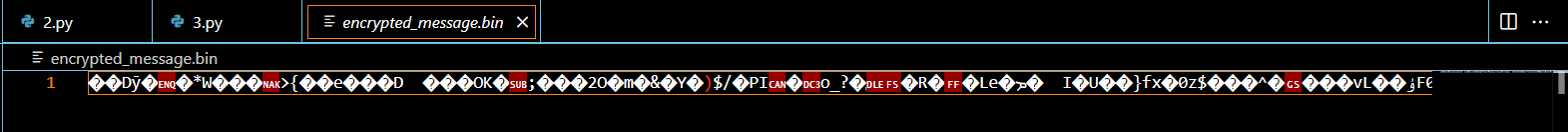


Рисунок 4 – Результат зашифрованого повідомлення

**Завдання № 4 Розшифрування повідомлення**

**Мета:** розшифрувати надіслане Вам повідомлення.

**Що зробити:**

Згенерувати власну пару RSA-ключів (відкритий + закритий ключ).

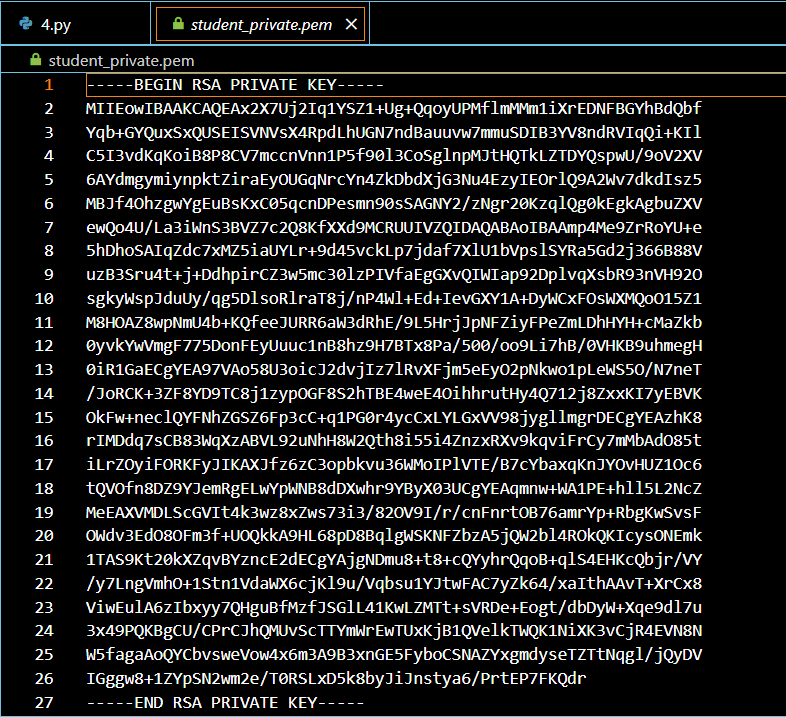


Рисунок 5 – Ключі

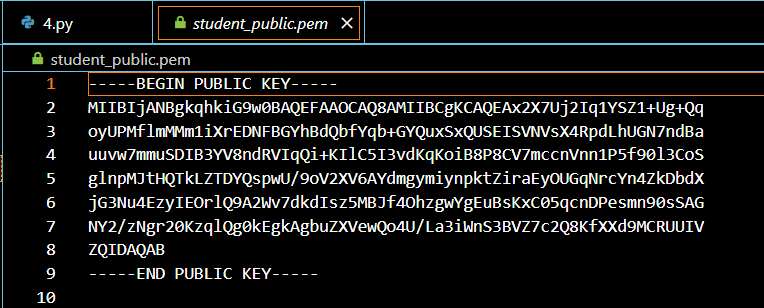


Рисунок 6 – Ключі

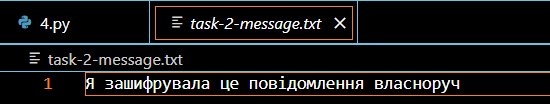


Рисунок 7 – Результат розшифрованого повідомлення

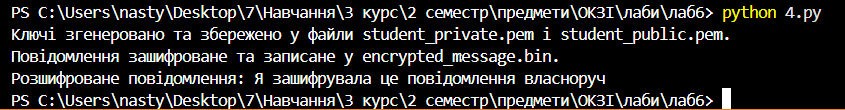


Рисунок 8 – Результат в терміналі

**Посилання на репозиторій:** [**https://github.com/Anasteishha/FCIP**](https://github.com/Anasteishha/FCIP)

**Контрольні запитання**

1. У чому принципова різниця між симетричною та асиметричною криптографією?

Принципова різниця між симетричною та асиметричною криптографією полягає в тому, що симетрична криптографія використовує один і той самий ключ для шифрування і розшифрування повідомлення, тоді як асиметрична криптографія працює з парою ключів — відкритим (публічним) ключем для шифрування і приватним (закритим) ключем для розшифрування. Відкритий ключ можна вільно розповсюджувати, а приватний зберігається в секреті.

2. Як забезпечується конфіденційність і автентичність у асиметричних системах?

Конфіденційність в асиметричних системах забезпечується тим, що повідомлення шифрується відкритим ключем отримувача і може бути розшифровано лише приватним ключем. Автентичність і цілісність забезпечуються через цифровий підпис, коли відправник підписує повідомлення своїм приватним ключем, а отримувач перевіряє підпис за допомогою відкритого ключа відправника.

3. Що таке функція Ейлера ϕ(n), як її обчислюють та її роль у криптографії RSA?

Функція Ейлера ϕ(n) визначає кількість цілих чисел, менших за n, які є взаємно простими з n. Якщо n розкладається як добуток двох простих чисел p і q, то ϕ(n) = (p-1)(q-1). У криптографії RSA ця функція використовується для вибору параметрів ключів, зокрема відкритої експоненти e і обчислення закритої експоненти d так, щоб d × e ≡ 1 (mod ϕ(n)).

4. Як відбувається генерація ключів у RSA?

Генерація ключів у RSA відбувається шляхом вибору двох великих простих чисел p і q, обчислення їх добутку n = p × q і значення ϕ(n) = (p-1)(q-1). Потім обирається відкритий ключ e, який є взаємно простим з ϕ(n), часто використовується число 65537. Далі обчислюється закритий ключ d, що задовольняє умову d × e ≡ 1 (mod ϕ(n)). Відкритий ключ — це пара (e, n), а закритий — (d, n).

5. Яка формула використовується для шифрування і розшифрування в RSA?

Для шифрування в RSA використовується формула C = M^e mod n, де M — повідомлення у числовому вигляді, а C — зашифрований текст. Для розшифрування застосовують формулу M = C^d mod n, де d — приватна експонента.

6. У чому різниця між шифруванням і цифровими підписами в RSA?

Різниця між шифруванням і цифровими підписами в RSA полягає в напрямку операції: шифрування відбувається відкритим ключем і розшифровується приватним, а цифровий підпис створюється приватним ключем і перевіряється відкритим ключем.

7. Чому використовується Padding у RSA?

Padding використовується у RSA для забезпечення безпеки, оскільки базовий алгоритм RSA уразливий до різних атак без додаткової обробки повідомлення. Padding додає випадковість і ускладнює аналіз зашифрованих даних. Наприклад, у шифруванні використовується схема OAEP, яка додає випадкові дані для захисту.

8. Чому у шифруванні використовується OAEP, а у підписах - PSS?

У шифруванні застосовується OAEP, оскільки це схема padding, оптимізована для безпечного шифрування, а для цифрових підписів використовується PSS — padding, який додає випадковість і підвищує стійкість підпису до підробок. Вони мають різні конструкції, тому застосовуються відповідно до задач.

9. Чому не можна використовувати один і той самий ключ RSA для підпису та шифрування одночасно?

Не можна використовувати один і той самий ключ RSA для підпису та шифрування одночасно через підвищені ризики безпеки. Різні операції мають різні вимоги, і компрометація ключа в одному випадку може поставити під загрозу і інші функції.

10. Які вразливості має RSA і як їх можна усунути?

RSA має кілька вразливостей, серед яких: атаки на голий RSA без padding, використання малих експонент, повторне використання ключів, атаки часу виконання і загроза факторизації модуля n при недостатньо великому розмірі ключів. Усунення цих вразливостей забезпечується використанням безпечних схем padding (OAEP для шифрування і PSS для підписів), застосуванням достатньо довгих ключів (2048 біт і більше), розділенням ключів для різних задач, захистом від атак часу та регулярною ротацією ключів.