**Міністерство освіти і науки України**

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Фізико-технічний інститут**

**Методи реалізації криптографічних механізмів**

Лабораторна робота №1

**Виконала:**

студентка ФІ-31мн

Панкєєва (Двоєглазова) С.Д.

**Перевірила:**

Селюх П. В.

Київ – 2024

**Лабораторна робота №1. Вибір та реалізація базових фреймворків та бібліотек**

***Мета:***

Вибір базових бібліотек/сервісів для подальшої реалізації криптосистеми.

***Завдання:***

Для другого типу лабораторних робіт ― вибір бібліотеки реалізації основних криптографічних примітивів з точки зору їх ефективності за часом та пам’яттю для різних програмних платформ.

Порівняння бібліотек OpenSSL, Crypto++, CryptoLib, PyCrypto для розробки гібридної криптосистеми під Linux платформу.

Оформлення результатів роботи. Опис функції бібліотеки реалізації основних криптографічних примітивів обраної бібліотеки, з описом алгоритму, вхідних та вихідних даних, кодів повернення. Контрольний приклад роботи з функціями. Обґрунтування вибору бібліотеки.

**Хід роботи:**

**Порівняння бібліотек**

 **OpenSSL**  
OpenSSL вважається золотим стандартом у світі криптографічного програмного забезпечення. Завдяки тому, що бібліотека написана на мові C, вона забезпечує високу продуктивність і підтримку широкого спектра алгоритмів, таких як AES, RSA, ECDSA та багато інших. OpenSSL активно використовується у промислових системах і серверних рішеннях. Однак її API може здатися складним, особливо для новачків, що може ускладнити процес розробки.

 **Crypto++**  
Crypto++ ― це сучасна бібліотека, яка написана на C++ і забезпечує кросплатформенну підтримку. Вона ідеально підходить для розробників, які працюють з об’єктно-орієнтованим підходом, оскільки має зручніший API порівняно з OpenSSL. Хоча продуктивність Crypto++ трохи поступається OpenSSL, її достатньо для більшості додатків, де важливі зручність та швидкість інтеграції.

 **CryptoLib**  
CryptoLib орієнтована на мінімалізм і простоту. Вона часто використовується у навчальних проектах або вбудованих системах, де не потрібна велика кількість алгоритмів. Її головний недолік ― обмежені можливості порівняно з більш повнофункціональними бібліотеками, такими як OpenSSL чи Crypto++.

 **PyCrypto**  
PyCrypto створена спеціально для Python-розробників, що робить її дуже популярною серед розробників веб-додатків і невеликих проектів. Її API інтуїтивно зрозумілий, що дозволяє швидко реалізовувати криптографічні функції. Проте її продуктивність обмежена через реалізацію на інтерпретованій мові, що робить її менш придатною для задач, де потрібна висока швидкість обчислень.

1. **OpenSSL**

**Основні переваги:**  
Висока продуктивність, підтримка великого спектра криптографічних алгоритмів, широка документація, підтримка багатьох платформ.

**Недоліки:**  
Складність використання для новачків, великий обсяг коду.

1. **Crypto++**

**Основні переваги:**  
Простота інтеграції з C++, модульність, доступність реалізацій новітніх алгоритмів.

**Недоліки:**  
Менша продуктивність у порівнянні з OpenSSL, слабша підтримка.

1. **CryptoLib**

**Основні переваги:**  
Легкість в освоєнні, орієнтована на початкові рішення.

**Недоліки:**  
Низька продуктивність, обмежений набір алгоритмів.

1. **PyCrypto**

**Основні переваги:**  
Інтуїтивно зрозумілий API, швидка інтеграція для Python-проектів.

**Недоліки:**  
Застарілість, низька продуктивність у порівнянні з OpenSSL.

**Вибір бібліотеки**

Для реалізації гібридної криптосистеми під платформу Windows було обрано бібліотеку OpenSSL через її:

* високу продуктивність;
* підтримку широкого набору криптографічних алгоритмів;
* стабільність та перевірену надійність.

**Опис програми :**

 **generate\_aes\_key**  
Генерує 256-бітний AES-ключ для симетричного шифрування.

 **encrypt\_with\_aes**  
Шифрує передане повідомлення у режимі CFB, що забезпечує потокове шифрування. Повертає вектор ініціалізації та зашифрований текст.

 **generate\_rsa\_keypair**  
Створює пару ключів RSA (2048 біт) для асиметричної криптографії.

 **sign\_data**  
Використовує приватний ключ для створення цифрового підпису з використанням алгоритму PSS і хеша SHA-256.

 **verify\_signature**  
Перевіряє достовірність підпису з використанням публічного ключа. У разі невідповідності повертає False.

 data = 'Важливі дані'.encode('utf-8').  
Це конвертує текст у формат байтів за допомогою кодування UTF-8.

 У рядку виводу даних для підпису додано data.decode("utf-8"), щоб відобразити текст у вихідному вигляді.

***Функції:***

**1. generate\_aes\_key**

* Ця функція генерує випадковий 256-бітний ключ для AES-шифрування (Advanced Encryption Standard). Використовується генератор випадкових чисел, що забезпечує високий рівень криптографічної стійкості.
* **Вихідні дані:** 256-бітний AES-ключ у вигляді байтів (bytes).

**2. encrypt\_with\_aes**

* Функція шифрує текстове повідомлення за допомогою алгоритму AES у режимі CFB (Cipher Feedback Mode). Режим CFB дозволяє шифрувати дані потоково, що зручно для роботи з динамічними повідомленнями. Для шифрування генерується випадковий вектор ініціалізації (IV).
* **Вхідні дані:**
  + - key (bytes): AES-ключ, згенерований функцією generate\_aes\_key.
    - plaintext (str): Текстове повідомлення, яке потрібно зашифрувати.
* **Вихідні дані:**
  + - iv (bytes): Вектор ініціалізації (16 байтів).
    - ciphertext (bytes): Зашифроване повідомлення у вигляді байтів.

**3. generate\_rsa\_keypair**

* Ця функція генерує пару ключів RSA (приватний та публічний). Приватний ключ використовується для підпису даних, а публічний ― для їхньої перевірки.
* **Вхідні дані:**  
  Немає (функція самостійно генерує ключі).
* **Вихідні дані:**
  + - private\_key (RSAPrivateKey): Приватний ключ RSA.
    - public\_key (RSAPublicKey): Публічний ключ RSA.

**4. sign\_data**

* Функція створює цифровий підпис для переданих даних. Цей підпис генерується за допомогою приватного RSA-ключа. Підпис гарантує, що дані не були змінені та підтверджує автентичність автора.
* **Вхідні дані:**
  + - private\_key (RSAPrivateKey): Приватний ключ для створення підпису.
    - data (bytes): Дані, які потрібно підписати (наприклад, текст або файл у вигляді байтів).
* **Вихідні дані:**
* signature (bytes): Згенерований цифровий підпис для переданих даних.

**5. verify\_signature**

* Функція перевіряє, чи є підпис дійсним. Для цього використовується публічний RSA-ключ. Якщо підпис відповідає даним, функція повертає True, інакше ― False.
* **Вхідні дані:**
* public\_key (RSAPublicKey): Публічний ключ для перевірки підпису.
* signature (bytes): Цифровий підпис, який потрібно перевірити.
* data (bytes): Дані, для яких був створений підпис.
* **Вихідні дані:**
  + - True, якщо підпис валідний.
    - False, якщо перевірка не вдалася.

**6. if \_\_name\_\_ == '\_\_main\_\_':**

Цей блок демонструє роботу всього коду:

**Генерація AES-ключа та шифрування:**

* Генерується AES-ключ.
* Шифрується текстове повідомлення ("Конфіденційна інформація").
* На виході ― вектор ініціалізації та зашифроване повідомлення, які друкуються у консоль.

**Генерація RSA-ключів:**

* + Генерується пара RSA-ключів (приватний і публічний).

**Підпис даних:**

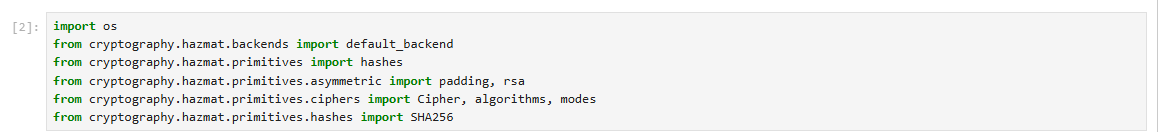
* Дані " важливі дані" підписуються приватним ключем RSA.
* Підпис друкується у консоль.

**Перевірка підпису:**

* + Публічний ключ використовується для перевірки підпису.
  + Результат перевірки (True або False) також виводиться в консоль.

Таким чином, цей код демонструє основні операції криптографії: шифрування даних, створення цифрового підпису та перевірку його дійсності.

**Код програми :**

****

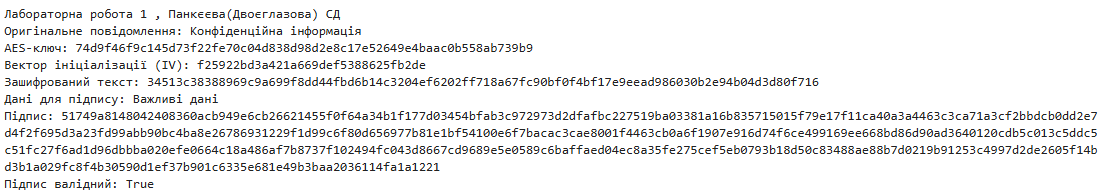
****

****

****

****

**Скрін виконання програми :**

****

**Використана література :**

1. Задірака В.К., Олексюк О.С. Комп’ютерна арифметика багаторозряних чисел: Наукове видання. – Київ: 2003. – 264 с.
2. Д. Э. Кнут Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд.: Пер. с англ.: Уч. пос.–М: Изд. дом «Вильямс», 2001.–832 с.
3. C.K.Koc High-Speed RSA Implementation TR RSA Data Security