Лабораторна робота № 4.

Мета: розробка реалізацій IT-систем у відповідності до стандартних вимог Crypto API або стандартів PKCS.

Виконали: Журибіда Ю.Б., Швець М.К., Шостак А.А.

Метою даної роботи є реалізація веб-сервісу для підпису файлів у форматі PKCS #7, використовуючи приватний ключ і публічний сертифікат, а також створення механізму для завантаження підписаного файлу через браузер.

Електронний цифровий підпис (ЕЦП) є важливим елементом в забезпеченні цілісності, аутентичності та незаперечності електронних документів. Підпис на електронному документі гарантує, що документ не було змінено після підписання і що підписане повідомлення належить особі, яка володіє відповідним приватним ключем.ро

Один із стандартів для створення підпису — PKCS #7 (який є частиною CMS — Cryptographic Message Syntax). Стандарт PKCS #7 визначає формат для створення і перевірки цифрових підписів та для упаковки зашифрованих даних. Цей формат широко використовується в різних криптографічних застосунках, таких як електронні підписи та сертифікати.

Приватний і публічний ключі:

- Приватний ключ використовується для підписування документа. Він має залишатися конфіденційним і не повинен бути переданий іншій стороні.
- Публічний ключ використовується для перевірки підпису та має бути доступний будь-якому користувачу для перевірки підпису.
- Для створення підпису хешу документа застосовується криптографічна хеш-функція (наприклад, SHA256).
- Після обчислення хешу застосовується асиметричний алгоритм шифрування (наприклад, RSA) для підпису хешу даних.

Були використані наступні алгоритми PyCryptodome та pyOpenSSL: Бібліотеки для криптографічних операцій, таких як підписування даних і створення контейнерів PKCS #7.

5. Процес підпису

- Клієнт надає файл, який потрібно підписати, приватний ключ для підпису та публічний сертифікат.
- Програмне забезпечення зчитує дані з цих файлів:
- Зчитує файл, який потрібно підписати.
- Завантажує приватний ключ та публічний сертифікат.
- Створюється хеш документу (за допомогою алгоритму SHA256).
- Підписується хеш документу за допомогою приватного ключа.
- Формується контейнер PKCS #7, який містить підпис і сертифікат публічного ключа.
- Підписаний файл повертається клієнту для завантаження.

Звіт по програмному коду

1. Опис структури коду

Код складається з кількох основних компонентів

- Імпорт необхідних бібліотек:

Flask для створення веб-сервісу.

pyOpenSSL для роботи з сертифікатами та підписами.

cryptography для виконання криптографічних операцій, таких як хешування та підпис.

- Завантаження файлів:

Програма отримує три файли через POST-запит:

1. Файл для підпису (file).
2. Приватний ключ для підпису (private_key).

Програма використовує приватний ключ для підпису хешу документа (алгоритм SHA256) за допомогою алгоритму RSA.

Формується контейнер PKCS#7, що містить підпис.

- Веб-сервіс Flask:

Веб-сервіс на Flask приймає POST-запити, обробляє файли та повертає підписаний файл для завантаження клієнтом.

- Функція для відправки файлу:

Після створення підпису, підписаний файл (формат .p7s) відправляється назад клієнту для завантаження через HTTP-запит за допомогою функції send_file.

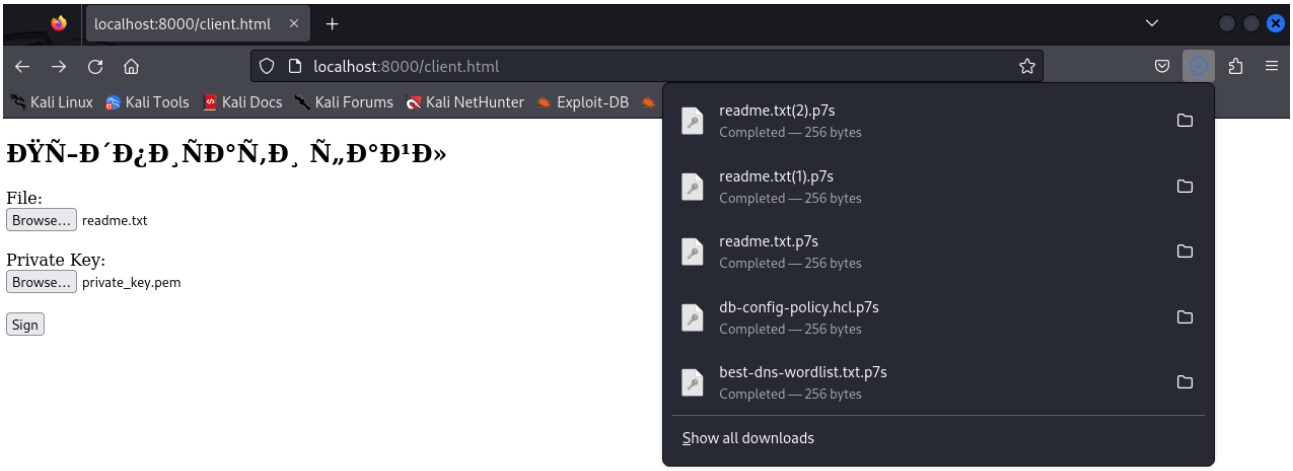
2. Основні функції коду

Завантаження приватного ключа:

```
# Завантаження приватного ключа
1 usage
def load_private_key(pem_file):
    with open(pem_file, 'rb') as key_file:
        private_key = serialization.load_pem_private_key(
            key_file.read(),
            password=None,
            backend=default_backend()
        )
    return private_key
```

Ця функція зчитує приватний ключ із файлу PEM для подальшого використання при підписанні.

2. Завантаження сертифіката публічного ключа:




```
1 usage
def sign_document(document_path, private_key_path):
    private_key = load_private_key(private_key_path)

    with open(document_path, 'rb') as document:
        document_data = document.read()

    # Підписування документа
    signature = private_key.sign(
        document_data,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )

    # Збереження підпису у форматі p7s (CMS)
    signed_file_path = f'{document_path}.p7s'
    with open(signed_file_path, 'wb') as p7s_file:
        p7s_file.write(signature)

    return signed_file_path
```