Чалий Олексій, ФБ-21 мн
Реалізація алгоритмів генерації ключів гібридних криптосистем.

Лаб 3

Варіант 3А Реалізація Web-сервісу електронного цифрового підпису.

Оформлення результатів роботи. Критерії для оформлення не задані

Зміст

Загальні відомості

Для реалізації було використана віртуальна система VirtualBox 6.1 та віртуальна машина Kali Linux 2023.3. В якосты серверу було використано Flask. Також була використана бібліотека cryptography. Також був використаний openssl для створення приватного та публічного ключа.

openssl genpkey -algorithm RSA -out private_key.pem

openssl rsa -pubout -in private_key.pem -out public_key.pem

Був створений веб сервіс з мінімальною загрузкою системи. Для створення була використана мова програмування Python 3

Код

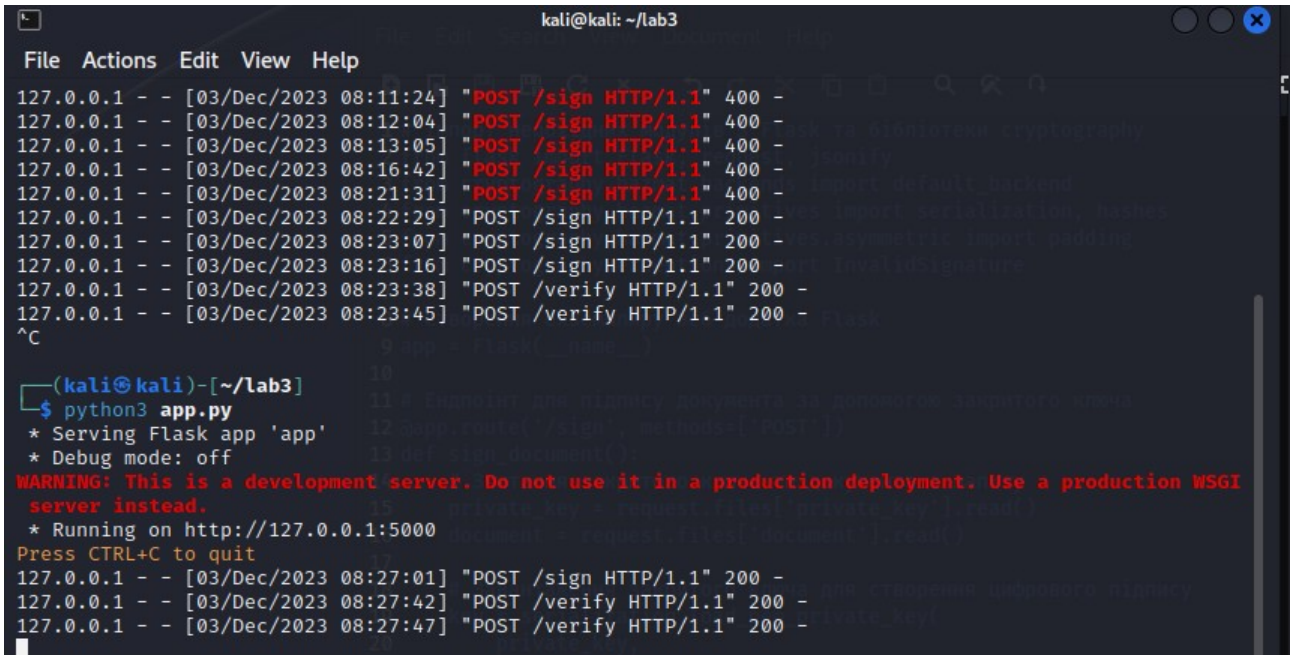
```
# Імпорт необхідних модулів з Flask та бібліотеки cryptography
from flask import Flask, request, jsonify
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.exceptions import InvalidSignature

# Створення екземпляру веб-додатка Flask
```

```
app = Flask(__name__)

# Ендпоінт для підпису документа за допомогою закритого ключа
@app.route('/sign', methods=['POST'])
def sign_document():
    # Зчитання закритого ключа та документа з запиту
    private_key = request.files['private_key'].read()
    document = request.files['document'].read()

    # Завантаження закритого ключа для створення цифрового підпису
    key = serialization.load_pem_private_key(
        private_key,
        password=None,
        backend=default_backend()
    )

    # Створення цифрового підпису для документа
    signature = key.sign(
        document,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )

    # Повернення цифрового підпису у вигляді відповіді JSON
    return jsonify({'signature': signature.hex()})

# Ендпоінт для перевірки підпису документа за допомогою відкритого ключа
@app.route('/verify', methods=['POST'])
def verify_signature():
```

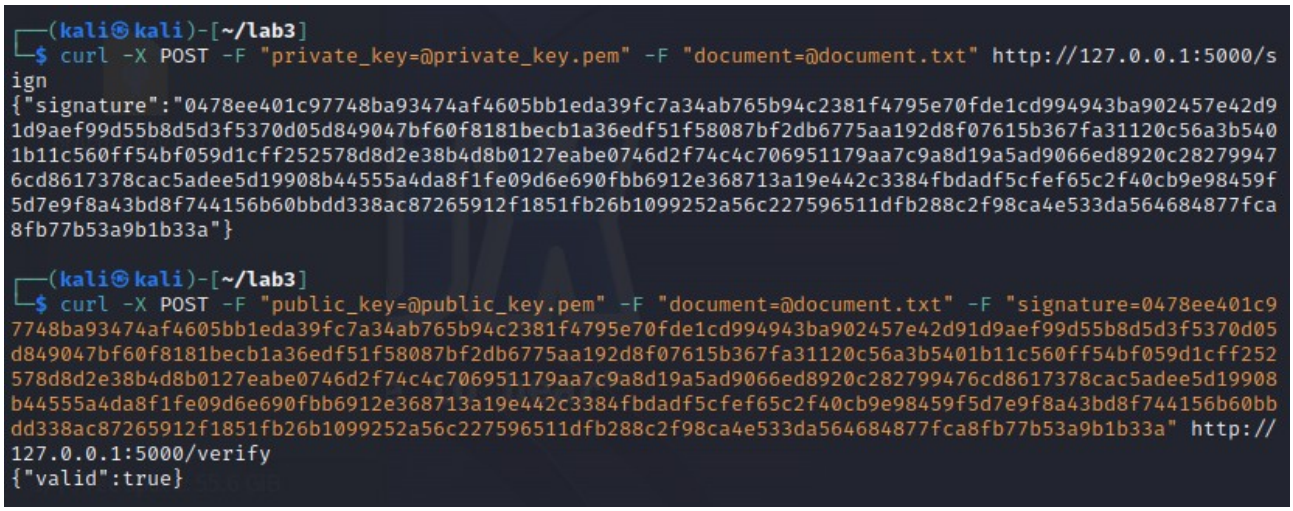
```
    # Зчитання відкритого ключа, документа та підпису з запиту
    public_key = request.files['public_key'].read()
    document = request.files['document'].read()
    signature = bytes.fromhex(request.form['signature'])

    # Завантаження відкритого ключа для перевірки цифрового підпису
    key = serialization.load_pem_public_key(
        public_key,
        backend=default_backend()
    )

    try:
        # Перевірка цифрового підпису документа
        key.verify(
            signature,
            document,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        # Повернення відповіді JSON, що вказує на валідність підпису
        return jsonify({'valid': True})
    except InvalidSignature:
        # Повернення відповіді JSON, що вказує на невалідність підпису
        return jsonify({'valid': False})

# Запуск веб-додатка Flask при виконанні цього скрипта
if __name__ == '__main__':
    app.run()
```

Результат

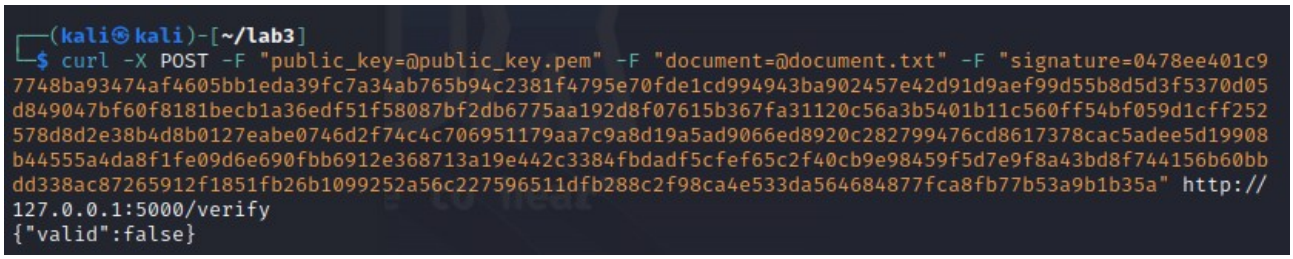
Запуск сервера



Створення сигнатури та перевірка її



Перевірка неправильної сигнатури



Висновки

В результаті, був створений простий веб сервіс для створення та перевірки електронного підпису. Перевірино правильність виконання коду, а також перевірено можливість створення електронного підпасу, а також можливість її перевірки. Також перевірена подія, коли сигнатури не співпадають.