

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.....	1
Код.....	1
Результат.....	2
Висновки.....	2

Загальні відомості

Для реалізації було використана віртуальна система VirtualBox 6.1 та віртуальна машина Kali Linux 2023.3. В якості 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

```
kali@kali: ~/lab3
File Actions Edit View Help
127.0.0.1 - - [03/Dec/2023 08:11:24] "POST /sign HTTP/1.1" 400 -
127.0.0.1 - - [03/Dec/2023 08:12:04] "POST /sign HTTP/1.1" 400 -
127.0.0.1 - - [03/Dec/2023 08:13:05] "POST /sign HTTP/1.1" 400 -
127.0.0.1 - - [03/Dec/2023 08:16:42] "POST /sign HTTP/1.1" 400 -
127.0.0.1 - - [03/Dec/2023 08:21:31] "POST /sign HTTP/1.1" 400 -
127.0.0.1 - - [03/Dec/2023 08:22:29] "POST /sign HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:23:07] "POST /sign HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:23:16] "POST /sign HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:23:38] "POST /verify HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:23:45] "POST /verify HTTP/1.1" 200 -
^C
(kali@kali)~[~/lab3]
$ python3 app.py
* Serving Flask app 'app'
* Debug Mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [03/Dec/2023 08:27:01] "POST /sign HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:27:42] "POST /verify HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2023 08:27:47] "POST /verify HTTP/1.1" 200 -
```

Створення сигнатури та перевірка її

```
(kali@kali)~[~/lab3]
$ curl -X POST -F "private_key=@private_key.pem" -F "document=@document.txt" http://127.0.0.1:5000/sign
{"signature": "0478ee401c97748ba93474af4605bb1eda39fc7a34ab765b94c2381f4795e70fde1cd994943ba902457e42d91d9aef99d55b8d5d3f5370d05d849047bf60f8181becb1a36edf51f58087bf2db6775aa192d8f07615b367fa31120c56a3b5401b11c560ff54bf059d1cff252578d8d2e38b4d8b0127eabe0746d2f74c4c706951179aa7c9a8d19a5ad9066ed8920c282799476cd8617378cac5adee5d19908b44555a4da8f1fe09d6e690fbb6912e368713a19e442c3384fbdadf5cfef65c2f40cb9e98459f5d7e9f8a43bd8f744156b60bbdd338ac87265912f1851fb26b1099252a56c227596511dfb288c2f98ca4e533da564684877fca8fb77b53a9b1b33a"}

(kali@kali)~[~/lab3]
$ curl -X POST -F "public_key=@public_key.pem" -F "document=@document.txt" -F "signature=0478ee401c97748ba93474af4605bb1eda39fc7a34ab765b94c2381f4795e70fde1cd994943ba902457e42d91d9aef99d55b8d5d3f5370d05d849047bf60f8181becb1a36edf51f58087bf2db6775aa192d8f07615b367fa31120c56a3b5401b11c560ff54bf059d1cff252578d8d2e38b4d8b0127eabe0746d2f74c4c706951179aa7c9a8d19a5ad9066ed8920c282799476cd8617378cac5adee5d19908b44555a4da8f1fe09d6e690fbb6912e368713a19e442c3384fbdadf5cfef65c2f40cb9e98459f5d7e9f8a43bd8f744156b60bbdd338ac87265912f1851fb26b1099252a56c227596511dfb288c2f98ca4e533da564684877fca8fb77b53a9b1b33a" http://127.0.0.1:5000/verify
{"valid":true}
```

Перевірка неправильної сигнатури

```
(kali@kali)~[~/lab3]
$ curl -X POST -F "public_key=@public_key.pem" -F "document=@document.txt" -F "signature=0478ee401c97748ba93474af4605bb1eda39fc7a34ab765b94c2381f4795e70fde1cd994943ba902457e42d91d9aef99d55b8d5d3f5370d05d849047bf60f8181becb1a36edf51f58087bf2db6775aa192d8f07615b367fa31120c56a3b5401b11c560ff54bf059d1cff252578d8d2e38b4d8b0127eabe0746d2f74c4c706951179aa7c9a8d19a5ad9066ed8920c282799476cd8617378cac5adee5d19908b44555a4da8f1fe09d6e690fbb6912e368713a19e442c3384fbdadf5cfef65c2f40cb9e98459f5d7e9f8a43bd8f744156b60bbdd338ac87265912f1851fb26b1099252a56c227596511dfb288c2f98ca4e533da564684877fca8fb77b53a9b1b33a" http://127.0.0.1:5000/verify
{"valid":false}
```

Висновки

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