НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря

СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №3

З дисципліни «Методи реалізації криптографічних механізмів»

«Реалізація основних асиметричних криптосистем»

Виконав:

студент групи ФБ-41мн

Білик Д. М.

Варіант: Бібліотека PyCryptodome під Linux платформу. Стандарт ECDSA.

1 Мета роботи

Дослідження можливостей побудови загальних та спеціальних криптографічних протоколів за допомогою асиметричних криптосистем.

2 Хід роботи

2.1 Асиметрична криптографія

Асиметричні криптосистеми використовують пару ключів:

- закритий ключ (private key) — зберігається в секреті, застосовується для підпису/розшифрування;
- відкритий ключ (public key) — може бути поширений, застосовується для перевірки підпису/шифрування.

Ключова перевага: відсутність необхідності передавати секретний ключ каналом зв’язку.

2.2 Цифровий підпис

Цифровий підпис забезпечує:

- автентичність (хто підписав),
- цілісність (повідомлення не змінене),
- невідмовність (підписант не може заперечити факт підпису, за коректної інфраструктури ключів).

2.3 ECDSA

ECDSA — алгоритм цифрового підпису на еліптичних кривих. Його безпека базується на складності задачі дискретного логарифмування в групі точок еліптичної кривої (ECDLP).

Загальна схема:

1. Обчислюється хеш повідомлення e = H(m) (наприклад, SHA-256).
2. Використовуючи закритий ключ і випадкове одноразове число k формується підпис (r, s).
3. Перевірка підпису виконується відкритим ключем: якщо підпис відповідає повідомленню та ключу — перевірка успішна.

3 Хід роботи

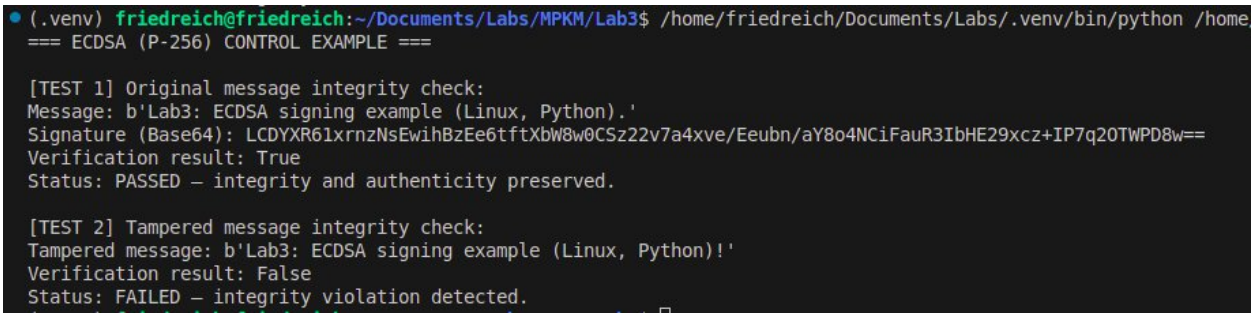
Було реалізовано програму, яка генерує ключову пару ECDSA, підписує задане повідомлення та перевіряє підпис. Для хешування використовується SHA-256, а для підпису стандарт FIPS-186-3.

Було підписано тестове повідомлення та виконано перевірку підпису. Додатково проведено експеримент зі зміненим повідомленням, де перевірка підпису завершилась невдало.

```
from Crypto.PublicKey import ECC
from Crypto.Hash import SHA256
from Crypto.Signature import DSS
from base64 import b64encode, b64decode


def generate_keys(curve_name: str = "P-256"):
    private_key = ECC.generate(curve=curve_name)
    public_key = private_key.public_key()
    return private_key, public_key


def sign_message(private_key: ECC.EccKey, message: bytes) -> bytes:
    h = SHA256.new(message)
    signer = DSS.new(private_key, mode="fips-186-3")
    signature = signer.sign(h)
    return signature


def verify_signature(public_key: ECC.EccKey, message: bytes, signature:
bytes) -> bool:
    h = SHA256.new(message)
    verifier = DSS.new(public_key, mode="fips-186-3")
    try:
```

```
        verifier.verify(h, signature)
        return True
    except ValueError:
        return False


def export_keys(private_key: ECC.EccKey, public_key: ECC.EccKey):
    private_pem = private_key.export_key(format="PEM")
    public_pem = public_key.export_key(format="PEM")
    return private_pem, public_pem


def main():
    message = b"Lab3: ECDSA signing example (Linux, Python)."

    private_key, public_key = generate_keys(curve_name="P-256")

    signature = sign_message(private_key, message)

    ok = verify_signature(public_key, message, signature)

    tampered_message = b"Lab3: ECDSA signing example (Linux, Python)!"
    ok_tampered = verify_signature(public_key, tampered_message,
signature)

    priv_pem, pub_pem = export_keys(private_key, public_key)

    print("=== ECDSA (P-256) CONTROL EXAMPLE ===")
```

```
    print("\n[TEST 1] Original message integrity check:")
    print(f"Message: {message!r}")
    print(f"Signature (Base64): {b64encode(signature).decode()}")
    print(f"Verification result: {ok}")

    if ok:
        print("Status: PASSED — integrity and authenticity preserved.")
    else:
        print("Status: FAILED")

    print("\n[TEST 2] Tampered message integrity check:")
    print(f"Tampered message: {tampered_message!r}")
    print(f"Verification result: {ok_tampered}")

    if not ok_tampered:
        print("Status: FAILED — integrity violation detected.")

    with open("ecdsa_private.pem", "wt", encoding="utf-8") as f:
         f.write(priv_pem)
    with open("ecdsa_public.pem", "wt", encoding="utf-8") as f:
         f.write(pub_pem)

if __name__ == "__main__":
    main()
```


(.venv) friedreich@friedreich:~/Documents/Labs/MPKM/Lab3$ /home/friedreich/Documents/Labs/.venv/bin/python /home/
=== ECDSA (P-256) CONTROL EXAMPLE ===
[TEST 1] Original message integrity check:
Message: b'Lab3: ECDSA signing example (Linux, Python).'
Signature (Base64): LCDYXR61xrnzNsEwihBzEe6tftXbW8w0CSz22v7a4xve/Eeubn/aY8o4NCiFauR3IbHE29xcz+IP7q2OTWPD8w==
Verification result: True
Status: PASSED — integrity and authenticity preserved.
[TEST 2] Tampered message integrity check:
Tampered message: b'Lab3: ECDSA signing example (Linux, Python)!'
Verification result: False
Status: FAILED — integrity violation detected.