Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Навчально-науковий фізико технічний інститут
Кафедра інформаційної безпеки

Звіт

З практичного завдання №3
із дисципліни «Методи реалізації криптографічних механізмів»
Тема: «Реалізація основних асиметричних криптосистем»

Виконав:
Студент групи ФБ-41мн
Шерстюк А. В.
Вигрaновський М.В

Київ-2025

Реалізація асиметричної криптосистеми під Linux платформу. Стандарт ECDSA

Мета роботи полягає у створенні робочої асиметричної криптосистеми цифрового підпису на основі стандарту ECDSA. Реалізація виконується у середовищі Linux за допомогою мови Python та бібліотеки PyCryptodome, яка є сучасним і сумісним продовженням PyCrypto та містить підтримку еліптичних кривих.

Асиметрична криптосистема працює на основі двох ключів. Приватний ключ залишається у власника і використовується для створення цифрового підпису. Публічний ключ передається іншим користувачам і дає змогу перевірити автентичність підпису. Такий підхід забезпечує захист від підробки та гарантує, що підписані дані не змінюються без виявлення.

Теоретична частина ґрунтується на криптографії еліптичних кривих, що використовує алгебраїчні структури над рівняннями вигляду $y^2 = x^3 + ax + b$ у полі скінченних чисел. Операції над точками кривої утворюють групу, що дозволяє будувати односторонні функції. Множення точки G на велике випадкове число d є обчислюваною операцією, а от зворотна дія - знаходження d за відомими G і Q = dG - є практично неможливою. Саме ця математична властивість лежить в основі криптографічної стійкості ECDSA. Приватний ключ у такій системі є випадковим числом, а публічний створюється шляхом множення цього числа на базову точку кривої. Обчислити приватний ключ на основі публічного неможливо через складність задачі дискретного логарифмування.

Алгоритм ECDSA використовує геш-функцію SHA-256, яка перетворює будь-яке повідомлення на фіксований 256-бітний геш. Підпис формується не для самого тексту, а саме для його гешу. Це дає змогу працювати з довільними обсягами даних без втрати ефективності. SHA-256 гарантує незворотність, чутливість до змін і стійкість до колізій, тому навіть мінімальна зміна повідомлення повністю змінює геш і робить підпис недійсним.

Цифровий підпис ECDSA складається з двох чисел r і s. Обчислення r пов'язане з множенням точки G на випадкове число k. Значення s залежить від приватного ключа, гешу повідомлення та того самого k. Саме випадковість k робить систему безпечною, оскільки неправильне або повторне використання цього числа створює можливість математично вивести приватний ключ. Через це стандарт FIPS 186-3 висуває суворі вимоги до генерації k. У сучасних реалізаціях часто застосовується детермінований метод згідно з RFC 6979, де k обчислюється на основі приватного ключа та гешу повідомлення, але не повторюється для різних підписів.

Перевірка підпису використовує лише публічний ключ і параметри підпису r та s. Під час верифікації формується точка на еліптичній кривій, яка має відповідати

параметрам підпису, якщо підпис справжній. Якщо повідомлення змінюється навіть незначно, обчислена точка вже не відповідає значенню r, і підпис вважається недійсним. Така поведінка гарантує цілісність та незмінність даних.

Практична частина

Генерація ключової пари виконується функцією *ECC.generate(curve="P-256")* і зберігається у форматі PEM.

```
from Crypto.PublicKey import ECC

def generate_keys():
    key = ECC.generate(curve="P-256")
    with open("ecc_private.pem", "wt") as f:
        f.write(key.export_key(format="PEM"))
    with open("ecc_public.pem", "wt") as f:
        f.write(key.public_key().export_key(format="PEM"))
```

Підпис формується шляхом обчислення SHA-256 гешу повідомлення і використання стандартного режиму DSS *"fips-186-3"*.

FIPS 186-3 - це федеральний американський стандарт, який описує механізми створення та перевірки цифрових підписів. Документ видається NIST і визначає офіційні вимоги до алгоритмів DSA, ECDSA та RSA для використання в державних та високонадійних системах безпеки.

```
from Crypto.Hash import SHA256
from Crypto.Signature import DSS

def sign_message(message, private_key):
    h = SHA256.new(message)
    signer = DSS.new(private_key, "fips-186-3")
    return signer.sign(h)
```

Перевірка підпису працює з тими ж алгоритмами, але застосовує лише публічний ключ.

```
from Crypto.Hash import SHA256
from Crypto.Signature import DSS

def verify_signature(message, signature, public_key):
    h = SHA256.new(message)
    verifier = DSS.new(public_key, "fips-186-3")
```

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

Під час виконання програми користувач генерує ключі, вводить повідомлення, отримує підпис і перевіряє його коректність. Якщо повідомлення залишається незмінним, підпис вважається достовірним. Якщо ж текст навіть трохи змінюється, перевірка показує помилку. Таким чином демонструється принцип незмінності даних і однозначної прив'язки підпису до конкретного повідомлення.

```
    def demo():
        generate_keys()

        priv_key = load_private_key()
        pub_key = load_public_key()

        message_str = "Це тестове повідомлення для підпису ECDSA"
        message = message_str.encode("utf-8")
        print("\nПовідомлення:", message_str)

        signature = sign_message(message, priv_key)
        print("\nПідпис (у hex):", signature.hex())

        ok = verify_signature(message, signature, pub_key)
        print("\nПеревірка правильного підпису:", "успішна" if ok else "неуспішна")

        fake_message = "Це вже інше повідомлення".encode("utf-8")
        ok_fake = verify_signature(fake_message, signature, pub_key)
        print("Перевірка підпису з підробленим повідомленням:", "успішна" if ok_fake
else "неуспішна")


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

```
(.venv) user@ubuntu:~/kpi/mrkm25-26/lab3/Lab3_Sherstiuk_Vygranovsky$ python3 ./main.py
Ключі згенеровано і збережено у файли:
- Приватний ключ: ecc_private.pem
- Публічний ключ: ecc_public.pem

Повідомлення: Це тестове повідомлення для підпису ECDSA

Підпис (у hex): 339b24378cca23732ab875ff6ea1363f42ae1007106b144ef9e252e4dda9e7b8cd9bc9c3cb15fced243f25d673e53f1b432749cd3e28eb2f080fd5eaf7e4a5e1

Перевірка правильного підпису: успішна
Перевірка підпису з підробленим повідомленням: неуспішна
```

Висновки

Створена асиметрична криптосистема на основі ECDSA повністю відповідає вимогам стандарту FIPS 186-3 та демонструє принципи роботи цифрового підпису на еліптичних кривих. Реалізація у середовищі Linux із використанням PyCryptodome забезпечує надійність, високу швидкість та правильне математичне функціонування алгоритму. Такий підхід може бути застосований у системах захисту інформації, автентифікації користувачів та захищеного обміну даними у сучасних інформаційно-телекомунікаційних середовищах.