Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

# ЗВІТ

**з лабораторної роботи № 3-4**

**Тема: Реалізація основних асиметричних криптосистем. Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації.**

**з дисципліни «Методи реалізації криптографічних механізмів»**

**Виконали:**
Студенти ФБ-41мн
Зубко Д.Е. та Тивонюк В. І.
«1» листопада 2025 р

**КИЇВ 2025**

**Мета роботи:** Дослідження можливостей побудови загальних та спеціальних криптографічних протоколів за допомогою асиметричних криптосистем. Отримання практичних навичок побудови гібридних криптосистем.

Обрали варіант:

Для третього типу лабораторних робіт – розробка реалізацій IT-систем у відповідності до стандартних вимог Crypto API або стандартів PKCS.

Підгрупа 3А. **Реалізація Web-сервісу електронного цифрового підпису**

Оформлення результатів: контрольний приклад роботи з асиметричною криптосистемою. Приклад атаки за побічним каналом або демонстрація їх неможливості.

# Теоретичні відомості

У основі розробленого веб-сервісу лежать фундаментальні принципи асиметричної криптографії, механізм електронного цифрового підпису (ЕЦП) та переваги, що надаються технологією блокчейн. Разом ці компоненти утворюють гібридну криптосистему, що забезпечує цілісність, автентичність та невідмовність для даних у цифровому середовищі.

### 1. Асиметрична криптографія

Асиметрична криптографія, також відома як криптографія з відкритим ключем, базується на використанні пари математично пов'язаних ключів: **закритого (приватного)** та **відкритого (публічного)**.

- **Закритий ключ (Private Key):** Є строго конфіденційним і відомий лише власнику. Він використовується для створення електронного цифрового підпису та для розшифрування повідомлень, зашифрованих відповідним відкритим ключем.
- **Відкритий ключ (Public Key):** Може вільно розповсюджуватися. Він використовується для перевірки електронного цифрового підпису та для шифрування повідомлень, які зможе прочитати лише власник відповідного закритого ключа.

В контексті Ethereum, який використовується у даній роботі, адреса гаманця користувача (0x...) генерується на основі його відкритого ключа. Кожна транзакція або підписане повідомлення підтверджується закритим ключем, що робить його унікальним і захищеним.

### 2. Електронний цифровий підпис (ЕЦП)

ЕЦП — це криптографічний механізм, що дозволяє підтвердити авторство та цілісність електронного документа. У нашому проєкті він реалізований за алгоритмом ECDSA (Elliptic

Curve Digital Signature Algorithm), що є стандартом для Ethereum. Процес створення та перевірки підпису складається з наступних етапів:

1. **Гешування:** Спочатку від вихідного цифрового контенту (файлу) за допомогою криптографічної геш-функції (у нашому випадку SHA-256) обчислюється його унікальний "відбиток" фіксованої довжини — **геш**. Цей процес є незворотним, і навіть мінімальна зміна у файлі призведе до кардинально іншого гешу.
2. **Підписання:** Отриманий геш "шифрується" (підписується) за допомогою **закритого ключа** відправника. Результатом цієї операції є сам **цифровий підпис** — унікальний рядок даних.
3. **Перевірка:** Для верифікації підпису виконуються зворотні дії. Одержувач:
    - Обчислює геш отриманого файлу за тим же алгоритмом (SHA-256).
    - Використовуючи **відкритий ключ** відправника, "розшифровує" (перевіряє) отриманий підпис, щоб дістати з нього оригінальний геш.
    - Порівнює два геші (обчислений та отриманий з підпису). Якщо вони збігаються, це доводить, що:
        - **Автентичність:** Документ був підписаний саме власником закритого ключа.
        - **Цілісність:** У документ не вносилися зміни після підписання.
        - **Невідмовність:** Підписант не може відмовитися від факту підписання.

**3. Побудова гібридної криптосистеми**

Розроблена система є гібридною, оскільки ефективно поєднує два типи криптографічних перетворень:

- **Симетричний елемент (гешування):** Функція SHA-256 швидко обробляє дані будь-якого розміру, створюючи компактний дайджест.
- **Асиметричний елемент (ЕЦП):** Алгоритм ECDSA застосовується не до всього файлу, що було б повільно, а лише до його невеликого гешу, забезпечуючи криптографічну надійність підпису.

**4. Блокчейн як децентралізований нотаріус**

Використання блокчейну Ethereum як основи для веб-сервісу надає ключові переваги порівняно з централізованими системами:

- **Децентралізація:** Відсутня єдина точка відмови або контролю. Реєстр підписів зберігається на тисячах незалежних вузлів.

- **Незмінність (Immutability):** Після того як транзакція з підписом потрапляє в блок, її практично неможливо змінити або видалити. Це створює надійний, захищений від підробки часовий штамп (timestamp).
- **Прозорість та публічна верифікація:** Будь-хто, маючи адресу смарт-контракту, може звернутися до нього і перевірити будь-який зареєстрований підпис, що ми й продемонстрували у ході роботи.

**5. Демонстрація неможливості атаки побічним каналом**

Атаки побічними каналами (side-channel attacks) націлені не на математичні слабкості криптоалгоритмів, а на аналіз фізичних характеристик процесу обчислення (час виконання, споживання енергії, електромагнітне випромінювання), щоб викрасти секретний ключ.

У розробленій архітектурі **така атака на серверні компоненти системи є неможливою** з фундаментальної причини: **ізоляція приватного ключа**.

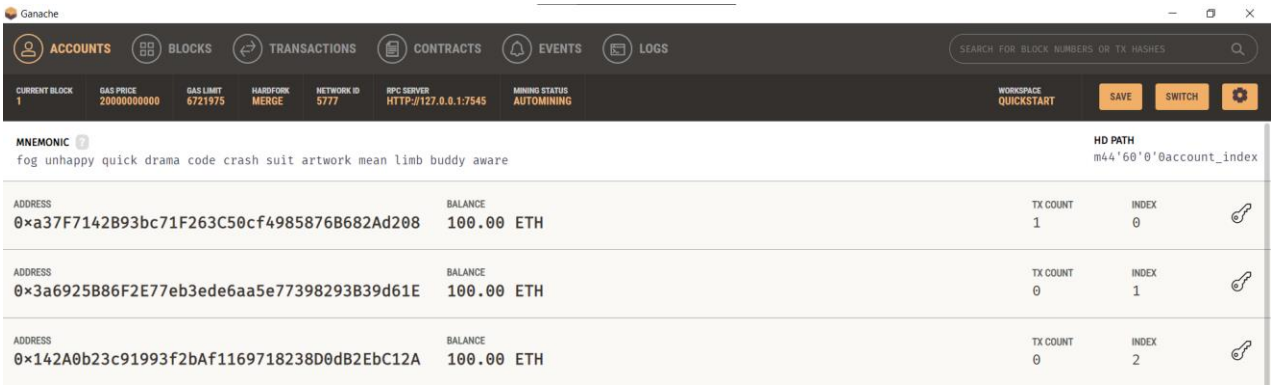
1. **Клієнтське підписання:** Критична операція — створення цифрового підпису за допомогою закритого ключа — відбувається **виключно на стороні клієнта** у захищеному середовищі браузерного гаманця MetaMask.
2. **Відсутність доступу у сервера:** Наш веб-сервер (Flask) **ніколи не отримує, не зберігає і не обробляє** приватний ключ користувача. Сервер оперує лише публічними даними: хешем файлу, готовим цифровим підписом та публічною адресою користувача.
3. **Перенесення відповідальності:** Захист від атак побічними каналами під час виконання операції підпису покладається на розробників клієнтського ПЗ (MetaMask), яке спеціально спроєктоване для безпечного зберігання та використання криптографічних ключів.

*Таким чином, архітектура системи, де ключові криптографічні операції винесені на сторону клієнта, за своєю природою є стійкою до атак побічними каналами, спрямованих на викрадення секретів з сервера.*

***Будемо реалізовано систему з трьох компонентів: смарт-контракту на Solidity для зберігання підписів, веб-серверу на Flask для логіки та клієнтського інтерфейсу з MetaMask для безпечної взаємодії користувача.***

# Хід роботи

Спочатку встановлення локального блокчейн середовища Ganache



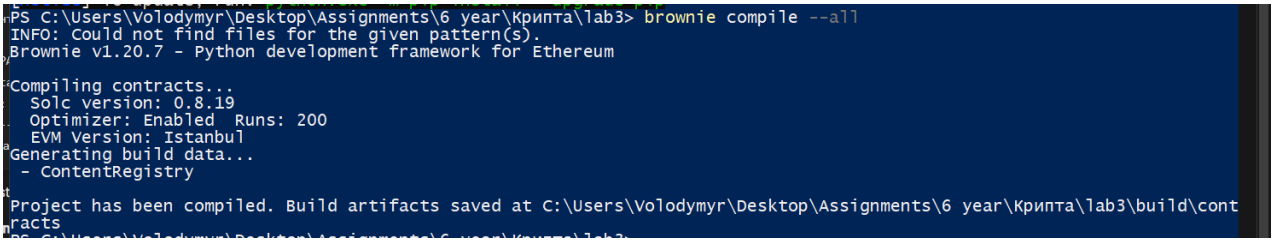

Модифікував контракт на додавання підписаного хешу

```
contract ContentRegistry {

    struct ContentInfo {
        address owner;
        uint256 timestamp;
        bytes signature; // поле для зберігання підпису
        bool exists;
    }

    mapping(bytes32 => ContentInfo) private contentRecords;

    event ContentRegistered(
        bytes32 indexed contentHash,
        address indexed owner,
        uint256 timestamp
    );
```

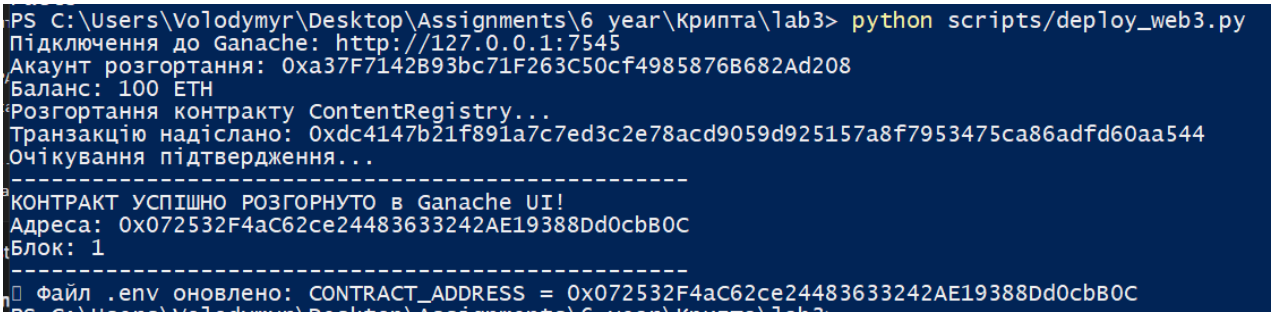
Ця функція дозволяє смарт-контракту перевірити, чи наданий цифровий підпис (_signature) дійсно був створений власником певного Ethereum-акаунта для конкретного повідомлення (_hash), відновлюючи адресу підписанта за допомогою вбудованої криптографічної операції ecrecover.

```
    // Допоміжна функція для верифікації підпису ECDSA
    function verifySignature(bytes32 _hash, bytes memory _signature) internal pure returns (address) {
        // Ethereum підписує повідомлення, додаючи префікс. Ми повинні відтворити цей хеш.
        bytes32 messageHash = keccak256(abi.encodePacked("\x19Ethereum Signed Message:\n32", _hash));

        // Розбиваємо підпис на компоненти v, r, s
        bytes32 r;
        bytes32 s;
        uint8 v;
        assembly {
            r := mload(add(_signature, 32))
            s := mload(add(_signature, 64))
            v := byte(0, mload(add(_signature, 96)))
        }

        // ecrecover повертає адресу, яка підписала повідомлення
        return ecrecover(messageHash, v, r, s);
    }
```

Скомпілював новий контракт

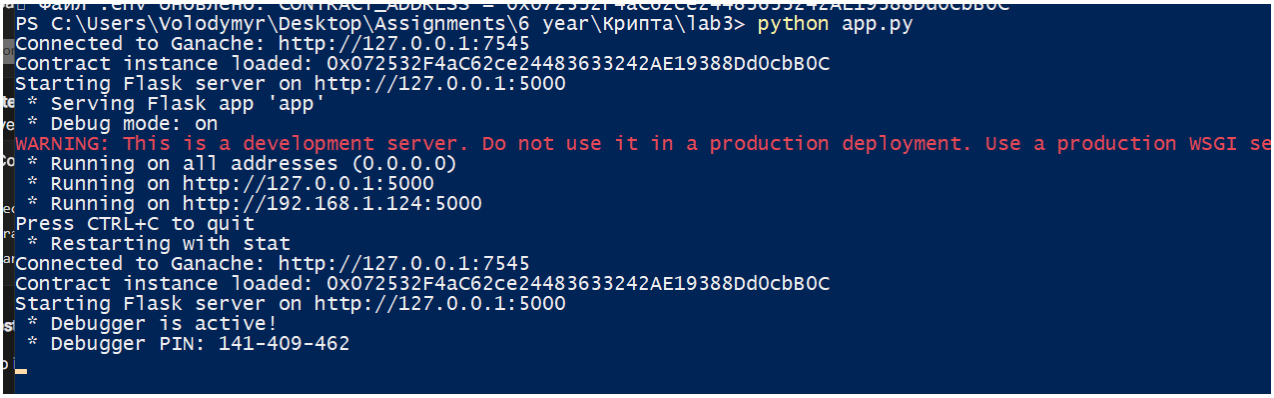



Далі скриптом деплоїмо контракт і записуємо його адресу

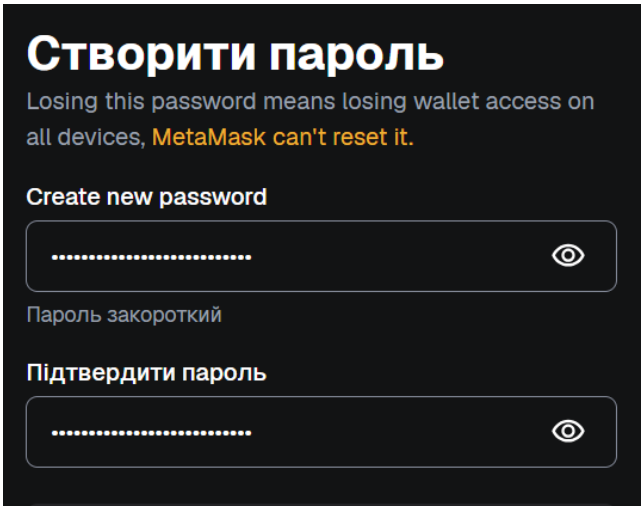


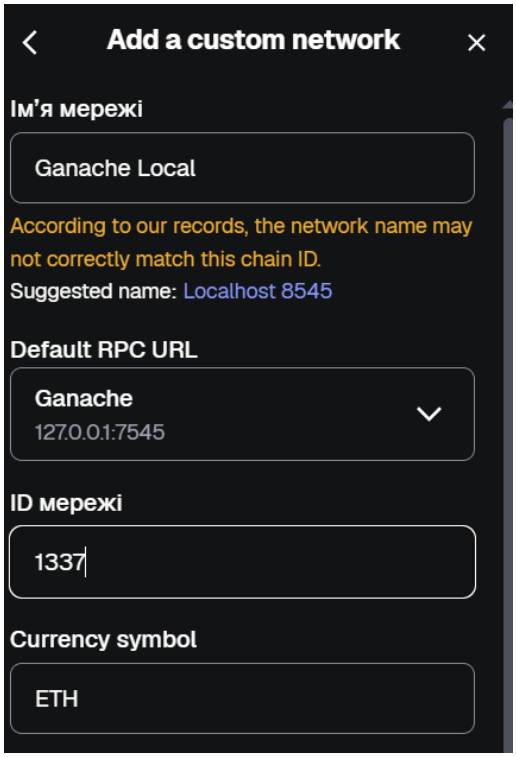


Бачимо її в ганаче і контракт: 0x072532F4aC62ce24483633242AE19388Dd0cbB0C




А це вже саме запуск фласк аппки, через яку працюватиме підпис.




Створюємо МетаМаск аккаунт, який виступає **універсальним та безпечним мостом** між вашим веб-додатком та блокчейном. Він виконує роль **менеджера ключів** користувача.

(важлива частина лабораторної)

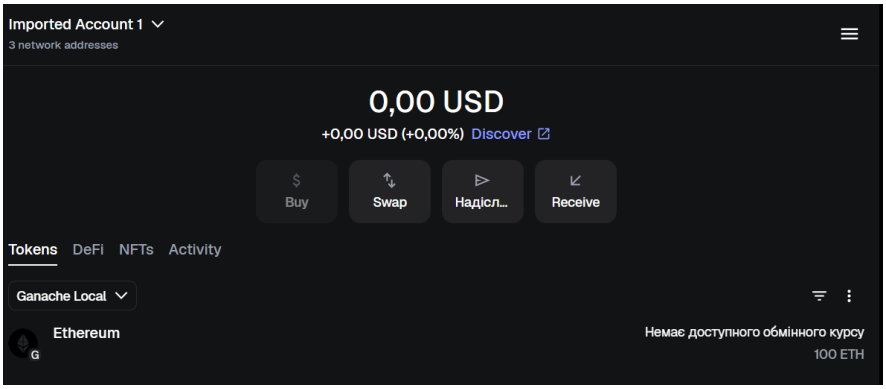
p£+77ldTi/6(wo=t[m6rF13<0EK

Ключ до гаманця:

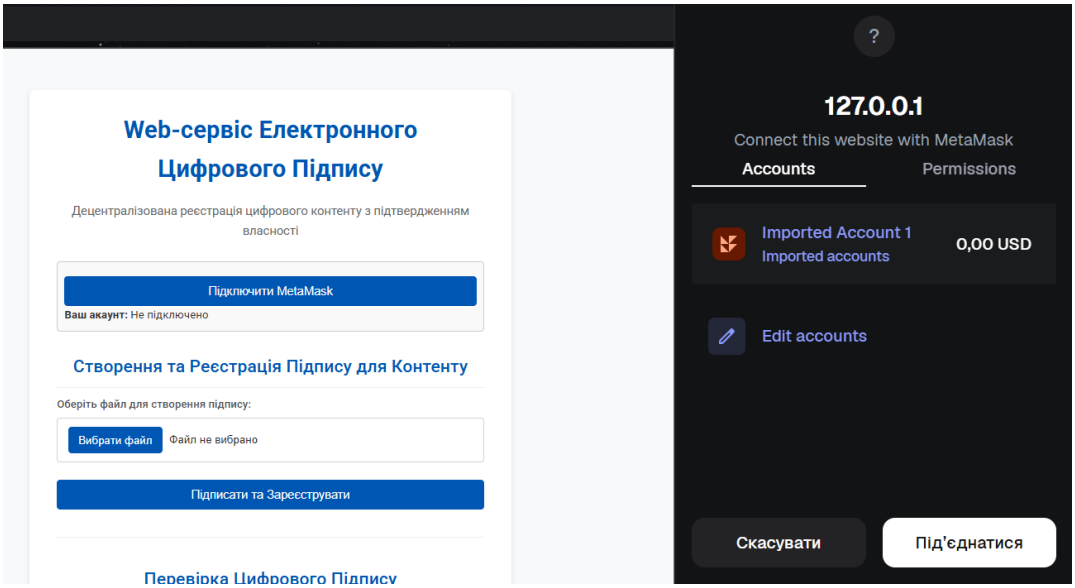
ship holiday seed digital luggage veteran water aspect blind unveil dry

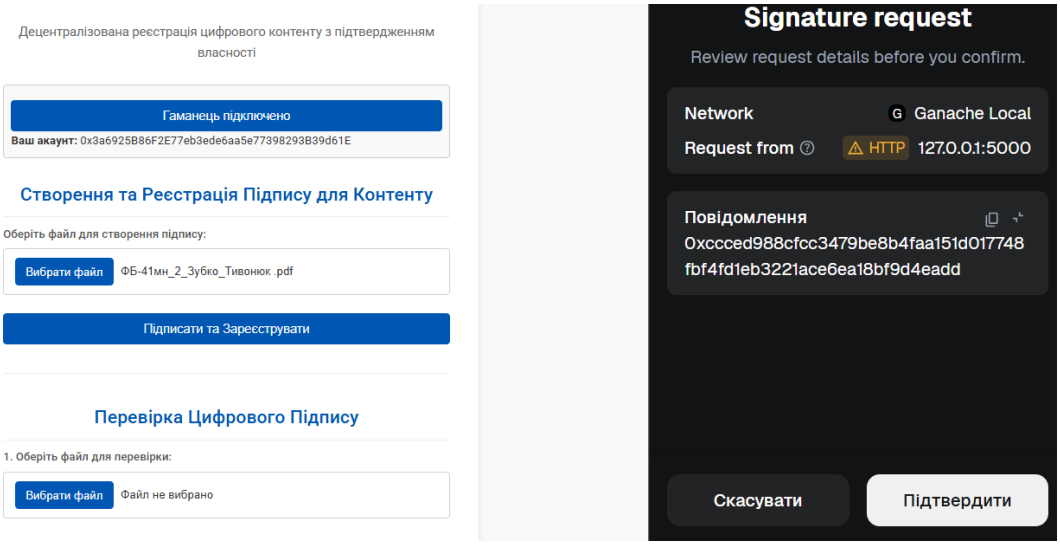
candya0e57c6527082afbfcd0e6676edf67da416befd5af4f15eadfc32b723b8acd61


Імпортуємо акаунт з ганаче



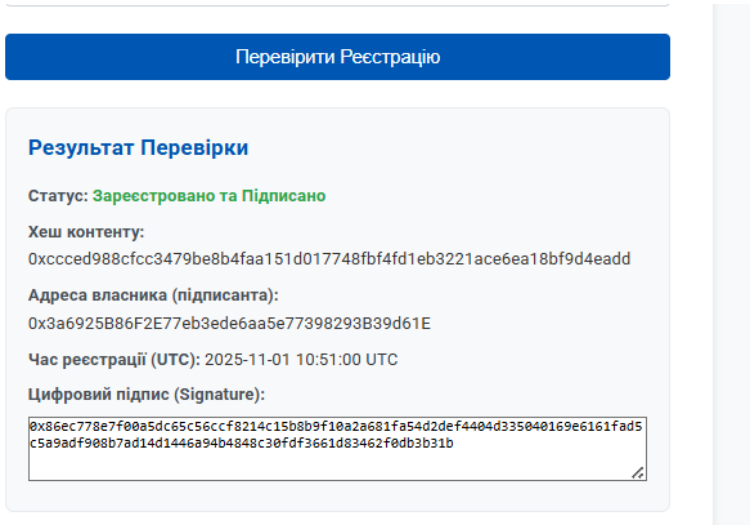

І тепер працюємо вже в веб застосунку:




Content signed and registered successfully! Tx:
0x7ee011d3a1c21b25b6c13e35cef927ff53f1af5460511b356d1d316e5f2eaf44

Перевіряємо реєстрацію:




Логфайли додатку підтвержують це:


Еджкейс підпису заново того ж файлу повертає помилку:


Отже отримали функціонуючу систему ЕЦП.

**Висновки**

У ході виконання лабораторної роботи було успішно розроблено та досліджено повнофункціональний децентралізований веб-сервіс електронного цифрового підпису на базі блокчейну Ethereum.

Маємо життєвий цикл системи: від підключення гаманця та створення підпису до його успішної реєстрації в блокчейні та подальшої публічної верифікації. Підтверджено коректну роботу логіки безпеки, зокрема, неможливість повторної реєстрації одного й того ж контенту. Також було теоретично обґрунтовано та продемонстровано архітектурну

стійкість системи до атак побічними каналами на сервері завдяки клієнтській ізоляції приватних ключів.