

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

## **ЗВІТ**

### **з лабораторної роботи № 3-4**

**Тема: Реалізація основних асиметричних крипtosистем. 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 крипtosистем. Отримання практичних навичок побудови гібридних крипtosистем.

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 крипtosистемою. 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 крипtosистему, 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 повідомень, зашифрованих відповідним відкритим ключем.
- **Відкритий ключ (Public Key):** Може вільно розповсюджуватися. Він використовується для перевірки електронного цифрового підпису та для шифрування повідомень, 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. Одержанувач:
  - 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 криptoалгоритмів, 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.

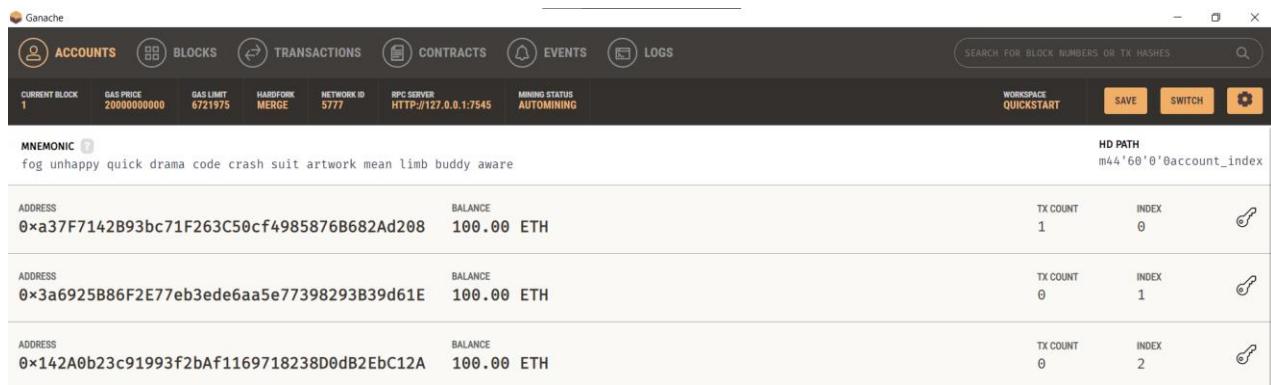
1. **Клієнтське підписання:** Критична операція — створення цифрового підпису за допомогою закритого ключа — відбувається **виключно на стороні клієнта** у захищеному середовищі браузерного гаманця MetaMask.
2. **Відсутність доступу у сервера:** Наш веб-сервер (Flask) **ніколи не отримує, не зберігає 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 підписання покладається на розробників клієнтського ПЗ (MetaMask), яке спеціально спроектоване для безпечноого зберігання та використання криптографічних ключів.

*Таким чином, архітектура системи, де ключові криптографічні операції винесені на сторону клієнта, за свою природу 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



The screenshot shows the Ganache interface with three Ethereum accounts listed:

ADDRESS	BALANCE	TX COUNT	INDEX
0xa37F7142B93bc71F263C50cf4985876B682Ad208	100.00 ETH	1	0
0x3a6925B86F2E77eb3ede6aa5e77398293B39d61E	100.00 ETH	0	1
0x142A0b23c91993f2bAf1169718238D0dB2EbC12A	100.00 ETH	0	2

HD PATH: m/44'/60'/0'/0/account\_index

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

```
PS C:\Users\Volodymyr\Desktop\Assignments\6 year\Крипта\lab3> brownie compile --all
INFO: Could not find files for the given pattern(s).
Brownie v1.20.7 - Python development framework for Ethereum

Compiling contracts...
  Solc version: 0.8.19
  Optimizer: Enabled Runs: 200
  EVM Version: Istanbul
Generating build data...
  - ContentRegistry

Project has been compiled. Build artifacts saved at C:\Users\volodymyr\Desktop\Assignments\6 year\Крипта\lab3\build\contracts
```

Далі скриптом деплоїмо контракт і записуємо його адресу

```
PS C:\Users\Volodymyr\Desktop\Assignments\6 year\Крипта\lab3> python scripts/deploy_web3.py
Підключення до Ganache: http://127.0.0.1:7545
Акаунт розгортання: 0xa37F7142B93bc71F263C50cf4985876B682Ad208
Баланс: 100 ETH
Розгортання контракту ContentRegistry...
Транзакцію надіслано: 0xdc4147b21f891a7c7ed3c2e78acd9059d925157a8f7953475ca86adfd60aa544
Очікування підтвердження...
-----
КОНТРАКТ УСПІШНО РОЗГОРНУТО в Ganache UI!
Адреса: 0x072532F4aC62ce24483633242AE19388Dd0cbB0C
Блок: 1
-----
Файл .env оновлено: CONTRACT_ADDRESS = 0x072532F4aC62ce24483633242AE19388Dd0cbB0C
```



Бачимо її в ганаче і контракт: 0x072532F4aC62ce24483633242AE19388Dd0cbB0C

```
PS C:\Users\Volodymyr\Desktop\Assignments\6 year\Крипта\lab3> python app.py
Connected to Ganache: http://127.0.0.1:7545
Contract instance loaded: 0x072532F4aC62ce24483633242AE19388Dd0cbB0C
Starting Flask server on http://127.0.0.1:5000
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.124:5000
Press CTRL+C to quit
 * Restarting with stat
Connected to Ganache: http://127.0.0.1:7545
Contract instance loaded: 0x072532F4aC62ce24483633242AE19388Dd0cbB0C
Starting Flask server on http://127.0.0.1:5000
 * Debugger is active!
 * Debugger PIN: 141-409-462
```

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

## Створити пароль

Losing this password means losing wallet access on all devices, **MetaMask can't reset it.**

**Create new password**

.....
👁

Пароль закороткий

**Підтвердити пароль**

.....
👁

Add a custom network

Ім'я мережі  
Ganache Local

According to our records, the network name may not correctly match this chain ID.  
Suggested name: [localhost 8545](#)

Default RPC URL  
Ganache 127.0.0.1:7545

ID мережі  
1337

Currency symbol  
ETH

Створюємо МетаМаск акаунт, який виступає **універсальним та безпечним мостом** між вашим веб-додатком та блокчейном. Він виконує роль **менеджера ключів** користувача.  
(важлива частина лабораторної)

pF+77ldTi/6(wo=t[m6rF13<0EK

Ключ до гаманця:

ship holiday seed digital luggage veteran water aspect blind unveil dry  
candy0e57c6527082afbfc0e6676edf67da416befd5af4f15eadfc32b723b8acd61

ACCOUNT INFORMATION

ACCOUNT ADDRESS

0x3a6925B86F2E77eb3ede6aa5e77398293B39d61E

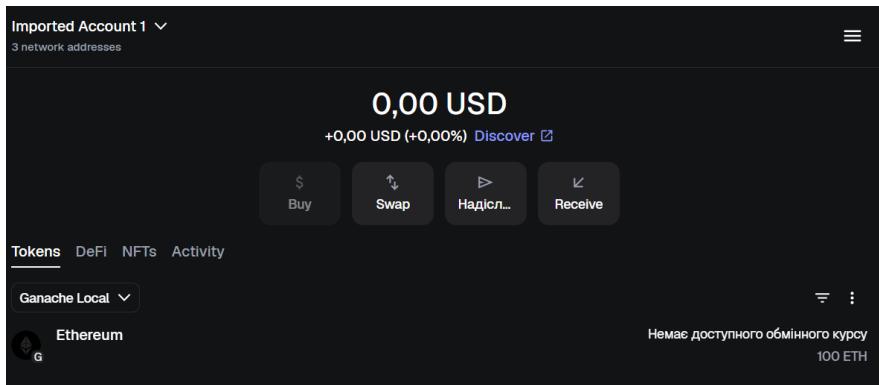
PRIVATE KEY

0xf5689e64e6c423302f5df206283f18251d2e785b3fb579b6da294aba2554e  
67

Do not use this private key on a public blockchain; use it for development purposes only!

DONE

Імпортуюмо акаунт з ганаче



І тепер працюємо вже в веб застосунку:

Two screenshots side-by-side. The left screenshot shows a web page titled "Web-сервіс Електронного Цифрового Підпису". It has a section for "Створення та Реєстрація Підпису для Контенту" with a file upload input and a "Підписати та Зареєструвати" button. Below it is a "Перевірка Цифрового Підпису" section. A blue bar at the top says "Підключити MetaMask" and another below it says "Ваш акаунт: Не підключено". The right screenshot shows the "Accounts" tab of the MetaMask extension. It displays "Imported Account 1" with a balance of "0,00 USD". There are "Edit accounts" and "Permissions" buttons at the bottom.

Three screenshots. The top one shows the same web page as before, but now with a blue bar indicating "Гаманець підключено" and "Ваш акаунт: 0x3a6925B86F2E77eb3ede6aa5e77398293839d61". The middle one is a "Signature request" dialog from MetaMask. It shows "Network: Ganache Local", "Request from: 127.0.0.1:5000", and a "Повідомлення" section with a hex string. The bottom one is a green success message box stating "Content signed and registered successfully! Tx: 0x7ee011d3a1c21b25b6c13e35cef927ff53f1af5460511b356d1d316e5f2eaf44".

Перевіряємо реєстрацію:

## Перевірити Реєстрацію

### Результат Перевірки

Статус: Зареєстровано та Підписано

Хеш контенту:

0xccced988cfcc3479be8b4faa151d017748fbf4fd1eb3221ace6ea18bf9d4eadd

Адреса власника (підписанта):

0x3a6925B86F2E77eb3ede6aa5e77398293B39d61E

Час реєстрації (UTC): 2025-11-01 10:51:00 UTC

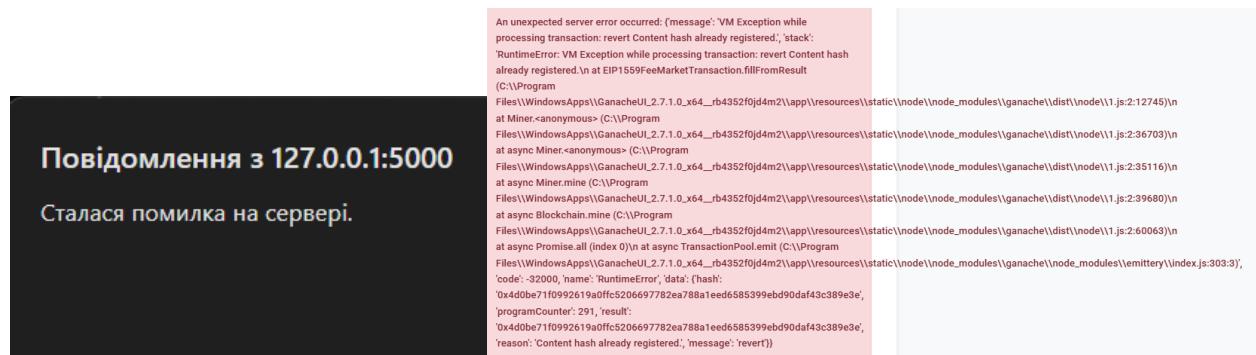
Цифровий підпис (Signature):

0x86ec778e7f00a5dc65c56ccf8214c15b8b9f10a2a681fa54d2def4404d335040169e6161fad5  
c5a9adf908b7ad14d1446a94b4848c30fd63661d83462f0db3b31b

Логфайли додатку підтверджують це:

```
Sending registerContent transaction from 0xa37F7142B93bc71F263C50cf4985876B682Ad208...
Hash: 0x88f0123388d418363f0f1a88560cdd08a920fef1403b2aef531537d7556c32a
Signature: 0x2887fa2d20ed51bc75d4d1b4f2ebc4da8819c5e533b1127350069f51964bb6052a88731a7a586d2dc2e1ef1853189070098b6f409
3c4e0c97f6c0c67a665081b
Waiting for transaction receipt: 0x52bb8a9ec31b68dd8719f5ee6081409fbe6b54433b7d89c23a5194b2ad189f6f
Transaction successful: 0x52bb8a9ec31b68dd8719f5ee6081409fbe6b54433b7d89c23a5194b2ad189f6f
127.0.0.1 - - [01/Nov/2025 11:58:31] "POST /register HTTP/1.1" 200 -
Hash for verification (input): 0xccced988cfcc3479be8b4faa151d017748fbf4fd1eb3221ace6ea18bf9d4eadd
Verification result: Owner=0x3a6925B86F2E77eb3ede6aa5e77398293B39d61E, Timestamp=1761994260
127.0.0.1 - - [01/Nov/2025 11:58:31] "POST /verify HTTP/1.1" 200 -
127.0.0.1 - - [01/Nov/2025 11:58:31] "GET /static/style.css HTTP/1.1" 304 -
Calculated hash for verification (from file): 0x88f10123388d418363f0f1a88560cdd08a920fef1403b2aef531537d7556c32a
Verification result: Owner=0xa37F7142B93bc71F263C50cf4985876B682Ad208, Timestamp=1761994711
127.0.0.1 - - [01/Nov/2025 11:58:42] "POST /verify HTTP/1.1" 200 -
```

Еджейс 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.