Лабораторна робота №1: Вступ до блокчейну та криптографічні основи

1. Теоретична частина

1.1. Концепція та визначення

Блокчейн — це децентралізована, розподілена база даних (або реєстр), що зберігає впорядкований ланцюжок записів (блоків), захищених за допомогою криптографії. Кожен блок містить криптографічний хеш попереднього блоку, мітку часу та дані транзакцій.

Ключові властивості:

1. **Децентралізація:** Відсутність центрального сервера або адміністратора. Дані зберігаються на всіх вузлах (нодах) мережі, що усуває "єдину точку відмови" (Single Point of Failure).
2. **Незмінність (Immutability):** Після запису даних у блок їх неможливо змінити без зміни всіх наступних блоків. Це забезпечується механізмом хеш-зв'язків.
3. **Прозорість:** Будь-який учасник мережі може перевірити валідність транзакцій та історію всього ланцюга.

1.2. Еволюція та Типи мереж

Розвиток технології почався з **Bitcoin (Blockchain 1.0)** як засобу передачі цінності, продовжився **Ethereum (Blockchain 2.0)** з впровадженням смарт-контрактів, і зараз еволюціонує в сторону масштабованих корпоративних рішень.

Параметр	Публічний (Public)	Приватний (Private)	Консорціум (Consortium)
Доступ	Відкритий для всіх	Тільки для дозволених осіб	Група організацій
Швидкість	Низька (через консенсус)	Дуже висока	Висока
Рівень довіри	Бездовірне середовище	Довіра до адміністратора	Довіра всередині групи
Приклад	Bitcoin, Ethereum	Hyperledger Fabric (internal)	R3 Corda, B3i

1.3. Криптографічний фундамент

Хеш-функція SHA-256

Це математичний алгоритм, що перетворює вхідні дані будь-якого розміру в рядок фіксованої довжини (256 біт).

- **Детермінованість:** Однаковий вхід завжди дає однаковий вихід.
- **Стійкість до колізій:** Практично неможливо знайти два різні входи, що дають однаковий хеш.
- **Ефект лавини:** Мінімальна зміна вхідних даних кардинально змінює хеш.

Асиметричне шифрування

Кожен користувач має пару ключів:

- **Публічний ключ:** Використовується як адреса (номер рахунку).
- **Приватний ключ:** Використовується для створення **Цифрового підпису**. Підпис доводить, що транзакція була створена власником ключа і не була змінена в процесі передачі.

1.4. Організація даних та Дерево Меркла

Блок складається з двох частин:

1. **Заголовок (Header):** Версія, хеш попереднього блоку, Merkle Root, мітка часу, Nonce (для Mining).
2. **Тіло (Body):** Список усіх транзакцій.

Дерево Меркла (Merkle Tree) — це структура даних, де кожний "листок" є хешем транзакції, а кожний батьківський вузол є хешем своїх дочірніх вузлів.

- **Merkle Root:** Фінальний хеш на вершині дерева. Він дозволяє перевірити наявність транзакції в блоці, не завантажуючи весь блок (через Merkle Proof), що критично для легких клієнтів (SPV).

2. Практична частина

Завдання 1: Реалізація базового блокчейну на Python

Вам необхідно створити спрощену модель блокчейну. Як приклад:

```
import hashlib
import time
import json
```

```
class Block:
    def __init__(self, index, transactions, previous_hash):
        self.index = index
        self.timestamp = time.time()
        self.transactions = transactions
        self.previous_hash = previous_hash
        self.nonce = 0
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        """Обчислює SHA-256 хеш вмісту блоку"""
        block_string = json.dumps({
            "index": self.index,
            "timestamp": self.timestamp,
            "transactions": self.transactions,
            "previous_hash": self.previous_hash,
            "nonce": self.nonce
        }, sort_keys=True).encode()
        return hashlib.sha256(block_string).hexdigest()

class Blockchain:
    def __init__(self):
        self.chain = [self.create_genesis_block()]

    def create_genesis_block(self):
        """Створення першого блоку в ланцюгу"""
        return Block(0, "Genesis Block", "0")

    def add_block(self, transactions):
        """Додавання нового блоку до ланцюга"""
        previous_block = self.chain[-1]
        new_block = Block(len(self.chain), transactions, previous_block.hash)
        self.chain.append(new_block)

# Приклад використання:
my_blockchain = Blockchain()
my_blockchain.add_block(["Alice sends 1 BTC to Bob"])
my_blockchain.add_block(["Bob sends 0.5 BTC to Charlie"])

for block in my_blockchain.chain:
    print(f"Block {block.index} | Hash: {block.hash} | Prev: {block.previous_hash}")
```

Завдання 2: Обчислення Merkle Root (Алгоритм)

Реалізуйте функцію, яка приймає список з 4-х транзакцій (T1, T2, T3, T4) і повертає корінь дерева.

1. Обчисліть хеші: h1 = hash(T1), h2 = hash(T2), h3 = hash(T3), h4 = hash(T4).
2. Обчисліть хеші пар: h12 = hash(h1 + h2), h34 = hash(h3 + h4).
3. Результат: Merkle_Root = hash(h12 + h34).

3. Контрольні запитання для захисту

1. У чому різниця між публічним та приватним ключем? Хто і для чого повинен їх знати?
2. Що станеться з блокчейном, якщо зловмисник змінить дані транзакції в 5-му блоці, коли в ланцюгу вже 10 блоків? Поясніть механізм ланцюгової реакції хешів.
3. Чому SHA-256 вважається "односторонньою" функцією? Чи можливо отримати вхідні дані, маючи лише хеш?
4. Як дерево Меркла допомагає мобільним гаманцям (Light Nodes) перевіряти транзакції без завантаження всього блокчейну (розміром у сотні гігабайт)?
5. Навіщо в заголовку блоку використовується мітка часу (Timestamp)?