МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Чернігівська політехніка»

Навчально-науковий інститут електроних та інформаційних технологій

Кафедра кібербезпеки та математичного моделювання

**ЗВІТ**

про виконання лабораторної роботи

з дисципліни «Основи криптографічного захисту інформації»

Виконав: здобувач 3 курсу, групи КБ-221 Регент Анастасія

Перевірив: Шелест Михайло Євгенович

НУ «Чернігівська політехніка» 2025

Лабораторна робота № 11

«**Public Key Infrastructure / TLS**»

**Мета роботи:** у цьому завданні вам необхідно реалізувати простий **Blockchain** з **Proof-of-work**, та зберегти на цьому blockchain наступні метрики:

values = [91911, 90954, 95590, 97390, 96578, 97211, 95090]

Кожне значення має зберігатись в окремому блоці.

Не зважаючи на те, що сучасні блокчейни розвинулись до надскладних структур та забезпечують безліч унікальних властивостей, основи принципу blockchain прості і можуть бути реалізовані всього кількома функціями.

**Додаткова інформація:**

**Кожен блок має містити наступні поля:**

data – дані, які ми зберігаємо.

prev\_hash – хеш значення попереднього блоку, що знаходиться на blockchain.

nonce – значення рандомізації, використане для підбору хешу необхідного формату.

hash – хеш-дайджест поточного блоку, що обчислюється з data, pref\_hash та nonce.

Для обчислення хеш використовується SHA-256.

**Реалізація Proof of Work**

Для забезпечення Proof of Work нам необхідно ускладнити генерування хешу блоку. Для цього визначимо, що згенерований хеш повинен мати x нулів на початку (в hex форматі).

У випадку, якщо x = 5 хеш-дайджест у форматі hex повинен мати наступний вигляд:

00000af39037019f4b51823c7fa014d37fc23aaa89be3555bc7823944f4b841a

Почніть з x=5, та регулюйте значення в залежності від потужності вашої системи так, щоб генерування одного блоку займало декілька секунд.

**Blockchain**

Для створення нашого блокчейну (ledger) необхідно задати складність хешування (обрати значення x), та створити "Genesis Block" – особливий блок, що є найпершим в ланцюгу та не містить prev\_hash (його можна залишити пустою строкою).

Якщо ви зберігаєте дані вже в найпершому (Genesis) блоці, важливо щоб для генерації його хешу також використовувався алгоритм Proof-of-Work. Альтернативно Genesis Block можна залишити повністю штучним і зберігати дані вже в наступних блоках.

**Рекомендована структура програми**

def hash(data, prev\_hash, nonce): функція hash, що генерує дайджест SHA-256 з набору даних data, prev\_hash, nonce.

def mine\_block(block, difficulty=x): функція mine\_block, що генерує генерує хеш-значення блоку, перебираючи значення nonce допоки результуючий дайджест не відповідатиме заданому формату (матиме достатньо кількість нулів на початку, задану x). Не забудьте записати nonce що створює хеш-дайджест необхідного формату до блоку.

Окремий блок блокчейну можна описати класом, або будь-якою структурою що міситимете усі необхідні поля блоку, описані в першій секції. Під час його створення у поле data необхідно зберегти дані, які ми хочемо записати у блокчейн.

def add\_block(blockchain, block): функція add\_block для додавання нового блоку до блокчейну. Ця функція має використовувати функцію mine\_block для створення коректного хешу для блоку та додавання його до масиву з блоками (що створюють ланцюг).

Ця структура є лише прикладом, а не вимогою – ви можете також все організувати на класах і методах, як вам зручніше.

Ось приклад коду простої реалізаціїя Blockchain з Proof-of-Work, яка задовольняє вимоги зберігання даних у блоках, забезпечує необхідний хеш з відповідною кількістю нулів на початку (для Proof-of-Work), а також додавання блоків до ланцюга.

**Формат виконання**

Напишіть свій або розберіть наданий код реалізації блокчейну та сформуйте блокчейн, що зберігає дані з масиву values. Завантажте отриманий блокчейн з вашими домашніми завданнями.



Рисунок 1 – Результат з терміналу

**Контрольні питання**

1. Що таке блокчейн і які основні принципи його роботи?

Блокчейн — це розподілений ланцюг блоків, у якому кожен блок містить дані (наприклад, транзакції), хеш попереднього блоку та свій власний хеш.

**Основні принципи:**

1. Незмінність – дані в блоках не можна змінити без зміни всього ланцюга;

2. Децентралізація – копії ланцюга зберігаються в усіх учасників мережі;

3. Консенсус – усі вузли погоджуються на єдину версію блокчейну (через алгоритми консенсусу, напр. Proof-of-Work);

4. Прозорість – дані доступні для перевірки кожному учаснику.

2. Чим Proof-of-Work відрізняється від інших механізмів консенсусу?

**Proof-of-Work (PoW):**

1. Вимагає вирішення складної обчислювальної задачі (майнінг);

2. Забезпечує безпеку, але споживає багато електроенергії.

**Відмінності від інших механізмів:**

1. Proof-of-Stake (PoS) – вибір вузла-записувача залежить від кількості монет, а не від обчислювальної потужності;

2. Proof-of-Authority (PoA) – довіра покладається на певні авторитетні вузли;

3. Proof-of-Space/Time – базується на використанні пам’яті (дискового простору) або часу.

3. Що таке Merkle Tree і яка його роль у криптовалютах?

Merkle Tree — це бінарне дерево, де листки — це хеші транзакцій, а внутрішні вузли — хеші об’єднання пар нижчого рівня.

**Роль:**

1. Дозволяє ефективно та безпечно перевіряти, чи належить транзакція певному блоку;

2. Зменшує обсяг даних, які потрібно перевіряти або передавати (наприклад, у SPV-гаманцях).

4. Як перевірити, чи належить транзакція певному блоку за допомогою Merkle Tree?

1. Знаходимо хеш транзакції;

2. Отримуємо Merkle Path — набір сусідніх хешів по шляху до кореня;

3. Обчислюємо Merkle Root;

4. Порівнюємо з Merkle Root у заголовку блоку;

5. Якщо вони збігаються — транзакція дійсно належить до блоку.

5. Що таке ієрархічний детермінований гаманець (HD Wallet), і в чому його переваги?

HD Wallet (Hierarchical Deterministic Wallet) — це гаманець, який генерує всі ключі з одного початкового seed (звичайно — мнемонічної фрази).

**Переваги:**

1. Один seed — нескінченна кількість адрес;

2. Зручне резервне копіювання;

3. Ієрархічна структура — різні рівні для управління, бірж, сервісів і т.д.

6. Як генеруються приватні та публічні ключі в HD Wallets?

1. Генерується seed з мнемонічної фрази;

2. З seed'а виводиться master private key та chain code;

3. Далі за схемою BIP-32/BIP-44 з них обчислюються дочірні приватні ключі та відповідні публічні ключі (через еліптичну криптографію) – це забезпечує детермінованість та відтворюваність.

7. Що таке мультипідписи (Multisig), і в яких ситуаціях вони використовуються?

Мультипідписи — це механізм, який вимагає кількох підписів для авторизації транзакції.

**Застосування:**

1. Спільні рахунки;

2. Гаманці з підвищеною безпекою;

3. Біржі, DAO, багатосторонні угоди.

**Переваги:**

1. Покращення безпеки (один ключ не дає доступу);

2. Захист від крадіжок або втрати одного ключа.

8. Що таке агрегація підписів і як вона покращує масштабованість системи?

Агрегація підписів — це об'єднання кількох цифрових підписів в одну (наприклад, через BLS або Schnorr).

**Переваги:**

1. Зменшення розміру блоків (одна підпис — замість багатьох);

2. Підвищення швидкості перевірки;

3. Менше навантаження на мережу.

9. Що таке commitment схема, і як вона забезпечує цілісність даних?

Commitment схема — це криптографічний механізм, що **дозволяє:**

1. "Зафіксувати" деяке значення (commit) без його розкриття;

2. Згодом «відкрити» значення (reveal), доводячи його незмінність.

**Приклад:**

commit = hash(value + random\_salt)

**Застосування:**

1. Аукціони, голосування, приватність у блокчейнах;

2. Забезпечує цілісність і прихованість до моменту розкриття.

10. Що таке докази з нульовим розголошенням (Zero-Knowledge Proofs)?

Zero-Knowledge Proof (ZKP) — це метод доведення того, що ви знаєте певну інформацію (наприклад, пароль), не розкриваючи саму інформацію.

11. У чому полягає принцип роботи ZKP і як він застосовується у криптовалютах?

**Принцип ZKP:**

1. Прокладач (prover) доводить знання певної інформації перевірячу (verifier), не розкриваючи її;

2. Базується на математичних алгоритмах, напр. zk-SNARKs, zk-STARKs;

3. **Застосування у криптовалютах:**

1. Zcash — анонімні транзакції;

2. Ethereum (в майбутньому) — приватні смарт-контракти;

3. Забезпечення конфіденційності, скалювання і верифікації без розголошення.