МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра кібербезпеки

«До захисту допущено»

Завідувач кафедри

_____ Володимир ЛЮБЧАК
_____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 125 Кібербезпека, освітньо-професійної програми Кібербезпека

на тему: «Аналіз криптографічних інструментів у контексті безпеки технології блокчейну»

Здобувача групи КБ-01 Коломійця Дениса Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Денис КОЛОМІЄЦЬ

Керівник:
старший викладач кафедри кібербезпеки,
кандидат фіз.-мат. наук
Олександр СТРАХ _____

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра кібербезпеки

«Затверджую»
Завідувач кафедри

_____ Володимир ЛЮБЧАК

«___»_____ 20___ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавр
зі спеціальності 125 – Кібербезпека, освітньо-професійної програми «Кібербезпека»
здобувача групи КБ-01 Коломієць Денис Сергійович

1. Тема роботи: «Аналіз криптографічних інструментів у контексті безпеки технології блокчейну».

затверджено наказом по СумДУ № 0212-VI від «04» березня 2024 р. зі змінами згідно Наказу № 0566-VI від «21» травня 2024 р.

2. Термін подання студентом роботи: «04» червня 2024 р.

3. Вихідні дані до роботи: _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): 1) Оцінка проблематики предметної області, визначення та формулювання цілей дослідження, 2) Огляд технологій для розробки блокчейну, 3) Розробка блокчейну, 4) Аналіз результатів.

5. Дата видачі завдання «04» березня 2024 р.

Завдання прийняв до виконання _____ Керівник _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Оцінка проблематики предметної області, визначення та формулювання цілей дослідження		
2	Огляд технологій для розробки блокчейну		
3	Розробка блокчейну		
4	Аналіз результатів		
5	Оформлення дипломної роботи		

Здобувач вищої освіти _____ Керівник _____

АНОТАЦІЯ

Кваліфікаційна робота виконана на 49 стор., 18 рис., 25 джерел.

Актуальність теми кваліфікаційної роботи обґрунтована швидким розвитком цифрових технологій та зростанням обсягу транзакцій в інтернеті, що робить питання безпеки даних надзвичайно важливим. Технологія блокчейн, будучи інноваційною і перспективною, забезпечує децентралізоване зберігання даних та прозорість, але також стикається з численними загрозами та атаками. Аналіз інструментів захисту блокчейну є критично важливим для забезпечення його надійності та безпеки.

Об'єктом дослідження виступає технологія блокчейн та її застосування в різних сферах, включаючи фінанси, логістику, охорону здоров’я та енергетику. Метою роботи є оцінка існуючих інструментів і методів захисту блокчейну, визначення їх ефективності та пропозиція можливих шляхів вдосконалення.

Методи дослідження включають теоретичний аналіз літературних джерел, практичне вивчення технологій гешування, цифрових підписів та механізмів підтвердження роботи (Proof of Work). В ході роботи також було проведено аналіз функціонування P2P-мереж та ролі майнерів у блокчейн-мережах.

Результати дослідження показали, що технологія блокчейн має значні переваги у забезпеченні безпеки даних, але також стикається з певними викликами. Практичне використання гешування, цифрових підписів та Proof of Work підтвердило їх важливість для безпеки блокчейну. Виявлені переваги та недоліки цих методів дозволили зробити висновки щодо їх ефективності та можливих напрямків удосконалення.

ЗМІСТ

ВСТУП

Уявіть собі цифрову книгу записів, де кожен аркуш (блок) надійно пов'язаний з попереднім, а записи в ній захищені від змін. Це і є блокчейн – децентралізована база даних, де інформація зберігається не в одного «господаря», а відразу в безлічі учасників мережі [1]. На відміну від звичайних баз даних, блокчейн не піддається фальсифікації. Кожен новий блок містить у собі «хеш» (унікальний відбиток) попереднього, що забезпечує прозорість і незмінність усього ланцюжка. Завдяки цьому блокчейн ідеально підходить для завдань, де потрібна абсолютна достовірність інформації, наприклад, у сфері фінансів (криптовалюти) або ланцюжків постачання.

Основними особливостями блокчейну є децентралізація, що полягає у відсутності єдиного центру управління, що підвищує надійність і безпеку системи. Прозорість забезпечується тим, що всі операції доступні для перегляду всім учасникам мережі. Незмінність означає, що записи в блокчейні неможливо змінити або видалити. Крім криптовалют, технологія блокчейн може застосовуватися в різних сферах, наприклад, в голосуванні – для забезпечення чесних і прозорих виборів, в медицині – для збереження конфіденційності пацієнтських даних, в інтелектуальній власності – для реєстрації авторських прав та захисту від плагіату, та в фінансах – для здійснення безпечних та швидких міжнародних переказів. Блокчейн – це потужний інструмент, який здатен змінити світ на краще, зробивши його прозорішим, надійнішим і справедливішим.

Однак безпеку технології блокчейну мають забезпечувати криптографічні методи захисту, які можуть зробити її стійкою до зломів. Відтак, оскільки кожен криптографічний інструмент, не лише теоретично, а й практично, все ж є вразливим до відповідних типів атак, безпека блокчейну може бути порушеною. Таким чином, обрана тема кваліфікаційної роботи є актуальною.

Об’єктом дослідження даної роботи є технологія блокчейну, а **предметом дослідження** – безпекова складова технології блокчейну.

Мета роботи: дослідити та проаналізувати криптографічні інструменти, що використовуються в технології блокчейн.

Для реалізації поставленої мети було визначено такі завдання: 1) опрацювати відповідні джерела інформації з даного предмету дослідження; 2) охарактеризувати особливості використання технології блокчейну; 3) вивчити існуючих методів криптографії у блокчейн-технології; 4) побудувати власний алгоритм реалізації технології блокчейну із застосуванням криптокрафічних інструментів.

1 ІНФОРМАЦІЙНИЙ ОГЯД

1.1 Переваги та недоліки технології blockchain

Децентралізація є однією з найбільш вагомих переваг блокчейн-технології. Це означає, що дані зберігаються на всіх вузлах мережі, а не на централізованому сервері. Це забезпечує високу стійкість до відмов та атак, оскільки немає однієї точки відмови [9].

Надійність та безпека також є важливими перевагами. Кожен блок даних зв'язаний з попереднім блоком за допомогою криптографічних геш-функцій, що робить маніпулювання чи зміну даних майже неможливими.

Прозорість є ключовою перевагою блокчейну. Транзакції у блокчейні публічні та доступні для перевірки будь-ким охочим. Це забезпечує високий рівень довіри до системи, оскільки всі учасники можуть перевірити правильність транзакцій.

Масштабованість є одним із найбільших недоліків технології блокчейн. Це через фіксований розмір блоку для зберігання інформації, який становить лише 1 Мб, тому він може вмістити лише пару транзакцій на одному блоці [10].

Незрілість є ще одним чинником, який ускладнює прийняття технології блокчейн. Блокчейн — це технологія, що розвивається лише протягом кількох років, і багато людей ще не відчувають повної довіри до неї. Хоча деякі застосування блокчейну успішно працюють у різних галузях, для його повного прийняття потрібно ще багато роботи з впровадження та будування довіри користувачів.

Енергоспоживання є ще однією серйозною проблемою технології блокчейн. Для перевірки кожної транзакції витрачається значна кількість енергії. Доказом цього, зокрема, є те, що через блокчейн-технологію під час перевірки транзакцій у 2018 році було використано 0,3% всієї світової електроенергії.

Затратність часу також є значною проблемою. Для додавання нового блоку до ланцюга майнерам потрібно багато разів обчислювати значення nonce, що робить цей процес трудомістким і повільним, особливо в промислових масштабах.

Юридичні обмеження також гальмують прийняття технології блокчейн. У деяких країнах використання додатків з блокчейн та криптовалюти заборонено через екологічні проблеми, які вони створюють, що ускладнює їх використання в комерційних цілях.

Проблеми зі сховищем також стають важливими при збільшенні обсягів транзакцій. Бази даних блокчейн зберігаються на всіх вузлах мережі, що може створити проблеми з пам'яттю при збільшенні кількості транзакцій.

Нарешті, правові обмеження і проблеми з фінансовими установами можуть ускладнити прийняття блокчейну в більш широкому масштабі. Для повного впровадження технології блокчейн буде необхідно розв'язувати ці питання та створити сприятливе середовище для її розвитку.

1.2 Механізм функціонування технології блокчейн

Мета блокчейна полягає в забезпеченні можливості запису та поширення цифрової інформації, при цьому вона залишається незмінною і нередагованою. Таким чином, блокчейн стає фундаментом для створення необхідних незмінних реєстрів чи записів транзакцій, які неможливо змінити, вилучити або зруйнувати (рис. 1.1).

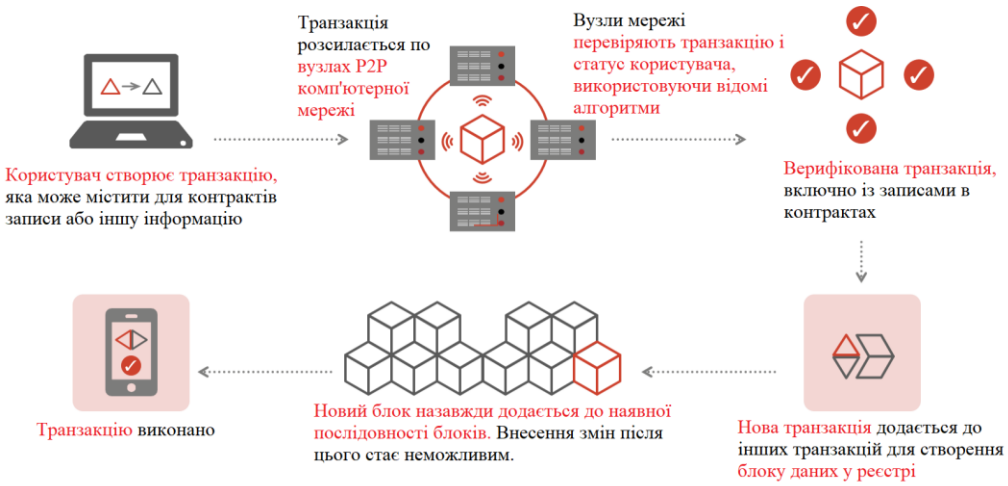

Рисунок 1.1 - Схема роботи блокчейна

Завдяки блокчейну можна безпечно, децентралізовано та прозоро обмінюватися будь-якою інформацією між двома сторонами. Наприклад, у криптовалютах це гарантує безпеку транзакцій, адже чітко ідентифікує їх учасників та отримувачів. Проте, багато людей ставлять під сумнів те, як саме блокчейн захищає всіх учасників.

В основі роботи блокчейну лежать криптографічні методи, механізми консенсусу та інші алгоритми, що гарантують надійну безпеку [17].

Механізм функціонування технології блокчейн можна описати як складну, але ефективну систему, що базується на декількох ключових принципах. Перш за все, блокчейн використовує розподілену мережу, у якій інформація розподіляється між безліччю вузлів або комп'ютерів. Кожен вузол мережі має копію всієї інформації, що забезпечує надійність та доступність даних.

Другим ключовим аспектом є створення блоків даних, які утворюють ланцюжок. Кожен блок містить у собі певну кількість транзакцій або іншої корисної інформації, а також унікальний ідентифікатор, який посилається на попередній блок, створюючи послідовний рядок блоків.

Третім важливим елементом є децентралізований консенсус, механізм, який визначає, які блоки будуть додані до ланцюжка. Цей процес може

використовувати різні алгоритми консенсусу, такі як Proof of Work або Proof of Stake, які допомагають у досягненні єдності в мережі.

Крім того, блокчейн використовує криптографічні методи для забезпечення безпеки та цілісності даних. Це включає в себе використання геш-функцій та цифрових підписів для підтвердження автентичності та непорушності інформації.

Незмінність даних є ще одним ключовим аспектом блокчейну. Одразу після того, як блок доданий до ланцюжка, його вже неможливо змінити або видалити. Це створює надійну базу даних, що забезпечує високий рівень довіри до інформації, збереженої в блокчейні.

1.3 Блоки в блокчейні

Блоки є фундаментальними елементами будь-якого блокчейну, які дозволяють створювати незмінний та безпечний реєстр транзакцій і даних. Кожен блок може містити різні типи інформації, включаючи транзакції, документи, контракти або будь-яку іншу корисну інформацію. Важливою характеристикою кожного блоку є його унікальний ідентифікатор, який називається гешем. геш генерується на основі вмісту блоку та попереднього блоку у ланцюжку, що робить його унікальним і надійним.

Кожен блок також містить посилання на попередній блок у ланцюжку, створюючи послідовний ряд блоків. Це дозволяє створювати хронологічний порядок транзакцій та подій у блокчейні, що забезпечує послідовність та невід'ємність даних. Крім того, кожен блок містить таймстемп, який вказує на час його створення, що дозволяє точно визначити, коли була здійснена певна транзакція або подія.

У разі зміни будь-якого блоку в ланцюжку, геші всіх наступних блоків автоматично стають недійсними, оскільки геші попередніх блоків входять до складу обчислених значень. Це забезпечує високий рівень безпеки та незмінності даних у блокчейні, оскільки будь-яка спроба змінити історію транзакцій вимагає внесення змін у всі підписані блоки, що є практично

неможливим. Така структура блоків робить блокчейн надійним та довіреним механізмом для збереження та обміну інформацією в цифровому середовищі.

Послідовність в контексті ланцюжка блоків означає, що кожен новий блок додається до кінця ланцюжка в хронологічному порядку, створюючи послідовну історію транзакцій і подій. Це ключовий аспект блокчейну, який гарантує, що інформація зберігається та відображається в правильному порядку від початку до кінця.

Коли новий блок формується та додається до ланцюжка, він отримує послідовний номер або інший ідентифікатор, який вказує на його місце у послідовності блоків. Це дозволяє легко визначати порядок подій та транзакцій у блокчейні.

Блокчейн забезпечує послідовність через механізм додавання нових блоків до кінця ланцюжка згідно з певними правилами та протоколами. Кожен новий блок містить у собі посилання на попередній блок у ланцюжку, що створює зв'язок між ними.

Цей принцип послідовності дозволяє користувачам блокчейну відстежувати та перевіряти послідовність подій та транзакцій, що відбуваються в системі. Крім того, він дозволяє забезпечити цілісність даних, оскільки будь-яка спроба змінити історію транзакцій потребує зміни всього ланцюжка, що робить такі спроби майже неможливими.

Одним з важливих елементів у будові блокчейну є зв'язок з попереднім блоком, що створює послідовний рядок блоків. Кожен блок містить геш (унікальний ідентифікатор) попереднього блоку у цьому рядку.

Блоки криптографічно пов'язані між собою. Блокчейн - це сукупність блоків [2]. Блок - це сукупність усіх транзакцій з криптографічним гешем попереднього блоку. Під час додавання нового блоку генерується новий геш, який записується в заголовок останнього блоку і в заголовок наступного блоку, який створив ланцюжок. На той час кожен заголовок блоку має геш попереднього блоку і геш наступного блоку. Такий спосіб зв'язку блоків відомий як криптографічний зв'язок блоків для перевірки та прийняття

консенсусного рішення. Старі блоки не можуть бути змінені; тільки нові блоки можуть бути додані в ланцюжок. Нові блоки дублюються по всій книзі в розподіленій мережі, оскільки блокчейн є розподіленою книгою, і для уникнення конфліктів встановлюються правила. Простіше кажучи, блокчейн - це розподілені незмінні цифрові реєстри криптографічно підписаних транзакцій, які згруповані в блоки (рис. 1.2).

Цей зв'язок визначає порядок блоків у рядку, починаючи з першого блоку і закінчуючи останнім. При створенні нового блоку, який додається до рядка, його посилання автоматично вказує на останній блок у цьому рядку, утворюючи таким чином послідовний ланцюжок блоків.

Рисунок 1.2 – Ланцюжок блоків

Цей зв'язок визначає порядок блоків у рядку, починаючи з першого блоку і закінчуючи останнім. При створенні нового блоку, який додається до рядка, його посилання автоматично вказує на останній блок у цьому рядку, утворюючи таким чином послідовний ланцюжок блоків.

Наприклад, якщо хтось намагається витратити одні й ті ж кошти двічі, це буде негайно виявлено, адже геші наступних блоків не відповідатимуть зміненим даним. Візуалізувати цей принцип можна у вигляді ланцюжка, де кожен блок - це ланка. Неможливо вилучити або змінити ланку, не порушивши цілісність всього ланцюжка.

Важливо зауважити, що геш попереднього блоку включається до обчислення гешу поточного блоку. Це означає, що навіть найменша зміна у попередньому блоку призведе до зміни гешу, і він вже не буде відповідати записаному у поточному блоку. Таким чином, будь-яка спроба змінити дані у будь-якому блоку відобразиться на гешах всіх наступних блоків, що забезпечує незмінність даних і додатковий рівень захисту від маніпуляцій чи втручань.

Розподілена мережа у своїй суті означає, що вона складається з безлічі вузлів, які розташовані по всьому світу, і кожен з них має копію даних. Це дозволяє уникнути централізованого контролю та зберігати незалежні копії інформації на кожному вузлі. Кожен вузол в мережі може виконувати обробку даних, а також приймати та передавати інформацію іншим вузлам.

Ця реплікація даних робить мережу стійкою до виходу з ладу окремих вузлів чи атак з боку зовнішніх загроз. Будь-яка зміна в системі потребує узгодження між багатьма учасниками мережі, що забезпечує консенсус та недозмінність даних. Взаємозв'язок між вузлами існує через постійну спілкування та обмін даними між ними, що забезпечує синхронізацію даних та узгодженість стану мережі.

Оскільки розподілена мережа не має одного центрального пункту вразливості, вона є відомим механізмом забезпечення безпеки даних. Інформація розподіляється по всій мережі, що ускладнює можливість несанкціонованого доступу або зміни даних. Такий підхід до зберігання та обміну даними робить розподілену мережу основою для різноманітних застосувань блокчейну, які потребують безпеки, довіри та децентралізації.

Розподілена мережа також забезпечує відкритий доступ до даних та можливість участі для будь-якого учасника мережі, без необхідності довіри до посередника чи централізованої організації. Це означає, що кожен користувач може перевірити та переглянути історію транзакцій або подій у блокчейні, що забезпечує прозорість та відкритість системи. Такий принцип дозволяє

створювати довіру між учасниками мережі та забезпечує рівноправність доступу до інформації для всіх користувачів.

Крім того, розподілена мережа дозволяє впровадження нових функцій та додатків без необхідності зміни основної архітектури чи інфраструктури. Це означає, що розвиток блокчейн-проектів може бути швидким і гнучким, адже додатки можуть бути створені та запущені на основі наявної мережі без значних втрат часу та ресурсів на перебудову системи.

Однією з переваг розподіленої мережі є також зменшення залежності від одного провайдера або постачальника послуг. Завдяки розподіленій архітектурі, мережа може функціонувати незалежно від будь-яких одного вузла чи централізованої організації, що забезпечує стійкість та надійність системи у разі виникнення проблем або збоїв.

1.4 Використання технології блокчейн у різноманітних аспектах повсякденного життя

Блокчейн-технологія відіграє ключову роль у фінансовому секторі, перетворюючи спосіб, яким здійснюються транзакції та управляються активами. Одним із найбільш відомих застосувань блокчейну у фінансах є криптовалюта, така як біткойн. Криптовалюта дає змогу безпосередньо пересилати цифрові активи між користувачами без участі посередників, таких як банки чи платіжні системи.

Однак блокчейн не обмежується лише криптовалютами. Він також використовується для побудови децентралізованих фінансових послуг, таких як смарт-контракти на базі Ethereum. Смарт-контракти це програмні додатки чи протоколи транзакцій, призначені для автоматичного виконання, контролю, а також документування подій і дій відповідно до умов контракту чи угоди. Вони можуть бути використані для створення різноманітних фінансових інструментів, таких як децентралізовані біржі, позики та страхування [18].

Блокчейн також відіграє важливу роль у забезпеченні безпеки та прозорості фінансових транзакцій. Блокчейн-мережі забезпечують неперевершену захищеність від шахрайства та зловживань завдяки своїй децентралізованій природі та криптографічним методам шифрування. Крім того, усі транзакції, що відбуваються в мережі, записуються в блокчейні, що робить їх публічно доступними та перевіряємими всіма учасниками мережі.

Одним з основних переваг використання блокчейну у фінансах є зниження витрат на транзакції та усунення посередників. Банківські та інші фінансові установи зазвичай вимагають високі комісії за здійснення транзакцій та інші послуги. Використання блокчейну може дозволити значно скоротити ці витрати та зробити фінансові послуги доступнішими для всіх.

Загалом, блокчейн відкриває безліч можливостей для інновацій у фінансовому секторі, забезпечуючи більшу ефективність, безпеку та доступність фінансових послуг для всіх користувачів.

Логістика та постачання - одна з ключових галузей, в яких технологія блокчейн має значний потенціал. Блокчейн може допомогти оптимізувати процеси логістики та постачання, забезпечуючи безпеку, прозорість та ефективність [4].

Ця технологія може бути використана для створення децентралізованих систем відстеження ланцюгів постачання. Кожен крок у постачальному ланцюгу може бути записаний в блокчейні, починаючи від виробника і закінчуючи покупцем. Це дозволяє всім учасникам мережі миттєво перевіряти походження та шлях товару, що забезпечує більшу довіру в продукт і зменшує ризик підробки.

Також, блокчейн може полегшити вирішення проблем з недостатньою інформацією про стан запасів або доставок. Всі учасники мережі можуть мати доступ до одного централізованого джерела інформації, що сприяє швидкому виявленню та вирішенню проблем.

Додатково, використання блокчейну може спростити процеси платежів у логістиці та постачанні. Смарт-контракти, які використовуються на

блокчейні, можуть автоматизувати платежі при досягненні певних умов, таких як доставка товару або прийняття послуги. Це дозволяє зменшити час та витрати, пов'язані з проведенням та виконанням фінансових транзакцій.

Усе це сприяє підвищенню ефективності та конкурентоспроможності логістичних та постачальних ланцюгів, що відображається на якості та швидкості обслуговування клієнтів, а також на загальних витрат.

2. ТЕОРЕТИЧНИЙ АНАЛІЗ

2.1 Технологія P2P та її функціонування

Технологія P2P (peer-to-peer) це архітектура мережі, де кожен учасник, або «пір», має можливість взаємодіяти безпосередньо з іншими учасниками. Ця модель розподілення функцій дозволяє мережі працювати без централізованого сервера або контрольної точки, замість цього керуючись безпосередньо кожним піром. В такій системі кожен пір може виконувати як клієнтські, так і серверні функції, що робить його активним учасником мережі незалежно від його ролі (рис. 2.1).

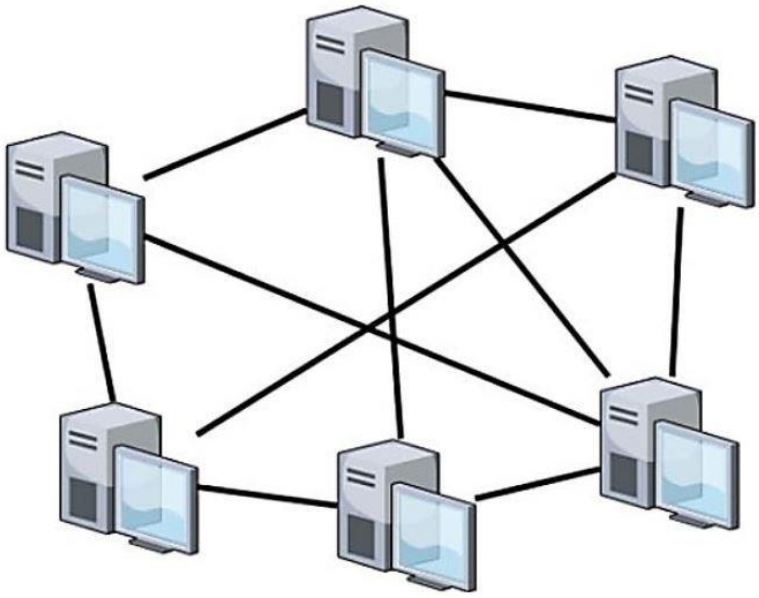

Рисунок 2.1 – Архітектура мережі виду peer2peer

Peer-to-Peer технологія яка грає важливу роль в блокчейн-мережах. У блокчейні P2P вузли взаємодіють один з одним без посередництва централізованого сервера. Кожен вузол мережі має однакові права та відповідальність, і кожен з них може бути як клієнтом, так і сервером.

Технологія P2P дозволяє забезпечити децентралізованість, прозорість та невідмовність у блокчейн-мережах. Кожен вузол мережі зберігає копію всього блокчейну і має змогу перевірити та підтвердити транзакції, що забезпечує безпеку та надійність мережі.

Крім того, технологія P2P дозволяє забезпечити швидке поширення інформації всією мережою. Коли формується новий блок, або відбувається нова транзакція, вона швидко розповсюджується між вузлами через механізм P2P, що дозволяє всій мережі бути синхронізованою.

Також важливою роллю технології P2P в блокчейнах є забезпечення можливості взаємодії користувачів мережі без посередництва третьої сторони. Це дозволяє користувачам взаємодіяти прямо між собою, використовуючи блокчейн для пересилання цифрових активів, укладання угод та виконання різних операцій без необхідності довіряти централізованій організації або посереднику.

Отже, технологія P2P в блокчейнах відіграє ключову роль у забезпеченні децентралізації, прозорості, швидкості та безпеки мережі, а також у сприянні прямій взаємодії між користувачами без посередництва.

Одним із головних переваг цієї архітектури є децентралізація та стійкість до виходу з ладу окремих вузлів. У випадку виходу з ладу одного чи кількох пірів, мережа може продовжувати працювати, оскільки кожен пір має незалежну роль та можливість спілкуватися з іншими учасниками. Це робить мережу P2P надійною та стійкою до внутрішніх та зовнішніх атак.

Ще одна особливість технології P2P - це здатність відображати інформацію в поточний момент часу без можливості перегляду даних в минулому форматі. Коли клієнт звертається до бази даних, він отримує актуальний стан даних, а не може переглянути їхню історію змін. У сучасних базах даних, хоч і існують можливості перегляду історії змін, відсутній механізм перевірки того, що дані не були змінені заднім числом. Це може викликати проблеми довіри до адміністраторів бази даних та створює необхідність в резервних копіях даних на регулярній основі.

Третя особливість технології P2P полягає у можливості клієнта виконувати основні операції з інформацією, відомі за абревіатурою CRUD (створення, читання, оновлення, видалення). Це означає, що клієнт може створювати нові записи в базі даних, зчитувати їх (переглядати), оновлювати

і видаляти існуючі. Такий функціонал дозволяє клієнтам зручно керувати інформацією у мережі P2P без необхідності постійного звертання до адміністраторів бази даних [11].

Крім того, в мережах P2P кожен учасник може взаємодіяти безпосередньо з іншими, що може підвищити швидкість комунікації та знизити затримки. Це забезпечує ефективну передачу даних між учасниками та дозволяє реалізовувати широкий спектр додатків та сервісів. У підсумку, технологія P2P виявляється корисною та потужною для багатьох сфер, включаючи файлообмін, блокчейн, спільну роботу, відеоконференції та багато іншого.

Фундаментально P2P-система підтримується розподіленою мережею користувачів, при цьому зазвичай відсутній центральний адміністратор або сервер, оскільки кожна нода містить копії файлів і виступає як клієнт і як сервер для інших вузлів. Це дозволяє кожній ноді як завантажувати файли з інших, так і відвантажувати файли на них. Саме така властивість відрізняє P2P-мережі від традиційних клієнт-серверних систем, де клієнтські пристрої завантажують дані з централізованого сервера.

У P2P-мережах здійснюється обмін файлами, що зберігаються на жорстких дисках пристроїв. За допомогою спеціальних програм користувачі можуть запитувати та завантажувати файли на інші пристрої в мережі. Після завантаження файл може слугувати джерелом для інших користувачів.

Простіше кажучи, коли ноди виступають як клієнти, вони можуть завантажувати дані з інших вузлів мережі, а коли вони ідентифікуються як сервери, то вони самі стають джерелом даних і інші ноди можуть завантажувати з них дані. Тим не менше, на практиці ноди можуть виконувати обидві функції одночасно (наприклад, завантаження файлу A та відвантаження файлу B).

Завдяки зберіганню, передачі та отриманню файлів кожною нодою, P2P-мережі стають швидшими та ефективнішими з ростом їхнього користувацького базису. Крім того, їхня розподілена архітектура забезпечує

високий рівень стійкості до кібератак, оскільки вони не мають однієї центральної точки вразливості.

Peer-to-peer системи можна класифікувати за їхньою архітектурою на три основних типи: неструктуровані, структуровані та гібридні мережі [11].

2.2 Види схем блокчейн мереж

Блокчейн являє собою таку структуру даних, яка додає нові записи в розподільну базу з публічним доступом до неї різних незалежних учасників [5]. Є 3 основні типи блокчейн-технології: публічна, приватна та гібридна (рис. 2.2).

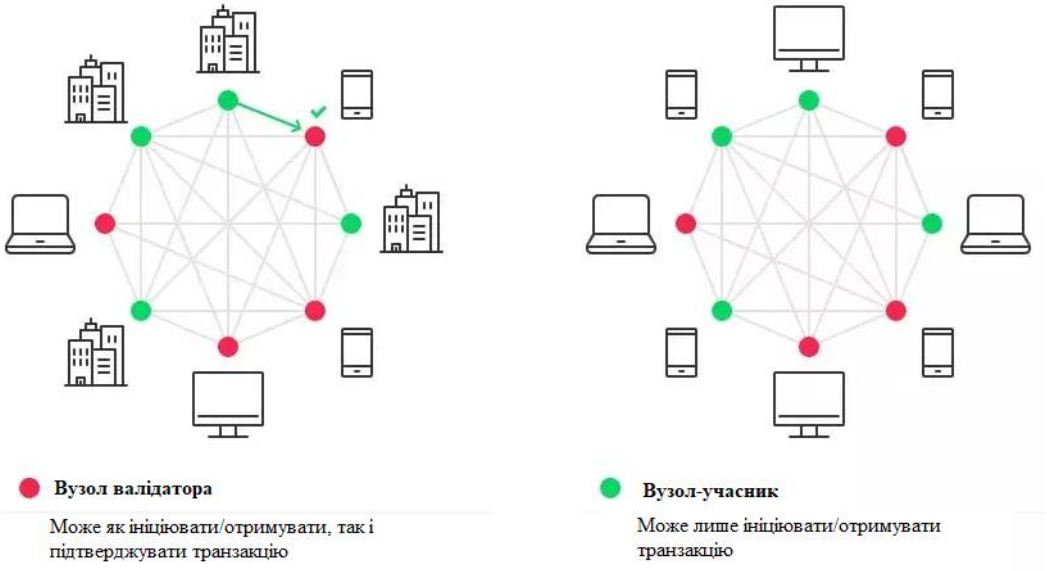

Рисунок 2.2 – Вузли в публічних і приватних блокчейнах

Публічний блокчейн – це тип блокчейн-мережі, який доступний для всіх учасників без будь-яких обмежень. Він є відкритим для перегляду, перевірки та участі. Основна ідея публічного блокчейна полягає в тому, щоб будь-хто міг приєднатися до мережі, переглядати всі транзакції та, за бажанням, навіть приєднатися до процесу блокчейн-консенсусу.

У публічному блокчейні мережі відомі своєю високою стійкістю до цензури і змін. Транзакції відбуваються за допомогою криптографії, що

робить їх майже неможливими до зміни або скасування. Ця характеристика робить публічні блокчейни привабливими для застосувань, де потрібна висока рівень надійності та безпеки даних.

Основні характеристики публічного блокчейна включають децентралізацію, відкритий доступ, анонімність, достовірність транзакцій, відкритість протоколу та можливість власності активів.

Публічний блокчейн використовується для різних цілей, включаючи створення криптовалют, децентралізованих фінансових послуг (DeFi), створення цифрових активів, забезпечення прозорості управління та багато іншого. Цей тип блокчейна широко застосовується в криптоспільноті та має значний вплив на різні сфери економіки та технологій.

Приватний блокчейн, на відміну від публічного, обмежений доступом тільки до певного кола учасників або організацій. Це означає, що лише певні користувачі мають право переглядати, виконувати транзакції та підтримувати мережу блокчейна. В основному, приватні блокчейни контролюються однією організацією або консорціумом, і вони можуть мати централізований характер.

Основні особливості приватного блокчейна включають обмежений доступ, централізоване управління, вищий рівень конфіденційності, швидкість та масштабованість, а також більша відповідальність та довіра. Приватні блокчейни часто використовуються в корпоративних та фінансових секторах, де потрібний баланс між прозорістю, конфіденційністю та швидкістю обробки транзакцій.

Вони можуть бути ефективним інструментом для вирішення специфічних завдань і дозволяти організаціям зберігати та обробляти дані в безпечному та ефективному середовищі. Це може бути особливо корисним у випадках, коли потрібно дотримуватися регуляторних вимог або забезпечити конфіденційність бізнес-інформації. Приватні блокчейни також можуть забезпечувати підтримку для децентралізованих додатків і послуг всередині

обмеженого кола учасників, що сприяє інноваціям та ефективному використанню технології блокчейн.

Гібридний блокчейн поєднує в собі елементи як публічного, так і приватного блокчейна. Цей тип блокчейна може мати як відкриту, так і закриту частину, яка може бути доступна для різних категорій користувачів. Такий підхід дозволяє забезпечити баланс між прозорістю та конфіденційністю даних.

Основна ідея гібридного блокчейна полягає в тому, щоб використовувати переваги обох типів блокчейнів у залежності від конкретних потреб проєкту або організації. Наприклад, відкрита частина блокчейну може використовуватись для публічних транзакцій або для взаємодії з широким загалом, тоді як закрита частина може використовуватись для конфіденційних операцій або для співпраці між обмеженим колом учасників.

Гібридні блокчейни часто використовуються в корпоративному секторі, де потрібна збалансована комбінація прозорості, конфіденційності та контролю над даними. Вони можуть бути корисними для великих компаній або урядових організацій, які шукають способи використання блокчейн-технологій, але водночас хочуть мати контроль над своєю інформацією та операціями.

Гібридні блокчейни можуть також бути використані в галузях, де важлива взаємодія між різними сторонами, такими як логістика, ланцюжок постачання або управління ліцензіями, де потрібно комбінувати публічні та приватні дані та операції.

2.3 Роль майнерів у блокчейні

Майнінг — це те, як нові одиниці криптовалюти випускаються у світ, як правило, в обмін на підтвердження транзакцій. Хоча з часом можливість майнити криптовалюту в системах підтвердження роботи, таких як біткойн, звичайній людині стає все дедалі складніше. Цей процес базується на

використанні децентралізованої мережі комп’ютерів, зосереджених по всьому світу, які виконують перевірку, а відтак і захист цілісності віртуальних книг блокчейна, які здійснюють документування операцій з криптовалютою. Використання обчислювальної потужності комп’ютерів такої мережі обмінюється на нові криптомонети. Після процесу гешування всі транзакції об’єднуються в геш-дерево, де вони гуртуються попарно до ти, доки процес не дійде до «вершини дерева» (рис. 2.3). Це доброчесне коло: майнери підтримують і захищають блокчейн, блокчейн нагороджує монетами, монети є стимулом для майнерів підтримувати блокчейн [6].

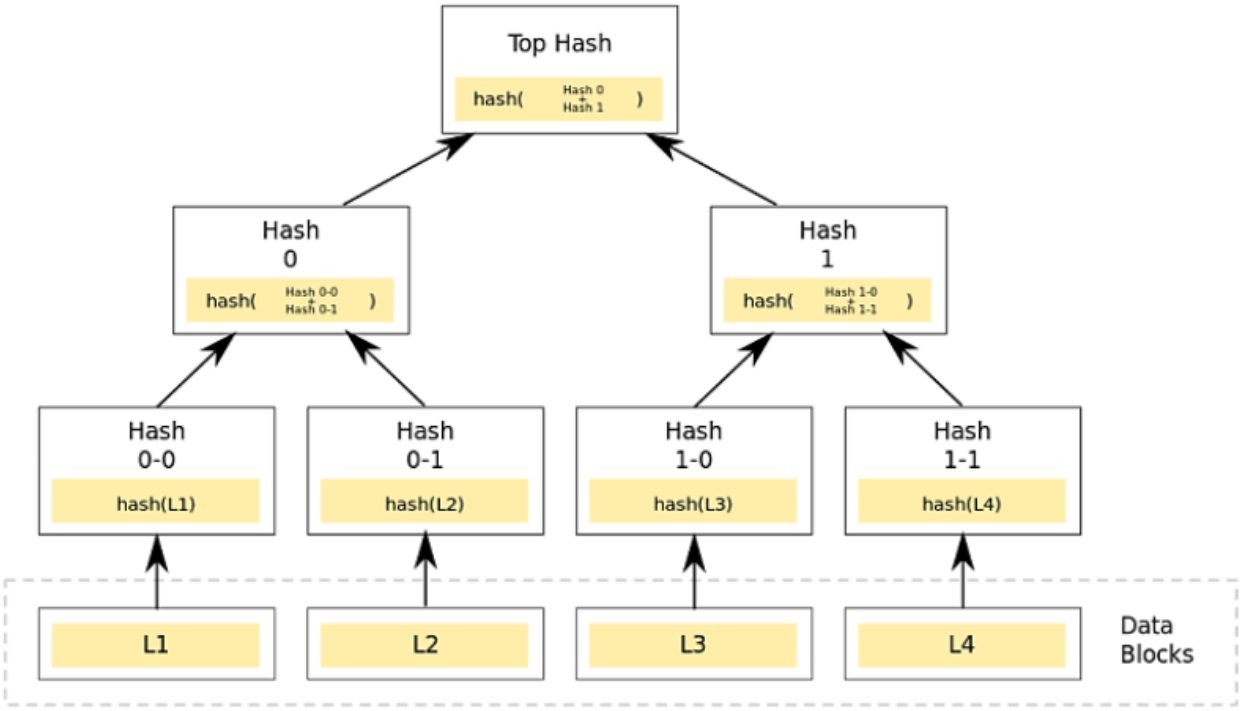

Рисунок 2.3 – геш-дерево

Кожен блок отримує свій унікальний ідентифікатор шляхом поєднання гешу цього блоку з гешем попереднього блоку і деяким випадковим числом, дотримуючись визначеного протоколу. Цей процес робиться за певним стандартом. Іноді виникає ситуація, коли два вузли додають підтверджений блок, і користувачі починають майнити на основі цієї неправильної інформації [21]. Це призводить до продовження конкуренції між майнерами до тих пір, поки не буде отриманий один блок, який базується на одному з двох попередніх блоків (рис. 2.4).

Рисунок 2.4 – Схема процесу переказу грошей

Крім випуску нових монет в обіг, майнінг є ключовим фактором безпеки біткойна (і багатьох інших криптовалют). Він виконує функції перевірки та захисту блокчейна, а це санкціонує однорангове функціонування криптовалюти без будь-якого стороннього контролю. Саме це стимулює приєднання нових майнерів до мережі, а також збільшення та технічне оновлення обчислювальних потужностей вже приєднаних.

2.4 Популярні методи гешування в блокчейні та їх переваги

У технології блокчейну застосовуються різноманітні алгоритми гешування, кожен з яких має свої унікальні характеристики, з метою забезпечення безпеки та цілісності даних. До найвідоміших алгоритмів гешування, які використовуються в блокчейні, відносяться:

Equihash – це алгоритм гешування на основі пам’яті, який є стійким до типу майнінгу Application-Specific Integrated Circuit[1] (ASIC). Кілька криптовалют, включаючи Zcash, використовують його для забезпечення більш

[1] Тип майнінгу криптовалюти, який для ефективної роботи використовує вузькоспеціалізоване обладнання.

справедливого середовища майнінгу, де спеціалізоване обладнання не дає переваги.

SHA-256 – це відомий алгоритмом гешування, який розроблений Агентством національної безпеки (АНБ) у 2001 році та широко використовується в області блокчейну. Він створює рядок із 64 символів, що формують 256-бітний геш, і завдяки своїм надійним функціям безпеки став популярним вибором для багатьох криптовалют, включаючи Біткойн.

Ethash, спеціально розроблений для блокчейна Ethereum, є алгоритмом гешування з жорстким доступом до пам'яті. Він створений для того, щоб бути стійким до майнерів ASIC. Мета Ethash - зробити майнінг доступним для широкого кола учасників.

Scrypt - це функція отримання ключів, яка призначена для інтенсивнішого використання пам'яті, ніж інші алгоритми. Високі вимоги до пам'яті роблять його більш стійким до атак з використанням спеціалізованого обладнання. Scrypt широко використовується в різних криптовалютах (зокрема у Litecoin) для підвищення їх безпеки від апаратних загроз.

Ефективність гешування на блокчейні означає, що такі ресурси, як простір для зберігання та обчислювальна потужність, використовуються максимально ефективно.

Зберігати менше даних: Оскільки зберігання всіх даних для кожної транзакції в блокчейні може бути дуже великим, гешування дозволяє замінити їх компактними геш-значеннями. Замість того, щоб зберігати самі транзакції, блокчейн зберігає лише їхні геші, що значно зменшує обсяг необхідного сховища. Це особливо важливо в децентралізованих системах, де кожен вузол повинен зберігати повну копію блокчейну.

Скорочення часу перевірки: коли до блокчейну додається новий блок, необхідно перевірити цілісність усіх попередніх блоків. Завдяки гешам кожен блок містить геш попереднього блоку. Якщо дані в попередньому блоці змінюються, то змінюється і його геш. Це дозволяє швидко виявляти будь-які

зміни і підтримувати цілісність блокчейну без необхідності повторної перевірки всіх даних.

Економія обчислювальних ресурсів: гешування вимагає відносно менше обчислювальних ресурсів, ніж інші методи забезпечення цілісності даних. Оскільки алгоритми гешування, як правило, швидкі та ефективні, вони зменшують навантаження на систему, особливо при обробці великої кількості транзакцій.

Конфіденційність є критично важливим аспектом у світі блокчейну, особливо коли йдеться про персональні дані або чутливу інформацію. гешування робить значний внесок у забезпечення конфіденційності даних завдяки своїй природі перетворення даних у непередбачувані геш-значення.

Коли дані піддаються гешуванню, вони перетворюються на рядок фіксованої довжини, який слугує своєрідним «відбитком» вихідних даних. Цей процес виконується односторонньо: з геш-значення неможливо отримати вихідні дані. Тобто, навіть невелика зміна у вихідних даних призведе до радикальної зміни геш-значення. Це означає, що якщо хтось отримає доступ до геш-значення, йому буде вкрай складно або практично неможливо відновити вихідні дані з цього гешу.

У контексті блокчейна, де конфіденційність даних відіграє критичну роль, це означає, що навіть якщо хтось отримає доступ до блокчейна або його фрагмента, він не зможе прочитати і зрозуміти вміст блоків завдяки гешуванню даних. Це допомагає захищати персональні дані користувачів, банківські транзакції та іншу чутливу інформацію, яка може бути включена в блокчейн.

2.5 Технологія Proof of work

Proof of Work (PoW) – досить важлива концепція в технології блокчейн. Вона використовується для підтвердження транзакцій та створення нових блоків в блокчейні. Основна ідея полягає в тому, щоб зробити процес

створення нових блоків у розподіленій системі, якій немає централізованого управління, достатньо складним, щоб зменшити ймовірність шахрайства та атак.

У PoW майнери вирішують складні математичні завдання, які вимагають великої обчислювальної потужності. Вирішення цих завдань вимагає великої кількості спроб та обчислювальних ресурсів. Після того, як майнер вирішує завдання, він анонсує це всім учасникам мережі, і його блок додається до ланцюжка блоків.

Однак PoW має і свої недоліки. Це досить енергоємний процес через велике споживання електроенергії, оскільки майнери змушені використовувати потужні комп'ютери для вирішення складних математичних завдань. Тому були розроблені інші методи підтвердження транзакцій, такі як Proof of Stake (PoS), які вимагають меншої енергії. Однак PoW залишається одним з найпоширеніших методів підтвердження транзакцій у біткойні та деяких інших блокчейн-мережах.

Алгоритм верифікації операцій важливий з кількох причин. По-перше, він забезпечує безпечний та децентралізований метод підтримки цілісності блокчейн-реєстру для учасників мережі. По-друге, PoW (Proof of Work) стимулює майнерів з усього світу використовувати обчислювальні ресурси для верифікації блоків, що допомагає утримати темп створення нових блоків на стабільному рівні.

Додатково, PoW контролює видобуток нових монет. Наприклад, у випадку з біткойнами, алгоритм автоматично регулює складність видобутку, щоб забезпечити сталість темпу створення нових блоків. Без цього контролю майнери можуть витрачати запаси BTC швидше, ніж потрібно для економічної стабільності. Крім того, зі зростанням потужності мережі стає вкрай складно атакувати систему, побудовану на основі ланцюжка PoW.

2.5.1 Як працює підтвердження роботи

Майнінг біткойнів через proof-of-work подібний до участі у лотереї, де кожні 10 хвилин випадають призи. Щоб брати участь, достатньо мати майнінгове обладнання та підключитися до мережі. Чим більше обладнання, тим більше шансів на виграш, але участь у пулі може збільшити їхню виграшну ймовірність та розподіл призів. Кожен учасник може перевірити правила та зміни, що кодуються в програмному забезпеченні біткойну [19].

У прикладі з лотерейними квитками, що використовуються для визначення гешрейту, нагородою є BTC за успішний блок Bitcoin. Швидкість гешування визначається як кількість гешів на секунду, яку майнінгове обладнання може виконати. Чим ефективніше обладнання, тим більше шансів отримати винагороду. Приєднання до майнінг-пулу збільшує шанси виграти, але розподіл виграшу відбувається пропорційно гешрейту. Критики вказують на централізуючий ефект пулів, але існують і механізми для збереження децентралізації. Участь у майнінгу відповідає правилам програмного забезпечення Bitcoin, і кожен може прийняти ці правила, вирішивши зайнятися видобутком біткойнів [12].

Користувачі можуть приєднатися до майнінг-пулів, щоб збільшити свої шанси на виграш. В приєднаному пулі виграш розподіляється пропорційно гешрейту учасників, але вони можуть вийти з нього за бажанням, щоб уникнути централізації або нечесності. Участь регулюється програмним забезпеченням Bitcoin.

В пулах для майнінгу Bitcoin виграш ділиться між учасниками пропорційно їх потужності. Участь добровільна, а вихід з пулу простий. Це нівелює ризик надмірної централізації.

Як і в лотереї, правила та винагороди чітко прописані в коді Bitcoin. Кожен може їх перевірити та вирішити, чи брати участь у видобутку.

2.5.2 Переваги та недоліки Proof of work

Будучи найпершою консенсусною моделлю для блокчейнів, плюси і мінуси систем підтвердження роботи стали очевидними лише в міру розвитку галузі. Незважаючи на новітні інновації, PoW залишається найбільш перевіреним, перевіреним часом методом досягнення консенсусу щодо публічного блокчейну.

Перш за все, PoW забезпечує безпечну операційну середу, де транзакції можуть бути підтверджені безпосередньо через розв'язання складних обчислювальних завдань. Це робить мережу менш вразливою до різних видів атак, оскільки нападники повинні витрачати величезні обчислювальні ресурси, щоб контролювати більшість обчислювальної потужності мережі.

Децентралізація полягає у тому, що учасники мережі, які виконують обчислювальні завдання для підтвердження транзакцій, можуть бути розподілені по всьому світу. Це означає, що немає центральної влади або контролю, і кожен майнер має однакові права і можливості. Така децентралізація сприяє безпеці мережі, оскільки важко атакувати систему, що розподілена та реплікована в багатьох вузлах. Крім того, вона забезпечує також інклюзивний характер мережі, дозволяючи будь-якій особі з обчислювальними ресурсами приєднатися до мережі та брати участь у її функціонуванні.

По суті, PoW забезпечує гармонійне поєднання безпеки та децентралізації, роблячи мережу більш стійкою до атак і забезпечуючи рівність учасників у процесі підтвердження операцій. Це дозволяє мережі ефективно функціонувати як розподілена система, що не піддається централізованому контролю чи маніпуляціям.

Регулювання видобутку нових монет є однією з ключових переваг Proof of Work (PoW) у криптовалютних мережах. У контексті PoW, цей процес автоматизований та інтегрований у саму структуру мережі. Основна ідея полягає в тому, що мережа автоматично контролює складність

обчислювальних завдань, які необхідно вирішувати для створення нових блоків та винагороди майнерів.

Прикладом може служити біткойн, де цільовий інтервал на створення нового блоку складає приблизно 10 хвилин. Алгоритм PoW регулює складність завдань таким чином, щоб зберігати цей інтервал стійким. Якщо мережа починає генерувати блоки швидше, ніж очікувалося, алгоритм автоматично підвищує складність завдань, необхідних для створення блоку. Це ускладнює процес майнінгу та знижує темп створення нових монет, що в свою чергу допомагає зберегти стабільність економіки криптовалюти.

Без такого регулювання, існує ризик перегенерації монет, що може призвести до інфляції або навіть дефляції, залежно від обставин. Автоматичне регулювання видобутку нових монет через PoW допомагає уникнути цих проблем, забезпечуючи стійкий інтерес до участі у мережі та стабільність криптовалютної економіки.

Існує кілька недоліків у системі Proof of Work (PoW). По-перше, енергетичні витрати великі. Майнінг криптовалют, який базується на PoW, потребує значних обчислювальних ресурсів, що призводить до великих витрат електроенергії. До прикладу, майнінг біткойнів наразі споживає електроенергію в річному вимірі 127 терават-годин, що перевищує річне споживання електроенергії в Норвегії. Тож це може мати негативний екологічний вплив та призводити до збільшення викидів вуглекислого газу.

По-друге, PoW може створювати централізацію. Майнінг стає все більш сконцентрованим у руках великих майнерських пулів, що може підірвати децентралізовану природу криптовалют.

Крім того, PoW не є ідеальним з точки зору масштабованості. Чим більша мережа, тим більше часу та енергії потрібно для верифікації та створення нових блоків, що може обмежувати продуктивність мережі при великому обсязі транзакцій.

Нарешті, PoW вразливий до атак 50%+1. Якщо зловмисник (або група зловмисників) контролює більшість обчислювальної потужності мережі, вони

можуть змінювати транзакції або навіть виконувати подвійні витрати, порушуючи безпеку та довіру до мережі.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

У цьому розділі роботи наведемо приклад програмної реалізації технології блокчейн із застосуванням можливостей криптографії. Для цього спершу розглянемо відповідне місце криптографічних інструментів, які мають бути присутніми в цій технології, та їх роль у подальшій реалізації. Саму ж програмну реалізацію будемо будувати, користуючись підказками, поданими в [3, 7, 8, 13 – 16, 20, 23, 25].

3.1 Використання гешування

Геш-функції відіграють вирішальну роль у безпеці блокчейну, генеруючи унікальні геш-коди фіксованого розміру для блоків даних. Ці односторонні функції забезпечують цілісність даних, оскільки навіть найменша зміна вихідних даних призводить до зовсім іншого гешу. Блокчейн використовує геш-функції для створення ланцюжка блоків, надійно зв’язуючи їх і забезпечуючи незмінний запис транзакцій.

Геш використовується для швидкого відрізнення однієї інформації від іншої без необхідності порівняння кожного біту, що значно прискорює перевірку транзакцій. Кожен блок складається з заголовку, кореня дерева Меркла, гешу попереднього блоку та транзакцій, при цьому кожен блок містить одну або декілька транзакцій. Кожен блок гешується з попереднім, а відсутній партнер гешується сам з собою. Операції проводяться до тих пір, поки не отримується єдиний геш – корінь дерева Меркла, що підтверджує достовірність блока та правильний порядок транзакцій. Геш-функція має три основні застосування в блокчейні:

В швидкій перевірці блокчейну використовується геш-функція, яка створює компактні представлення різних даних. Це дає можливість порівнювати дві транзакції шляхом порівняння їхніх геш-значень, щоб визначити, чи є вони ідентичними.

Захист від фальсифікації: для переконання, що дані не були змінені під час передачі, досить одночасно відправляти їх зведення. Для перевірки

еквівалентності між переданим і створеним зведенням, отримувач данних відтворює зведення. Якщо вони ідентичні, це свідчить про те, що дані не зазнали змін під час передачі.

Доказ робочого навантаження, який використовується головним чином в алгоритмі консенсусу Proof-of-Work (POW), вимагає надання певної кількості даних перед дозволом на пошук додаткових даних. Це призводить до створення геш-значення, яке менше за встановлений поріг. Зараз алгоритм консенсусу POW застосовується як у Bitcoin, так і в Ethereum.

Геш захищає блокчейн кількома способами:

Незмінність: будь-яка зміна даних в блоці викликає зміну його гешу. Це означає, що будь-яка спроба змінити інформацію буде автоматично виявлена, оскільки зміна гешу неминуче призведе до порушення цілісності блокчейну.

Цілісність: геші блоків можна порівняти, щоб перевірити, чи були дані змінені або модифіковані. Оскільки кожен блок містить геш попереднього блоку, навіть незначні зміни в даних будуть відображені в гешах наступних блоків, що унеможливлює приховування підроблених даних.

Стійкість до підробки: криптографічні геш-функції дуже важко підробити, що робить невигідним зміну даних в блокчейні. Навіть невеликі зміни в оригінальних даних призведуть до значних змін у геші, що робить майже неможливим підробку блоків непоміченою.

Верифікація транзакції в контексті блокчейна означає перевірку автентичності та цілісності транзакції перед її включенням у блок і додаванням у блокчейн.

Коли учасник блокчейн-мережі надсилає транзакцію, вона містить різноманітні дані, як-от адреса відправника, адреса одержувача, сума транзакції та інші метадані. Ці дані гешуються для створення унікального ідентифікатора транзакції.

Після створення транзакції вона транслюється в мережу, де інші учасники блокчейна можуть перевірити її справжність і цілісність. Це здійснюється шляхом перевірки підпису транзакції з використанням

відкритого ключа відправника та порівняння гешу транзакції з гешем, включеним у блокчейн.

Кожен вузол у мережі перевіряє транзакцію перед її включенням у наступний блок. Якщо транзакція пройшла всі перевірки і визнана дійсною, її додають до пулу непідтверджених транзакцій, і зрештою її включають у блок під час майнінгу.

У разі, якщо транзакція не проходить перевірку (наприклад, якщо підпис недійсний або сума транзакції перевищує доступні кошти), її відхиляють і не включають у блокчейн.У контексті блокчейну, ідентифікація відіграє ключову роль у забезпеченні унікальності та безпеки різних елементів системи. Гешування використовується для створення унікальних ідентифікаторів для різних об'єктів і сутностей у блокчейні. Наприклад, адреси гаманців у криптовалютних системах генеруються шляхом гешування публічних ключів, забезпечуючи кожному користувачеві унікальний ідентифікатор для проведення транзакцій і зберігання активів.

Ці унікальні ідентифікатори допомагають відстежувати та ідентифікувати різні угоди та операції в блокчейні без необхідності розкривати особисту інформацію або приватні ключі. Завдяки використанню гешів для ідентифікації, блокчейн забезпечує анонімність і безпеку користувачів, а також надійність системи загалом, оскільки зміна ідентифікатора вимагає зміни вихідних даних, що явно виявляється під час перевірки цілісності.

```rust
pub struct App {
    pub blocks: Vec,
}

#[derive(Serialize, Deserialize, Debug, Clone)]
pub struct Block {
    pub id: u64,
    pub hash: String,
    pub previous_hash: String,
    pub timestamp: i64,
    pub data: String,
    pub nonce: u64,
}
```

Рисунок 3.1 – Структура даних блокчейна

Структура App зберігає стан нашого додатка. Блокчейн не буде зберігатися, тому він зникне після завершення роботи додатка.

Стан додатка — це список блоків. У прикладі буде представлено, як будуть додаватися нові блоки в кінець цього списку, і це буде структурою даних блокчейну. Логіка додатка перетворить цей список блоків у ланцюжок, де кожен блок посилається на геш попереднього блоку.

Блок складається з ідентифікатора (id), що є індексом, який починається з 0, гешу sha256, гешу попереднього блоку, тимчасової мітки (timestamp), даних у блоці та nonce, який потрібне для майнінгу.

Перед майнінгом, було реалізовано функції валідації для підтримки стану в узгодженому вигляді, а також базові механізми консенсусу, щоб кожен клієнт знав, який блокчейн є правильним у разі виникнення конфліктів.

```
impl App {
    fn new() -> Self {
        Self { blocks: vec![] }
    }

    fn genesis(&mut self) {
        let genesis_block = Block {
            id: 0,
            timestamp: Utc::now().timestamp(),
            previous_hash: String::from("genesis"),
            data: String::from("genesis!"),
            nonce: 2836,
            hash: "0000f816a87f806bb0073dcf026a64fb40c946b5abee2573702828694d5b4c43".to_string(),
        };
        self.blocks.push(genesis_block);
    }

```

Рисунок 3.2 – Впровадження даних у структуру «App»

Спочатку потрібно ініціалізувати додаток з порожнім ланцюжком. Логіка критерія консенсусу: запитуємо інші вузли під час запуску про їхній ланцюжок і, якщо він довший за наш, використовуємо їхній.

Метод «genesis» створює перший, жорстко закодований блок у нашому блокчейні. Це «особливий» блок, оскільки він не зовсім відповідає тим самим правилам, що й інші блоки. Наприклад, у нього немає дійсного «previous_hash», оскільки до нього просто не було жодного блоку.

Це потрібно для завантаження нашого вузла, оскільки перший вузол починає роботу.

Тут ми отримуємо останній блок у ланцюзі — наш попередній блок — і перевіряємо, чи є блок, який ми хочемо додати, дійсно валідним. Якщо ні, ми просто фіксуємо помилку у журналі. У цьому додатку не буде реалізовано жодного реального оброблення помилок.

```
49  impl App {
50
51      fn try_add_block(&mut self, block: Block) {
52          let latest_block = self.blocks.last().expect("Має бути хоча б один блок");
53          if self.is_block_valid(&block, latest_block) {
54              self.blocks.push(block);
55          } else {
56              error!("Не вдалося додати блок - невірний");
57          }
58      }
59
```

Рисунок 3.3 – Функціонал додавання нових блоків

Метод «try_add_block» оголошений як частина структури «App». Він отримує змінний об'єкт «&mut self», що дозволяє змінювати стан об'єкта «App», та новий блок типу «Block», який додаємо до ланцюга.

Використовується метод last() для отримання посилання на останній блок у векторі «blocks». Якщо вектор порожній, метод «expect» викличе повідомленням «має бути хоча б один блок». Це означає, що функція очікує наявність хоча б одного блоку в ланцюзі.

Метод «is_block_valid» перевіряє, чи є новий блок валідним щодо останнього блоку. Це включає перевірку таких речей, як правильність геша, відповідність індексу, необхідних для валідного блоку. Якщо блок валідний, він додається до вектора blocks. Якщо блок не валідний, у журналі фіксується помилка з повідомленням «не вдалося додати блок – невірний».

Давайте також розглянемо утиліту для фактичного створення гешу SHA256.

```rust
fn calculate_hash(id: u64, timestamp: i64, previous_hash: &str, data: &str, nonce: u64) -> Vec<u8> {
    let data = serde_json::json!({
        "id": id,
        "previous_hash": previous_hash,
        "data": data,
        "timestamp": timestamp,
        "nonce": nonce
    });
    let mut hasher = Sha256::new();
    hasher.update(data.to_string().as_bytes());
    hasher.finalize().as_slice().to_owned()
}
```

Рисунок 3.4 – Обчислення гешу

Спочатку створюємо JSON-представлення даних блоку, використовуючи поточний nonce, і пропускаємо його через SHA256-хешер.

Якщо запитуємо нові блоки від інших вузлів, перевіряємо їх і, якщо вони в порядку, додаємо до ланцюжка. Якщо отримуємо повний блокчейн від іншого вузла, також перевіряємо його і, якщо він довший за наш (тобто містить більше блоків), замінюємо ним наш власний ланцюжок.

Оскільки кожен вузол реалізує саме таку логіку, блоки і узгоджені ланцюжки можуть швидко поширюватися мережею, і мережа сходиться до того ж стану.

3.2 Цифровий підпис

Цифровий підпис - це застосування криптографічного алгоритму, який базується на асиметричному шифруванні. За допомогою закритого ключа, що належить автору (він же - єдиний власник цього ключа), можна підписати будь-яке повідомлення. Зазвичай підпис здійснюється на гешованих даних, що містяться у повідомленні. Після цього будь-який користувач може за допомогою відкритого ключа (який доступний у публічному доступі) переконатися, що саме власник ключа підписав повідомлення. У випадку блокчейна користувач підписує будь-яку транзакцію, яка виходить від нього, своїм закритим ключем. Одержувач, так само як і будь-який інший учасник мережі, може розшифрувати транзакцію та переконатися, що вона справді

походить від цього відправника, використовуючи відкритий ключ, який наданий відправником [24].

За допомогою цифрових підписів неможливо витратити кошти користувача без його відома. Це відбувається тому, що без його закритого ключа транзакція не отримає правильного підпису і, отже, не буде прийнята мережею вузлів, що беруть участь у блокчейні. При цьому неможливість підробки підпису забезпечується складністю злому алгоритму асинхронного шифрування, який лежить в основі обраного методу підпису [22].

У механізмі Біткойн для забезпечення безпеки транзакцій застосовується алгоритм ECDSA, що розшифровується як Еліптична криптографія з цифровим підписом. Цей алгоритм включає в себе використання як відкритих, так і закритих ключів для створення електронних підписів.

Один із ключових етапів у процесі генерації відкритого ключа (рис. 3.5) в ECDSA включає в себе такі кроки:

1. Спочатку створюється секретний ключ.
2. Потім відбувається операція множення секретного ключа на точку генерації на еліптичній кривій.
3. У результаті виходить відкритий ключ.

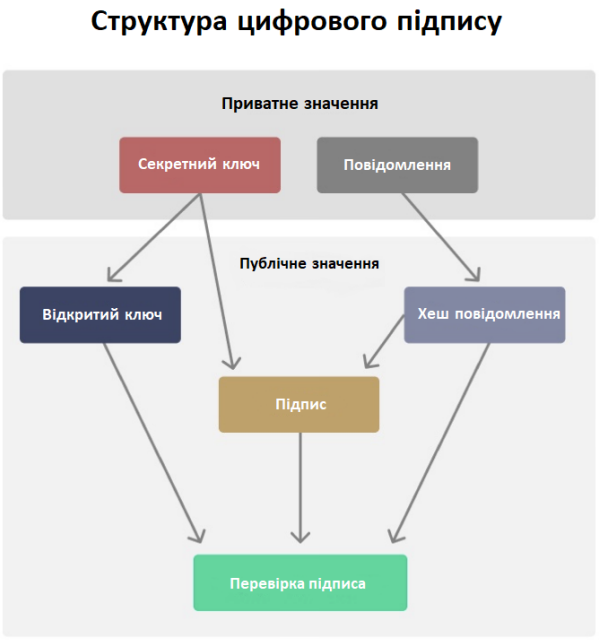

Рисунок 3.5 – Структура цифрового підпису

В контексті ECDSA множення має специфічний сенс - це точкове множення, яке відрізняється від звичайного математичного множення. Це також означає, що зворотне обчислення точкового ділення неможливе. Завдяки цій властивості, знаючи тільки відкритий ключ, неможливо отримати секретний, що є основою безпеки ECDSA.

У підсумку, ECDSA забезпечує захист транзакцій у мережі Біткойн, застосовуючи криптографічні ключі для створення електронних підписів. Секретність закритого ключа відіграє ключову роль у забезпеченні безпеки всієї системи.

```rust
use ed25519_dalek::{Keypair, Signature, Signer, Verifier, PUBLIC_KEY_LENGTH, SIGNATURE_LENGTH, SECRET_KEY_LENGTH};
use rand::rngs::OsRng;

fn main() {
    // Генеруємо нову пару ключів
    let mut csprng = OsRng {};
    let keypair: Keypair = Keypair::generate(&mut csprng);

    // Повідомлення, яке ми хочемо підписати
    let message: &[u8] = b"Hello, blockchain!";

    // Створюємо підпис
    let signature: Signature = keypair.sign(message);

    // Перевіряємо підпис
    let verified = keypair.public.verify(message, &signature).is_ok();

    println!("Публічний ключ: {:?}", keypair.public);
    println!("Підпис: {:?}", signature);
    println!("Перевірка підпису пройшла успішно: {}", verified);
}
```

Рисунок 3.6 – Код цифрового підпису

Цей код починається з імпорту необхідних модулів з бібліотек `ed25519_dalek` та `rand`. Модулі з `ed25519_dalek` забезпечують необхідні криптографічні функції для створення та перевірки цифрових підписів, а `rand` використовується для генерації криптографічно безпечних випадкових чисел. Спочатку код генерує нову пару ключів, яка складається з секретного і публічного ключів. Це робиться за допомогою методу `Keypair::generate(&mut csprng)`, де `csprng` є криптографічно безпечним генератором випадкових чисел, створеним з використанням `OsRng`. Генерація ключів є основоположним кроком для створення цифрових підписів, оскільки

секретний ключ використовується для підписання повідомлень, а публічний ключ – для їх перевірки.

Далі, код визначає повідомлення, яке потрібно підписати, у вигляді байтового масиву `b"Hello, blockchain!"`. Повідомлення може бути будь-яким набором даних, який потрібно захистити цифровим підписом. Після цього створюється підпис для цього повідомлення за допомогою методу `keypair.sign(message)`, який використовує секретний ключ згенерованої пари ключів. Підпис є криптографічним перетворенням повідомлення і секретного ключа, що дозволяє перевірити автентичність і цілісність повідомлення.

Останнім кроком є перевірка підпису. Це робиться за допомогою публічного ключа і методу `keypair.public.verify(message, &signature)`. Метод `verify` перевіряє, чи відповідає підпис повідомленню і публічному ключу. Якщо підпис правильний, метод повертає `Ok(())`, інакше – помилку. У коді результат перевірки зберігається в змінній `verified`, яка потім використовується для виведення результату перевірки. В кінці програми виводяться публічний ключ, підпис і результат перевірки підпису. Цей приклад демонструє основні кроки, необхідні для створення і перевірки цифрових підписів, що є важливим для забезпечення безпеки даних у блокчейн-системах.

3.3. Основна частина додатку

Почнемо зі створення файлу p2p.rs, який міститиме більшу частину піринговой логіки, яку ми будемо використовувати в нашому додатку.

Там же ми визначимо деякі основні структури даних і константи, які нам знадобляться.

Починаючи зверху, визначаємо пару ключів і похідний ідентифікатор однорангової мережі. Це властивості libp2p для ідентифікації клієнта в мережі.

Потім визначимо дві так звані теми: ланцюжки і блоки. Для зв'язку між вузлами ми будемо використовувати протокол «FloodSub», простий протокол публікації/підписки.

Його перевага полягає в тому, що він дуже простий у налаштуванні та використанні, але його недолік полягає в тому, що потрібно транслювати кожну частину інформації. Тож навіть якщо хочемо відповісти на «запит нашого ланцюжка» одного клієнта, цей клієнт надішле цей запит усім вузлам, до яких він підключений у мережі, і також надішлемо нашу відповідь усім їм. Це не є проблемою з точки зору коректності, але з точки зору ефективності ні. З цим можна було б впоратися за допомогою моделі запиту/відповіді «точка-точка», яку підтримує libp2p, але це просто додасть ще більшої складності до цього і без того складного прикладу.

Також можливе використання більш ефективний GossipSub замість FloodSub. Але його не так зручно налаштовувати, і проект не особливо зацікавлений в продуктивності на даному етапі.

У будь-якому випадку, теми - це в основному «канали», на які можна підписатися. Ми можемо підписатися на «ланцюжки» і використовувати їх, щоб відправляти наш локальний блокчейн на інші вузли і отримувати їхні id. Те ж саме стосується і «блоків», які використовуваються для трансляції та отримання нових блоків.

Концепція ChainResponse, що містить список блоків і одержувача. Це структура, якщо хтось надішле нам свій локальний блокчейн, і використовувати, щоб відправити йому наш локальний ланцюжок.

LocalChainRequest - це те, що запускає цю взаємодію. Якщо відправимо LocalChainRequest з peer_id іншого вузла в системі, це призведе до того, що він надішле нам свій ланцюжок назад, як ми побачимо пізніше.

Для обробки вхідних повідомлень, лінивої ініціалізації та введення даних з клавіатури користувачем клієнта ми визначаємо зчислення EventType, яке допоможе нам надсилати події по всьому додатку, щоб синхронізувати стан додатку з вхідним та вихідним мережевим трафіком.

Нарешті, ядром P2P-функціональності є AppBehaviour, який реалізує NetworkBehaviour, концепцію libp2p для реалізації децентралізованого мережевого стеку.

AppBehaviour містить екземпляр FloodSub для зв'язку pub/sub і екземпляр Mdns, який дозволить автоматично знаходити інші вузли в локальній мережі, але не за її межами.

Також додаємо до цієї поведінки блокчейн-додаток, а також канали для надсилання подій як для ініціалізації, так і для обміну запитами/відповідями між частинами додатку.

```rust
pub static KEYS: Lazy = Lazy::new(identity::Keypair::generate_ed25519);
pub static PEER_ID: Lazy = Lazy::new(|| PeerId::from(KEYS.public()));
pub static CHAIN_TOPIC: Lazy = Lazy::new(|| Topic::new("chains"));
pub static BLOCK_TOPIC: Lazy = Lazy::new(|| Topic::new("blocks"));

#[derive(Debug, Serialize, Deserialize)]
pub struct ChainResponse {
    pub blocks: Vec,
    pub receiver: String,
}

#[derive(Debug, Serialize, Deserialize)]
pub struct LocalChainRequest {
    pub from_peer_id: String,
}

pub enum EventType {
    LocalChainResponse(ChainResponse),
    Input(String),
    Init,
}

#[derive(NetworkBehaviour)]
pub struct AppBehaviour {
    pub floodsub: Floodsub,
    pub mdns: Mdns,
    #[behaviour(ignore)]
    pub response_sender: mpsc::UnboundedSender,
    #[behaviour(ignore)]
    pub init_sender: mpsc::UnboundedSender,
    #[behaviour(ignore)]
    pub app: App,
}
```

Рисунок 3.7 – Мережеві налаштування P2P

По-перше, реалізуємо обробники для даних, що надходять з інших вузлів. Якщо виявлено новий вузол, додаємо його до нашого списку вузлів FloodSub, щоб вони могли спілкуватися. Після закінчення терміну дії,

видаляємо його знову. Більш цікавою є реалізація NetworkBehaviour для протоколу зв'язку FloodSub.

Для вхідних подій, які є FloodsubEvent::Message, перевіряємо, чи відповідає корисне навантаження будь-якій з наших очікуваних структур даних. Якщо це ChainResponse, це означає, що отримали локальний блокчейн від іншого вузла. Перевіряємо, чи дійсно є одержувачем цього блоку даних, і якщо так, то реєструємо вхідний блокчейн і намагаємося виконати консенсус.

```
impl NetworkBehaviourEventProcess for AppBehaviour {
    fn inject_event(&mut self, event: FloodsubEvent) {
        if let FloodsubEvent::Message(msg) = event {
            if let Ok(resp) = serde_json::from_slice::(&msg.data) {
                if resp.receiver == PEER_ID.to_string() {
                    info!("Response from {}:", msg.source);
                    resp.blocks.iter().for_each(|r| info!("{:?}", r));

                    self.app.blocks = self.app.choose_chain(self.app.blocks.clone(), resp.blocks);
                }
            } else if let Ok(resp) = serde_json::from_slice::(&msg.data) {
                info!("sending local chain to {}", msg.source.to_string());
                let peer_id = resp.from_peer_id;
                if PEER_ID.to_string() == peer_id {
                    if let Err(e) = self.response_sender.send(ChainResponse {
                        blocks: self.app.blocks.clone(),
                        receiver: msg.source.to_string(),
                    }) {
                        error!("error sending response via channel, {}", e);
                    }
                }
            } else if let Ok(block) = serde_json::from_slice::(&msg.data) {
                info!("received new block from {}", msg.source.to_string());
                self.app.try_add_block(block);
            }
        }
    }
}
```

Рисунок 3.8 – Обробка вхідних повідомлень

Якщо він дійсний і довший за ланцюжок, замінюємо ним наш ланцюжок. В іншому випадку, зберігаємо наш власний ланцюжок.

Якщо вхідні дані є LocalChainRequest, перевіряємо, чи не від нас вони хочуть отримати ланцюжок, перевіряючи from_peer_id. Якщо так, просто відправляємо їм JSON-версію локального блокчейну.

Якщо надходить блок, це означає, що хтось інший здобув блок і хоче, щоб додали його до нашого локального ланцюжка. Перевіряємо, чи блок дійсний, і якщо так, то додаємо його.

Спочатку було задано константу «DIFFICULTY_PREFIX». Вона є основою майнінгової схеми. Під час майнінгу блоку майнер повинен гешувати дані блоку, а саме за допомогою SHA256 і знайти геш, який у двійковому форматі починається з "00" (двох нулів). Це визначає рівень складності в мережі.

```
77
78  const DIFFICULTY_PREFIX: &str = "00";
79
80  fn hash_to_binary_representation(hash: &[u8]) -> String {
81      let mut res: String = String::default();
82      for c in hash {
83          res.push_str(&format!("{:b}", c));
84      }
85      res
86  }
87
88  impl App {
89      fn is_block_valid(&self, block: &Block, previous_block: &Block) -> bool {
90          if block.previous_hash != previous_block.hash {
91              warn!("блок з id: {} має неправильний попередній хеш", block.id);
92              return false;
93          } else if !hash_to_binary_representation(
94              &hex::decode(&block.hash).expect("може декодувати з hex"),
95          )
96          .starts_with(DIFFICULTY_PREFIX)
97          {
98              warn!("блок з id: {} має невірну складність", block.id);
99              return false;
100         } else if block.id != previous_block.id + 1 {
101             warn!(
102                 "блок з id: {} не є наступним блоком після останнього: {}",
103                 block.id, previous_block.id
104             );
105             return false;
106         } else if hex::encode(calculate_hash(
107             block.id,
108             block.timestamp,
109             &block.previous_hash,
110             &block.data,
111             block.nonce,
112         )) != block.hash
113         {
114             warn!("блок з id: {} має невірний хеш", block.id);
115             return false;
116         }
117         true
118     }
119 }
120
```

Рисунок 3.9 – Основна частина логіки

Для знаходження відповідного гешу значно збільшується, якщо нам потрібно три, чотири, п’ять або навіть 20 початкових нулів. У справжній блокчейн-системі цей рівень складності є параметром мережі, який узгоджується між вузлами на основі алгоритму консенсусу та залежить від потужності гешування мережі, що дозволяє мережі гарантувати створення нового блоку за певний проміжок часу.

Для спрощення було встановлено складність на рівні двох початкових нулів. Це не займає занадто багато часу для обчислення на звичайному обладнанні, тому не потрібно турбуватися про довге очікування під час тестування.

Допоміжна функція, яка є просто двійковим представленням заданого байтового масиву у вигляді рядка. Це використовується для того, щоб зручно перевіряти, чи відповідає геш нашій умові «DIFFICULTY_PREFIX».

Тепер перейдемо до логіки валідації блоку. Це важливо, тому що вона гарантує, що наш блокчейн дотримується своїх властивостей ланцюжка і його важко підробити. Складність зміни чогось зростає з кожним блоком, оскільки вам доведеться перерахувати (тобто, заново видобути) решту ланцюжка, щоб знову отримати дійсний ланцюжок.

Існує кілька емпіричних правил, яких слід дотримуватися:

- Попередній_хеш повинен фактично відповідати гешу останнього блоку в ланцюжку.
- Хеш повинен починатися з нашого «DIFFICULTY_PREFIX» (тобто з двох нулів), який вказує на те, що він був видобутий правильно.
- Ідентифікатор повинен бути останнім ідентифікатором, збільшеним на 1.
- Хеш повинен бути дійсно правильним; гешування даних блоку повинно дати нам геш блоку (в іншому випадку, ви можете просто створити випадковий геш, що починається з 001).

Якщо два вузли одночасно створюють блок з ідентифікатором 5, кожен з них створить наступний блок з ідентифікатором 6, посилаючись на блок 5 як на попередній. В результаті буде отримано обидва блоки. Під час перевірки буде додаватися один з них, але другий буде відхилений, оскільки вже існує блок з ідентифікатором 6. Ця проблема є невід'ємною частиною системи, тому для вирішення таких конфліктів потрібен алгоритм консенсусу, який допомагає вузлам домовитися про те, які блоки (тобто, який ланцюг) приймати і використовувати.

В оптимальному випадку, якщо блок, який здобули, не буде додано до узгодженого ланцюжка, доведеться добути його ще раз і сподіватися, що наступного разу він буде працювати краще. У простому випадку цей механізм повторної спроби не буде реалізовано, якщо така гонка відбудеться, цей вузол по суті вийде з гри.

```rust
impl App {

    fn is_chain_valid(&self, chain: &[Block]) -> bool {
        for i in 0..chain.len() {
            if i == 0 {
                continue;
            }
            let first = chain.get(i - 1).expect("has to exist");
            let second = chain.get(i).expect("has to exist");
            if !self.is_block_valid(second, first) {
                return false;
            }
        }
        true
    }
        // Ми завжди вибираємо найдовший дійсний ланцюжок
    fn choose_chain(&mut self, local: Vec, remote: Vec) -> Vec {
        let is_local_valid = self.is_chain_valid(&local);
        let is_remote_valid = self.is_chain_valid(&remote);

        if is_local_valid && is_remote_valid {
            if local.len() >= remote.len() {
                local
            } else {
                remote
            }
        } else if is_remote_valid && !is_local_valid {
            remote
        } else if !is_remote_valid && is_local_valid {
            local
        } else {
            panic!("локальні та віддалені ланцюги є недійсними");
        }
    }
}
```

Рисунок 3.10 – Логіка ланцюга

Ігноруючи блок генезису, ми просто проходимо всі блоки і перевіряємо їх. Якщо один блок не пройшов валідацію, ми не пройдемо весь ланцюжок.

Це відбувається, якщо ми запитуємо у іншого вузла його ланцюжок, щоб визначити, чи є він "кращим" (згідно з нашим алгоритмом консенсусу) за локальний ланцюжок.

Критерій - це просто довжина ланцюжка. У реальних системах, як правило, існує більше факторів, таких як складність і багато інших можливостей. Для цілей цієї вправи, якщо один ланцюжок (валідний) довший за інший, то беремо його.

Перевіряється локальний та віддалений ланцюг, і береться довший. Цей функціонал буде використовуватися під час запуску, коли буде запит на інші вузли про їхні ланцюжки. Оскільки блок включає лише блок генезису, то одразу почнеться з "узгодженим" ланцюжком.

```
impl Block {
    pub fn new(id: u64, previous_hash: String, data: String) -> Self {
        let now = Utc::now();
        let (nonce, hash) = mine_block(id, now.timestamp(), &previous_hash, &data);
        Self {
            id,
            hash,
            timestamp: now.timestamp(),
            previous_hash,
            data,
            nonce,
        }
    }
}
```

Рисунок 3.11 – Створення нового блоку

Коли новий блок створено, ми викликаємо функцію «mine_block», яка повертає nonce та геш. Після цього ми можемо створити блок з його міткою часу, даними, ID, попере

Спочатку оголошуємо про видобування блоку, встановлюємо nonce в 0. Потім запускається нескінченний цикл, який збільшує nonce на кожному

кроці. Усередині циклу, записуємо кожні 100000 ітерацій, щоб мати приблизний індикатор прогресу, обчислюємо геш над даними блоку за допомогою функції «calculate_hash».

Потім використовуємо помічник «hash_to_binary_representation» і перевіряємо, чи відповідає обчислений геш критерію складності, який полягає в тому, що він починається з двох нулів.

нім гешем, новим гешем і nonce.

```
fn mine_block(id: u64, timestamp: i64, previous_hash: &str, data: &str) -> (u64, String) {
    info!("mining block...");
    let mut nonce = 0;

    loop {
        if nonce % 100000 == 0 {
            info!("nonce: {}", nonce);
        }
        let hash = calculate_hash(id, timestamp, previous_hash, data, nonce);
        let binary_hash = hash_to_binary_representation(&hash);
        if binary_hash.starts_with(DIFFICULTY_PREFIX) {
            info!(
                "mined! nonce: {}, hash: {}, binary hash: {}",
                nonce,
                hex::encode(&hash),
                binary_hash
            );
            return (nonce, hex::encode(hash));
        }
        nonce += 1;
    }
}
```

Рисунок 3.12 – Майнінг блоку

Якщо так, записуємо його в журнал і повертаємо nonce, ціле число, що збільшується, де це сталося, і геш закодований у шістнадцятковому форматі. В іншому випадку, збільшуємо nonce і робимо це знову.

Намагаємося знайти фрагмент даних - в даному випадку, nonce і число, які разом з нашими блоковими даними, гешованими за допомогою SHA256, дадуть нам геш, що починається з двох нулів.

Потрібно записати цей nonce в блоці, щоб інші вузли могли перевірити геш, оскільки nonce гешується разом з даними блоку. Наприклад, якщо знадобиться 52 342 ітерації для обчислення відповідного гешу (починаючи з двох нулів), то nonce буде 52341 (на 1 менше, оскільки він починається з 0).

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було проведено аналіз криптографічних інструментів у контексті безпеки технології блокчейну та виявлено їхню важливу роль у забезпеченні надійності, конфіденційності та цілісності даних в блокчейн-мережах.

Криптографічні методи, такі як геш-функції, цифрові підписи та система proof of work, використовуються для захисту блоків і транзакцій в блокчейні від несанкціонованого доступу та змін. Вони дозволяють підтверджувати автентичність та цілісність даних, а також забезпечують конфіденційність особистої інформації.

При аналізі криптографічних інструментів у контексті безпеки технології блокчейну було виявлено, що використання сучасних криптографічних методів дозволяє підвищити рівень безпеки мережі блокчейн. Однак, необхідно також ретельно оцінювати інші аспекти безпеки, такі як захист приватності даних, запобігання атакам 50%+1, а також враховувати потенційні вразливості в реалізації криптографічних протоколів.

Практична реалізація включала дослідження використання гешування, цифрового підпису та механізму підтвердження роботи (Proof of Work) на мові програмування Rust. Ці інструменти грають ключову роль у забезпеченні безпеки блокчейну. Важливість цифрового підпису полягає у забезпеченні автентичності та цілісності даних, тоді як Proof of Work гарантує, що транзакції є достовірними та незмінними.

У підсумку, криптографічні інструменти грають критичну роль у забезпеченні безпеки та довіри до технології блокчейну. Подальший розвиток та вдосконалення цих інструментів допоможуть підвищити ефективність та надійність блокчейн-рішень у майбутньому.

СПИСОК ЛІТЕРАТУРИ

1. Блокчейн. URL: https://en.wikipedia.org/wiki/Blockchain
2. Блокчейн. Ланцюжок блоків. URL: https://www.researchgate.net/figure/Blocks-are-cryptographically-linked-together-Blockchains-are-collection-of-blocks-A_fig1_341788606
3. Вступ до Rust для розробки блокчейну. URL: https://education.web3.foundation/docs/introrust
4. Де застосовується блокчейн. URL: https://finances.in.ua/de-zastosovuietsia-blokchejn-vykorystannia-tekhnolohii-blokchejn-u-riznykh-sferakh/
5. Дослідження систем блокчейну. URL: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://jarch.donnu.edu.ua/article/view/12652/12552&ved=2ahUKEwjFj4SBye6FAxWlavEDHXRxCWUQFnoECBEQAQ&usg=AOvVaw0tl5OeiMW9UTViska9uvKs
6. Майнери у блокчейні. URL: https://www.blockchain-council.org/blockchain/blockchain-mining-a-comprehensive-step-by-step-guide/
7. Мова програмування Rust. URL: https://www.rust-lang.org/
8. Найкращі 5 блокчейнів, що використовують мову програмування Rust. URL:https://academy.moralis.io/blog/top-blockchains-using-the-rust-programming-language
9. Переваги та недоліки блокчейна. URL: https://www.forbes.com/sites/forbestechcouncil/2022/10/20/advantages-and-disadvantages-of-blockchain-technology/?sh=27cc77603453
10. Переваги та недоліки блокчейна. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-blockchain/
11. Пояснення Peer-to-Peer мереж. URL: https://academy.binance.com/uk/articles/peer-to-peer-networks-explained
12. Пояснення підтвердження роботи. URL: https://www.forbes.com/advisor/investing/cryptocurrency/proof-of-work/

13. Програмування Rust Блокчейн. URL: https://www.gyata.ai/rust/rust-programming-blockchain/
14. Реалізація блокчейну в Rust. URL: https://blog.devgenius.io/implementing-a-blockchain-in-rust-28cce9a3c968
15. Створення блокчейну в Rust & Substrate. URL: https://hackernoon.com/building-a-blockchain-in-rust-and-substrate-a-step-by-step-guide-for-developers-kc223ybp
16. Створення першого блокчейну Rust: Покрокове керівництво. URL: https://webisoft.com/articles/rust-blockchain/
17. Технологія блокчейн та її роль в екосистемі Bitcoin. URL: https://lb.ua/economics/2023/04/13/550425_tehnologiya_blokcheyn_ii_rol.html
18. Що таке блокчейн? URL: https://artline.ua/uk/news/chto-takoe-blokcheyn
19. Що таке підтвердження роботи (PoW) у блокчейні? URL: https://blockworks.co/news/what-is-proof-of-work
20. Як побудувати блокчейн з нуля в Rust. URL: https://dev.to/ecj222/how-to-build-a-blockchain-from-scratch-in-rust-46
21. Як працює майнінг біткоїнів? URL: https://www.investopedia.com/tech/how-does-bitcoin-mining-work/
22. Як працюють цифрові підписи в блокчейні. URL: https://ubc.digital/how-digital-signatures-work-in-blockchain/
23. Як створити блокчейн в Rust. URL: https://blog.logrocket.com/how-to-build-a-blockchain-in-rust/
24. Blockchain Technology Needs to Be Changing Education. URL: https://medium.com/age-of-awareness/blockchain-technologyneeds-to-be-changing-education-28324281e2
25. Ethereum для Rust розробників. URL: https://ethereum.org/en/developers/docs/programming-languages/rust/