Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ БЛОКЧЕЙН-ТЕХНОЛОГІЇ ДЛЯ ПІДТВЕРДЖЕННЯ ОРИГІНАЛЬНОСТІ ДОКУМЕНТІВ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-21-1
Яресько К. В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Яресько Ксенії Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження блокчейн-технології для підтвердження оригінальності документів»

затверджена наказом по університету від 9 листопада 2022 року № 1469Ст

2. Термін подання студентом роботи до екзаменаційної комісії 21 листопада 2022 р.

3. Вихідні дані до роботи технології блокчейну та сфери їх застосування, математичні моделі оброблення даних, програмні засоби платформи Ethereum, застосування для оброблення документів.

4. Перелік питань, що потрібно опрацювати в роботі

1. Дослідити особливості технології блокчейн та платформи Ethereum.

2. Розробити смарт-контракт для процесів додавання документів до блокчейну та їх перевірки на оригінальність.

3. Розроблення вебінтерфейсу для роботи зі спеціалізованим смарт- контрактом.

4. Розрахунок обчислювальних затрат у вигляді вартості газу для деплою смарт-контракту та проведення транзакцій для збереження та перевірки документів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність теми, сфери застосування блокчейну, постановка задачі, опис вхідних даних, схема аналізу даних, результати тестування та дослідження.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	09.11.2022	
2	Аналіз завдання, підбір літератури	09.11.22-10.11.22	
3	Аналіз літератури з досліджуваної проблеми	10.11.22-11.11.22	
4	Аналіз та вибір технологій	11.11.22-15.11.22	
5	Обгрунтування методу та засобів	15.11.22-17.11.22	
6	Програмна реалізація	17.11.22-20.11.22	
7	Оформлення пояснювальної записки	20.11.22-23.11.22	
8	Перевірка на плагіат	23.11.2022	
9	Рецензування	25.11.2022	
10	Підготовка презентації та доповіді	27.11.2022	
11	Занесення роботи в електронний архів	28.11.2022	
12	Попередній захист кваліфікаційної роботи	30.11.2022	

Дата видачі завдання 9 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ _проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 69 с., 5 табл., 31 рис., 43 джерела.

BLOCKCHAIN, КРИПТОГРАФІЯ, ХЕШУВАННЯ, СМАРТ-КОНТРАКТ, ETHEREUM, JAVASCRIPT, ВЕБЗАСТОСУНОК, ОРИГІНАЛЬНІСТЬ ДОКУМЕНТУ.

Об’єктом дослідження є прикладний смарт-контракт для зберігання та перевірки істинності даних про документи.

Метою дослідження є розробка та дослідження вебзастосунок для підтвердження оригінальності документів із застосуванням блокчейн-технології Ethereum.

Шляхом тестування проведено вивчення працездатності розробленого вебзастосунку, здійснено розрахунок витрат при його впровадженні.

У результаті проведеного дослідження та тестування смарт-контракту та вебзастосунку підтверджено працездатність та результативність програмного засобу для збереження і перевірки оригінальності документів.

BLOCKCHAIN, POW, POS, CRYPTOGRAPHY, HASH, SMART CONTRACT, ETHEREUM, SOLIDITY, KECCAK, JAVASCRIPT, WEB3, WEB-APP, DAPP.

The object of the research is a smart contract for storing, verifying and authentication of document data.

The aim of the research is to вevelop and research a web application for document authenticity verification using Ethereum blockchain technology.

The developed web application performance was tested, researched, and the implementation costs were calculated.

The result of the research and testing confirmed the functionality and effectiveness of the smart contract and the web application for verifying the documents.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів……..6
Вступ……7
1 Аналіз і суть технології блокчейн……...9
1.1 Взаємодія централізованих та децентралізованих структур даних…….9
1.2 Суть блокчейн-технології…….. 12
1.3 Застосування хешування та криптографії у блокчейні……... 19
1.4 Аналіз зображень і блокчейн……24
1.5 Постановка задачі дослідження…… 26
2 Технологія Ethereum та смарт-контракти……28
2.1 Віртуальна машина Ethereum та смарт-контракти…….. 28
2.2 Транзакція Ethereum і структура повідомлень…….32
2.3 Криптографія в Ethereum. Алгоритм Keccak……... 36
3 Дослідження результативності застосунку для підтвердження оригінальності документів……48
3.1 Створення аккаунту в Ethereum…… 48
3.2 Розроблення прикладного смарт-контракту……52
3.3 Розроблення клієнтської програми (вебзастосунок)……... 56
3.4 Розрахунок витрат у мережі Ethereum……...59
Висновки…… 64
Перелік джерел посилання……... 65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

API – Application Programming Interface (програмний інтерфейс програми)

P2P – Peer to Peer, Person to Person (від людини до людини)

TCP/IP – Transmission Control Protocol/Internet Protocol – (протокол керування передачею / Інтернет-протокол)

EVM – Ethereum Virtual Machine (віртуальна машина Ethereum)

EOA – Externally Owned Accounts (зовнішні облікові записи)

DApp – Decentralized Applications (децентралізовані програми)

NFT – Non Fungible Token (Невзаємозамінні токени)

ВСТУП

Блокчейн на початку свого виникнення був фінансовим інструментом, який забезпечував можливість для безпечних, швидких та однорангових транзакцій з даними. Та по мірі свого розвитку блокчейн став використовуватися в більш широких сферах людської діяльності. Наразі існують програмні застосунки на основі блокчейн для криптовалют, охорони здоров'я, смарт-контрактів, Інтернету речей, управління розумними мережами, ланцюга постачання тощо [1-7]. Блокчейн забезпечує креативний та надійно захищений підхід до зберігання інформації, виконання транзакцій, завдань і побудови довіри.

Сучасні програмні застосунки базуються на властивостях конфіденційності, захищеності та водночас відкритості блокчейну. Блокчейн - технологія, що лежить в основі валюти біткойн, є децентралізованою книгою обміну, яка записує дані від різних сторін, які беруть участь у транзакціях мережі біткойн. Мережа Bitcoin, зокрема, використовує блокчейн для зберігання історії транзакцій, а також іншої інформації, пов'язаної з транзакціями, такої як час завершення транзакції, адреса відправника (або споживача) та адреса одержувача. Це допоможе тим, хто витрачає гроші, уникнути подвійних витрат. Щоб забезпечити конфіденційність блокчейн, уся інформація зашифрована. Блокчейн також можна визначити як спільну книгу, оскільки вона містить всю інформацію про всі транзакції [4-5].

Світ переходить у сучасну епоху, технології стрімко змінюють життя людей. Через багаточисельну мобільність та збільшення числа працівників та іноземних студентів у всьому світі, а також поточну проблему підробки дипломів та продажу фальшивих сертифікатів про освіту пропонуються сучасні технологічні рішення [1-3]. Поєднання технології блокчейн із базою інформації про сертифікати та дипломи про вищу освіту, дає можливість

зробити процес перевірки оригінальності чи істинності документів більш спрощеним та прозорим.

Така система ідентифікації освітніх документів може бути корисною для роботодавців, випускників та навчальних закладів, які оцінюють абітурієнтів. Вона спрощує життя студентам, які подають документи на міжнародні програми. Існує посилений риск, щоб забезпечити легітимність і справжність документів, сертифікатів і дипломів, і бажано без поточного «клопоту» щодо визнання їх офіційними установами. Нові технологічні досягнення з розвитком блокчейну та смарт-контрактів з їхніми характеристиками незмінності, децентралізації, безпеки, відстежуваності та консенсусу можна вважати вдалим підходом для впровадження надійного рішення для боротьби з шахрайством при видачі цифрових документів. Інновації, такі як зв’язування блокчейну та бази сертифікатів чи дипломів, передбачають значні зміни та новизну. Поєднання блокчейну та документів потенційно може позитивно вплинути та принести користь мільйонам людей у всьому світі, особливо молодому поколінню [2-5].

Не можно говорити про використання технологій, не обговорюючи питання захисту. Недотримання належних процедур захисту призведе до збільшення використання фінансових і людських ресурсів.

У результаті виконання кваліфікаційної роботи буде запропоновано програмну структуру на основі блокчейну для безпечного та надійного управління даними для документів про вищу освіту і не тільки.

Додатково можно стверджувати, що наша тема привертає увагу та зусилля дослідників у всьому світі, а деякі вищі навчальні заклади вже зараз впроваджують спеціальні рішення. Насправді на сьогодні бракує єдиної відповіді на результативне вирішення проблеми автоматичної та надійної сертифікації документів, дипломів.

1 АНАЛІЗ І СУТЬ ТЕХНОЛОГІЇ БЛОКЧЕЙН

1.1 Взаємодія централізованих та децентралізованих структур даних

Причина, по якій треба розглянути сумісно поняття централізації та децентралізації, полягає в тому, що блокчейн розроблено для децентралізації, кидаючи виклик централізованому дизайну. Однак терміни децентралізації та централізації не завжди чіткі. Вони дуже погано визначені та в багатьох місцях вводять в оману. Причина в тому, що практично не існує реальної системи, яка б була чисто централізованою чи децентралізованою [3].

Централізована розподілена система – це система, в якій є головний вузол, відповідальний за розподіл завдань або даних і розподіл навантаження між вузлами. З іншого боку, децентралізована розподілена система – це система, де немає «головного» вузла як такого, але обчислення можуть бути розподіленими. Блокчейн є одним із таких прикладів.

Централізована/децентралізована система не обмежується лише технічною архітектурою. Система може бути централізованою чи децентралізованою технічно, але може бути централізована логічно, політично або в іншому сенсі.

Технічна архітектура: враховується, скільки фізичних комп’ютерів (або вузлів) використовується для розробки системи, скільки вузлів вона може втратити, перш ніж вся система вийде з ладу тощо.

Політична перспектива: ця перспектива вказує на контроль, який особа, або група людей, або організація в цілому має над системою. Якщо комп’ютери системи управляються ними, то система природно централізована. Однак, якщо жодна конкретна особа чи групи не контролюють систему і всі мають рівні права в системі, тоді це децентралізована система в політичному сенсі.

Логічна перспектива: система може бути логічно централізованою і децентралізованою залежно від того, як вона виглядає, незалежно від того, централізованою чи децентралізованою вона є технічно чи політично. Альтернативна аналогія може полягати в тому, що якщо ви вертикально розріжте систему (скажімо, обчислювальні пристрої) навпіл, причому кожна половина має постачальників послуг і споживачів, якщо вони можуть працювати як незалежні одиниці, вони децентралізовані, а в іншому випадку централізовані.

Централізована система має централізований контроль з усіма адміністративними повноваженнями (рис. 1.1). Такі системи легко розробити, підтримувати, нав’язати довіру та керувати ними, але вони страждають від багатьох властивих обмежень, а саме:

– вони мають центральну точку відмови, тому менш стабільні;

– вони більш вразливі до атак і, отже, менш захищені;

– централізація влади може призвести до неетичних операцій;

– здебільшого таку систему важко масштабувати.

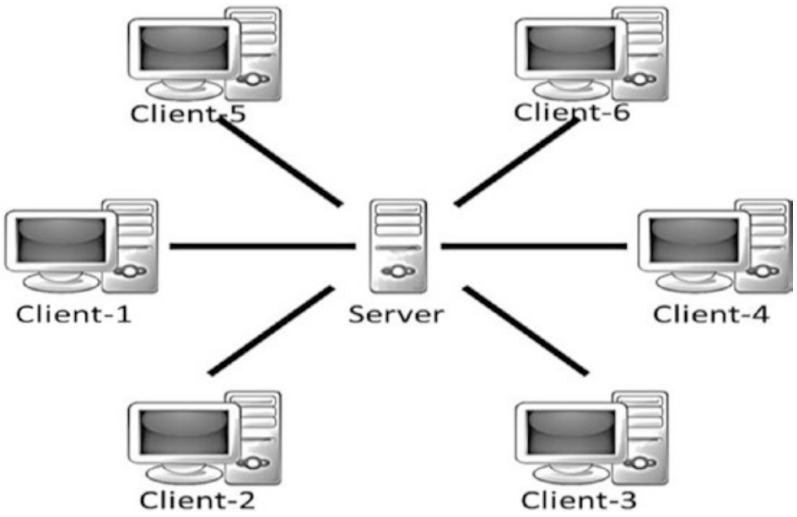


Рисунок 1.1 – Централізована система

Децентралізована система не має централізованого контролю, і кожен вузол має рівні повноваження. Такі системи складно проектувати,

підтримувати, керувати або нав’язати довіру. Однак вони не страждають від обмежень звичайних централізованих систем. Децентралізовані системи пропонують такі переваги:

– вони не мають центральної точки відмови, тому більш стабільні та відмовостійкі;

– стійкість до атак, оскільки немає центральної точки, щоб легко атакувати, і, отже, більш безпечна;

– симетрична система з рівними повноваженнями для всіх, тому здійснює – менше неетичних операцій і зазвичай демократична за своєю природою.

Розподілена система може бути децентралізованою, якою і є блокчейн! На відміну від звичайних розподілених систем, завдання не розбивається і не делегується вузлам, оскільки в блокчейні немає майстра, який би цим займався. Додаткові вузли не працюють над частиною роботи, скоріше, зацікавлені вузли (або вибрані навмання) виконують всю роботу.

На рисунку 1.2 наведена типова децентралізована та розподілена система, яка фактично є одноранговою (peer-to-peer) системою.

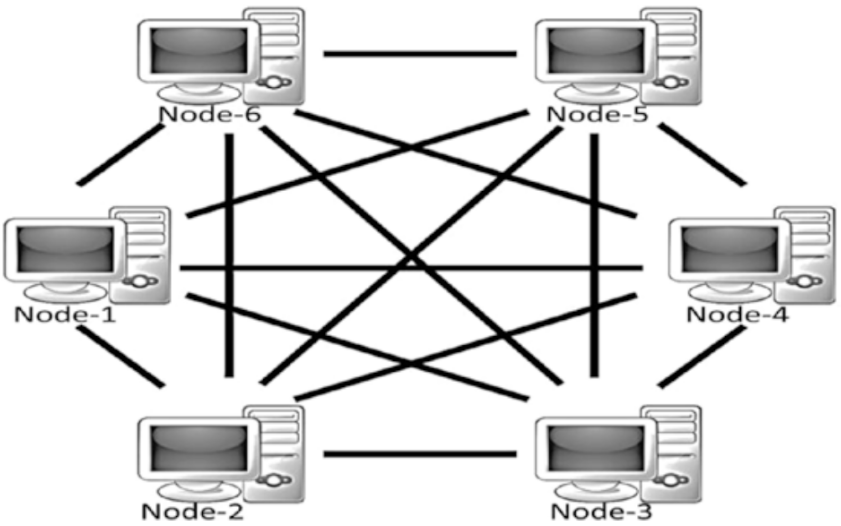


Рисунок 1.2 – Децентралізована однорангова система

1.2 Суть блокчейн-технології

Інтернет змінив багато аспектів життя, суспільства та бізнесу. Однак спосіб, у який люди та організації здійснюють транзакції один з одним, коли не перебувають поруч, за останні кілька десятиліть не сильно змінився. Для здійснення такої транзакції потрібна третя сторона, якій можна було б довіряти. Вважається, що блокчейн і є тим компонентом, який завершує головоломку Інтернету та робить його більш відкритим, доступнішим і надійнішим [1-3].

Блокчейн з точки зору бізнесу – це система записів для однорангової транзакції вартості. Це означає, що немає потреби в надійному посереднику, такому як банки, брокери чи інші служби умовного депонування, щоб служити надійною третьою стороною. Наприклад, якщо один користувач платить іншому користувачеві 10 доларів, чому це повинне проходити через банк? (рис. 1.3) [4]. Децентралізовані фінанси виділяються як альтернатива традиційним, оскільки вони можуть позбутися фінансової бюрократії, яка є тягарем сучасної фінансової системи. Децентралізовані фінанси дозволяють робити транзакції з мінімальними комісіями та майже миттєво у будь-які куточки світу.

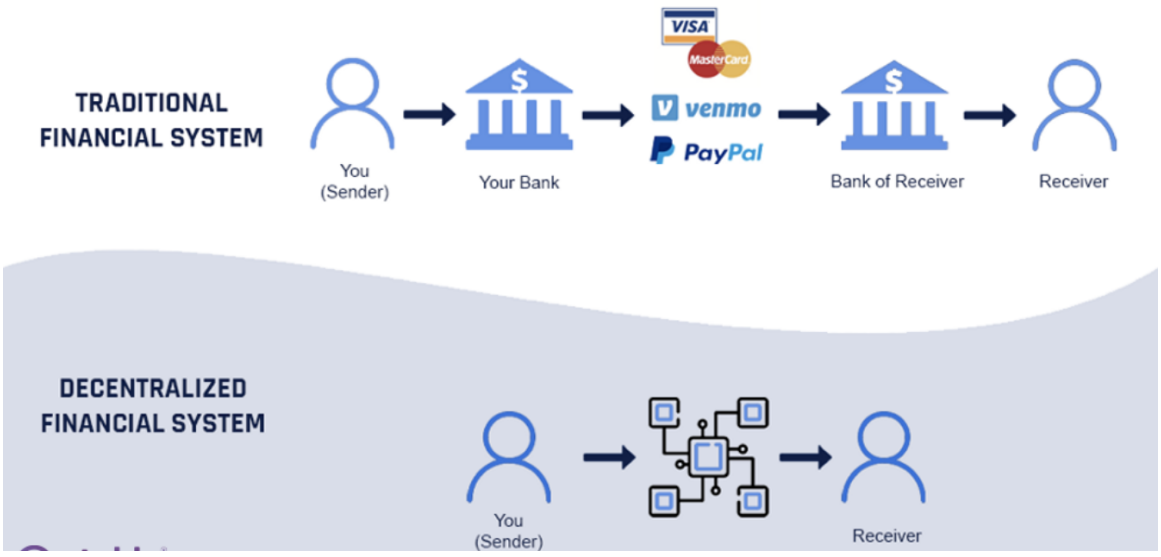


Рисунок 1.3 – Транзакція з посередником та P2P транзакція

Блокчейн – це тип зберігання даних, який дозволяє цифровим чином ідентифікувати та відслідкувати транзакції. Передаючи цю інформацію по розподіленій комп'ютерній мережі, створювати розподілену мережу довіри в якомусь сенсі [7].

З прикладної точки зору блокчейн – це інновація, що базується на трьох концепціях: мережі P2P, асиметричній криптографії та розподіленому консенсусі на основі вирішення математичних задач. Ці ідеї не є новими і розвиваються далі.

Блокчейн може розглядатися як синхронізована БД з такою кількістю копій, як і вузлів в мережі; як суперкомп'ютер, сформований складовими з усіх центральних та графічних процесорів, що належать до нього. Він використовується для збереження і обробки інформації, або як API. Різниця в тому, що не потрібно розробляти бекенд, і можно бути впевненими, що всі дані добре захищені та правильно оброблені в мережі (рис. 1.4). Можна припустити, що блокчейн – це журнал, в якому факти об'єднуються в ланцюг однотипних вузлів [8, 9].

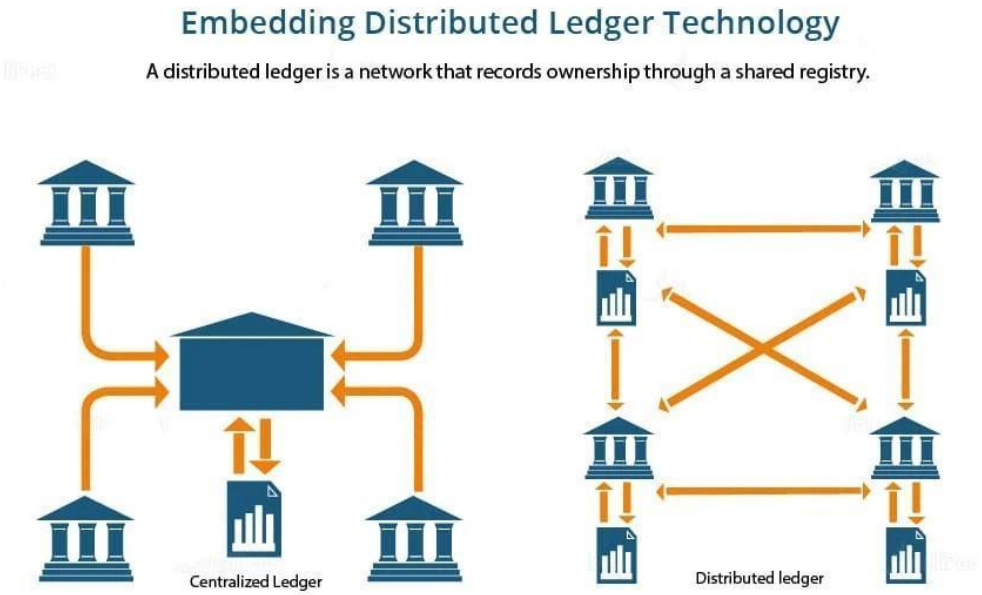


Рисунок 1.4 – Централізована та децентралізована архітектури

Мережі P2P та інші розподілені системи повинні розв’язувати непросту проблему: вирішення конфліктів або їх аналіз та узгодження. Реляційні БД забезпечують цілісність через посилання, але така функціональність не існує в розподілених системах. За умови, що 2 несумісні факти надходять одночасно, система повинна мати інструкцію, щоб визначити, який факт є правильним.

Сатоші Накамото запропонував в 2008 році концепцію – надання єдиної точки доступу, до великої кількості даних для всіх учасників. У 2009 році була перша реалізація на практиці – біткойн – цифрова валюта [2].

Дані в блокчейн збираються та поступово накопичуються, створюючи певний журнал з даними, що постійно доповнюється (рис. 1.5). В даний журнал не можна вносити які-небудь зміни або видаляти хоч якусь інформацію. Туди можна записати скільки завгодно транзакцій, що робить таку технологію унікальною.

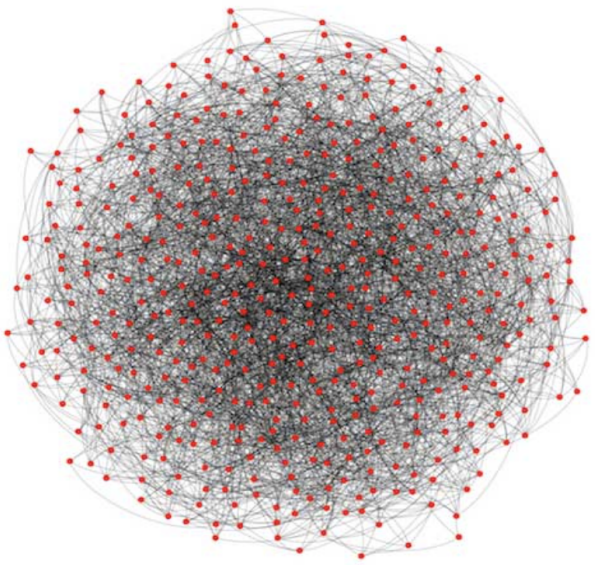


Рисунок 1.5 – Візуалізація децентралізованого Блокчейн

Можно сказати, що технологія блокчейн є гибрид, тобто поєднання централізованого та децентралізованого способів, але все одно основним буде скоріш за все один із типів. Наприклад, найважливіші вразливі дані

будуть зберігатися децентралізовано (у блокчейні), а якісь складні розрахунки чи якийсь невеликий обсяг даних, який дуже дорого записувати у блокчейн, може зберігатися десь на централізованому сервісі. Щоб затвердити транзакцію, її формат та підписи повинно бути перевірено, а потім записано в блок. Факти згруповуються в блоки, і в мережі залишається лише один блоковий ланцюжок. Кожен новий блок має посилання до попереднього блоку (рис. 1.6). Перш ніж додавати до блоку, факти розглядаються як неузгоджені.

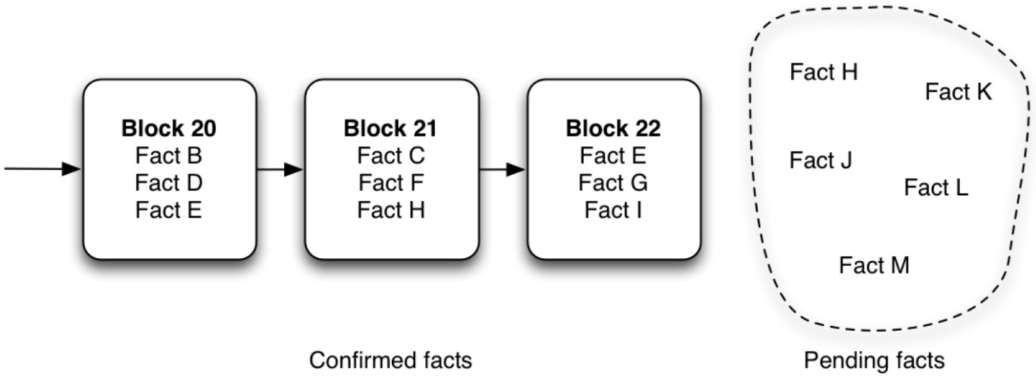


Рисунок 1.6 – Блоки в технології блокчейн

Основна концепція, яку розв’язує блокчейн, насамперед зосереджена на забезпеченні приватного та безпечного спілкування між двома сторонами. Насамперед це було зумовлено першим впровадженням технології блокчейн для цифрової валюти, і це не випадково, оскільки гроші все-таки є комунікаційною технологією, оскільки обмін грошима залежить від комунікації.

Таким чином, блокчейн можна подати так:

– однорангова система транзакцій цінностей без довірених третіх сторін між ними (немає потреби в довірених третіх сторонах, які виступають посередниками для перевірки, захисту та розрахунків за транзакціями);

– спільна, децентралізована та відкрита книга транзакцій. Ця база даних реєстру відтворюється на великій кількості вузлів;

– база даних реєстру є базою даних лише для додавання і не може бути змінена чи змінена. Це означає, що кожен запис є постійним. Будь-який новий запис у ньому відображається на всіх копіях баз даних, розміщених на різних вузлах.

Так само, як TCP/IP був розроблений для досягнення відкритої системи, технологія блокчейн була розроблена для справжньої децентралізації. Щоб це зробити, творці біткойна зробили його відкритим, щоб він міг надихнути багато децентралізованих програм.

Технологію блокчейн можна розглядати як шарову структуру.

Мета полягає не в тому, щоб стандартизувати технологію блокчейн, а в тому, щоб краще розуміти. Всі шари присутні на всіх вузлах (рис. 1.7).

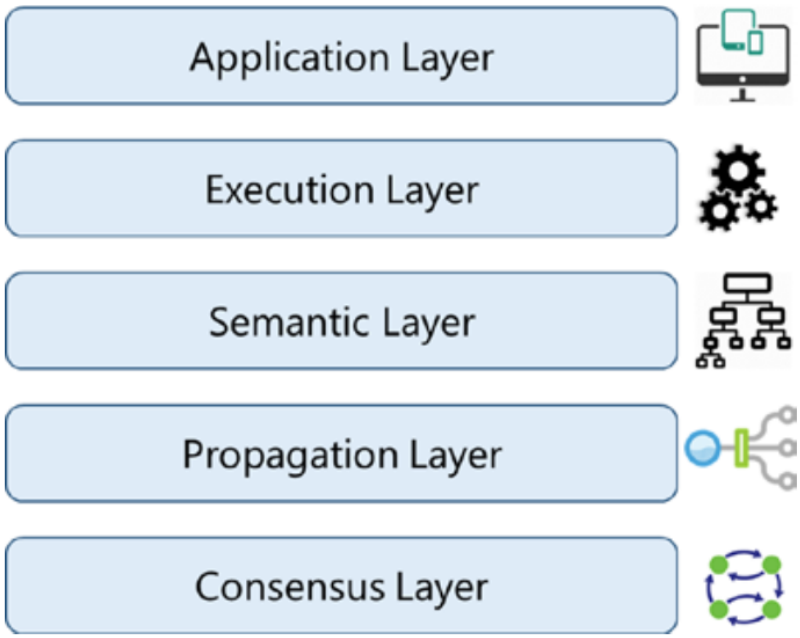


Рисунок 1.7 – Шари блокчейну

Application Layer. Це включає традиційний стек технологій для розробки програмного забезпечення, як-от конструкції програмування на стороні клієнта, сценарії, API, фреймворки розробки тощо. Для програм, які розглядають блокчейн як серверну частину, ці програми, можливо, потрібно буде розмістити на деяких вебсерверах, і для цього може знадобитися

веброзробка застосунків, серверне програмування та API тощо. В ідеалі хороші блокчейн-програми не мають моделі клієнт-сервер, і немає централізованих серверів, до яких клієнти мають доступ, саме так і працює біткойн.

Execution Layer. Рівень виконання – це місце, де виконання інструкцій, замовлених прикладним рівнем (Application Layer), відбувається на всіх вузлах у мережі блокчейн. Інструкції можуть бути простими інструкціями або набором кількох інструкцій у формі смарт-контракту. Усі вузли в мережі блокчейн повинні виконувати програми/скрипти незалежно. Детерміноване виконання програм/скриптів на одному наборі вхідних даних і умов завжди дає однаковий вихід на всіх вузлах, що допомагає уникнути невідповідностей.

Semantic Layer. Також називається логічним рівнем, займається перевіркою транзакцій, що виконуються, перевіркою блоків, що генеруються в мережі, відповідає за зв’язування блоків, створених у мережі.

Propagation Layer. На цьому рівні вузли синхронізуються один з одним відносно поточного стану мережі. Коли транзакція здійснюється, вона транслюється на всю мережу. Так саме, коли вузол хоче запропонувати блок, він поширюється на всю мережу, щоб інші вузли могли будувати його, вважаючи його останнім блоком.

Consensus Layer. На цьому базовому рівні забезпечується безпека та захист блокчейну. Мета цього рівня – узгодити стану реєстру між усіма вузлами. У Bitcoin або Ethereum консенсус досягається за допомогою методів стимулювання, які називаються «майнінг». Біткойн і Ethereum використовують механізм консенсусу Proof of Work (PoW). Після створення нового блоку блок поширюється на всі інші вузли, щоб перевірити, чи дійсний новий блок із транзакціями в ньому, і на основі консенсусу з усіх інших вузлів новий блок додається до блокчейну. Існує багато різних варіантів консенсусних алгоритмів, таких як Proof of Stake (PoS), delegated PoS (dPoS), Practical Byzantine Fault Tolerance (PBFT).

Основні властивості блокчейну, що викликали його широке застосування, полягають у наступному.

Незмінність. Запис транзакції неможливо змінити. Коли транзакція додається в мережу, то кожен вузол має її копію. Незмінність зростає із кількістю блоків записаних поверх блоку, і через певний час стає повністю незмінним., оскільки вони криптографічно захищені. Зареєстрована транзакція назавжди залишається в блокчейні.

Стійкість до підробок. У блокчейні транзакції є відкритими. Криптографічний хеш і цифрові підписи використовуються для забезпечення стійкості системи до підробок. Обчислювально неможливо підробити чийсь підпис. Якщо користувач робить транзакцію та підписуєте її хеш, ніхто не зможе пізніше змінити транзакцію та сказати, що була підписана інша транзакція. Крім того, користувач не можете пізніше стверджувати, що ніколи не здійснювали транзакцію, оскільки він її підписав.

Захист від подвійних витрат. Такі атаки можуть здійснюватися як у грошових, так і в негрошових транзакціях. Подвійні витрати – це спроба витратити ту саму суму кільком людям. Приклад: у користувача в обліковому записі є 100 доларів США, а він платить 90 доларів двом чи більше сторонам. Або якщо хтось володіє землею і продає ту саму ділянку землі двом людям.

У централізованій системі легко запобігти подвійним витратам, оскільки центральний орган знає про всі транзакції. Блокчейн-рішення також має бути захищеним від таких атак. Єдиний можливий спосіб запобігти подвійним витратам у блокчейні – це бути в курсі всіх транзакцій. Якщо знати про всі транзакції в минулому, то можно визначити, чи є транзакція спробою подвоїти витрати. Таким чином, вузли, які перевірятимуть транзакції, обов’язково повинні бути доступні для всіх даних блокчейну, починаючи з блоку генезису.

Стійкість. Мережа має бути достатньо стійкою, щоб витримувати тимчасові збої вузлів, недоступність деяких обчислювальних вузлів час від часу, затримку у мережі, падіння пакетів тощо.

Можливість проведення аудиту. Блокчейн – це ланцюжок блоків, які з’єднані між собою хешами. Можливість перевірки існує завдяки тому, що блоки пов’язані у зворотному напрямку до блоку генезису, і вона не повинна порушиться будь-якою ціною. Крім того потрібна можливість швидкої перевірки транзакція, яка мала місце у минулому.

1.3 Застосування хешування та криптографії у блокчейні

Криптографія гарантує, що ніхто не може здійснити транзакцію від іншого імені, підробивши підпис. Використовуючи криптографію, можна переконатися, що дійсний користувач ініціює транзакцію, і ніхто не зможе підробити шахрайську транзакцію [8, 9].

Використання криптографії обумовлено такими властивостями:

– конфіденційність (також приватність): лише авторизований одержувач може зрозуміти повідомлення;

– цілісність даних: криптографія не може запобігти зміні даних, вона може виявити те, що дані були змінені;

– автентифікація: гарантується автентичність відправника та може бути перевірена одержувачем;

– невідмова: відправник (користувач чи система) не може відмовитися від права власності на попереднє зобов’язання (транзакцію).

Криптографія із симетричним ключем (Symmetric Key Cryptography) зводиться до наступного [7].

Один і той же ключ використовується як для шифрування, так і для дешифрування, це називається криптографією з симетричним ключем. Користувачі повинні узгодити ключ «загальний секрет» (shared secret), перш ніж обмінюватися зашифрованим текстом. Для симетричної криптографії потрібен безпечний канал розподілу, щоб поділитися ключем (рис. 1.8).

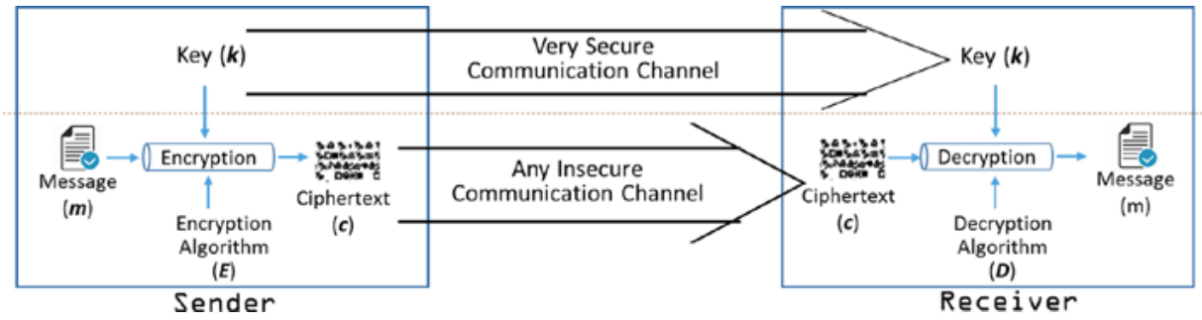


Рисунок 1.8 – Symmetric Key Cryptography

Криптографічні хеш-функції (Cryptographic Hash Functions) використовуються в цифрових підписах (Digital Signatures), автентифікації повідомлень (message authentication codes) та в асиметричній криптографії. Криптографічна хеш-функція – це одностороння функція, яка перетворює вхідні дані довільної довжини та створює вихідні дані фіксованої довжини. Вихідні дані зазвичай називають «хеш-значенням» (hash value) або «дайджест» (message digest) (рис. 1.9).

Основні властивості хеш-функції:

– вхідні дані будь-якого розміру, але вихідні мають фіксовану довжину;

– хеш-значення має бути обчислюваним для будь-якого повідомлення;

– детермінована – той самий вхідний сигнал, переданий тій самій хеш-функції, щоразу повертає однакове хеш-значення;

– нездійсненно (хоча й не неможливо!) інвертувати повідомлення з його

– хеш-значення, за винятком спроби для всіх можливих повідомлень;

– невелика зміна в повідомленні має сильно впливати на вихідний хеш, щоб неможливо було співвіднести нове хеш-значення зі старим після невеликої зміни.

Криптографія з асиметричним ключем (Asymmetric Key Cryptography) також відома як «криптографія з відкритим ключем» (public key cryptography), вирішила проблему розподілу ключів у симетричній системі шляхом впровадження цифрових підписів.

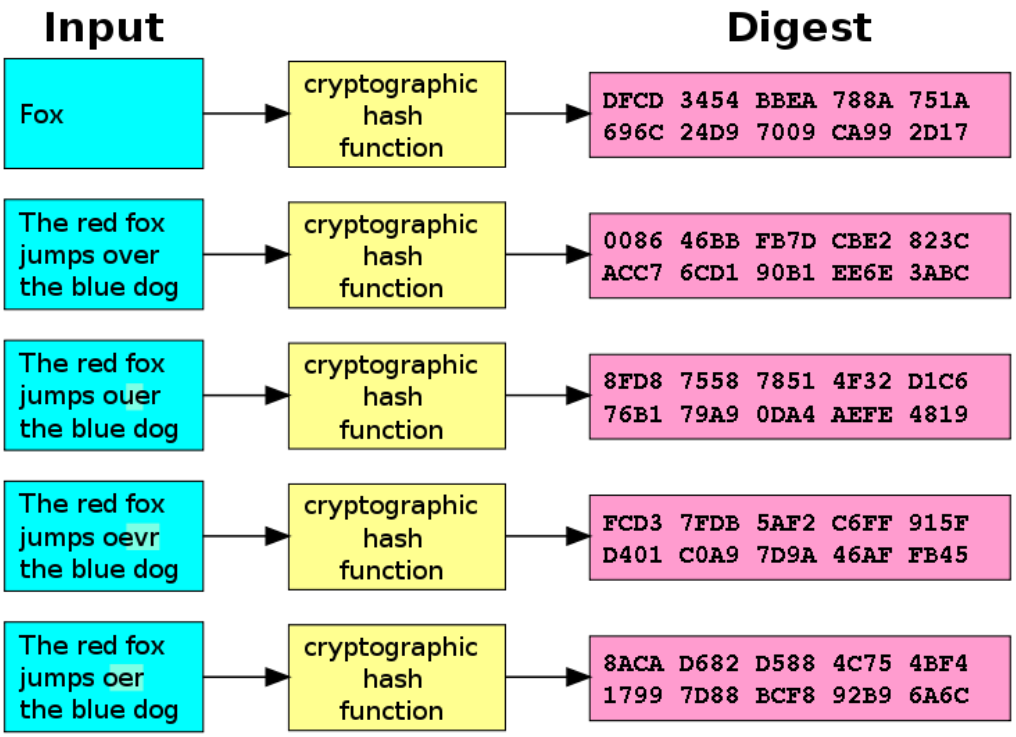


Рисунок 1.9 – Приклад роботи SHA-1. Невелика зміна на вході (у слові «over») різко змінює вихід (дайджест). Це лавинний ефект (avalanche effect)

Відправник шифрує повідомлення за допомогою алгоритму шифрування та відкритого ключа отримувача та надсилає зашифрований текст. Приймач розшифровує текст за допомогою алгоритму дешифрування і свого закритого ключа, щоб отримати текст (рис. 1.10).

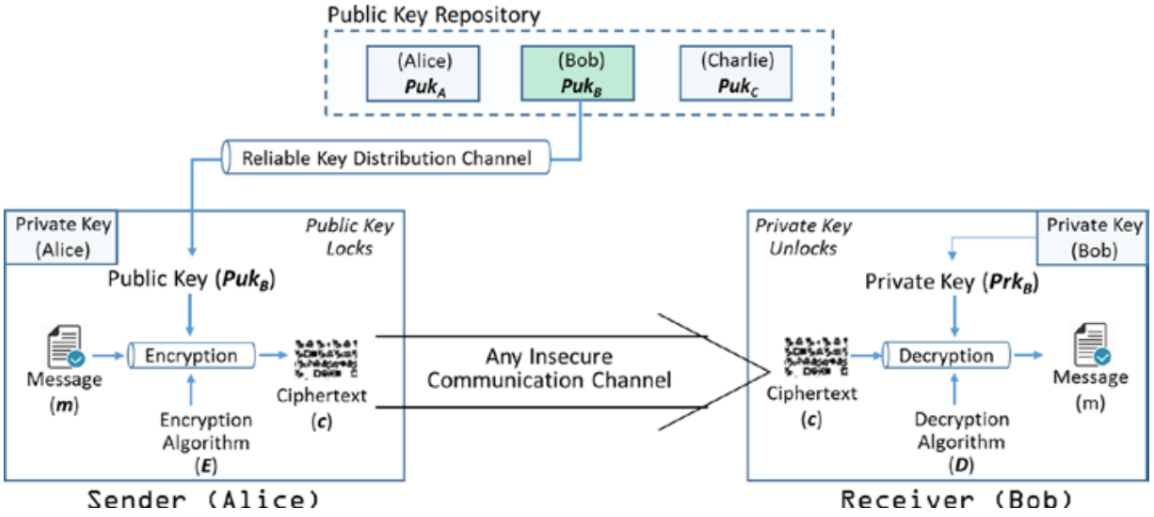


Рисунок 1.10 – Криптографія з відкритим ключем

Відкриті ключі відомі та доступні кожному, використовуються для шифрування повідомлення або для перевірки підписів. Особисті ключі володар повинен зберігати у таємниці. Вони використовуються для розшифровки повідомлення або для створення підписів.

Блокчейн – це ланцюжок блоків, пов’язаних разом. Блок – це лише одна транзакція або декілька транзакцій, об’єднаних разом. Хеш (transaction hash, transaction id) є основним будівельним блоком структури даних блокчейна.

Хеш вказує на попередній блок даних і надає спосіб перевірити, чи не були дані підроблені. Блокчейн повинен бути стійкий до втручання, щоб вважатися єдиним джерелом істини. Для досягнення цієї мети хеш попереднього блоку зберігається в заголовку поточного блоку, а хеш поточного блоку з його заголовком блоку буде збережено в заголовку наступного блоку. Кожен блок вказує на свій попередній блок, відомий як «батьківський блок». Кожен новий блок, який додається до ланцюжка, стає батьківським блоком для наступного блоку, який буде додано. Він йде аж до першого блоку, який створюється в блокчейні, який називається «блок генезису». У такій конструкції, де блоки пов’язані назад за допомогою хешів, практично неможливо змінити дані в блоці бо хеші не зберігатимуться, якщо дані буде змінено (завдяки властивостям хеш-функцій).

Вузли в блокчейн-мережі анонімні і працюють в умовах відсутності довіри. Ускорити і упростити процес перевірки даних допомагають дерева Меркла [6].

Хеш-значення блоку збирається за допомогою дерева Меркла.

Дерево Меркла, або хеш-дерево, – це двійкове дерево, заключні вузли якого – це хеши транзакций, а внутрішні вершини – результати складання значень пов’язаних вершин (рис. 1.11).

Блок будується з назви і списку транзакцій. Назва складається з свого хеш-значення, хеша попереднього блоку, хешів транзакцій і додаткової інформації. Перша транзакція – отримання комісії, що потім стане винагородою для юзера, який створив блок.

Дерево Merkle захищене від втручання. Втручання на будь-якому рівні в дереві не збігається з хешем, що зберігається на одному рівні вище в ієрархії, а також до кореневого вузла. Це забезпечує цілісність порядку транзакцій. Якщо змінити порядок транзакцій, то також зміняться хеші в дереві до кореня Merkle.

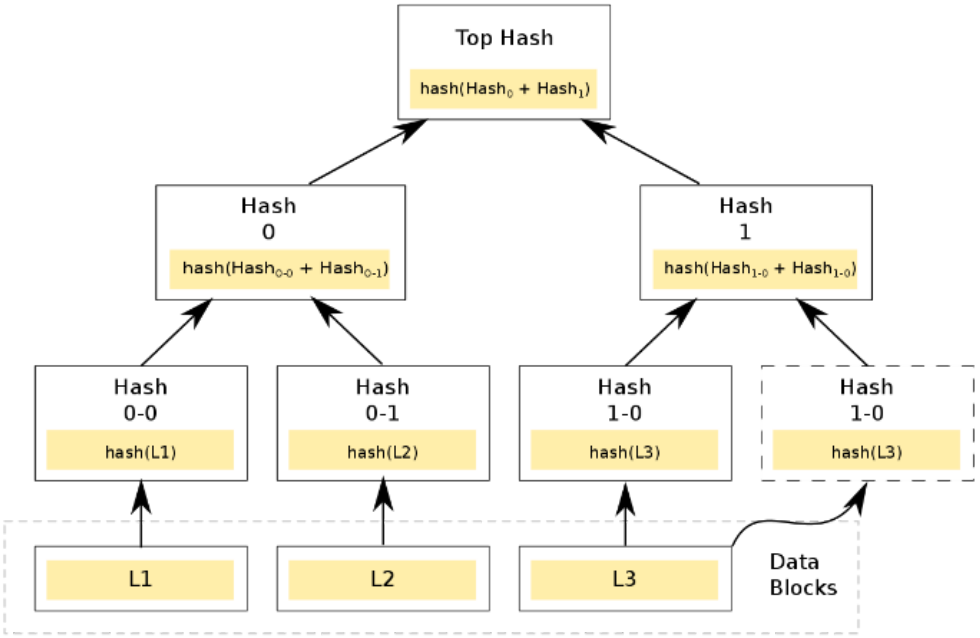


Рисунок 1.11 – Дерево Меркла

1.4 Аналіз зображень і блокчейн

Методи аналізу та розпізнавання зображень із забезпеченням необхідних прикладних характеристик результативності знаходять все більше впровадження у сучасних інтелектуальних системах комп'ютерного зору [10-34]. Блокчейн-системи теж вносять свій вагомий внесок у розвиток комп'ютерного зору шляхом використання нових ефективних принципів збереження, захисту та перевірки даних [29-35].

Набуває популярності незамінний токен (NFT – Non-Fungible Token) – це невзаємозамінна одиниця даних, що зберігається у блокчейні, яку можна продавати у інтернет-мережі [35]. Типи блоків даних NFT можуть бути пов’язані з цифровими файлами, такими як фотографії, відео та аудіо. Оскільки кожен токен можна унікально ідентифікувати, NFT відрізняються від блокчейн-криптовалют, таких як біткойн.

«Незамінний» означає, що він унікальний і не може бути замінений чимось іншим. Наприклад, біткойн є взаємозамінним– можно обміняти один біткойн на інший, і отримати те саме. Але єдину в своєму роді рідкісну колекційну картку змінити не можна. При обміні її на іншу картку отримується щось зовсім інше.

Цей цифровий актив може також представляти об’єкти реального світу, такі як мистецтво, музика, ігрові елементи та відео. Їх купують і продають онлайн, часто за криптовалюту, і вони кодуються за допомогою того самого програмного забезпечення, що лежить в основі багатьох криптовалют [22].

NFT дозволяє володіти оригінальним предметом і містить вбудовану автентифікацію, яка служить доказом власності.

Цифрові представлення NFT базуються на цифрових форматах, таких як зображення, відео, текст та аудіо. Вони працюють із різними підрозділами аналізу даних. Наприклад із комп’ютерним зором: наразі NFT насамперед стосуються зображень і відео та можуть використовувати засоби комп’ютерного зору.

Останніми роками такі технології, як згорточні нейронні мережі (CNN), генеративні ворожі мережі (GAN) розширили горизонти комп’ютерного зору. Створення зображень, розпізнавання об’єктів і розуміння сцени – це сучасні технології комп’ютерного зору, які можна застосувати до наступної хвилі розвитку NFT [18]. Генеративне мистецтво здається чіткою сферою для поєднання комп’ютерного зору та NFT.

Все більше переваг отримують NFT, створені системами штучного інтелекту. Досвід впровадження методів глибокого навчання в таких сферах, як комп’ютерний зір, створюють новий досвід у створенні NFT.

Приклад можна побачити у цифрового художника Рефіка Анадола, який вже використовував методи AI для створення NFT. Деякі з найвідоміших проектів PFP NFT (profile pics – фото профілю) включають Bored Ape Yacht Club (рис. 1.12) і Pudgy Penguins Series (рис. 1.13). Схоже, вони використовують успіх Crypto Punks.

Інноваційні технології призвели до розвитку багатьох незвичайних видів мистецтва. Художники тепер можуть використовувати штучний інтелект і комп’ютерні алгоритми в цьому процесі.

Світ генеративного мистецтва змінюється завдяки підтримці і підвищенню популярності алгоритмів у порівнянні з традиційними методами. NFT, створені AI, унікальні та цікаві. Крім того, алгоритм створює зображення із заздалегідь визначеним стилем.

Рисунок 1.12 – Колекція NFT – Bored Ape Yacht Club, згенерована за допомогою методів AI

Рисунок 1.13 – Колекція NFT – Pudgy Penguins Series, згенерована за допомогою методів AI

1.5 Постановка задачі дослідження

Здійснення технологічного рішення для задачі виявлення факту фальсифікації документів та полегшення процесу перевірки їх оригінальності є актуальною задачею на сьогодні. Нові технологічні досягнення з розвитком блокчейну та смарт-контрактів з їх характеристиками незмінності, децентралізації, безпеки, відстежуваності та компонентного консенсусу можна вважати результативним підходом для впровадження надійного рішення щодо боротьби із шахрайством при видачі цифрових документів. Інновації, такі як зв’язування блокчейну та бази сертифікатів чи дипломів, передбачають значні ефективні зміни та технологічну новизну. Поєднання блокчейну та бази наявних документів потенційно може позитивно вплинути та принести значну прикладну користь мільйонам роботодавців та працівників.

Об’єктом дослідження є прикладний смарт-контракт для зберігання та перевірки істинності даних про документи.

Метою дослідження є розробка та дослідження вебзастосунка для підтвердження оригінальності документів із застосуванням блокчейн-технології Ethereum.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити особливості технології блокчейн та платформи Ethereum;

– розробити смарт-контракт для процесів додавання документів до блокчейну та їх перевірки на оригінальність;

– розроблення вебінтерфейсу для роботи зі спеціалізованим смарт-контрактом;

– розрахунок обчислювальних затрат у вигляді вартості газу для деплою смарт-контракту та проведення транзакцій для збереження та перевірки документів.

2 ТЕХНОЛОГІЯ ETHEREUM ТА СМАРТ-КОНТРАКТИ

2.1 Віртуальна машина Ethereum та смарт-контракти

Технологія блокчейн з’явилася разом із біткойнами в 2009 році [6]. Варіанти використання блокчейну вийшли за межі банківського та фінансового секторів і охопили такі галузі, як логістика, ритейл, електронна комерція, охорона здоров’я, енергетика та державний сектор. Блокчейн може використовуватися для транзакцій і відстеження будь-чого цінного, а не лише криптовалюти [7-9]. Наприклад «доказ існування» – це один із таких випадків використання, коли хеш документа було введено в мережу блокчейну біткойн, щоб будь-хто згодом міг перевірити, що такий документ існував у такий-то момент часу. В. Бутерін представив блокчейн-платформу Ethereum, яка може сприяти транзакціям не лише грошей, але й акцій, землі, цифрового контенту, транспортних засобів та багатьох інших, які мають певну внутрішню цінність.

На відміну від Біткойна, технологія Ethereum вже підтримує повний Тюрінг, що дозволяє писати смарт-контракти, які можуть робити що завгодно з точки зору програмування. Крім того, Ethereum відстежує стан облікового запису, що дуже відрізняється від біткойна, де все залишається як транзакція та немає внутрішньої постійної пам’яті для сценаріїв. Завдяки цим перевагам на базі Ethereum швидко розробляються децентралізовані програми, які взаємодіють між собою, та забезпечують належну безпеку. Ethereum робить це завдяки абстрактному базового шару. Основні складності приховані від розробників; та дозволяють їм отримувати гнучкість проектування власних функцій для прямої передачі цінності та інформації, а також форматів транзакцій.

Основною інновацією Ethereum стала віртуальна машина Ethereum (EVM). Підтримка повного Тюрингу через EVM полегшує створення

блокчейн-застосунків. Подібно до того, як віртуальна машина Java (JVM) потрібна для запуску коду Java, EVM потрібна для запуску смарт-контрактів. Смарт-контракти (smart contract – «розумний контракт») – це сценарії Ethereum, написані мовою Solidity, які автоматично виконуються у випадку, якщо відбудеться заздалегідь визначена подія.

На відміну від традиційної розробки програмного забезпечення, DApps (Децентралізовані додатки) не потрібно розміщувати на сервері. Код вбудовано в транзакції, так би мовити, які потім надсилаються до вузлів в мережі Ethereum.

Облікові записи Ethereum. В біткойні немає поняття балансу рахунку як такого. На відміну від нього, Ethereum має статус, і його основною одиницею є обліковий запис. Кожен обліковий запис має пов’язаний із ним стан, а також має 20-байтову (160 біт) адресу, через яку він ідентифікується та посилається на нього. Метою блокчейну в Ethereum є відстеження змін стану. Існують такі типи облікових записів Ethereum [6]:

– зовнішні облікові записи (EOA, Externally Owned Accounts): це облікові записи, які зазвичай належать користувачам або пристроям, які керують цими обліковими записами за допомогою закритих ключів. EOA можуть надсилати транзакції до інших EOA або смарт-контрактів, підписавши закритим ключем. Операція між двома EOA зазвичай полягає в передачі будь-якої форми вартості;

– контрактні акаунти: це і є смарт-контракти, вони контролюються лише кодом, який міститься в них. Контрактні облікові записи здатні виконувати бізнес-логіку за допомогою коду, який вони містять, та вони не можуть ініціювати нові транзакції самостійно і завжди залежать від EOA. На відміну від EOA в контрактах немає приватного ключа, яким можно підписати транзакцію.

Кожна транзакція в Ethereum (запис у блокчейн) потребує обчислювальних ресурсів від майнерів. Транзакції в Ethereum виконуються на «газі», фундаментальній одиниці обчислень в Ethereum. Кожна транзакція,

будь то EOA чи контракт, повинна мати gasLimit і gasPrice для обчислення комісії. Ця комісія виплачується майнерам, щоб компенсувати їм внесок у ресурси та роботу, яку вони виконують. Очевидно, у майнерів є вибір: включити транзакцію та стягнути комісію.

Плата за газ сплачується в рідній валюті Ethereum, ефірі (ETH). Ціни на газ позначаються в gwei, який сам по собі є номіналом ETH – кожен gwei дорівнює 0,000000001 ETH (10-9 ETH). Наприклад, замість того, щоб сказати, що ваш газ коштує 0,000000001 ефіру, можно сказати, що газ коштує 1 gwei. 1 gwei дорівнює 1 000 000 000 wei.

Сам wei є найменшою одиницею ETH (рис. 2.1).

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

Рисунок 2.1 – Система деномінацій

Зазвичай обчислювальний крок коштує лише одну одиницю газу, але деякі з обчислювальних операцій коштують дорожче. Для кожного байта даних транзакції потрібно близько п’яти газів. Наприклад: додавання двох чисел вимагає приблизно трьох газів; множення двох чисел вимагає приблизно п’яти газів; обчислення хешу (SHA3) вимагає близько 30 газів (інтенсивне обчислення). Вартість зберігання також обчислюється подібним чином, але досить дорого.

Ціна транзакції залежить від кількості газу, спожитого транзакцією, помноженого на ціну однієї одиниці газу, вказану в транзакції ініціатором транзакції. З іншого боку, майнери мають стратегію розрахунку ціни на газ, яка має бути найменшою, яку має вказати відправник транзакції, щоб майнер не відхилив транзакцію. Загальна вартість «ефіру» транзакції базується на двох факторах: gasUsed і gasPrice. Загальна вартість = використаний газ * ціна газу. Компонент gasUsed – це загальний газ, спожитий під час виконання кодів операцій EVM для інструкцій, а gasPrice – це той, який вказує користувач.

В Ethereum розмір блоку контролюється лімітом газу для блоку. Різні транзакції мають різні ліміти газу; тому, залежно від ліміту блокового газу, певна кількість транзакцій об’єднується разом, щоб загальний ліміт газу для транзакцій був меншим за ліміт газу блоку. Межа блокового газу розраховується динамічно. Протокол Ethereum дозволяє майнеру блоку регулювати ліміт газу блоку з коефіцієнтом 1/1024 (0,0976%) у будь-якому напрямку.

Лістинг 2.1 Код для оцінки вартості транзакції:

```
var MyContract = artifacts.require("./MyTest.sol");
MyContract.web3.eth.getGasPrice(function(error, result){
  var gasPrice = Number(result);
   console.log("Current gasPrice is " + gasPrice + " wei");
   MyContract.deployed().then(function(instance) {
      giveAwayDividend()
      return instance.giveAwayDividend.estimateGas(1);
  }).then(function(result) {
      var gas = Number(result);
console.log("Total gas estimation = " + gas + " units"); console.log("Total
Transaction Cost estimation in Wei = " + (gas * gasPrice) + " wei");
```

```
console.log("Total Transaction Cost estimation in Ether = " +
MyContract.web3.fromWei((gas * gasPrice), 'ether') + " Ether");
  });
});
```

2.2 Транзакція Ethereum і структура повідомлень

Щоб транзакція була кваліфікована майнерами або вузлами Ethereum, вона повинна мати стандартизовану структуру. Типова транзакція Ethereum складається з таких полів:

– nonce: це ціле число, просто лічильник, що дорівнює кількості транзакцій, надісланих обліковим записом відправника, тобто порядковий номер транзакції;

– gasPrice: ціна, яку ви готові платити з точки зору кількості Wei, яку потрібно заплатити за одиницю газу;

– gasLimit: максимальна кількість газу, який має бути використаний для виконання цієї транзакції, що також обмежує максимальну кількість обчислювальних кроків, які дозволяється виконувати транзакцію;

– to: 160-бітна адреса одержувача або адреса контракту. Для транзакції, яка використовується для створення контракту;

– значення: загальний ефір (кількість Wei), який буде передано одержувачу відправником транзакції;

– *v*, *r*, *s*: значення, що відповідають підпису транзакції ECDSA; також представляють відправника цієї транзакції;

– init: це не є необов’язковим полем, воно використовується лише з транзакціями, що використовуються для створення контрактів. Це поле може містити байтовий масив необмеженого розміру, що вказує EVM-код для процедури ініціалізації облікового запису;

– код операції «init» використовується лише один раз для ініціалізації нового контрактного облікового запису та після цього скидається. Він повертає тіло коду облікового запису після пов’язування його з обліковим записом контракту. Цей зв’язок є постійним явищем і ніколи не змінюється;

– data: не обов’язкове поле, яке може містити повідомлення, яке буде надіслано на договірний або простий обліковий запис. Він не має спеціальної функції як такої за замовчуванням, але EVM має код операції, за допомогою якого контракт може отримати доступ до цього поля даних, виконати необхідні обчислення та помістити їх у сховище.

Зазначені поля надаються у такому порядку та всі закодовані RLP, за винятком назв полів. Отже, транзакція Ethereum насправді означає підписаний пакет даних із цими полями. Поля gasPrice і gasLimit важливі для запобігання відмові службової атаки.

Транзакції Ethereum насправді є «функціями переходу стану», оскільки успішна транзакція змінює стан. Крім того, результат цих транзакцій можна зберігати.

Повідомлення Ethereum схожі на транзакції, але ініціюються лише контрактними обліковими записами, а не EOA. Крім того, повідомлення призначені для передачі лише між контрактними рахунками, через що вони також називаються «внутрішніми транзакціями». Отже, контракти мають можливість надсилати повідомлення іншим контрактам. Повідомлення завжди є необробленими і ніколи не серіалізуються чи десеріалізуються. Повідомлення містить такі поля:

– sender: відправник повідомлення як неявний параметр;

– recipient: адреса контракту одержувача, на яку потрібно надіслати;

– value: сума Wei, яку потрібно переказати на адресу контракту разом із повідомленням;

– data: не обов’язкове поле, але може містити вхідні дані для договору одержувача, надані відправником;

– gasLimit: значення, яке обмежує максимальну кількість газу, яку може споживати виконання коду, коли ініціюється повідомленням. Його також називають «startGas».

Транзакція Ethereum може здійснюватися від EOA до EOA або від EOA до контрактного рахунку. Існує ще одна ситуація, коли транзакція з EOA ініціюється для створення контрактного облікового запису (поле «init»).

Транзакція – це міст між зовнішнім світом і блокчейном Ethereum, а також це інструкція, ініційована EOA шляхом її підписання, яка серіалізується та надсилається в блокчейн.

EOA або контрактний рахунок складається з таких чотирьох компонентів:

– баланс рахунку: Загальний баланс «ефіру» на рахунку. Точніше, кількість Wei, якими володіє адреса (1ETH = 1018 Wei);

– codeHash: це хеш «коду». У кожному контрактному обліковому записі є «код», який виконується на EVM. Хеш цього коду зберігається в цьому полі CodeHash. Однак для облікових записів EOA не існує «коду», тому поле CodeHash містить хеш порожнього рядка;

– storageRoot: це 256-бітний кореневий хеш, який кодує вміст сховища облікового запису. Зберігання кореневого хешу цього дерева в полі storageRoot допомагає відстежувати вміст облікового запису, а також забезпечує його цілісність;

Після успішного виконання транзакцій, відбувається зміна стану блокчейну. Побачити живі транзакції можна на https://etherscan.io (рис. 2.2).

Ethereum на відміну від Bitcoin пропонує набагато більше можливостей саме завдяки смарт-контрактам. В смарт-контракті кодується алгоритм дій, тобто це хоч і називається контракт – це все ж таки програмний код. Смарт-контракти програмуються на мові Solidity, яка була розроблена безпосередньо для виконання такої задачі. Ця мова зараз використовується на платформі Ethereum та інших блокчейн-застосуваннях, де використовуються EVM.

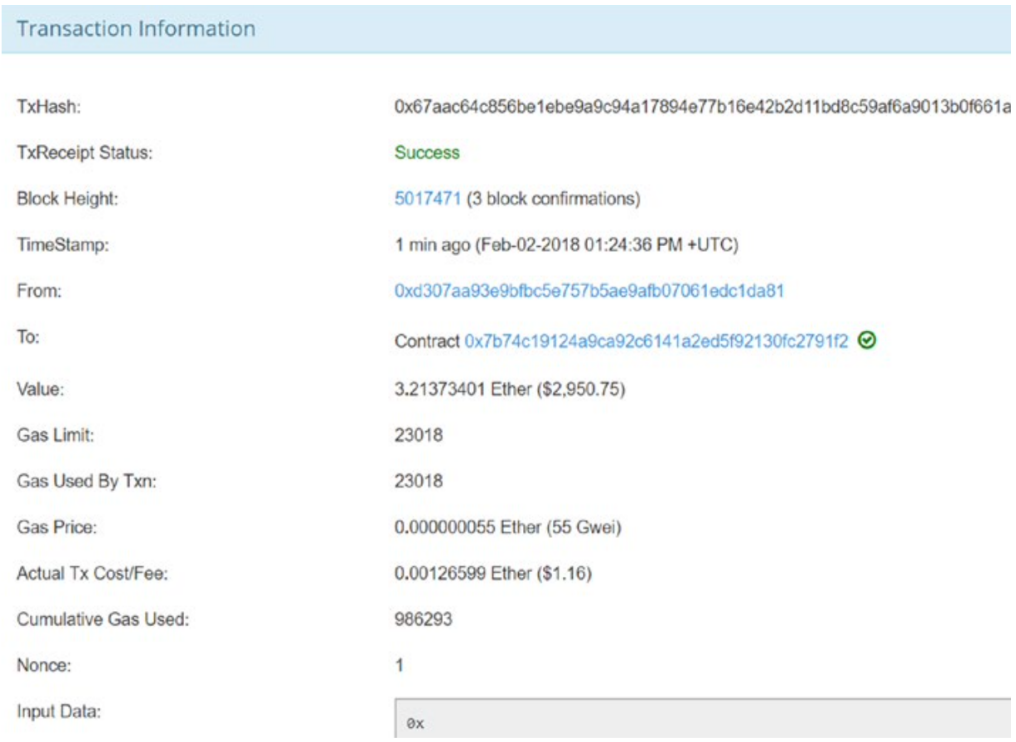


Рисунок 2.2 – Транзакції на etherscan

Смарт-контракти мають власну адресу та баланс. Вони можуть надсилати повідомлення та отримувати транзакції. Смарт-контракт виконує свій код, коли отримує транзакцію. Як і для інших транзакцій, для них також застосовуються плата за виконання та плата за зберігання (газ сплачує акаунт, який відправляє транзакцію і тим самим ініціює якусь функцію у коді смарт-контракту).

Весь код в Ethereum, включно зі смарт-контрактами, скомпільовано в мову байт-кодів низького рівня на основі стека, що називається кодом EVM, який працює на EVM. Популярними мовами високого рівня, які використовуються для написання смарт-контрактів, є Solidity, Serpent і LLL, де відповідні компілятори перетворюють код мови високого рівня в байт-код EVM. EOA може додавати контракти в блокчейн. Оскільки обчислення та зберігання в Ethereum дуже дорогі, доцільно, щоб логіка була написана якомога простіше та була оптимізована. Коли смарт-контракт розгорнуто в мережі блокчейну Ethereum, будь-який EOA може викликати функції смарт-контракту. Але функції зазвичай мають закодовані перевірки, які

запобігають несанкціонованому доступу; тим не менш, можна робити спроби, хоч і неуспішні.

2.3 Криптографія в Ethereum. Алгоритм Keccak

За останні два десятиліття було запропоновано велику кількість хеш-функцій. На практиці, безумовно, найпопулярнішими є хеш-алгоритми так званого сімейства MD4. MD5, сімейство SHA та RIPEMD базуються на принципах MD4.

У 2004 році Xiaoyun Wang оголосив про атаки з пошуком колізій проти MD5 і SHA-0. Роком пізніше було заявлено, що атаку можна поширити на SHA-1. Проте перша успішна атака зіткнення в реальному світі була здійснена в 2017 році. Науковці з Google і CWI створили два файли з однаковим хешем SHA-1 під час першої в історії атаки зіткнення SHA-1.

Згодом NIST вирішив розробити додаткову хеш-функцію під назвою SHA-3 через публічний конкурс. Цей підхід дуже схожий на процес відбору AES наприкінці 1990-х років. Однак, на відміну від AES, який явно мав на увазі заміну DES, планувалося, що SHA-2 і SHA-3 повинні існувати разом за умови відсутності нових атак на SHA-2. Фактично, на момент написання статті, тобто. , початок 2013 року, SHA-2 все ще вважається дуже безпечним. З цієї причини як SHA-2, так і SHA-3, коли його буде завершено, будуть федеральними стандартами США [36].

9 грудня 2010 року NIST оголосили п’ятьох кандидатів на 3 раунд. Це хеш функції:

– BLAKE;

– Grøstl;

– JH;

– Keccak;

– Skein.

NIST вибирає Keccak 2 жовтня 2012 року як основу для хеш-функції SHA-3.

Слід підкреслити, що Keccak має зовсім іншу внутрішню структуру, ніж хеш-функції, які належать до сімейства MD4, включаючи SHA-1 і SHA-2.

Коли Ethereum було запущено в липні 2015 року, Sha-3 використовувався в Ethash і в функціях віртуальної машини Ethereum (EVM) для створення хешів для облікових записів, блоків, транзакцій тощо. Однак NIST внесло незначні зміни в Sha Алгоритм -3 у травні 2014 року, а офіційний стандарт Sha-3 було оновлено в серпні 2015 року. Цей новий стандарт Sha-3 створював різні хеші, ніж раніше визнана версія, яку використовував Ethereum. Версія Sha-3 в Ethereum більше не була стандартом Sha-3, а оновлений стандарт Sha-3 створював інші хеші, ніж хеші «Sha-3» в Ethereum.

Щоб зменшити плутанину між версією Sha-3 Ethereum та офіційним стандартом Sha-3, спільнота Ethereum почала називати його версію Sha-3 «Keccak-256». Отже, коли згадується «Sha-3» в Ethereum, насправді мається на увазі «Keccak-256».

Головною вимогою NIST для хеш-функції SHA-3 була підтримка таких вихідних довжин:

– 224 bits;

– 256 bits;

– 384 bits;

– 512 bits.

Якщо до хеш-функції застосовується пошук колізій – атака, яка через парадокс дня народження завжди можлива, SHA-3 із 256, 384 і 512 бітами результат показує складність атаки приблизно 2^{128}, 2^{192} і 2^{256} відповідно. Це точно відповідає криптографічний міцності, яку три довжини ключа AES забезпечують проти атак грубою силою. Подібним чином, 3DES має криптографічну міцність 2^{112}, а SHA-3 з 224-бітним виводом демонструє таку саму стійкість проти атак зіткнень [36].

Keccak також дозволяє генерувати довільну кількість вихідних бітів. Це повністю відрізняється від хеш-функцій SHA-1 і SHA-2, які виводять блок фіксованої довжини.

На відміну від SHA-1 і SHA-2, Keccak не покладається на конструкцію Merkle-Damgard. Швидше, хеш-функція базується на тому, що називається губчастою конструкцією (sponge construction) (рис. 2.3). Після попередньої обробки (яка ділить повідомлення на блоки та забезпечує доповнення), конструкція губки складається з двох фаз:

– Absorbing – фаза поглинання (або введення) Блоки повідомлень xi передаються в алгоритм і обробляються;

– Squeezing – фаза стиснення (або виведення) Обчислюється вихід конфігурованої довжини.

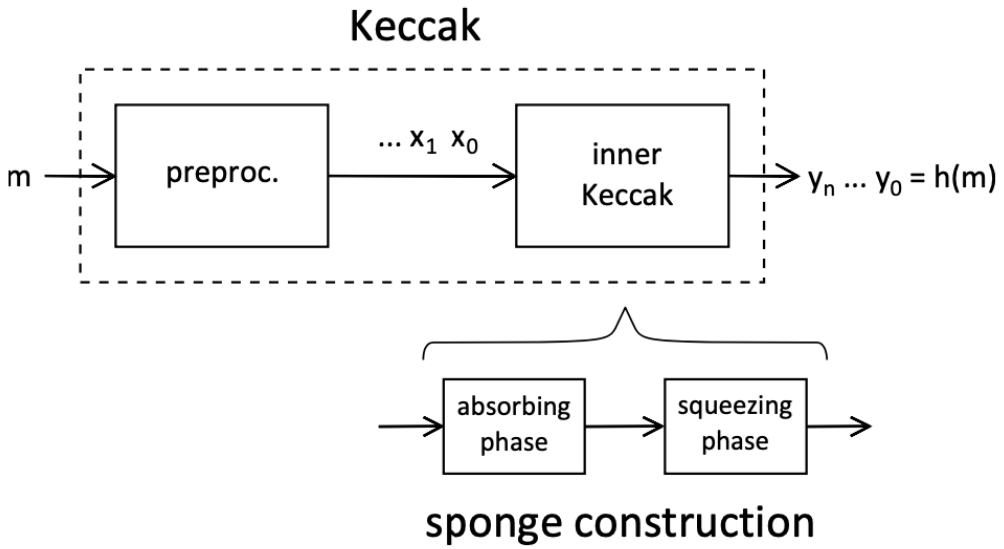


Рисунок 2.3 – Фази алгоритму Keccak

Для обох фаз використовується та сама функція. Ця функція називається Keccak-*f* (або inner-keccak). Конструкція губки читає вхідні блоки x_i і генеруються вихідні блоки y_j. Конструкція губки допускає вихідні дані довільної довжини $y_0 \ldots y_n$. Коли SHA-3 використовується як заміна SHA-2, потрібні лише перші біти першого вихідного блоку y_0 (рис. 2.4).

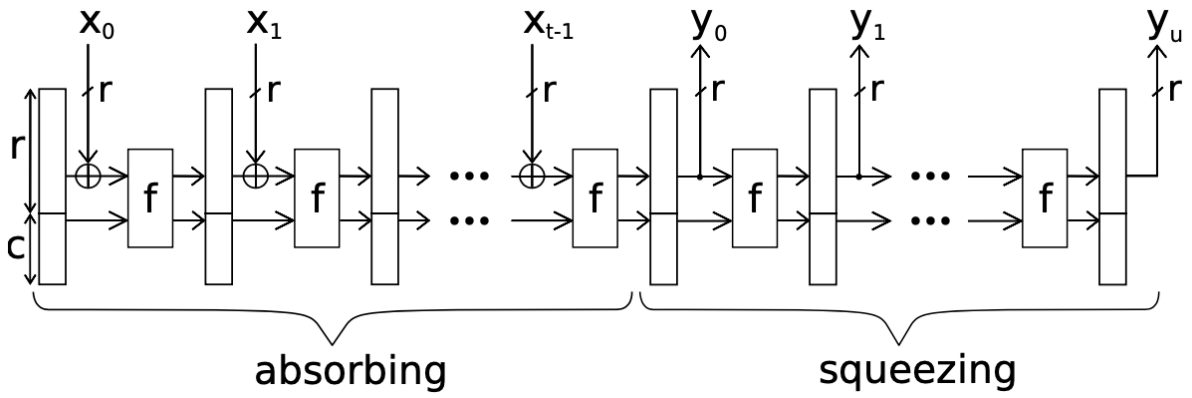


Рисунок 2.4 – Фази алгоритму Keccak

Є кілька параметрів, за допомогою яких можна налаштувати вхідні та вихідні розміри, а також рівень безпеки Keccak.

Відповідні параметри: b – ширина стану, тобто $b = r + c$ (рис. 2.4). b в свою чергу залежить від показника l і може приймати такі значення:

$$b = 25 \cdot 2^l, \tag{2.1}$$

де $l = 0,1,...,6$,

$b \in \{25,50,100,200,400,800,1600\}$.

Параметри $b = 25$ і $b = 50$ є лише значеннями для аналізу алгоритму і не повинні використовуватися на практиці.

r називається бітовою швидкістю (bit rate). r дорівнює довжині одного блоку повідомлення x_i.

c називається ємністю (capacity). Має дотримуватися, що $r + c$ є дійсною шириною стану, тобто $r + c = b \in \{25,50,100,200,400,800,1600\}$.

Для SHA-3 використовується стан $b = 1600$ біт.

Перед фактичною обробкою повідомлення m хеш-функцією, вхідні дані повинні бути доповнені. Однією з причин цього є те, що доповнений вхід має довжину, кратну r біт. (блоки з r бітів подаються в SHA-3.) Існують також міркування безпеки, які вимагають спеціального доповнення, що

використовується в SHA-3. Правило заповнення для вхідного повідомлення m таке:

$$pad(m) = m||P10\star1 = \ldots, x1, x0. \quad (2.2)$$

Схема додає заздалегідь визначений бітовий рядок P, за яким слідує 1, потім найменше число 0 і кінцева 1, щоб загальна довжина нового рядка була кратною r. Рядок $0\star = 0 \ldots 0$ може бути порожнім рядком, тобто не містити нулів. Значення P залежить від режиму, в якому використовується SHA-3, і наведено в таблиці 2.1. Якщо використовувати хеш-функцію як заміну SHA-2, мінімальна кількість бітів, доданих за правилом заповнення, становить чотири (тобто біти 0111), а максимальна кількість бітів дорівнює $r + 3$. Останній випадок має місце, якщо останній блок повідомлення складається з $r - 3$ бітів. В іншому режимі, тобто за допомогою SHA-3 зі змінною довжиною виводу, додається принаймні 6 біт і не більше $r + 5$ біт. Наприкінці процесу заповнення ми отримуємо серію блоків x_i, де кожен блок x_i має довжину r біт.

Вихідні дані. При використанні режиму заміни SHA-2 останній виклик функції Keccak-*f*, тобто останній раунд фази поглинання, створить хеш-вихід, який є частиною y_0. Навпаки, коли використовується режим виведення змінної довжини, фаза стиснення конструкції губки дозволяє обчислити стільки блоків хеш-виводу, скільки забажає користувач. З рисунка 2.5 видно, Keccak обчислює порції r вихідних бітів. У випадку SHA-3 $r = 1152$, $r = 1088$, $r = 832$ або $r = 576$, тобто y_0 має принаймні 576 біт. Якщо SHA-3 використовується як заміна SHA-2, потрібні лише 224, 256, 384 або 512 бітів. Щоб отримати потрібну довжину вихідних даних, молодші біти y_0 використовуються як хеш-виведення, а решта бітів y_0 відкидаються. При використанні Keccak у режимі виведення змінної довжини можна використовувати всі r бітів y_0, а також, звичайно, усі наступні блоки виводу $y_1, y_2, \ldots$

Функція Keccak-*f* лежить в основі хеш-алгоритму і використовується на обох фазах побудови губки (sponge construction). Keccak-*f* також називають перестановкою Keccak-*f*. Остання назва походить від того факту, що функція переставляє 2*b* вхідних значень, тобто кожне *b*-бітне ціле відображається точно на одне *b*-бітне вихідне ціле.

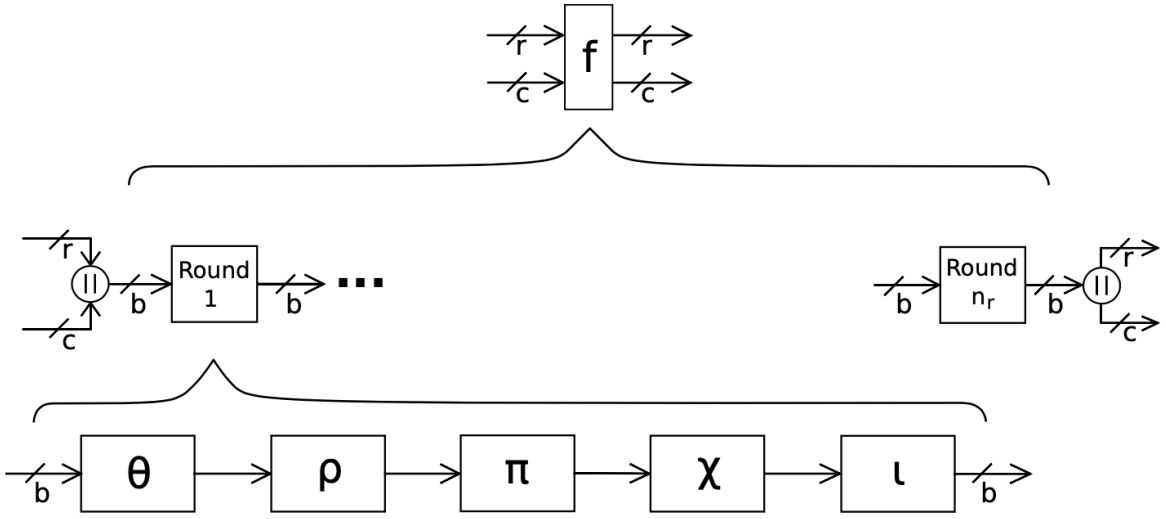


Рисунок 2.5 – Структура Keccak-*f*

Функція складається з n_r раундів. Кожен раунд має вхід, який складається з $b = r + c$ бітів. Кількість раундів залежить від параметра l:

$$n_r = 12 + 2^l. \tag{2.3}$$

l також визначає ширину стану $b = 25 \cdot 2^l$ (табл. 2.1). Зазначимо, що для SHA-3 є $n_r = 24$ раунди, оскільки $l = 6$. Раунди ідентичні, за винятком константи раунду RC[i], яка приймає різне значення в кожному раунді i. Константи округлення використовуються лише в кроці йоти функції округлення.

Таблиця 2.1 – Кількість раундів у Keccak-*f* (для SHA-3: b=1600 і n_r = 24)

Ширина стану *b* [біт]	**Раунди n_r**
25	12
50	14
100	16
200	18
400	20
800	22
1600	24

Кожен раунд складається з послідовності з п’яти кроків, позначених грецькими літерами: θ (*meta*), ρ (*po*), π (*ni*), χ (*xi*) та ι (*йота*). Кожен крок маніпулює всім станом. Стан можна розглядати як тривимірний масив. Масив станів складається з $b = 5\times5\times w$ бітів, де $w = 2^l$ (рис. 2.6).

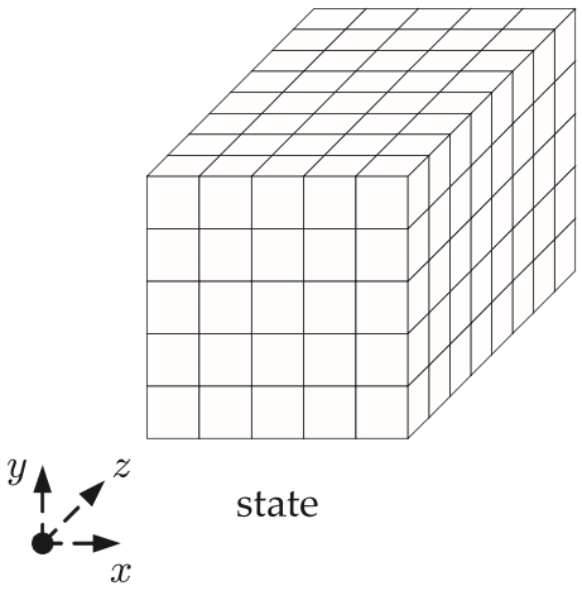


Рисунок 2.6 – Стан Keccak, де кожен кубик представляє один біт

Для SHA-3 станом є масив 5×5×64 біт. Потрібно вибрати $l = 6$ для SHA-3 і, отже, $w = 64$ біти

Біти w для даної координати (x, y) називаються смугою (тобто біти в слові вздовж осі z). Далі опишемо п’ять кроків θ, ρ, π, χ та ι функції Keccak-f. Спочатку потрібно обчислити θ крок, порядок виконання решти чотирьох кроків не має значення.

Крок 1. Крок θ (*Тета*).

Якщо розглянути стан як двовимірний масив (точніше: масив 5×5), де кожен елемент масиву складається з одного слова з w бітами, як показано на рисунку 2.6. Якщо позначити цей масив $A(x, y)$, де $x, y = 0, 1, \dots, 4$, крок θ виконує таку операцію:

$$C[x] = A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4], \quad (2.4)$$

$$D[x] = C[x - 1] \oplus rot(C[x + 1], 1), \quad (2.5)$$

$$A[x, y] = A[x, y] \oplus D[x], \quad (2.6)$$

де $x, y = 0, 1, 2, 3, 4$.

$C[x]$ і $D[x]$ є одновимірними масивами, які містять п’ять слів довжиною w біт. $\oplus$ позначає порозрядну операцію XOR двох w-бітових операндів, а r$ot(C[],1)$ позначає поворот операнда на один біт. Це обертання відбувається в напрямку осі z. Всі індекси беруться за модулем 5, наприклад, $C[-1]$ відноситься до $C[4]$.

Кожен біт замінюється сумою *XOR* 10 бітів «по сусідству» та самого вихідного біта. Один додає до біта, що обробляється, п’ять бітів, які утворюють стовпець ліворуч, плюс стовпець, який знаходиться праворуч, і одну позицію «попереду» (рис. 2.7). В стані загалом $25w = 25 \cdot 64 = 1600$ біт.

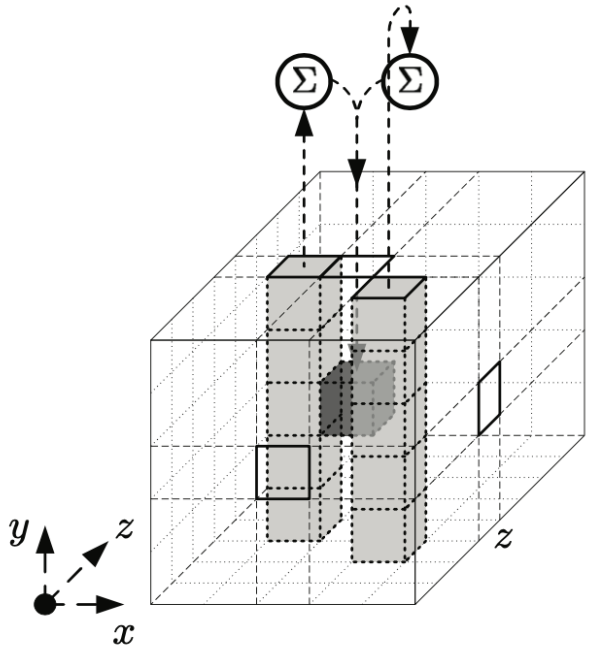


Рисунок 2.7 – Крок θ функції Keccak-*f*

Крок 2-3. Кроки *Rho* (ρ) і *Pi* (π).

Наступні два кроки обчислюють допоміжний масив B 5 × 5 із массиву станів A. $B[i, j]$ відноситься до слова з w бітами. Обидва кроки можуть бути виражені разом наступним псевдокодом.

$$B[y, 2x + 3y] = rot(A[x, y], r[x, y]), \tag{2.7}$$

де $x, y = 0, 1, 2, 3, 4$.

$rot(A[],i)$ повертає одне слово A на і бітових позицій. Кількість обертань визначається $r[x, y]$, що є таблицею з цілими значеннями, які називаються зсувами обертання (табл. 2.2). Записи в таблиці є константами.

Робота кроків ρ і π: вони беруть кожну з 25 смуг (тобто слова з w бітами) масиву стану A, повертають його на фіксовану кількість позицій , і розмістіть повернуту смугу в іншу позицію в новому масиві B. Як приклад, розглянемо смугу в місці [3,1], тобто w-бітне слово $A[3, 1]$. По-перше, це слово повертається на 55 бітових позицій, для $x = 3$, $y = 1$. Повернуте слово

потім розміщується в масиві B у місці $B[1,2 \cdot 3 + 3 \cdot 1] = B[1, 4]$. Індекси обчислюються за модулем 5.

Таблиця 2.2 – Константи обертання (зміщені обертання)

	***x*=3**	***x*=4**	***x*=0**	***x*=1**	***x*=2**
***y*=2**	25	39	3	10	43
***y*=1**	55	20	36	44	6
***y*=0**	28	27	0	1	62
***y*=4**	56	14	18	2	61
***y*=3**	21	8	41	45	15

Крок 4. *Чи* (χ) крок.

Крок χ маніпулює масивом B, обчисленим на попередньому кроці, і розміщує результат у масиві стану A. Крок χ працює зі смугами, тобто словами з w бітами. Псевдокод кроку виглядає наступним чином:

$$A[x, y] = B[x, y] \oplus ((B^{-}[x + 1, y]) \wedge B[x + 2, y]), \qquad (2.8)$$

де $x, y = 0, 1, 2, 3, 4$

B $[i, j]$ позначає порозрядне доповнення смуги за адресою $[i, j]$, $a \wedge \epsilon$ булевою AND (bitwise Boolean AND) двох операндів. Як і на всіх інших кроках, індекси слід приймати за модулем 5. Крок χ бере смугу в точці $[x,y]$ і виконує її XOR за допомогою логічного AND смуги за адресою $[x + 2, y]$ і зворотний у точці $[x + 1, y]$ (рис. 2.8).

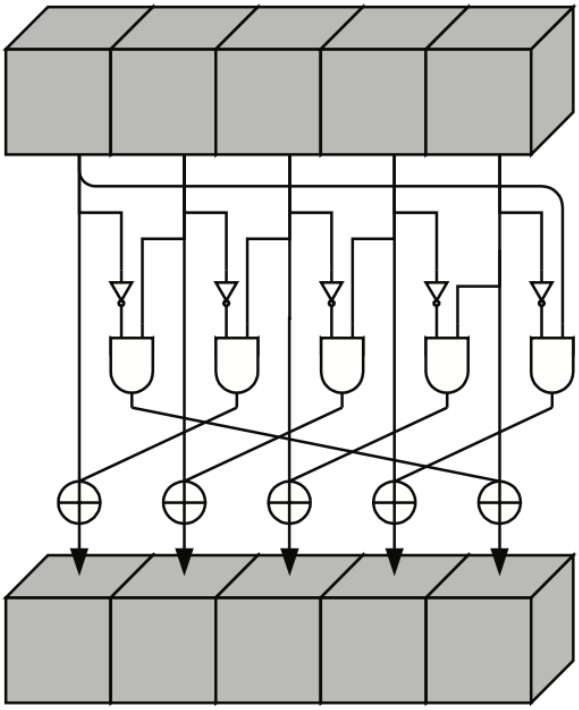


Рисунок 2.8 – Крок χ Keccak-*f*. Верхній рядок представляє п’ять доріжок масиву *B*, тоді як нижній рядок показує п’ять доріжок стану *A*

Крок 4. Крок йоти (ι).

Крок йота додає попередньо визначену константу *w*-bit – смуга в точці [0, 0] масиву станів *A*: $A[0, 0] = A[0, 0] \oplus RC[i]$.

Константа $RC[i]$ відрізняється залежно від того, який раунд і виконується (табл. 2.3). Кількість раундів n_r змінюється залежно від параметра *b*, обраного для Keccak. Для SHA-3 є $nr = 24$ раунди.

Keccak цілком піддається програмній реалізації. Він поділяє цю властивість з іншими хеш-алгоритмами SHA. Високо оптимізована реалізація SHA-3 на сучасних процесорах Intel Core може виконуватися зі швидкістю приблизно 13 циклів/байт, що означає, наприклад, пропускну здатність приблизно 230 Мбайт/с (або приблизно 1,84 Гбіт/с). якщо процесор працює на частоті 3 ГГц [7].

Keccak дуже добре підходить для апаратних реалізацій. Алгоритм значно ефективніший для апаратного забезпечення, ніж SHA-2.

Таблиця 2.3 – Відповідні константи RC[0]...RC[23]

RC[0] = 0x0000000000000001	RC[12] = 0x000000008000808B
RC[1] = 0x0000000000008082	RC[13] = 0x800000000000008B
RC[2] = 0x800000000000808A	RC[14] = 0x8000000000008089
RC[3] = 0x8000000080008000	RC[15] = 0x8000000000008003
RC[4] = 0x000000000000808B	RC[16] = 0x8000000000008002
RC[5] = 0x000000008000000	RC[17] = 0x8000000000000080
RC[6] = 0x8000000080008081	RC[18] = 0x000000000000800A
RC[7] = 0x8000000000008009	RC[19] = 0x800000008000000A
RC[8] = 0x000000000000008A	RC[20] = 0x8000000080008081
RC[9] = 0x0000000000000088	RC[21] = 0x8000000000008080
RC[10] = 0x0000000080008009	RC[22] = 0x0000000080000001
RC[11] = 0x000000008000000A	RC[23] = 0x8000000080008008

3 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ЗАСТОСУНКУ ДЛЯ ПІДТВЕРДЖЕННЯ ОРИГІНАЛЬНОСТІ ДОКУМЕНТІВ

3.1 Створення аккаунту в Ethereum

Популярність технології блокчейн зумовлена тим фактом, що вона потенційно може вирішувати різноманітні проблеми реального світу, оскільки забезпечує більшу прозорість і безпеку (захист від несанкціонованого доступу), ніж звичайні технології [6].

Щоб реалізувати потенціал блокчейну, створюються програми, які працюють поверх блокчейнів. Програми, які взаємодіють із блокчейном, називаються «децентралізованими програмами» – dApps, оскільки ці програми покладаються не на централізовану базу даних, а на децентралізоване сховище даних на основі блокчейну. Для цих програм немає єдиної точки відмови або контролю. Децентралізовані програми призначені для взаємодії з вузлами блокчейну.

На сьогодні розроблено такі варіанти взаємодії з блокчейном [37-40]:

– застосунок і вузол запускаються локально: додаток і вузол працюють на локальній машині. Користувачі програми запускають локальний вузол блокчейну та спрямовують програму на з’єднання з ним. Ця модель буде чисто децентралізованою моделлю запуску програми. Прикладом цієї моделі є браузер Mist на основі Ethereum, який використовує локальний вузол geth (рис. 3.1);

– загальнодоступний вузол: програма спілкується з публічним вузлом, розміщеним третьою стороною. Користувачам не потрібно розміщувати локальний вузол.

У кожного підходу є переваги і недоліки. Користувачам не потрібно платити за електроенергію та сховище для запуску локального вузла, але потрібно довіряти третій стороні для трансляції їхніх транзакцій у блокчейн.

Плагін браузера Метамаск використовує цю модель і підключається до загальнодоступних вузлів (рис. 3.2).

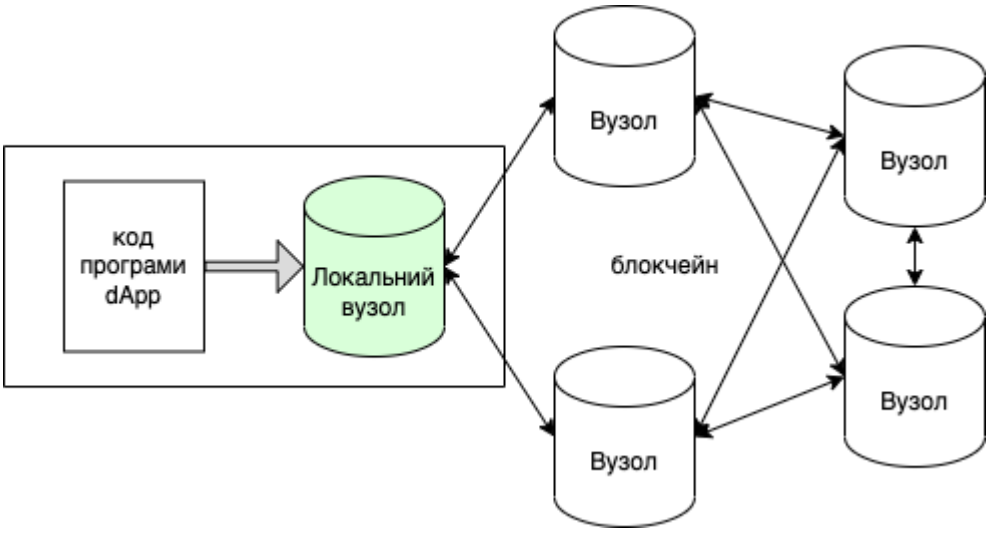


Рисунок 3.1 – DApp підключається до локального вузла

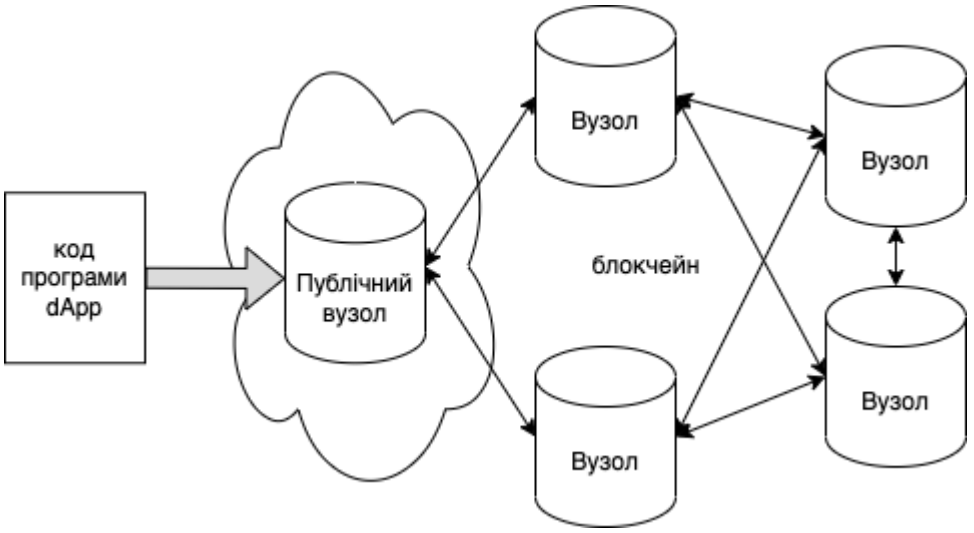


Рисунок 3.2 – DApp підключається до публічного вузла

Треба створити обліковий запис Ethereum (EOA), який може зберігати Ether. Лише EOA може ініціювати транзакції в Ethereum. Цей обліковий запис потрібен для створення смарт-контрактів і транзакцій пізніше в розробці DApp. Створення EOA є безкоштовним.

У дослідженні використовується другий метод.

Для створення облікового запису можна скористатися крипто-гаманцем MetaMask. Його можна встановити як розширення до браузеру Chrome, Firefox і Opera. Після встановлення MetaMask створюється обліковий запис. Під час створення облікового запису генеруються секретні слова – початкова фраза з 12 слів (seed phrase). Дуже важливо зберегти її і бажано не в цифровому вигляді.

Програмне забезпечення гаманця має список слів, взятих зі словника, кожному слову присвоєно номер. Початкову фразу можна перетворити на число, яке використовується як початкове ціле число для детермінованого гаманця, який генерує пари ключів, що використовуються в гаманці. Тобто ці слова і є ключем до облікового запису, по сід-фразі можна відновити обліковий запис та отримати доступ до нього. Використовуються слова а не просто набір чисел, щоб зменшити вірогідність опечатки (рис. 3.3).

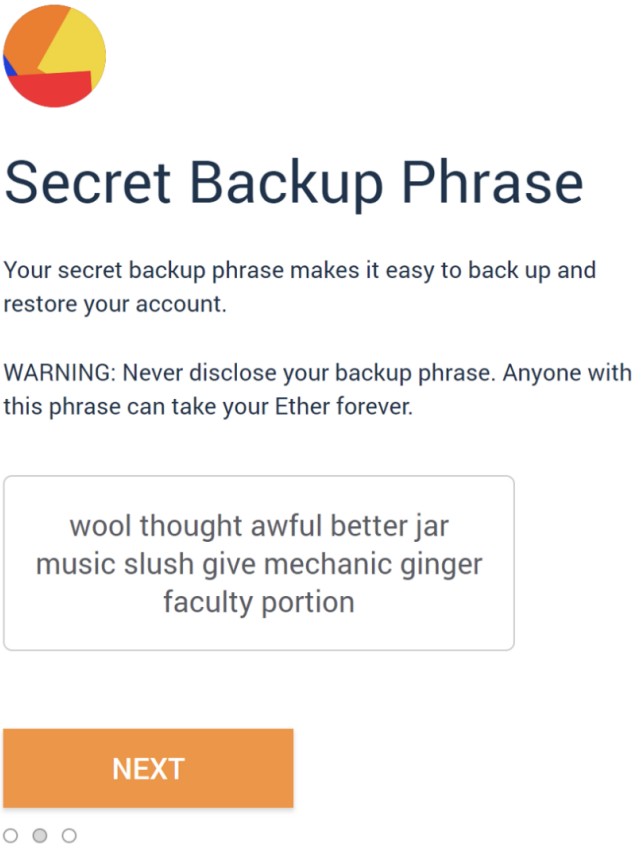


Рисунок 3.3 – Приклад секретної резервної фрази облікового запису

Генерується пара ключів: закритий та відкритий. Закритий ключ – секретний номер, за допомогою якого робиться цифровий підпис транзакції, наприклад: ac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80.

Відкритий ключ – число, отримане через односторонню функцію з закритого ключа. Його можна відкрити для всіх і використовувати для перевірки цифрового підпису, зробленого за допомогою відповідного закритого ключа. Адреса Ethereum – це шістнадцяткова адреса з 42 символів, яка походить від останніх 20 байтів відкритого ключа, який керує обліковим записом з 0x попереду, наприклад: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266.

Тепер можно використовувати крипто-гаманець (рис. 3.4). В гаманці можно переключати мережі, переключати акаунти, скопіювати публічний ключ гаманця, побачити кількість доступного ефіру – основної валюти блокчейну, яка необхідна для проведення транзакцій.

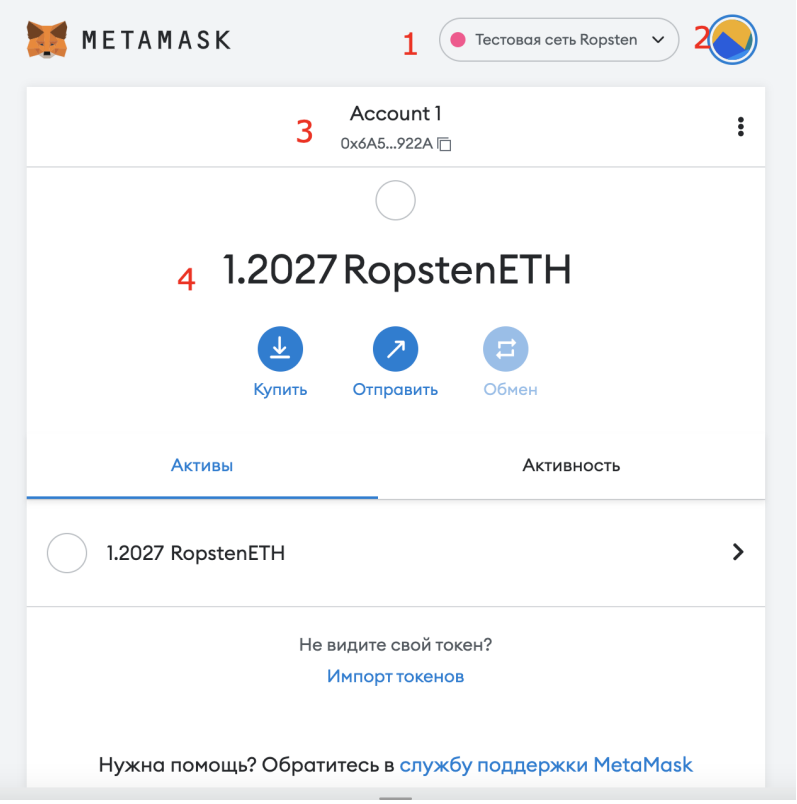


Рисунок 3.4 – Інтерфейс крипто гаманця metamask

3.2 Розроблення прикладного смарт-контракту

Створення смарт контракту здійснюється на мові solidity. Solidity – об’єктно-орієнтована та предметно-орієнтована мова програмування розумних контрактів для платформи Ethereum [41-43]. Спочатку створюємо файл з розширення .sol – Documents.sol. Далі описуємо клас Documents з функцією constructor, яка буде викликатися один раз при створенні смарт контракту. В конструкторі зберігаємо змінну адресу акаунту, що створює смарт-контракт для того, щоб надалі була змога дозволяти зберігати документи лише з акаунту, з якого було створено смарт контракт. Для цього використовуємо msg.sender. Об’єкт msg – це глобальна змінна, яка містить дані, важливі для доступу до блокчейну:

– msg.sender використовується найчастіше. Це адреса початкового відправника;

– msg.value це зазвичай містить кількість ефіру, який надсилається разом із повідомленням;

– msg.sig поверне хеш сигнатури поточної функції;

– msg.data поверне корисне навантаження в байтах. Він містить дані виклику, як це зазначено в термінології EVM, або параметри, які передаються функції. Якщо передати складні структури даних у функцію, отриманий msg.data буде структурований наступним чином: вхідні параметри в байтах, які розмежовуються 32-байтовим цілим числом.

Також у смарт контракт додається *idCounter*, який буде використовуватися як унікальний ідентифікатор для документів, які додаються, та буде починатися з 0 та збільшуватися на 1 для кожного нового документу

Лістинг 3.1 Конструктор смарт-контракту:

```
pragma solidity 0.8.7;
contract Documents {
```

```
    address private owner;
    uint256 public idCounter;
    constructor() {
        owner = msg.sender;
        idCounter = 0;
    }
}
```

Документи у вигляді хешу будуть зберігатися у хеш-таблиці, які складаються з типів ключів і відповідних пар типів значень. Ключем буде виступати унікальний ідентифікатор (число від 0 до 2**256 – 1) та хеш документу:

Лістинг 3.2 Мапінг документів у смарт-контракті:

```
mapping(uint256 => bytes32) public documents;
```

Смарт-контракт повинен включати наступні функції:

– addDocument – функція, яка хешує дані про документ та зберігає їх, захищена и може викликатися лише аккаунтом, який створив смарт контракт;

– checkDocument – функція яка перевіряє чи є цей документ у маппингу, порівнюючі збережене хеш значення та хеш з нової строки.

Перетворювати строку на хеш будемо за допомогою вбудованої в solidity функції keccak256. Криптографічна хеш-функція – це алгоритм, який приймає будь-яку кількість даних і виводить зашифрований текст фіксованого розміру. На рисунку 3.5 видно, що найдрібніша зміна даних спричиняє повну зміну результату.

Лістинг 3.3 Додавання і перевірка документів у смарт-контракті:

```
function addDocument(string memory documentInfo) public isOwner returns
(uint256) {
```

```
    idCounter++;
    documents[idCounter] = keccak256(abi.encodePacked(documentInfo));
    emit CreateDocument(idCounter);
    return idCounter;
}
function checkDocument(uint256 _idCounter, string memory documentInfo)
public view returns (bool) {
        bool res = documents[_idCounter] ==
        keccak256(abi.encodePacked(documentInfo));
    return res;
}
```

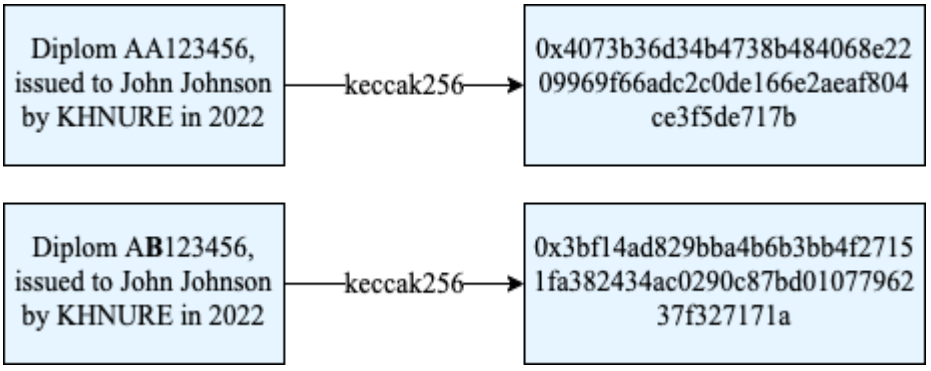


Рисунок 3.5 – Приклад хешування даних методом keccak256

Після написання смарт контракту його код потрібно скомпілювати. Результатом компілювання буде ABI та байт-код контракту, щоб його можно було розгорнути в мережі. Компілювати і деплоїти у мережу смарт контракт будемо за допомогою Hardhat і ethers.js.

Hardhat – це середовище розробки програмного забезпечення Ethereum. Воно складається з компонентів для редагування, компіляції, налагодження та розгортання смарт-контрактів і dApps, які разом створюють повне середовище розробки [41].

Лістинг 3.4 Компілювання смарт-контракту:

```
npx hardhat compile
```

Лістинг 3.5 Деплой смарт-контракту у мережу, виконання deploy.ts:

```
npx hardhat run --network localhost scripts/deploy.ts
```

Лістинг 3.6 Вміст файлу deploy.ts:

```
import { ethers } from "hardhat";
async function main() {
 const Documents = await ethers.getContractFactory("Documents");
 const doc = await Documents.deploy();
 await doc.deployed();
}
main().catch((error) => {
 console.error(error);
 process.exitCode = 1;
});
```

Результатом розгортання смарт контракту в мережі є отримання адреси контракту (рис. 3.6). Теперь у dApp можно викликати функції смарт контракту використовуючи його адресу у мережі та ABI.

```
Contract deployment: Documents
Contract address:    0x5fbdb2315678afecb367f032d93f642f64180aa3
Transaction:         0x5b98a63102f58929932cf3dfdb4ab8f6994b6aa2b85b67dade3735e7e46363fa
From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
Value:               0 ETH
Gas used:            479439 of 479439
Block #1:            0xbfdd219ac80b9c136de265574011af8d3ed7a25be003dad95b14cac6e287c0a8
```

Рисунок 3.6 – Транзакція деплою смарт контракту

3.3 Розроблення клієнтської програми (вебзастосунок)

Для взаємодії користувача зі смарт-контрактом потрібен клієнтський додаток. Вебзастосунок використовує бібліотеку ethers для виклику функції смарт-контракту у браузері.

Вебзастосунок складається з кнопки підключення крипто-гаманця та із двох форм для вводу даних:

– форма для вводу даних документу та їх збереження у блокчейні;

– форма для вводу даних документу та їх перевірки.

Для розроблення документу, що зберігається, створюємо форму з такими полями (рис. 3.7):

– тип документу;

– номер документу;

– виданий кому;

– виданий ким;

– виданий коли.

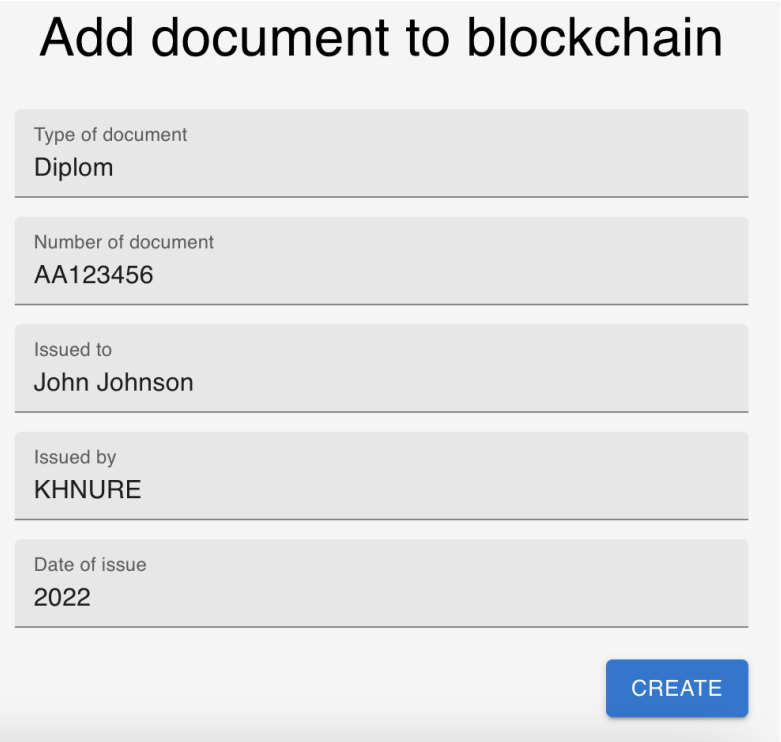


Рисунок 3.7 – Приклад заповнення форми для створення документу

Далі ініціалізуємо об’єкт ethers за допомогою HTTP-провайдера MetaMask. MetaMask додає провайдера у глобальний об’єкт браузера (window.ethereum). Потім створюємо екземпляр об’єкту смарт-контракту і викликаємо функцію створення документу та отримання id створеного документу.

Лістинг 3.7 Зберігання документу:

```
const provider = new ethers.providers.Web3Provider(window.ethereum);
await provider.send("eth_requestAccounts", []);
const signer = provider.getSigner();
let contract = new ethers.Contract(contractAddress, abi, provider);
let contractWithSigners = new ethers.Contract(
  contractAddress, abi, signer
);
const tx = await contractWithSigners.addDocument(`${documentType}
  ${documentNumber}, issued to ${documentIssuedTo}, by
  ${documentIssuedBy}, in ${documentIssuedDate}`
);
await tx.wait();
const documentId = await contract.idCounter();
```

Виклик функції смарт-контракту addDocument є транзакцією і при виклику потребує підпису приватним ключем акаунту, з якого транзакція викликається (рис. 3.8).

Для перевірки оригінальності документу створюємо форму з такими ж полями полями як збереження, плюс унікальний номер запису документу (був створений на попередньому кроці) (рис. 3.9).

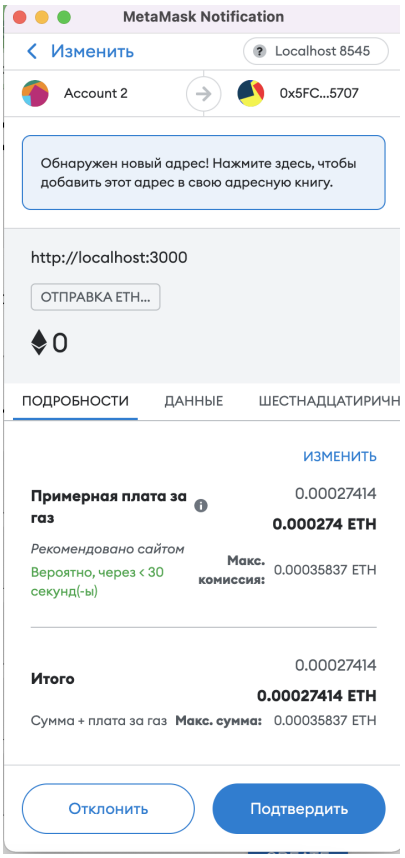


Рисунок 3.8 – Підтвердження транзакції в MetaMask

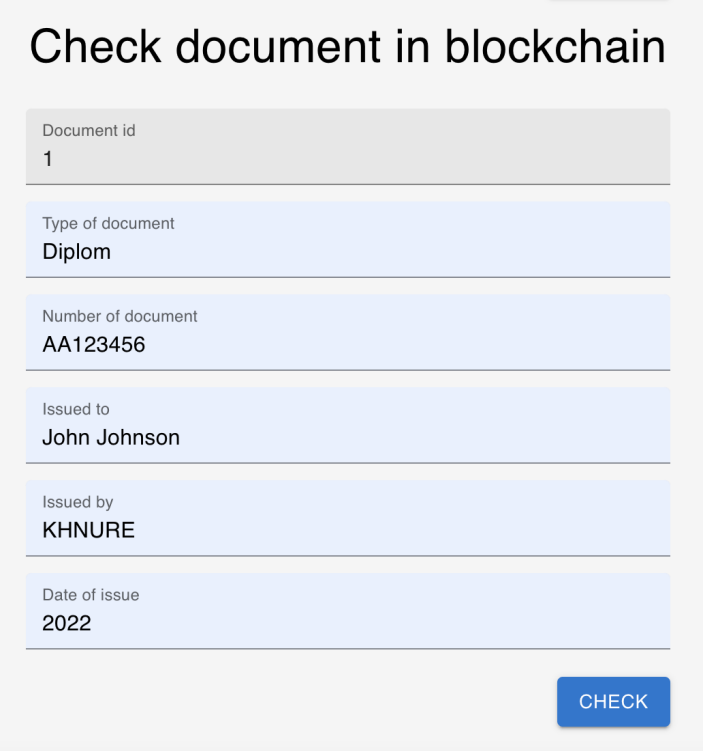


Рисунок 3.9 – Приклад заповнення форми для перевірки документу

Для перевірки документу також створюємо екземпляр об’єкту смарт-контракту за допомогою ABI контракту та HTTP-провайдера MetaMask. І викликаємо функцію перевірки документу (checkDocument). Аргументи функції – це ідентифікатор створеного документу і параметри документу (вид, номер, кому, ким і коли виданий).

Лістинг 3.8 Перевірка документу:

```
const provider = new ethers.providers.Web3Provider(window.ethereum);
await provider.send("eth_requestAccounts", []);
let contract = new ethers.Contract(contractAddress, abi, provider);
await contract.checkDocument(Number(data.id), str)
```

Виклик функції перевірки документу не є странзакцією, тому що не записує нічого у блокчейн, а лише зчитує дані, тому не потребує підпису в крипто гаманці та сплати за газ.

3.4 Розрахунок витрат у мережі Ethereum

Gas – це внутрішня валюта мережі Ethereum, яка використовується для укладання угод та контрактів [16]. Gas є одиницею, яка вимірює обсяг обчислювальних зусиль, які будуть потрібні для виконання певних операцій. Коли користувач надсилаєте токени, викликає функції смарт-контракту, надсилаєте ETH або робите щось інше в блокчейні, він повинен заплатити за ці обчислення. Цей платіж розраховується в газі, а газ оплачується в ETH. Оплата за обчислення відбувається незалежно від того, успішна чи невдала транзакція. Навіть якщо це не вдається (наприклад відхиляється, бо має помилку в розрахунках), майнери повинні обчислити транзакцію (підтвердити та виконати), і тому користувач повинен заплатити за це

обчислення так само, як би заплатив за успішну транзакцію. Оплата за роботу відбувається незалежно від успішності транзакції.

Одна із ідей плати за Gas полягає в тому, щоб обмежити нескінченні цикли. EVM є Т'юрінг повним. Так, наприклад, 1 Gas може виконувати рядок коду чи деяку команду. Якщо обліковий запис має недостатньо ефіру, щоб виконати транзакцію або надіслати повідомлення, то транзакція вважається недійсною. Задум полягає в тому, щоб припинити атаки на відмову в обслуговуванні нескінченних циклів, підвищити ефективність коду та змусити недобросовісного користувача платити за ресурси, які він використовує, від пропускної спроможності до обчислень центрального процесору та зберігання даних.

Кількість газу, яку доведеться заплатити, зростає пропорційно зі зростанням складності команди, яку потрібно виконати. Наприклад, якщо користувач хоче надіслати іншому користувачеві 1 Ether – загальна сума в 1.00001 Ether повинна бути сплачена користувачем. Однак якщо користувач хоче взаємодіяти зі смарт-контрактом, буде більше рядків виконуваного коду і більше споживання енергії, розміщеного у розподіленій мережі Etherium, і, отже, користувач повинен буде заплатити більше ніж 1 Gas для виконання транзакції.

Деякі обчислювальні кроки коштують дорожче, ніж інші, або тому, що є дорогими обчислювальними, або тому, що вони збільшують обсяг даних, які повинні зберігатися в блокчейні.

На витрати у мережі Ethereum впливають:

– Gas limit – це кількість газу, валідатор виконає для конкретної транзакції та залежить від важкості розрахунків в функції смарт контракті;

– Gas price – визначається поточним станом мережі, її загруженістю та конкурентними транзакціями;

– ціна ETH – ціна визначається ринковою пропозицією та попитом на токен, подібно до того, як визначається ціна акцій.

В застосунку перевірка документів – виклик функції checkDocument не не є тран транзакцією і не потребує сплати Gas-у, а запис даних документу у блокчейн – тобто функція addDocument, потребує сплати, тому розрахуєто вартість Gas-у для ціє функції.

Вартість газу за транзакцію визначається за формулою:

$$G_{cost} = G_{limit} \times G_{price}, \quad (3.1)$$

де G_{limit} – це максимальна кількість газу, яку користувач готовий витратити на певну транзакцію,

G_{price} – ціна на газ – кількість ефіру, яку користувач готовий заплатити за кожну одиницю газу, і зазвичай вимірюється в «Gwei» (1 Gwei = 0.000000001 ETH).

Максимальна кількість газу (G_{limit}), яка потрібна на певну транзакцію розраховується за формулою:

$$G_{limit} = G_{transaction} + G_{txdatanozero} \times dataByteLength + opcodesExecs, \quad (3.2)$$

де $G_{transaction}$ = 21000 gas,

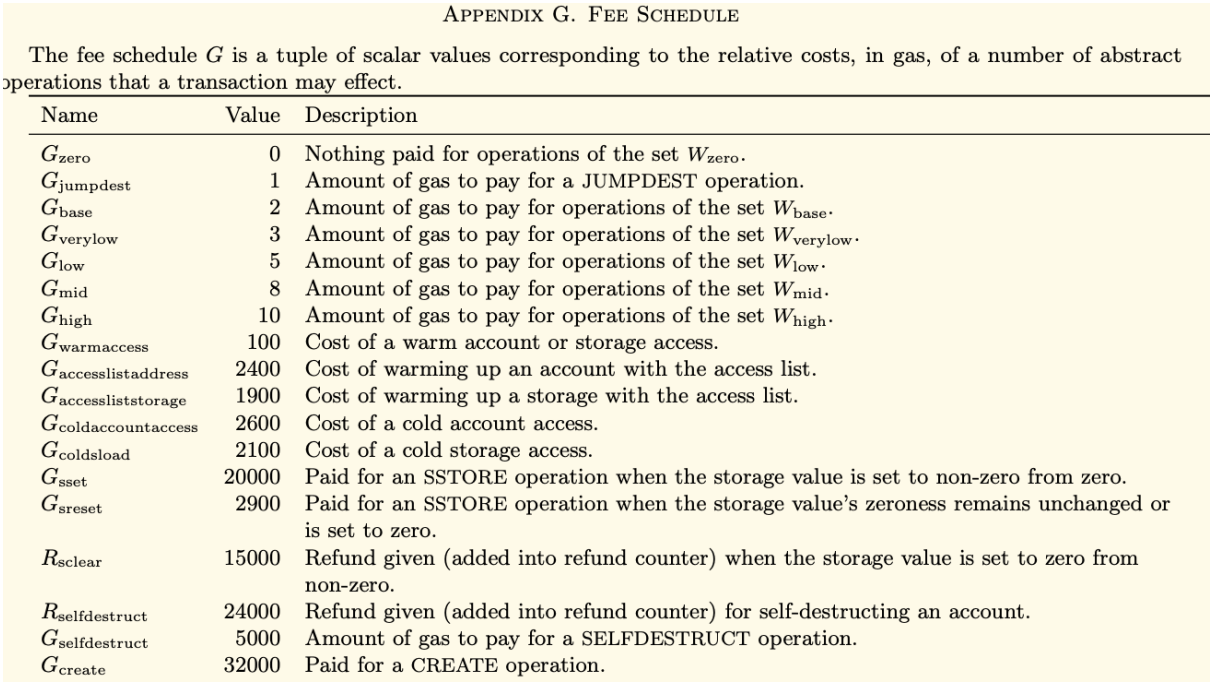
$G_{txdatanonzero}$ = 68 gas,

dataByteLength – розмір даних у байтах,

opcodesExecs – виконання коду в solidity, кожна дія коштує кількість газу в залежності від операції, всі вони розписані в офіційній документації, наприклад 30 газу потрібно сплатити за keccak256 операцію (рис. 3.10).

При здійсненні транзакції користувач може вказати максимальну кількість газу, яку він готовий витратити на транзакцію. Якщо транзакцію виконано, але вона перевищує ліміт газу, усі зміни скасовуються, але користувач все одно повинен буде заплатити за виконане обчислення, яке є

лімітом газу. Якщо транзакція виконується, але вимагає менше газу, ніж ліміт газу, користувач платитиме лише за фактично використаний газ.



APPENDIX G. FEE SCHEDULE

The fee schedule G is a tuple of scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{jumpdest}	1	Amount of gas to pay for a JUMPDEST operation.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
G_{verylow}	3	Amount of gas to pay for operations of the set W_{verylow}.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{\text{warmaccess}}$	100	Cost of a warm account or storage access.
$G_{\text{accesslistaddress}}$	2400	Cost of warming up an account with the access list.
$G_{\text{accessliststorage}}$	1900	Cost of warming up a storage with the access list.
$G_{\text{coldaccountaccess}}$	2600	Cost of a cold account access.
$G_{\text{coldsload}}$	2100	Cost of a cold storage access.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	2900	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{\text{selfdestruct}}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{\text{selfdestruct}}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.

Рисунок 3.10 – Вартість газу для різних операцій в solidity

Тобто ліміт газу залежить від самої транзакції і залишається незмінним для конкретної транзакції. Зменшити ліміт можно лише оптимізацією коду контракту.

Ціна газу (G_{price}) визначається поточним станом мережі. Оскільки простір блоку обмежений, чим більше транзакцій відбувається в будь-який момент часу, тим більша конкуренція буде за транзакції в кожному блоці. Це призводить до вищих комісій, оскільки користувачі змагаються за підтвердження своїх транзакцій, підвищуючи ціни на газ. Чим вища ціна за газ вказана користувачем – тим швидше буде виконана транзакція. При заниженій ціні на газ транзакція може бути взагалі не включена в блок.

У таблиці 3.1 наведено середню ціну за газу та до ймовірності опрацювання транзакції майнерами.

Таблиця 3.1 – Вартість газу для опрацювання транзакції майнерами

Ціна Gas'у	**22,00**	**21,64**	**21,50**	**21,35**	**21,15**	**21,07**
Вірогідність включення транзакції в блок	99%	95%	90%	85%	80%	70%

Щоб порахувати вартість транзакції, спочатку порахуємо загальну плату за Gas: G_{cost} = 72201 × 22 = 1588422 Gwei.

Щоб порахувати вартість транзакції у USD, треба Gwei перевести в ETH (табл. 3.2).

Таблиця 3.2 – Одиниці вимірювання ETH

Назва одиниці вимірювання	**Wei значення**	**Wei**
Wei	1 wei	1
Kwei	1e3 wei	1 000
Mwei	1e6 wei	1 000 000
Gwei	1e9 wei	1 000 000 000
microether	1e12 wei	1 000 000 000 000
milliether	1e15 wei	1 000 000 000 000 000
ether	1e18 wei	1 000 000 000 000 000 000

1588422 Gwei = 0,001588422 ETH. Станом на 07.10.2022 року вартість 1 ETH складала 1440 USD. За таким курсом вартість додавання даних документу складає: 2,29 USD.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований програмний застосунок для підтвердження оригінальності документів з використанням засобів блокчейн-технологій.

Технологія блокчейн обрана для реалізації цього застосунку завдяки таким своїм властивостям як незмінність, стійкість до підробок, захищеність мережі та можливість проведення аудиту. Була обрана платформа Ethereum, яка може сприяти транзакціям не лише грошей, але й акцій, землі, цифрового контенту, транспортних засобів та багатьох інших, які мають певну внутрішню цінність. На відміну від Біткойна, технологія Ethereum дозволяє писати смарт-контракти, які можуть робити що завгодно з точки зору програмування. Для збереження даних використовувалась хеш-функцію Keccak. Алгоритм функції Keccak відповідає всім вимогам до хеш-функції: фіксована довжина хешу, детермінованість, незворотність.

У процесі дослідження розроблено смарт-контракт та вебінтерфейс для збереження та перевірки документів, проведено аналіз витрат та розраховано вартість транзакції збереження документів у блокчейні.

Дослідження показало, що верифікація документів за допомогою блокчейну є надійним, конфіденційним, ефективним, швидким та середнім по витратам. Витрати на застосування не є стабільними і можуть збільшуватись або зменшуватись з часом.

Результати роботи апробовано у вигляді тез доповідей під час XXXIII International Scientific and Practical Conference «Trends in the development of science in the modern world» [5].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Untung Rahardja, S. K., & EkaPurnamaHarahap, Q. (2020). Authenticity of a diploma using the blockchain approach. Int. J., 9(1. 2).

2. Bellucci, M., Bianchi, D. C., & Manetti, G. (2022). Blockchain in accounting practice and research: systematic literature review. Meditari Accountancy Research, 30(7), 121-146.

3. Trivedi, U. B., & Sharma, S. (2023). Digitally Signed Document Chain (DSDC) Blockchain. In Proceedings of Third International Conference on Computing, Communications, and Cyber-Security (pp. 715-727). Springer, Singapore.

4. Vatsaraj, V., Shah, J., Verma, S., & Dholay, S. (2021, July). Decentralized Document Holder Using Blockchain. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.

5. Яресько, К. (2022). Блокчейн для підтвердження оригінальності документів. TRENDS IN THE DEVELOPMENT OF SCIENCE IN THE MODERN WORLD, 33, 420.

6. Кравченко, П. О., Скрябін, Б. Б., & Дубініна, О. М. (2019). Блокчейн і децентралізовані системи. Частина 1. ПРОМАРТ.

7. Swan, M. (2015). Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc.".

8. Tariq, A., Haq, H. B., & Ali, S. T. (2022). Cerberus: A blockchain-based accreditation and degree verification system. IEEE Transactions on Computational Social Systems.

9. Treiblmaier, H., & Clohessy, T. (2020). Blockchain and distributed ledger technology use cases. Springer.

10. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини

дескрипторів ключових точок зображень. Радіоелектроніка, інформатика,управління, No. 2, с. 100–107.

11. Gorokhovatsky, V.O. and Gadetska, S.V. (2019) Determination of Relevance of Visual Object Images by Application of Statistical Analysis of Regarding Fragment Representation of their Descriptions, Telecommunications and Radio Engineering, 78 (3), pp. 211–220.

12. Gorokhovatsky V.A. Putyatin Y. P. (2009) Image Likelihood Measures of the Basis of the Set of Conformities. Telecommunications and Radio Engineering, 68 (9), p. 763–778.

13. Гороховатский В.А. (2003) Распознавание изображений в условиях неполной информации, Харків: ХНУРЭ, 112с.

14. Gorokhovatskyi, V., Putyatin, Y., Gorokhovatskyi, O, Peredrii, O. (2018) Quantization of the Space of Structural Image Features as a Way to Increase Recognition Performance. The Second IEEE International Conference on DataStream Mining & Processing 21-25 August 2018, Lviv, Ukraine. pp. 464 – 467.

15. Гороховатський, В.О., Творошенко, І.С., Чмутов, Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. Сучасні інформаційні системи, т. 6, №3, с. 5–12.

16. Гороховатський В.О., Гадецька С.В., Стяглик Н.І., Власенко Н.В. (2020) Класифікація зображень на підставі ансамблю статистичних розподілів за класами еталонів для компонентів структурного опису. Радіоелектроніка, інформатика, управління, №4 , с. 85–94.

17. Gorokhovatskyi V., Gadetska S., Ponomarenko R. (2020) Recognition of Visual Objects Based on Statistical Distributions for Blocks of Structural Description of Image. Lecture Notes in Computational Intelligence and Decision Making. Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision Making and Problems of Computational Intelligence” (ISDMCI'2019), Ukraine, May 21–25, 2019, pp. 501-512.

18. Gadetska, S.V., Gorokhovatskyi, V. O., Stiahlyk, N. I., Vlasenko, N.V. (2021) Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points. Radio Electronics, Computer Science, Control, №4, pp. 58-68.

19. Gorokhovatskyi, V., Stiahlyk, N., Tsarevska, V. (2021). Combination method of accelerated metric data search in image classification problems. Advanced Information Systems, 5 (3), pp. 5–12.

20. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, Computers, Materials & Continua, 72(1), pp. 1785-1797.

21. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, International Journal of Engineering and Information Systems, 6(2), pp. 40-48.

22. Gadetska S., Gorokhovatskyi V., Stiahlyk N., Vlasenko N. (2022) Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2022), 3137, pp. 68-77.

23. Гороховатський В.О., Творошенко І.С. (2022) Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків, ХНУРЕ, 124с.

24. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, Computers, Materials & Continua, 73 (3), pp. 6069–6084.

25. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), pp. 10-16.

26. Гороховатский В.А. (2014) Структурный анализ и интеллектуальная обработка данных в компьютерном зрении: монография, Харьков, Компания СМИТ, 316с.

27. Гороховатський, В.О., Гадецька, С.В. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень (монографія), Харків, ФОП Панов А.Н., 128 с.

28. Gorokhovatskyi, V., Rusakova, N., Tvoroshenko, I. (2020) The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79 (20), pp. 1801-1811.

29. Gorokhovatsky V.A. Putyatin Y. P. (2009) Image Likelihood Measures of the Basis of the Set of Conformities. Telecommunications and Radio Engineering, 68 (9), p. 763–778.

30. M. A. Ahmad, V. Gorokhovatskyi, I. Tvoroshenko, N. Vlasenko, S. K. Mustafa (2021) The Research of Image Classification Methods Based on the Introducing Cluster Representation Parameters for the Structural Description, International Journal of Engineering Trends and Technology, 69(10), pp. 186-192.

31. Gorokhovatskyi V.A. (2018) Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors. Telecommunications and Radio Engineering, 77 (9), pp. 787-797.

32. Гороховатський В.О., Пупченко Д.В., Солодченко К.Г. (2018) Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення. Системи управління, навігації та зв’язку. С. 93–98.

33. Гороховатский В.А., Передрий Е.О. (2009) Корреляционные методы распознавания изображений путем голосования систем фрагментов. Радіоелектроніка, інформатика, управління, №1 (20), с.74–81.

34. Gorokhovatskyi V.A., Zamula A.A. (2016) Employment of Intelligent Technologies in Multiparametric Control Systems. Telecommunications and Radio Engineering. Vol. 75, No 19, p. 1775–1785.

35. Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. arXiv preprint arXiv:2105.07447.

36. Paar, C., & Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

37. Turkanović, M., Hölbl, M., Košič, K., Heričko, M., & Kamišalić, A. (2018). EduCTX: A blockchain-based higher education credit platform. IEEE access, 6, 5112-5127.

38. Oliver, M., Moreno, J., Prieto, G., & Benítez, D. (2018). Using blockchain as a tool for tracking and verification of official degrees: business model.

39. Serranito, D., Vasconcelos, A., Guerreiro, S., & Correia, M. (2020, September). Blockchain ecosystem for verifiable qualifications. In 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 192-199). IEEE.

40. Brinkkemper, F. L. (2018). Decentralized credential publication and verification: A method for issuing and verifying academic degrees with smart contracts (Master's thesis, University of Twente).

41. Butijn, B. J., Tamburri, D. A., & Heuvel, W. J. V. D. (2020). Blockchains: a systematic multivocal literature review. ACM Computing Surveys (CSUR), 53(3), 1-37.

42. Schär, F., & Mösli, F. (2019). Blockchain diplomas: Using smart contracts to secure academic credentials. Journal of Higher Education Research, 41(3), 48-58.

43. Gresch, J., Rodrigues, B., Scheid, E., Kanhere, S. S., & Stiller, B. (2018, July). The proposal of a blockchain-based architecture for transparent certificate handling. In International Conference on Business Information Systems (pp. 185-196). Springer, Cham.