Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Метод розподілу крипто-токенів для проєктування децентралізованих захищених інформаційних систем

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

КвРКІ. 170261.21.01.02 ПЗ

Виконав: студент 2 курсу, група КІ1м-21-1 Віхтюк А.Р.
Підпис

Керівник: д. т. н, проф Савенко О.С.
Підпис

Нормоконтролер ст. викладач кафедри кібербезпеки Мостовий С.В.
Підпис

До захисту допускаю:
Зав. кафедри кібербезпеки, к.т.н., доц Кльоц Ю.П.
Підпис

6 12 2022 р.

Хмельницький, 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХОЛОГІЙ

Кафедра КІБЕРБЕЗПЕКИ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ МАГІСТРА

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П. Кльоц

"6" 12 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Віхтюку А.Р.
Прізвище, ім'я, по батькові студента

Тема проекту (роботи) Метод розподілу крипто-токенів для проектування децентралізованих захищених інформаційних систем

1. Керівник проекту (роботи) д.т.н., проф. Муляр І.В.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом № 83 ректора університету додаток №25до наказу від 01.07.2022 р.

2. Строк подання студентом проекту (роботи) на кафедру 3.12.2022

3. Вихідні дані до проекту (роботи) веб онтології, семантична мережа, фреймворки, Mashup системи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Проведено аналіз розвитку децентралізованих програм. Це було зумовлено децентралізованою природою нових програм та технологіями, які використовуються для зберігання даних такі, як блокчейн. Визначними в розробці подібних програм є розуміння архітектури, підходів до проектування децентралізованих систем, а також безпека, надійність та захист даних, оскільки часто річ іде про чужі активи та кошти. На основі вдосконаленої моделі розроблено метод розподілу крипто-токенів у децентралізованих програмах на Ethereum, визначено основні етапи розробки децентралізованих програм, запропоновано алгоритм розподілу крипто-токенів

. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Загальна характеристика кваліфікаційної роботи; модель взаємодії в мережі користувача зі смарт-контрактом; досконалена модель розподілу крипто-токенів; етапи розробки децентралізованих додатків на основі технології блокчейн; вдосконалений алгоритм розробки безпечних децентралізованих додатків; архітектура децентралізованого додатку; структура та залежність між смарт-контрактами програми

6. Консультанти розділів кваліфікаціної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання при
Відповідальний за оформлення ДП	Мостовий С.В.		

7. Дата видачі завдання «1» лютого 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи	Термін виконання етапів проекту (роботи)	Прим
1	Вибір напряму дослідження та узгодження тематики КРМ з керівником	3.02.2022	
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення предмета та об'єкта дослідження	3.03.2022	
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	2.04.2022	
4	Робота над розділом 2 – розробка моделей і методів для вирішення поставленої задачі	2.05.2022	
5	Робота над науковою статтею	2.06.2022	
6	Робота над розділом 3 – розробка алгоритмів та технологій, їх аналіз	3.09.2022	
7	Робота над розділом 4 – проектування ПЗ для вирішення поставленої задачі	01.10.2022	
8	Узгодження отриманих результатів; оформлення пояснювальної записки згідно вимог	02.11.2022	
9	Оформлення графічної частини	12.11.2022	
10	Попередній захист КРМ	25.11.2022	
11	Захист ДРМ на засіданні ЕК	15.12.2022	

Студент _____ А.Р. Віхтюк
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О.С Савенко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод розподілу крипто-токенів для проєктування децентралізованих захищених інформаційних систем

Автор роботи: Віхтюк А.Р.

Керівник роботи: д.т.н., проф. Савенко О.С.

Пояснювальна записка: 75 с., 41 рис., 2 табл., 3 дод., 36 джерел.

БЛОКЧЕЙН, СМАРТ-КОНТРАКТ, ФРЕЙМВОРКИ

Мета кваліфікаційої роботи полягає в підвищенні стійкості децентралізованих програм за рахунок використання надійніших алгоритмів обміну крипто-токенами.

Вдосконалено метод розподілу крипто-токенів у децентралізованих програмах на платформі Ethereum та розроблено алгоритм та програмне забезпечення для платформи Ethereum, яке реалізує розроблений метод.

Дата 30.11.2022 р. Підпис студента 

ANNOTATION

a master's degree work of Vikhtiuk Andrii

entitled «Cryptotoken distribution method for designing decentralized secure information systems».

Mentor: Oleh Savenko

Total volume of work: 75 pages, 41 figures, 2 tables, 3 appendices, 36 references.

BLOCKCHAIN, SMART CONTRACT, FRAMEWORKS

The purpose of the qualification work is to increase the stability of decentralized programs due to the use of more reliable crypto-token exchange algorithms.

The method of distributing crypto-tokens in decentralized applications on the Ethereum platform has been improved, and an algorithm and software for the Ethereum platform that implements the developed method have been developed.

Date 30.11.2022р. Signature

ЗМІСТ

ВСТУП

Блокчейн, біткойн, децентралізована обробка даних – це ідеї та технології, потенціал яких випередив свій час і постійно розвивається. Вони не лише змінили бачення традиційної економіки та фінансового сектору, а й змінили парадигму програмування. Разом вони створюють ефективну та надійну модель взаємодії, якій довіряють мільйони людей у всьому світі. Сьогодні технології блокчейну стрімко розширюються і не обмежуються транзакціями лише з біткойнами, а біткойн – не єдина криптовалюта. В роботі [1] надано список із понад 1000 криптовалют, які працюють за допомогою технології блокчейн.

Досягнення Blockchain засновані на руйнуванні організаційної парадигми централізованих систем зберігання та обробки даних.

Організація та робота централізованих програмних систем проста, базується на відомих методах і алгоритмах, тому вони популярні. Більшість веб-додатків є прикладами таких систем. Централізація за своєю організацією означає, що веб-додаток має один або кілька серверів, які є центрами та серцем системи. На серверах запускається веб-сервер програми, на якому виконується програмний код. Вся взаємодія з системою централізована і зосереджена на використанні цих серверів. Стабільність такої системи безпосередньо залежить від провайдера сервера.

Зберігання всіх даних на локальних серверах і використання ресурсів одного провайдера робить їх ненадійними для централізованих програм і ризиком втрати конфіденційності інформації [2].

Принцип децентралізації як основа технології блокчейн дозволяє створювати ефективні та надійні програмні додатки, але використання цієї можливості залежить від методу розробки децентралізованих додатків.

Блокчейн успішно використовується в багатьох видах фінансових операцій [3], для обслуговування розумних контрактів [4], надання державних послуг [5], обміну конфіденційною інформацією [2], контролю цілісності

відеоархівних даних [6], створення безпеки системи та інші додатки [3,7,8,9].

Однією з головних характеристик, яка призвела до поширення блокчейна, є надійність. Надійність блокчейну забезпечується ланцюговим хешуванням інформаційних блоків, розподіленим зберіганням даних із множинним резервуванням, використанням алгоритму цифрового підпису з асиметричним еліптичним шифруванням (ECDSA) для автентифікації транзакцій [8].

Популярності технології блокчейн додає можливість збереження анонімності при виконанні транзакцій, відсутність потреби в банківських та інших фінансових установах, висока продуктивність систем надання послуг [9].

Технологія блокчейн постійно вдосконалюється та впроваджується. Багато популярних проектів знаходяться в стадії активної розробки та вдосконалення [10]: Ethereum (Wood 2016), Hyperledger (Androulaki et al. 2020), IOTA (IOTA 2021).

Розвиток децентралізованих методів створення додатків призвів до розробки альтернативних підходів до проектування програмних систем блокчейн і появи принципово нових програмних продуктів. Це було викликано децентралізованою природою нових програм і технологій, що використовуються для зберігання даних. Такі функції, як смарт контакт проекту Ethereum, сприяли створенню великих децентралізованих програм [10], децентралізовані системи голосування [Pilkington 2016], децентралізованих систем медичних вимог [Ekblaw et al. 2016].

Незважаючи на те, що у світі існує багато децентралізованих програм і пропозицій, підходи та принципи розробки децентралізованого програмного підходу на основі блокчейн-структур знаходяться лише на початку і мають великі перспективи. Цей процес цілком нормальний для нових технологій, їх адаптація вимагає часу і часто вимагає зміни традиційних підходів. Розуміння такої архітектури означає розуміння архітектури, підходів до проектування децентралізованих систем, а також безпеки, надійності та захисту даних,

оскільки часто йдеться про чиїсь активи та фонди.

Принцип децентралізації як базова основа технології блокчейн – це актуальна задача, вирішення якої допоможе створення ефективних і надійних програмних застосунків, але використання цієї можливості напряму залежить від способу розробки децентралізованих додатків

Мета дослідження. Підвищення стійкості децентралізованих програм за рахунок використання надійніших алгоритмів обміну крипто-токенами.

Для досягнення мети в роботі вирішені наступні завдання:

- проведено аналіз методів та засобів створення децентралізованих програм;
- удосконалено модель функціонування смарт-контрактів для покращення стабільності роботи;
- вдосконалено метод розподілу крипто-токенів у децентралізованих програмах на платформі Ethereum;
- розроблено алгоритм та програмне забезпечення для платформи Ethereum, яке реалізує розроблений метод.
- досліджено ефективність розробленого алгоритму та засобів його реалізації.

Об’єкт дослідження – децентралізовані програми, побудовані з використанням смарт контрактів на основі платформи Ethereum, що працюють з розподілом крипто-токенів.

Предмет дослідження – моделі, алгоритми та методи, що використовуються при реалізації децентралізованих програм на блокчейні.

Методи досліджень. отримані результати в кваліфікаційній роботі базуються на використанні методів криптографії, теорії імовірності, теорії алгоритмів, теорії захисту інформації.

Наукова новизна одержаних результатів. Вдосконалено метод розподілу крипто-токенів, який забезпечує більшу надійність смарт контрактів та систем, побудованих на них.

Практичне значення одержаних результатів полягає в розробці

алгоритму та його реалізації для розподілу крипто-токенів, який покращує стійкість та надійність смарт контрактів і децентралізованих програм, які їх використовують.

1 АНАЛІЗ ПОТОЧНОГО СТАНУ РОЗРОБКИ ДЕЦЕНТРАЛІЗОВАНИХ ПРОГРАМ

1.1 Огляд технології блокчейн

Ідеї децентралізованих криптовалют - була відома в технологічному просторі десятиліттями. Але у 2009 році Сатоші Накамото [11] першим впровадив на практиці децентралізовану систему криптовалюти. Він поєднав визначені засоби управління власністю на основі криптографії з відкритим ключем із консенсусним алгоритмом, який визначав, хто є власником монет [12]. По суті, біткойн — це розподілена децентралізований журнал, в якому реєструються всі фінансові операції**.** Цей журнал реалізований за допомогою технології, відомої як блокчейн. Кожен блок у блокчейні представляє серію транзакцій. Після виконання достатньої кількості транзакцій блок буде завершено, і його неможливо буде змінити.

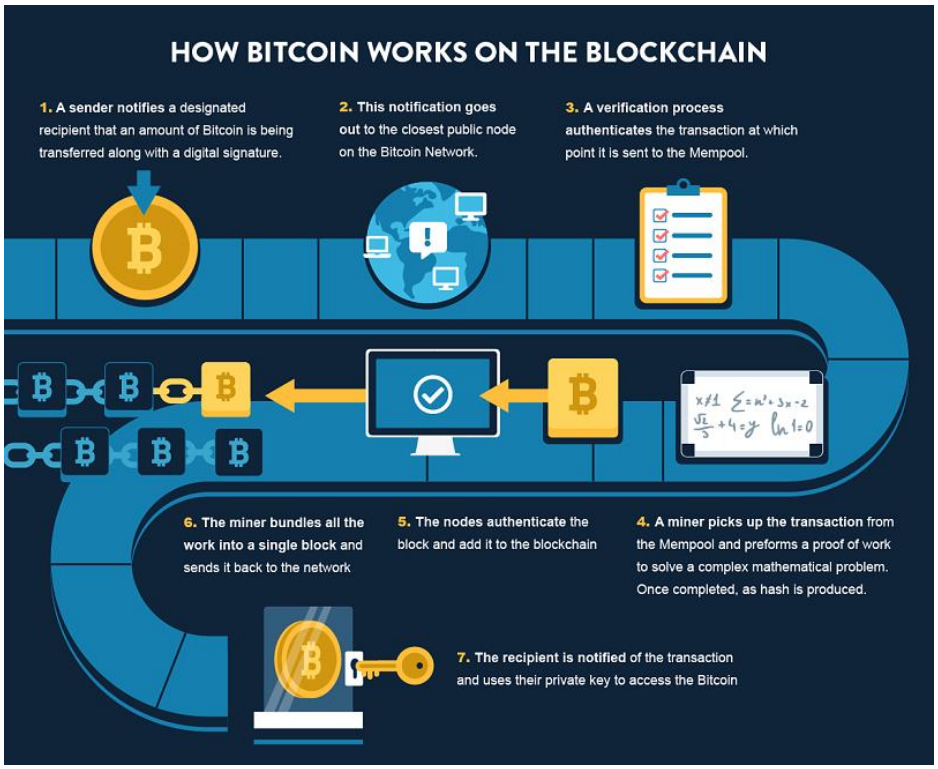


Рисунок 1.1 - Загальна схема роботи системи Біткойн

Блокчейн - багатофункціональна багаторівнева комп'ютерна технологія, призначена для надійного ведення різноманітних активів. Потенційно ця технологія охоплює всі сфери господарської діяльності та має багато сфер застосування. До них відносяться: економіка та фынанси, а також операції з матеріальними (нерухомість, власність, автомобілі тощо) і нематеріальними активами (авторське право, право голосу, медичні дані, особиста інформація тощо). Блокчейн створює нові можливості для пошуку, організації, оцінки та передачі будь-якої цифрової одиниці. Власне кажучи, це нова організаційна парадигма координації будь-якого виду людської діяльності [13]. Зупинемося на визначені, що блокчейн - це одноранговий криптографічно захищений незмінний розподілений реєстр, якаий підтримує лише додавання блоків і оновлюється лише після укладання угоди між усіма учасниками [14] (рис 1.2).

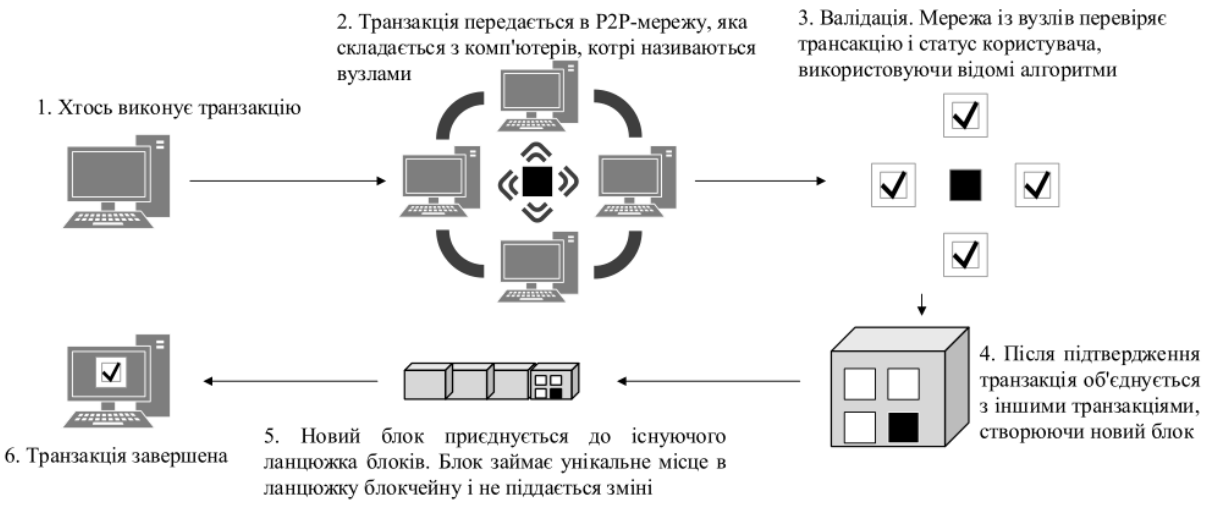


Рисунок 1.3 - Принцип роботи технології блокчейн

Використовуючи публічний реєстр, біткойн прагне вирішити низку проблем:

– Децентралізація - тобто один орган влади чи особа не може контролювати його. Код програмного забезпечення є відкритим і підтримується волонтерами. Система керується відкритою мережею

комп’ютерів, поширених по всьому світу. Кожен, хто хоче взяти участь, може долучитися та почати робити внесок.

– Анонімність. Ваша особистість — це ваша біткойн-адреса. Ваша здатність здійснювати фінансові операції залежить лише від наявності достатньої кількості коштів на вашому рахунку.

– Незмінність. Базовий блокчейн є незмінними. Це означає, що після завершення транзакції її не можна скасувати.

– Обмежена пропозиція. Традиційні валюти мають необмежену пропозицію, оскільки резервні банки можуть друкувати скільки завгодно грошей. Однак кількість біткойнів, які можна коли-небудь виготовити, обмежена 21 мільйоном. Цінність валюти визначається попитом і передбачуваною вигодою, яку люди бачать у ній.

Коли нова інформація надходить у блокчейн, вона має бути перевірена та підтверджена всіма користувачами блокчейну (устаткування, підключене до блокчейну, діє як користувачі, тому всі операції виконуються миттєво). Перевірка та підтвердження інформації здійснюється за допомогою алгоритмів консенсусу: Proof-of-Work (PoW), Proof-of-Stake (PoS) та інших [15]. Алгоритм підтвердження роботи є не поганим вибором, оскільки він забезпечує простий і ефективний алгоритм консенсусу, який дозволяє учасникам мережі колективно перевіряти та погоджувати зміни, що відбулися в системі Bitcoin. Це також дозволяє вільний доступ до консенсусу для нових учасників. Тобото це вирішує проблему дискримінації та вибору того, хто яку владу має мати в мережі.

Як тільки всі користувачі підтвердять правдивість інформації, створюється блок, що містить кілька одиниць інформації (наприклад, кілька транзакцій). Кожен блок містить не тільки отриману інформацію, але також позначку часу та посилання на попередній блок. За допомогою цього ви можете перевірити вміст кожного блоку.

Блокчейн складається з послідовності блоків, що зберігаються та копіюються між публічними серверами.

Кожен блок складається з чотирьох головних елементів: хеш попереднього блоку; дані блоку (наприклад, записи реєстрі); поняття, яке використовується для генерації нового хешу; хеш блоку [16].

Додаючи хеш попереднього блоку, кожен наступний блок посилює запит на дійсність попереднього блоку. Блоки на початку ланцюжка не можна змінити без зміни всіх наступних блоків. Подібним чином додавання даних до хешу робить дані незмінними без порушення послідовності (рис.1.4) .

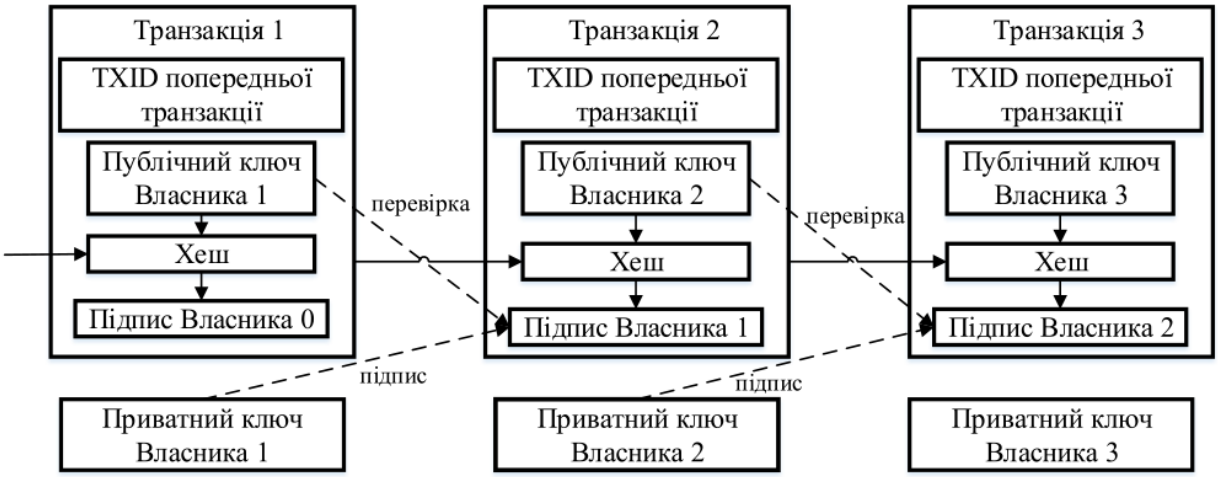


Рисунок 1.4 - Схема підтвердження правомірності здійснення транзакції

Новий блок послідовно приєднується до ланцюжка існуючих блоків. Ланцюжок блоків містить інформацію про всі операції, які виконуються в базі даних (розподіленому реєстрі). Весь ланцюжок з однаковим набором інформації зберігається кожним учасником блокчейну на багатьох комп’ютерах по всьому світу. Перезаписати інформацію в блоці неможливо, оскільки зміна будь-якого блоку призведе до змін у всьому ланцюжку. Оскільки ланцюжок зберігається на багатьох комп'ютерах, інформація, що міститься в ньому, буде різною, і інші учасники ланцюга просто проігнорують її (для них це буде некоректно) [17].

Ще однією особливістю блокчейну є використання пари закритих і відкритих цифрових ключів. Кожен учасник блокчейну має закритий ключ, що

використовується для підпису цифрових повідомлень і відомий лише одному користувачеві. Відкритий ключ є загальнодоступним і використовується для перевірки особи відправника цифрового повідомлення [18].

Крім того, криптографія з відкритим ключем відіграє життєво важливу роль у безпеці блокчейну. Зараз у блокчейні використовується криптографія еліптичної кривої. Її безпека заснована на нерозв'язності задачі про дискретний логарифм еліптичної кривої. Основні функції криптографії з відкритим ключем полягають у наступному.

Використання закритого ключа для створення підпису повідомлення, від якого підписувач не може відмовитися. Захист від зловмисної підробки повідомлення транзакції.

Відкритий ключ використовується для участі в обміні адресами, як адреса для отримання платежів. Закритий ключ використовується для захисту та керування криптовалютою.

Цей цифровий підпис створюється за допомогою алгоритму хешування та асиметричного шифрування (рис.1.5).

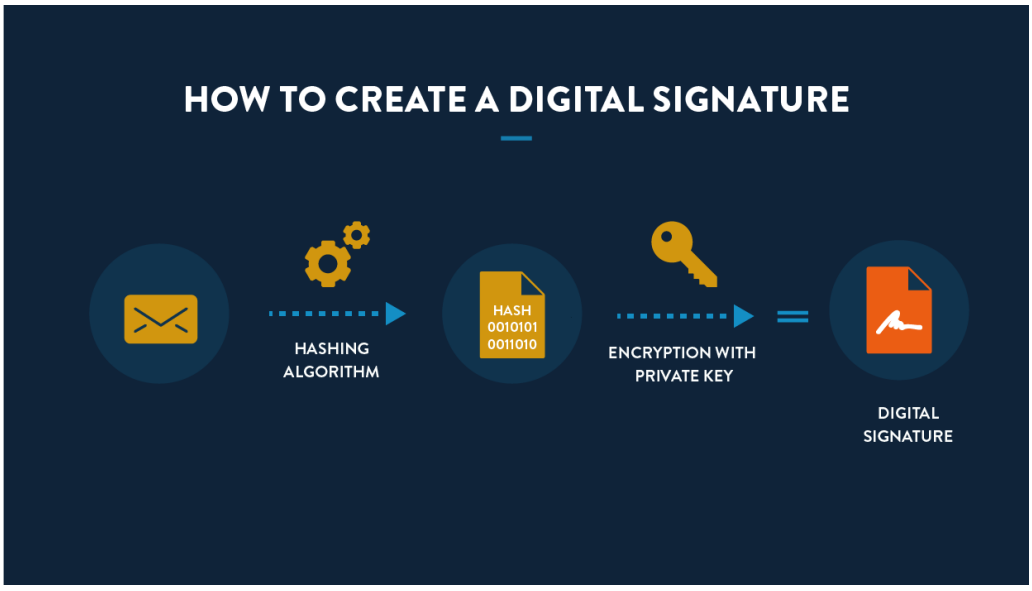


Рисунок 1.5 – Створення цифрового підпису

Цифровий підпис забезпечує дві речі:

1. Визначений відправник надіслав повідомлення.
2. Повідомлення не змінено.

Щоб створити цифровий підпис, передане в мережу повідомлення спочатку потрібно хешувати. Потім хеш необхідно зашифрувати. Процес аутентифікації зображено на рисунку 1.6.

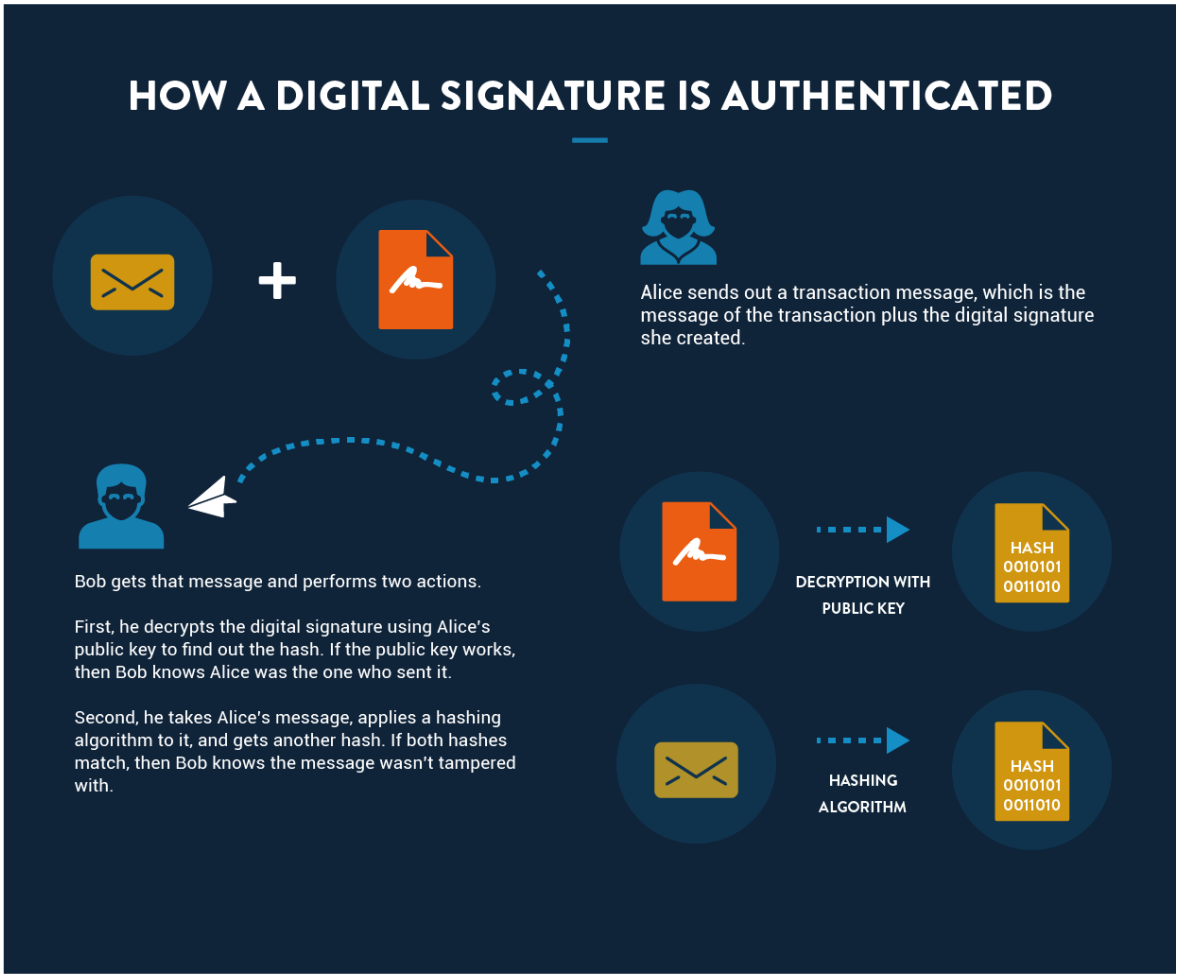


Рисунок 1.6 - Процес аутентифікації цифрового підпису

Це унеможливлює вектор атаки «Sybil», для якого вразливі мережі з визначеною ієрархією прав серед її учасників [18] (рис. 1.7). У найгіршому випадку атаки «Sybil» майже всі вузли, з якими контактує пошуковий вузол, є вузлами злочинців, які намагаються обдурити вузол, надсилаючи схожу неправдиву інформацію. Однак хоча б одного чесного вузла достатньо, щоб переконатися, що його інформація є єдино правильною серед запропонованих

іншими злочинними вузлами. Таким чином, формальна перешкода для участі в мережі перетворюється на економічну перешкоду, при якій вага голосу учасника в консенсусі прямо пропорційна його обчислювальній потужності.

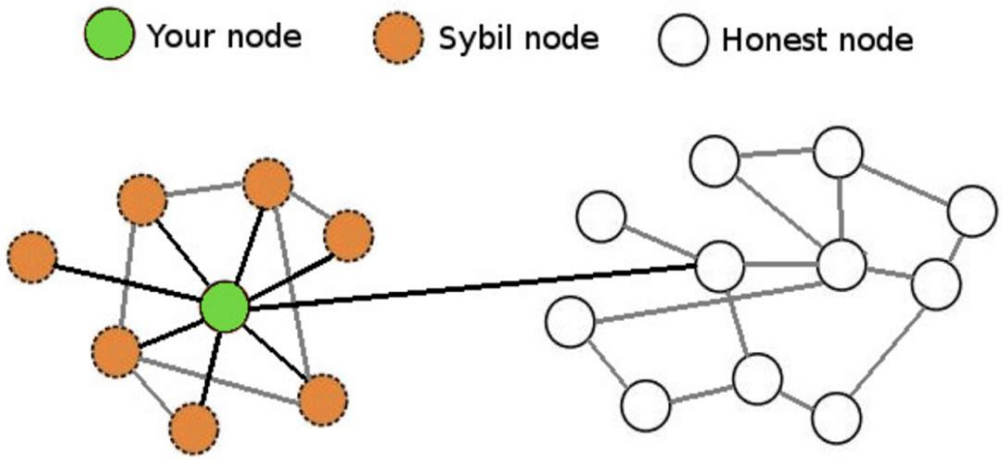


Рисунок 1.7 - Суть атаки “Sybil”

Біткойн також можна розглядати як розподілений реєстр, що складається з таблиці балансу для зареєстрованих облікових записів і загальнодоступного списку транзакцій токенів біткойн між учасниками, що сприяє довірі між ними. Біткойн і його технологічний стек завоював інтерес і популярність серед розробників програмного забезпечення. В результаті почали з'являтися проекти, які взяли за основу блокчейн і використовували її не для переказу грошей. З’явилися так звані «альтернативні валюти» [19], які являли собою окремі блокчейни з побудованою на них власною валютою.

Незважаючи на те, що в світі існує багато децентралізованих програм і пропозицій, на даний момент підходи і принципи розробки децентралізованого програмного підходу на основі блокчейн-структур знаходяться лише на початковій стадії і мають великі перспективи. Цей процес цілком нормальний для нових технологій, їх адаптація потребує часу і часто вимагає зміни традиційних підходів. Важливе значення при розробці таких програм має розуміння архітектури, підходів до проектування

децентралізованої системи, а також безпеки, надійності та захисту даних, оскільки часто залучаються чужі активи та кошти.

1.2 Використання платформи Ethereum для розробки децентралізованих програм

Блокчейн Ethereum — це, по суті, система стану транзакцій. В інформатиці таке поняття, як «система станів» або «державний автомат» – це система, яка обробляє вхідну інформацію і на основі останньої перетворюється в новий стан.

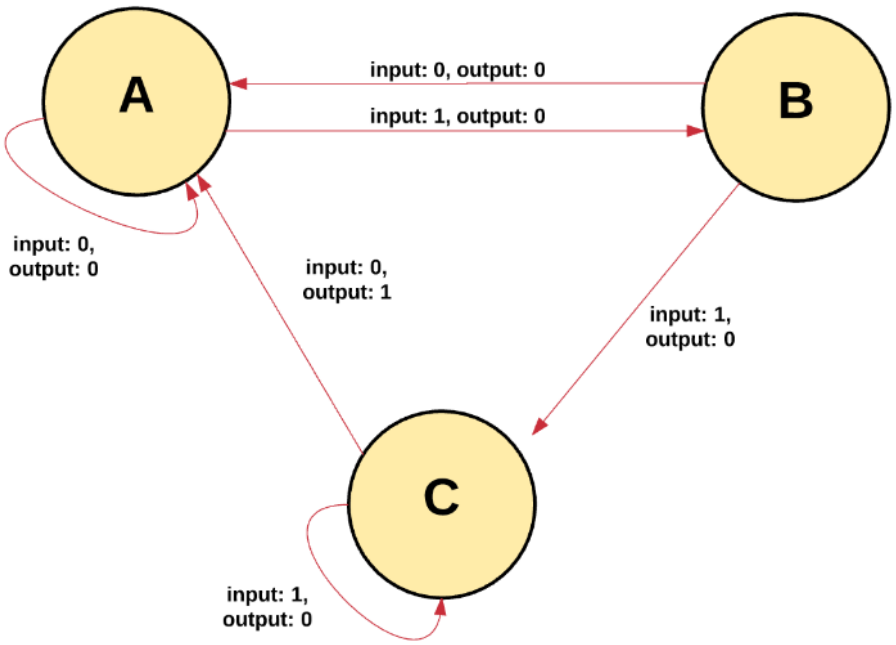


Рисунок 1.8 - Стан транзакцій

Стан Ethereum має мільйони транзакцій. Ці транзакції згруповані в певні «блоки». Блок містить серію транзакцій, причому кожен наступний блок пов’язаний з попереднім, і це забезпечує свого роду ланцюг блоків.

Для переходу з одного стану в інший транзакція має бути дійсною. Транзакція вважається правильною лише тоді, коли вона пройшла процес перевірки, так званий «майнінг». Майнінг — це коли група вузлів

(комп'ютерів) витрачає свої обчислювальні ресурси на створення блоку успішних транзакцій.

Будь-який вузол у мережі, який претендує на майнер, може спробувати створити та підтвердити блок транзакцій. Загальний досвід полягає в тому, що багато майнерів намагаються створити та підтвердити блок транзакцій одночасно. Кожен майнер надає власний математичний «доказ», коли він надсилає блок до блокчейну, і цей доказ діє як свого роду гарантія: якщо доказ існує, транзакції в блоці вважаються правильними.

Глобальний загальний стан платформи Ethereum складається з багатьох невеликих об'єктів – облікових записів, які взаємодіють один з одним через парадигму обміну повідомленнями. Кожен обліковий запис має певний стан і 20-байтову адресу. Адреса Ethereum — це 160-бітний ідентифікатор, який використовується для ідентифікації будь-якого облікового запису [23].

Всього існує два типи облікових записів:

– зовнішні облікові записи контролюються за допомогою закритих ключів. Однак такі записи не мають пов'язаного з ними коду;

– контрактні рахунки, що контролюються спеціальним кодом, зазначеним в умовах договору, і мають асоційований з ними код.

–

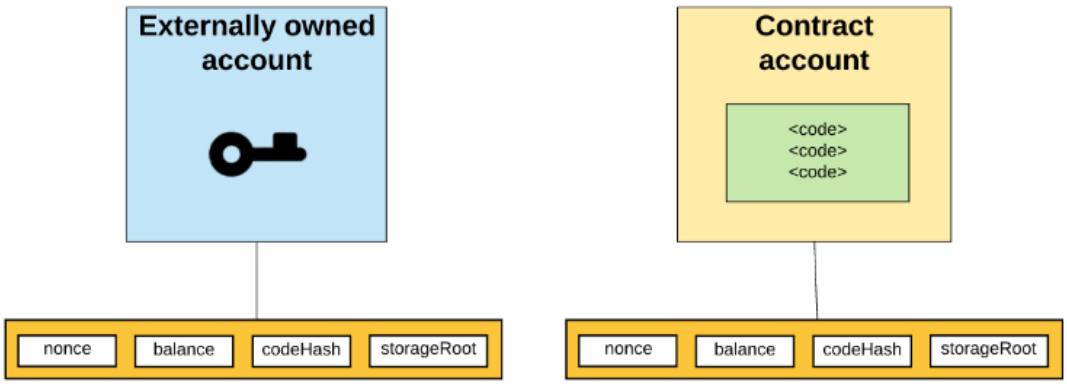


Рисунок 1.9 - Типи облікових записів

Розумний контакт — це лише частина програмного коду, вбудованого в блокчейн. Код містить умови договору. Після їх завершення автоматично

відбувається транзакція [17].

Ethereum значно розширив технологію, що лежить в основі біткоін, щоб запропонувати мережу, яка дозволяє створювати та розгортати інші децентралізовані програми (DApps). Ці програми можуть бути абсолютно новими концепціями або новими версіями вже існуючих. Наприклад, децентралізовані фінанси (DeFi) є одним із найпопулярніших напрямків, які зараз розвиваються в мережі Ethereum [11]. Це традиційні фінансові послуги, які проходять без потреби сторонніх посередників. Уявіть собі, що ви можете позичати гроші в усьому світі, але без банку чи компанії, що стягує високий відсоток комісії. Натомість усі отримані відсотки та комісії за транзакції надходять безпосередньо кредиторам. Це справжня однорангова система, де кожна взаємодія відбувається безпосередньо між користувачами.

Зовнішній обліковий запис має можливість надсилати повідомлення будь-яким обліковим записам. Для цього необхідно створити та записати нову транзакцію за допомогою закритого ключа. Повідомлення між двома зовнішніми обліковими записами – це лише певне значення для передачі. І навпаки, повідомлення, надіслане із зовнішнього облікового запису на контрактний, передбачає обов’язкову активацію коду контрактного облікового запису, при цьому стає можливим виконання деяких дій (наприклад, за допомогою такого повідомлення можна створювати та передавати токени, записувати значення у внутрішню пам’ять, створювати нові контракти тощо).

За допомогою контрактних облікових записів, на відміну від зовнішніх, неможливо самостійно розпочати нові діїї. Натомість контрактні рахунки можуть лише ініціювати транзакції у відповідь (наприклад, отримані із зовнішнього рахунку чи іншого контрактного рахунку).

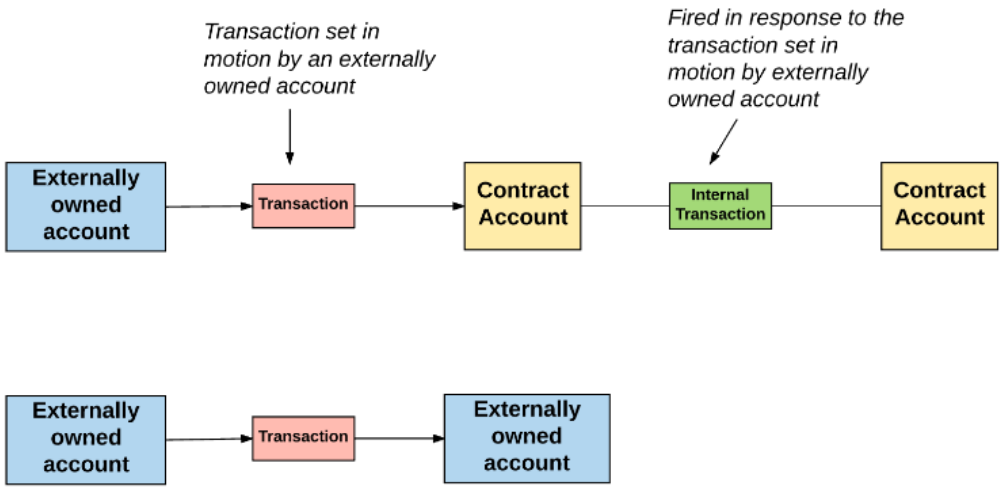


Рисунок 1.10 - Взаємодія облікових записів

Статус кожного з облікових записів, наведених на риунку 1.9 незалежно від типу, може приймати одне з чотирьох значень:

– nonce - кількість транзакцій, надісланих з адреси рахунку, якщо рахунок є зовнішнім. або це кількість контрактів, створених у цьому обліковому записі якщо він є контрактним.

– balance - загальна сума wei, придбана з цього рахунку (ефір (ETH), містить 10^18 wei) [7].

– storageRoot: хеш кореневого вузла префіксного дерева Меркла[8] (рис.2.4).

– codeHash: хеш коду EVM (Ethereum Virtual Machine) облікового запису. Детальніше EVM буде розглянуто в третьому розділі. Для контрактних облікових записів це поле є хешованим кодом і зберігається як codeHash.

Хеш у дереві Меркла поширюється від нижніх гілок до верхніх, і якщо зловмисник намагається замінити вихідну транзакцію фальшивою в нижній частині дерева Меркла, це змінить хеш верхнього вузла, і це, у свою чергу, змінить хеш вузла, розташованого над ним, і так далі по цепочці, поки в кінцевому підсумку це не призведе до зміни кореня.

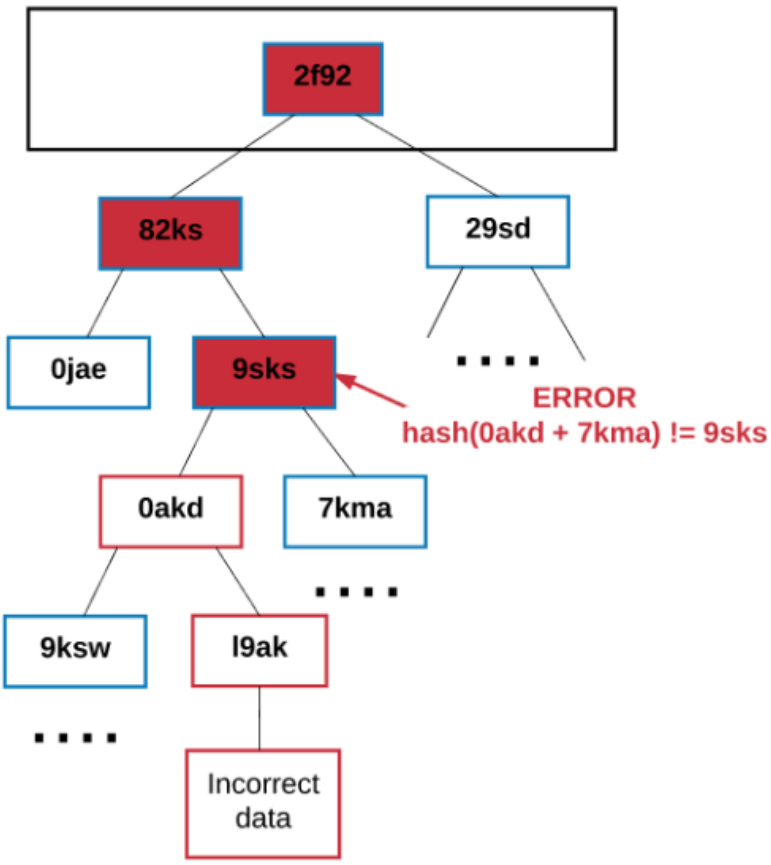


Рисунок 1.11 – Префіксне дерево Меркла

За будь-який розрахунок, який здійснюється в результаті транзакцій в мережі Ethereum, береться певна комісія. Номінал цього платежу називається «газ» (gas). Для будь-якої транзакції відправник повинен встановити ліміт газу, а також ціну. Ціна газу та ліміт — це максимальна сума в wei, яку відправник готовий заплатити за транзакцію. У випадку, якщо відправник не надав необхідну кількість газу для проведення операції, остання буде проведена «без газу» і вважатиметься недійсною. І що також важливо: оскільки до того, як у відправника закінчився газ, машина вже витратила певні зусилля на виконання обчислень, то логічно було б припустити, що збитки, пов'язані з витратою газу, більше не будуть відшкодовуватися відправнику. Всі гроші, витрачені відправником на закупівлю газу, надсилаються на адресу бенефіціара, яким в більшості випадків є адреса майнера. Оскільки майнери виконують обчислення та підтверджують транзакції, саме вони отримують

комісію за «газ» як винагороду.

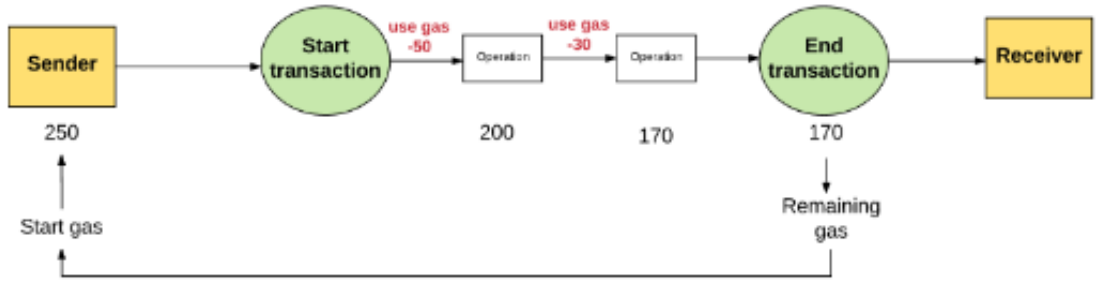


Рисунок 1.12 – Оплата за транзакції

Коли один із контрактів надсилає внутрішню транзакцію іншому контракту, виконується певний код, який існує в обліковому записі контракту одержувача.

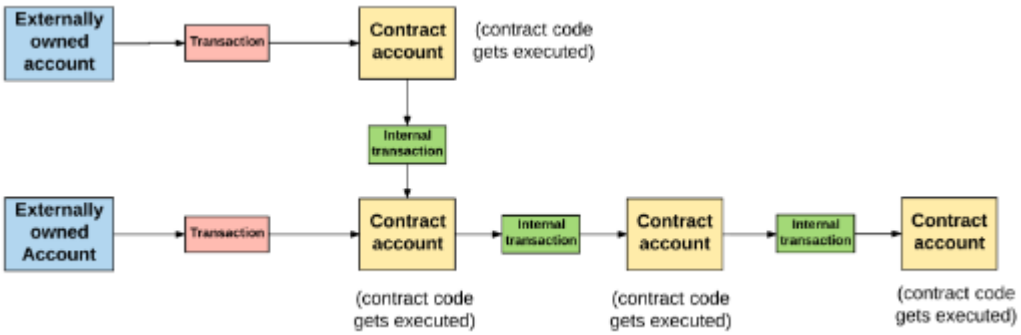


Рисунок 1.13 – Виконання контрактів

Смарт-контракти — це унікальна нова функція, представлена Ethereum, яка стала каталізатором для розробки незліченної кількості нових децентралізованих програм. Як і їхні фізичні аналоги, смарт-контракти можна розуміти як обов’язкові угоди між двома сторонами. Він представлений комп’ютерним кодом, який працює точно так, як це передбачено в блокчейні Ethereum. Після розгортання він працює автоматично і не піддається цензурі чи зміні. Це полегшує транзакції або обмін грошима, даними, вмістом або чимось цінним. Розумні контракти є самодостатніми, що означає, що їхній код розроблений для виконання певних дій, коли виконуються певні умови.

Технологія Blockchain змінює парадигму та підходи до розробки програм. Традиційні підходи зосереджені на розробці програмного

забезпечення. У них протоколи взаємодії відіграють невелику роль і є стандартними [23]. В системах, заснованих на блокчейні, основними є протоколи і методи взаємодії. За допомогою криптографічних токенів можна монетизувати протоколи, і компанії, які будують свої продукти на їх основі, можуть отримати вигоду. Наприклад, Ethereum є прикладом протоколу, який сам по собі не має цінності, але є чудовою основою для створення децентралізованих додатків [14].

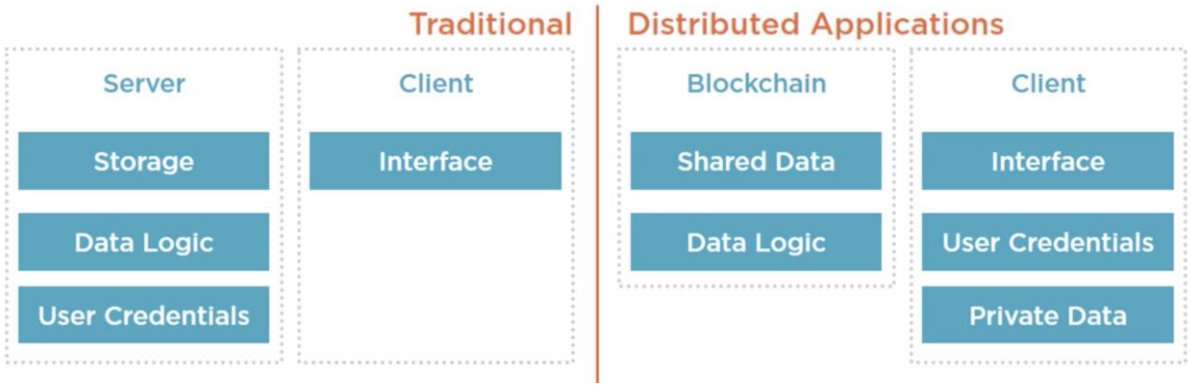


Рисунок 1.14 - Порівняння децентралізованої та традиційної архітектури

Враховуючи думку про важливість протоколів у створені децентралізованого додатку, методи проєктування можна розділити на три категорії [5]:

– децентралізовані застосунки, що мають власну реалізацію блокчейна та протоколів;

– децентралізовані застосунки, що використовують програми першого типу для створення на їхній основі власних протоколів та криптотокенів;

– децентралізовані застосунки, що використовують програми другого типу та побудовані на них протоколи для вирішення власних задач;

Прикладом програм першого типу можуть бути Біткойн - перша успішна реалізація децентралізованої системи. Іншим хорошим прикладом є Ethereum. Він почав своє існування, коли існував уже Біткойн. Проте його автори були

вимушені створювати власну реалізацію блокчейна та протоколу. При розробці власного протоколу чи блокчейн рішення, необхідно звертати увагу на спосіб децентралізованого консенсусу [17], іншими словами що буде стимулом для учасники мережі. Популярними є два підходи:

- доказ роботою;
- доказ статусом.

Доказ роботою залежить від можливості учасника вирішувати складну обчислювальну задачу. На це впливає швидкість та кількість перевірених та здобутих блоків, винагорода для такого учасника буде більшою за вклад у роботу мережі.

Доказ статусом напряму залежить від кількості криптотокенів, якими володіє учасник. Від цього також залежить його ефективність перевірки та майнінгу блоків. Цей підхід має кілька переваг перед доказом роботи:

- зменшення вірогідності атаки “51%”;
- витрачається менше енергії на перевірку транзакцій.

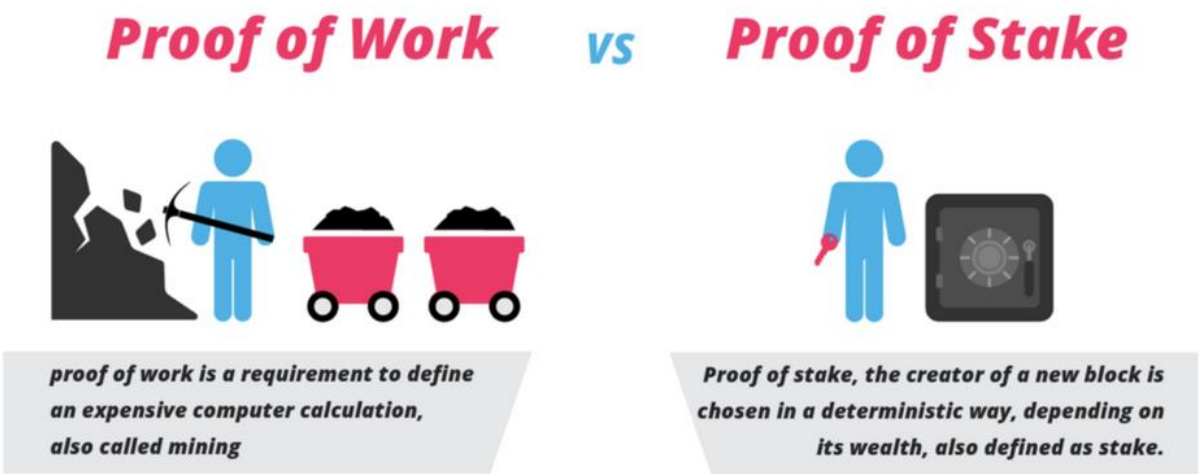


Рисунок 1.15 - Різниця між доказом статусу та доказом роботою

Важливою особливістю децентралізованих програм є відкритість їх програмного коду. Цей фактор є дуже важливим і безпосередньо впливає на довіру користувачів і учасників мережі до децентралізованої програми.

Підхід у розробці децентралізованих програм, що використовують програмні застосунки другого типу та відповідні протоколи для вирішення

своїх завдань, має такі переваги як високу швидкість децентралізованої розробки продукту та нижчий поріг входу для учасників. Однак це обмеження з точки зору його здатності впливати на алгоритм згоди або спосіб підписання транзакцій. Для більшості програм ці завдання не є проблемою, тому другий і третій підходи є оптимальним вибором для продовження роботи над власним застосунком.

1.4 Постановка задачі

Принцип децентралізації як базова основа технології блокчейн – це актуальна задача, вирішення якої допоможе створення ефективних і надійних програмних застосунків, але використання цієї можливості напряму залежить від способу розробки децентралізованих додатків

Метою дослідження є підвищення стійкості децентралізованих програм за рахунок використання надійніших алгоритмів обміну крипто-токенами.

Для досягнення мети в роботі поставлені наступні завдання:

- провести аналіз методів та засобів створення децентралізованих програм;
- удосконалити модель функціонування смарт-контрактів для покращення стабільності роботи;
- вдосконалити метод розподілу крипто-токенів у децентралізованих програмах на платформі Ethereum;
- розробити алгоритм та програмне забезпечення для платформи Ethereum, яке реалізує розроблений метод.
- дослідити ефективність розробленого алгоритму та засобів його реалізації.

2 МОДЕЛЮВАННЯ BLOCKCHAIN

2.1 Криптографічна модель в основі технології Blockchain

Блокчейн зазвичай представляють у вигляді журналу криптовалюти. Кожна сторінка містить інформацію про те, як були витрачені або зароблені кошти користувачів. Витрата або заробіток криптовалюти — це, по суті, передача права власності з одного облікового запису на інший. У рамках аналізу математичної сутності систем блокчейн буде корисно представити блокчейн біткоін по-іншому.

Нехай блокчейн буде системою, яка може змінювати свій стан. Така система може складатися з двох компонентів: стану системи та функції, яка її модифікує. Стан системи – це право власності на всі криптовалюти в системі. Системна функція приймає стан системи та транзакцію як аргументи. В результаті функціонування отримано новий стан системи, в якому відповідно зміняться права власності на криптовалюту.

У ній під станом можна розуміти таблицю балансів користувачів. Тоді транзакція - це запит на передання X токенів від користувача A до користувача B. Відповідно функцію стану можна визначити, як зменшення балансу користувача A на X токенів, та, відповідно збільшення балансу користувача B на X токенів. Але коли коштів у користувача A менше, а ніж X токенів, тоді виникне помилка виконання.

Формально модель можна представити таким чином [18]:

$$APPLY(S, TX) -> S' \qquad (2.1)$$

ERROR, у випадку помилки

У блокчейн системі, під станомбудемо розуміти колекцію із всіх токенів, які фактично є залишками після виконання транзакцій, що були добуті та ще не використані. У них є власники та їх номінал. Власник - представляє собою

20-байтову адресу, в якості якої виступає публічний ключ. Транзакції представляють собою блоки з одним або більше “входів”, що містять посилання на попередні залишки. Крім цього, кожна транзакція містить цифровий підпис, створений за допомогою приватного ключа, що асоціюється із користувачем (його адресою в системі). Також кожна транзакція містить один або більше виходів. Вони містить нові залишки від транзакцій, які в свою чергу будуть частиною нового стану системи.

Цифрові підписи - грають важливу роль в системі блокчейн. Вони дають можливість підтвердити право власності на токени і, розпоряджатися ними. Як згадувалося вище, для підпису потрібні публічний та приватні ключі. Володіння обома ключами дозволяє підтвердити власність на акаунт в системі. У біткоін подібних системах використовуються алгоритм підпису даних ECDSA (Elliptic Curve Digital Signature Algorithm) [13]. Успіх запропонованого підходу зумовлений тим, що вын досягаэ таких же показників стійкості, які є у поліноміальних криптосистемах та числових, але при цьому розмір ключа має набагато менший.

В Україні діє стандарт ДСТУ 4145-2002 (“Інформаційні технології та криптографічний захист інформації. Цифровий підпис, який ґрунтується на еліптичних кривих. Формування та перевірка”) [19]. Він пов'язаний із застосуванням матапарату еліптичних кривих над кінцевими полями.

Еліптична крива - це просто множина точок, що описується рівнянням, яке задається у канонічній формі Вейерштрасе [23]:

$$y^2 = x^3 + ax^2 + bx + c \qquad (2.2)$$

У формулі - права частина розглядається, як многочлен третьої степені. Коефіцієнти *a, b, c* - дійсні числа. Тому многочлен третьої степені *f(x)* буде мати мінімум один дійсний корінь. Він може бути розкладений:

$$f(x) = (x - \alpha)(x^2 + \beta x + \gamma) \qquad (2.3)$$

На рисунку 2.1 наведено еліптичну криву, яка має лише один корінь, тоді $y = 0$, відповідно $x = \alpha$:

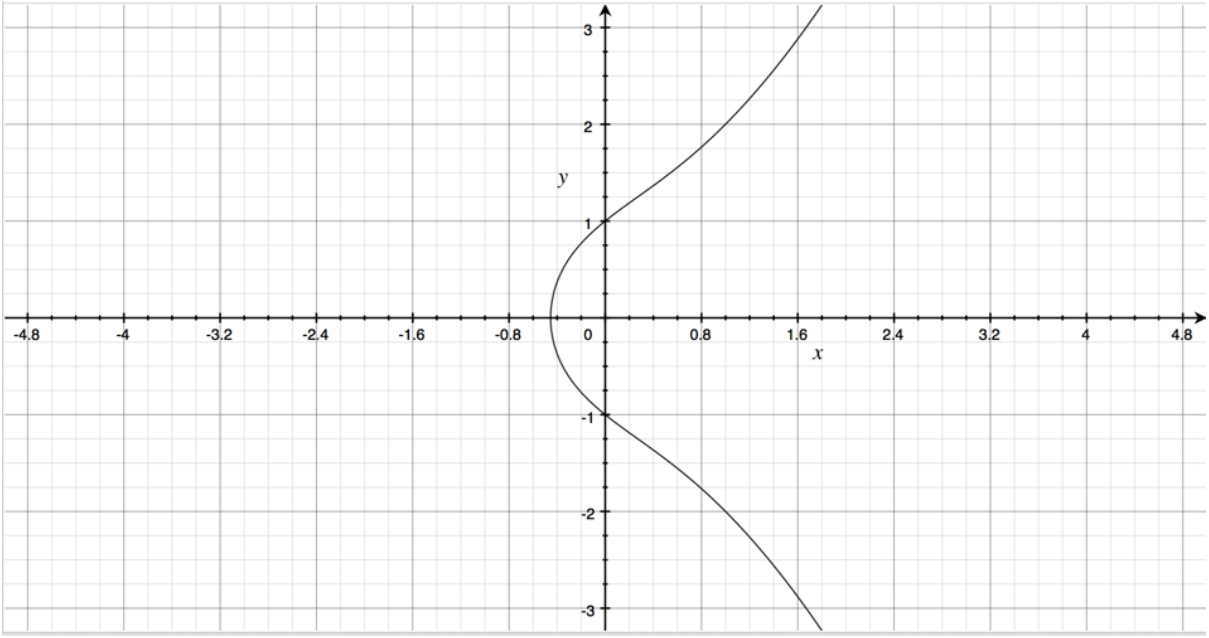


Рисунок 2.1 - Еліптична крива, що має один корінь

Якщо розв’язок має три кореня, то еліптична крива буде подібна до рисунка 2.2:

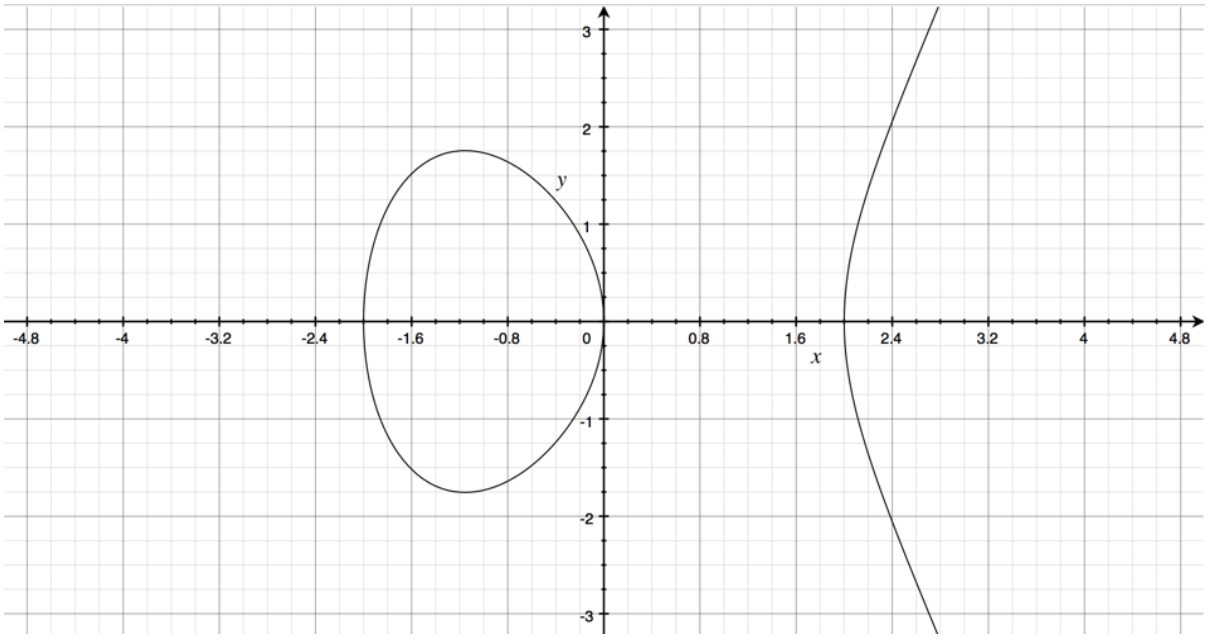


Рисунок 2.2 - Еліптична крива, що має 3 кореня

Еліптичні криві застосовуються для створення групи із значень її точок. Для будь-яких точок P і Q із групи сумою $P+Q$ буде існувати третя точка, яку утворює дотична $g = PQ$ з еліптичною кривою, що симетрично відображена відносно осі OX:

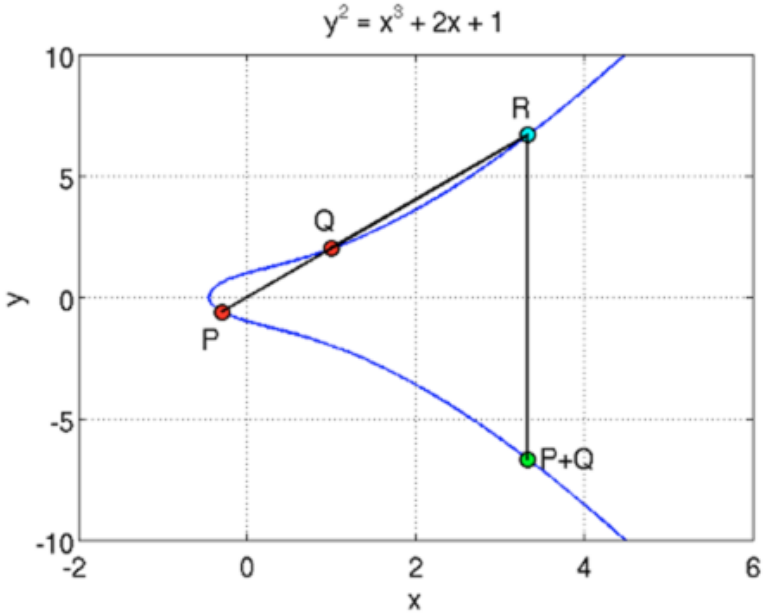


Рисунок 2.3 - Геометричний зміст

Під точкою сингулярності – розуміють точку, у якій похідна функції кривої рівна нулю або не існує. Тому для криптосистем ECDSA застосовують еліптичні криві для яких не існує точок сингулярності, інакше кажучи всі точки мають визначені дотичні у кожній точці. Їх називають несингулярними або гладкими.

Використовують еліптичні криві над кінцевими полями тобто набір точок, що належать кінцевому полю.

Наприклад, Bitcoin та Ethereum використовують криву $y^2=x^3+7$

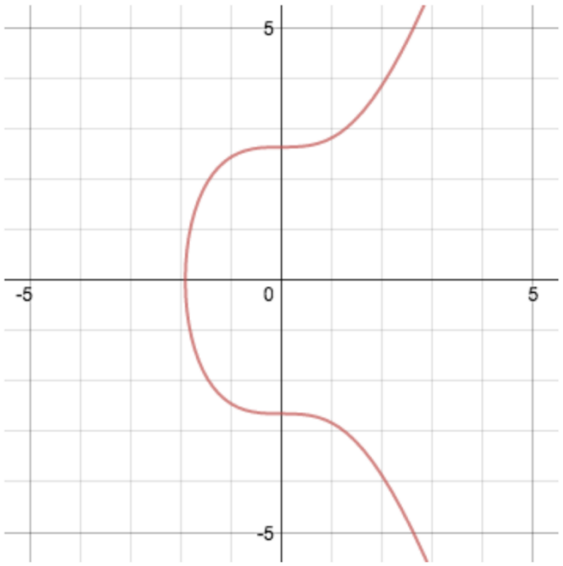


Рисунок 2.4 – Еліптична крива, яка використовується в Bitcoin та Ethereum

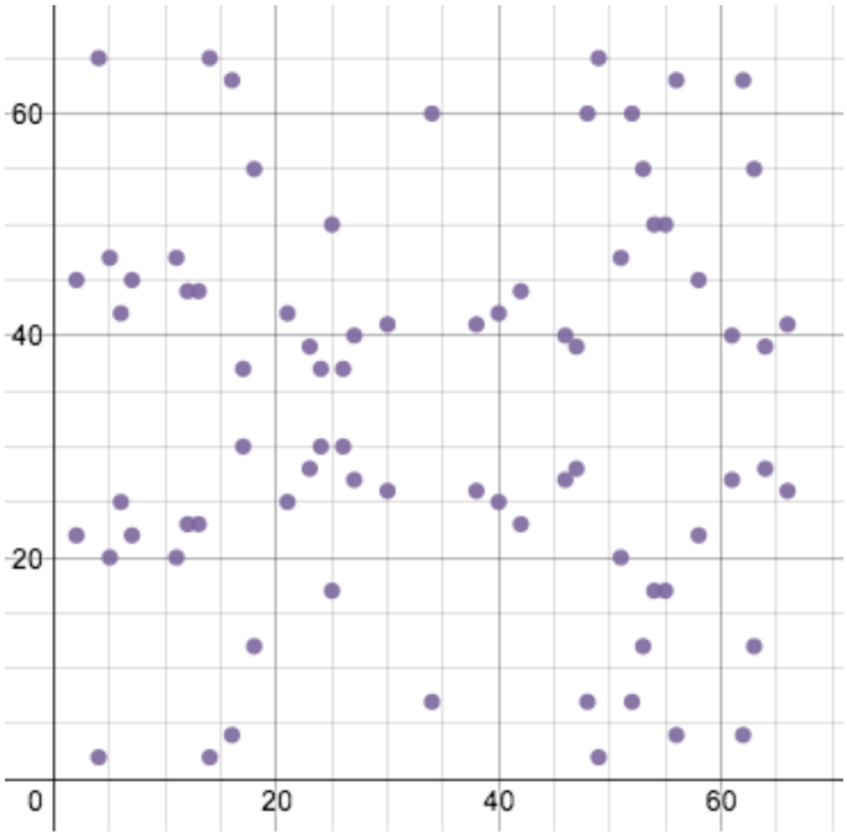


Рисунок 2.5 - Вигляд еліптичної кривої на кінцевому полі по модулю 67

Зміни у вигляді кривої викликані тим, що значення кривої по осі абсцис та ординат - це числа із діапазону [0; 66]. Навіть не дивлячись на ці зміни, крива зберегла свою горизонтальну симетрію. Для виконання визначених

операцій множення та додавання, потрібно враховувати, що результат ї не повинен виходити за межі діапазону кінцевого поля. Досягнувши порогу заданого модулем 67, алгоритм продовжить роботу, але повернеться на початок графіка, зберігши кутовий коефіцієнт, але уже із деяким зсувом.

Головна особливість еліптичної кривої полягає в тому, що її точки можна за особливим правилом множити на цілі позитивні числа. За такого множення[1]. В результаті точка переміщається певним чином.

Якщо G – точка на кривій, а k – число, то для нової точки $k \cdot G$ є три варіанти розташування:

– $k \cdot G = G$, тобто множення на k нічого не дає. Наприклад, множення на одиницю завжди залишає точку на місці;

– $k \cdot G$ не збігається з G, але при цьому все одно лежить на кривій. Тобто в результаті множення на q точка якось ковзає вздовж кривої;

– точки $k \cdot G$ не існує. Насправді це означає, що у формулах для підрахунку її координат зустрілося розподіл на 0.

Розглянемо алгоритм генерації ключів KeyGen. Секретний ключ sk вибирається випадковим чином від 1 до $n-1$. Точка $pk = sk \cdot G$ береться як відкритий ключ pk.

Тут вступає в дію ще одна ключова характеристика операції множення точки на число. Нехай задано точки G і $d \cdot G$, де d невідоме. Тоді виявляється, що пошук d не можливий, він є обчислювально нерозв'язною задачею. Цей факт має назву проблеми дискретного логарифмування еліптичних кривих.

З цього випливає, що, знаючи базову точку G і відкритий ключ pk, ми не зможемо знайти відповідний секретний ключ sk. У той же час, оскільки порядок точки G дорівнює n, то вона має лише $n-1$ можливих позицій на кривій, і це означає, що існує лише $n-1$ можливих значень для відкритого ключа.

На виході алгоритм видає пару sk, pk, де sk — число, pk — точка, тобто пара чисел.

Розглянемо алгоритм створення підпису Sig. Він приймає на вхід закритий ключ *sk* і число *m* — яке є хешем повідомлення, що підписується. На виході в якості підпису дається два числа*: (r, s)=Sig(sk, m*). Це пов'язано з тим, що всередині самого алгоритму є вибір додаткового випадкового числа *k*, яке впливає на тип підпису. Цей випадковий параметр потрібен для того, щоб гарантувати секретність ключа *sk*. Адже, якби підпис повністю визначався тільки вхідними параметрами *sk* і *m*, то секретний ключ *sk* обраховувся при наявності лише двох залишених разом з ним підписів.

Наявність випадкового параметра призводить до того, що цей алгоритм може видати різні результати для однакових вхідних значень. Таким чином, для того щоб підпис був перевірений, деяка інформація про параметр *k* повинна бути спільною. Перший елемент підпису *r* містить саме цю інформацію. Число *r* однозначно визначається числом *k*; При цьому всі три параметри *k, sk* і *m* вже беруть участь у розрахунку числа *s*.

Алгоритм перевірки підпису Ver приймає як вхідні дані відкритий ключ *pk*, хеш повідомлення *m* і підпис *(r, s)*.

На основі підпису та хешу цей алгоритм знаходить два числа u_1 та u_2. Далі ми знаходимо дві точки на еліптичній кривій: $u_1 \cdot G$ і $u_2 \cdot pk$.

Якщо підпис *s* був фактично обчислений за допомогою ключа *sk*, який відповідає ключу *pk*, і повідомлення дійсно не зазнало змін з моменту створення підпису, то точки $u_1 \cdot G$ і $u_2 \cdot pk$ будуть у деякому спеціальному положенні відносно кожної інший. Якщо розташування точок відносно одна одної справді таке, як має бути, тоді підпис вважається правильним, і Ver повертає 1 (TRUE). В іншому випадку повертається 0 (FALSE).

Важлива математична модель, яка лежить в основі розробки децентралізованих програм, належить до доказу твердження, що на практиці неможливо одержати паралельно ланцюжок із наору блоків, через який атакуючий міг би відмінити свої транзакції та ввесьти в оману власника акаунта, для якого була здійсненна транзакція. В теорії, є можливість одержати окремо довший ланцюжок із блоків і цим заставити інших учасників

блокчейну підтвердити, що якраз цей ланцюжок - є справжнім і має бути використаним замість правильного та коректного. Подібний вектор атаки на на блокчейн називають “атака 51%”.

Атака 51% — це вразливість блокчейну PoW, яка дозволяє зловмиснику контролювати підтвердження транзакцій і створення блоків [28].

З 51% доступної потужності зловмисники можуть:

– заборонити іншим майнерам (валідаторам) знаходити блок **(**selfish mining**)**;

– провестиподвійне витрачання монет для крадіжки у постачальників послуг, бірж або обмінників **(**double spend**)**;

– взлом основного блокчейну, розділяючи мережу на два конкуруючих ланцюги;

– не допускати підтвердження транзакцій;

– під час атаки вони збирають усі винагороди за одержання блоку та комісії за транзакції.

Атака є більш серйозною, якщо зловмисники контролюють значно більше ніж 51% мережі. Потім вони можуть здійснювати:

– крадіжки з будь-яких контрактів типу deposit-challenge-verify, і канали станів - Lightning Network, якщо зловмисники були їх учасниками;

– маніпулювати складністю мережі;

– красти монети, шляхом відкату старих блоків і повторного отримання винагороди;

– видалити контракти або історію транзакцій (шляхом відкату старих блоків і редагування списку транзакцій).

Шкідливий майнінговий пул може найняти додаткові ресурси та здійснити атаку на обрану криптовалюту. На основі даних [30] автори дослідження «Вивчення типів атак на блокчейн» склали таблицю з шести криптовалют і вказали вартість атаки за годину.

Система	CAP	Алгоритм	Hash Rate	Цена
Bitcoin	112.7M	SHA-256	35,604 PH/s	486K
ETH	49.5M	Ethash	222 TH/s	347K
B.Cash	14.9M	SHA-256	5,023 PH/s	68K
LTC	5.7M	Scrypt	327 TH/s	60K
Dash	2.1M	X11	2 PH/s	15K
Monero	2.3M	CryptoNight	365 MH/s	17K

Рисунок 2.6 – Вартість атаки на годину

Як видно з рисунка 2.6 для атаки на біткоіни потрібно витрачати $486 тис. на годину.

Погоню між чесними учасниками мережі та атакуючими змоделюємо за допомогою біноміального випадкового блукання. У цьому випадку, успішною подією буде продовження валідного ланцюжка блоків на один. Не успішна подія це створення нового блоку у ланцюжку злочинця. Якщо взяти до уяви, що атакуючий має необмежений кредит та починає перегони із деякого дефіциту але маючи нескінченну кількість спроб, щоб відігратись. Тоді імовірність того, що атакуючий випередить чесний ланцюжок блоків, можна знайти за допомогою формули:

$$q_z = \begin{cases} 1 & if\ p \le q \\ (q/p)^z & if\ p > q \end{cases} \qquad (2.4)$$

де p - ймовірність появи блока в коректному ланцюжку, q - ймовірність того, що блок створить атакуючий, q_z - ймовірність, що зловмисник дожене різницю в z блоків. Якщо $p > q$, тоді ймовірність того, що атакуючий наздожене коректний ланцюжок зменшується стрімко експоненціально (рис. 2.7).

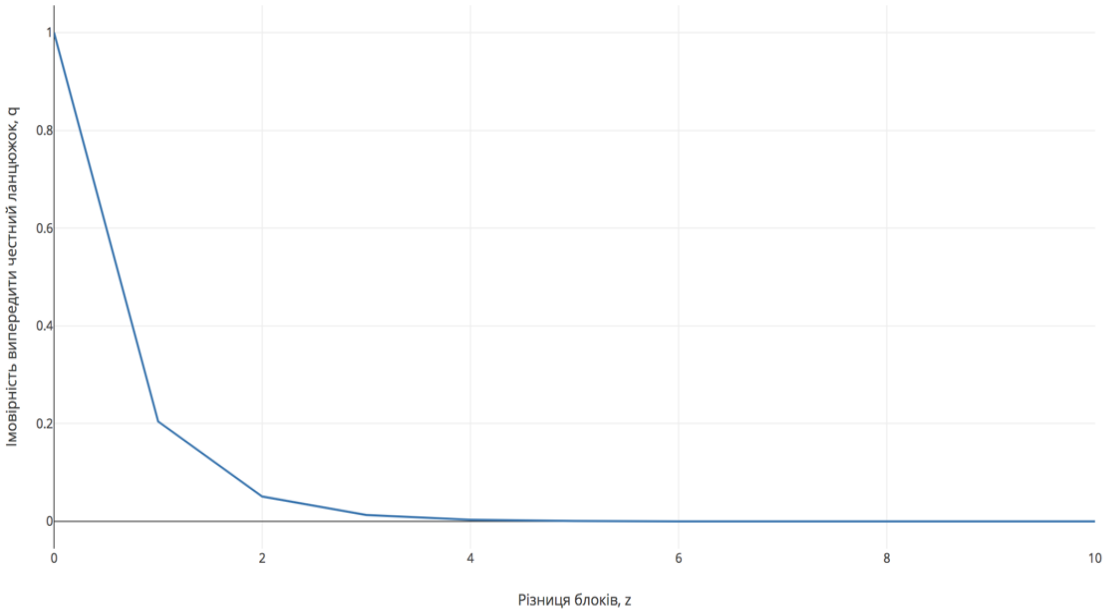


Рисунок 2.7 - Графік ймовірності, що атакуючий може наздогнати різницю у *z* блоків, при *z = 0..10, q = 0.1*

Без вдалого ривка на самому початку гонки, шанси атакуючого зменшуються експоненціально та стають мізерними.

2.2 Удосконалена модель функціонування смарт-контрактів

У децентралізованих програмах [21] з використанням технології блокчейн важливими є операції, що відображують дії над крипто-токенами. По суті, крипто-токени в децентралізованій програмній мережі (внутрішні активи учасників).

Взаємодія між учасниками відбувається за допомогою транзакцій. Учасниками транзакцій в мережі можуть виступати розумні контракти і люди. Смарт-контракти - це також повноцінні учасники мережі на основі блокчейну. Вони можуть приймати та передавати крипто-токени, розподіляти їх між іншими учасниками мережі.

Формально, процес традиційного здійснення транзакцій можна представити моделлю взаємодії в мережі учасника-людини і розумного контракту (рис.2.11).

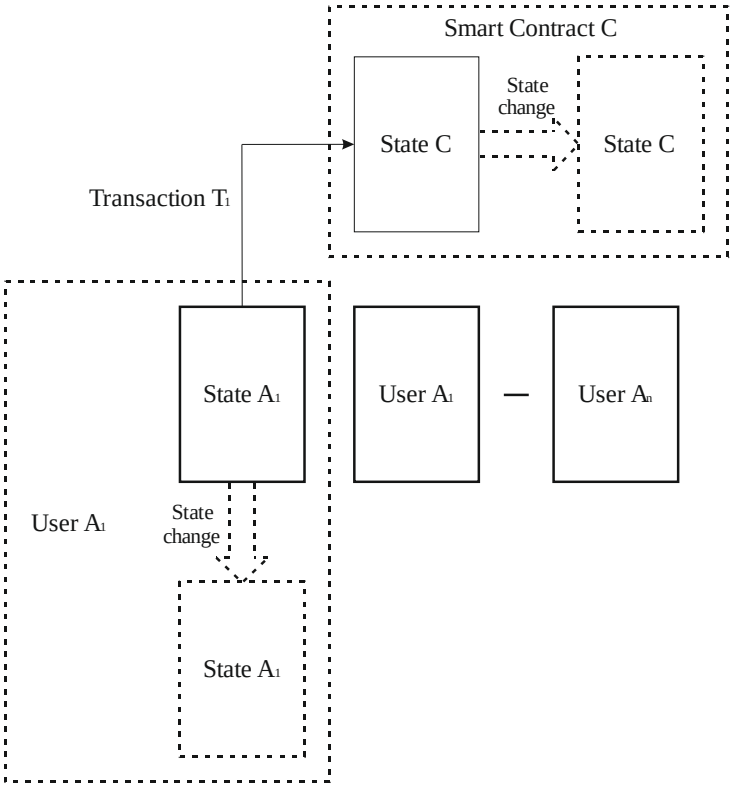


Рисунок 2.11 - Реакція контракту *C* на транзакцію T_1 учасником A_1

Згідно з рис.2.11, в мережі *M* існує смарт-контракт *C*, з початковим станом *C*. Також в мережі *M* існують інші учасники $A_1, A_2,\ldots, A_n$, з початковими станами $A_1, A_2,\ldots, A_n$.

Учасники виконують транзакції $T_1, T_2,\ldots, T_n$, що змінюють стан смарт-контракту *C* на *C*', *C*'',…, $C^{(')n}$, а стани учасників відповідно на $A_1', A_2',\ldots,A_n'$.

Коли в мережі *M* насупає подія P_1, смарт-контракт *C* виконує транзакцію CT_1 та змінює свій стан на $C^{(')(n+1)}$, в якій розподілені крипто-токени повертаються назад до учасників, при цьому змінюючи стани машин-автоматів A_i на A_1''', A_2''',…, A_n''' .

В ідеальній ситуації транзакції виконуються без помилок і учасники отримують свої крипто-токени.

Різноманітні події *P* в мережі можуть відбуватись лише один раз або наступати періодично.

Учасником може бути смарт-контракт або людина. В системі немає можливості зрозуміти хто керує контрактом, тому, з імовірністю q - учасник є іншим смарт-контрактом, із імовірністю p учасник події є людиною.

Розглянемо ту ситуацію, коли смарт-контракт C переходить в стан $C^{(')(n+1)}$.

При умові що учасником є людина, то у неї немає способу зупинити транзакцію в мережі M.

Транзакції $T_1, T_2,\ldots, T_n$ та CT_n будуть виконанні тільки за умови відсутності помилок роботи смарт-контракту.

Але помилки можуть бути на етапі відправки транзакції смарт-контраком C або на етапі її прийому учасниками $A_1, A_2,\ldots, A_n$. Помилки в програмному коді смарт-контрактів можуть зупинити виконання транзакції та скасувати її результат.Але при цьому вже може бути знята плата за виконання транзакції. Якщо смарт-контракт C не містисть програмних помилок і може відправляти транзакції учасникам, тоді збої можуть статися на стадії отримання транзакцій. Причиною помилок можуть бути атакуючі дії відносно ресурсів мережі M.

Особливо небезпечним розглянутий випадок є у мережах, де подія P_n наступає лише один раз. Це може викликати зависання стану смарт-контракту та проблем під час одержання права на свої крипто-токени від контракту C.

Потрібно запропонувати спосіб, який зробив би стабільним виконання розумного контракту C під час настання події P_1 та обов'язкове виконання транзакції CT_n, в ситуації коли в мережі існує розумний контракт, який має помилки в своєму коді або є зловмисним.

Під стабільним виконанням розумного контракту C будемо вважати здійснення трьох умов:

– можливість повернути свої крипто-токени для учасника, який викликав помилку;

– можливість решті учасників повернути свої крипто-токени;

– продовжити роботи контракту *C* для інших учасників блокчейну при настанні подій $P_2, P_3,\ldots, P_n$.

Для оцінки можливої нестабільності та загрози можна використати методи теорії імовірності та матстатистики. Для цього потрібно провести оцінювання двох ситуацій:

– розрахувати імовірність випадкової взаємодія із іншими нестабільними розумними контрактами, що могли би вплинути на стабільність роботи контракту *C*;

– розрахувати імовірність, нестабільної роботи системи, якщо існує в зловмисник, знайомий з вектором атаки.

Щоб провести оцінювання імовірності випадкової взаємодії із іншими старт-контрактами, що мають проблеми і недопрацювання в прийомі транзакцій, використаємо статистику сервісу Etherscan про користувачів Etherium платформи [22]. За його інформацією, в системі зареєстровано: 44 080 162 користувачі, з яких 46 487 - це розумні контракти.

Нехай подія *B* – учасник транзакції є розумним контрактом. Тоді подія *B*’ - учасник транзакції є людиною. Події *B* і *B*’ утворюють повну множину можливих варіантів:

$$P(B) = \frac{N(B)}{N} = \frac{46\,487}{44\,080\,162} = 0.001054601; \quad (2.5)$$

$$P(B') = 1 - P(B) = 0.998945399 \quad (2.6)$$

Тоді вірогідність події *A’*, при якій хоча б один із *n* учасників транзакції буде розумний контракт розрахуємо по формулі:

$$P(A') = 1 - (P(A))^n \quad (2.7)$$

Так, для кількості учасників *n*=10, ймовірність *P(A)* буде дорівнювати:

$$P(A) = 0.010496102 \quad (2.8)$$

Нехай подія E - це присутність нестабільного розумного контракту, який може викликати нестабільності контракту C. Подія E визначається ймовірністю $P(E)$. Тоді можна вирахувати імовірність того, що в системі буде хоча б один нестабільний розумний контракт із проблемним кодом (подія D):

$$P(A) = P(E)P(A) \quad (2.9)$$

Графік розподілу ймовірностей подій A (червоний) і D (синій) представлено на рис.2.12

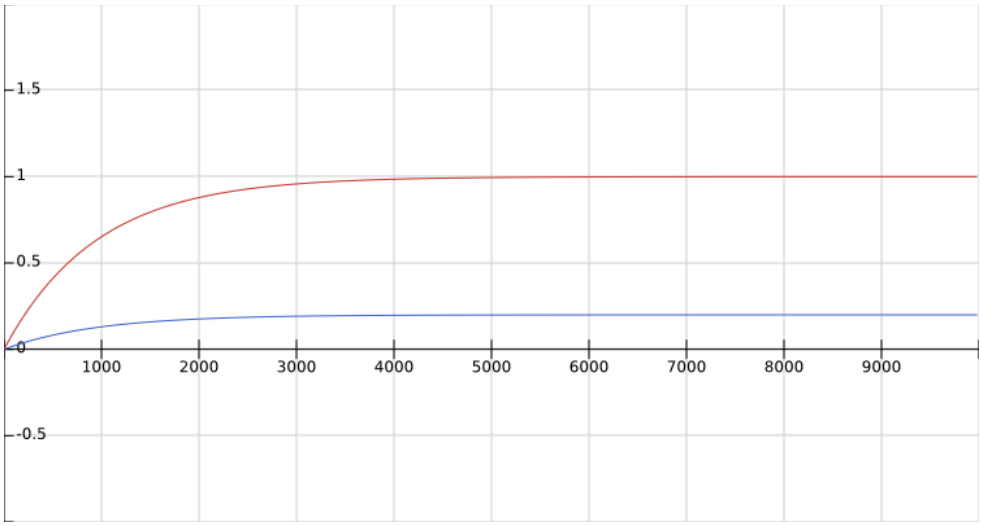


Рисунок 2. 12 - Графік імовірності подій при $P(E) = 0.2$

Як видно з рисунку 2.12, імовірності ростуть при збільшенні росту учасників у системі, тому імовірність випадкової помилки надто мала та залежить від присутності нестабільного розумного контракту $P(E)$. Для оцінки порядку і характеру ризику величину було взято $P(E)= 0.2$, при цьому ризик із зростанням кількості учасників буде прямувати до $P(E)$, як це видно з рисунку.

Розглянемо ситуацію, коли взаємодія є не випадковою і зумовлена намірами атакуючого. Тоді імовірність $P(D)$ буде наближеною до 1, а подія D існування хоча б одного нестабільного контракту як учасника із проблемним кодом буде вважатися достовірною. Єдиним стримуючим фактором, як описувалося раніше це є фінансово-економічний. Запуск власного контракту буде коштувати зловмиснику грошей. Якщо це буде економічно вигідним, вразливість може бути використана для організаціїя нестабільності у системі смарт-контракту C.

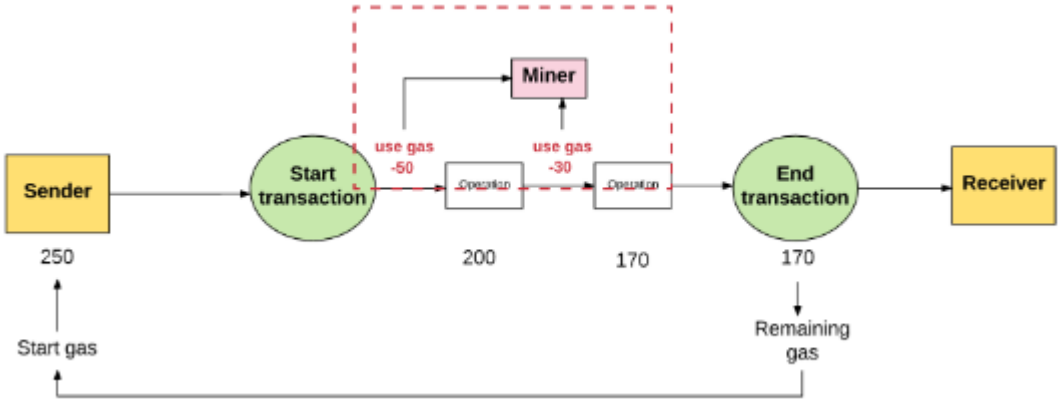


Рисунок 2. 13 – Процедура оплати за перевірку виконання транзакції

Як правило, чим вище вартість «газу», яку готовий заплатити відправник, тим більший платіж отримує майнер в результаті транзакції, і, крім того, тим більша ймовірність того, що майнер зробить свій вибір саме на її користь. Таким чином, майнери можуть вільно вибирати, які транзакції вони хочуть перевіряти, а які – ігнорувати. Часто майнери повідомляють відправникам, яку ціну їм слід встановити за «газ», щоб ті були готові виконувати транзакції.

В Ethereum ви самі встановлюєте розмір комісії за операцію.

Коли ви ставите будь-яку з транзакцій у чергу, ви вказуєте:

– адресу одержувача;

– сума ETH для переказу (може бути 0);

– скільки максимально «газу» ви готові витратити на виконання операції;

– ваша ціна на «газ».

Якщо ви встановите занадто низьку ціну, то угода може зависнути і довго не виконуватися.

Є ще одна небезпека зі розумними контрактами: може виявитися, що поданого «газу» не вистачить для його виконання. У цьому випадку майнер виконуватиме частину контракту до тих пір, поки буде достатньо газу для завершення операцій. За виконану роботу він отримає гроші, але угода не буде оформлена.

Важливим аспектом роботи Ethereum є те, що будь-яка операція, яка виконується мережею, також одночасно виконується кожним повним вузлом [25]. Однак усі етапи обчислення на віртуальній машині Ethereum надто дорогі. Таким чином, для вирішення простих завдань (наприклад, перевірки підписів, а також інших операцій, пов'язаних з криптовалютою), смарт-контракти Ethereum можуть цілком підійти, на відміну від тих випадків, коли потрібні більш складні завдання, наприклад виконувати машиннео навчання, яке можуть спричинити надмірне використання мережі. Введення оплати запобігає діям користувача, спрямованим на зайве навантаження на мережу.

Стандартний підхід реалізований на платфомі Ethereum працює у цьому випадку нестабільно. Він полягає у тому, щоб при наступанні події P_n перерозподіляти крипто-токени та передавати їх усі разом через транзакцію CT_n (рис.2.13). Якщо виникне помилка транзакція буде скасована і може статись так, що стан контракту C залишиться незмінним. Це призведе до проблем із перерозподілом крипто-токенів у мережі M.

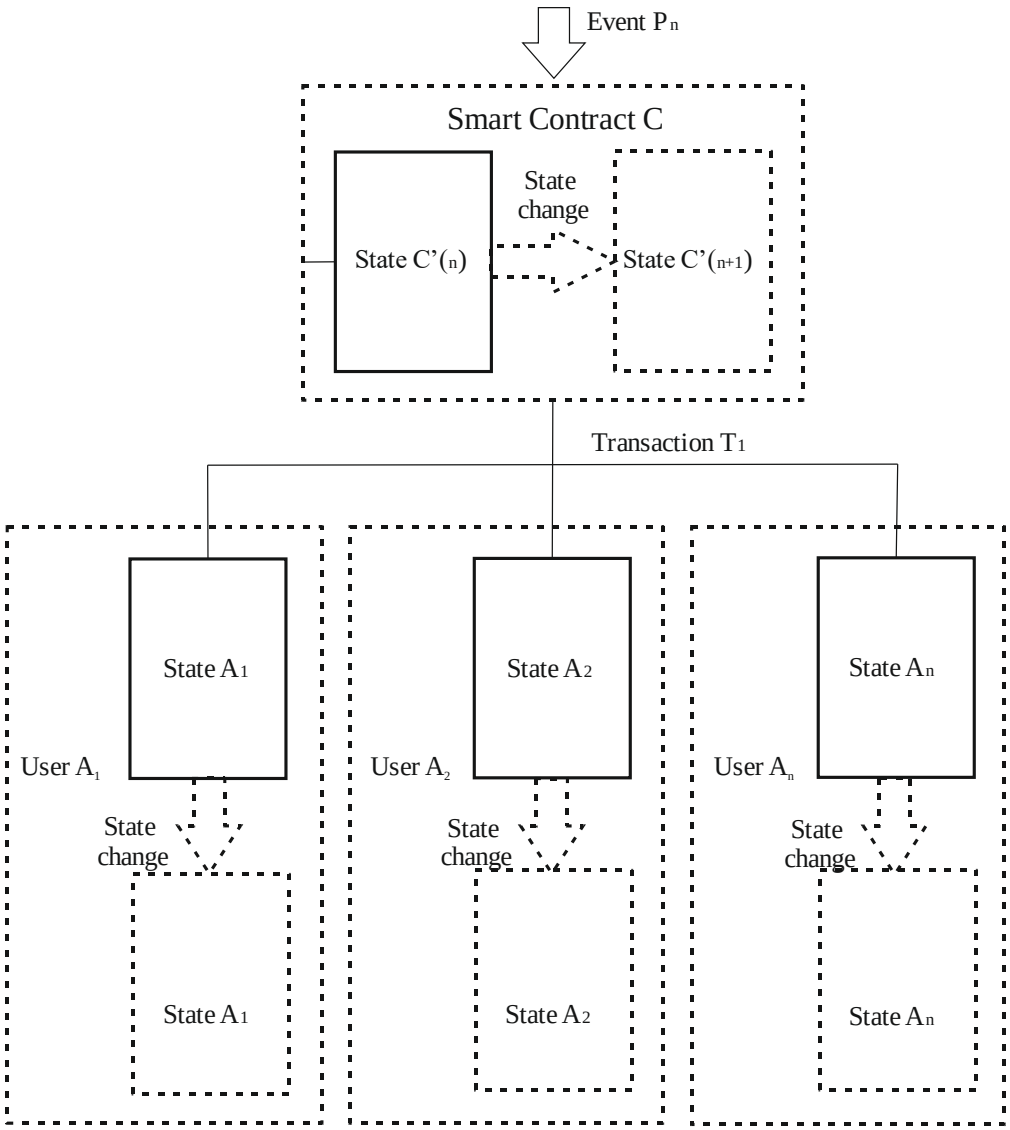


Рисунок 2.14 - Стандартний підхід функціонування смарт-контрактів

Альтернативний варіант, запропонований у роботі потребує керування розподілом крипто-токенів у ситуаціях, коли виникає помилка на етапівиконання транзакції CT_n і потрібно забезпечити можливість учасникам транзакції отримати свої крипто-токени [12].

Для вирішення задачі в роботі пропонується наступна модель (рис.2.14).

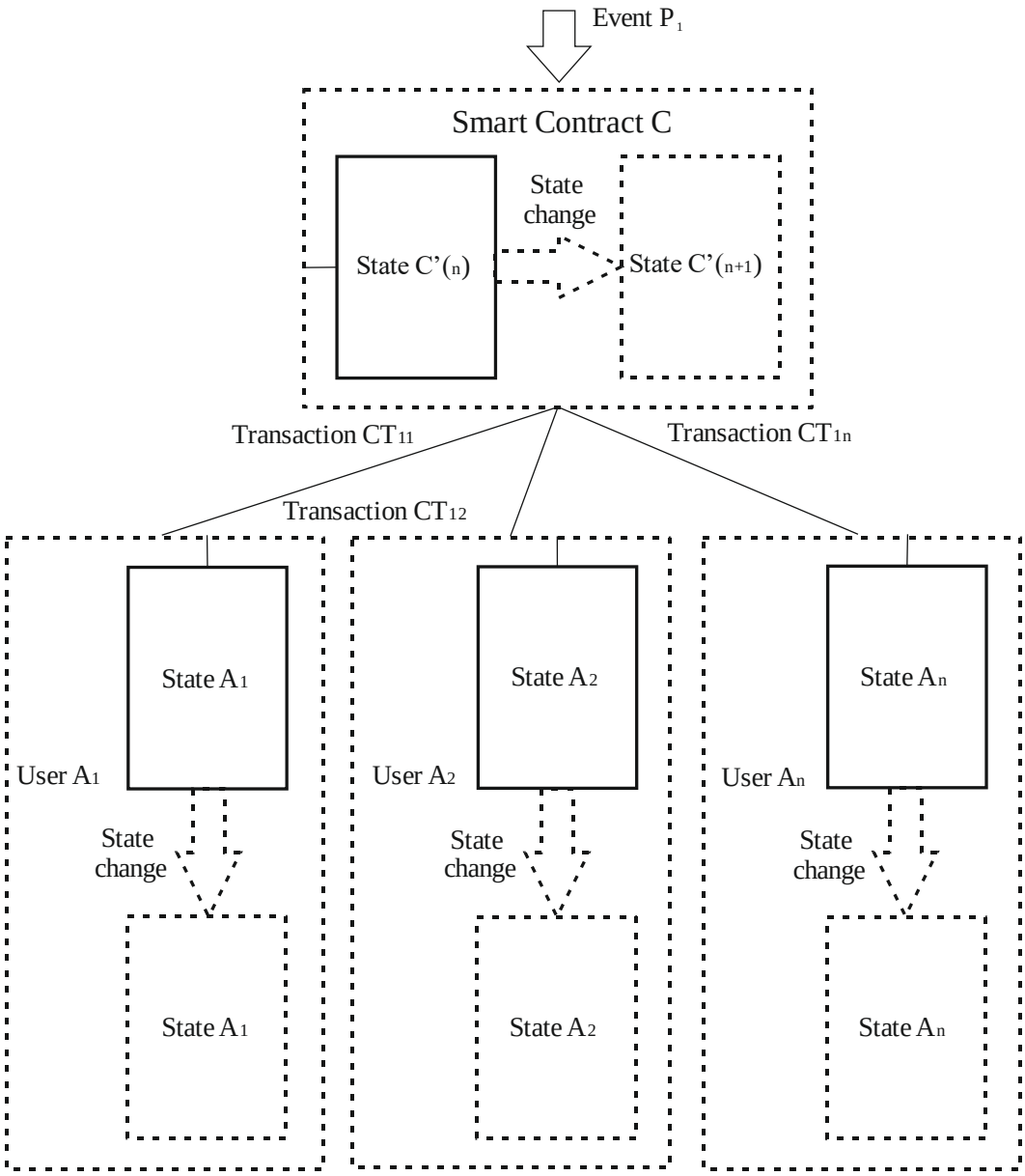


Рисунок 2.15 – Запропонований підхід до функціонування розумних контрактів

Нехай взаємодія відбувається між смарт-контрактом C та учасниками $A_1, A_2,\ldots, A_n$. Учасники виконують транзакції $T_1, T_2,\ldots, T_n$, після яких стан розумного контракту C змінюється на проміжні стани C', C'',…, $C^{(')n}$. Стани учасників відповідно змінюються на $A_1', A_2',\ldots, A_n'$. В той момент, коли в мережі M настає подія P_1, контракт C змінює свій стан на $C^{(')(n+1)}$. При цьому смарт-контракт C розподіляє крипто-токени всередині свого стану на відповідні рахунки учасників $A_1', A_2',\ldots, A_n'$. На цьому відбувається закунчення

ефекту події P_1. Учасники A_1', A_2',…, A_n' мають можливість перевірити стан своїх акаунтів в смарт-контракті C та при потребі виконати зняття коштів через транзакції CT_{11}, CT_{12},…, CT_{1n}. У випадку, якщо під час транзакції CT_1 виникне помилка, то учасник A_i зможе повторити транзакцію, але не може завадити іншим учасникам отримати свої крипто-токени.

Запропонована модель може використовуватися для розподіленого виведення крипто-токенів, та дозволяє уберегтись від реніше описаного вектору атак на розумні контракти, коли атакуючий намагається виконати транзакції T_i через власний смарт-контракт, який запрограмований на створення помилки для транзакції CT_i. Зловмисник не має можливості глобально впливати на роботу смарт-контракту C та на його стабільність для інших учасників системи блокчейн.

2.3 Висновки

Децентралізовані додатки на архітектурі блокчейн – на сьогодні найбільш популярні технології. Сьогодні у світі існує багато децентралізованих програм та пропозицій, але підходи та принципи децентралізованої розробки я на основі технології Blockchain постійно вдосконалюються. Цей процес цілком нормальний для нових технологій, оскільки їх адаптація потребує часу і часто вимагає зміни традиційних підходів.

Мережу блокчейн представлено системою, яка може змінювати свій стан. Така система може складатися з двох компонентів: стану системи та функції, яка її модифікує. Стан системи – це право власності на всі криптовалюти в системі. Системна функція приймає стан системи та транзакцію як аргументи. В результаті функціонування (виконання транзакцій) отримано новий стан системи, в якому відповідно зміняться права власності на криптовалюту. В розділі проаналізовано криптографічні основи, які лежать в основі Blockchain технології. Виявлено недоліки в існуючому

методі функціонування смарт-контрактів. Для усунення знайдених недоліків вдосконалену модель розподілу крипто-токенів при функціонуванні смарт-контрактів. Запропонована модель може використовуватися для розподіленого виведення крипто-токенів, та дозволяє уберегтись від вектору атак на розумні контракти, через власний смарт-контракт, який запрограмований на створення помилки при виконанні транзакції.

3 МЕТОД РОЗПОДІЛУ КРИПТО-ТОКЕНІВ ПРИ ПРОЄКТУВАННІ ДЕЦЕНТРАЛІЗОВАНИХ ДОДАТКІВ

3.1 Метод розробки децентралізованих додатків

Ethereum - це система стану транзакцій. Іншими словами, завдяки транзакціям, які відбуваються між різними обліковими записами, глобальний стан Ethereum змінюється або переміщується з одного стану в інший.

Простіше кажучи, транзакція - це криптографічно підписана інструкція, яка спочатку встановлюється зовнішнім обліковим записом, а потім замовляється та передається в блокчейн.

Всього існує два типи транзакцій: надсилання повідомлень і створення контракту (іншими словами, такі транзакції створюють нові смарт-контракти в мережі Ethereum).

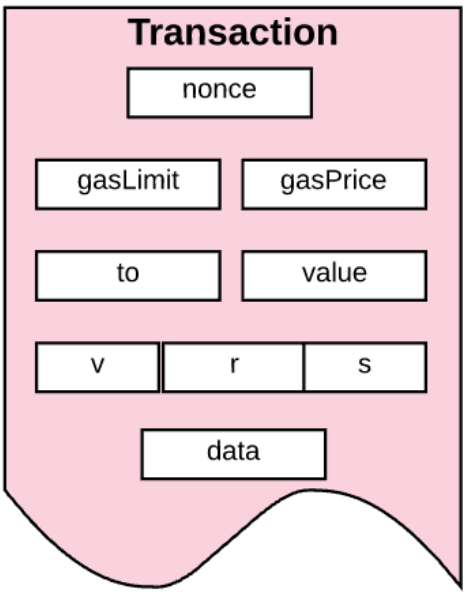


Рисунок 3.1 – Структура транзакції

Транзакції - як для виклику повідомлень, так і для створення розумних контрактів - ініціюються зовнішніми обліковими записами, а потім пересилаються в блокчейн. Іншими словами, транзакції - це своєрідний міст,

що з'єднує зовнішній світ і внутрішній стан платформи Ethereum.

Усі транзакції якимось чином згруповані в «блоки». Блокчейн містить кілька таких блоків, пов'язаних між собою.

Ці блоки складаються з:

– заголовок блоку;

– інформація про серію транзакцій, що входять до цього блоку;

– набір інших заголовків блоків для поточних оммерів (блок, батьком якого є батьківський елемент поточного блоку)

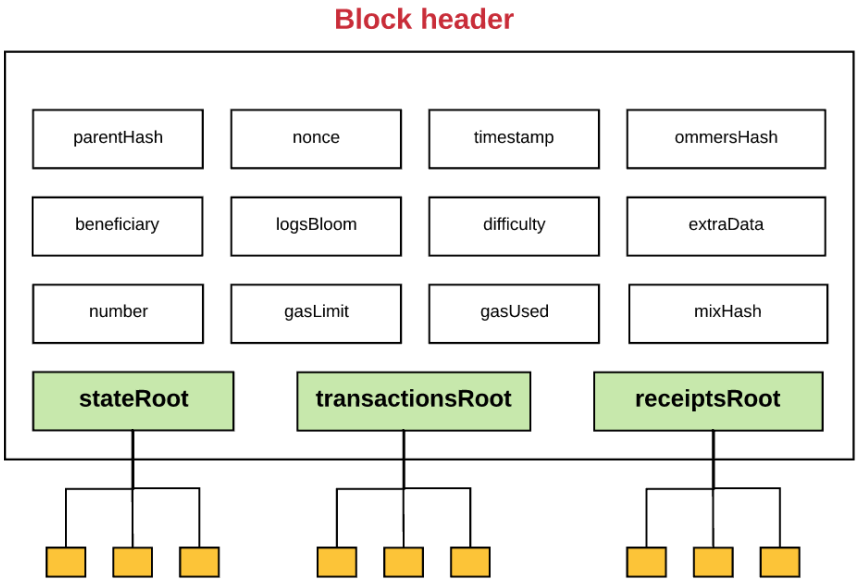


Рисунок 3.2 – Структура блоків

Щоб створити новий контракту, ми повинні спочатку оголосити адресу облікового запису, який буде створено, використовуючи спеціальну формулу. Після цього створюється новий обліковий запис. Щоб виконати данну операцію, необхідно виконати ряд дій:

– встановити значення nonce дорівнює нуль;

– встановити balance свого облікового запису рівним платі за транзакцію (у випадку, якщо відправник готовий надіслати деяку кількість ефіру в якості оплати за транзакцію);

– розрахувати суму платежу, яка надходить на баланс створеного рахунку з рахунку відправника;

– вказати, що сховище більше не використовується вами;

– встановити хеш-код контракту як хеш порожнього рядка.

Частина протоколу, яка обробляє транзакції в операційній системі Ethereum, називається віртуальною машиною Ethereum (EVM).

Крім того, EVM має всі особливості стекової архітектури. Стекова машина - це комп’ютер, який використовує алгоритм LIFO.

Розмір будь-якого елемента стека в EVM становить 256 біт, а максимальний розмір стека - 1024 біта.

EVM має певний обсяг пам'яті, який не є постійним. У ньому елементи зберігаються у вигляді масивів байтів з доступом до слів.

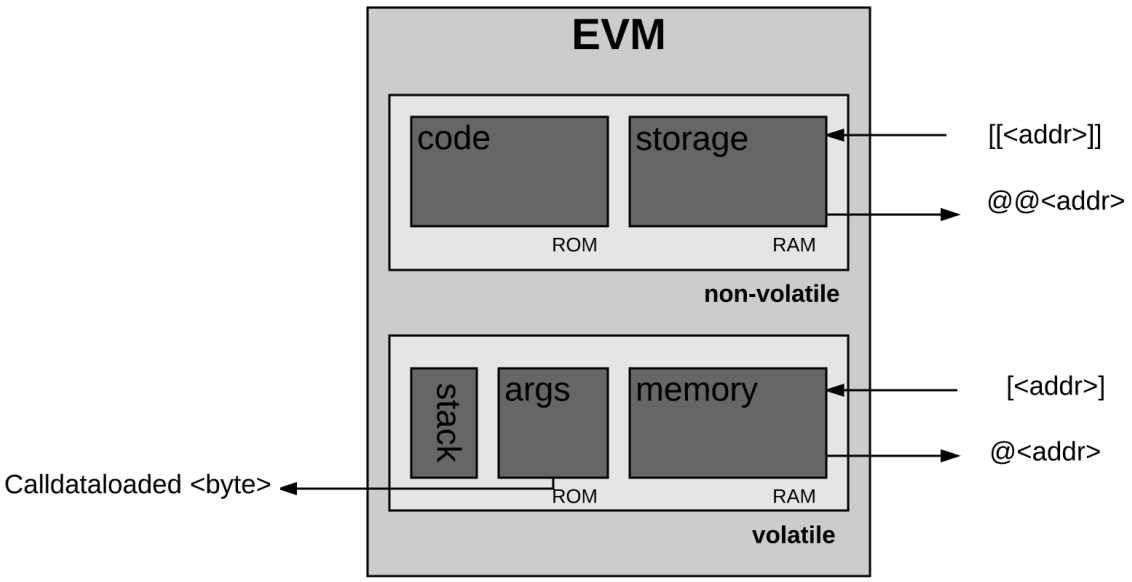


Рисунок 3.3 - Віртуальна машина Ethereum

Будь-яка транзакція має відповідати певним вимогам, щоб її виконання не було скасовано, а саме:

– транзакції мають відповідати вимогам RLP. RLP — це рекурсивний префікс довжини (від англ. Recursive Length Prefix), який є форматом даних, який використовується для кодування вкладених масивів двійкових даних;

– наявність дійсного підпису транзакції;

– дійсне одноразове значення. (nonce - це кількість транзакцій, відправлених з поточного рахунку). Щоб таке значення було дійсним, воно має збігатися зі значенням nonce для облікового запису відправника.

– ліміт «газу» для транзакції (рис 3.4).

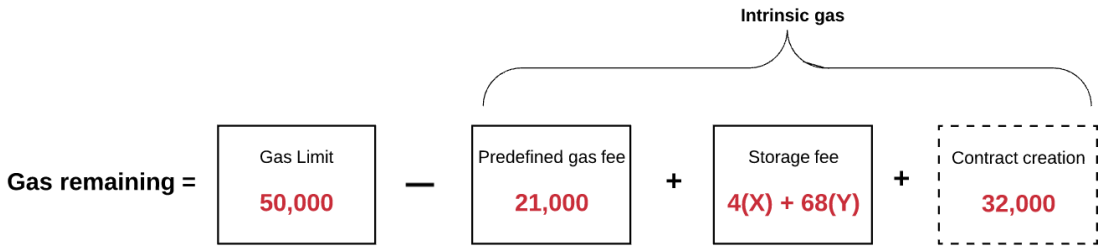


Рисунок 3.4 – Обрахунок вартості транзакції

Попередньо визначена вартість дорівнює 21 000 одиниць «газу», необхідного для завершення транзакції. Комісія за «газ», що використовується для надсилання даних транзакції (4 одиниці «газу» за кожен байт даних або код, що дорівнює нулю, і 68 за кожен ненульовий байт даних або ненульовий код). Додатково 32 000 одиниць «газу», якщо операція пов'язана з укладенням договору.

Згідно Вікіпедії, децентралізований додаток (decentralized application, DApp) — це комп’ютерний застосунок, що базується на технології блокчейн разом із механізмом розподіленого виконання необхідних інструкцій. Станом на 2022 рік для цього найчастіше використовується Ethereum з його механізмом смарт-контрактів.

На основі досвіду створення децентралізованих додатків, аналізу можливості, що дає технологія блокчейн, та запропонованих у другому розділі моделей, вдосконалено метод розробки децентралізованих додатків на основі технології блокчейн.

В реалізації запропонованого підходу можна виділити такі основні етапи (рис.3.5)

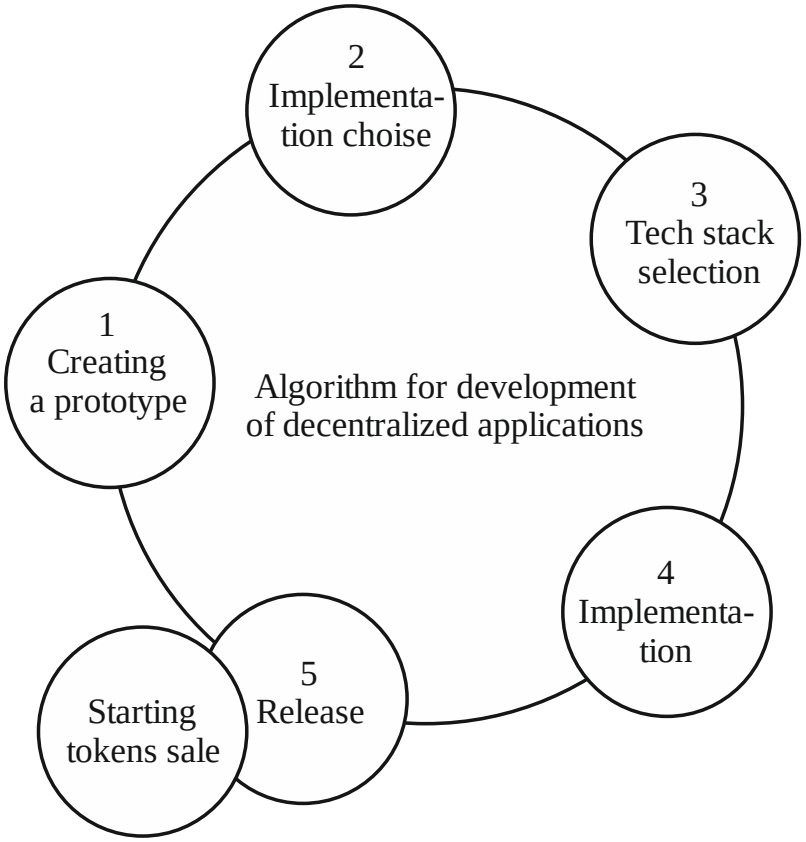


Рисунок 3.5 - Етапи методу розробки децентралізованої програми

Першим етапом є створення прототипу. Прототипування – може бути виконано як текстовий опис суті застосунку, який би максимально охоплював і уточнював ідею та функціонал на етапі продумування логіки програми. Формалізована ідея та концепція програми мають стати базовою інформацією для наступних кроків та головним чинником для прийняття необхідних рішень.

Вибір технології блокчейн. Оскільки розробка ведеться з використанням технології блокчейн, необхідно вибрати підхід, який відповідає потребам прототипу та є оптимальним для бюджету проекту. Одним із найважливіших показників тут є необхідність створення власного протоколу. Протокол є головним у децентралізованих додатках на блокчейні. Програми, які потребують реалізації нових ідей у світі блокчейну, мають реалізувати власний протокол. Так, наприклад, системам для децентралізованого керування файлами знадобиться власна реалізація протоколів, оскільки наразі

не існує ідеальної платформи для цього унікального завдання. Якщо бізнес і програма зможуть використовувати існуючі протоколи, це значно знизить витрати на її розробку. Прикладом існуючої платформи є Etherium, який дозволяє швидко розширити ваш блокчейн-проект, дозволяє використовувати токени платформи та кошти учасників, які вже капіталізували цю платформу. Це означає, що система використовуватиме токени з платформи, яка вже завоювала довіру. Etherium також дозволяє створювати власні токени на основі ефірів. Дуже важливою перевагою є можливість використовувати віртуальну машину Etherium разом зі смарт-контрактами. У деяких випадках існує додаткова проблема конфіденційності блокчейну, крім самої можливості того, що доступ до нього має лише обмежена кількість людей. Ці фактори визначають, як блокчейн буде реалізовано в проекті. Використання вдосконаленого методу розповсюдження криптографічних токенів дозволяє підвищити стабільність і надійність децентралізованих додатків.

На етапі tech stack selection – визначається спосіб взаємодії з блокчейном. Він буде асинхронним. Клієнтська програма робить запит практично на будь-яку дію з блокчейном, а потім очікує відповіді. Це важливий фактор потрібно враховувати при виборі стека технологій для реалізації вебклієнта для взаємодії з блокчейном. JavaScript ідеально підходить для процесу асинхронної інтеграції, і існує багато фреймворків, які можуть полегшити розробку. Усе це, а також вибір того, як відбувається комунікація в блокчейні, є вибором технологічних стеків. Так як наша програма буде використовувати розумні контракти, нам також потрібно вибрати зручний спосіб для їх робзробки та відлагодження. Велику увагу на цьому етапі потрібно приділити прогнозування того, як буде виглядати система для та розробки та тестування нашого Fapp. Це забезпечить швидшу розробку та скоротить час, витрачений на усунення помилок. Для кращої розширюваності додатку використаємо парадигму предметно-орієнтованого програмування при розробці вебклієнта.

На етапі реалізації відбувається практичне створення

децентралізованого блокчейн додатку. Цей етап базується на рішеннях, запропонованих на попередніх етапах розробки. В нашому випадку фактично, розробка зводиться до реалізації розумних контрактів для блокчейн платформ з використанням вдосконаленого методу розповсюдження криптографічних токенів та розробки вебклієнта, який реалізує зручний інтерфейс для використання блокчейн-сховища. Це зробить більш доступним його використання для широкого кола людей, адже не потрібно буде інсталювати настільні програми для комп'ютера.

Під час релізу відбувається запуск готової робочої версії програми та запуск різних маркетингових компаній для капіталізації блокчейну та для реклами продукту.

При власнй реалізацією токен-протоколу потрібно організувати продаж початкових токенів тобто потрібно залучити інвестиції та провести капіталізацію перших токенів децентралізованої системи.

Перших два етапи належать до планувальної стадії проекта. На них творюється стратегія та вирішуються базові архітектурні та стратегічні питання. Формується чітке технічне завдання із конкретно визначеними межами та скінченною кількістю функцій або властивостей. Прийняті рішення допоможуть зрозуміти навички та отримати досвід, який має бути у команди розробників для успішного та професійного ведення та розробки проєктування децентралізованої блокчейн системи.

Далі важливо розробити концепти ідеї та забезпечити розробників відповідним середовищем, адже воно може допомогти серйозно прискорити процес створенняи якісної програми. Найкраще використати існуючі фреймворки для створення фронтенд вебклієнта та проектування розумних контрактів, для пришвидшення роботи.

В залежності від обраних підходів та протоколів, як вже згадувалося вище, потрібно додатково організувати продаж крипто-токенів для капіталізації розробленого протоколу.

3.2 Алгоритм удосконаленого розподілу криптографічних токенів

Головною особливістю в запропонованому алгоритмі розподілу крипто-токенів є передача відповідальності за виведення токенів з смарт-контракту на учасників мережі. Такий підхід реалізується за рахунок створення додаткового словника, де адреса учасника буде відповідати кількості токенів, доступних для виведення на його рахунку. Таким чином, коли відбудеться перерозподіл токенів, контракт додасть їх спочатку до словника, а не надсилатиме безпосередньо на облікові записи учасників, що може призвести до нестабільності системи. Крім того, учасники зможуть індивідуально перевіряти стан своїх акаунтів на повернення токенів і окремо формувати запити на повернення криптографічних токенів. При реалізації такого алгоритму помилки на етапі запиту стосуватимуться лише того учасника, який їх спричинив. Всі решта учасників будуть у безпеці та ізольовані від впливу. Суть запропонованого підходу полягає в тому, що смарт-контракт може перерозподіляти криптотокени або одразу всім учасникам системи або нікому. Якщо транзакція буде скасована, то це вплине на всіх учасників, яким потрібно було надіслати криптографічні токени. Це приведе до зриву контракту, а всі крипто-токени повернуться учасникам.

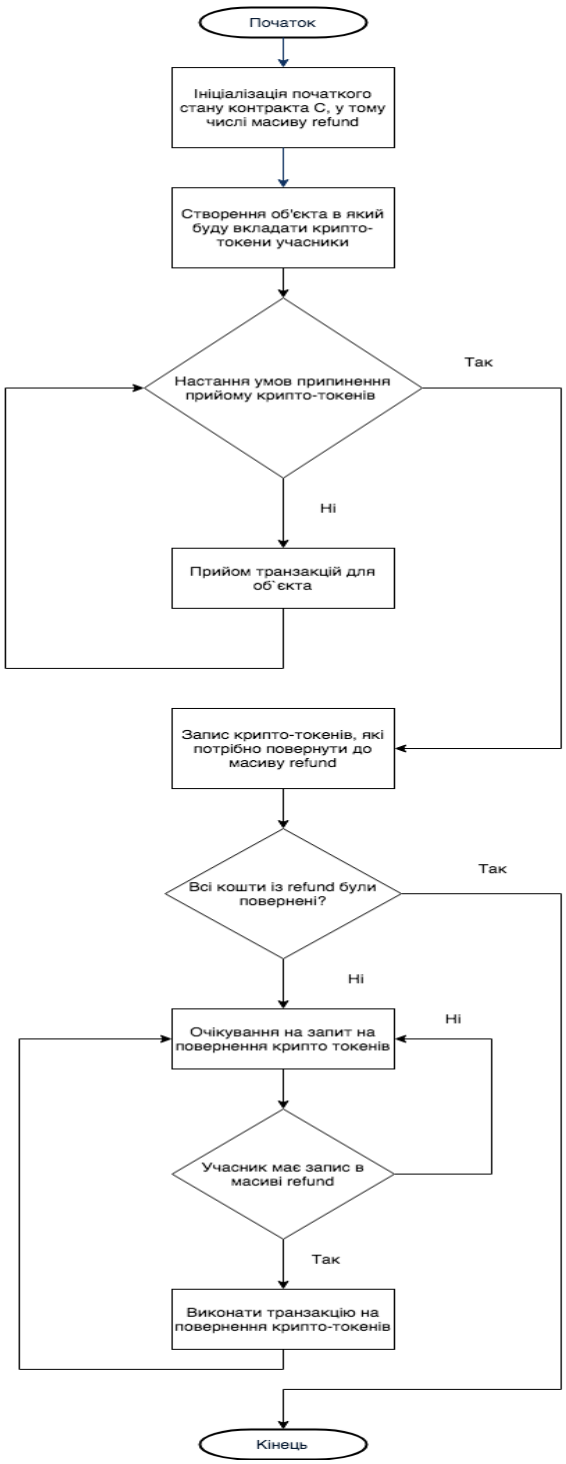


Рисунок 3.6 - Алгоритм удосконаленого розподілу криптографічних токенів

В цілому, алгоритм роботи, наведений на рисунку 3.6 полягає в наступному:

– програмний код смарт-контракту завантажується на платформу Ethereum;

– розумний контракт отримує змогу керувати крипто-токенами, таким чином учасники консенсусу можуть посилати свої токени смарт-контракту. Смарт-контракт приймає транзакції та токени учасників угоди;

– при заключені угоди смарт-контракт здійснює розподіл крипто-токенів. Отриманий результат зберігається в спеціально створеному словнику refund. ;

– смарт-контракт дозволяє учасникам угоди отримати інформацію про стан свого нового refund значення;

– учасники формують запити на отримання свої криптографічних токенів. Смарт-контракт перерозподіляє токени та обнуляє значення refund у словнику для кожного учасника.

З точки зору структур даних, ефективним буде використання ассоціативних таблиць або словників.

Асоціативний масив (aassociative array) або словник (dictionary), хеш (hash) — абстрактний тип даних (або інтерфейс до сховища даних), що дозволяє для зберігання даних використовувати набір пар ключ-значення та доступу до значень по їхньомум ключу.

Реалізації словників в більшості мов програмування зазвичай підтримують операції додавання, пошук і видалення пари за ключем.

Якщо списки - це масиви посилань на об’єкти, які підтримують доступ доелементів за їх позиціями, то словники є невпорядкованими таблицями посилань на об’єкти, які використовують доступ до елементів за допомогою ключа. Внутрішньо словники реалізовані як хеш-таблиці, це забезпечує високу швидкість пошуку за асиптотику O(1).

Передбачається, що асоціативний масив не може мати двох пар з однаковими ключами. Залежно від реалізації, ключі та значення можуть бути вказані як набори значень.

З точки зору інтерфейсу словники зручно розглядати як звичайний масив, в якому в якості індексів можуть використовуватися не тільки цілі числа, а й можливі значення інших типів, наприклад, рядок.

Словник має містити інформацію про адресу користувача облікового запису та суму розподілених токенів, які йому належать. Інформація про адрес користувача є унікальною і може бути отримана із запита на перевірку стану крипто-токенів. Завдяки особливостям словників, інформація про стан може бути отримана швидко за сталий час.

Таблиця 3.1 Структура словника

Адреса користувача	Токени
0xd60e75848F7c0312503D39761F549826E80B761d	105 ETH
0x078F6D43909Ff1F7279164cC013Ae7B0d118028a	10 ETH
…	
0x1aB29735dCDd242B4a43aDa126c878AEB1E37590	52 ETH

3.3 Висновки

Запропоновано метод реалізації децентралізованих застосунків за рахунок використання удосконалений метод розподілу крипто-токенів. Головною особливістю є передача відповідальності за виведення токенів з смарт-контракту на учасників мережі. Такий підхід реалізується за рахунок створення додаткового словника, де адреса учасника буде відповідати кількості токенів, доступних для виведення на його рахунку. Таким чином, коли відбудеться перерозподіл криптографічних токенів, смарт-контракт додасть їх спочатку до словника, а не надсилатиме безпосередньо на облікові записи учасників, що може призвести до нестабільності системи. Крім того, учасники зможуть індивідуально перевіряти стан своїх акаунтів на повернення токенів і окремо формувати запити на повернення криптографічних токенів. При реалізації такого алгоритму помилки на етапі запиту стосуватимуться лише того учасника, який їх спричинив.

Було визначано основі етапи проєктування децентралізованих застосунківна основі технології Blockchain, що притаманні додаткам такого

типу, з використанням парадигми об’єктно-орієнтовано тпроєктування.

Система використовуватиме токени з платформи, яка вже завоювала довіру. Etherium також дозволяє створювати власні токени на основі ефірів. Дуже важливою перевагою є можливість використовувати віртуальну машину Etherium разом з розумними контрактами. Ці фактори визначають, як блокчейн буде реалізовано в проекті. Використання вдосконаленого методу розповсюдження криптографічних токенів дозволяє підвищити стабільність і надійність децентралізованих додатків.

4 РЕАЛІЗАЦІЯ МЕТОДУ РОЗРОБКИ ДЕЦЕНТРАЛІЗОВАНИХ ДОДАТКІВ

4.1 Розробка прототипу додатку

В рамках кваліфікаційної роботи реалізовано прототип децентралізованого додатку з використанням технології Blockchain. Програмна реалізація прототипу створює децентралізовану платформу функціонквання аукціонних торгів для цифрової власності.

Вимого мінімальної життєздатної версії програми сформовано наступним чином:

- надати можливість створювати торгівельні лоти:
- надати можливість реєструвати цифрові активи на аукціон;
- зформувати час, на який буде активним аукціон;
- зформувати поріг суми заробітку, при досягнення якої буде відбуватись викуп лота;
- дозволити власникам блокчейну формувати комісію за успішні викупи лотів;
- забезпечити надійний розподіл крипто-токенів;
- забезпечити повернення коштів, з не виграшних ставок аукціону;
- власник лота має одержпти найвищу ставку, яку запрпонували за лот у ході торгів;
- забезпечення консенсусу та відсутність арбітражної сторони для визначення результатів аукціону.

Для мінімальної життєздатної версії було застосовано варіант торгів за віртуальну NFT, яку можна буде реєструвати через клієнт програми.

NFT — це запис у реєстрі блокчейнів, який представляє реальний об’єкт. Завдяки NFT можна оцифрувати будь-який об’єкт, відео, зображення, тощо. По суті, це цифровий сертифікат, що підтверджує володіння певним активом. Якщо NFT є цифровим активом, то об’єкт може існувати в певній формі.

Наприклад, фотографії, відео та інше. Більше того, їх можна отримати в необмеженій кількості з інтеренту. Сам NFT токен дає право володіти оригіналом. NFT не є альтернативою авторському праву. Так як NFT не можна скопіювати, він зберігає ту саму вартість, що й оригінал. Тому виникає певна плутанина, що NFT надає авторські права на оригінал.

Цьому помилковому уявленню сприяє той факт, що NFT часто збігаються з продуктами, які є об'єктами авторського права. Наприклад, в Україні авторське право згідно законодавства поширюється на музику, твори мистецтва, фотографії, тощо. Вони виникають автоматично з моменту створення твору автором. Іншими словами, творцям не має необхідності звертатися за додатковим захистом від держави, а лише потрібно мати підтвердження того, що ви є автором.

Стандарти NFT — це принципи, які забезпечують безперебійну роботу технології. Вони описують, як створити незамінні токени в протоколі блокчейн.

Найпопулярнішими блокчейн-платформами NFT є Ethereum, Flow і Tezos. Їхні стандарти керують ринком. Розглянемо стандарти Ethereum:

ERC-721 є найстаршим стандартом токенів, який досі популярний. Тут описано, процес створення NFT токенів на платформі Ethereum. Кожен токен ERC-721 унікальний та може оцінюватися незалежно від інших. З цієї причини більшість цифрових художників використовують токени ERC-721 для своїх творінь. Їх також не можна знищувати чи копіювати.

ERC-998 — це технологія взаємозамінних токени, які можна комбінувати з токенами інших цифровими активами та продавати як єдине ціле. Цей стандарт часто використовується як портфоліо для організації цифрових активів.

ERC-1155 - був описаний в основному для ігор. Взаємозамінні токени, є внутрішньоігровою валютоюа, часто використовуються для придбання різних віртуальних внутрішньоігрових предметів та інших цифрових предметів колекціонування у формі NFT. Може використовуватися як для

взаємозамінних так і незамінних токенів.

Кожна така власність буде представлена записом в мережі блокчейн, у якого є власник. В перспективі, далі можна буде реалізувати зв'язок із предметами реального світу. Така інтеграція може мати практичне застосування, проте вимагає багато затрат і напрацювань в галузі інтернету речей для того, щоб зв’язати показники та давачі, для оцифровування реального світу із блокчейном.

4.2 Обгрунтування стеку технологій

Для реалізації проєкту необхідно вибрати оптимальний спосіб реалізації блокчейну. Адже програма децентралізованого аукціону використовує мережу блокчейн як публічну базу даних. Для аукціону необхідна відсутність арбітражної сторони, тому її можна замінити використанням смарт-контрактів. Аукціон працює з цифровою власністю використовуючи криптографічні токени. Протокол, який був взятий за основу розподілу токінів, не матиме значення для здійснення обмінів. Щоб уникнути додаткової капіталізації краще використатити наявність існуючої мережі користувачів. Це допоможе виключити етап початкового продажу власних токенів з процесу розробки програми. В роботі буде використовуватися платформа Ethereum, і запропонована нею віртуальна.

Алгоритм консенсусу та криптографія, яка реалізована на Ethereum є також достатніми для поточної задачі.

Крім блокчейна, наш додаток повинен мати зручний інтерфейс для роботи і з блокчейном і зі смарт-контрактами. Так як обрана платформа дає можливість взаємодії для різних мов програмування, зокрема JavaScript, то в якості клієнтської частини може виступати вебдодаток. Вебдодаток має бути асинхронний, адже взаємодія з блокчейном носить асинхронний характер. Для таких потреб підходить мова програмування JavaScript.

Для розробки JavaScript додатка та організації бізнес логіки процесу був

використаний фреймворк Knockout.js [14].

Knockout — це бібліотека JavaScript, яка допомагає створювати розширені, адаптивні інтерфейси користувача для дисплеїв і редакторів із чистою основною моделлю даних. Якщо у вас є розділи інтерфейсу користувача, які оновлюються асинхронно і динамічно (наприклад, змінюються на основі дій користувача або коли змінюється зовнішнє джерело даних), Knockout допоможе вам реалізувати їх простіше та зручніше.

Відстеження залежностей - автоматично оновлює правильні частини вашого інтерфейсу користувача, коли ваша модель даних змінюється.

Декларативні прив’язки - це простий і очевидний спосіб зв’язати частини вашого інтерфейсу користувача з моделлю даних. Ви маєте змогу легко створювати складні динамічні інтерфейси користувача, використовуючи довільно вкладені зв’язувальні контексти. Доступна реалізація власної поведінки, як нових декларативних прив’язок для легкого повторного використання всього за кілька рядків коду.

Чиста бібліотека JavaScript - працює з будь-якою серверною або клієнтською технологією. Можна додати поверх існуючої вебпрограми без серйозних архітектурних змін.

Розробники, знайомі з Ruby on Rails, ASP.NET MVC або іншими технологіями MV*, можуть розглядати MVVM як форму MVC реального часу з декларативним синтаксисом [17].

В нашому децентралізовану аукціоні є багато місць, щоі міняються після виконання транзакцій на блокчейні, тому такий функціонал потрібний для зручності розробки. В проекті також використанано модульність, ефективне оголошення змінних, зручний запис функцій-зворотнього виклику. Не всі з цих можливостей завжди підтримуються браузером, тому для підготовки оптимального JS файла, використовується популярний пакувальник Webpack [25].

Webpack — це статичний модульний збірник для програм JavaScript. Програми, написані на JavaScript, постійно ускладнюються, тому для збору

модулів все частіше використовується спеціальний інструмент - бандлер. Такі інструменти дозволяють розробникам пакувати, компілювати, організовувати всі ресурси, необхідні для проекту. Ви можете використовувати не тільки сторонні бібліотеки, але і власні файли. Ця модульна система дозволяє досягти кращої організації проекту, так як він розділений на невеликі модулі.

Webpack на теперішній час є одним із найпотужніших подібних бандлерів, тобто модульних асемблерів. Він має відкритий код і дозволяє вирішувати найрізноманітніші завдання. Для роботи з платформую Ethereum, вебклієнту потрібна бібліотека web3.js [7]. Web3.js — це бібліотека JavaScript, яка використовується для взаємодії з блокчейном Ethereum. Цікавим у Web3.js є його роль у відході від фази Web 2.0. З переходом до Web 3.0 старі сайти, які заповнюють простір Web 2.0, будуть змушені знайти спосіб підключення до блокчейну.

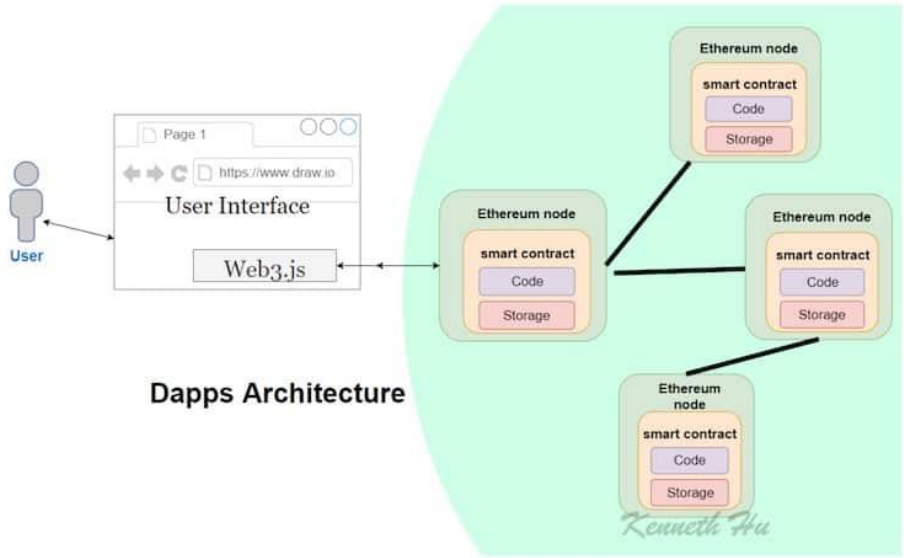


Рисунок 4.1 – Використання Web3.js

Вона дозволяє формувати та відправляти запити до блокчейна, виконувати транзакції та взаємодіяти із смарт-контрактами в зручний спосіб. Для використання цієї бібліотеки потрібно, щоб клієнт мав встановлений плагін, який би дозволив web3.js використовувати блокчейн. MetaMask - це криптовалютний гаманець. Інтерфейс дуже популярний серед користувачів

блокчейну і є, мабуть, найпопулярнішим програмним забезпеченням для взаємодії з криптовалютою. MetaMask доступний як розширення для браузера, яке працює в Firefox, Chrome, Brave.

Загалом, MetaMask - це гаманець без кастодіального права, який дозволяє зберігати криптоактиви в мережах Binance Smart Chain, Ethereum, Polygon та інших блокчейнах. На відміну від гаманця банку чи біржі, MetaMask не контролює та не обмежує транзакції з вашими монетами. Таким чином, якщо ви належним чином зберігаєте криптографічні активи та надійно зберігаєте приватні ключі, гроші в гаманці належать виключно вам. Він розповсюджується, як плагін для браузера в якому можна керувати своїми рахунками.

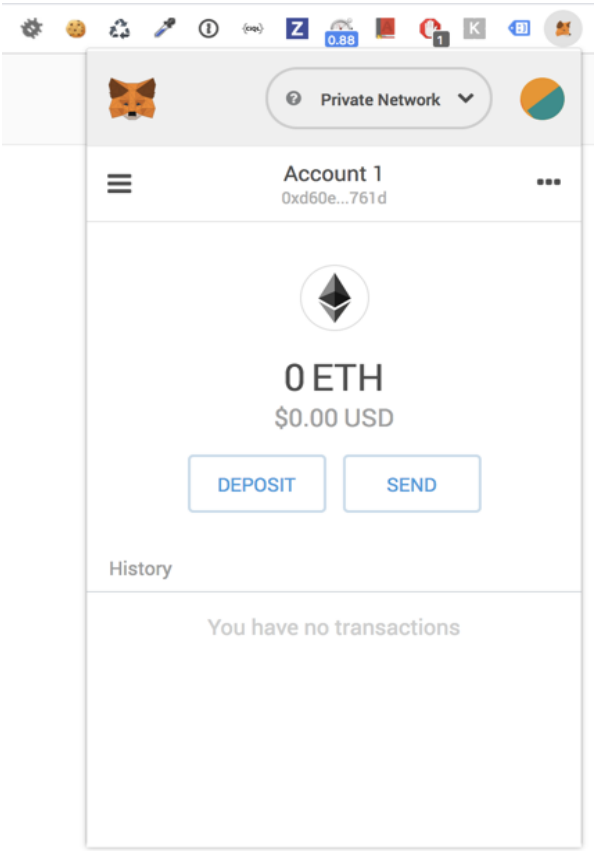


Рисунок 4.2 - Графічний інтерфейс плагіну для браузера MetaMask

MetaMask описують як «гарячий гаманець». Це означає, що він підключений до Інтернету та сумісний з деякими функціями в Інтернеті. Це робить взаємодію з децентралізованими програмами на основі блокчейну дуже

простою.

Для зручної роботи із смарт-контрактами використано Truffle фреймворк [19]. Truffle — це найпопулярніше середовище розробки для Ethereum, яке має на меті значно спростити ваше життя. Особливості Truffle Framework:

– автоматизоване тестування контракту для швидкої розробки;

– вбудована компіляція смарт-контрактів, зв'язування, розгортання та керування двійковими файлами.

– сценарна структура розгортання та міграції;

– зовнішній механізм створення сценаріїв, який виконує сценарії в середовищі truffle;

– керування мережею для розгортання як у публічних, так і в приватних мережах;

– доступ до великої кількості зовнішніх пакетів.

– створений для швидкості.

Він дозволяє організувати завантаження, та відлагодження смарт-контрактів та полегшити звернення до них. В даному фреймворці, використовується можливість зручного та швидкого завантаження смарт контрактів на блокчейн. Для цього Truffle, дає можливість створювати скрипти для міграції.

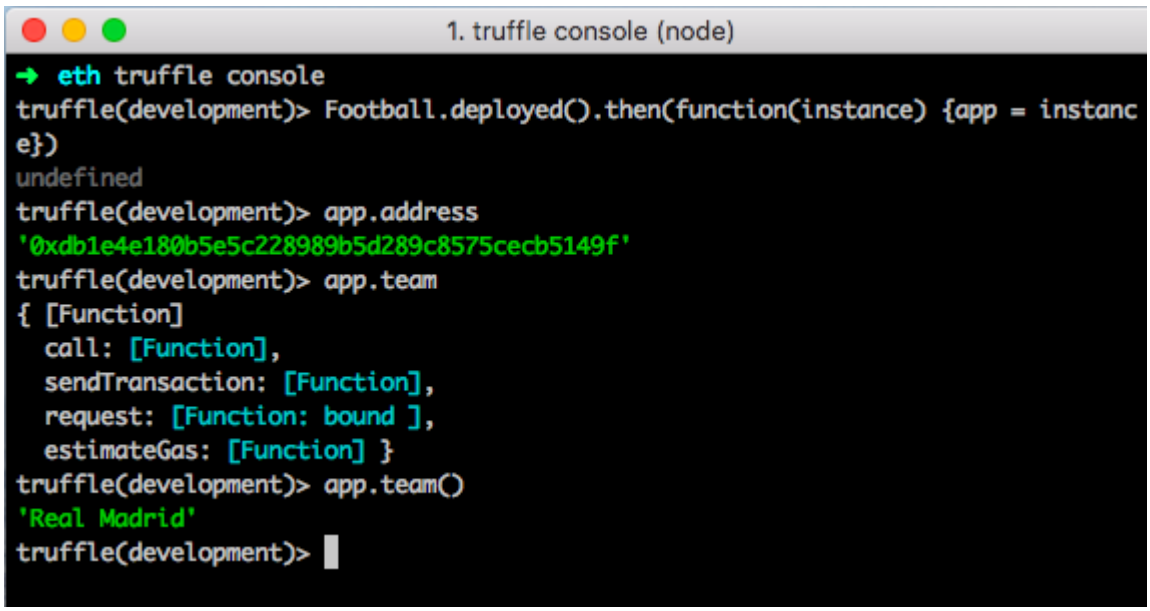


Рисунок 4.3 – Використання Truffle

Для збереження та використання інформації про завантажені контракти, використовується окремий контракт Migrations.

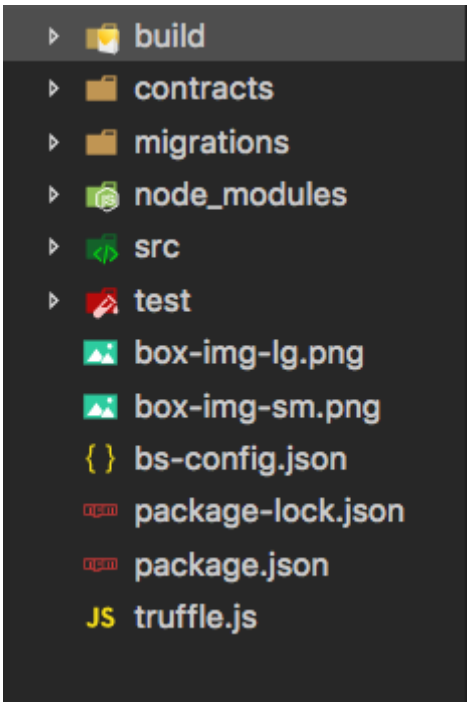


Рисунок 4.4 – Структура папок проєкту

На рисунку 4.5 наведено приклад смарт-контракту:

```
pragma solidity ^0.4.2;

contract Migrations {
  address public owner;
  uint public last_completed_migration;
  modifier restricted() {
    if (msg.sender == owner)
        _;
}
  function Migrations() {
    owner = msg.sender;
}
  function setCompleted(uint completed) restricted {
    last_completed_migration = completed;  }
  function upgrade(address new_address) restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

Рисунок 4.5 - Приклад смарт-контракту

Для написання смарт-контрактів використано Solidity. Ця мова програмування дає розробникам можливість створювати власні децентралізовані програми. З магазину додатків можна завантажити будь-які програми, від ігор до фітнесу та банківських послуг. Dapps те саме. Основна відмінність полягає в тому, що вони мають відкритий код і не мають посередників. Подібно до того, як програми в магазині додатків Apple створені для роботи на iOS, Dapps на Ethereum створені для роботи на Solidity.

У Dapps є фрагменти коду, відомі як смарт-контракти. Вони допомагають людям обмінювати гроші, акції, майно та майже будь-що цінне за певних умов. Це усуває потребу в дорогій третій стороні, такій як нотаріус.

4.3 Архітектура децентралізованого застосунку

Наш децентралізований додаток складається із клієнтської частини у вигляді вебдодатку, смарт-контрактів реалізованих на блокчейні платформи Ethereum. Користувач, має бути зареєстрований на Ethereumі та використовувати браузер із MetaMask плагіном для зв’язку із блокчейном та використання гаманця. MetaMask розширенням виступає проміжною ланкою між деценталізованим блокчейном та смарт-контрактами. Не потрібно завантажувати весь блокчейн локально, а є можливість роботи прямо з браузера. Він підключає бібліотеку web3.js, яка виконує зв’язуючу роль. При цьому додається глобальний об’єкт в змінну window, в якій розташовано провайдер - web3.currentProvider.

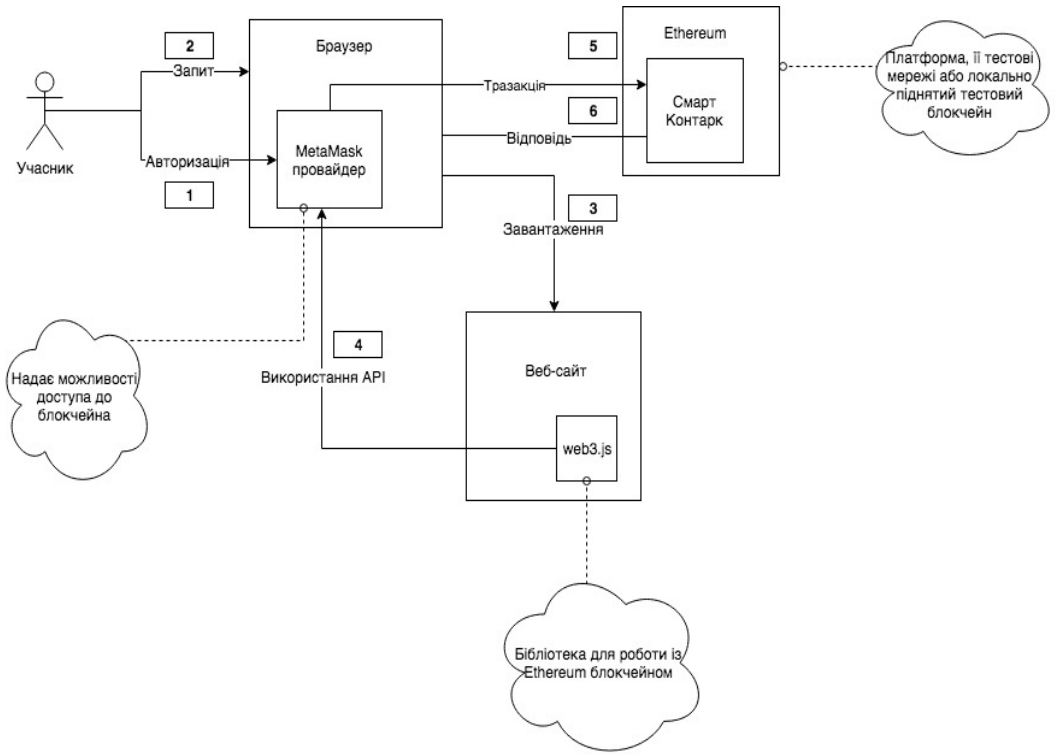


Рисунок 4.6 - Архітектура децентралізованого аукціону

Головним компонентом архітектури є смарт-контракти. Вони реалізовані на мові програмування Solidity версії 0.4.2.

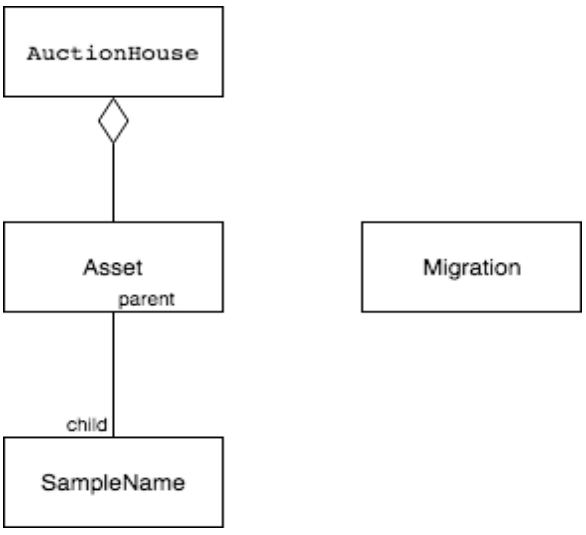


Рисунок 4.7 - Структура смарт-контрактів додатку

Головним смарт-контрактом в додатку є AuctionHouse. Він керує роботою аукціона та проводить розподіл крипто-токенів. Він оперує компонентом Asset, який є абстрактним представленням віртуальної

власності. Смарт-контракт SampleName є прикладом реалізації простої власності. AuctionHouse дає можливість для створення нових лотів, формування ставок від учасників мережі та виконую арбітражну роль, тобто визначає переможця. Як ми згадували, смарт-контракт Migration є системним та автоматично створюється фреймворком Truffle для швидкого завантаження контрактів та зберігання всієї системної інформації про смарт-контракти на блокчейні.

В смарт контракті AuctionHouse реалізовано запропонований у роботі удосконаленй алгоритму розподілу крипто-токенів. Перерозподіл токенів відбувається, коли приходить нова ставка на транзакцію.

Для AuctionHouse важливим є запам’ятати учасника, ставка якого була останньою для конкретного аукціону.

Вебдодаток є зручним способом взаємодії із блокчейн та зручним інтерфейсом для проведення дій над аукціонами. Вебдодаток є JavaScript застосунком, що складається із трьох основних сторінок.

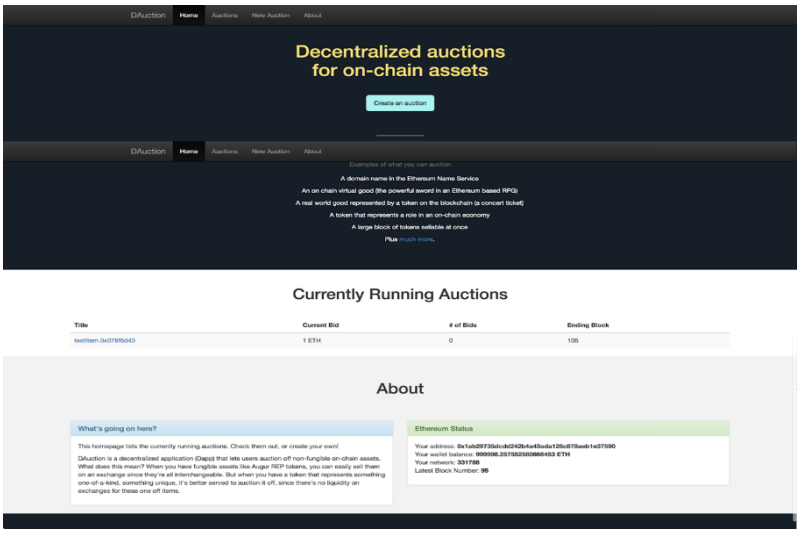


Рисунок 4.8 - Головна сторінка додатку

На головній сторінці відображається список створених аукціонів. На ній показується інформація про стан блокчейну, адресу рахунку, кількість доступних токенів на рахунку.

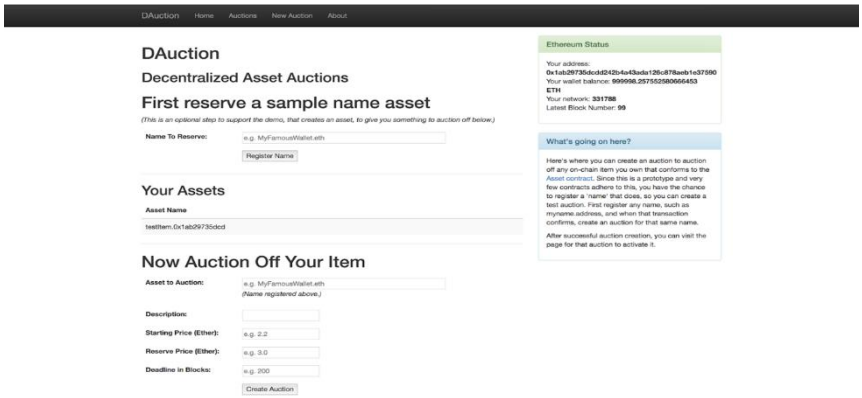


Рисунок 4.9 - Створення нового аукціону

Створення нового аукціону складається із форми для реєстрації тестового SampleName власності, та форми створення нового аукціону. На ній знаходиться інформація про стан блокечейну. Час завершення вказаний в кількості блокчейн блоків, через які настане закінчення аукціону. Також є можливість задати мінімальний поріг ставки, до якої цифрова власність не буде вважатись проданою.

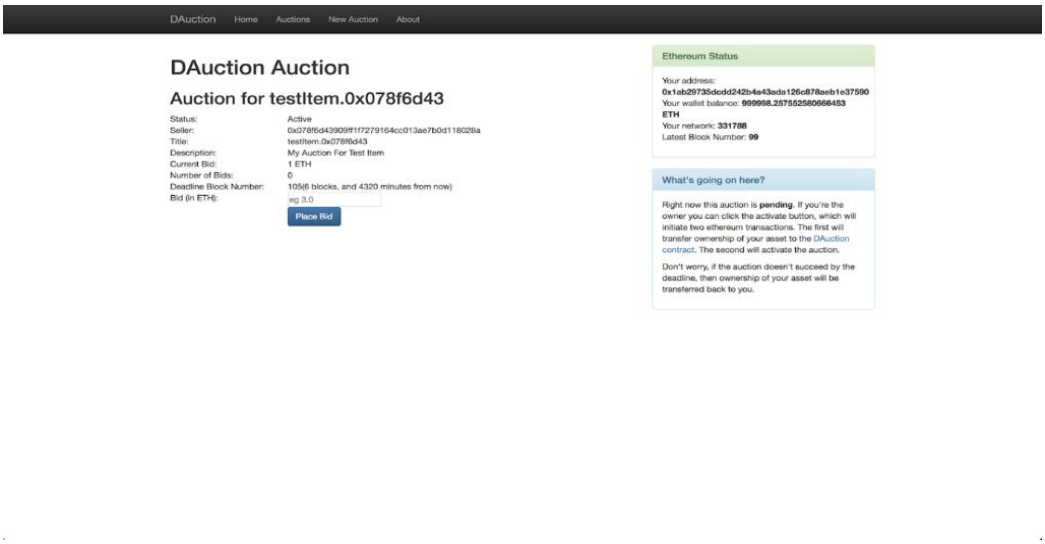


Рисунок 4.10 - Активований аукціон

Інформація про аукціон містить статус аукціону, адресу продавця, найвищу поточну ставку, кількість ставок, час завершення, базуючись на швидкості створення блоків.

Розробка додатку велась в середовищі створеному за допомогою

технології віртуалізації Docker [21]. Для роботи було створено два докер контейнера auction_geth і auction_app. На auction_geth для імітації роботи Ethereum платформи налаштовано та запущено Ganache - тестовий сервер, який дозволяє ефективно виконувати транзакції на блокчейні, перевіряти статус їх виконання та виконувати Truffle тести. В контейнері auction_app міститься код веб-клієнта, який запускається за допомогою Webpack сервера. Перед запуском, пакувальни Webpack компілює JavaScript в стандарт ES5, котрий розуміють всі сучасні браузери, та компілює створені смарт-контракти. Веб-клієнт із докер контейнера auction_app по локальній IP адресі тестового блокчейна на контейнері auction_geth взаємодіє із ним, коли йде мова про міграцію смарт-контрактів. Прискорення роботи застосунку досягається за допомогою використання швидкого тестового блокчейна та можливості вчасно виконувати транзакції і взаємодіяти із смарт-контрактами. Додатково, використання тестового блокчейну не потребує реальних фінансових витрат при тестуванні та виконанні транзакції, запуск власних смарт-контрактів в тестовому режимі.

Рисунок 4.11 – Налаштування середовища віртуалізації Docker

Docker - це програмна платформа для швидкої розробки, тестування та розгортання програм. Docker компонує програмне забезпечення в стандартизовані блоки, які називаються контейнерами. Кожен контейнер має усе необхідне для роботи програми: системні інструменти, бібліотеки, код і середовище виконання. За допомогою Docker ми можемо швидко розгортати та масштабувати свої програми в будь-якому середовищі та бути впевненими,

що ваш код працюватиме. Він базується на стандартизованому способі виконання коду. Docker - це контейнерна операційна система. Контейнери створюють віртуальне представлення операційної системи сервера. Після встановлення на кожному сервері Docker надає доступ до простих команд, необхідних для створення, виконання, запуску або зупинки контейнерів.

4.4 Висновки

В рамках кваліфікаціної роботи реалізовано прототип децентралізованого додатку з використанням технології Blockchain. Програмна реалізація прототипу створює децентралізовану платформу функціонквання аукціонних торгів для цифрової власності. Клієнтська програма робить запит практично на будь-яку дію з блокчейном, а потім очікує відповіді. JavaScript ідеально підходить для процесу асинхронної інтеграції, і існує багато фреймворків, які можуть полегшити розробку. Усе це, а також вибір того, як відбувається комунікація в блокчейні, є вибором технологічних стеків.

Для мінімальної життєздатної версії було застосовано варіант торгів за віртуальну NFT, яку можна буде реєструвати через клієнт програми. Крім блокчейна, наш додаток має зручний інтерфейс для роботи і з блокчейном і зі смарт-контрактами. Так як обрана платформа дає можливість взаємодії для різних мов програмування, зокрема JavaScript, то в якості клієнтської частини може виступати вебдодаток. Вебдодаток має бути асинхронний, адже взаємодія з блокчейном носить асинхронний характер.

Створено архітектуру, яка відповідає потребам децентралізованого застосунку.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було отримано такі результати:

– проведено аналіз методів та засобів створення децентралізованих програм;

– удосконалено модель функціонування смарт-контрактів для покращення стабільності роботи;

– вдосконалено метод розподілу крипто-токенів у децентралізованих програмах на платформі Ethereum;

– розроблено алгоритм та програмне забезпечення для платформи Ethereum, яке реалізує розроблений метод;

– було створено прототип децентралізованого додатку для організації аукціонних торгів;

– досліджено ефективність розробленого алгоритму та засобів його реалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. “Crypto-currency market capitalizations,” 2022. [Електронний ресурс]. – Режим доступу: https://coinmarketcaps.com

2. David Farooq, (2019). A multi-layered blockchain framework for the smart mobility data-markets. Transportation Research Part C: Emerging Technologies. 111. 10.1016/j.trc.2020.01.002.

3. Richard Adams, Phil Godsiff and Glenn Parry, “The future of money and further applications of the blockchain,” First published: 21 December 2021, [Електронний ресурс]. – Режим доступу: https://doi.org/10.1002/jsc.2141.

4. A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: A blockchain model of cryptography and privacy-preserving smart contracts,” in Proceedings of IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 2016, pp. 839-858.

5. B. W. Akins, J. L. Chapman, and J. M. Gordon, “A whole new world: the income tax of the bitcoin economy,” 2013. [Електронний ресурс]. – Режим доступу: https://ssrn.com/abstract=2394738

6. V. Yatskiv, N. Yatskiv, and O. Bandrivskyi “Proof of Video Integrity Based on Blockchain,” in Proc. Advanced Computer Information Technologies (ACIT), 2019 IEEE 9th International Conference on, 2019, pp. 431-434.

7. D. Johnson, S. Vanstone, “The elliptic curve digital signature algorithm (ECDSA),” International Journal of Information Security, vol. 1, no. 1, pp. 36-63, 2001.

8. Laphou Lao, LI Ze-cheng, Bin Xiao, Songtao Guo and Yuanyuan Yang, “A Survey of IoT Applications in Blockchain Systems: Architecture, Traffic Modeling and Consensus,” 2021 [Електронний ресурс]. – Режим доступу: https://www.semanticscholar.org/paper/A-Survey-of-IoT-Applications-in-Blockchain-Systems/

9. “Decentralized Asset Exchange,” 2021 [Електронний ресурс]. – Режим доступу: https://bitshares.org/technology/decentralized-asset-exchange/

10. Віхтюк А.Р. Аналіз підходів до проєктування децентралізованих систем з використанням технології блокчейн / І.В. Муляр, О.В. Мірошніченко, А.Р. Віхтюк, В.Ю. Пічура, Л.В. Солдєва // Тези доповідей XVIII Міжнародної науково-практичної конференції " Військова освіта і наука: сьогодення та майбутнє" [Текст] – К. : ВІКНУ, 2022. – 57 с.

11. “Ethereum (ETH) Blockchain Explorer,” 2021 [Електронний ресурс]. – Режим доступу: https://etherscan.io

12. Blind signatures for untraceable payments [Електронний ресурс]. - Режим доступу: http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF (дата звернення 21.09.2022)

13. Hashcash - Wikipedia [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Hashcash (дата звернення 27.09.2022)

14. Satoshi Nakamoto Institute [Електронний ресурс]. - Режим доступу: https://nakamotoinstitute.org/ (дата звернення 17.09.2022)

15. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс]. - Режим доступу: https://bitcoin.org/bitcoin.pdf (дата звернення 17.09.2022)

16. Bitcoin's Security Model [Електронний ресурс]. - Режим доступу: https://www.coindesk.com/bitcoins-security-model-deep-dive/ (дата звернення 13.11.2022)

17. What is Ethereum? [Електронний ресурс]. - Режим доступу: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html (дата звернення 13.11.2022)

18. What is Blockchain Technology? A Step-by-Step Guide For Beginners [Електронний ресурс]. - Режим доступу: https://blockgeeks.com/guides/what-is-blockchain-technology/

19. Turing complete - Wikipedia [Електронний ресурс]. - Режим доступу: https://simple.wikipedia.org/wiki/Turing_complete (дата звернення 27.09.2022)

20. Solidity by Example [Електронний ресурс]. - Режим доступу: https://solidity.readthedocs.io/en/v0.4.21/solidity-by-example.html (дата звернення 27.09.2022)

21. Internet of things - Wikipedia [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Internet_of_things (дата звернення 11.09.2022)

22. Муляр І.В. Ітераційно - геометричний метод стійкого перцептуального хешування зображення / В.М. Джулій, Ю.П. Кльоц, І.В. Муляр, В.М. Чешун // Вісник Хмельницького націо-нального університету. - Хмельницький, 2020, № 1 (281). - С.76-79.

23. What is Tokenization? [Електронний ресурс]. - Режим доступу: https://blockonomi.com/tokenization-blockchain/ (дата звернення 31.07.2022)

24. Building for the Blockchain [Електронний ресурс]. - Режим доступу: https://blog.ycombinator.com/building-for-the-blockchain/ (дата звернення 09.11.2022)

25. Fat Protocols | Union Square Ventures [Електронний ресурс]. - Режим доступу: http://www.usv.com/blog/fat-protocols (дата звернення 08.08.22)

26. Why Dagger-Hashimoto for Ethereum? [Електронний ресурс]. - Режим доступу: https://medium.com/verifyas/why-dagger-hashimoto-for-ethereum-773f0792a689 (дата звернення 08.08.22)

27. A Review of Different Database Types: Relational versus Non-Relational [Електронний ресурс]. - Режим доступу: http://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/

28. Safe Decentralized Applications Development Using Blockchain Technologies / Viktor Cheshun, Ihor Muliar, Vasyl Yatskiv, Ruslan Shevchuk, Serhii Kulyna, Taras Tsavolyk // 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), 16-18 Sept. 2020, Deggendorf, Germany. – Publisher: IEEE, 2020. – P. 800-805.

29. 8 Steps to Start Blockchain Development [Електронний ресурс]. - Режим доступу: https://www.newgenapps.com/blog/8-steps-how-to-start-blockchain-development-dapp

30. How to Build a Decentralized Application (DApps) [Електронний ресурс]. - Режим доступу: https://www.devteam.space/blog/how-to-build-a-decentralized-application-dapps/

31. What are Decentralized Applications, DApps? [Електронний ресурс]. - Режим доступу: https://hackernoon.com/what-are-decentralized-applications-dapps-3b63b4d587fe

32. Муляр І.В. Метод розробки децентралізованих захищених програм на основі технології BLOCKCHAIN / Р.М. Глушко, І.В. Муляр // «Інтелектуальний потенціал – 2018» - збірник наукових праць молодих науковців і студентів з нагоди 30-річчя підготовки ІТ- фахівців в ХНУ/ Колектив авторів – Хмельницький: ПВНЗ УЕП, 2018. – Ч.2: Математичне моделювання та інженерія програмного забезпечення. – С. 28-31

33. Державна служба спеціального зв'язку та захисту інформації України [Електронний ресурс]. – Режим доступу: http://www.dsszzi.gov.ua/dsszzi/control/uk/publish/article?art_id=81998&cat_id=38835

34. Муляр І.В. Предметно-орієнтований погляд на розробку програмного забезпечення / І. В. Муляр, В. О. Браун, В. К. Шваб, Я. М. Проценко, Р. М. Глушко// Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2017. – Вип. № 58. – С. 123-133

35. Муляр І.В. Модель оцінки ймовірнісно-часових характеристик інформаційної взаємодії в мережі інтернет речей / В.М. Джулій, Б.М. Кізюн., І.В. Муляр // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2019. – Вип. №63. – С. 51-60

36. Enterprise Container Platform | Docker [Електронний ресурс]. – Режим доступу: https://www.docker.com/ (дата звернення 08.09.22)

ДОДАТОК А

(обов’язковий)

Копія графічного матеріалу

Мета кваліфікаційної роботи магістра підвищення стійкості децентралізованих програм за рахунок використання надійніших алгоритмів обміну крипто-токенами.

Об'єкт дослідження: децентралізовані програми, побудовані з використанням смарт контрактів на основі платформи Ефіріум, що працюють з розподілом крипто-токенів.

Предмет дослідження: моделі, та алгоритми, що використовуються при реалізації децентралізованих програм на блокчейні.

Задачі досліджень у роботі формулюються наступним чином:

- провести аналіз методів та засобів розроблення децентралізованих програм;
- удосконалити модель розподілу крипто-токенів для досягнення кращої стабільності роботи;
- вдосконалити метод розподілу крипто-токенів у децентралізованих програмах на платформі Ефіріум;
- розробити алгоритм та програмне забезпечення для платформи Ефіріум, яке реалізує розроблений метод.
- дослідити ефективність розробленого алгоритму та засобів його реалізації.

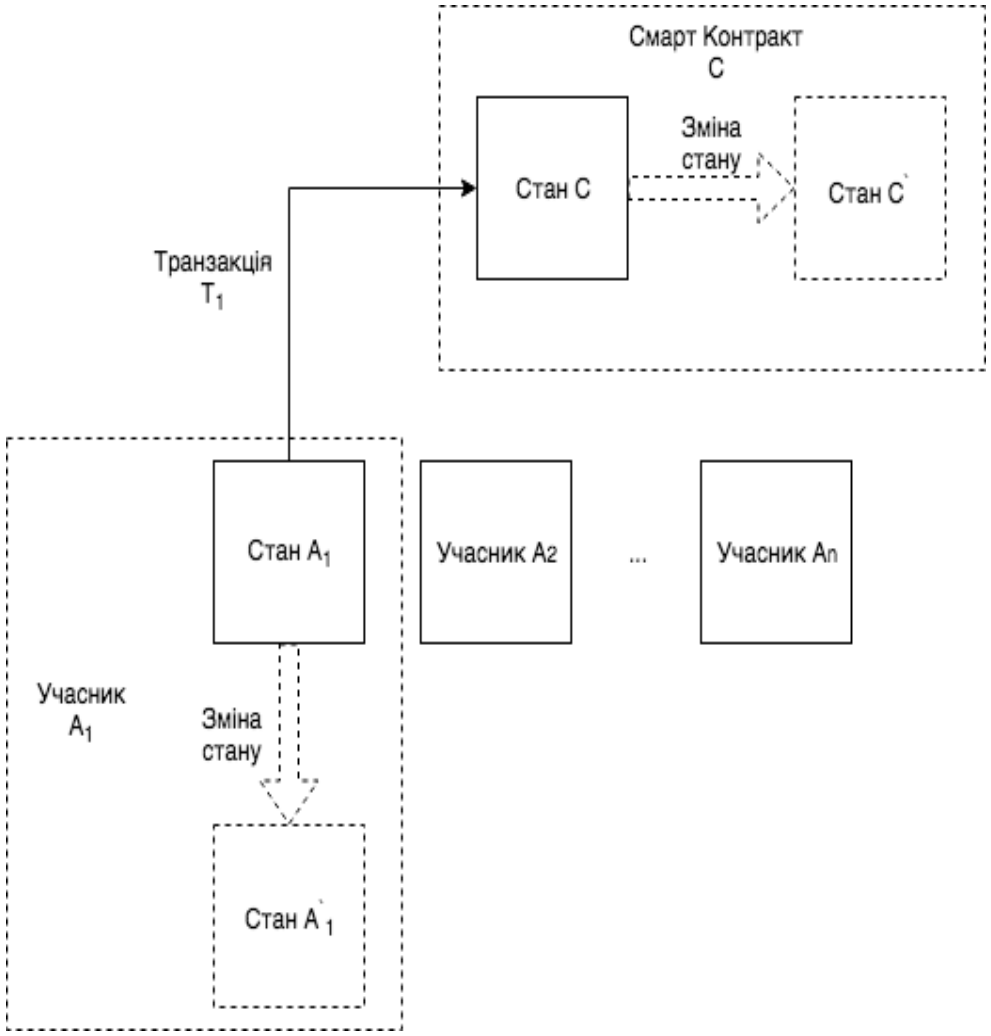
Наукова новизна роботи

Вдосконалено метод розподілу крипто-токенів, який забезпечує більшу надійність смарт контрактів та систем, побудованих на них.

Практична цінність. полягає в розробці алгоритму та його реалізації для розподілу крипто-токенів, який покращує стійкість та надійність смарт контрактів і децентралізованих програм, які їх використовують.

Публікації. По темі магістерської роботи опубліковано 1 стаття, 1 - тези доповідей на всеукраїнській конференції.

Модель взаємодії в мережі користувача A зі смарт-контрактом C.



Традиційне виконання транзакцій можна відобразити моделлю взаємодії в мережі користувача A зі смарт-контрактом C.

Смарт-контракт C є машиною станів із початковим станом C.

Також в мережі існують учасники $A_1, A_2,\ldots, A_n$, які є машинами станів із початковими станами $A_1, A_2,\ldots, A_n$.

Учасники роблять транзакції $T_1, T_2,\ldots, T_n$, після яких стан смарт-контракту C змінюється на C', C'',…, $C^{()n}$, а стани учасників змінюються на A_1', A_2',…,A_n'.

Коли в мережі настає подія P_1, смарт-контракт C змінює свій стан на $C^{()(n+1)}$ та виконує транзакцію CT_1, в якій розподілені крипто-токени повертаються назад до учасників, що змінює стани машин-автоматів A_i на A_1''', A_2''',…, A_n''' відповідно.

В ідеальній ситуації транзакції проходять без помилок і учасники отримують свої токени.

В такій моделі транзакції будуть виконанні тільки за умови відсутності помилок роботи смарт-контракту.

Але помилки можуть виникати через збої відправки і прийому транзакцій, вади в програмному коді смарт-контрактів і зловмисні дії стосовно ресурсів мережі тощо.

Оцінка ймовірності випадкової взаємодії з нестабільними смарт-контрактами

Оцінку проводили за статистикою користувачів платформи Etherium, наданої Etherscan, за якою з 44 080 162 перевірених користувачів 46 487 - це зареєстровані смарт-контракти.

Нехай подія B - учасник є смарт-контрактом, а подія B’ - учасник є людиною. Події B і B’ утворюють повну множину можливих варіантів і ми можемо оцінити їх ймовірність за формулами 1 і 2:

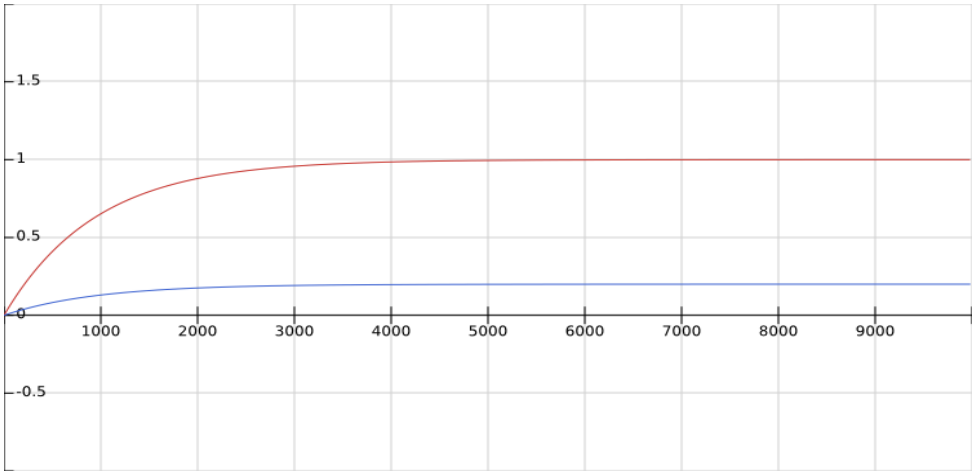
$$P(B) = \frac{N(B)}{N} = \frac{46\ 487}{44\ 080\ 162} = 0.001054601;$$

$$P(B') = 1 - P(B) = 0.998945399. \ ]$$

Тоді ймовірність події *A’*, при якій хоча б один із n учасників буде смарт-контрактом можна визначити за формулою 3:

$$P(A') = 1 - (P(A))^n.$$

Нехай подія *E* - це наявність смарт-контракту, який може стати причиною нестабільності контракту *C*.



Графік розподілу ймовірності подій A (червоний) та події D (синій) при P(E) = 0.2

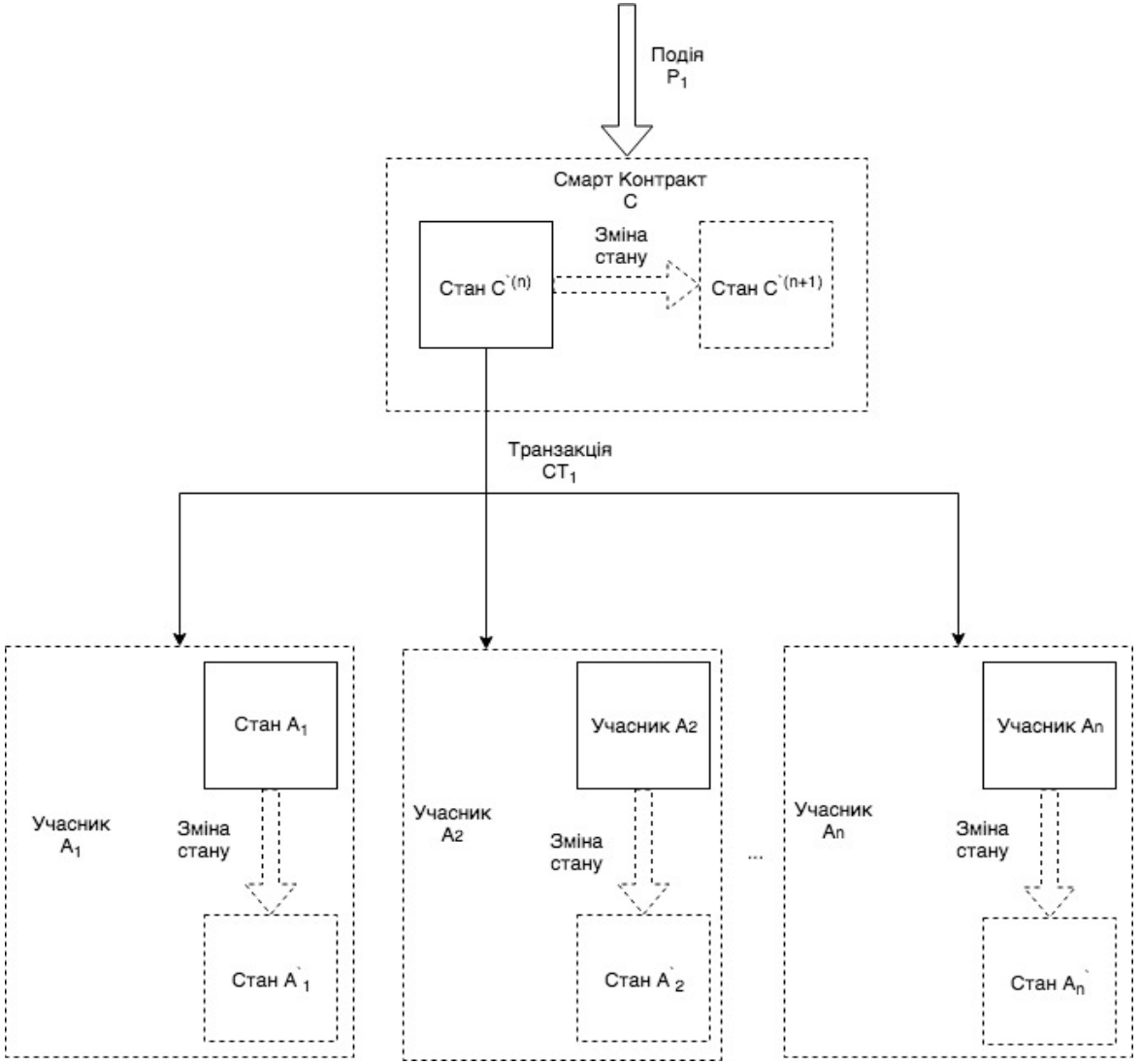
Як видно з графіку, ймовірності ростуть із збільшенням кількості учасників у системі, тому ймовірність випадкової помилки достатньо мала та залежить від ймовірності *P*(*E*).

Наступна ситуація - взаємодія не випадкова і обумовлена наміром нападника. Тоді P (D) буде наближено до 1, а подія D - ймовірність того, що в системі буде хоча б один смарт-контракт із проблемним кодом

Стандартний підхід до виконання смарт-контракту

Стандартний підхід працює у цьому випадку нестабільно. Він заключається у тому, щоб під час настання події P_n розподіляти крипто-токени і передавати їх усі разом через транзакцію CT_n.

У випадку помилки транзакція буде скасована та може статись так, що стан машини C залишиться незмінним. Це призведе до проблем із передачею та розподілом крипто-токенів у мережі.

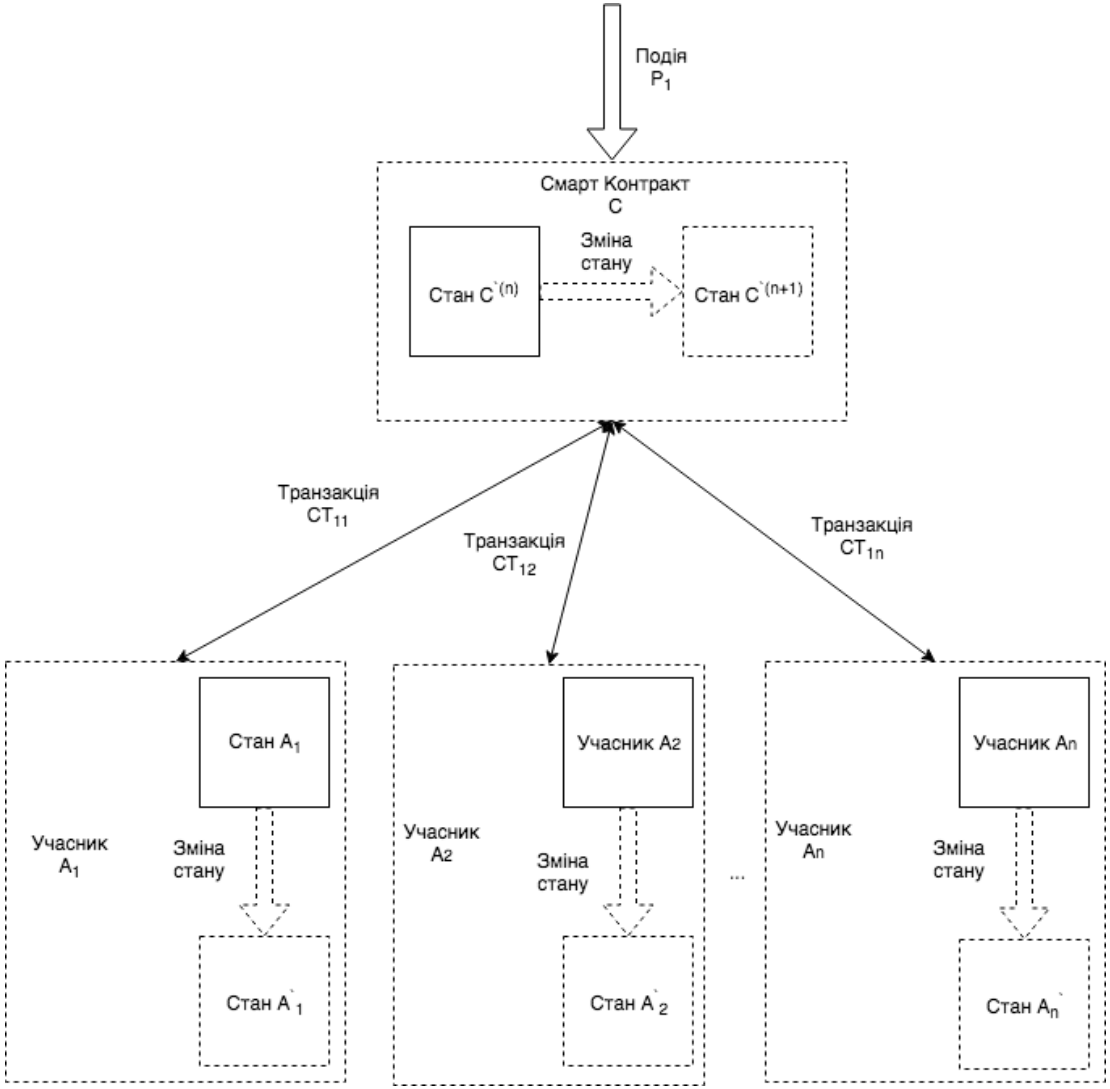


Вдосконалена модель розподілу крипто-токенів

Нехай взаємодія відбувається між контрактом C та учасниками *A1, A2,…, An.* Учасники здійснюють транзакції *T1, T2,…, Tn* „ після чого статус договору C змінюється на *C ', C' ', ..., C (') n*. Стани учасників змінюються на *A1 ', A2', ..., An '*. Коли подія *P1* відбувається в мережі *M*, контракт C змінює свій стан на $C^{(')}(n+1)$.

Під час зміни стану контракт C розподіляє крипто-токени всередині свого стану на рахунки учасників A_1', A_2', … , A_n'. На цьому ефект події P_1 закінчується.. Учасники *A1 ', A2', ..., An'* можуть перевірити стан своїх рахунків у договорі *C* та здійснити зняття коштів через транзакції *CT11, CT12,…, CT1n.*

Модель захищає від вектора нападів на смарт-контракти, коли зловмисник намагається відправляти транзакції через власний контракт, який запрограмований на помилку для корисної транзакції



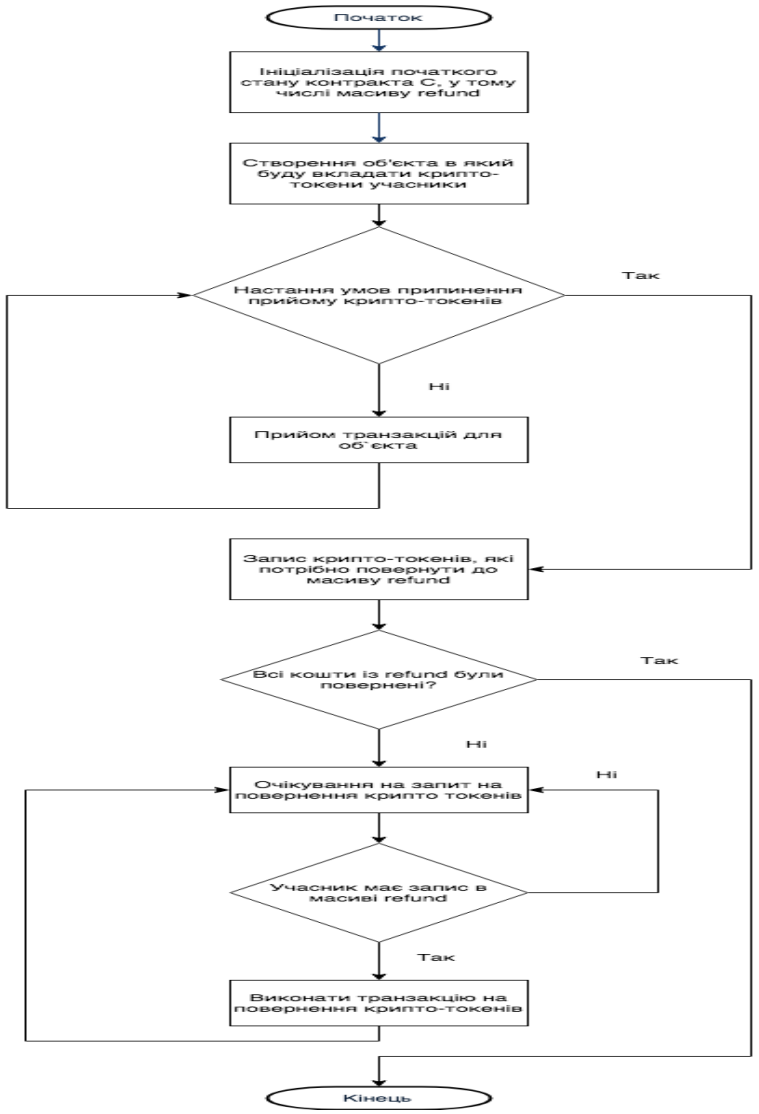
Етапи розробки децентралізованих додатків на основі технології блокчейн

Вдосконалений алгоритм розробки безпечних децентралізованих додатків

1. Код смарт-контракта завантажується на Ефіріум платформу
2. Смарт-контракт повинен мати змогу управляти крипто-токенами, таким чином учасники можуть йому посилати свої токени контракту. До настання певної події, смарт контракт приймає транзакції та токени учасників.
3. Коли настає подія, смарт-контракт починає розподіл крипто-токенів. Отриманий результат зберігається в словнику refund. Якщо для учасника у словнику були вже токени, до них додається нове значення.
4. Контракт дозволяє учасникам отримати інформацію про стан свого refund значення.
5. Учасники роблять запити на отримання свої токенів. Смарт контракт відправляє токени та обнуляє значення словника refund для учасника.

Головним в алгоритмі розподілу крипто-токенів є перекладення відповідальності за вивід токенів із програми контракту на учасників мережі

.

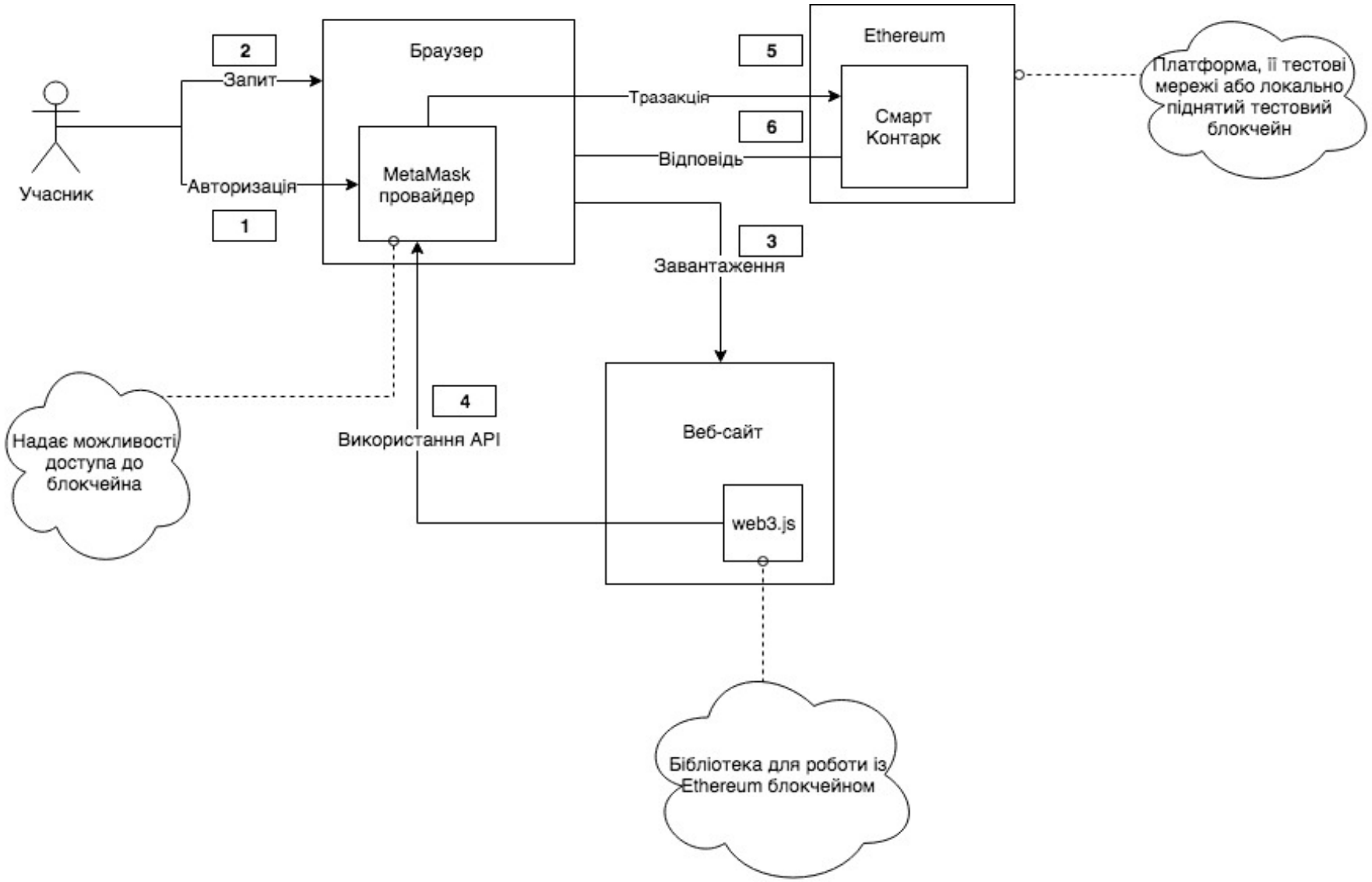


Асоціативний словник обліку розподілів криптотокенів

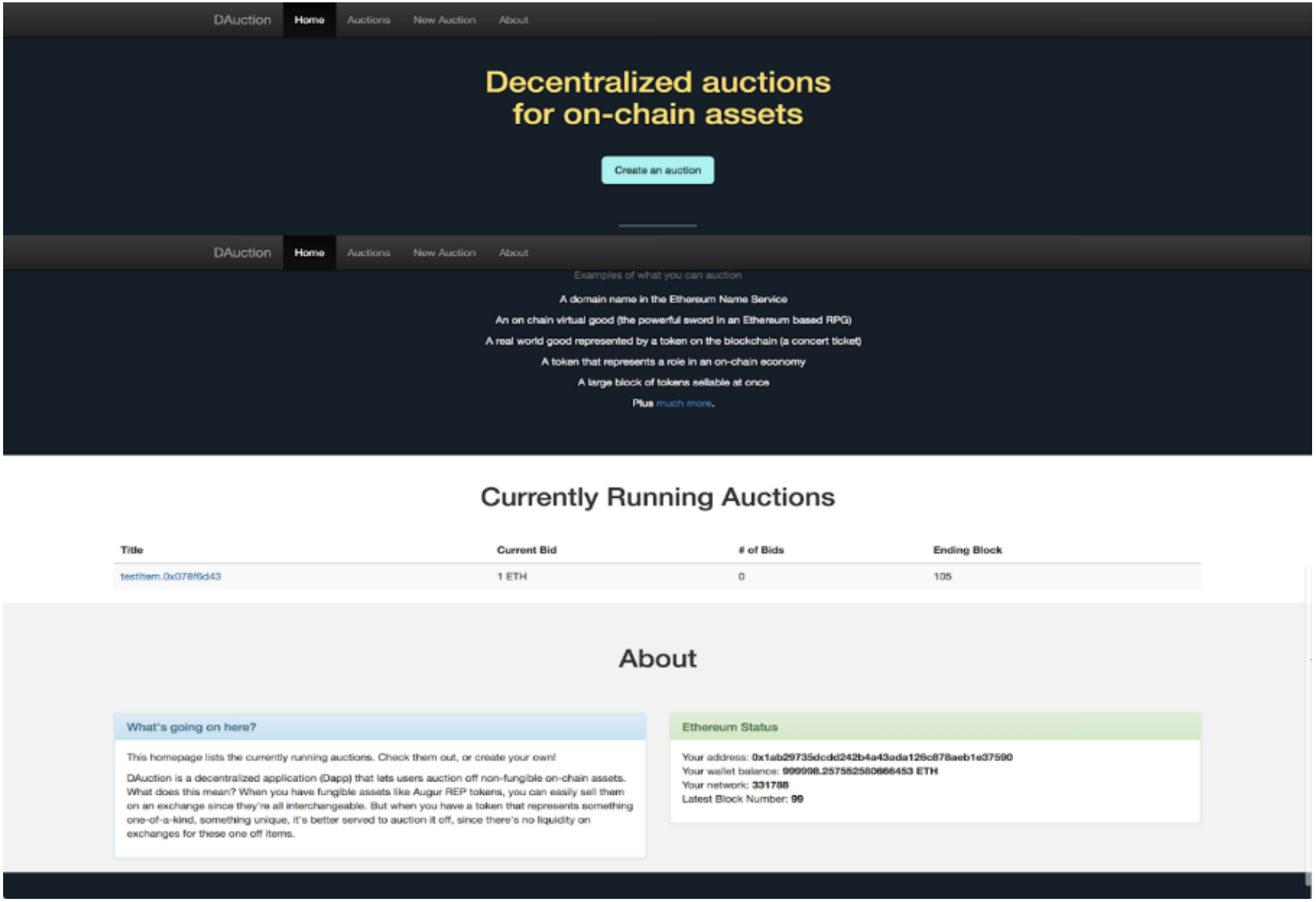
User Address	Amount of tokens to withdraw
0x1aB29735dCDd242B4a43aDa126c878AEB1E37590	5 ETH
0x078F6D43909Ff1F7279164cC013Ae7B0d118028a	1 ETH
…	
0xd60e75848F7c0312503D39761F549826E80B761d	105 ETH

З точки зору структур данних, алгоритм розподілу крипто-токенів використовує ассоціативні таблиці або словники. Словник містить інформацію про адресу користувача та суму розподілених токенів, які йому належать. Інформація про адресу користувача унікальна та може бути взята із запиту на перевірку стану крипто-токенів.

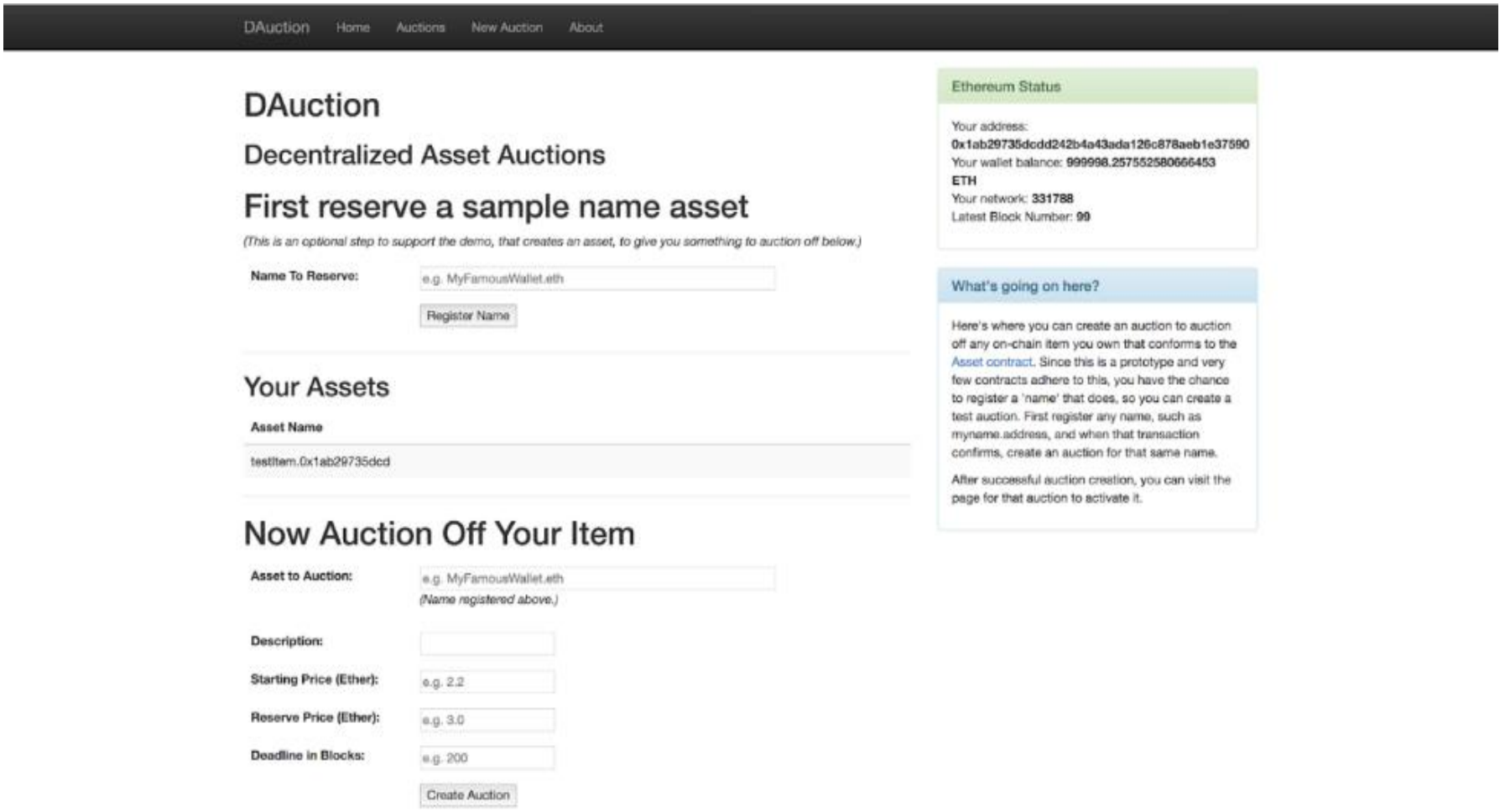
Архітектура децентралізованого додатку



Головна сторінка вебдодатку



Сторінку створення нового аукціону



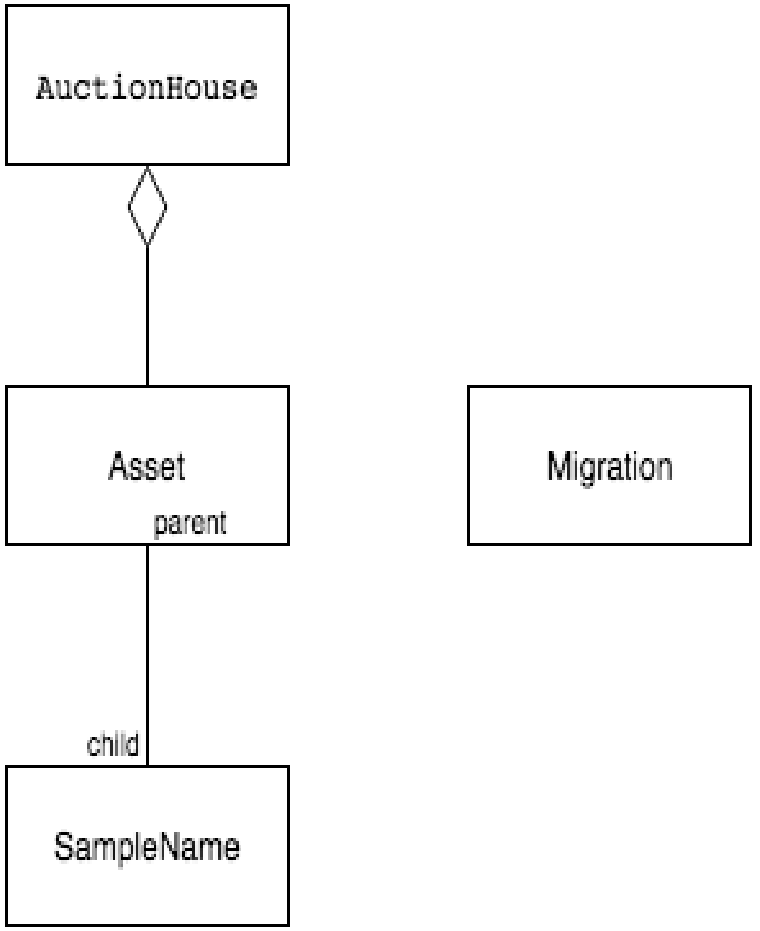
Структура та залежність між смарт-контрактами програми

AuctionHouse. - є відповідальним за правила роботи аукціона та розподіл крипто-токенів,надає інтерфейс для створення нових лотів, можливість відправлення ставок від учасників мережі та визначення переможця, релізу удосконалено алгоритм розподілу крипто-токенів

Asset - абстрактним представленням віртуальної власності на блокчейні

SampleName - приклад реалізації простої власності

Migration - системним смарт-контракт, що створюється фреймворком Truffle для зручного завантаження контрактів та зберігання системної інформації про збережені контракти на блокчейні



ВИСНОВКИ

У ході виконання кваліфікаційної роботи було отримано такі результати:

- проведено аналіз методів та засобів створення децентралізованих програм;
- удосконалено модель функціонування смарт-контрактів для покращення стабільності роботи;
- вдосконалено метод розподілу крипто-токенів у децентралізованих програмах на платформі Ethereum;
- розроблено алгоритм та програмне забезпечення для платформи Ethereum, яке реалізує розроблений метод.
- було створено прототип децентралізованого додатку для організації аукціонних торгів
- досліджено ефективність розробленого алгоритму та засобів його реалізації.

ДОДАТОК Б

(обов'язковий)

Копія публікацій

к.т.н., доц. Муляр І.В. *(ХмНУ),*
Віхтюк А.Р. *(ХмНУ),*
Пічура В.Ю. *(ХмНУ)*

АНАЛІЗ ПІДХОДІВ ДО ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Здобутки блокчейн базуються на руйнуванні організаційної парадигми систем централізованого зберігання і обробки даних.

Організація і функціонування централізованих програмних систем відрізняється простотою, їх розробка є формалізованою і базується на відомих методах та алгоритмах, тому вони є популярними. Прикладами таких систем є більшість веб-додатків. Під централізованою їх організацією мається на увазі, що вебдодаток має один або декілька серверів, які є центрами і серцем системи. На серверах запущено вебсервер додатку та виконується його програмний код. Всі взаємодії із системою є централізованими і орієнтованими на використання цих серверів. Стабільність такої системи безпосередньо залежить від провайдера серверів.

Розміщення всіх даних на локальних серверах і використання ресурсів одного провайдера зумовлюють ненадійність централізованих додатків і ризики втрати конфіденційності інформації [1].

Однією із основних характеристик, яка зумовила поширення блокчейн, є надійність. Надійність блокчейн забезпечується ланцюговим хешуванням інформаційних блоків, розподіленим зберіганням даних з багатократним їх резервуванням, використанням алгоритму цифрового підпису з асиметричним шифруванням на основі еліптичної кривої (ECDSA) для перевірки автентичності транзакцій.

Популярності технології блокчейн додає можливість збереження анонімності у виконанні операцій, відсутність потреб у послугах банківських та інших фінансових установ, висока продуктивність систем надання послуг.

Розвиток методів створення децентралізованих додатків призвів до розвитку альтернативних підходів до проектування програмних систем на блокчейн і до появи принципово нових програмних продуктів. Це було зумовлено децентралізованою природою нових програм та технологіями, які використовуються для зберігання даних. Такі функції, як розумний контакт проекту Ethereum, сприяли створенню великих децентратлізованих додатків [2]. Принцип децентралізації як основа технології блокчейн – це актуальна задача, вирішення якої робить можливою створення ефективних і надійних програмних додатків, але використання цієї можливості залежить від

способу розробки децентралізованих додатків

ЛІТЕРАТУРА:

1. David López and Bilal Farooq, (2019). A multi-layered blockchain framework for smart mobility data-markets. Transportation Research Part C: Emerging Technologies. 111. 10.1016/j.trc.2020.01.002

2. Safe Decentralized Applications Development Using Blockchain Technologies / Viktor Cheshun, Ihor Muliar, Vasyl Yatskiv, Ruslan Shevchuk, Serhii Kulyna, Taras Tsavolyk // 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), 16-18 Sept. 2020, Deggendorf, Germany. – Publisher: IEEE, 2020. – P. 800-805.

УДК 371.64:378.14:004

Ігор МУЛЯР
Хмельницький національний університет
ORCID http://orcid.org/0000-0002-6659-605X
muliariv@khmnu.edu.ua
Андрій ВІХТЮК
Хмельницький національний університет
e-mail: andrii.pain@gmail.com
Віталій ПІСТОЛЮК
Хмельницький національний університет
e-mail: kogic11@i.ua

ВИКОРИСТАННЯ СУЧАСНИХ ДЕЦЕНТРАЛІЗОВАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗМЕЖУВАННЯ ДОСТУПУ В ХМАРНОМУ СЕРЕДОВИЩІ

В роботі вирішено актуальну науково-технічну задачув із розробкаи методу розмежування доступу до сервісів хмарного середовища, за рахунок динамічно сформованих правил фільтрації для віртуальних брандмауерів. Запропонована в роботі модель враховує динамічний характер розподілу виділених ресурсів і характеристики протоколів мережевої взаємодії. На вхід моделі надходить потік мережних пакетів, що в режимі реального часу надходять на фаєрвол системи захисту в хмарному середовищі. Модель здійснює розділення пакети на віртуальні з'єднання, та визначає підмножини правил фільтрації для всіх інформаційних з'єднань, що дозволяють фільтрувати мережеву взаємодію для дотримання політики доступу. Взаємодія віртуальних машин в рамках одного гіпервізора здійснюється без використання фізичних ліній зв'язку і забезпечується програмним методом, наприклад за рахунок використання смарт-контрактів.

Ключові слова: моделі, алгоритми, хмарне середовище, блокчейн.

Igor MULIAR,
Andrii VIKHTIUK, Vitalii PISTOLIUK
Khmelnytsky National University

USE OF MODERN DECENTRALIZED TECHNOLOGIES FOR DISTRIBUTION OF ACCESS IN THE CLOUD ENVIRONMENT

Abstract. The work solves the actual scientific and technical problem of developing a method of demarcating access to cloud services using dynamically generated filtering rules for virtual firewalls. The model proposed in the work takes into account the dynamic nature of the allocation of allocated resources and the characteristics of network interaction protocols. The input of the model receives a stream of network packets that are sent to the firewall of the protection system in the cloud environment in real time. The model divides packets into virtual connections, and defines subsets of filtering rules for all information connections that allow filtering network interaction to comply with access policie.

The integration of access control functions into the components of the cloud environment reduces its performance, provided that the firewalls that control information interaction use the hardware resources of the regular hypervisor. The virtual connection classification algorithm proposed in the work uses the existing technologies of parallel computing and the structure of the TCP/IP stack, and is implemented using the Netgraph network subsystem. This makes it possible to increase the performance of firewalls and more efficiently use the computing power of existing hardware platforms. This reduces the cost of access delimitation tools in the cloud environment. The developed algorithms and method expand the possibilities of using the technology of inter-network shielding. The interaction of virtual machines within the framework of one hypervisor is carried out without the use of physical communication lines and is ensured by a software method, for example through the use of smart contracts.

Keywords: models, algorithms, cloud environment, blockchain

Постановка проблеми у загальному вигляді
та її зв'язок із важливими науковими чи практичними завданнями

Сьогодні до хмарних технологій та реалізації на їх основі середовища хмарних обчислень проявляється великий інтерес, а технологічно розвинені корпорації тадержави їх уже реалізували та широко застосовують. В Україні Доктрина інформаційної безпеки визначає поняття інформаційної сфери як сукупність інформаційної інфраструктури, інформації, суб'єктів, які збирають, формують, поширюють та використовують інформацію, а також систему регулювання суспільних відносин, що при цьому виникають [1].

Інформаційна безпека в широкому розумінні - це такий стан об'єкта захисту, який виключає можливість пошкодження властивостей об'єкта внаслідок його взаємодії з інформаційною сферою.

Актуальною завдачою розмежування доступу є формалізація вимог щодо обмеження доступу до інформаційних сервісів у середовищі хмарних обчислень, які можуть бути представлені за допомогою динамічно сформованого набору правил фільтрації, які забезпечують відповідність вимогам політиці безпеки.

Метою дослідження є вдосконалення методу формування правил фільтрації для розмежування доступу в хмарному середовищі. Ряд публікацій присвячений аналізу існуючих загроз, побудові моделей загроз та зловмисника у хмарі. Так, у роботі [2] він розглядається питання довіри до провайдера, безпеки ключів, управління ризиками, безпеки хмарної архітектури, контролю доступу, захисту даних та ізоляції програм. Ця робота отримала подальший розвиток у вигляді рекомендацій NIST SP 800-144 [3]

Постановка задачі

Актуальною науково-технічною задачею в роботі є розробка методу розмежування доступу до сервісів хмарного середовища, за рахунок динамічно сформованих правил фільтрації для віртуальних брандмауерів. Для досягнення поставленої мети в роботі потрібно вирішити наступні завдання:

- розробити модель інформаційної взаємодії з врахуванням розмежування доступу в середовищі хмарних обчислень;
- вдосконалити метод динамічного формування правил фільтрації що враховує параметри віртуальних з'єднань;
- розглянути можливість використання технології блокчейн для побудови системи розмежування доступу.

Основна частина

Віртуалізація використовується в усіх хмарних середовищах. Віртуалізація - це структура або методологія поділу ресурсів одного фізичного сервера на кілька середовищ виконання шляхом застосування однієї або кількох концепцій або технологій, таких як поділ апаратного та програмного забезпечення, поділ часу, часткове або повне машинне моделювання та емуляція [4].

Віртуальну машину в неактивному стані можна визначити як набір образів жорсткого диска та метаданих, що описують конфігурацію віртуальної машини. Метадані містять інформацію про обсяг пам'яті віртуальної машини, кількість обчислювальних ядер або процесорів, фізичних мережевих інтерфейсів та інших периферійних пристроїв введення/виведення [5]. При цьому запущена віртуальна машина також має атрибути використовуваного гіпервізора, унікальний ідентифікатор і налаштування мережевого інтерфейсу. Взаємодія віртуальних машин, код яких виконується в ізольованих доменах, здійснюється по мережі. У зв'язку з цим контроль мережевої взаємодії в середовищі хмарних обчислень займає перше місце. Для функціонування мережевої підсистеми гіпервізор надає віртуальним машинам функціональність програмного

мережевого мосту, який називається віртуальним комутатором, який є програмним компонентом ОС гіпервізора. При цьому, якщо необхідно забезпечити зв'язок віртуальних машин із зовнішнім світом, фізичний інтерфейс гіпервізора, який знаходиться під управлінням ОС гіпервізора, також підключається до програмного мосту. При цьому для розмежування доступу і побудови програмних компонентів гіпервізора (віртуального фаєрвола) можна використати популярну технологію блокчейн. Таким чином, взаємодія віртуальних машин в рамках одного гіпервізора здійснюється без використання фізичних ліній зв'язку і забезпечується програмним методом, наприклад за рахунок використання смарт-контрактів [6]. Відповідно, засобами контролю такого трафіку також може бути децентралізований програмний комплекс, що працює в рамках гіпервізора, або апаратно інтегрована з програмною реалізацією міжмережевого екрану.

На рисунку 1 наведено модель середовища хмарних обчислень.

Рисунок 1 - Модель хмарного середовища

Розмежування доступу як служба інформаційної безпеки нерозривно пов'язано з політикою доступу, визначає повноваження суб'єктів доступу до об'єктів і механізм, які реалізують цю політику.

Розглянемо інформаційну взаємодію хмарному середовищі. Згідно [7] для опису розмежування доступу та основних процесів, які при цьому виникають будемо використовувати поняття суб'єкта s та об'єкта o. Суб'єкт є ініціатором взаємодії, активною сутністю, і здійснює дію по відношенню до об'єкта o_2, при цьому відбувається обмін інформацією по комп'ютерній мережі.

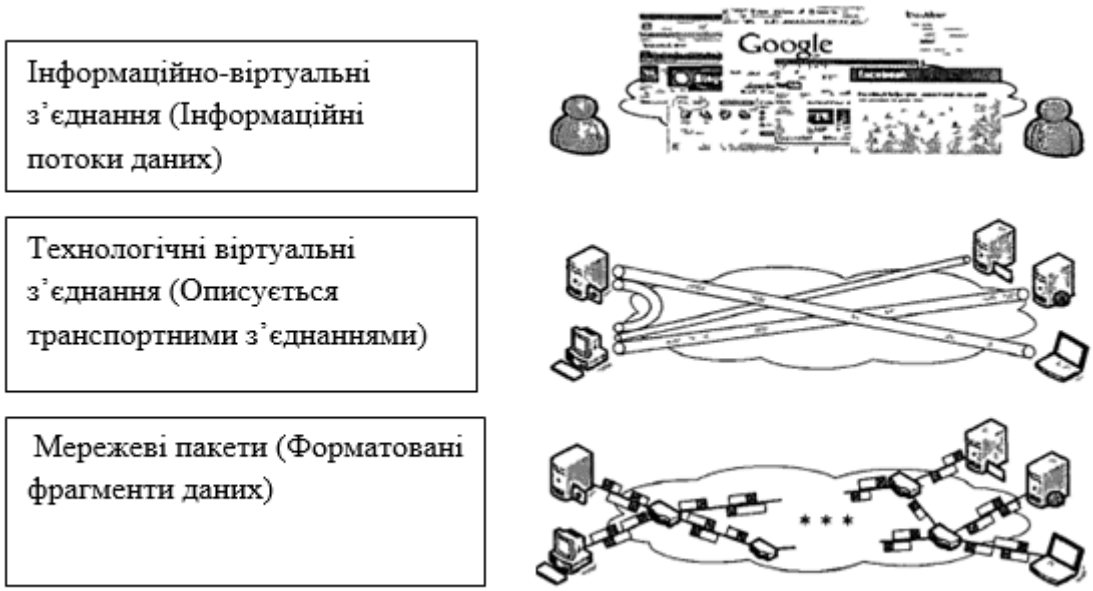


Рисунок 2 – Представлення інформаційної взаємодії в формі віртуального з'єднання

Для опису процесів розмежування доступу будемо використовувати рольову модель [8]. Фаєрвол формує дозвіл або заборону інформаційного віртуального з'єднання.

Наприклад, привілей може бути задано: {user:Vetal, [{transport:"TCP", port:"8080", protocol: HTTP ext:[{method:"POST"}]}]}. Таким чином формується превілей доступу до ресурсу по протоколу http з використанням методу POST по 8080 порту до віртуальних машин, що належать користувачу Vetal.

Інформаційну взаємодію в хмарному середовищі можна уявити у вигляді двох частини: підмережі віртуальних машин N_{vm}, та підмережі керування N_{man},

Кожній підмережі призначається множина IP-адрес, які можуть бути надані віртуальним машинам чи серверам.

Правила фільтрації задаються наступним чином:

$$Rul_{man} - \{rul_{mani}\}, i = \overline{1..n} \quad (1)$$

Суб'єкт та об'єкт обмінюються інформацією за допомогою моделі технологічного з'єднання та працюють з привілеями користувачів хмари, які здійснили запуск віртуальних машин [8].

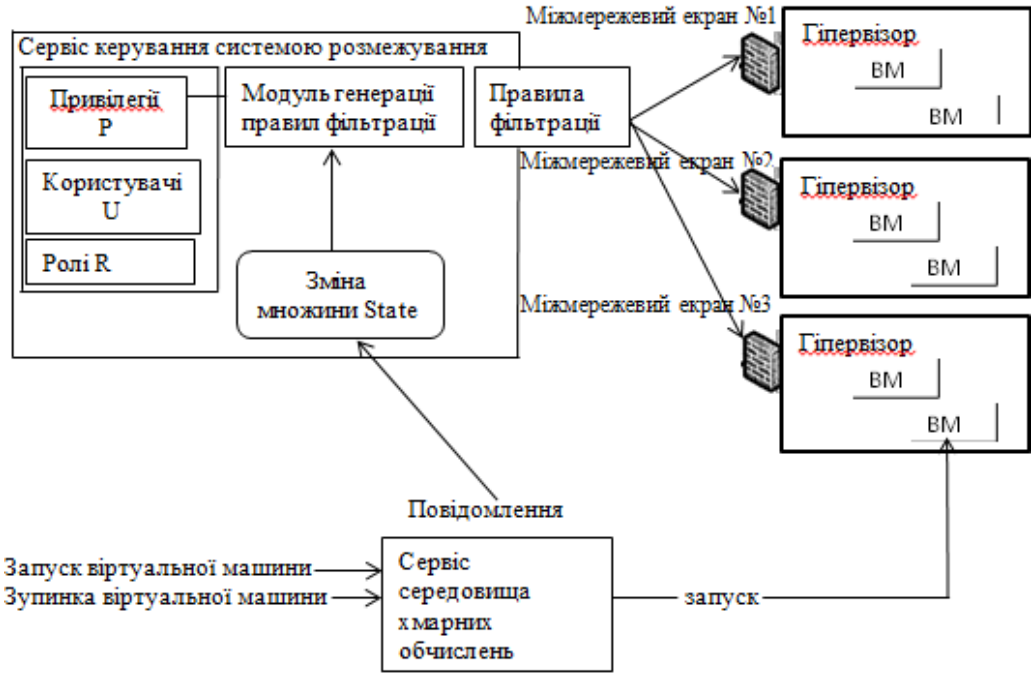


Рисунок 3 - Адаптивний метод конфігурації ПФ

Розглянемо можливість використання технології блокчейн для побудови системи розмежування доступу. Розподілена хеш-таблиця (словник) складається на кожному вузлі і містить відображення хеш-функцій вмісту ресурсу (унікальних ідентифікаторів та самого ресурсу). Побудову децентралізованого застосунку, децентралізований програмний комплекс, що працює в рамках гіпервізора з програмною реалізацією міжмережевого екрану доречно виконати, застосовуючи технологію блокчейна на платформі Ethereum [9]. Важливим аспектом роботи Ethereum є те, що будь-яка операція, яка виконується мережею, також одночасно виконується кожним повним вузлом [10]. Однак усі етапи обчислення на віртуальній машині Ethereum надто дорогі. Таким чином, для вирішення простих завдань (наприклад, перевірки підписів, а також інших операцій, пов'язаних з криптовалютою), смарт-контракти Ethereum можуть цілком підійти, на відміну від тих випадків, коли потрібні більш складні завдання, наприклад виконувати машиннео навчання, яке можуть спричинити надмірне використання мережі. Введення оплати запобігає діям користувача, спрямованим на зайве навантаження на мережу.

Стандартний підхід реалізований на платфомі Ethereum працює у цьому випадку нестабільно. Він полягає у тому, щоб при наступанні події P_n перерозподіляти крипто-токени та передавати їх усі разом через транзакцію CT_n Якщо виникне помилка транзакція буде скасована і може статись так, що стан контракту C залишиться незмінним. Це призведе до проблем із перерозподілом крипто-токенів у мережі M.

Альтернативний варіант, запропонований у роботі потребує керування розподілом крипто-токенів у ситуаціях, коли виникає помилка на етапівиконання транзакції CT_n і потрібно забезпечити можливість учасникам транзакції отримати свої крипто-токени рис 4.

Нехай взаємодія відбувається між смарт-контрактом C та учасниками $A_1, A_2,\ldots, A_n$. Учасники виконують транзакції $T_1, T_2,\ldots, T_n$, після яких стан розумного контракту C змінюється на проміжні стани C', C'',…, $C^{()n}$. Стани учасників відповідно змінюються на A_1', A_2',…, A_n'. В той момент, коли в мережі M настає подія P_1, контракт C змінює свій стан на $C^{()(n+1)}$. При цьому смарт-контракт C розподіляє крипто-токени всередині свого стану на відповідні рахунки учасників A_1', A_2',…, A_n'. На цьому відбувається закунчення ефекту події P_1. Учасники A_1', A_2',…, A_n' мають можливість перевірити стан своїх акаунтів в смарт-контракті C та при потребі виконати зняття коштів через транзакції CT_{11}, CT_{12},…, CT_{1n}. У випадку, якщо під час транзакції CT_1 виникне помилка, то учасник A_i зможе повторити транзакцію, але не може завадити іншим учасникам отримати свої крипто-токени.

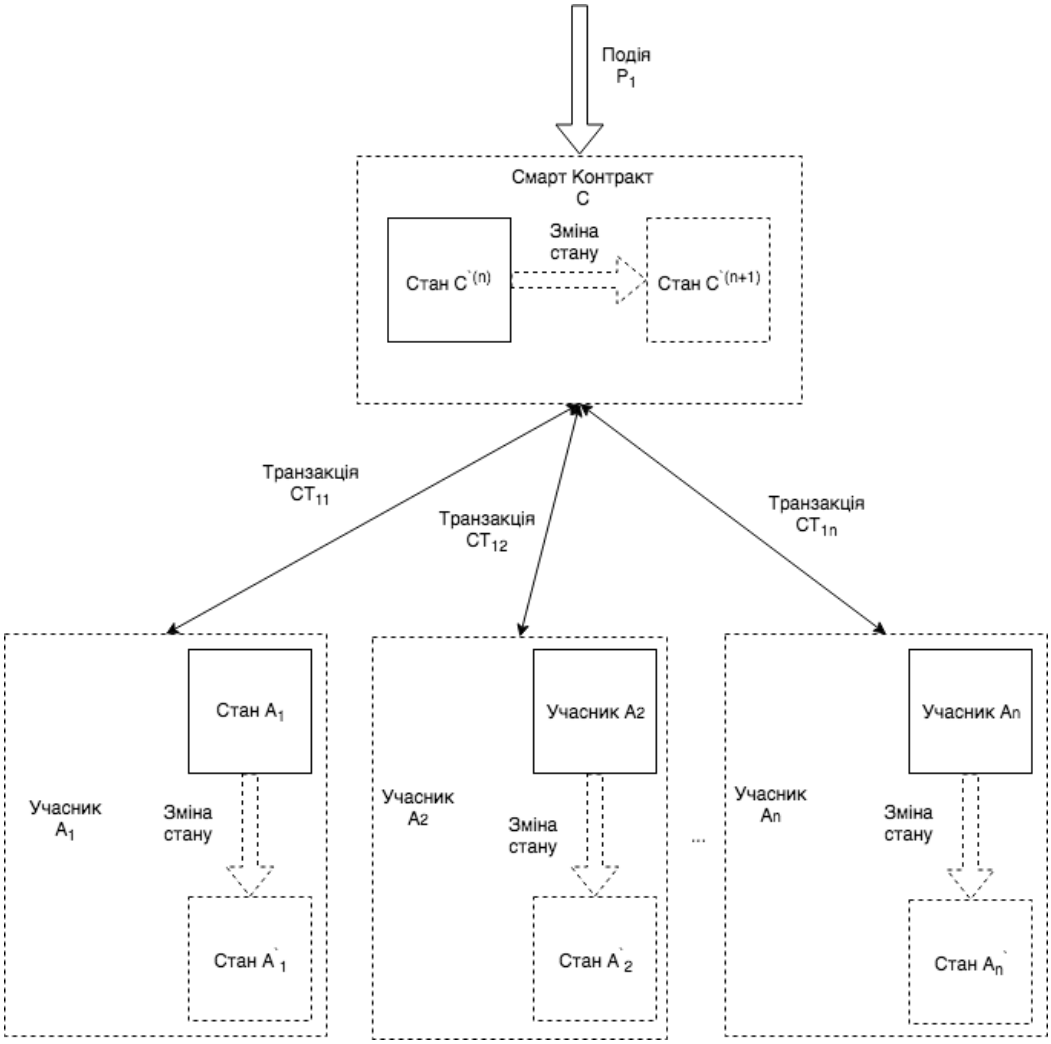


Рисунок 4 – Запропонований підхід до функціонування розумних контрактів

Запропонована модель може використовуватися для розподіленого виведення крипто-токенів, та дозволяє організувати процес розмежування доступу в хмарному середовищі, коли атакуючий намагається виконати транзакції T_i через власний смарт-контракт, який запрограмований на створення помилки для транзакції CT_i. Зловмисник не має можливості глобально впливати на роботу смарт-контракту C та на його стабільність для інших учасників системи блокчейн.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

На підставі опису загроз стану безпеки ресурсів хмарних обчислювальних середовищ та методів боротьби з ними виявляється, що використання стандартних підходів не дозволяє вирішити проблему підвищення рівня безпеки хмарних обчислювальних середовищ. Отже, для створення інформаційної безпеки в середовищі хмарних обчислень необхідно розробити нові методи, алгоритми, програмні продукти, які дозволять запобігти проникненню різноманітних загроз або швидко виявляти та нейтралізувати їх. Мережу блокчейн представлено системою, яка може змінювати свій стан. Така система може складатися з двох компонентів: стану системи та функції, яка її модифікує. Стан системи – це право власності на всі криптовалюти в системі. Системна функція приймає стан системи та транзакцію як аргументи. В результаті функціонування (виконання транзакцій) отримано новий стан системи, в якому відповідно зміняться права власності на криптовалюту, яка може бути використана в правилах розмежування доступу до ресурсів хмарного середовища.

Літертура

1 Про схвалення Стратегії розвитку сфери інноваційної діяльності на період до 2030 року [Електронний ресурс]. - Режим доступу : https://zakon.rada.gov.ua/laws/show/526-2019-%D1%80#Text. - Назва з екрана.

2 NIST Cloud Computing Synopsis and Recommendations, SP 800-146 [Електронний ресурс]. Режим доступу: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST SP800146.pdf

3 Остапов С. Е. Технології захисту інформації: навчальний посібник / С.Е. Остапов, С.П. Євсеєв, О.Г. Король–Харків : Вид-во ХНЕУ, 2016. – 476 с.

4 Safe Decentralized Applications Development Using Blockchain Technologies / Viktor Cheshun, Ihor Muliar, Vasyl Yatskiv, Ruslan Shevchuk, Serhii Kulyna, Taras Tsavolyk // 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), 16-18 Sept. 2020, Deggendorf, Germany. – Publisher: IEEE, 2020. – P. 800-805.

5 Пістолюк В.О. Аналіз процесу обміну інформацією в середовищі хмарних обчислень / А.В. Джулій, Ю.П. Кльоц, І.В. Толок, О.С. Лєнков, В.О. Пістолюк // Тези доповідей XVIII Міжнародної науково-практичної конференції "Військова освіта і наука: сьогодення та майбутнє" [Текст] – К. : ВІКНУ, 2022. – 30 с.

6 Лавров, Є. А. Математичні методи дослідження операцій : підручник / Є. А. Лавров, Л. П. Перхун, В. В. Шендрик – Суми : Сумський державний університет, 2017. – 212 с.

7 Довгий, С.О. Сучасні телекомунікації: мережі, технології, економіка, управління, регулювання / С.О. Довгий, О.Я. Савченко, П.П. Воробієнко – К.: Український Видатничій Центр, 2012. – 520 с.

8 David Farooq, (2019). A multi-layered blockchain framework for the smart mobility data-markets. Transportation Research Part C: Emerging Technologies. 111. 10.1016/j.trc.2020.01.002.

9 V. Yatskiv, N. Yatskiv, and O. Bandrivskyi “Proof of Video Integrity Based on Blockchain,” in Proc. Advanced Computer Information Technologies (ACIT), 2019 IEEE 9th International Conference on, 2019, pp. 431-434.

10 M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “A multi-step outlier-based anomaly detection approach to network-wide traffic,” Inf. Sci. (Ny)., vol. 348, pp. 243–271, 2016

11 Джулій, В.М., Кльоц Ю.П., Муляр І.В., Жилевич М.Л., Джулій А.В. Контроль додатків інтернет-трафіка комп’ютерних мереж методами машинного навчання. Вісник Хмельницького національного університету. Технічні науки. 2021. № 5. С. 22-26.

References

1. Pro skhvalennia Stratehii rozvytku sfery innovatsiinoi diialnosti na period do 2030 roku [Elektronnyi resurs]. - Rezhym dostupu : https://zakon.rada.gov.ua/laws/show/526-2019-%D1%80#Text. - Nazva z ekrana.

2. NIST Cloud Computing Synopsis and Recommendations, SP 800-146 [Elektronnyi resurs]. Rezhym dostupu: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST SP800146.pdf

3. Ostapov S. E. (2016) Tekhnolohii zakhystu informatsii: navchalnyi posibnyk / S.E. Ostapov, S.P. Yevseiev, O.H. Korol–Kharkiv : Vyd-vo KhNEU, 2016. – 476 s.

4. Viktor Cheshun, Ihor Muliar, Vasyl Yatskiv, Ruslan Shevchuk, Serhii Kulyna, Taras Tsavolyk (2020), “Safe Decentralized Applications Development Using Blockchain Technologies”, in Proc. Advanced Computer Information Technologies (ACIT), 2019 IEEE 9th International Conference on, 2020. – pp. 800-805.

5. Pistoliuk V.O. (2022) Analiz protsesu obminu informatsiieiu v seredovyshchi khmarnykh obchyslen / A.V. Dzhulii, Yu.P. Klots, I.V. Tolok, O.S. Lienkov, V.O. Pistoliuk // Tezy dopovidei KhVIII Mizhnarodnoi naukovo-praktychnoi konferentsii "Viiskova osvita i nauka: sohodennia ta maibutnie" [Tekst] – K. : VIKNU, 2022. – 30 s.Lavrov, Ye. A. (2017.), Matematychni metody doslidzhennia operatsii : pidruchnyk / Ye. A. Lavrov, L. P. Perkhun, V. V. Shendryk – Sumy : Sumskyi derzhavnyi universytet, – 212 p.

6. Dovhyi, S.O. (2012), Suchasni telekomunikatsii: merezhi, tekhnolohii, ekonomika, upravlinnia, rehuliuvannia /S.O. Dovhyi, O.I. Savchenko, P.P. Vorobiienko – K.: Ukrainskyi Vydatnychii Tsentr. – 520p.

7. David Farooq, (2019). A multi-layered blockchain framework for the smart mobility data-markets. Transportation Research Part C: Emerging Technologies. 111. 10.1016/j.trc.2020.01.002.

8. V. Yatskiv, N. Yatskiv, and O. Bandrivskyi (2019), "Proof of Video Integrity Based on Blockchain," in Proc. Advanced Computer Information Technologies (ACIT), 2019 IEEE 9th International Conference on, 2019, pp. 431-434.

9. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "A multi-step outlier-based anomaly detection approach to network-wide traffic," Inf. Sci. (Ny)., vol. 348, pp. 243–271, 2016

10. Dzhulii V.M., Klots Yu.P., Muliar I.V., Zhylevych M.L., Dzhulii A.V. (2021), Kontrol dodatkiv internet-trafika kompiuternykh merezh metodamy mashynnoho navchannia. Visnyk Khmelnytskoho natsionalnoho universytetu. Tekhnichni nauky. – Khmelnytskyi. – №5. – pp. 22–26.

11. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, (2016), "A multi-step outlier-based anomaly detection approach to network-wide traffic," Inf. Sci. (Ny)., vol. 348, pp. 243–271, 2016

ДОДАТОК В

(обов'язковий)

Фрагмент програмного коду

src/app/scripts/component/account-manager.js

```
var accountManager = {

   account: null,

   getAccount: function () {
      return this.account;
   },
   setAccount: function (account) {
      this.account = account;
   }

}

export default function () {
   return accountManager;
}
```

src/app/scripts/component/ethereum-manager.js

```
var ethereumManager = {
   getContractAbi: (contractAbi) => {
      return web3.eth.contract(contractAbi)
   },
   fetchTransactionReceipt: (transactionId, callback) => {
      web3.eth.getTransactionReceipt(transactionId, callback)
   },
   fetchAccountAddress: (callback) => {
      web3.eth.getAccounts(function (error, addresses) {
         if (error != null) {
            alert('There was an error fetching your accounts.')
            return
         }
         callback(addresses)
      })
   },

   fetchCurrentBalanceInEth: (account, callback) => {
      web3.eth.getBalance(account, function (error, balance) {
         var balanceInEth = web3.fromWei(balance, 'ether')
         callback(error, balanceInEth)
      })
   },
   fetchCurrentBlockNumber: (callback) => {
      web3.eth.getBlockNumber(callback)
   },
   fetchNetworkId: function (callback) {
      web3.version.getNetwork(callback)
   },
   /**
    * @param {Function} callback
    */
   watchLatestBlock: (callback) => {
      const latestBlockFilter = web3.eth.filter('latest')
```

```
      latestBlockFilter.watch(callback)
    }
}
export default function () {
    return ethereumManager;}
```

src/app/scripts/component/status-manager.js

```
import ko from 'ko';
let statusManager = {
    message: ko.observable(''),
    statusLabel: ko.observable(''),
    statusType: ko.observable('success'),
    setStatus: function (label, type) {
        this.statusLabel(label)
        this.statusType(type)
    },

    hideSpinner: function () {
        $('#spinner').hide();
    },

    showSpinner: function () {
        $('#spinner').show();
    }
}
export default function () {
    return statusManager;
}
```

src/app/scripts/component/web3-loader.js

```
import { default as Web3 } from 'web3'
export default {
    load: function () {
        if (typeof web3 !== 'undefined') {

            // Use Mist/MetaMask's provider
            window.web3 = new Web3(web3.currentProvider)
        } else {
            console.warn(
                'No web3 detected. Falling back to http://127.0.0.1:8545.' +
                ' You should remove this fallback when you deploy live, as it\'s inherently insecure.' +
                ' Consider switching to Metamask for development.' +
                ' More info here: http://truffleframework.com/tutorials/truffle-and-metamask'
            )
            // fallback - use your fallback strategy (local node / hosted node + in-dapp id mgmt / fail)
            window.web3 = new Web3(new Web3.providers.HttpProvider('http://127.0.0.1:8545'))
        }
    }
}
```

src/app/scripts/data/auction-marketer-data.js

```
export default {
  cutOff: 5,
  address: '0xd60e75848F7c0312503D39761F549826E80B761d'
}
```

src/app/scripts/data/contract-address-data.js

```
export default {
  sampleName: '0xd38c8a1986afbe997a9fef2f01aa652ec711f71a',
  auctionHouse: '0x67a865e6c4674167184c38bbd3917c37a28c8dcb'
}
```

src/app/scripts/mapper/auction-mapper.js

```
export default {

    /**
```

```
    * Map Auction raw data to object
    *
    * @param {Array} auctionData
    *
    * @returns {{seller: *, contractAddress: *, recordId: *, title: *, description: *, blockNumberOfDeadline: *,
startingPrice: *, reservePrice: *, currentBid: *, bidCount: *}}
    */
   map: (auctionData) => {
      return {
         seller: auctionData[0],
         contractAddress: auctionData[1],
         recordId: auctionData[2],
         title: auctionData[3],
         description: auctionData[4],
         blockNumberOfDeadline: auctionData[5],
         startingPrice: auctionData[8],
         reservePrice: auctionData[9],
         currentBid: auctionData[10],
         bidCount: auctionData[11],
      }

   }
}
src/app/scripts/service/auction/close-auction-service.js
import auctionHouseContract from "../../smart-contract/auction-house-contract";
import statusManager from '../../component/status-manager'
export default function() {
   return {
      /**
       * @return {void}
       */
      execute: (account, auction) => {
         statusManager().setStatus('Closing auction...', 'warning')
         statusManager().showSpinner()

         auctionHouseContract().use((instance) => {
            return instance.endAuction(auction.auctionId, { from: account, gas: 1400000 })
         }).then((transactionData) => {
            console.log(`End Auction txnId: ${transactionData.tx}`)
            console.log(transactionData)
            statusManager().setStatus('Auction has been closed successfully.', 'success')
            statusManager().hideSpinner()
         }).catch((e) => {
            console.log(e)
            statusManager().setStatus(e, 'error')
            statusManager().hideSpinner()
         })
      }

   }
}
src/app/scripts/service/auction/open-auction-service.js
import auctionHouseContract from "../../smart-contract/auction-house-contract";
import statusManager from '../../component/status-manager'
import sampleNameContract from "../../smart-contract/sample-name-contract";
import contractAddressData from "../../data/contract-address-data";
export default function() {
   return {

      /**
       * @return {void}
       */
      execute: function (account, auction) {
```

```
            if (auction.seller !== account) {
                statusManager().setStatus('Only seller can activate auction.', 'error')
                return
            }
            statusManager().setStatus('Transferring ownership to the contract...', 'warning')
            statusManager().showSpinner()
            const recordId = auction.recordId
            sampleNameContract().use((sampleNameInstance) => {
                return sampleNameInstance.owner.call(recordId)
            }).then((ownerAddress) => {

                if (ownerAddress == contractAddressData.auctionHouse) {
                    // Asset is already owned by the contract
                    this.openAuction(account, auction)
                    return;
                }
                sampleNameContract().use((sampleNameInstance) => {
                    return sampleNameInstance.setOwner(
                        recordId,
                        contractAddressData.auctionHouse,
                        { from: account, gas: 500000 }
                    );
                }).then((transactionData) => {
                    console.log(`Set Owner TxID: ${transactionData.tx}`)
                    statusManager().setStatus('Ownership transfer complete!', 'success')
                    statusManager().hideSpinner()

                    this.openAuction(account, auction)
                }).catch((e) => {
                    console.log(e)
                    statusManager().setStatus(e, 'error')
                    statusManager().hideSpinner()
                })
            }).catch((e) => {
                console.log(e)
                statusManager().setStatus(e, 'error')
                statusManager().hideSpinner()
            })
        },
        /**
         * Perform activation of the auction
         *
         * @return {void}
         */
        openAuction: function (account, auction) {
            // Activate the auction
            statusManager().setStatus('Activating auction...', 'warning')
            statusManager().showSpinner()

            auctionHouseContract().use((instance) => {
                return instance.activateAuction(auction.auctionId, { from: account, gas: 500000 });
            }).then((transactionData) => {
                console.log(transactionData)
                statusManager().setStatus('Auction has been activated!', 'success')
                statusManager().hideSpinner()
            });
        }
    }
}
```

src/app/scripts/updater/auction-user-updater.js

```
import ko from 'ko';
import auctionHouseContract from '../smart-contract/auction-house-contract'
```

```
import accountUpdater from '../updater/account-updater'
const auctionUserUpdater = {
    account: accountUpdater().account,
    userActionList: ko.observableArray([]),
    /**
     *
     */
    update: function () {
        setInterval(() => this.updateUserAuction(), 3000);
    },
    /**
     * @return {void}
     */
    updateUserAuction: function () {
        const account = this.account();
        auctionHouseContract().use((instance) => {
            return instance.getAuctionsCountForUser.call(account);
        }).then(
            (countBigNumber) => this.processUserAuctions(countBigNumber)
        ).catch((e) => {
            console.log(e)
        });
    },

    /**
     * @param countBigNumber
     */    processUserAuctions: function (countBigNumber) {
        const account = this.account();
        const count = parseInt(countBigNumber.valueOf())

        if (count === this.userActionList().length) {
            return
        }

        console.log(`${account} user has ${count} auction(s)`)
        for (let index = 0; index < count; index++) {
            auctionHouseContract().use((instance) => {
                console.log(`[${account}]loading #${index} auction`)
                return instance.getAuction.call(index);
            }).then(
                (auctionData) => {
                    auctionHouseContract().use((instance) => {
                        // getting global auction ID
                        return instance.getAuctionIdForUserAndIdx.call(account, index);
                    }).then((auctionId) => {
                        let underlyingUserAuctionList = this.userActionList()

                        underlyingUserAuctionList[index] = {
                            auctionId: auctionId,
                            seller: auctionData[0],
                            contractAddress: auctionData[1],
                            recordId: auctionData[2],
                            title: auctionData[3],
                            description: auctionData[4],
                            blockNumberOfDeadline: auctionData[5],
                            //distributionCut: auctionData[6],
                            //distributionAddress: auctionData[7],
                            startingPrice: auctionData[8],
                            reservePrice: auctionData[9],
                            currentBid: auctionData[10],
                            bidCount: auctionData[11],
                        };
```

```
                    this.userActionList.valueHasMutated()
                })
            }
        ).catch((e) => {
            console.log(e)
        });
    }
  }
}
export default function () {
  return auctionUserUpdater;
}
```

src/app/scripts/updater/eth-updater.js

```
import ko from 'ko'
import accountUpdater from '../updater/account-updater'
import auctionHouseContract from "../smart-contract/auction-house-contract";
import ethereumManager from "../component/ethereum-manager";
var ethUpdater = {
  account: accountUpdater().account,
  networkId: ko.observable(''),
  currentBlockNumber: ko.observable(''),
  walletBalance: ko.observable(''),
  withdrawBalance: ko.observable(0),
  /**
   * @return {void}
   */
  update: function () {
    this.account.subscribe((account) => this.handleAddressUpdate(account))
  },
  /**
   * @param {string} account
   */
  handleAddressUpdate: function (account) {
    this.updateCurrentBlockNumber()
    this.updateNetworkId()
    this.updateWalletBalance(account)
    this.updateWithdrawBalance(account)
  },
  /**
   * @param {string} account
   *
   * @return {void}
   */
  updateWalletBalance: function (account) {
    ethereumManager().fetchCurrentBalanceInEth(account, (error, walletBalance) => {
      this.walletBalance(walletBalance)
    })
  },
  /**
   * @return {void}
   */
  updateNetworkId: function () {
    ethereumManager().fetchNetworkId((error, networkId) => {
      this.networkId(networkId)
    })
  },
  /**
   * @return {void}
   */
  updateCurrentBlockNumber: function () {
    ethereumManager().fetchCurrentBlockNumber((error, blockNumber) => {
      this.currentBlockNumber(blockNumber)
      console.log(`Current Block Number: ${blockNumber}`)
```

```
        })
        ethereumManager().watchLatestBlock((error, blockNumber) => {
            this.currentBlockNumber(blockNumber)
            console.log(`Current Block Number: ${blockNumber}`)
        })
    },
    /**
     * @param {string} account
     *
     * @return {void}
     */
    updateWithdrawBalance: function (account) {
        auctionHouseContract().use((instance) => {
            return instance.getRefundValue.call({ from: account })
        }).then(
            (withdrawBalanceBigNumber) => {
                const withdrawBalanceEth = web3.fromWei(withdrawBalanceBigNumber.valueOf(), 'ether')
                this.withdrawBalance(withdrawBalanceEth)
            }
        ).catch((e) => {
            console.log(e)
        })
    }
}
export default function () {
    return ethUpdater;
}
src/app/scripts/updater/updater-manager.js
const updaterManager = {
    updaters: ko.observableArray([]),
    start: function () {
        this.updaters().forEach((updater) => {
            updater().update()
        })
    }
}
export default function (updaters = []) {
    updaterManager.updaters(updaters)
    return updaterManager;
}
```

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 16%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 109027 Название: Метод розподілу крипто-токенів для проєктування децентралізованих захищених інформаційних систем Добавлено в БД: 2022-12-06 Авторы: Віхтюк А.Р. Руководители: Савенко О.С. Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	70641	1118	2005 (3%)	36 (3%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1013209783

Дата перевірки:
06.12.2022 12:35:36 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.12.2022 14:22:11 EET

ID користувача:
100008300

Назва документа: **Магістерська_Віхтюк**

Кількість сторінок: **74** Кількість слів: **11981** Кількість символів: **92545** Розмір файлу: **2.35 MB** ID файлу: **1012971527**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.87%
Схожість

Найбільша схожість: **1.02%** з джерелом з Бібліотеки (ID файлу: **1012971520**)

1.68% Джерела з Інтернету	88	Сторінка 76
1.99% Джерела з Бібліотеки	77	Сторінка 77

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	4
Підозріле форматування	15 сторінок

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод розподілу крипто-токенів для проєктування децентралізованих захищених інформаційних систем

Автор: Віхтюк Андрій Русланович
Спеціальність: 123 – Комп'ютерна інженерія
Освітня програма: Програмування та захист комп`ютерних систем і мереж
Науковий керівник: Савенко Олег Станіславович, д.т.н, проф.
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 2.87% і адресується до 88 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Олег САВЕНКО

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

Дата: 06.12.2022

Завідувачу кафедри КбКСМ
д-р техн.наук, доцент. Кльоц Ю.П.

Віхтюк Андрій Русланович
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ1м-21-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.11.2022р.
дата

підпис

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Магістр Віхтюк А.Р.

Тема Метод розподілу крипто-токенів для проєктування децентралізованих захищених інформаційних систем

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «магістр»:

кількість листів креслень 9; кількість сторінок записки 73

1. Короткий зміст кваліфікаційної роботи та прийнятих рішень в рамках роботи запропоновано алгоритм реалізації децентралізованих захищених програм для розподілу крипто-токенів, який використовує удосконалений метод розподілу крипто-токенів, та визначає основі етапи розробки децентралізованих програм на основі технології Blockchain, які притаманні додаткам такого типу, на основі класичних підходів до розробки програм.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна магістерська робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі висвітлюється актуальність теми роботи, дається аналіз досліджуваної проблеми і обґрунтовується застосовуваний підхід до її вирішення, формулюються цілі і завдання дослідження, описується наукова новизна і практична значимість отриманих результатів. У першому розділі розглядаються питання особливостей застосування технологій децентралізованих програм. Наступні розділи присвячені розроблено та реалізовано алгоритм для розподілу крипто-токенів, який дозволяє покращити надійність та стійкість смарт контрактів та децентралізованих програм, які працюють на їхній основі

4. Позитивні сторони роботи Кваліфікаційна робота містить ряд інноваційних рішень, зокрема запропонований підхід полягає в тому, що смарт-контракт може перерозподіляти крипто-токени або одразу всім учасникам системи або нікому. Якщо транзакція буде скасована, то це вплине на всіх учасників, яким потрібно було надіслати криптографічні токени.

5. Негативні сторони роботи Варто надати детальніше
застосування асоціативних таблиць або словників в розробленому алгоритмі

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно. Пояснювальна записка відповідає нормам для її оформлення.

7. Відгук про роботу в цілому В загальному кваліфікаційної роботи заслуговує позитивної оцінки. Весь матеріал дипломної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої задачі.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «добре»

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) зав. Дувач кафедри телекомунікаційних медійних та інтелектуальних технологій
д.т.н. Підченко Сергій Костянтинович

« 1 » 12 2022. (підпис)

ХНУ. Зам. 54. тир. 6000. 2007