Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему: «Криптовалютна інформаційна технологія підвищення безпеки роботи з фінансовими активами»

Виконав: студент 2 курсу,
групи 1КН-18м
спеціальності 122 «Комп'ютерні науки»
Білик Р.В.
Керівник: к.т.н., доцент. каф. КН, Сілагін О. В.
Рецензент: к.т.н., доц. каф. ПЗ, Войтко В.В.

Вінниця, 2019 рік

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д-р техн. наук, проф. Яровий А.А.
(наук. ст., вч. зв., ініц. та прізв.) (підпис)
"____"_____2019 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра наук зі спеціальності: 122 «Комп'ютерні науки »
(шифр – назва спеціальності)

08-22.МКР.001.18.000.ПЗ

Магістранта групи 1КН-18м Білик Руслан Володимирович
(назва групи) (прізвище, ім'я і по батькові)

Тема магістерської кваліфікаційної роботи: «Криптовалютна інформаційна технологія підвищення безпеки роботи з фінансовими активами»

Вхідні дані: Мова програмування: об'єктно-орієнтована, середовище розробки: підтримує роботу з об`єктно-орієнтованими мовами; Операційна система Windows7/8/10 з підтримкою бібліотеки Web3. Реляційна база даних з потужністю до 1000 записів, підтримка стандарту ODBS. Час виконання транзакції до 30 сек, розмір повідомлення до 780 kB.

Короткий зміст частин магістерської кваліфікаційної роботи

1. Графічна: схема алгоритму модуля по роботі з Blockchain мережею по валідації даних, схема алгоритму роботи смарт-контрактів мережею по валідації даних, діаграма класів торгової платформи по валідації даних, діаграма послідовності торгової платформи, ER-модель бази даних, структура функціонування смарт-контрактів, приклад роботи програми

2. Текстова (пояснювальна записка): вступ, обгрунтування доцільності роботи криптовалютної інформаційної технології, аналіз предметної області, аналіз платформ випуску токенів, особливості розробки додатків на базі Blockchain, розробка і моделювання криптовалютної інформаційної технології, аналіз програмних технологій для взаємодії з Blockchain, математична модель механізмів захисту, проектування та програмна реалізація криптовалютної інформаційної технології, економічна частина, тестовий приклад роботи програмних модулів та аналіз результатів, висновки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Обґрунтування доцільності розробки криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Розробка і моделювання інформаційної технології підвищення безпеки роботи з фінансовими активами			Математичні моделі, розділ 2
3	Проектування та програмна реалізація криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами.			розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

“____”_____20___р. _____Сілагін О. В.
ініціали та прізвище

2. Економічна частина _____ канд. екон. наук, доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)

“____”_____20___р. _____М. В. Бальзан _____
ініціали та прізвище

Дата попереднього захисту роботи “____”_____20___р.

Рецензент _____ __доц., доц. кафедри ПЗ
(підпис) наук. Ступінь, вчене звання (посада)

__В. В. Войтко_____

Завдання видав
науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

“____”_____20___р. О. В. Сілагін
ініціали та прізвище

Завдання отримав магістрант _____ Р.В. Білик____
(підпис) ініціали та прізвище

“____”_____20___р.

АНОТАЦІЯ

Магістерська робота присвячена розробці криптовалютної інформаційної технології по роботі з фінансовими активами. Були розглянуті основні задачі по створенню токенів і процесами по взаємодії з ними, визначені переваги і недоліки технології Blockchain. Було оглянуто існуючі рішення, на основі яких було сформовано вимоги до програмного модуля. . Розроблено загальну схему алгоритму роботи даної технології. Створений програмний додаток написаний на мові програмування JS, програмне забезпечення характеризується зручністю та зрозумілістю інтерфейсу, швидкістю та точністю опрацювання даних, що забезпечує всі вимоги користувача щодо організації процесу.

ABSTRACT

The master's thesis is devoted to the development of cryptocurrency information technology on working with financial assets. The basic tasks of creation of tokens and processes for their interaction were considered, the advantages and disadvantages of Blockchain technology were identified. Existing solutions were reviewed, on the basis of which the requirements for the software module were formed. . The general scheme of algorithm of work of this technology is developed. Built in JS programming language, the software is characterized by user-friendliness and user-friendliness, speed and accuracy of data processing that meets all the requirements of the user in organizing the process.

ЗМІСТ

Вступ

Актуальність. У сучасному світі наше життя все більше нерозривно пов'язане з грошима, даними і документами. Це породжує тенденцію до збільшення об’єму важливої інформації що, зберігається на серверах різних компаній. Також почали з’являтись електронні платежі, різновиди яких, в наш час існує досить широкий спектр. Така варіативність важливої інформації створює проблеми: недовіра до посередників, що пов’язані з надійністю зберігання цієї інформації, наявність великих комісійних зборів та швидкість обробки фінансових операцій, адже іноді переказ коштів між країнами може зайняти близько одного робочого тижня.

Анонімні комерційні транзакції можуть захистити конфіденційність споживачів. Деякі споживачі вважають за краще використовувати готівку при купівлі товарів повсякденного попиту (наприклад, продукти харчування та інструменти), щоб не допустити того, щоб продавці збирали інформацію і користувалися нею. Кредитні картки пов'язані з ім'ям людини і можуть використовуватися для пошуку інших даних, таких як поштовий адрес, номер телефону і т.д. Існують анонімні системи електронних грошей, непідконтрольні державі і іншим фінансовим організаціям на кшталт і ін. На цих сервісах відкриваються рахунки (електронні гаманці) і користувачі можуть переводити гроші в різних валютах один одному і організаціямпартнерам даних систем. Однак такі організації працюють в суворій відповідності з національним законодавством.

Тому, найбільш актуальним завданням на сьогоднішній день являється створення розподіленої, публічної системи, що неможливо буде підробити. На даний момент є більш анонімні і альтернативні види грошей, ліквідність і затребуваність яких визначають не скільки споживчі кошики і біржові котирування (хоча такі валюти успішно торгуються на спеціалізованих біржах), і не законодавче закріплення їхнього статусу, оскільки інтерес самих користувачів і довіру ширшого кола компаній, які приймають її в якості оплати. Йдеться про криптовалюти, їх емісія виробляється за допомогою обчислювальних потужностей користувачів (для емісії потрібно затратити

енергію і обчислювальні ресурси) і зазвичай алгоритмічно обмежена. Дозволяють анонімно і безпечно володіти, емітувати і передавати грошові кошти. В 2009 році була реалізована система електронних платежів на основі криптографії (Blockchain), що дозволяє користувачам надсилати кошти напряму без потреби довіряти третім особам. Суспільству потрібен новий, безпечний підхід до залучення інвестицій. Підхід на базі Blockchain використовує інформацію з логіки блок-ланцюга, що забезпечує хорошу трансформацію для всього бізнесу і являється радикальною зміною, яка може принести нові відкриті й автоматизовані системи. Робота з фінансовими активами за допомогою Blockchain технології, може замінити систему довіри, оскільки Blockchain технологія визначається як децентралізована публічна база даних цифрових транзакцій, яка постійно здійснює запис в свій реєстр і ніколи не може бути змінена або стерта.

Дане рішення дозволяє вирішити проблеми, описані вище: надійність, анонімність, збереження даних та швидкість виконання транзакцій. Під анонімністю розуміють процес захисту ідентифікатора і даних про місцезнаходження користувача. Здатність забезпечувати анонімний доступ до послуг, при якому уникає відстеження персональної інформації про користувача і про його поведінку, такої як місце розташування користувача, частота користування послугою і т.д. Важливий аспект збереження анонімності і конфіденційності за умови впливу методів соціальної інженерії або будь-якого тиску на оператора сервера. Багаторівневе шифрування і розподілене характер анонімних мереж, усувають єдину точку відмови і єдиний вектор атак, дозволяють зробити перехоплення трафіку або навіть злом частини вузлів мережі не фатальною подією. За анонімність користувач розплачується збільшенням часу відгуку, зниженням швидкості, а також великими обсягами мережевого трафіка.

Тому актуально розробити таку криптовалютну інформаційну технологію, яка допоможе користувачеві безпечно та анонімно працювати з фінансовими активами.

Зв’язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень

кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення криптостійкості розробка криптовалютної інформаційної технології по роботі з фінансовими активами.

Для досягнення поставленої мети необхідно розв'язати такі наступні завдання:

- провести аналіз предметної області, обґрунтування доцільності та постановка задачі розробки криптовалютної інформаційної технології по роботі з фінансовими активами;
- провести моделювання криптовалютної інформаційної технології по роботі з фінансовими активами;
- здійснити проектування криптовалютної інформаційної технології по роботі з фінансовими активами;
- виконати програмну реалізацію криптовалютної інформаційної технології по роботі з фінансовими активами;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – процес розробки технологій підвищення криптостійкості по роботі з фінансовими активами.

Предмет дослідження – є техногології, математичні моделі, алгоритми, програмні засоби по роботі з фінансовими активами на основі базової технології Blockchain.

Методи дослідження. Для досягнення мети дослідження застосовувалися методи: симетричного, блочного, асиметричного, шифрування, криптографія на еліптичних кривих, криптографічна хеш-функція, аналіз технології та сам принцип роботи Blockchain, концепції створення криптовалюти, методи об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

Удосконалена базова технологія забезпечення криптовалютної стійкості Blockchain, та смарт-контрактів, за рахунок введення етапу хешування даних користувачів з використанням алгоритму SHA-3, що підвищує критостійкість роботи з фінансовимии активами. Удосконалена базова математична модель процесу перевірки даних користувачів, на серверній стороні веб-додатку, за рахунок застосування процесу перевірки базової технології блокчейн, що підвищує криптостійкість роботи з фінансовими активами.

Практичне значення одержаних результатів полягає в тому,

- розроблено алгоритм взаємодії смарт-контрактів з web-сервісом;
- реалізовано програмне забезпечення для підвищення безпеки роботи з фінансовими активами у вигляді веб-додатку.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання теоретичних знань, експериментальними дослідженнями на основі тестування програмної реалізації криптовалютної інформаційної технології на базі Blockchain.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати роботи доповідались на регіональній науково-практичній конференції “ФІТКІ 2019” та опубліковані в електронному репозитарії ВНТУ [1], Міжнародній науково-практичній конференції "Інтернет-Освіта-Наука" [2], а також опубліковані в збірнику праць. Програми і алгоритми розроблені в даній роботі планується до впровадження на підприємстві ФОП « Груша В.В.».

Публікації. За результатами досліджень опубліковано одну статтю у науковому журналі, що входить до переліку фахових видань з технічних наук [2] та двоє тез доповідей науково-технічних конференцій [1]. Прийнято до публікування одну статтю у науковому журналі, що входить до переліку фахових видань з технічних наук.

ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ РОБОТИ КРИПТОВАЛЮТНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДВИЩЕННЯ БЕЗПЕКИ РОБОТИ З ФІНАНСОВИМИ АКТИВАМИ

1.1 Аналіз предметної області криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами

Криптовалюта – електронний механізм обміну, цифровий актив, які зазвичай децентралізовані. Функціонування системи відбувається в рамках розподіленої комп'ютерної мережі. При цьому зазвичай вся інформація про транзакції не шифрується і завжди доступна у відкритому вигляді. Криптографія використовується не для обмеження доступу до даних про транзакції, а для гарантування незмінності ланцюжка блоків, бази транзакцій. Термін закріпився внаслідок статті о Bitcoin «Crypto currency» (Криптографічна валюта), опублікованій в 2011 році в журналі Forbes. Сам же автор Bitcoin, як і багато інших, використовував термін «електронна готівка» [1].

Blockchain – це принципово нова надійна технологія зберігання записів, яка може кардинально змінити підхід до формування і зберігання баз даних. Основа технології Blockchain – в розподіленому зберіганні інформації. Це дозволяє зберігати важливу інформацію одночасно на багатьох серверах (у всіх учасників мережі), при цьому зберігати відкрито і безпечно. Наприклад, на базі цієї технології можна зберігати як історію банківських транзакцій клієнтів, так і базу контрактів, результати голосувань, відбитків пальців або історій хвороб. Інформацію, яка одночасно зберігається у багатьох місцях неможливо підробити, неможливо вкрасти, тому що оригінальні записи тут же можуть бути відновлені з сусідніх джерел. Blockchain є ланцюжком блоків даних, які створюються і зберігаються на комп'ютерах учасників ланцюжка. Всі учасники мережі діляться на дві категорії: звичайні користувачі, які створюють нові записи, і Майнер, які створюють блоки [2].

Майнер перевіряють записи, які створюють звичайні користувачі, формують з них блоки, а потім розсилають ці блоки по мережі. Звичайні користувачі отримують ці блоки і зберігають їх у себе в комп'ютері. Учасники Blockchain-мережі мають доступ до інших комп'ютерів мережі, завдяки чому можна обмінюватися даними. Кожен користувач перевіряє коректність нових даних. Якщо вони достовірні, він зберігає їх і передає далі по мережі [2].

Блок транзакцій – спеціальна структура для запису групи транзакцій в системі біткойнів і аналогічних їй. Щоб транзакція вважалася достовірною («підтвердженої»), її формат і підписи повинні перевірити і потім групу транзакцій записати в спеціальну структуру – блок. Інформацію в блоках можна швидко перевірити ще раз. Кожен блок завжди містить інформацію про попередньому блоці. Всі блоки можна вибудувати в один ланцюжок, яка містить інформацію про всіх скоєних коли-небудь операціях в цій базі. Найперший блок в ланцюжку - первинний блок (англ. Genesis block) – розглядається як окремий випадок, так як у нього відсутня батьківський блок (рис. 1.1).

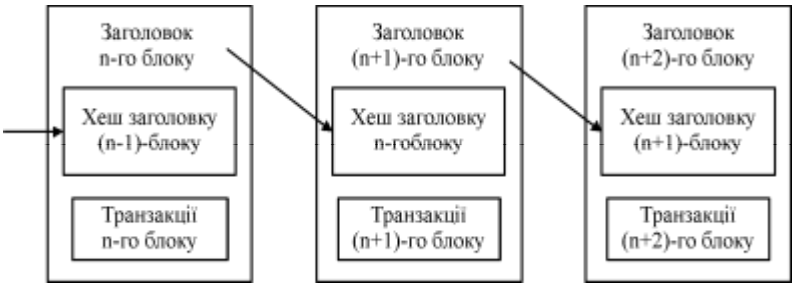


Рисунок 1.1 – Спрощена послідовність блоків

Блок складається з заголовка і списку транзакцій. Тема блоку включає в себе свій хеш, хеш попереднього блоку, хеші транзакцій і додаткову службову інформацію. В системі біткойнів першої транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок. Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Для транзакцій в блоці використовується деревоподібна хешування, аналогічне формування хеш-суми для файлу в протоколі

BitTorrent. Транзакції, крім нарахування комісії за створення блоку, містять всередині атрибута input посилання на транзакцію з попереднім станом даних (в системі біткойнів, наприклад, дається посилання на ту транзакцію, по якій були отримані витрачаються біткойни) (рис. 1.2). Комісійні транзакції можуть містити в атрибуті будь-яку інформацію (для них це поле зветься англ. Coinbase parameter), оскільки у них немає батьківських транзакцій [3].

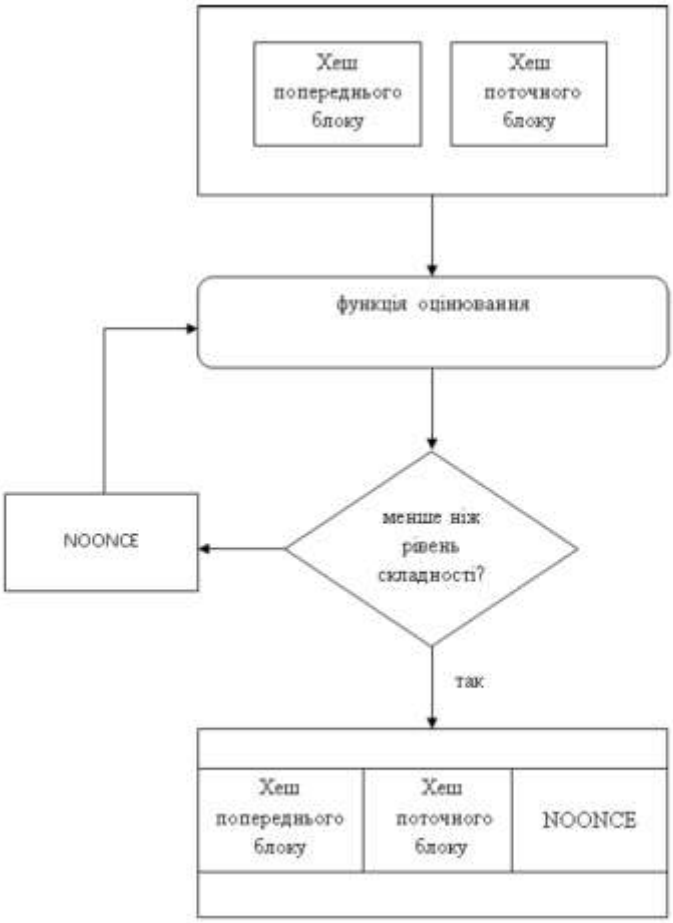


Рисунок 1.2 - Схема прийняття блоку

Створений блок буде прийнятий іншими користувачами, якщо числове значення хешу заголовка одно або нижче певного числа, величина якого періодично коригується. Так як результат хешування (функції SHA-256) є незворотним, немає алгоритму отримання бажаного результату, крім випадкового перебору. Якщо хеш не задовольняє умові, то в заголовку

змінюється параметр nonce і хеш перераховується. Зазвичай потрібна велика кількість перерахунків. Коли варіант знайдений, вузол розсилає отриманий блок іншим підключеним вузлів, які перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок і наступний блок повинен включити в себе його хеш (рис. 1.3).

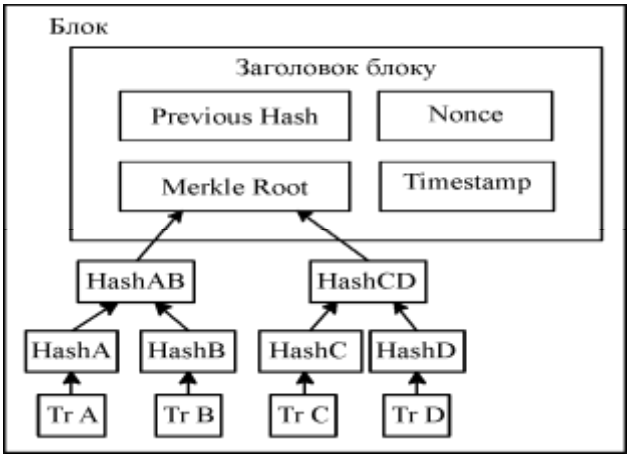


Рисунок 1.3 – Структура блоку

Блоки одночасно формують безліч «Майнерів». Регулярно виникають ситуації, коли кілька нових блоків вважають попереднім один і той же блок, тобто ланцюжок блоків розгалужується. Цілком можливо обмеження обміну даними із загальною мережею - наприклад, одна з ланцюжків може розвиватися в рамках локальної мережі. У цьому випадку можливо паралельне нарощування різних гілок. Коли ретрансляція блоків відновлюється, мережа автоматично буде вважати основною (істинної) більш довгий ланцюжок. У разі рівного розподілу довжини паралельна робота триватиме до створення нового блоку - в якій з ланцюжків блок з'явиться раніше, та й стане довшим, тобто вона буде визнана основною, а робота над паралельної ланцюжком припиниться [3].

Транзакції, що увійшли тільки в відхилену гілку, вважаються тепер поза блоком і будуть поставлені в чергу для включення в черговий блок. Транзакції отримання винагороди за створення відсічені блоків не дублюються в іншій

гілці, тобто біткойни, отримані за формування відсічені блоків, «зникають».

Поки транзакція не включена в блок, система вважає, що кількість біткойнів на якомусь адресу залишається незмінним. У цей час є технічна можливість оформити кілька різних транзакцій з передачі з однієї адреси одних і тих же біткойнів різним одержувачам. Але як тільки одна з подібних транзакцій буде включена в блок, інші транзакції з цими ж біткойнів система буде вже ігнорувати. Наприклад, якщо в блок буде включена пізніша транзакція, то більш рання буде вважатися помилковою. Є невелика вірогідність, що при розгалуженні дві подібні транзакції потраплять в блоки різних гілок. Кожна з них буде вважатися правильною, лише при відмирання гілки одна з транзакцій стане вважатися помилковою. При цьому не буде мати значення час здійснення операції [2].

Таким чином, попадання транзакції в блок є підтвердженням її достовірності незалежно від наявності інших транзакцій з тими ж біткойнів. Кожен новий блок вважається додатковим «підтвердженням» транзакцій з попередніх блоків. Якщо в ланцюжку 3 блоку, то транзакції з останнього блоку будуть підтверджені 1 раз, а поміщені в перший блок матимуть 3 підтвердження. Досить дочекатися декількох підтверджень, щоб звести ймовірність скасування транзакції до мінімуму.

Деревоподібне хешування (рис.1.4) – тип хеш-функції. Використовується для того, щоб перевіряти цілісність даних (файлів), отримати унікальний ідентифікатор файлу, а також дає можливість відновити файл.

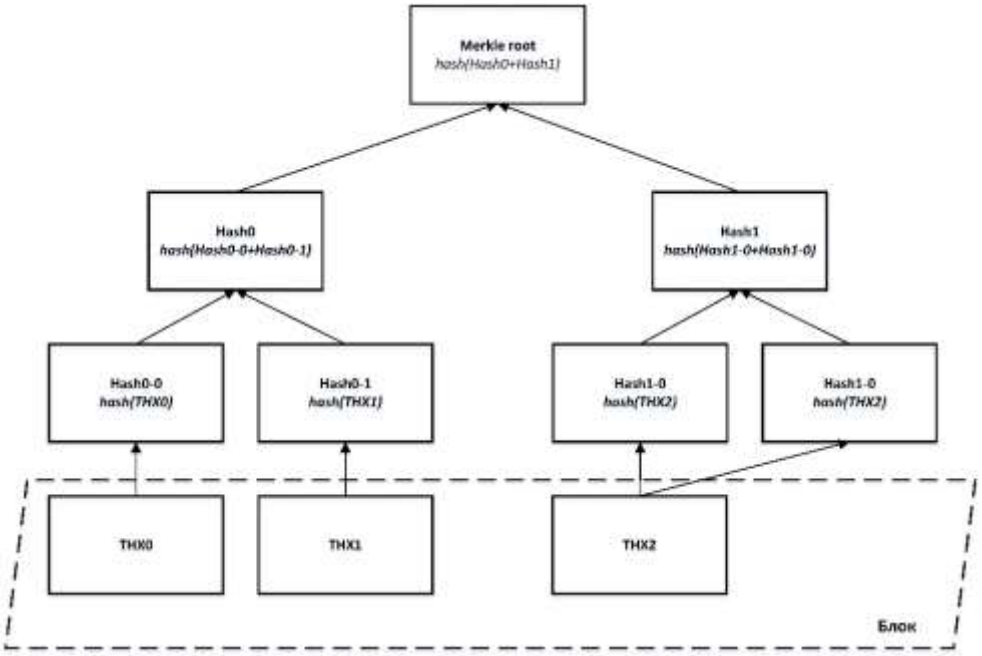


Рисунок 1.4 – Бінарне хеш-дерево

Дані поділяються на маленькі частини - блоки, які індивідуально хешуються за допомогою Leaf Tiger Hash, потім з кожної пари хешів черзі обчислюється Internal Tiger Hash. Якщо хешу немає пари, то він переноситься в нову ланцюжок без змін. Далі в ланцюжку для кожної пари знову обчислюється Internal Tiger Hash. Ця процедура повторюється до тих пір, поки не залишиться один хеш. Цей єдиний залишився Internal Tiger Hash називають Tiger Tree Root. Саме його використовують для однозначної ідентифікації файлу і вказують в різних P2P посиланнях [3].

1.1.1 Різновиди систем обробки інформації

Наразі відомі три системи керування : централізовані, децентралізовані та розроділені

Централізовані системи мають тільки одну точку управління, в якій зосереджений весь контроль за системою (рис. 1.5). Всі процеси виконуються тільки в цій точці, в ній же приймаються і всі рішення. Однак, це робить систему надзвичайно слабкою, адже у будь-який збій - цей єдиний центр управління може обрушити всю систему [4].

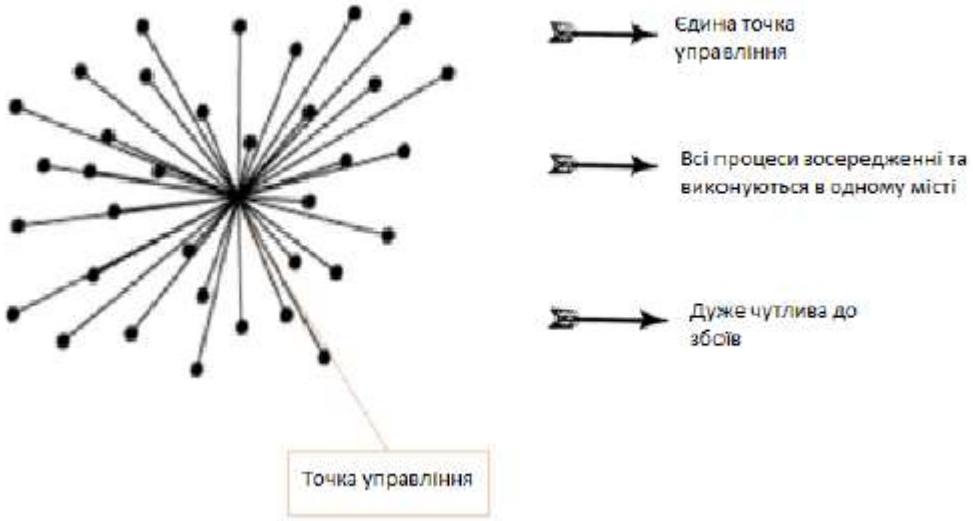


Рисунок 1.5 – Схема централізованої системи

Плюси централізованої системи:

1. Її легко реалізувати і всі рішення приймаються набагато швидше. Так як в ній тільки одна точка управління, в якій зосереджений весь контроль за системою, консенсусу не потрібно.

2. При великому масштабі, система позбавляє від необхідності подвійної роботи, яка іноді робиться при наявності декількох точок управління. Так як в системі тільки одна точка управління, не потрібно змушувати безліч точок виконувати одні і ті ж функції, що дає економію при великих масштабах системи.

Мінуси централізованої системи:

1. Залежність від однієї точки управління. Наявність лише однієї точки управління робить систему уразливою, так як будь-яка атака на цю єдину точку управління дестабілізує всю систему. Нескладно уявити собі ситуацію з сервером, коли атакується єдине джерело інформації і немає ніяких резервних копій, - інформація в цьому випадку буде втрачена.

2. Система з однією точкою управління бюрократична за своєю суттю, що додає в неї безліч шарів та ієрархій.

3. Ця система непрозора і тому має схильність до шахрайства.

Децентралізовані системи – системи, в яких існує кілька точок управління і повноваження диверсифіковані (рис. 1.6). Це робить систему менш чутливою до збоїв, так як вихід з ладу однієї точки управління не призведе до падіння всієї системи. Ієрархія такої системи більше наближена до горизонтальної в порівнянні з централізованою системою [4].

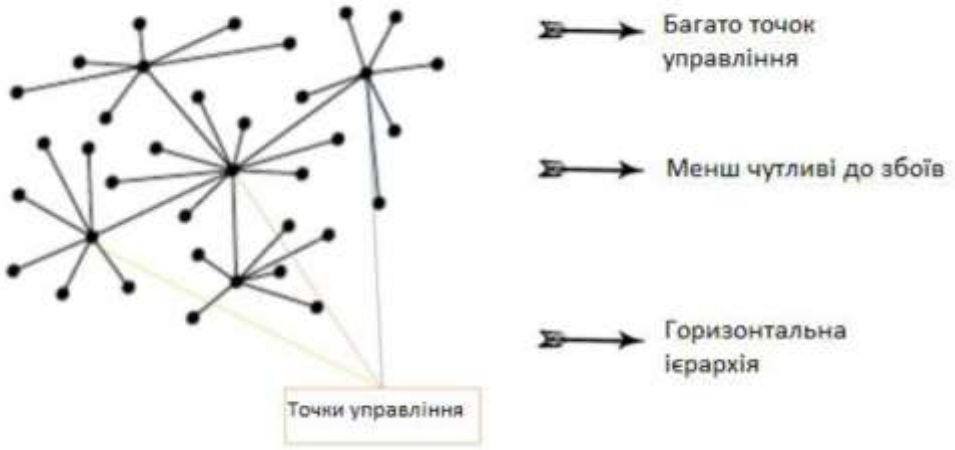


Рисунок 1.6 – Схема централізованої системи

Плюси децентралізованої системи:

1. В даній системі рішення ухвалюються на рівні, більш наближеному до користувача. Таким чином, у органів (точок), які приймають рішення, набагато більше інформації про кінцевого користувача (якщо мова йде про продукт) або про людей (якщо мова йде про уряд).

2. Ця система менше схильна до атак та збоїв, так як тепер в ній кілька точок управління. Збій в одній точці не призведе до дестабілізації всієї системи, як у випадку з централізованою системою.

Мінуси децентралізованої системи:

1. Негативний економічний ефект, пов'язаний зі збільшенням масштабів системи. У такій системі через збільшення кількості точок управління можна отримати «проблему дублювання завдань».

2. Незважаючи на те, що децентралізовані системи надійніше централізованих, вони все одно схильні до збоїв, тому їх не можна назвати

повністю надійними.

У розподілених системах будь-яка точка - це точка управління (рис. 1.7). Тому такі системи фактично несприйнятливі до падінь. Це не означає, що їх не будуть зламувати, однак, щоб вивести з ладу таку систему, зловмисник повинен взяти під контроль або змінити більше 50% точок управління. Витрати на те, щоб зробити подібне самостійно, зведуть нанівець більшу частину прибутку і зроблять економічно недоцільним спроби злому. Ієрархія таких систем повністю горизонтальна. Кожна точка управління дорівнює будь-якій іншій точці управління, і будь-який суб'єкт, будь-який учасник системи є точкою управління.[5] Отже, усі рівні, що і призводить до горизонтальної ієрархії.

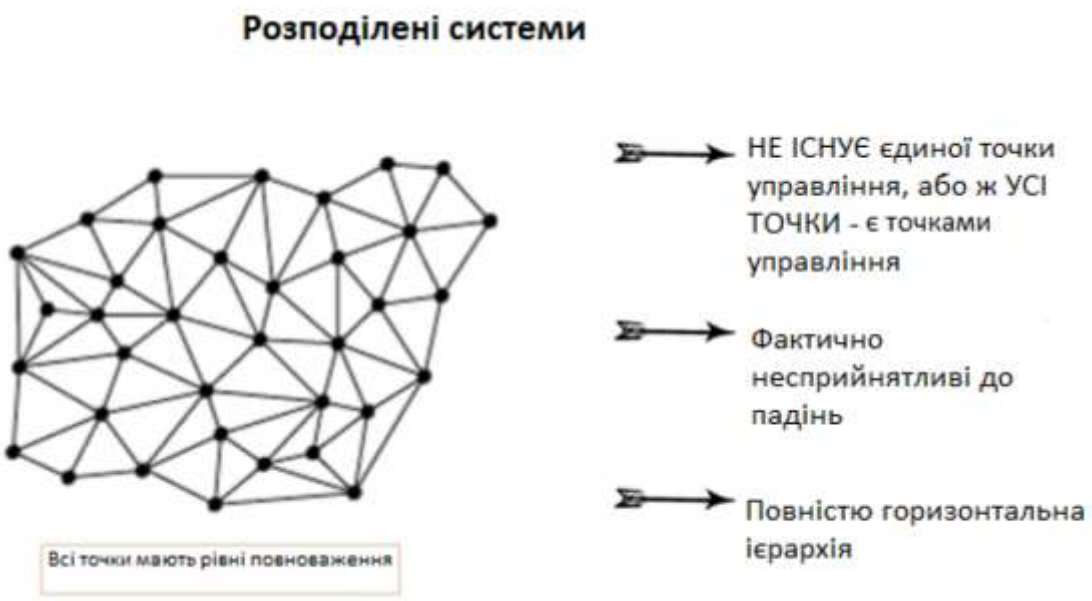


Рисунок 1.7 – Схема розподіленої системи

Плюси розподіленої системи:

1. Відсутність необхідності посередників.

2. Цю систему економічно недоцільно зламувати, що робить її надзвичайно надійною і найбезпечнішою з трьох вище названих систем.

3. Розподілена система повністю прозора, завдяки чому вчинення

шахрайства стає малоймовірним, так як це досить складно.

Мінуси централізованої системи:

1. Такі системи до цих пір вважаються новими, і їх технології знаходяться в процесі постійної розробки.

2. Для стабілізації таких систем буде потрібно багато часу і значна капіталізація.

1.1.2 Розвиток систем фінансової обробки інформації

1. Бартерну систему люди використовували 6 тисяч років до нашої ери. Потім з'явилися монети - приблизно в 650 році до н. е. Люди почали застосовувати перші форми друкованої валюти приблизно в 960 році, а перша форма чека з'явилася приблизно в 1717 році н. е. Вперше ідея "електронних грошей" була запропонована американським фахівцем з теорії складності Девідом Чоумом ще в кінці 70-х років. Перша система електронних грошових коштів PayPal була представлена в 1998 році, в Росії - WebMoney також з'явилася в 1998 році. І, нарешті, в 2008 році з'явилася ідея біткоіни [3].

Система бартеру - перший засіб обміну, відомій людині, - в своєму зародженні була досить простою і досить логічною: будь-яка людина виробляє той товар або послугу, на якій він спеціалізувався, і обмінював свій товар або послугу на ті, які йому були потрібні. Ця система відмінно працювала деякий час, але в ній були присутні недоліки, про які стало відомо пізніше. Розглянемо приклад, щоб це зрозуміти. представимо дві особи - А і Б; А виробляє пшеницю, а у Б є стадо кіз. А та Б вирішили обмінятися - 10 центнерів пшениці за одну козу. В даний конкретний момент А і Б вирішили, що одна коза по вартості дорівнює 10 центнерів пшениці. Однак завтра це може змінитися, і пшениця особи А може не коштувати стільки ж у третьої особи. Припустимо, особі А потрібні банани, він шукає того, хто їх виробляє, і тепер йому доведеться пропонувати різну загальну вартість кожного разу, коли він хоче придбати різні товари. Тому першим і головним недоліком системи бартеру була відсутність загальної одиниці вартості.

Наступний недолік полягав у тому, що товари легко псуються. Кози, пшениця - все це продукти, піддані псування. У них короткий термін життя. Однак, якщо сьогодні є пшениця, а потрібна коза, але через рік, то пшениця не доживе до цього моменту. Тому в даній системі завжди існував розрив у часі між попитом і пропозицією.

Наступним недоліком була неефективна система перевезення товарів. Наприклад, перевезення великих товарів завжди ставала проблемним моментом.

І, нарешті, в бартерній платіжній системі не забезпечувалася безпека - товари могли бути вкрадені і використані кимось іншим, і у власника цих товарів не було докази того, ресурси належать йому.

Монетна платіжна система

Проте люди швидко зрозуміли, що товаром-посередником може служити золото, яке цінувалося усіма і прекрасно вирішило одну з основних проблем бартерної системи - відсутність одиниці загальної вартості. До того ж золоті монети не швидко псуються і інфляції. Золоті запаси в світі завжди обмежені, а це значить, що попит на золото завжди буде підвищуватися, в той час як пропозиція завжди буде обмежена. Це робить будь-які ресурси не піддаються інфляції. Їх вартість буде тільки зростати і ніколи не буде падати. Нарешті, золото було легко транспортувати. Монети були дрібними за розміром і найчастіше однакової форми. Людям більше не доводилося перевозити два різних за розміром товару.

Однак у золотих монет були і мінуси, через які люди перестали їх використовувати. Перший мінус: видобуток золота - витратна і нелегка справа. Другий мінус: незважаючи на те, що золоті монети були порівняно дрібними за розміром і їх легко було перевозити, порівняння з сучасними паперовими грошима вони не витримували. І, нарешті, в системі платежів золотими монетами також не забезпечувалася безпека: їх міг вкрасти і використовувати хто завгодно. Це означає, що дана система не вирішувала саму основну проблему бартеру - не забезпечувала безпеку[6].

Отже, виникла необхідність вирішити питання безпеки платіжної системи з використанням золотих монет, тому люди стали віддавати монети на зберігання третій стороні, яка в наші дні відома як банки. В обмін на це золото банки випускали боргові розписки, які підтверджували отримання золота. Будь-яка людина, що має таку розписку, міг де завгодно обміняти її на товари. Таким чином, боргові розписки були нічим іншим як першою формою паперових грошей. Однак їхні переваги спиралися на фактично існуюча кількість золота.

2. Існувало безліч плюсів паперової платіжної системи. По-перше, такі гроші були легше золотих монет і їх було простіше виробляти. По-друге, ці гроші підкріплювалися чимось: на перших порах - фізично існуючим цінним ресурсом - золотом, а пізніше - недосяжним ресурсом, таким як довіру уряду, яке їх випускало.

Однак і у цієї системи є свої недоліки. І один з них полягає в тому, що в системі паперових грошей не забезпечується безпека. Банкноти можна вкрасти і використовувати. Другий недолік - паперові гроші схильні до інфляції. Отже, паперові гроші - це валюта, підкріплена нематеріальним товаром, наприклад, довірою уряду. Але через те, що якогось відчутного ресурсу, підкріплює банкноти, не існує, уряд може випустити стільки паперових грошей, скільки можливо. Але через таких ринкових сил, як попит і пропозиція, більшу кількість випущених банкнот з часом втрачає свою вартість.

Ще один недолік паперової платіжної системи - це так звані "брудні" гроші і тіньова економіка. У той час не було способу відстежити всі грошові потоки, визначити, на які цілі використовуються гроші, і хто ними володіє.

І, нарешті, з золотомонетной системою в 1970-х було покінчено, тому що вона знижувала гнучкість центрального банку в здійсненні кредитно-грошової і фіскальної політик, потрібних для регулювання економіки країни.

3. Чекова платіжна система і система біткоіни мають багато спільного. По-перше, в чекової платіжній системі нарешті покращилася ситуація з безпекою в засобах обміну: на чеках вказувалося ім'я чекодавця, ім'я

чекодержателя, а також їх адреси. Цей момент має дуже важливе значення для розуміння системи біткоіни. По-друге, в системі чеків використовувалася базова форма криптографії, іменована "підписом", щоб підтвердити, що саме чекодавець доручає провести операцію по чеку (ще одна важлива властивість з точки зору системи біткоіни). І, нарешті, користувач міг поміняти суму на чеку - люди більше не спиралися на певну суму, як у випадку з паперовими грошима[6].

Недоліки чекової платіжної системи пов'язані з появою форми чеків під назвою "чек на пред'явника" - той, хто пред'являв даний чек, міг отримати суму, зазначену в ньому. Це знижувало рівень безпеки. По-друге, та форма криптографії, яка використовувалася в чекової платіжній системі, була надто елементарної, і підписи можна було легко підробити. Нарешті, найбільшим недоліком чекової платіжної системи було час верифікації переказу грошових коштів. Верифікація відбувалася з великою затримкою. Чекодавець вручав чекодержателю чек, чекодержатель відправлявся в свій банк в будь-який зручний для нього час і передавав чек банку. Банк чекодержателя зв'язувався з роздільною інстанцією, яка, в свою чергу, пов'язана з банком чекодавця, щоб підтвердити, що у чекодавця є на рахунку необхідна сума і операція з переказу коштів може відбутися. Проблема ж полягала в тому, що під час переказу грошей чекодержатель був змушений віддавати чекодавцеві товар в обмін на його чек, не маючи при цьому ніяких гарантій того, що у чекодавця є на рахунку сума для завершення угоди.

4. На перший погляд ця система здається ідеальною - вона не вимагає наявності "живих" грошей, людям більше не потрібно носити з собою гаманець. З розвитком індустрії смартфонів і Інтернету і підвищенням рівня застосовуваної криптографічного захисту безпеку електронних гаманців стала набагато вище. В наші дні практично в будь-якому смартфоні є сканер сітківки і сканер відбитків пальців, що може служити засобом підтвердження банківських операцій, а це набагато більш технічно просунуті способи, ніж підпис. Завдяки швидкості, ефективності та зручності цих способів

верифікація операції займає кілька секунд, що вирішує найбільшу проблему системи платежів банківськими чеками. Нарешті, можливість обліку операцій за допомогою електронних гаманців у фінансовій системі набагато вище. Електронні гаманці дозволяють регулюючим органам відслідковувати грошові потоки, контролювати, скільки у кого грошей і для чого ці 30 гроші використовуються. Ця система дає можливість знизити кількість "брудних" грошей і грошей, що використовуються в протизаконних цілях[5].

Отже, з першого погляду система електронних гаманців здається ідеальною. Але виникає питання: чому виникла необхідність в альтернативній системі - нової, вдосконаленої системи під назвою біткоіни? Тому що головна проблема системи електронних гаманців - це те, що не очевидно навіть зараз: ця система є частиною існуючої фінансової інфраструктури. У 2008 році світ побачив недоліки і недоліки існуючої фінансової інфраструктури. Виявилося, що діюча фінансова система неефективна і корумпована. Відбулася криза субстандартного іпотечного кредитування, через якого фінансові ринки обвалилися за принципом доміно, великі інвестиційні банки стали банкрутами, урядам довелося надати фінансову допомогу багатьом банкам. До того ж на світових фінансових ринках був зареєстрований найбільший з 1929 року спад ділової активності [6]. Але це були економічні наслідки, а існували ще й емоційні - люди втратили віру в фінансову систему. Вся суть банків полягала в тому, що вони були третьою стороною, яка надійно зберігає матеріальні цінності і є кращою альтернативою, ніж зберігання грошей вдома. Люди довіряли банкам збереження своїх коштів, але банки не впоралися, - вони неправильно використовували ці гроші і втратили все. Тоді люди зрозуміли, що альтернативної та надійної системи зберігання грошей не існує. Почався масовий вивід коштів з банківських рахунків, і стало зрозуміло, що заміни існуючої системи немає. Люди опинилися перед вибором: або зберігати всі гроші вдома, що в сьогоднішніх обставинах не найкраща ідея, або віддавати їх банку, якому вони більше не довіряли. Виникла потреба в альтернативній системі. В кінці 2008 року особа або група осіб під

псевдонімом Сатоси Накамото опублікував документ під назвою "біткоіни: одноранговая валютна система". Цей революційний документ кидав виклик самій концепції грошей і всієї ідеї існування посередника, який займається грошима. В цьому і полягає причина того, що біткоіни - це найбільш продумана і підриває все підвалини технологія з усіх коли-небудь створених.

1.2 Аналіз платформ для випуску токенів

Робота з токенами залежить від безлічі факторів, однак одним з вирішальних є правильний вибір майданчика. Рішення про те, де саме буде розміщено первинне пропозицію токенів, повністю залежить від розробників.

Найвідоміша платформа для випуску токенів – Ethereum. Її засновник Віталік Бутерін зі своєю командою створив власний алгоритм Ethash, що є гібридом між Proof-of-Work і Proof-of-Stake. По суті, це цифровий підпис, завдання якої – зберегти цілісність повідомлення. Можливість того, що який-небудь елемент хеша виявиться відповідним для різних повідомлень [4].

Плюси використання Ethereum як платформа по випуску токенів:

- широка платоспроможна аудиторія ;
- можливість складання смарт-контрактів;
- активна спільнота;
- якісна техпідтримка;
- пророблена документальний супровід.

До мінусів можна віднести порівняльну повільність - блокчейн Ethereum здатний пропускати 3200 угод за секунду, і з липня почала накопичуватися чергу що чекали на токенізацію проектів – мережа просто не справляється з власною популярністю.

Waves – продукт російського програміста Олександра Іванова. Розробники спочатку зробили ставку на PoS-протокол, а пізніше в просунутій версії - на Leased PoS (LPoS). Це дозволило розділити вузли, що генерують блоки, на «легкі» та «повні», що підвищило швидкість здійснення транзакцій

до 10 тисяч в секунду. Теоретично це може знижувати безпеку [4].

Плюси: швидкість – основна перевага Waves, і це стосується не тільки транзакцій. Досить зручні сервіси дозволяють, не витрачаючи зайвий час, випустити токен.

Мінуси: єдиний недолік, помічений на етапі підготовки – зайва закритість спільноти. Знайти консультанта з практичним досвідом, практично неможливо – так само як і отримати публічні відповіді на найпоширеніші запитання користувачів. Також він несумісний зі стандартом ERC-20, а значить, може торгуватися тільки на внутрішній біржі Waves DEX (обсяг торгів на ній невеликий), і дорога на інші кріптовалютні майданчики для нас буде закрита.

Розробник японської платформи NEM Лон Вонг першим став орієнтуватися на репутацію кожного аккаунта (чим краще репутація - тим більша ймовірність генерації блоку) і запропонував в якості блокчейн-консенсусу модель PoI (proof-of-importance, доказ важливості) [4].

Вона не так вимоглива до обчислювальних потужностей. NEM витримує до 3 тисяч транзакцій в секунду.

Плюси: можливість проектувати на її основі клієнтські сервіси і розвивати різні бізнес-моделі. Причому користуватися алгоритмом можна, не вникаючи в деталі самої технології. Крім того, на NEM дуже вигідно створювати і обмінювати токени.

Мінуси: консенсус вимагає задіяння одночасно великого обсягу мережі, що не завжди просто.

EOS майданчик пропонує асортимент опцій для бізнес-проектів для фандрайзингу. Зовсім свіжа блокчейн-платформа EOS заснована на алгоритмі асинхронних смарт-контрактів. За рахунок паралельного проведення великої кількості транзакцій система може проводити до 100 тисяч угод в секунду [4].

Це рекорд для блокчейнов. Тут застосовується консенсус DPoS (делегованого докази володіння), при якому верифікацію транзакцій проводять вибрані «свідки». Це скорочує число ланок в ланцюжку і збільшує

швидкість транзакцій, але, на думку скептиків, дає можливості шахрайства для нечистих на руку «свідків» [4].

Плюси: операційна система EOS надає рішення для планування, аутентифікації і створення інтернет-додатків. Сервіси дозволяють повністю зосередитися на бізнес-процесах, позбавивши стартаперів головного болю від криптографії і блокчейна. У проекті приваблює можливість швидкого масштабування і відсутність користувальницьких комісій.

Мінуси: розробники поки не показали світові ні детальної технічної документації, ні власне коду.

1.3 Особливості розробки додатків на базі Blockchain

Для того, щоб провести будь-яку угоду, необхідно звернутися до нотаріуса або адвоката, оплатити документи і чекати їх оформлення. Найчастіше, багато пунктів цих документів містять посилання на законодавчі статті, які можна інтерпретувати під себе, обійти. У разі невиконання умов угоди, в реальному житті людям доводиться звертатися до суду, знову витрачати гроші на процес і доводити свою правоту. При укладанні таких угод взагалі не може йти мова про довіру учасників договору.

Найефективніший спосіб проведення операцій в середині Blockchain розробка смарт-контрактів. Смарт-контракти – це фрагменти коду, які використовують можливості технології блокчейн і служать для полегшення, перевірки або гарантованого виконання укладених угод або контрактів. Вперше концепція смарт-контрактів була сформульована Ніком забою в 1996 році. Він назвав «розумний» контракт «набором обіцянок, сформульованих в цифровому вигляді, включаючи протоколи, в рамках яких сторони виконують ці самі обіцянки» [7].

З тих про смарт-контракти захопили уми багатьох візіонерів. У приклад можна привести Віталіка Бутеріна, російсько-канадського програміста і одного з співзасновників Ethereum - найпопулярнішою децентралізованої

платформи смарт-контрактів на базі блокчейна. Але технологія розвивається, причому стрімко, і вже зараз існують десятки інших блокчейн-проектів, здатних створювати «розумні» контракти.

За словами фахівців сфери Джона Риму, Ян Чу і Девіда Шацького «смарт-контракти представляють собою наступний крок у справі просування технології блокчейн від протоколу фінансових транзакцій до універсальної утиліти», пише Deloitte Insights. «Вони (смарт-контракти, - прим. Ред.) Є частиною програмного забезпечення, а не контрактами в юридичному сенсі. При цьому "розумні" контракти розширюють можливості блокчейна в плані збереження записів про фінансові транзакції, що автоматично дає гарантії виконання багатосторонніх угод. Вони використовують комп'ютерну мережу, яка, в свою чергу, базується на узгоджених протоколи для забезпечення виконання умов, закладених в коді контракту, в вірної послідовності» [6].

Смарт-контракти мають широкий спектр потенційних сфер застосування і, крім того, пропонують ряд істотних переваг для підприємств і організацій на тлі загальноприйнятого інструментарію:

Смарт-контракти – це, по суті своїй, шматки коду, які виконуються відповідно до заздалегідь обумовленими умовами. Оскільки «розумні» контракти є за своєю природою автономними і самодостатніми структурами, їх обробка проходить настільки швидко, наскільки дозволяють обчислювальні потужності блокчейн-мережі, в якій вони працюють. Порівняйте це з традиційними контрактами, виконання яких залежить від посередників, які, в свою чергу, можуть мати певні години роботи і, що найголовніше – обмеження в продуктивності, тобто, можуть виконати обмежений обсяг завдань. При розгляді питання з цієї позиції стає очевидним, наскільки смарт-контракти можуть бути корисні підприємствам і бізнесу в їх прагненні бути більш продуктивними [6].

Вчинення помилок - частина людської природи. Якісь із них серйозні, якісь – не дуже, але коли справа доходить до бізнесу, всі вони так чи інакше призводять до втрати дорогого часу і інших ресурсів. Смарт-контракти же

завжди працюють безпомилково, саме так, як було запрограмовано. Вони не залишають місця неточностей, які можуть виникнути через людський фактор, за умови, що не було допущено помилок на стадії програмування.

Оскільки смарт-контракти працюють в рамках незмінної децентралізованої блокчейн-мережі, їх результати не можна підробити заради неправомірного отримання вигоди. Це властивість «розумних» контрактів має величезне значення не тільки для бізнесу, але і для держсектора, про що ми розповімо трохи нижче.

Так як смарт-контракти вимагають набагато меншого участі людини і виключає більшість посередників, вартість їх укладення значно нижче (щодо існуючих практик - прим. Ред). В майбутньому типові «розумні» контракти можуть стати загальнодоступною практикою для всіх, хто зможе їх скачати і використовувати за призначенням. Приблизно та ж ситуація зараз спостерігається в сфері використання типових юридичних форм, створених ліцензованими юристами заради економії часу і грошей клієнтів. І ви самі можете в цьому переконатися, якщо забажаєте.

Всі ці можливості описані в стандарті ERC-20: в мережі викладено багато вихідних кодів, що відповідають за створення власних токенов всередині мережі Ethereum, що працюють на підставі різних вихідних даних. Однак для запуску повноцінного ICO потрібно визначити обсяг розміщується грошової маси, (кількість токенов в розміщенні по раундах), кількість раундів розміщення і вартість токена для кінцевого покупця в кожному з раундів. Всі ці умови можуть бути реалізовані як контракти-доповнення до основного смарт-контракту [8].

Обробка кожного кроку виконання коду варто певної кількості газу. Якщо до транзакції було докладено більше газу, ніж було необхідно для здійснення операції - вартість невикористаного залишку повернеться на баланс відправника. Можлива й інша ситуація, коли в процесі виконання коду ліміт прикладеного ресурсу буде вичерпано. У цьому випадку виконання коду контракту переривається, вже отримані результати анулюються, а сама

транзакція, яка містить договір не потрапляє в блокчейн. Уже витрачений газ не повернеться до відправника.

Другим важливим обмеженням є відсутність підтримки всіляких реалізацій таймерів і затримок виконання коду. Фактично, це говорить про те, що смарт-контракт не вміє працювати «в фоні», він може бути запущений тільки транзакцією від користувача або від іншого контракту. Контракт, повинен буде зробити необхідні переклади на гаманці учасників. Друге уточнення до функціоналу - відлік часу повинен бути реалізований і синхронізований між учасниками в межах мобільного додатка, а не всередині смарт-контракту.

І третій аспект полягає в тому, що смарт-контракти оперують тільки даними з транзакцій і виконання дій за межами віртуальної машини їм недоступно. Наприклад, не можна зробити http запит, щоб завантажити актуальну інформацію про курс валют або безпосередньо опублікувати на веб-ресурсі будь-які дані. Оскільки транзакцію в блок може включити одночасно безліч Майнер (але пощастить тільки одному), одночасно тисячі хостів будуть звертатися із запитом до якогось ресурсу, що, по суті, є DDoS-атакою [9].

1.4 Аналіз систем аналогів

1. PayPal є систему електронних платежів, за допомогою якої можна здійснювати покупки, оплачувати рахунки, приймати перекази.

Однойменна компанія була заснована в 2000 році. Для її створення довелося провести злиття двох інших компаній. Зручність електронних платежів, швидкість здійснення транзакцій і лояльні умови для знову підключилися клієнтів за кілька місяців дозволило залучити в систему сотні тисяч нових користувачів. А початок обслуговування популярного вже тоді аукціону eBay зумовило майбутній успіх сервісу (рис. 1.8).

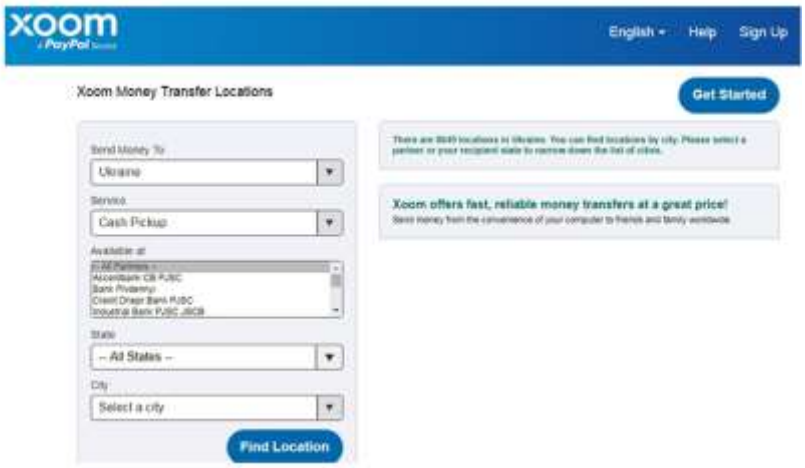


Рисунок 1.8 – Вікно виводу коштів в системі PayPal

Рекордні показники зростання популярності даної системи платежів призвели до того, що вже в 2002 році власником PayPal Inc. стала компанія eBay [6].

В даний час PayPal представлений в більш ніж двох сотнях країн, а в самій системі зареєстровано понад 137 мільйонів користувачів.

PayPal пропонує проводити розрахунки в 26 видах національних валют. Серед найбільш затребуваних: євро і долар США, фунт стерлінгів і японська єна, канадський долар, швейцарський франк, шекелі, мексиканські і філіппінські песо, долари Гонконгу і Сінгапуру, тайські бати і російські рублі.

Комісії для країн колишнього Радянського Союзу відрізняються сумами. При виконанні внутрішніх доларових переказів PayPal з України, Білорусі та Молдови з користувачів утримуються 3,4% плюс 0,3 у.о. У Росії ж перекази в доларах, як і іншій іноземній валюті, заборонені.

Російська комісія за внутрішні переклади становить 3,9% плюс 10 рублів. При переказі коштів за кордон комісія складає від 0,4 до 1,5% і залежить від конкретної країни, куди направляється платіж. Операція з переказу коштів за кордон з російської банківської карти обходиться користувачам додатково в 3,4% плюс 10 рублів [6].

Отримання валютних коштів з-за кордону і їх відправлення в інші країни дозволені для користувачів всіх країн - Росії, України, Білорусі та Молдови.

Комісії передбачені і для одержувачів. Так, російські продавці (одержувачі оплати) змушені оплачувати з кожної операції від 2,9 до 3,9% плюс 10 рублів. Більш детальні умови комісій для одержувачів описані на офіційному сайті системи PayPal.

Для українських користувачів оплата покупок в інтернеті залишається безкоштовною, а для продавців з кожної операції утримується від 2,4 до 3,4% плюс 0,3 долара США.

Платежі по банківській карті в особистих платежах по Україні обходиться в 3,4% плюс 0,3 долара США. При цьому комісійні можуть бути оплачені і одержувачем, і відправником [6].

Стати клієнтом платіжної системи PayPal може кожен. І для цього не знадобиться якихось спеціальних знань або документів.

Процес реєстрації в системі покроково виглядає наступним чином:

- на головній сторінці офіційного сайту PayPal необхідно зайти в форму «Sign Up»;

- вказати країну проживання;

- вибрати «Особистий» тип аккаунта;

- заповнити всі запропоновані поля;

- підтвердити свою згоду галочкою у відповідному полі;

- ввести дані персональної кредитної картки, з якої будуть здійснюватися платежі (при її відсутності даний крок можна пропустити);

- перейти по посиланню, яке прийде на вказану вами при реєстрації електронну пошту;

- підтвердити кредитну карту, перейшовши на сайті відповідне посилання (цей крок зніме з вашої картки 1,95 у.о.).

Після цього на вашу картку надійде невелика сума, а в додатках до перекладу буде вказано чотиризначний число. Його і потрібно буде ввести на PayPal.com.

Звертаємо увагу, що тестовий переклад від PayPal може бути проведений не відразу, а протягом найближчих декількох днів.

Безумовно, ряд обмежень діє для всіх клієнтів PayPal. Але існують спеціальні умови і залежно від країни.

- аноніми не можуть знімати гроші;

- не можуть виконувати переказів на загальну суму понад 40 тисяч рублів щомісячно;

- сума одноразового переказу не повинна перевищувати 15 неоподатковуваних мінімумів доходів громадян.

Користувачі, які пройшли просту перевірку ідентифікації, можуть здійснювати операцій на суму до 200 тисяч рублів на місяць, а для ідентифікованих клієнтів доступна можливість щодня знімати з рахунку до 5 тисяч доларів США.

Виведення коштів з рахунку PayPal в Україні може здійснюватися не готівкою, а шляхом оплати товарів і послуг по всьому світу - на аукціонах, в інтернет-магазинах, для оплати зв'язку, інтернету.

Недоліками системи PayPal можна вважати:

- можливість блокування рахунку без пояснення причин (зрозуміло, блокування можна оскаржити, але для цього потрібен час);

- неможливість проведе

2. WebMoney вважається однією з найбільш популярних платіжних ресурсів. Але одночасно це і сама суперечлива система. При всіх плюсах і мінусах WebMoney, її послугами користується величезна кількість людей в країнах, що є членами СНД і в Росії. Вона була першою російською системою грошових переказів і отримала значний гандикап перед аналогами, які були створені набагато пізніше. За обсягами гаманців, вона займає в російськомовному сегменті позицію лідера і не тільки в Росії, але і в країнах пострадянського простору [7].

ЕРС допомагає здійснювати грошові перекази і вести комерційну діяльність на просторах всесвітньої павутини. І при цьому не потрібно залучати готівку. Спеціальними компаніями-гарантами підтримується легітимність виконуваних операцій.

Системою, щоб створити додаткові зручності для клієнтів, запускаються додаткові сервіси, в числі яких мобільний банкінг (рис. 1.9). Періодично проводяться масові виплати і видаються кредити.

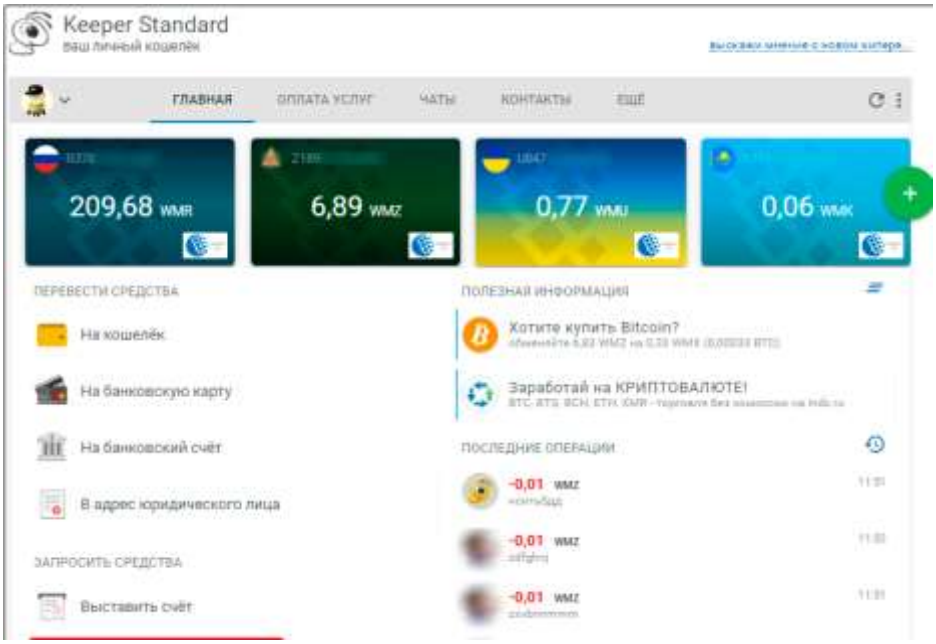


Рисунок 1.9 – Головне вікно персонального кабінету WebMoney

Плюси WebMoney:

- Платіжна система широко розгалужена. Для людей, які працюють в мережі або здійснюють в ній покупки активне користування WebMoney-гаманцем – нормальна практика. Користуючись цією системою можна придбати практично все. Крім того, із застосуванням Вебмані можна оплачувати різного роду послуги, що надаються в Інтернеті. Правда, тут потрібно зробити застереження, що це справедливо для операцій, що здійснюються в межах Росії і країн СНД.
- Високий рівень гарантованої безпеки, безумовно, слід віднести до плюсів WebMoney. Правда, час від часу, в мережі можна прочитати повідомлення викраденні грошей з Вебмані-гаманця. Але найчастіше, ці поодинокі випадки відбуваються через те, що власники лінуються належним чином захищати конфіденційність своїх даних. Фахівцями платіжної системи ведеться постійна робота над підвищенням рівня захисту. А це є гарантією

збереження коштів кожного користувача.

- Наявність мультивалютних рахунків. Після установки програми у користувача є можливість здійснювати керування відразу декількома валютами:
  - Доларами США.
  - Російськими рублями.
  - Євро.
  - Біткоіни.
  - І багатьма іншими.

При цьому можна транзакції виконувати саме з того гаманця, який має найбільший інтерес на поточний момент. На думку багатьох користувачів, ця функція з лишком покриває всі мінуси WebMoney, мова про які піде нижче.

- Доступність системи. Будь-якому новачкові, що бажає розібратися в роботі системи і отримати навички користування нею, це цілком під силу. Мережа буквально кишіла інформацією, яка крок за кроком проведе користувача через процедуру реєстрації та отримання потрібного атестата. Крім того, можна отримати максимальну кількість інформації, що стосується бізнес-рівнів та інших не менш важливих моментів. Все це до мінімуму зводить можливість обманних дій з боку, так званих, «кидал». І при всій простоті системи її параметри підробити неможливо. Завдяки цьому ресурс залишається відкритою, зрівнюючи плюси і мінуси WebMoney [7].

Різноманітність способів, що дозволяють поповнити гаманець або вивести з нього кошти. Плюсом WebMoney є її тривалий час функціонування в мережі. Це дає їй можливість запропонувати користувачам великий асортимент способів, використання яких дозволяє виконувати різні операції з WebMoney-гаманцем.

- Проведення оффлайн-платежів. Удосконалення системи проводиться регулярно. Вже на нинішньому етапі є можливість, використовуючи WebMoney, здійснити оплату різних штрафів, виробляти квартплату, розрахуватися за різноманітні послуги, купити квиток на літак і

зробити багато іншого. Спектр, пропонованих послуг стає різноманітнішою з кожним роком.

Послуги системи доступні цілодобово. А значить, у клієнта є можливість провести необхідні операції в будь-який зручний для нього час. При цьому не має ніякого значення, відкриті в цей час банки чи ні. І це теж плюс WebMoney.

- Можливість отримання кредитів. В системі є спеціальна програма, яка дозволяє кожному клієнту взяти кредит у інших користувачів. При цьому відсотки нараховуються мінімальні. Для цього достатньо мати певний бізнес-рівень і досвід роботи в платіжній системі.
- Функція автоматичної конвертації в потрібну валюту. Тобто клієнт здійснює будь-яку покупку по доларової ціною, а система сама виробляє конвертацію, якщо гроші є тільки на рублевому рахунку.

Мінуси WebMoney:

- В системі WebMoney не дотримуються регламенту, обумовленим на сайті. При тому, що ЕРС вже багато років працює, і всі пункти правил чітко обумовлені, не всі їх дотримуються. Чомусь багато хто вважає, що система буде працювати так, як їм потрібно. Ці мінуси WebMoney створюють для її користувачів певні труднощі.
- Необгрунтована блокування гаманців. Іноді у користувачів складається враження, що для адміністрації системи, введення обмежень для певних портмоне - розвага. Нібито їм це приносить задоволення. Як виправдання, частіше за все, самим потерпілим, пред'являються претензії в тому, що ними проводяться сумнівні угоди. І відсотків на 80 ці претензії цілком обгрунтовані, але ж є ще й 20% чесних підприємців, бізнес яких страждає від цих дій адміністрації. І при цьому відбувається блокування не тільки одержувача, але і відправника.
- Виключена можливість роботи з HYIP проектами. Багатьма розглядаються плюси і мінуси WebMoney з точки зору можливості здійснювати інвестиції власних коштів. Однак якщо робиться спроба вкладення коштів в який-небудь HYIP-проект користувача під певні відсотки

– рахунок миттєво блокується.

- Процес блокування абсолютно непрозорий. Користувача не сповіщають про прийняте системою вирішенні. Відмовка про те, що зниження бізнес-рівня служить попередженням не проходить. Адже робиться це трохи пізніше.

- Техпідтримка не на належному рівні. Виражається це і в тому, що відповіді на запити надходять з великою затримкою. І вираження в цих відповідях не завжди доброзичливі. Не кажучи вже про те, що, як правило, відповідь не повністю висвітлює проблеми, що містяться в запиті. Створюється враження, що клієнтів тут не поважають абсолютно.

- Відсутність анонімності. Щоб повноцінно працювати в системі, необхідно буде укласти персонального атестата. А для цього користувачеві необхідно надати повну інформацію про себе. Якщо відмовитися від цього, то перекривається доступ до багатьох сервісів.

- Відсутність інтересу до системи поза межами Росії та країн СНД. За кордоном попиту на цю платіжну систему немає зовсім. А значить, після перетину кордону слід чекати проблем з виведенням коштів.

- Складнощі доступу. Для отримання до власного рахунку доступу зі стороннього комп'ютера, користувач повинен носити з собою спеціальний файл гаманця і пам'ятати відразу кілька паролів. Інакше ніякі операції з грошима не будуть доступні.

3. Платіжна система Ківі в Україні до недавнього часу була представлена у вигляді мережі терміналів, які давали можливість користувачам взаємодіяти з відомим сервісом електронних платежів. На даний момент робота Qiwi в країні частково обмежена через політичні розбіжності. Незважаючи на це ми розглянемо можливості системи, які були доступні громадянам до часткової заборони її функціонування (рис 1.10).

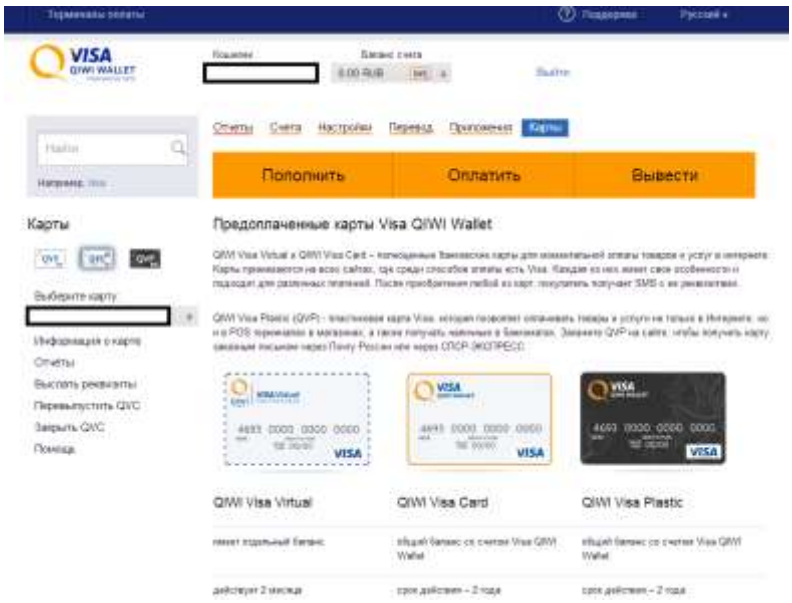


Рисунок 1.10 – Головне вікно персонального кабінету Qiwi

Головне обмеження, спровоковане обстановкою, що склалася в політичних колах, - резиденти України не можуть замовити і використовувати фірмову платіжну карту. Хоча до цього, карту Visa Qiwi Wallet можна було замовити і поштою. Але після того як Україна ввела ряд заходів, спрямованих на обмеження функціонування російських фінансових систем на території країни, місцеві користувачі втратили можливість безпосередньо виводити кошти за допомогою банкоматів [9].

Незважаючи на тимчасові труднощі, багато українців, чия професійна діяльність пов'язана з глобальною мережею, продовжують використовувати систему електронних платежів Qiwi для здійснення взаєморозрахунків. Це зручно і перспективно, якщо говорити про зайнятість в Інтернеті.

Але це не все. Система Qiwi як в Білорусі, так і Україні, має обмеження на проведення фінансових операцій для нових клієнтів. До слова, навіть російські користувачі змушені проходити процес ідентифікації, щоб отримати більш високий рівень доступу до можливостей сервісу. У питанні обмежень громадянство не має абсолютно ніякого значення. Правила «статусів» застосовуються для всіх клієнтів незалежно від резидентства.

На відміну від російських клієнтів сервісу, у громадян України досить є обмежені можливості для поповнення рахунку в системі Qiwi. Ці моменти

повинні враховуватися користувачами, які задаються питанням, чи працює Ківі в Україні. Відповідь може бути несподіваним, але цілком логічним, з огляду на ситуацію, що склалася, - система Ківі працює в Україні, але частково.

Одним з небагатьох способів поповнення електронного Ківі-гаманця є віртуальні обмінні пункти, наприклад, Bankcomat.org. Це якийсь противагу, або навіть протест проти існуючої традиційної фінансової системи. Віртуальні обмінники стали оплотом для всіх користувачів, які змушені слідувати несправедливо встановлених правил фінансових інститутів деяких держав. Зрозуміло, що Казахстан і Білорусь більш лояльно ставляться до системи Qiwi, ніж Україна, але глобальна мережа сьогодні може запропонувати кілька обхідних і цілком легальних шляхів, призначених для ефективної роботи з електронними грошима [10].

Ще один спосіб поповнення і виведення коштів з системи Qiwi є прив'язка гаманця до аккаунту на сервісі електронних платежів WebMoney. Правда, для цього потрібно буде отримати статус не нижче «Стандарт». Алгоритм дій досить простий: прив'язуєте Ківі-гаманець до WMR-гаманця і отримуєте можливість переказувати кошти між системами з комісією всього в 0,8% від суми транзакції..

Це все, що можна сказати по Україні. Ще не нормалізується обстановка в політичних колах між країнами, користувачам, що бажають мати Qiwi-гаманець, доведеться використовувати описані вище обхідні методи для вдалого взаємодії з системою без обмежень.

Пллюси системи:

- Швидка реєстрація, яка займає в цілому 3-5 хвилин. Введення паспортних даних не потрібно, а номером гаманця буде номер вашого мобільного телефону, що надзвичайно зручно для запам'ятовування.
- Поповнити гаманець можна як через термінал, так і через «Зв'язковий». В обох випадках комісія не стягується.
- Здійснюється безкоштовне смс-інформування про всі здійснені

операції.

- Відсутня комісія за оплату послуг і внутрішні перекази.
- Сервіс надає можливість використання реальних і віртуальних пластикових карт. Крім того, до системи можна прикріпити власну банківську карту.

Слабка технічна підтримка платіжної системи. За численними відгуками користувачів, в більшості спірних ситуацій клієнти не змогли домогтися відповідного обслуговування і вирішення конфлікту. Саме тому, не дивлячись на те, що переважна більшість платежів проходить нормально, термінові і серйозні платежі рекомендується здійснювати за допомогою іншої, спеціально призначеної для цього платіжної системи - наприклад, WebMoney.

1.5 Висновки

У результаті аналізу сучасного стану роботи з фінансамми було з’ясовано, що ця задача має велику кількість проблемних місць і на сьогоднішній день, такі технічні рішення використовуються у багатьох сферах людської діяльності. Зокрема в роботі платіжних методів, фінансових систем, банків і т.д.

Проаналізовано існуючі задачі по роботі з фінансовими активами на базі Blockchain.

Існуючі аналоги по роботі з фінансовими активами потребують значних витрат коштів на впровадження та супроводження, а також відсутність контролю власних інвестицій;

Було визначено, що обраний метод робота з фінансовиим активами має високу криптостійкість, але з’являється складність роботи з Blockchain технологією, для розробки програмного продукту сформовано технічне завдання, яке враховує всі вимоги по створенню веб-додатку по роботі з фінансовими активами.

РОЗРОБКА І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДВИЩЕННЯ БЕЗПЕКИ РОБОТИ З ФІНАНСОВИМИ АКТИВАМИ

2.1 Аналіз програмних технологій для взаємодії з Blockchain

Для створення смарт-контрактів ми будемо використовувати Blockchain Ethereum. Ethereum – це блокчейн, який дає вам можливість запускати програми в його довіреної середовищі. Що контрастує з блокчейном біткоіни, так як він дозволяє вам тільки управляти криптовалюта.

У Ethereum для цієї мети є віртуальна машина під назвою Ethereum Virtual Machine (EVM). Віртуальна машина Ethereum (EVM) – віртуальна машина, що складається з безлічі комп'ютерів, в якій всі транзакції зберігаються локально на кожному вузлі мережі і виконуються з відносною синхронністю [8].

До теперішнього часу EVM може чітко позначити себе як: єдина, захищена, не має власників віртуальна машина, яка пропонує дешевий функціонал автоматизованих систем грошових переказів. Хоча будь-хто може використовувати цю глобальну віртуальну машину, ніхто не може створювати в ній підроблені гроші або переказувати кошти без дозволу [11].

2.1.1 Аналіз віртуальної машина Ethereum

Частина протоколу, який виконує обробку транзакцій в операційній системі Ефіріума називається віртуальною машиною Ефіріума (EVM).

EVM є машиною Тьюринга. Єдина відмінність EVM від типової машини Тьюринга полягає в тому, що для роботи першої потрібно віртуальне «пальне». Таким чином, всі обчислення, які можуть бути виконані в EVM, так чи інакше обмежені кількістю циркулюючого в ній, віртуальній машині, «пального».

Крім того, EVM притаманні всі особливості стековой архітектури. Стекова машина - це комп'ютер, в якому застосовується алгоритм LIFO.

Розмір будь-якого елементу стека в EVM дорівнює 256 бітам, а

максимальний розмір стека досягає 1024 бітів.

Для EVM передбачений певний обсяг пам'яті, який не є постійним. У ньому елементи зберігаються у вигляді масивів байтів зі зверненням до слів.

Для EVM також передбачено певна область зберігання. На відміну від обсягу пам'яті, таке сховище (або область зберігання) не змінюється і є частиною стану системи. У EVM програмний код зберігається в окремій віртуальній ROM, доступ до якої можна отримати тільки за допомогою певних інструкцій. З цієї точки зору така EVM відрізняється від типової архітектури фон Неймана, в якій програмний код зберігається в пам'яті комп'ютера (рис 2.1).

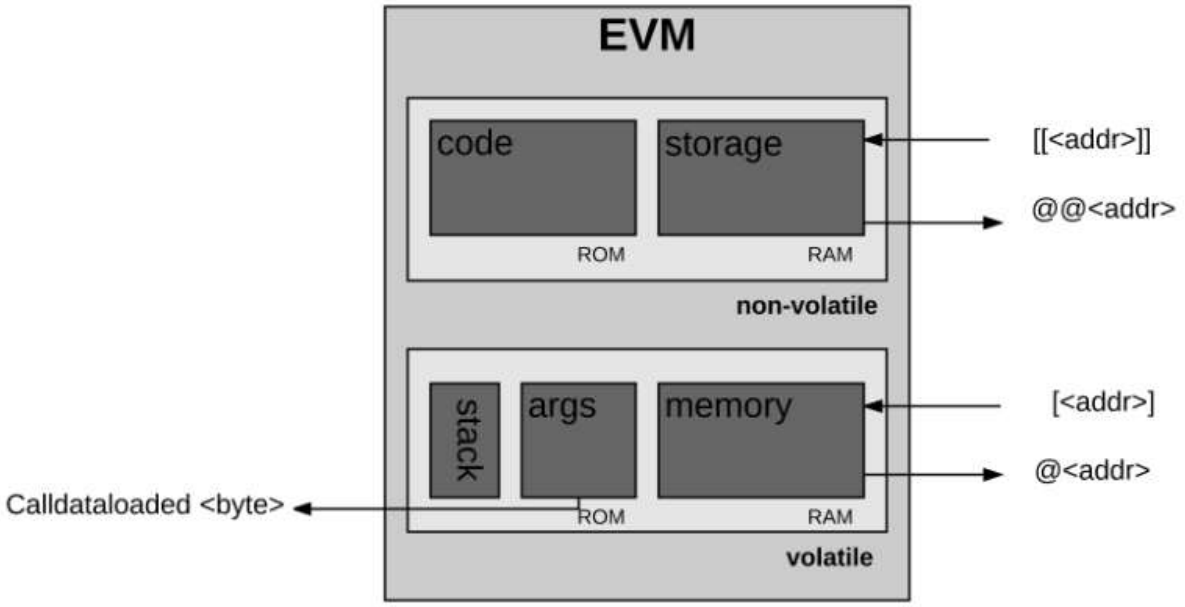


Рисунок 2.1 – структура Ethereum Virtual Machine

Для EVM також передбаченна своя спеціальна мова - байт-код EVM. Коли програміст, такий як ви або, наприклад, я, пише смарт-контракт, який буде виконуватися в системі Ефіріума, це зазвичай відбувається за допомогою високорівневого мови, такого як Solidity. Після написання такого коду ми компілюємо його в байт-код EVM, щоб EVM могла зрозуміти написану нами команду.

Перед тим, як виконати певне обчислення процесор повинен

переконатися в тому, що наведена нижче інформація є валідной і доступною:

Інформація про достатній для виконання необхідної операції кількості пального:

- Адреса облікового запису, якому належить виконуваний код;
- Адреса відправника транзакції – ініціатора виконання поточної операції;
- Адреса облікового запису – ініціатора виконуваного коду (може відрізнятися від адреси відправника-ініціатора) ;
- Інформація про необхідному для виконання транзакції кількості пального;
- Вхідні дані для виконання операції;
- Кількість Wei, який має бути відправлений на рахунок цього облікового запису в результаті проведення поточної операції;
- Інформація про виконуваному машинному коді;
- Інформація про заголовку блоку для поточного блоку;
- Глибина виконання поточного повідомлення або створення контракту;

Безпосередньо до початку виконання програми пам'ять системи є абсолютно порожній, а лічильник команд дорівнює нулю.

Після чого в EVM починається рекурсивне виконання транзакції: обчислення стан системи і стан машини для кожного циклу. Стан системи - це глобальний стан Ефіріума. Стан машини включає в себе:

- доступне кількість пального;
- лічильник команд;
- вміст пам'яті;
- активне кількість слів у пам'яті;
- контент стека.

Елементи стека додаються або видаляються з лівого краю фрагмента коду. Для кожного циклу з остатками кількості пального віднімається його певна частина, при цьому лічильник команд збільшується (рис 2.2). Всього

існує три можливих варіанти закінчення циклу:

Операції, що виконуються машиною, досягають виняткового стану (наприклад, зважаючи на брак віртуального пального, невірних інструкцій, недостатню кількість елементів стека, значення елемента стека, що перевищує розмір в 1024 біт, невірного призначення JUMP / JUMPI) і, таким чином, процес виконання операції припиняється.

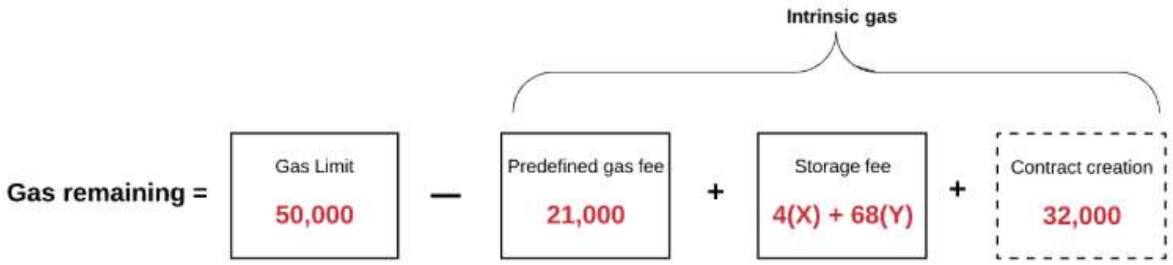


Рисунок 2.2 – Принцип нарахування кількості пального (газу)

Послідовність дій переходить до виконання наступного циклу

Операції, що виконуються машиною, досягають логічного завершення (завершення виконання процесу)

У разі якщо обчислення, що виконуються машиною, досягають логічного завершення, а не виняткового стану, то в результаті цього машина видає результуючий стан, а також інформацію про які залишилися пальному і результуючі вихідні дані.

Програми, написані на високорівневої мовою програмування Solidity, компілюються в байткод і завантажуються в блокчейн Ethereum за допомогою клієнтської програми, такого як браузер Mist, або повного вузла. Вся розподілена мережа, кожен вузол, виконує кожну програму, виконувану на платформі. Коли один користувач завантажує смарт-контракт через свій вузол Ethereum, він включається в останній блок і поширюється по мережі, де зберігається на кожному іншому вузлі в мережі [12].

2.1.2 Аналіз середовища розробки смарт-контрактів

Трюфель – середовище розробки, платформа тестування. Іншими словами, це допомагає вам розробляти смарт- контракти, публікувати і перевіряти їх [11]. Приклад компіляції смарт-контракту в середовищем розробки трюфель представлено на рисунку 2.3. З трюфелем ви отримуєте:

- Вбудована інтелектуальна компіляція контрактів, зв'язування, розгортання управління смарт-контрактів.
- Автоматизоване тестування контрактів з Mocha і Chai.
- Розгортання і міграції смар-контрактів.
- Управління мережею для розгортання в багатьох загальнодоступних і приватних мережах.
- Інтерактивна консоль для прямої контрактного зв'язку.
- Миттєва перебудова активів під час розробки.
- Зовнішній сценарій, який виконує скрипти в середовищі Truffle.

2.1.3 Аналіз емуляторів Blockchain мережі

Ganache – раніше він називався TestRPC. Ganache створює швидкий і гнучкий спосіб емуляції викликів до блок-чейну без накладних витрат на запуск фактичного вузла Ethereum віртуальний ланцюжок Ethereum і генерує деякі штучні облікові записи, які ми будемо використовувати під час розробки [11]. Процес генерації штучних облікових записів.

- Особливості:
- Облікові записи можна повторно використовувати .
- Конкретні рахунки можуть бути показані з фіксованою кількістю ефіру (немає необхідності в зборі).
- Вартість газу та швидкість видобутку можна модифікувати.

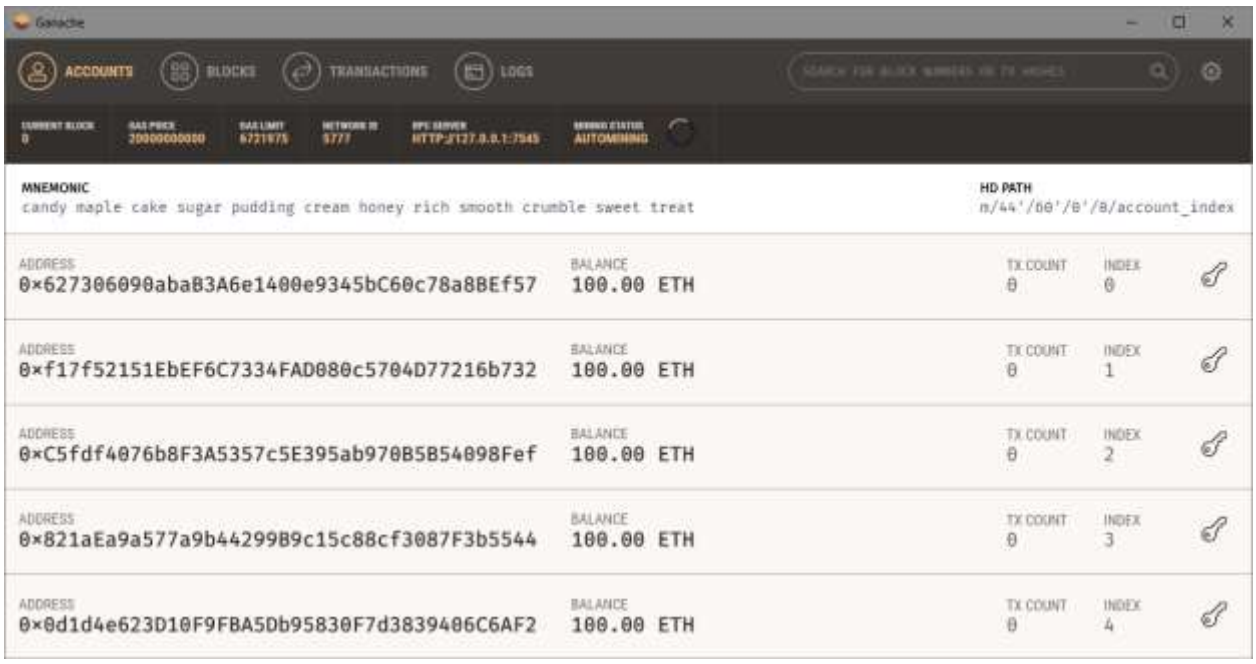


Рисунок 2.3 – Процес генерації штучних облікових записів

Блокчейн дозволяє створювати новий тип додатків – Dapps, щоб з ними безпечно взаємодіяти потрібен посередник. Його роль і виконує плагін Метамаск. Якщо працювати без нього, то вам доведеться кожного разу публікувати своїм базовим гаманцем (а не всі сайти заслуговують таку довіру).

Плагін виконує роль проміжної ланки між вашим основним гаманцем і звичайними сайтами. Ви перекладаєте на нього ефір, які хочете активно використовувати, а всі інші ваші активи спокійно і непомітно зберігаються на базовому гаманці (я рекомендую MyEtherWallet). Головне вікно представлено на рисунку 2.4.

Це робиться, щоб захистити дані ваших криптовалютних гаманців, а також, щоб зробити роботу з ефіром та іншими токенами зручною і швидкою.

Так ви можете завести декілька гаманців під різні цілі і тримати їх на різних комп'ютерах. Так і з МетаМаск: встановили плагін для Chrome, ввели дані, Metamask їх зашифрував, поповнили новий гаманець, а далі ви просто заходите в плагін для перевірки балансу або створення транзакції.

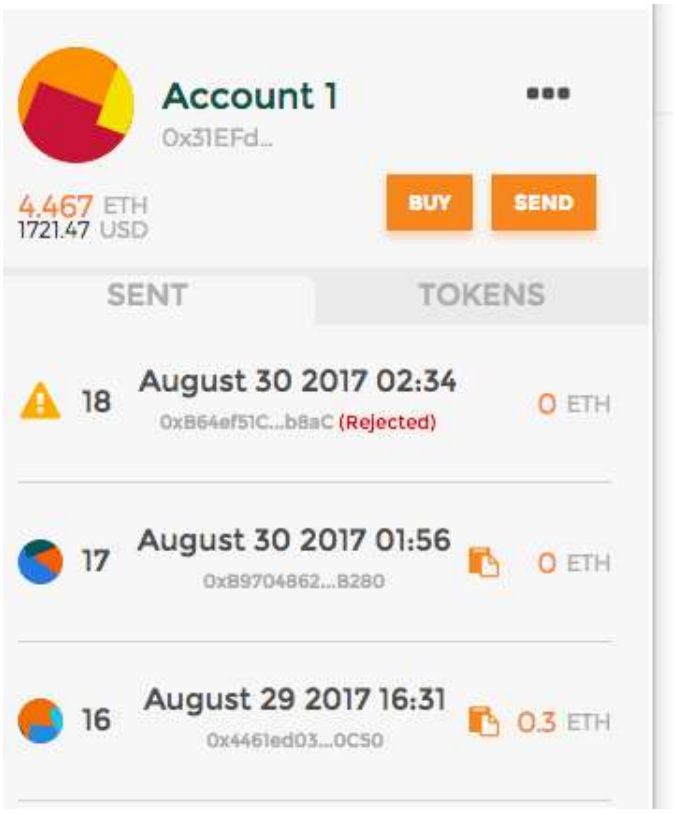


Рисунок 2.4 – головне вікно браузерного розширення MetaMask

2.2 Математична модель механізмів захисту в криптовалютній інформаційній технології підвищення безпеки роботи з фінансовими активами

Криптовалюта використовує однорангову децентралізовану систему для проведення своїх транзакцій. Оскільки весь процес проходить в онлайні, є побоювання, що транзакції можуть перериватися або бути зламуваними хакерами. Тому, ми розповімо, як криптовалюта використовує криптографію задля безпеки своїх транзакцій [13].

Криптографія – це метод використання передових математичних принципів для зберігання і передачі даних в певному формі, так що тільки ті, для яких вони призначені, можуть читати і обробляти інформацію. Криптографія використовується вже тисячі років для таємного спілкування.

Найперше використання криптографії було зафіксовано в гробниці, знайденої в Стародавньому царстві в Єгипті близько 1900 року до нашої ери.

Шифрування – один з найбільш важливих інструментів, що використовується в криптографії. Це засіб, за допомогою якого повідомлення перетворюється в закодований набір символів. Тільки відправник і одержувач знають, що приховано в листі.

У сучасних технологіях широко використовуються три форми шифрування:

- Симетрична криптографія
- Асиметрична криптографія
- Хешування

2.2.1 Симетричне шифрування

Симетричне шифрування – спосіб шифрування, в якому для шифрування і дешифрування застосовується один і той же криптографічний ключ. Ключ алгоритму повинен зберігатися в секреті обома сторонами. До появи схеми асиметричного шифрування єдиним існуючим способом було симетричне шифрування.

Алгоритми шифрування і дешифрування даних мають популярне місце у застосувані в комп'ютерній техніці в системах приховування конфіденційної і комерційної інформації від не коректного використання сторонніми особами. Головним принципом у них є умова, що особа яка приймає зашифрований текст, заздалегідь знає алгоритм шифрування, а також ключ до повідомлення, без якого ці дані є всього лише набір символів, які не мають жодного змістового сенсу.

Симетричні криптоалгоритми виконують перетворення невеликого (зокрема це 1 біт або 32-128 біт) блоків даних в залежності від ключа таким чином, що отримати оригінал тексту, можна тільки тільки у тому випадко, коли ти знаєш цей секретний ключ.

Концепція дуже проста:

- Є повідомлення А, котре Ви бажаєте відправити своєму другові
- Ви зашифровуєте дане повідомлення за допомогою певного ключа і отримуємте зашифрований текст В
- Ваш друг отримує цей зашифрований текст В
- Потім він розшифровує закодований вами текст використовуючи той же ключ і в результаті отримує текст А. Візуально можна представити на рисунку 2.5.

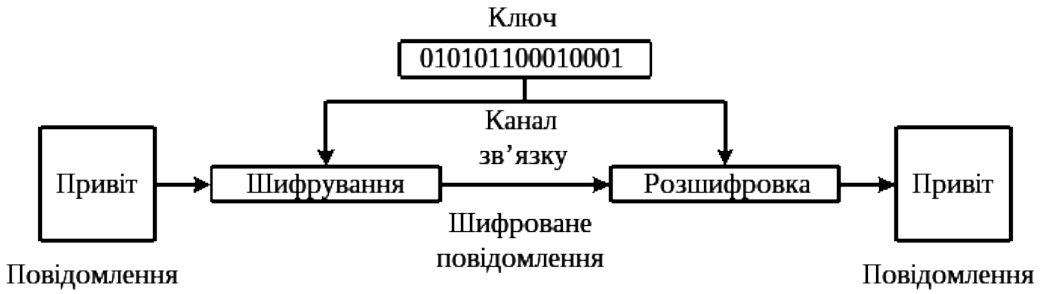


Рисунок 2.5 – Схема роботи симетричного штфрування

Криптографічних алгоритмів існує безліч. В загальному вони призначені для захисту інформації. Симетричні криптоалгоритми відносяться до криптоалгоритмів з ключем.

Вони поділяються на:

1. Потокові шифри – побітна обробка інформації. Шифрування і дешифрування в такому випадку обриваються в довільний момент часу, як тільки з'ясовується, що потік що передається перервався, і також відновлюється при виявленні факту продовження передачі.

2. Блочні шифри – перетворення блоку вхідної інформації фіксованої довжини. і отримують результуючий блок того ж обсягу.

Використовуються різні принципи потокового шифрування, AND, OR або XOR (виключає АБО).

Розглянемо, приклад поточного шифрування:

Для цього шифрування потрібен ключ, який має таку ж кількість символів, що і повідомлення, і його потрібно використовувати тільки один раз (звідси назва «одноразовий»).

Припустимо, що ми відправляємо повідомлення Бобу «СКОРО УВИДИМСЯ». Але ми не хочемо, щоб хтось перехоплював наше повідомлення. Так що ми з Бобом вирішили використовувати одноразовий блокнот, який виглядає наступним чином [14]:

«Г Н П В К У О Х З Ю Т Э М»

Як бачимо, запис має стільки символів, що і повідомлення.

Це дуже простий приклад застосування одноразового блокнота, щоб краще зрозуміти його принципи.

- З'являється ще одна річ: кожна буква з алфавіту замінюється числовим еквівалентом. Візуально можна представити на рисунку 2.6.

А	0	К	11	Х	22
Б	1	Л	12	Ц	23
В	2	М	13	Ч	24
Г	3	Н	14	Ш	25
Д	4	О	15	Щ	26
Е	5	П	16	Ъ	27
Ё	6	Р	17	Ы	28
Ж	7	С	18	Ь	29
З	8	Т	19	Э	30
И	9	У	20	Ю	31
Й	10	Ф	21	Я	32

Рисунок 2.6 – Приклад роботи блочного шифрування

В ході процесу шифрування буде 6 одиниць даних:

- OM вихідне повідомлення: оригінал якого ми написали «СКОРО УВИДИМСЯ».
- NOM числове вихідне повідомлення (його цифровий еквівалент)
- OTP одноразовий блокнот
- NOTP одноразовий блокнот
- NCT числовий зашифрований тест, який є сумою числового вихідного повідомлення і одноразового блокноту. Якщо сума більша,

вираховується 33 у нашому випадку, по кількості букв в алфавіті з 0.

- CT зашифрований текст, який є алфавітним еквівалентом пункту який вище

Давайте почнемо по-порядку:

Так працює даний процес шифрування. А як же працює процес дешифрування ? Процес дешифрування відбувається за тим же самим ключем. Той хто отримав повідомлення ФШЮТЩЖРЮЛЖЯОЛ, має:

- Зашифровану фразу
- Загальний ключ
- Таблицю із числовими еквівалентами

І так, виникає питання. Як Боб дешифрує повідомлення, використовуючи ці дані?

- Він відобразить числові значення ключа, так зашифрованого повідомлення, так щоб отримати NTC та NOTP
- Потім, підрахує NOM (числові значення вихідного повідомлення), виконав даний підрахунок : NOM = NCT – NOTP mod 33
- Надалі, використаємо таблицю для отримання відповідних букв.

Тепер, варто розібратись з блочним шифруванням.

Блоковий шифр має фіксовану довжину та кілька однакових блоків. Повідомлення розбивається на ті ж блоки, а якщо розмір фрази менше, її доповнюють, щоб повністю зайняти блок [14].

У одноразового блокнота довжина ключа дорівнює довжині повідомлення, а в блочному шифруванні ця властивість не є необов'язковою. Одним ключем можна зашифрувати довгий текст.

Найпростіший – метод підстановки Атбаш, коли алфавіт просто перевертається: А -> Я і навпаки. Також відомий Шифр Цезаря, коли буква приймає значення котре стоїть за нею (або розташованої на Х).

Наприклад, слово БАК -> ВБЛ, зрушено на 1:

- А Б В Г Д Е Ё Ж З И Й К Л
- Б В Г Д Е Ё Ж З И Й К Л А

Вище наведені приклади є простими, зазвичай шифрування відбується з великими фрагментами даних, рисунку 2.7.

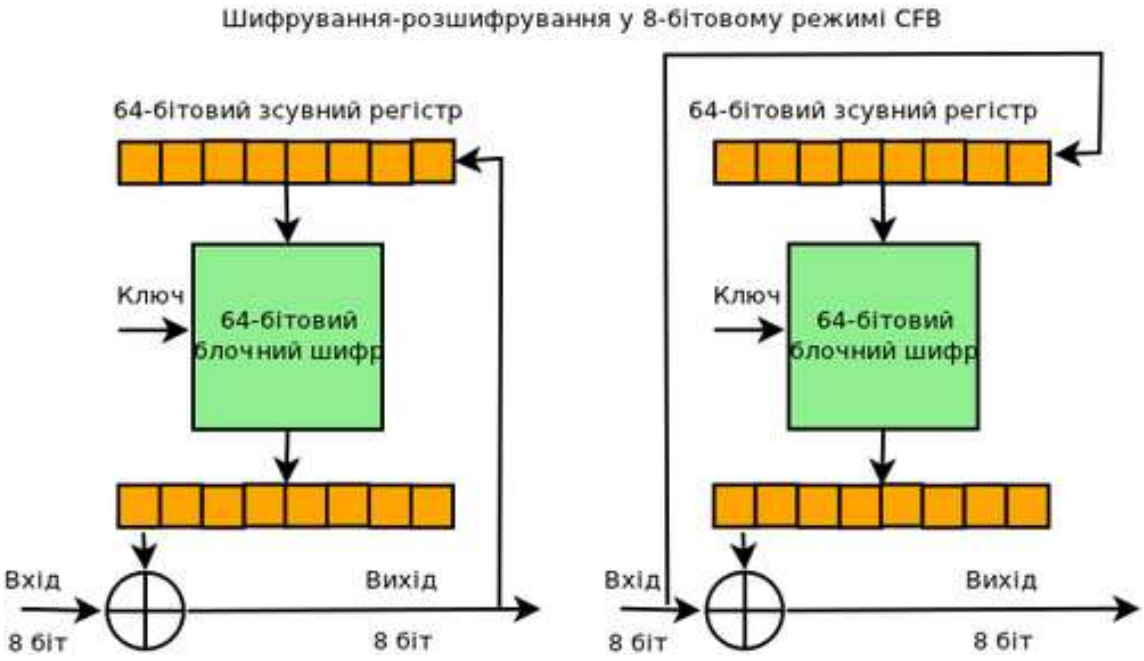


Рисунок 2.7 – Схема роботи блочного штфрування

Незважаючи на те, що симетрична криптографія має деякі серйозні проблеми (що ми обговоримо нижче), проте найбільша її перевага полягає в тому, що вона вимагає дуже значних ресурсів. Все що потрібно, це надати одержувачу один той самий ключ, щоб метод спрацював.

Навіть зараз велика кількість ПО використовує цей метод в поєднанні з асиметричною криптографією для забезпечення швидких і ефективних служб шифрування / дешифрування.

Але наведемо декілька проблем симетричного шифрування:

- Головна проблема – шифрування і дешифрування виконується за допомогою одного ключа. Якщо хтось перехопить ключ, всі дані будуть скомпрометовані
- Немає масштабованості, тобто, припустимо, що інформаційний центр відправляє дані за допомогою симетричної кріптографії. Все нормально, поки клієнтів всього 3-4. Але чим більше їх, тим більше незручно обробляти всі ключі. Дані вразливості симетричного ключа сприяли появі нового рішення в 1970-х роках.

2.2.2 Асиметричне шифрування

У 1970 році британський математик і інженер Джеймс Елліс запропонував ідею, заснована на прості концепці. Що, якщо шифрування і дешифрування - зворотні операції на основі двох різних ключів? У традиційній, тобто симетричній криптографії, повідомлення повинно бути надіслано разом з ключем, щоб інша сторона розшифрувала повідомлення.

Елліс припустив, що одержувач повідомлення не може бути пасивною стороною, і їм потрібно було мати «замок» і «ключ» для себе. Замок можна було відправити кому завгодно в світі, але ключ повинен залишатися приватним [15].

Була теорія, а практичне застосування цієї теорії складається з двох блискучих принципів:

- Одностороння функція з постійним входом: легко перейти від одного стану до іншого, але повернутися у вихідну позицію без ключа неможливо; K - відкритий ключ, к - приватний. Ключі математично пов'язані один з одним через функцію,

$$K = f(k). \qquad (2.2.1)$$

Простий приклад - множення великих чисел. Наприклад, у вас є числа 1847 і 19837. Їх добуток 36638939. Однак, якщо ви просто знаєте це число(добуток), ви не знайдете ні один з множників. Потрібно знати хоча б один, щоб дізнатись другий [13].

- Обмін ключами Діффі-Хеллмана, Протокол Діффі-Хеллмана. Це обмін ключами по незахищених каналах зв'язку. Наведемо приклад, для більш зрозумілості: Аліса і Боб живуть в країні, де немає таємного листування. Але наші друзів дуже хочуть зберегти свою таємницю. Тоді Аліса відправляє замкнений замком ящик з листом Бобу. Боб його отримує, вішає свій замок, відправляє Алісі посилку з двома замками: старим і новим. Аліса отримує,

знімає свій замок, відправляє Бобу. Він отримує, відкриває замок своїм ключем, відкриває ящик і читає лист. Так ні до Аліси, ні до Бобу не потрапляли чужі ключі, і також ніхто не вкрав би їх по дорозі.

Асиметрична криптографія використовує два ключа, відкритий ключ та приватний, для шифрування і дешифрування конкретних даних. Використання одного ключа скасовує використання іншого [9].

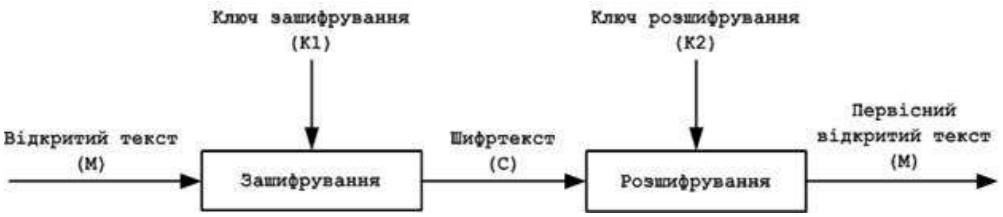


Рисунок 2.8 – Схема роботи асиметрична штфрування

2.2.3 Приватні ключі

Приватний ключ – це просто число, яке взяте навмання. Володіння приватним ключем необхідно для контролю користувача над засобами, пов'язаними з відповідною біткоїн-адресою. Приватний ключ використовується для створення підпису, яка необхідна як доказ володіння операціями в транзакції. Приватний ключ зберігатися в повному секреті, тому що його витік в інші руки еквівалентно передачі контролю над засобами, «замкненими» цим ключем. Приватний ключ також повинен мати резервну копію і захищений від випадкової втрати, тому що якщо він буде втрачений, то і кошти так само будуть втрачені назавжди.

Ще раз, приватний ключ – це просто число. Можете взяти випадковий приватний ключ, використовуючи тільки монету, олівець і папір: кинути монету 256 раз, і у вас є двійкове число випадкового приватного ключа, який можна використовувати в біткоїн-гаманці. Публічний ключ може бути згенерований з приватного ключа [15].

Перший і найважливіший крок при генерації ключів – це знайти надійне джерело ентропії, або випадковості. Створення ключа Bitcoin, по суті, те ж

саме, що «взяти число між 1 і 2256». Точний спосіб вибору числа не має значення до тих пір, поки не передбачуваний або повторимо. Програмне забезпечення біткоіни використовує генератори випадкових чисел операційної системи. Як правило, генератор випадкових чисел ОС инициализируется джерелом випадковості користувача, тому вам може бути запропоновано випадково поворухнути мишею протягом декількох секунд. Для справжніх параноїків, ніщо не зрівняється з кістками, олівцем, і папером.

Більш точно, приватний ключ може бути будь-яким числом між 1 і n - 1, де n константа (n = 1.158 * 1077, трохи менше, ніж 2256) визначається як порядок еліптичної кривої використовуваної в Bitcoin. Щоб створити подібний ключ, ми випадково вибираємо 256-бітове число, і переконуємося, що воно менше, ніж n - 1. В термінах програмування, це зазвичай досягається шляхом подачі великий рядка випадкових бітів, взятих з криптографически стійкого джерела випадковості, на вхід хеш алгоритму SHA256, який видасть зручне 256-бітове число. Якщо результат менше, ніж n -1 [10], ми отримуємо відповідний приватний ключ. В іншому випадку, ми просто пробуємо ще раз з іншим випадковим числом.

Нижче наведений випадковим чином згенерований секретний ключ (k) в 16-тирічному форматі:

Публічний ключ обчислюється з приватного ключа за допомогою незворотного множення на еліптичних кривих:

$$K = k * G \qquad (2.2.2)$$

де k - це приватний ключ, G - константна точка, яка називається генераторної точкою, а K - результуючий публічний ключ. Зворотна операція, відома як "знаходження дискретного логарифма", обчислення k при відомому K можлива тільки за допомогою повного перебору k. Перш, ніж ми продемонструємо, як створити публічний ключ з приватного, давайте поглянемо на криптографію на еліптичних кривих трохи більш докладно.

2.2.4 Криптографія на еліптичних кривих

Останнім часом в криптографії інтенсивно почали використовувати еліптичні криві. ЕК – це розділ криптографії, який вивчає асиметричні криптосистеми, заснованих на еліптичних кривих над кінцевими полями [3]. Асиметрична криптографія заснована на складності рішення деяких математичних задач. Напротиріч раннім криптосистеми з відкритим ключем, такі як алгоритм RSA, наприклад, безпечні завдяки тому, що досить важко розкласти складене число на прості множники. У випадку застосування алгоритмів на еліптичних кривих вважається, що не існує субекспоненціальних алгоритмів щодо розв'язку задачі «дискретного логарифмування в групах їх точок» [14].

У сучасній криптографії на основі еліптичних кривих бінарної розмірності в діапазоні від (150 до 350) забезпечується рівень криптографічної стійкості, який потрібно використовувати у відомих криптографічних системах бінарної розмірності від (600 до 1400) і більше.

Розглянемо принципи побудови і використання еліптичних кривих в криптографії (рисунок 2.9). Кубічна крива на декартовій системі координат *XY*, задана рівнянням, що не має особливих точок, називається еліптичною кривою [9].

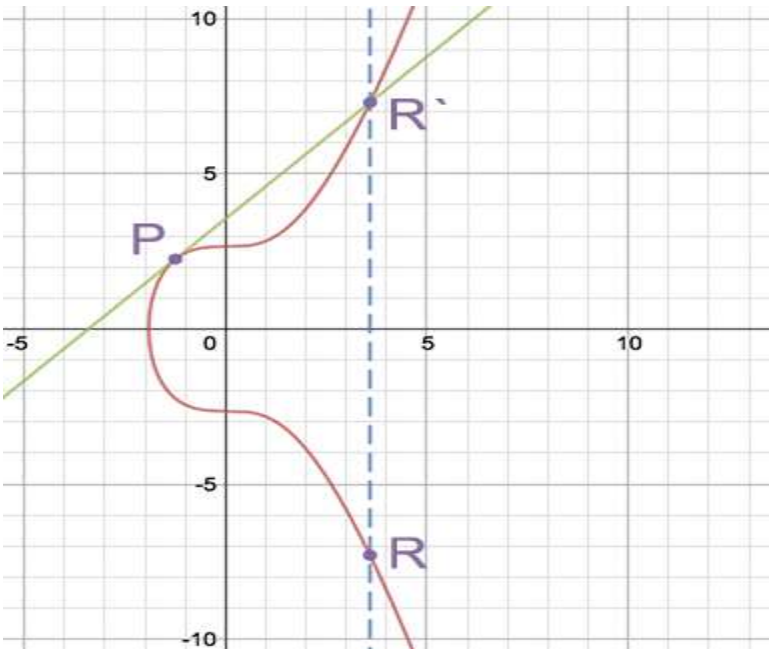


Рисунок 2.9 – Представлення еліптичної кривої

В реальних криптосистемах використовується рівняння:

$$y2 = x3 + ax + b \pmod{p}, \qquad (2.2.3)$$

де a, b належать полю

$$GF(p), 4a3 + 27b2 \pmod{p} \neq 0, \qquad (2.2.4)$$

p – просте число.

Множині Ep(a, b) належать усі точки

$$(x, y), x \geq 0, p > y, \qquad (2.2.5)$$

що задовольняють рівняння 2.2.1, і точку в нескінченності O, яка є нулем для площини еліптичних кривих. Кількість точок еліптичної кривої позначається Ep(a, b) [14].

Множина точок Ep(a, b) має такі властивості:

- якщо P = (x, y) є точкою ЕК, то

$$(x, y) + (x, -y) = O, \quad (2.2.6)$$

точка (x, -y) позначається як –P; Якщо

$$P=(x1, y1), \quad (2.2.7)$$

$$Q = (x2, y2), \quad (2.2.8)$$

де P≠Q, то

$$P + Q = (x3, y3) \quad (2.2.9)$$

обчислюється так:

$$x3 = (y2–y1)/(x2–x1)–x1–x2; \quad (2.2.10)$$

$$y3 = –y1–(y2–y1)(x2–x1)/(x2–x1) \quad (2.2.11)$$

- якщо P = Q, то подвоєння точки P = (x3, y3) обчислюється за формулою [14]:

$$x3 = ((3x12+a)/2/y1)2–2x1; \quad (2.2.12)$$

$$y3 = –y1–(3x12+a)/2/y1 \quad (2.2.13)$$

Одним із найрозповсюджинішим застосуванням криптографічних еліптичних кривих є розподіл ключів. Даний криптографічний протокол обміну ключами з використанням еліптичної кривої має представлення:

- користувач повинен вибрати та оповістити всім форму еліптичної кривої та точку G на даній кривій, яка власне і є генеруючою точкою;
- користувач 1 повинен вибрати ціле число k і знайти точку

$$PA= k*G \quad (2.2.14)$$

- користувач 2 повинен вибрати число m та обчислити точку . Після даної операції користувачі обмінюються власними результатами та спільним їх секретним ключем стає точка

$$k*m*G. \quad (2.2.15)$$

Для того щоб розкрити даний протокол потрібно за відомими k*G та G обчислити k<p, що є аналогічною важкою задачею дискретного логарифмування, як от у випадку алгоритму RSA: де легко обчислити P за відомими k і G, проте важко обчислити k за відомими P і G.

Для криптографічного застосування еліптичних кривих однією з проблем є вибір надійної випадкової кривої. Саме стійкість результуючої криптосистеми визначає вирішення цієї проблеми [16].

Алгоритм формування надійної випадкової кривої :

1. Обирається випадкове просте число p.

2. Далі вибираються випадкові числа a і b, такі, що a, b ≠ 0 і

$$(4a3 + 27b2) (\text{mod } p) \neq 0 \quad (2.2.16)$$

Задля ефективності розрахунків можна рандомно обирати тільки b, а a приймати рівним невеликому цілому числу [14].

3. Визначається кількість точок на кривій

$$п = Ep(a, b). \quad (2.2.17)$$

Слід зазначити, що є важливим щоб п мало великий простий дільник q (це розмір підмножини точок кривої).

4. Виконується процедура перевірки виконання умови

$$(pk – 1) \text{ mod } q \neq 0 \quad (2.2.18)$$

У випадко, коли умова не виконується, то повертаэмось до пункту 2.

5. Виконується перевірка виконання умови q ≠ p. Якщо ні, то повертаються до кроку 2.

6. Знаходимо точку G – генератор підмножини точок q. Коли q = n, то будь-які точки (крім O) є генераторними. Коли q < n, то вибираються випадкові точки G', до тих пір, поки не виконається умова

$$G = [n/q]G' \neq 0 \qquad (2.2.19)$$

2.2.5 Криптографічна хеш-функція

Криптографічна хеш-функція, або хешування — це процес перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини [15].

Хеш-функція – це певна функція h(K), котра бере ключ K і в результаті повертає адресу, по якому відбувається пошук в хеш-таблиці, для того щоб отримати інформацію, пов'язану з K.

Колізія — це ситуація, коли h(K1) = h(K2) [15]. У випадку з колізією, необхідно знайти вже нове місце для зберігання даних. Певна річ, що кількість колізій повинна бути максимально наближена до 0. Хеш-функція повинна задовольняти таким вимогам:

- її обчислення повинно виконуватися дуже швидко;
- вона повинна мінімізувати число колізій

Перша властивість хешування залежить прямою мірою від потужностей комп’ютера, друга — від даних.

Одна із причин застосувань хешування полягає в тому, що вона створює певного роду копію, або ж, ще називають “відбиток пальця” для повідомлення, текстового рядка і т. п. Цей “Відбиток пальця” може прагнути як і до “унікальності”, так і до “схожості”. При створенні хеш-функцій односпрямованого характеру часто використовують функцію стиснення (що видає значення довжини n при вхідних даних більше довжини m і працює з

кількома вхідними блоками). При хеш-функціях враховується довжина повідомлення, щоб виключити проблему появи однакових хеш-адрес для повідомлень різної довжини. Найбільшу популярність мають такі хеш-функції [2]: MD4, MD5, RIPEMD-128 (128 біт), RIPEMD-160, SHA (160 біт). У українському стандарті цифрового підпису використовується розроблена вітчизняними криптографами хеш-функція (256 біт) стандарту ГОСТ 34.311.

Tiger - криптографічний хеш-функція, був призначений для особливо швидкого виконання на 64-розрядних комп'ютерах. Tiger не має патентних обмежень, може використовуватися вільно як з еталонною реалізацією, так і з її модифікаціями. Розмір значення хешу - 192 біта (Tiger / 192), хоча є також більш короткі версії для сумісності з SHA-1 (Tiger / 160) і з MD4, MD5, RIPEMD, Snefru (Tiger / 128). Швидкість роботи - 132 Мбіт / с (перевірено на одному процесорі Alpha 7000, модель 660). На сучасних процесорах значно швидше (навіть при тесті на 32-бітному AMD Sempron 3000+ швидкість близько 225 Мбіт / с) [16].

Кількість використовуваних S-box'ів - 4. S-box виконує перетворення 8 біт в 64 біта. Тобто в кожному з них 256 64-бітних слів і загальний розмір пам'яті, необхідної для зберігання S-box'ов 4 * 256 * 8 = 8192 = 8 Кбайт. Для цього вистачає кеша більшості процесорів, хоча можуть бути складності при реалізації на мікроконтролерах.

Як і в сімействі MD4, до повідомлення додається біт "1", за яким слідують нулі. Вхідні дані діляться на n блоків по 512 біт.

Вибираємо перший 512-бітний блок. Цей блок ділиться на вісім 64-бітових слів x0, x1, ..., x7. Порядок байтів - little-endian.

Беруться три 64-бітових регістра з початковими значеннями (значення хешу $h0$):

Тепер для переходу від значення hi :до значення $hi+1$ виконуються наступні операції (рисунок 2.10):

1. save_abc;

2. pass(a, b, c,5);

3. key_schedule;

4. pass(c, a, b,7);

5. key_schedule;

6. pass(b, c, a,9);

7. feedforward;

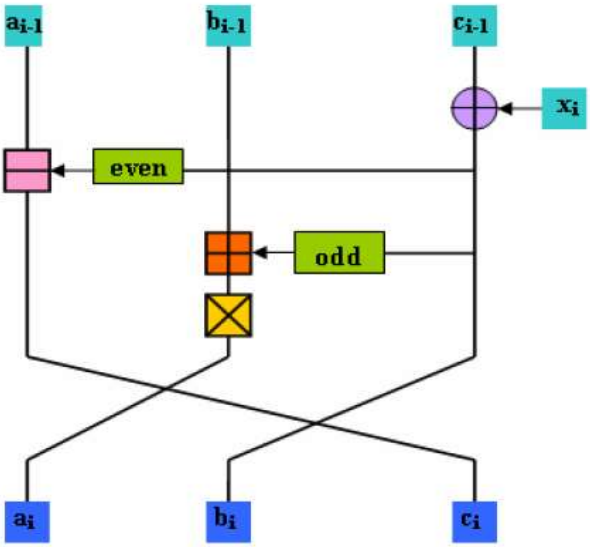


Рисунок 2.10 – Один раунд перетворень Tiger

Основні аспекти безпеки Tiger:

- нелінійність перетворення S-box'ів;
- використання 64-бітових регістрів прискорює лавиноподібне зміна всіх трьох регістрів при зміні будь-якого біта повідомлення;
- генерація ключів забезпечує значні зміни всіх біт на наступних перетвореннях при незначній зміні повідомлення;
- множення регістра b в кожному раунді також сприяє перемішуванню і збільшує опір атакам на пов'язані ключі;
- перешкоджає проміжним атакам днів народження.

Атака на пов'язані ключі є атакою, при якій криптоаналітик може обчислювати хеш для кількох різних значень ініціюючих векторів, які він не знає, але знає деяку взаємозв'язок між ними (наприклад, що вони відрізняються на один біт або якась частина всіх векторів одна і та ж). Через

атаки такого типу з шифрування WEP довелося перейти на WPA.

Проміжна атака днів народження - атака днів народження, застосована на проміжних раундах для досягнення бажаних хеш-значень. Хоча, на думку авторів, атаки подібного типу навряд чи приведуть до складності менше 2 ^ 96 (відповідно до парадоксом днів народження).

2.3 Проектування програмних засобів криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами

Розробка структурної архітектури програми є одним з найбільш важливих етапів у процесі розробки програмного забезпечення, неправильний вибір архітектури веде до ризику зриву всього проекту в майбутньому. Даний етап є базовим для всього процесу розробки. Продумана архітектура дозволяє легко модифікувати програмний продукт, якщо відбудуться зміни вимог до нього.

Архітектура – це набір значущих рішень з приводу організації системи програмного забезпечення, набір структурних елементів та їх інтерфейсів, за допомогою яких компонується система, разом з їх поведінкою, обумовленим у взаємодії між цими елементами, компонування елементів в поступово укрупнюючих підсистеми, а також стиль архітектури який направляє цієї організації – елементи та їх інтерфейси, взаємодії і компонування [6].

Під архітектурою розуміється сукупність компонентів програми, а також зв'язку і способи організації інформаційного обміну між ними. Логічна структура описує об'єкти, які пов'язані з іншими об'єктами. Вона визначає зв'язки між ними.

Для роботи в мережі Blockchain Ethereum будемо використовувати смарт-контракти: ERC-20, Pool Stable Coin, Aitdrops, Swap, MultiSig. ERC20 – це технічний стандарт, використовуваний для інтелектуальних контрактів в Blockchain Ethereum для реалізації токенів.

Стандарт ERC 20 – певний список правил. При їх дотриманні токен може взаємодіяти з іншими монетами. Токен, як якийсь блокчейн-актив, що має цінність, можна відправляти і отримувати як будь-яку іншу криптовалюта. Стандарт ERC 20 включає 6 обов'язкових функцій: TotalSupply, BalanceOf, Transfer, TransferFrom, Approve, Allowance. За допомогою цих 6 параметрів бірж і творцям гаманців вдалося розробити унікальну єдину базу коду, яка може працювати з абсолютно будь-якими смарт-контрактами стандарту eth erc-20.

Pool – смарт-контракт дає можливість кожному, хто має токени роздати їх певним адресатам в певній пропорції. Цей коутракт дає можливість обійти обмеження мінімальних інвестицій або отримати більш приємні умови та бонуси, а також дозволяє адміністратору спільноти заробляти на асигнування з індивідуальними умовами. Створено різні типи пулів (пули спільноти для багаторазових виплат на різні адреси, цільові пули для збору коштів для конкретного проекту), можливість створення вайт-списку Для запуску pool адміністратору потрібно встановити: hard cap, кількість учасників, комісійні адміністраторам.

Stable coin – криптовалюта, призначена для усунення нестабільності цін. Ви можете створити індивідуальний Stable, прив’язаний до фіатної валюти на ваш вибір, і таким чином підтримувати стабільність платежів і зменшити коливання курсу.

Airdrop – такий тип смарт-контрактів, які спрямовані на залучення клієнтів / трафіку до послуги. У першому випадку користувачеві потрібно виконати деякі дії: пошук помилок, створення статей, публікація публікацій на особистому форумі або коментарі в соціальних мережах, реклама тощо. У другому - все набагато простіше. Найчастіше достатньо реєстрації та вступу до групи телеграм. Компанія-організатор вибирає інструмент залежно від цілей та кількості токеніа винагороди. Незалежно від вибору баунті або airdrop, організатору потрібно буде розподілити певну кількість токенів по багатьом адресам і одночасно мінімізувати власні витрати.

Вище перерахлвані контракти матимуть такі функції, як:

- Функція totalSupply визначає загальна кількість токенів. Після досягнення максимуму смарт-контракт перестає їх випускати (рисунок 2.11).

```
/**
 * @dev See {IERC20-totalSupply}.
 */
function totalSupply() public view returns (uint256) {
    return _totalSupply;
}
```

Рисунок 2.11 – Функція totalSupply

- Функція balance0f привласнює первинне число токенів будь-якою адресою (зазвичай це адреса власників). Для розподілу між користувачами і перекладу від одного користувача до іншого необхідні два методу перенесення. Вони вкрай важливі для вторинного ринку (рисунок 2.12).

```
/**
 * @dev See {IERC20-balanceOf}.
 */
function balanceOf(address account) public view returns (uint256) {
    return _balances[account];
}
```

Рисунок 2.12 – Функція balance0f

- Функція transfer переводить токени з первинного адреси індивідуальним користувачам (рисунок 2.13).

```
/**
 * @dev See {IERC20-transferFrom}.
 *
 * Emits an {Approval} event indicating the updated allowance. This is not
 * required by the EIP. See the note at the beginning of {ERC20};
 *
 * Requirements:
 * - `sender` and `recipient` cannot be the zero address.
 * - `sender` must have a balance of at least `amount`.
 * - the caller must have allowance for `sender`'s tokens of at least
 * `amount`.
 */
function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
    return true;
}
```

Рисунок 2.13 – Функція transfer

- Функція transferFrom використовується для пересилання токенів від одного користувача до іншого. Ще дві функції необхідні для перевірки двох попередніх: Функція approve перевіряє, чи залишилися токени у смарт-контракту (рисунок 2.14).

```
/**
 * @dev See {IERC20-transferFrom}.
 *
 * Emits an {Approval} event indicating the updated allowance. This is not
 * required by the EIP. See the note at the beginning of {ERC20};
 *
 * Requirements:
 * - `sender` and `recipient` cannot be the zero address.
 * - `sender` must have a balance of at least `amount`.
 * - the caller must have allowance for `sender`'s tokens of at least
 * `amount`.
 */
function transferFrom(address sender, address recipient, uint256 amount) public returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
    return true;
}
```

Рисунок 2.14 – Функція transferFrom

- Функція allowance гарантує, що на якомусь адресу досить токенів для їх пересилання на іншу адресу. Для того, щоб смарт-контракти отримали можливість розпоряджатися грошовими коштами, що знаходяться на їхньому балансі, в алгоритм роботи мережі потрібно внести ряд змін, які зачіпають основи організації системи зберігання одиниць валюти на балансі (рисунок

2.15).

```
/**
 * @dev See {IERC20-allowance}.
 */
function allowance(address owner, address spender) public view returns (uint256) {
    return _allowances[owner][spender];
}
```

Рисунок 2.15 – Функція allowance

Окрім функцій прямого перерозподілу коштів, розумний котракт виконує функції аудиту. Наслідками будь-якої дії, окрім безпосередньої функції, є емітування повідомлень (Event), що містять інформацію, які дії відбулися, і всі параметри та результати функції. Це призначено для обліку дій акціонерів, інвесторів та фінансових служб. Solidity забезпечує можливість для створення подій, тому завдання розробника полягає лише в описі структури цих повідомлень та визначення часу та умов, за яких їх буде трансльовано в Blockchain.

Було розроблено набір подій, кожна з яких відповідає безпосередній дії користувача, подій перерахування коштів тут немає, оскільки факт оплати зафіксований в Blockchain і не вимагає жодних додаткових доказів (рисунок 2.16).

```
// Івент, що емітується при ініціалізації смарт-контракту
event AssociationCreated(
  bytes32[] shareholderNames, // Імена всіх співвласників
  uint[] shareholderShares, // Кількість акцій всіх співвласників
  address account // адреса рахунку підприємства
);
```

Рисунок 2.16 – Подія AssociationCreated

При додатковій емісії акцій, виплаті дивідендів, ліквідації, а також коли кошти перераховуються на сторонні рахунки, виникають пропозиції, які

мають бути підтримані більшістю співвласників, тому смарт-контракт повинен передбачати голосування. Це означає, що лише тоді, коли "за" рішення про участь в угоді проголосують 50% акцій вона може бути прийнята до реалізації. В системі ця функціональність реалізується за допомогою структури VoteProposal, яка включає в себе дані питань (рисунок 2.17).

```
struct VoteProposal {
  uint id; //Ідентифікатор пропозиції для голосування
  uint _type; // Тип операції, яка буде виконана після завершення голосування
  address creator;
  uint sharesVotesCount; // Кількість голосів "за"
  bool status; //Статус голосування (true - відкрите, false - закрите)

  //Додаткова інформація про голосування
  address addr;
  bytes32 name;
  bytes32 descr;
  uint share;
}
```

Рисунок 2.17 –Структура VoteProposal

Коли один з користувачів активує одну з функцій, розпочнеться процес голосування:

- addNewShareholderProposal – призначити нових акціонерів із зазначеною кількістю акцій;

- withdrawProposal – випуск дивідендів;

- destroyContractProposal – закінчення діяльності смарт-контракту;

- transferProposal – передача коштів третім особам;

Після висунення питання, Blockchain отримує сигнал про подію NewVoting, в якому міститься опис ідентифікатора пропозиції та самого питання. Для прикладу можна звернути увагу на addNewShareholderProposal (Рис. 3.8). Серед параметрів функції є всі змінні, які описують можливих нових співвласників, а тіло функції лише додає питання до наявного списку голосувань та надає йому ідентифікатор для подальшого голосування. В кінці всіх вищезазначених функцій в Blockchain відправляється подія про початок голосування (рисунок 2.18).

```
function vote(uint id) public {
  bool voteSuccess = false;
  bytes32 errMessage = "Vote ID is incorrect";
  for (uint i = 0; i < voteProposals.length; i++) {
    //Пошук питання що подано на розгляд
    if (voteProposals[i].id == id && voteProposals[i].status) {
      errMessage = "Cannot find voter";
      for (uint j = 0; j < shareholders.length; j++) {
        //Пошук інформації про людину, що голосує
        if (shareholders[j].account == msg.sender) {
          //Додання її акцій до тих, що виступають "за"
          voteProposals[i].sharesVotesCount += shareholders[j].share;
          voteSuccess = true;
          errMessage = "";
        }
        break;
      }
      break;
    }
  }
  //Емітування сигналу про те, що користувач успішно/неуспішно проголосував
  emit GotVote(id, msg.sender, voteSuccess, errMessage);
}
```

Рисунок 2.18 – Функція addNewShareholderProposal

Ті, хто підтримує цю ініціативу,ініціюють функцію vote, вказуючи в ідентифікатор голосування, взятого з сигналу про початок голосування, таким чином додаючи кількість своїх акцій в лічильник акцій “за” ініціативу. В кінці процесу голосування, акціонер, що виставив питання порядок денний оголошує результати функцією finishVote. Результатом виконання цієї функції є емітування сигналу про неуспішне голосування або ж приведення 56 угоди в дію. У другому випадку виконується одна з 4 функцій, наведених нижче:

- addNewShareholder – додавання нового акціонеріа;
- withdraw – виплата дивідендів;
- destroyContract – ліквідація смарт-контракту;
- transferProposal – передача коштів на рахунок.

Тепер розглянемо функцію, що відповідає тій, що ми описували вище AddNewShareholder. Дана функція захищена від виконання вручну приватним ідентифікатором досупу, тому вона можлива до виконання тільки після голосування. Під час нього параметри виклику функції зберігаються в

структурі питання на голосування. Робота функції аналогічна до конструктора. Відповідно до вище описаних методів можна створити схему алгоритму роботи смарт-контрактів в мережі Blockchain, рисунок 2.19.

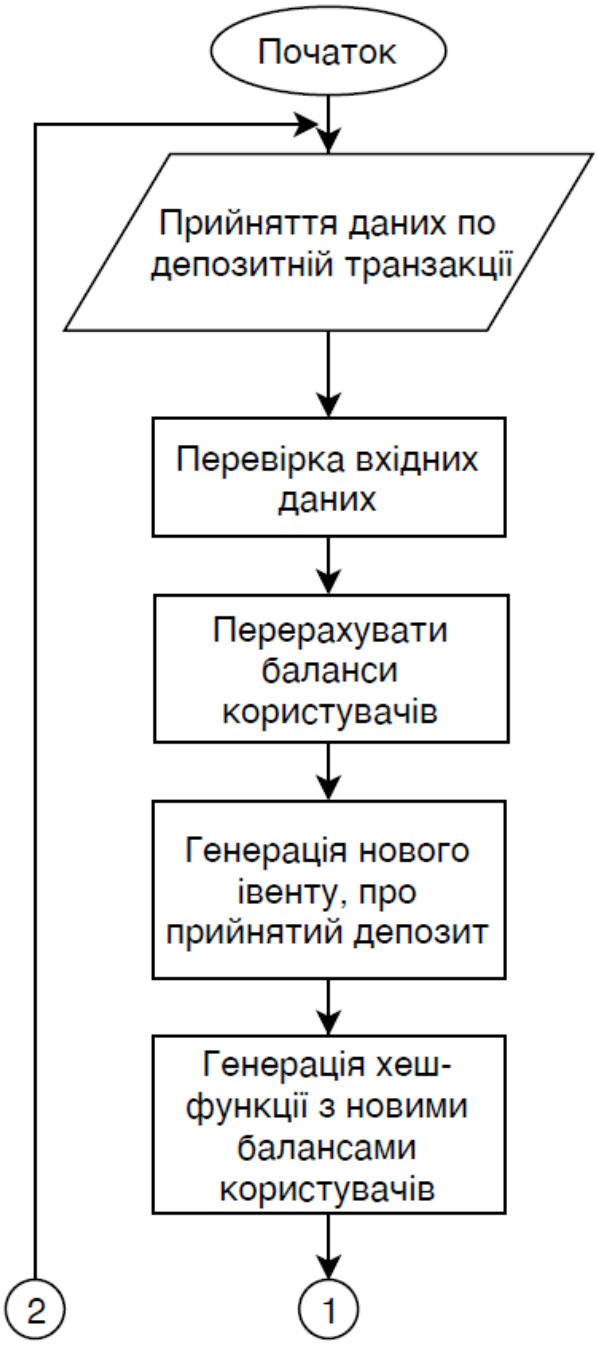
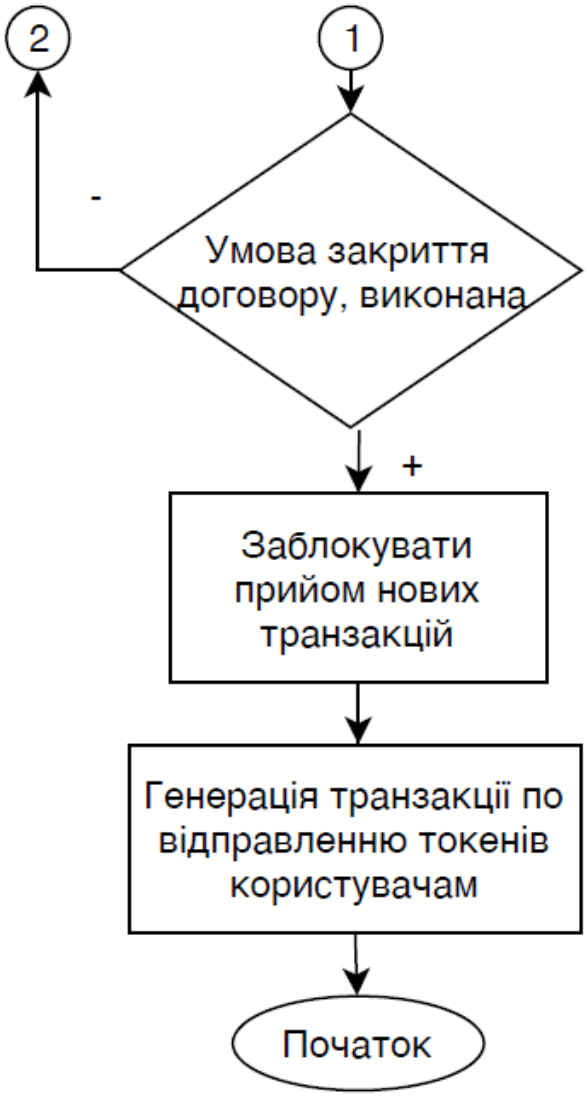


Рисунок 2.19 – Схема алгоритму роботи смарт-контрактів

Продовження рисунку 2.19



Структура смарт-контрактів представляє собою смарт-контракт Factory Цей контракти успадковує від інших контрактів, в яких реалізовані частини функціональності. Наприклад, BonusableToken реалізує нарахування бонусних токенів при покупці. Структура успадкування контрактів показана на рисунку 2.19.

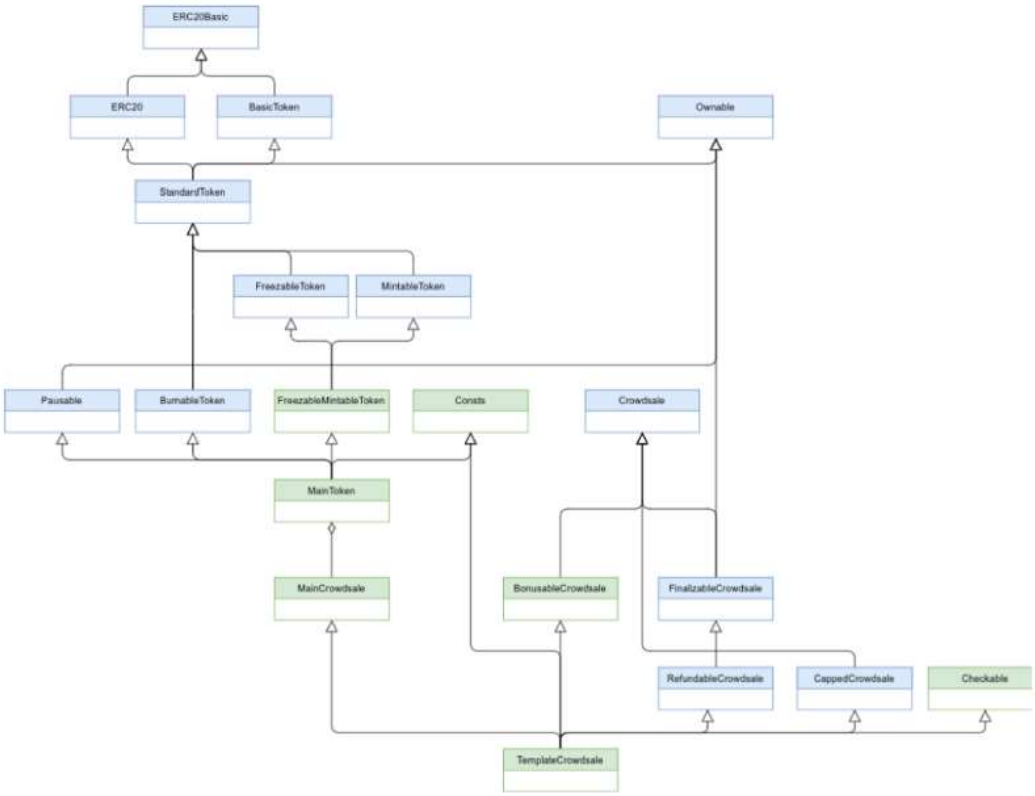


Рисунок 2.20 – Структура функціонування смарт-контрактів

Серед додатків, що працють з Blockchain Ethereum поширена серверна архітектура, в рамках якої весь обмін даними здійснюється між клієнтом, сервером і блокчейном.

Для читання даних з блокчейна необхідно виконати з'єднання з активним вузлом мережі Ethereum. Такий вузол повинен працювати через клієнт geth або parity і перебуває у відкритому доступі. Цей вузол буде грати роль інтерфейсу для виклику так званих постійних функцій, що не змінюють внутрішнього стану контракту. Також можливо використати безкоштовний публічний вузол, який надає команда Infura. За установку з'єднання з вузлом відповідає web3-провайдер.

Існує кілька бібліотек, які реалізують злагоджену роботу і веб-додатків та смарт-контрактів. Прикладом такого програмного модулю є Web3.js, який на думку багатьох експертів і користувачів ресурсу StackOverflow є на данний момент найбільш пристосованим до роботи в реальних умовах. Web3.js - Бібліотека JavaScript для взаємодії з Ethereum смарт-контрактами, а точніше

контрактними блоками [29].

Бібліотека має кілька основних цілей:

- Ініціювати виконання функції смарт-контракту;

- Моніторинг подій, які виникають при виконання функцій;

Існує кілька способів побудови взаємодій між веб-додатками та розумними контрактами. Перша – завантажити Web3 в браузер, щоб у подальшому працювати з ним безпосередньо з клієнтської сторінки. Хоча цей підхід є не цілком правильним, оскільки це відкриває інтерфейс RPC, що не є цілком безпечно. Іншим способом взаємодії додатків та контрактів є написання server-side бібліотеки-обгортки, яка реалізує підключення до вузла Ethereum для безпосереднього зв'язку з розумним контрактом. Цікавим моментом є те, що готові бібліотеки-обгортки вже є у відкритому доступі і можуть бути реалізовані не тільки мовою JavaScript, що дозволяє мультимовну взаємодію та значно розширює стек технологій, можливих до використання в блокчейн-проектах [29]. Особливий інтерес для нас представляє Web3Scala, що дозволяє нам користуватись усіма функціями даної бібліотеки без надлишкових дій. Тепер розглянемо детальніше можливості бібліотеки Web3.js, код для якої є по-суті псевдокодом для використання цієї бібліотеки з будь-яким компілятором. Після підключення, потрібно створити екземпляр класу для установки з’єднання, рисунок 2.21

```
if (typeof web3 !== 'undefined') {
  web3 = new Web3(web3.currentProvider);
} else {
  // set the provider you want from Web3 providers
  web3 = new Web3(new
Web3.providers.HttpProvider("http://localhost:8495"));
```

Рисунок 2.21 – Екземпляр класу для установки з’єднання з Blockchain провайдером

Методи цієї бібліотеки дозволяють відстежувати події. Подія Ethereum має безліч призначень:

- отримати значення, що повертають методи, що змінюють стан смартконтракту;

- отримання повної історії дій над смарт-контрактом.

Наприклад, тут ми отримуємо інформацію про контракт і реальному часі за допомогою інтерфейсу програми allEvents():

Крім цих прикладів, що наведені вище, цей програмний модуль має величезну кількість можливих сценаріїв застосунку.

Таким чином, можна буде скористатися з'єднанням з власними надійними вузлами. Після чого наш додаток матиме можливість відстежувати події контракту з боку клієнта, щоб оновити інтерфейс, як тільки транзакція, відправлена поза додатку, буде виконана. Оскільки відстеження подій – це не що інше, як читання даних з блокчейна, його можна здійснювати через публічний вузол.

Для вирішення цих задачі будемо використовувати браузерне розширення, Metamask, яке виступає в ролі легкого клієнта для взаємодії з блокчейном, а також потрібно буде використовувати набір бібліотек, які дозволяють взаємодіяти з локальним або віддаленим вузлом Ehereum (web3-провайдер), а саме бібліотекою web3.js.

Для того щоб виконувати функції смарт-контракту, нам потрібно буде написати власний API-PRIVATE, яка має бути написана з використанням бібліотеки web3.js. При виклику цих функцій ми матимемо можливість отримати інформацію про транзакцію, а також зміну стану в середині самого контракту. Діаграма діяльності підключення веб-додатку до мережі блокчейн представлена на рисунку 2.22

Рисунок 2.22 – Діаграма діяльності підключення веб-додатку до мережі блокчейн

Тепер додамо до нашої архітектурі сервер. Який матиме локальний вузол Ethereum після чого ми матимемо можливість використовувати інтерфейс JSON-RPC для здійснення всіх операцій в блокчейні. Одночасне взаємодію клієнта і сервера з блокчейном може зажадати координації їх дій. Нам буде потрібно, щоб сервер реагував на будь-яку дію клієнта в блокчейні, або щоб клієнт відобразив зміни в стані контракту, що представлене на рисунку 2.23.

Крім того, можна окремо відстежувати конкретні транзакції, відправлені безпосередньо з нашого додатку, щоб підтвердити їх успішне завершення. Таким чином, серверу не доведеться відстежувати всі події.

Рисунок 2.23 – Діаграма діяльності взаємодії клієнта, серверу з мережею блокчейн

Так як ми плануємо побудувати традиційний клієнт-серверний додаток, тому сервер відповідатиме за бізнес-логіку і координацію роботу клієнтів. Для коректної роботи з користувачем, нам потрібно буде створити базу даних в якій буде лише дві таблиці: user і transaction. В таблиці user буде зберігатися технічна інформація про користувача (логін, захешований пароль, електрона пошта, країна і місто проживання, планована кількість інвестицій, а також інформація про виданий гаманець для інвестицій). В таблиці transaction знаходиться інформація про гаманець, приватний ключ гаманця, що був виданий користувачеві, для інвестування. А також інформвція про саму інвестицію, хеш транзакції, кількість монет заінвестованих, а також публічний ключ гаманця з якого була проведена інвестиція. Так як користувач матиме можливість провести інвестицію декілька разів, тому зв’язок між таблицями буде 1:N. ER-модель створеної бази даних представленої на рисунку 2.24

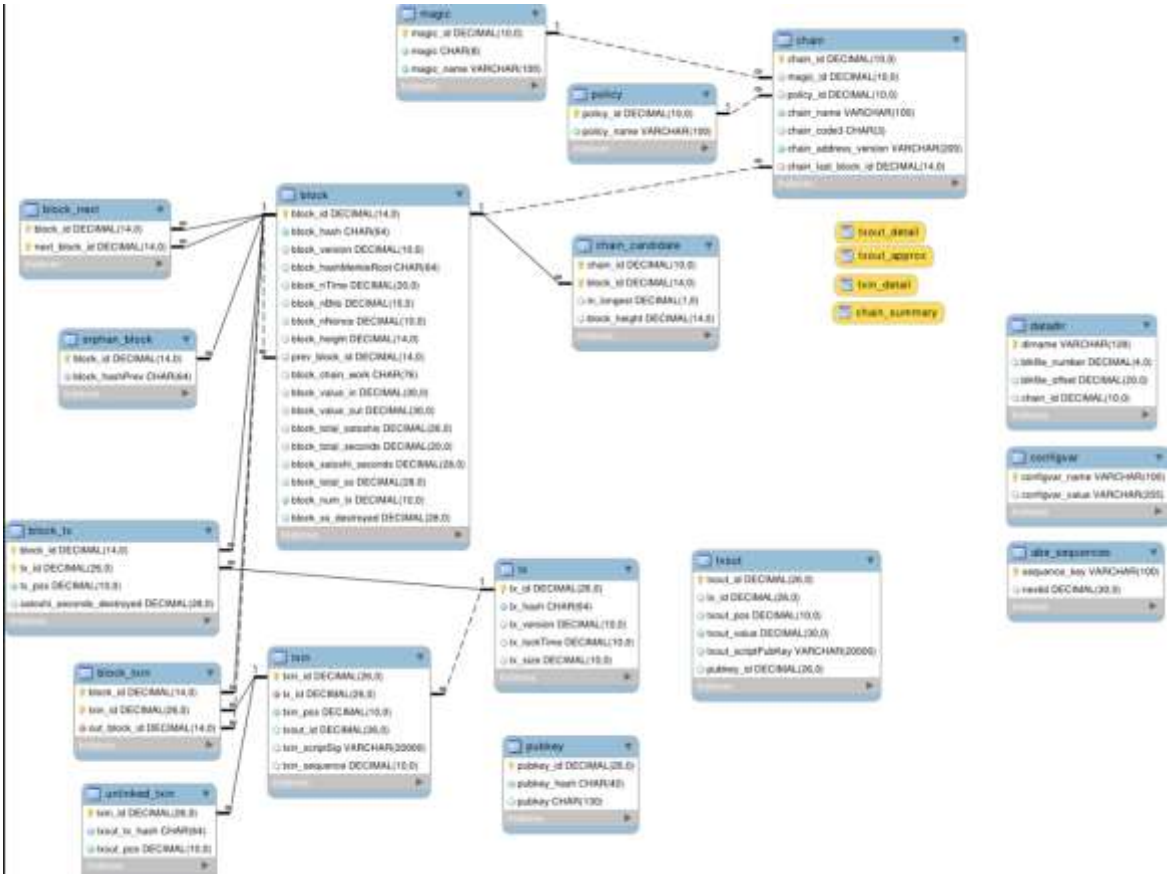


Рисунок 2.24 – ER-модель бази даних

Код, що виконується в блокчейні, не може безпосередньо працювати з офчейн-сервісами. А значить, коли нам буде потрібно інтеграція зі сторонніми сервісами, імпорт курсу USD / ETH та інших даних із зовнішніх джерел або можливість відправки електронної пошти, то нам не обійтися без сервера. Сервер також буде кешувати або індексувати смарт-контракти. Проте блокчейн як і раніше буде грати роль головного джерела інформації, але в той же час клієнти зможуть звертатися до сервера для пошуку даних, а потім підтверджувати їх в блокчейні. Також за допомогою логіки, яка буде описана на сервері, організація, або людина, що проводитиме операції потребуватиме окремий кабінет, в якому адміністратор матиме можливість змінювати основну інформацію, а також відслідковувати перебіг збору коштів.

Для опису структури серверної частини проекту використаємо діаграму UML. Зокрема: діаграму класів.

Діаграма класів вказує типи класів системи і різного роду зв'язки, що виникають між ними [7]. На цих діаграмах вказуються також їхні атрибути, операції(методи) та обмеження, що накладаються на зв'язки між класами (рис.2.25) .

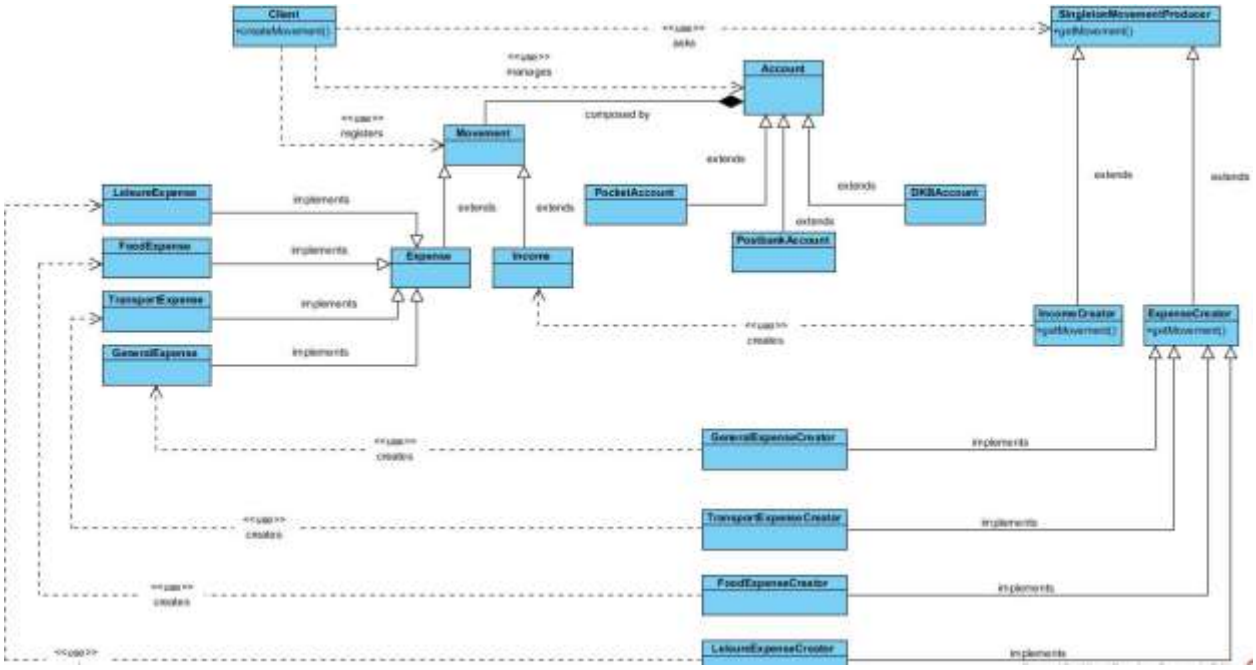


Рисунок 2.25 – Діаграма класів торгової платформи

Після аналізу структурної організація смарт-контрактів, а також роботи клієнт-серверного додатку, з’являється можливість показати різні процеси або об'єкти, що існують водночас, для цього використаємо діаграму послідовності (рис. 2.26) .

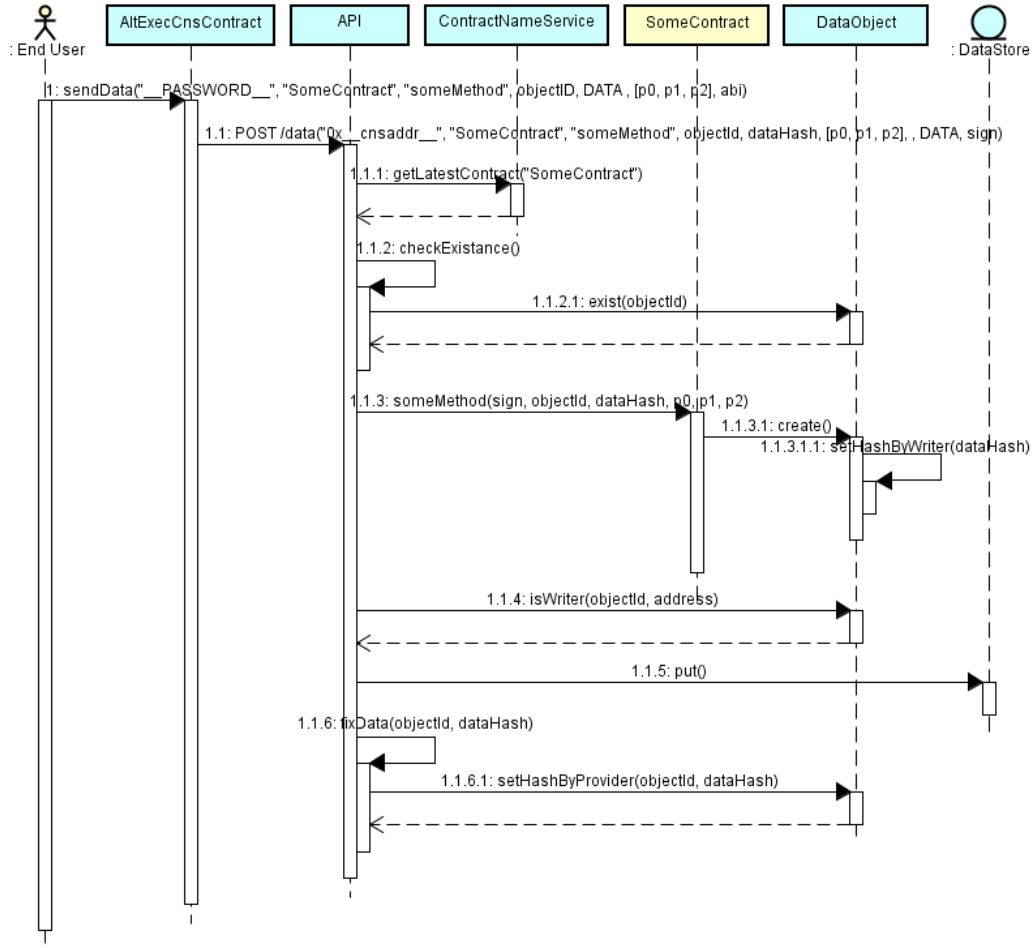


Рисунок 2.26 – Діаграма послідовності торгової платформи

Для відображення коректної й актуальної інформації, про стан інвестиції й обсяг отриманих токенів сервер буде використовувати функції, що реалізовані в нашій API-PRIVATE. Сам модуль API-PRIVATE складається з сімох класів (checkMoneyLib, fillPaymentAddressesLib, fillTokenHolderLib, getMoneyLib, getTokenBalanceLib, getTokenLib, transferToknLib), кожен з яких виконує свої функції, що разом роблять можливим правильне функціонування програми.

Всі класи з початку включають бібліотеки для можливості подальшого доступу до функцій, які використовує web-додаток для функціонування. Кожна бібліотека має великий набор функцій, що допомагають програмісту спростити реалізацію того, що він задумав та зменшити клопоти по написанню деяких функцій, які є дуже популярними.

Клас fillPaymentAddressesLib виконує підключення до локальної ноди мережі Blockchain Ethereum, після чого генерує адреси гаманців, сканує усі адреси гаманців в мережі Blockchain Ethereum, на їхню валідність і визначає їхні початкові статуси в середині нашого серверу. В разі підтвердження валідності гаманця і те, що він не був виданий в минулому, записуємо до бази даних й оновлюємо статус гаманця.

Клас checkMoneyLib при реєстрації користувача викликає методи, що були реалізовані в класі fillPaymentAddressesLib, після чого видає гаманець користувачеві. Також реалізована логіка, яка сканує усі адреси виданих гаманців, на наявність монет ефіру**.** В разі отримання інформації про те, що на гаманці є монети, скрипт викликає клас getMoneyLib.

Клас getTokenBalanceLib викликає методи, що були реалізовані в класі fillPaymentAddressesLib, а також реалізована логіка, яка сканує усі адреси виданих гаманців, на наявність монет стандарту ERC-20. В разі отримання інформації про те, що на гаманці є монети, скрипт отримує доступ до гаманця і починає процес виведення монет на адресу основного гаманця, шляхом виклику скриптів класу getTokenLib.

Клас getMoneyLib сканує гаманці, на які прийшли інвестиції, отримує доступ до даних гаманців і починає процес виведення монет на адресу основного гаманця, після чого оновлює їхній статус в базі даних, для коректної роботи сервера.

Клас getTokenLib сканує гаманці, на які прийшли інвестиції у вигляді монет стандарту ERC-20, отримує доступ до даних гаманців і починає процес виведення монет на адресу основного гаманця, шляхом виклику методу смарт-контракту (transfer), після чого оновлює їхній статус в базі даних, для коректної роботи сервера.

Клас fillTokenHolderLib сканує дані про суму загальних інвестиції, кожного гаманця, на протязі певного етапу й обраховує який коефіцієнт отримає кожен користувач, в залежності від суми й дати проведення інвестиції. Після чого зв'язується з crowdsale контрактом в мережі Blockchain

Ethereum, який буде розсилати токени усім учасникам, в пропорціних долях, за допомогою скриптів реалізованих в класі transferToknLib.

Клас transferToknLib підключається до crowdsale контракта в мережі Blockchain Ethereum викликає метод transfer, для перерахування токенів, які були згенеровані нашим контрактом, на адресу користувача сервісу.

Відповідно до вище описаної логіки класів ми можемо створити схему алгоритму модуля по валідації даних, а також роботі з Blockchain мережою (рис. 2.27).

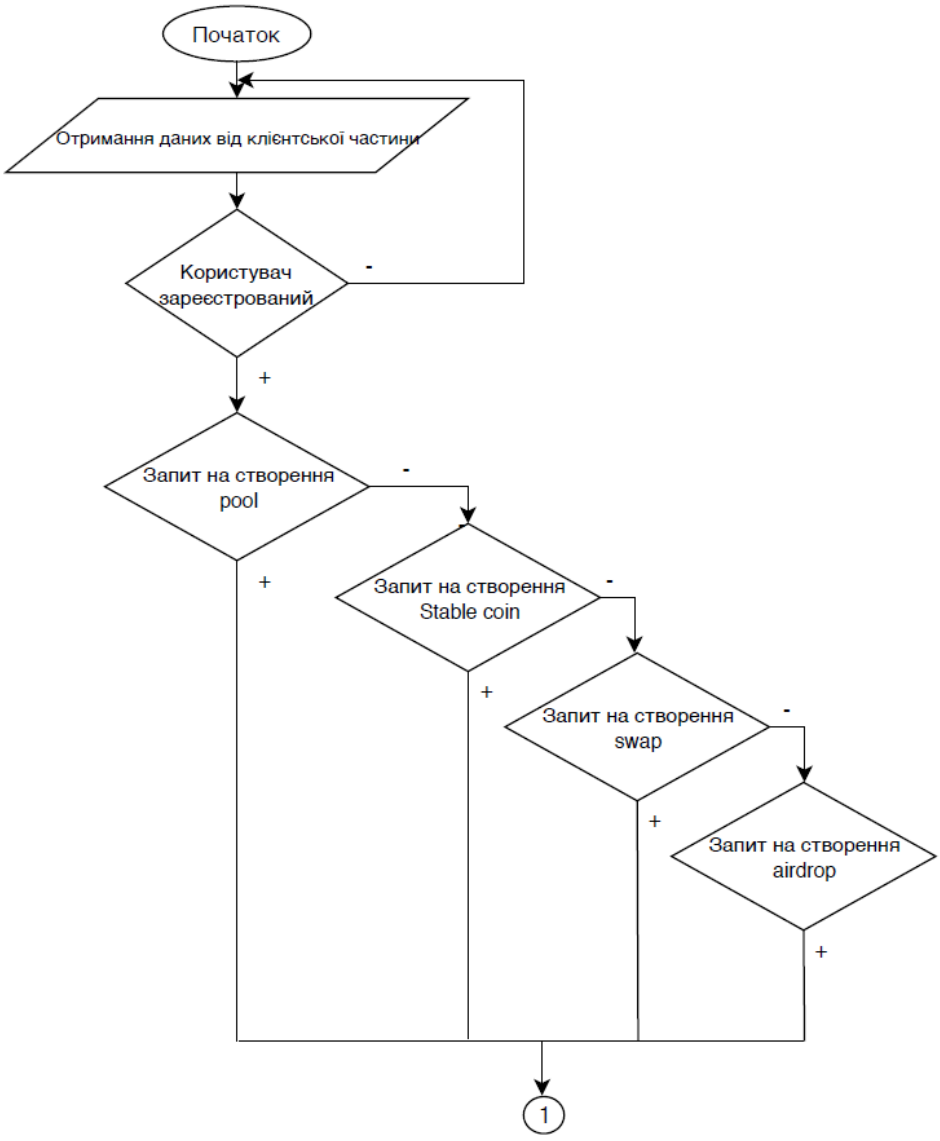
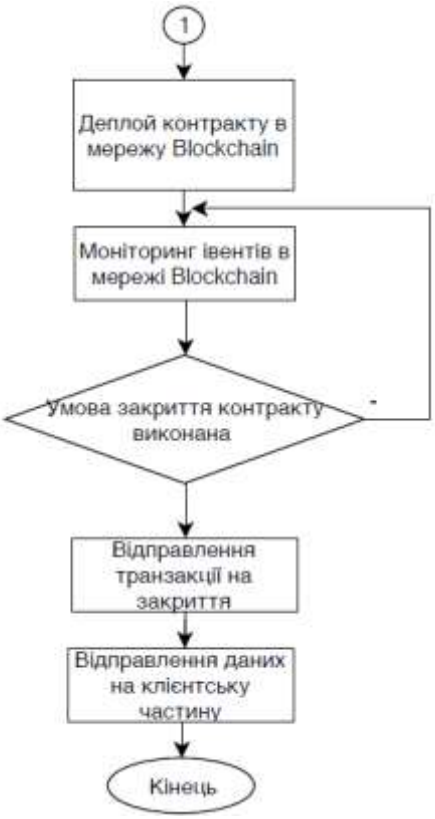


Рисунок 2.27 – Схема алгоритму модуля по роботі з Blockchain мережею по валідації даних

Продовження рисунку 2.27



2.4 Висновки

В результаті аналізу, проведеного в даному розділі, ми отримали чітке розуміння про технологої, які будуть використані в плануванні, реалізації та тестуванні програмого продукту. Було розглянуто майбутню структуру проекту, створено діаграми діяльності підключення веб-додатку до мережі блокчейн, діаграма діяльності взаємодії клієнта, серверу з мережею блокчейн, а також діаграма класів й компонентів системи.

Проведено аналіз програмних технологій, що є необіхдними для роботи з Blockchain Ethereum, Для організації взаємодії між смарт-контрактом і веб-додатком буде використана бібліотека Web3.js. Описано роботу api-private для оптимізації результатів отримання й запису інформації до мережі Blockchain. Розглянуто методи, що використовуються блокчейном для обчислень і захисту інформації від взлому: підпис, хеш-функція і еліптичні криві і т.д.

ПРОГРАМНА РЕАЛІЗАЦІЯ КРИПТОВАЛЮТНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДВИЩЕННЯ БЕЗПЕКИ РОБОТИ З ФІНАНСОВИМИ АКТИВАМИ

3.1 Обґрунтування вибору мови програмування та середовища розробки

При виборі засобів для розробки програмного проекту необхідно врахувати надзвичайно велику кількість різноманітних аспектів, найбільш важливим із яких є мова програмування, тому що вона в значній мірі визначає інші доступні засоби. При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність роботи додатку та ефективність використання пам’яті.

Web–програмування стрімко розвивається, і перед розробниками постає питання вибору між усталеними важкоатлетами Java, C, Perl і сучасними веб–орієнтованими мовами, такими як Node js, PHP. Наш вибір має величезне значення, накладаючи свій відбиток на роботу програми [17].

3.1.1 Обґрунтування вибору мови розробки Web-додатку

Node.js і Java – два дуже популярних рішення для веб–розробки.

Тим часом, коли ми говоримо про Java, ми говоримо не тільки про мову, а про віртуальну машину Java, а також повсюдно екосистему та інфраструктуру побудували навколо цієї машини. Як мінімум, їх можна порівняти за цим визнанням – як результат, у обох випадках, ми називаємо сріду виконавчих оцінок. У випадку Java – це віртуальна машина. У випадку, якщо node.js – це движок V8, який представлений у більшості ОС, таких як Windows, Linux, MacOS і трохи відомих.

Розробники можуть писати код, використовуючи одній і темний же мова, і це буде працювати більш м'яким одинаковим дизайном на різних ОС для тогочасного, що існує в середу виконавчих висновків. Середовище виконання впливає на то, як відбувається взаємодія з ОС. Крім того, їх можна

порівняти т.к. вони використовують для вирішення схожого кола задач.

Коли в v8 потрапляє код JS, виконується саме вчасно компіляція у байтовому коді, який використовується у віртуальній машині, код на JS виконується все швидше і швидше.

Байт код– це перемінний язик високого рівня, тому у віртуальній машині Java пішуть не тільки на Java, але й на Scala та Kotlin.

Якщо ви вважаєте, що зараз у найближчому майбутньому для V8 можна використовувати не тільки JS но і TypeScript або інші. На даному моменті ідеально перекладайте цих язиків в JS. У майбутньому вони, мабуть, будуть підтримувати коробки, і все буде працювати з нами, колись швидше.

Сам node.js включає в себе кілька основних складових:

- движок V8;
- бібліотеку libuv, яка відповідає за центральну частину node - цикл подій (event loop), який здійснює взаємодію з ОС, а так само за асинхронний ввід / вивід (I / O);
- набір різних JS бібліотек і безпосередньо самої мови JS.

Плюси Node js:

- легкість і швидкість написання
- легковажність
- відносна простота (в порівнянні з java)
- npm (node package manager (величезна кількість бібліотек які можуть бути встановлені в один рядок)

Мінуси Node js:

гнучкість і швидкий розвиток породжує також і мінуси тому треба постійно стежити за оновленнями, деякі речі виходять недостатньо протестованими;
був випадок, коли розробник видалив свою бібліотеку з NPM і безліч додатків використовують її перестали працювати [17];

Плюси Java:

- швидкість роботи,
- поширеність (в ВНЗ багатьох країн вивчають java, також на java

зручно вивчати ООП),

- величезний набір бібліотек.

Мінуси Java:

- ваговитість,
- деякі парадигми Java створювалися давно і вже застаріли,
- JDK пропріетарен, тому Java розвивається повільно.

Останнім часом JS починає обганяти Java (і чим далі, тим більше).Також Java йде зі світу Android, їй на зміну приходить Kotlin який хоч і використовує JVM, але все ж є іншою мовою. Java була створена компанією Sun, яка пізніше була викуплена компанією Oracle і донині належить їй. З цієї причини, для багатьох компаній використання Java створює деякі проблеми.

Порівняльна харектеристика продуктивності Java і node.js представлена на рисунку 3.1.

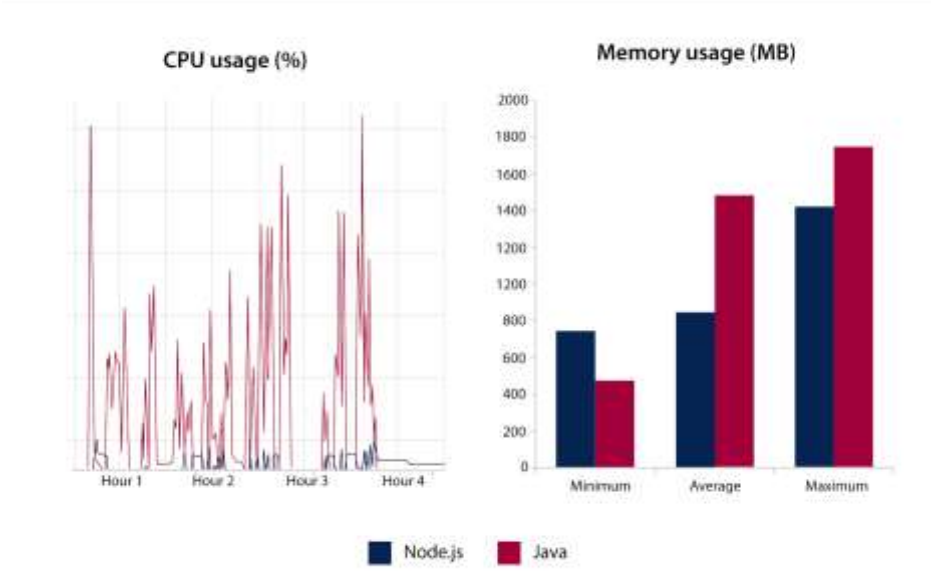


Рисунок 3.1 – Порівняльна харектеристика продуктивності Java і node.js

Якщо запустити якусь просту задачу, на кшталт зведення в квадрат, то в тестах показники можуть відрізнятися до 10 разів. Якщо запустити цикли в мільйони завдань калькуляції, Java практично завжди буде перевершувати node.js. Плюс, величезна різниця між Java і node.js в тому, що node є однопоточні, це є як його перевагою, так і недоліком з іншого боку. Java вміє

працювати з потоками, які підтримуються на рівні ОС, і виходить, що програма написана на Java найбільш повно використовує можливості ОС. І якщо потрібно написати високонавантажених додаток, яке буде використовувати велику кількість обчислень, то Java для цього однозначно підійде краще. Проблема в тому, що навіть маленький сервер написаний на Java буде займати багато пам'яті - як на диску, так і оперативної.

Node.js є легковажним за рахунок архітектури побудованої на обробці подій. Він побудований для роботи в якості веб-сервера і дуже добре справляється з обслуговуванням легковажних завдань. Наприклад, простий запит на зразок розрахунку чого-небудь, або записи в базу даних відбувається дуже швидко. А якщо запитів стає дуже багато і ми хочемо масштабувати систему в node, можна використовувати веб-сервер Nginx або Apache. Можна завести багато однакових інстанси node. Тоді все буде розподілятися через балансування навантаження по round-robin. Якщо ми запустимо 8 інстанси node на 16 ядер відповідно, ОС сама розподілить інстанси між ядрами. Node js цим не керує, у нього буде один потік [18].

В Java ми можемо створити додаток і запустити в ньому 8 потоків. За рахунок того, що відбувається більш тісний контакт з ОС, можна розподілити навантаження.

Як відомо, один з веб-серверів написаних на Java – це tomcat. Там можна чітко простежити, що коли користувач робить запит, запускаються додаткові потоки. А коли приходить запит на node, цикл подій (event loop) буде оброблений і відправлений назад, потім прийде наступний запит. І за рахунок того, що ми не чекаємо результатів першого, він теж буде підхоплений. Поки запити легковагі, все добре. Однак, коли проводиться важке обчислення, при наявності одного інстанси, node зупиняється і настає тайм-аут, рисунку 3.2.

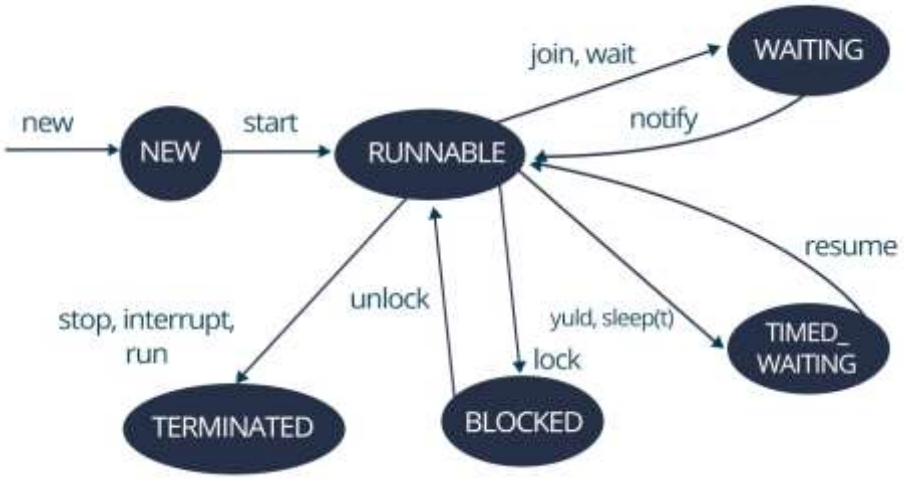


Рисунок 3.2 – Управління потоками в Java

На node можна прописати буквально кілька рядків коду і отримати найпростіший веб-сервер. Природно, для більш широкого функціоналу, де будуть нотифікації, авторизації, логирование і т.д. це складніше реалізувати, але існують фреймворки які дозволяють вирішувати такі питання, рисунку 3.3.

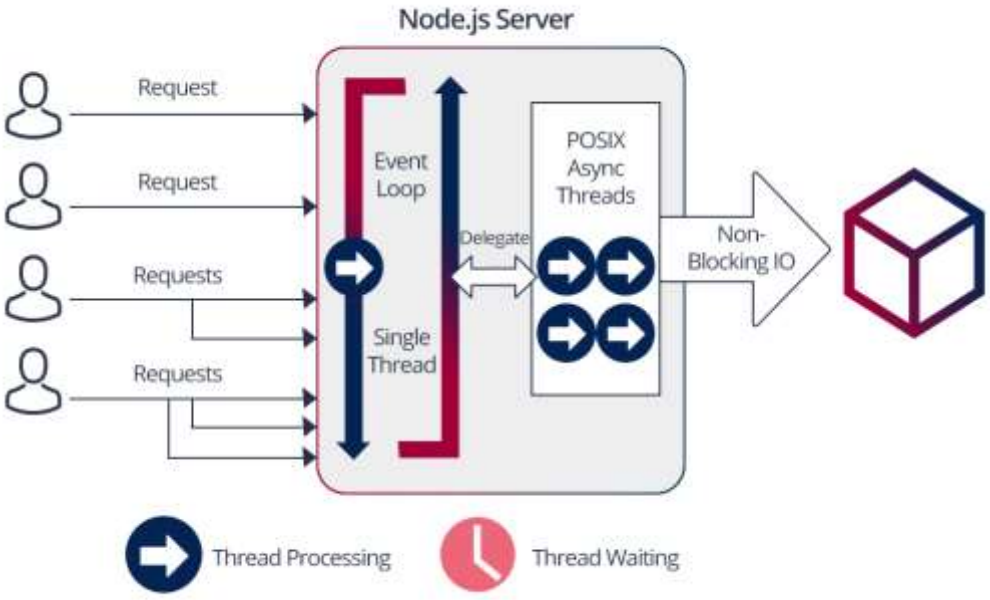


Рисунок 3.3 – Управління потоками в node.js

3.1.2 Обґрунтування вибору мови розробки смарт-контракту

Для написання смарт–контрактів на сьогоднішній день існує лише один Тюринг–повна мова Solidity. Solidity – JavaScript-подібна мова програмування для розробки розумних контрактів для EVM. Програми на Solidity транслюються в байт-код EVM. Solidity дозволяє розробникам розробляти програми, які містять логіку для створення алгоритмів проведення транзакцій в мережі Ethereum [27].

Використання синтаксису ECMAScript за задумом Вуда має допомогти прийняттю мови дійсними веб–розробниками. Однак, на відміну від ECMAScript, мова отримала статичну типізацію змінних і динамічні типи значень. Порівняно з компілюємими в такий же байт–код мовами Serpent і Mutan, мова Solidity має важливі відмінності. Підтримуються комплексні змінні контрактів, включаючи довільні ієрархічні відображення (mappings) і структури. Контракти підтримують спадкування, включаючи множинне. Підтримується бінарний інтерфейс програмування (ABI), що має безліч типобезпечних функцій в кожному контракті (пізніше з'явився також і у Serpent). Специфікована система документування коду, для пояснення послідовності викликів, що отримала назву «Специфікації на природній мові Ethereum» (Ethereum Natural Format Specification) [17].

На даний момент повноцінного релізу не було (поточна версія – 0.5.12), тому деякі особливості мови поки існують в урізаному вигляді, але незважаючи на це зі своїми задачами Solidity справляється. Solidity доступний для великого числа IDE і редакторів:

- IntelliJ IDEA ;
- Microsoft Visual Studio ;
- Atom Linker.

Solidity має наступні особливості [28]:

а) статично типізований. Це викликано необхідністю аналізу змінних перед запуском контракту, а також для передачі змінних в параметри функцій;

б) Розширений набір ідентифікаторів доступу:

- public – може бути викликана ззовні контракту;

- private – може бути викликана тільки всередині контракту;

- view – не змінює стан, а зчитує деякі дані (аналог геттерів);

- pure – функція відповідає принципам функціонального програмування без сторонніх ефектів;

- internal – доступний в контрактах - наслідниках (аналог protected);

- external – може бути викликана тільки ззовні контракту

в) змінні також мають 2 можливих ідентифікатора доступу:

- storage – змінні, які зберігаються в Blockchain ;

- memory – тимчасові змінні.

3.1.3 Обґрунтування вибору середовище розробки

WebStorm – середовище розробки для JavaScript, так само підходить як для frontend'a, так і для створення додатків на Node.js. Цей інструмент розроблено компанією JetBrains є платним.

Переваги WebStorm

- Його головною перевагою є зручний і розумний редактор JavaScript, HTML і CSS, який підтримує також і інші мови, наприклад TypeScript, CoffeeScript, Dart, Less, Sass і Stylus і фреймворки, наприклад, Angular, React і Meteor.

- WebStorm робить розробку проекту простий і зручною, забезпечуючи підсвічування і автодоповнення коду, його аналіз по ходу редагування, швидку навігацію і рефакторинг. Він має потужні інструменти налагодження та інтеграції з системами управління версіями (Git, GitHub, Subversion, Perforce, Mercurial, CVS), розуміє структуру проекту і код, відстежує помилки за допомогою систем ESLint, JSHint, JSLint, TSLint, Stylelint і пропонує їх рішення. Вбудовані в IDE інструменти для тестування і роботи з проектом допомагають в розробці і роблять її зручніше і продуктивніше [19].

У WebStorm можна ефективно розробляти програми на Node.js. Він підтримує повноцінну налагодження Node.js додатків. Новий додаток можна

створити, використовуючи шаблон Node.js Express, а необхідні модулі встановити за допомогою вбудованого в WebStorm менеджера npm.

Звичайно, через великого функціоналу WebStorm важкоатлет і вимагає багато ресурсів. Але за велику кількість булочок, стабільність роботи і взагалі приємний інтерфейс програми, можна сказати про неї, як про кращу IDE для веб–розробки.

Sublime Text – це кращий з кращих, найшвидший редактор коду. Низький вбудований функціонал з лишком компенсується величезною кількістю плагінів. Але на відміну від того ж Vim, він має низький поріг входження і людський інтерфейс. Sublime Text також можна розглядати в якості легковагій IDE.

Переваги Sublime Text

- У Sublime Text дуже приємна смуга прокрутки з превью коду, і, звичайно ж, можливість редагувати текст за допомогою кількох курсорів. Ще з фішок є підсвічування і автокомпліт практично чого завгодно, непоганий пошук по проекту, що дозволяє знайти потрібний рядок або файл всього по декількох буквах, автоматичний перенос слів по заданій ширині рядка, перевірка граматики, підтримка різних кодувань і переносів рядків, що настроюється ширина відступів , щоб вам було комфортно і зручно працювати. Ще до плюсів можна віднести гнучке налаштування шрифтів і колірних схем [16].

Недоліки Sublime Text

- До мінусів редактора можна віднести проблеми з зворотну сумісність і відсутність вбудованої консолі.

Sublime Text – легкий і неймовірно розширюваний редактор, який, при грамотному використанні, може виконувати будь–які завдання, пов'язані з редагуванням тексту, будь то код, файли конфігурацій або послання інопланетян на невідомій мові. Якщо ви цінуєте зручність роботи і вважаєте, що великовагові IDE не для вас, то Sublime Text – однозначно ваш вибір.

Visual Studio Code – багатоплатформовий редактор коду, що підтримує базові можливості інтегрованого середовища розробки (IDE), створений в Microsoft [19].

Переваги Visual Studio Code

- Позиціонується як «легкий» редактор коду для кроссплатформенной розробки веб–і хмарних додатків. Visual Studio Code поширюється безкоштовно і розробляється як програмне забезпечення з відкритим вихідним кодом.

- Visual Studio Code дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб–сайти, веб–додатки, веб–служби як в рідному, так і в керованому кодах для всіх платформ .

- У редакторі присутні вбудований відладчик, інструменти для роботи з Git і засоби рефакторинга, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішення, яке дозволяє обійтися без повної інтегрованого середовища розробки. Великим плюсом редактора є підтримка великої кількості мов, таких як C ++, C #, Python, PHP, JavaScript та інших [15].

Visual Studio Code вийшов відносно недавно і вже почав поступово набирати свою популярність. Якщо вам хочеться спробувати в цьому році щось новеньке, то варто сміливо зупинити ваш вибір на цьому редакторі.

3.2 Обґрунтування вибору програмного інструментарію

Для полегшення розробки складної системи і поєднанати різних компонентів будемо використовувати фреймворк. Фреймворк – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно – програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти.

Структура Express.js – це легка і гнучка інфраструктура додатків Node.js, яка надає широкий спектр функцій для створення односторінкових, багатосторінкових і гібридних веб–додатків. Ця структура робить створення веб–сайтів і додатків з використанням Node.js дуже простим.

Переваги Express.js, проміжного програмного забезпечення і маршрутизації спрощують процес тестування і покращують продуктивність додатків.

За допомогою потужного маршрутизації API розробники можуть виконувати завдання: від створення сервера API REST Express.js до створення маршрутів для простого веб–додатки, а потім перейти до наступного рівня з використанням параметрів маршруту і рядків запиту [16].

API Express.js також використовує вузол вузла Node.js для поширення і установки незліченних сторонніх плагінів. Ви легко можете додати інтеграцію OAuth /соціальні логіни в веб–додаток без особливих проблем, використовуючи це проміжне програмне забезпечення для аутентифікації для підключення.

Тим не менше, немає рекомендованого методу організації, який може бути проблемою для початківців і досвідчених розробників, оскільки за відсутності організації це може привести до нездійсненності проектів. Крім того, є багато ручних трудомістких завдань, які можуть вплинути на використання пам'яті, якщо вони не обробляються професійно.

Sails.js – це структура MVC, яка дозволяє створювати додатки Node.js. Він заснований на шаблоні MVC (Model–View–Controller), такому як Ruby on

Rails, але також підтримує вимоги сучасних додатків: програмні інтерфейси з масштабованої сервіс–орієнтованою архітектурою.

Завдяки ряду автоматичних генераторів він працює на більш високому рівні в стеці розробки, такому як Express.js.

Він надає безліч адаптерів бази даних Sails.js. Адаптери доступні в співтоваристві Sails.js для MySQL, MongoDB, PostgreSQL, Redis і Microsoft SQL Server. Можливість використовувати єдину структуру незалежно від фону сховища даних робить життя розробників набагато простіше.

Середній шар ORM, званий Waterline, забезпечує узгоджений синтаксис для доступу до різних баз даних, що дозволяє розробникам забувати про різні запити до баз даних. ORM може бути підключений до будь–якої базі даних або навіть до призначених для користувача веб–службам [16].

Структура Sails найкраще підходить для створення додатків з велиеими бізнес–додатками. Ця структура особливо підходить для розробки чатів, панелей інструментів в реальному часі і багатокористувацьких ігор.

Оскільки Sails.js підходить для швидкого старту проекту, швидких стартапів, які не передбачають розширення в майбутньому, додатків реального часу, де потрібно моментальна реакція, саме тому цей фреймворк було обрано для подальшої розробки.

3.3 Тестовий приклад роботи програмних модулів та аналіз результатів

3.3.1 Аналіз програмної реалізації в контексті підвищення криптостійкості

Відповідно до результатів дослідження, усі алгоритми оцінювалися за такими характеристиками, як:

- Розмір алгоритму;
- Розмір блока;
- Кількість раундів;
- Які логічні операції використовуються;
- Чи був алгоритм скомпроментований;
- Довжина, повідомлення

Отож після проведення ряду тестів отримано результати дослідження, які відображені в таблиці 3.1.

Таблиця 3.1 – Результати тестування

Алгоритм		Розмір виходу	Розмір внутрішнього стану	Розмір блока	Кількість раундів	Операції	атаки		Продуктивність на Skylake	
							Атака днів народження	Атака подовження повідомлення	Довжина повідомлення	8 bytes
MD5		128	128 (4 × 32)	512	64	And, Xor, Rot, Add (mod 232), Or	≤18 Знайдено колізії	0	4,99	55.0
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Rot, Add (mod 232), Or	≤34 Знайдено колізії	0	≈ SHA-1	≈ SHA-1
SHA-1							≤63 Знайдено колізії		3,47	52
SHA-2	SHA-224	224	256 (8 × 32)	512	64	And, Xor, Rot, Add (mod 232), Or, Shr	112	32	7.62	84.50
	SHA-256	256					128	0	7.63	85.25
	SHA-384	384	512 (8 × 64)	1024	80	And, Xor, Rot, Add (mod 264), Or, Shr	192	128 (≤ 384)	5.12	135.75
	SHA-512	512					256	0	5.06	135.50
	SHA-512/224	224					112	288	≈ SHA-384	≈ SHA-384
	SHA-512/256	256					128	256		
SHA-3	SHA3-224	224	1600 (5 × 5 × 64)	1152	24	And, Xor, Rot, Not	112	448	8.12	154.25
	SHA3-256	256		1088			128	512	8.59	155.50
	SHA3-384	384		832			192	768	11.06	164.00
	SHA3-512	512		576			256	1024	15.88	164.00
	SHAKE128	d (довільне)		1344			min(d/2, 128)	256	7.08	155.25
	SHAKE256	d (довільне)		1088			min(d/2, 256)	512	8.59	155.50

Відповідно до наведених даних можна зробити такі висновки:

- Алгоритми MD5 і SHA-1 скомпрометовані, аткаою шляхом дней народження, всього за 18 і 34 ітерації. Вони не повинні використовуватися, якщо тільки їх швидкість не в кілька разів нижче, ніж у SHA-256 або SHA-512.
- Алгоритми SHA-2, SHA-256 обчислюється з 32-бітними словами, SHA-512 з 64-бітними словами, являються також скомпрементованими, атками подовженням повідомлення.
- Найфективнішим алгоритмом являється SHA-3, для якого потрібно лише 24 раунди, що дозволить нам зменшити ціну за виконання транзакції в мережі Blockchain, рисунок 3.1. В порівняні з алгоритмом SHA-2, алгортм SHA-3 виконує на 30 % менше операцій.

Рисунок 3.1 – Кількість раундів проведених кожним алгоритмом хешування

- Агоритм SHA-3 також має найбільший розмір блока, що дозволяє обробити більшу кількість інформації, а саме 1088 біт, алгоритм SHA-2, має дещо менший показник – 1024 біт, що на 6.25 % менше, рисунок 3.2

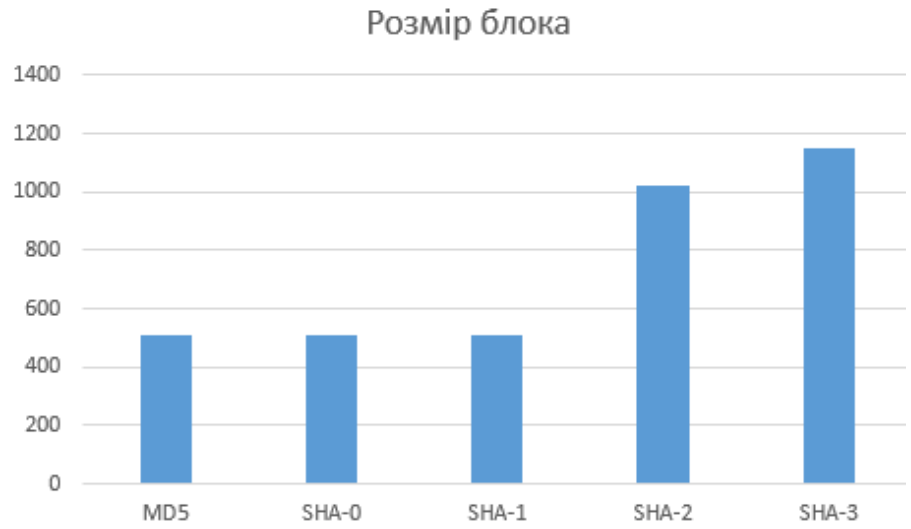


Рисунок 3.2 – Максимальна можлива кількість біт в одному блоці

- Важливим показником криптостійкості, являється чи можливо скомпрементувати дане повідомлення, тобто отримати колізії, в алгоритмі SHA-3, потрібно провести 2^{250} операцій, що на сьогоднішній день являється найкращим покащником, для алгоритма SHA-2 потрібно провести 2^{128}, рисунок 3.3

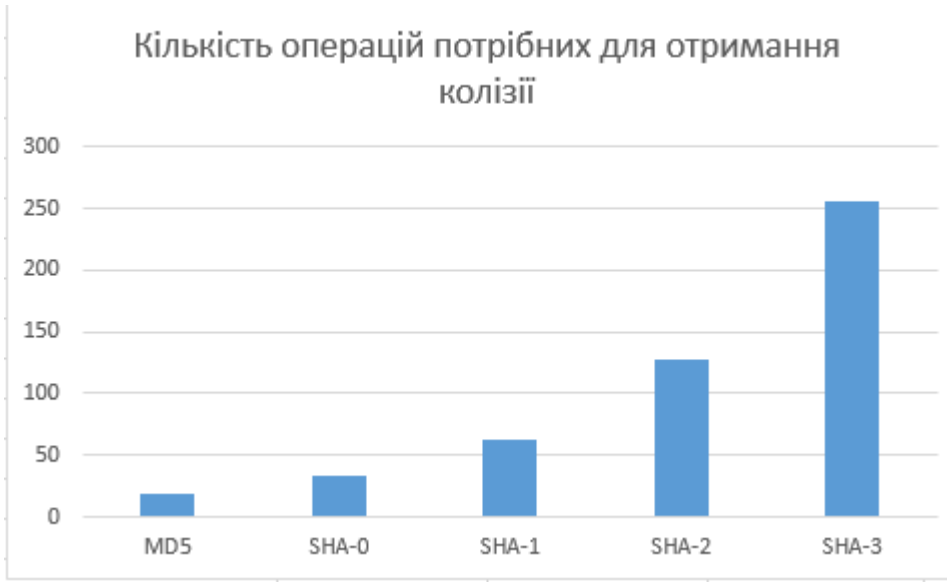


Рисунок 3.3 – Кількість операцій, для отримання колізій

Ще одним важливим показником являється час виконання кожного з алгоритмів. Отож було проведено шість тестів, кожному алгоритму подавалося повідомлення з однаковою довжиною, а саме: 32, 64, 128, 256, 512, 1024 біт, результати тестування представленні в таблиці 3.5

Таблиця 3.2 – Результати часу виконання хешування повідомлення

Hash	Case 1 (ms)	Case 2 (ms)	Case 3 (ms)	Case 4 (ms)	Case 5 (ms)	Case 6 (ms)
MD5	627.4	765.6	1488.8	839	1029.4	1738.2
SHA-1	604	748.2	1325	916.8	1009.6	1632.4
SHA-2	737.8	851	1504.4	1168.2	1260	1963.6
SHA-3	1056.4	1158.8	1837.4	1118.4	1227.4	1923

Отож, як ми можемо бачити найшвидшим алгоритмом, являється алгоритм SHA-1, з середнім часом виконання 939.16 мс, наступним по швидкодії являється алгоритм MD-5, з середнім часом виконаннння 1081.4 мс. На третому місці алгоритм SHA-2, з середнім часом виконанн 1247.5 мс, найгірший час виконання у алгоритма SHA-3 – 1386.9 мс, в порівняні з алгоритмом SHA-1, він гірший на 67.7 %.

3.3.2 Тестування і аналіз роботи програмного продукту

Наведемо результати тестування роботи web–додатку. Перед початком роботи з додатком, користувачеві потрібно мати браузерний додаток MetaMask, для спрощеної роботи по взаємодії токенів, або ефіру.

Після того як користувач, зайде на сайт, потрібно пройти реєстрацію, де необхідно ввести інформацію (електрону пошту, а також пароль від власного кабінету), рисунок 3.4

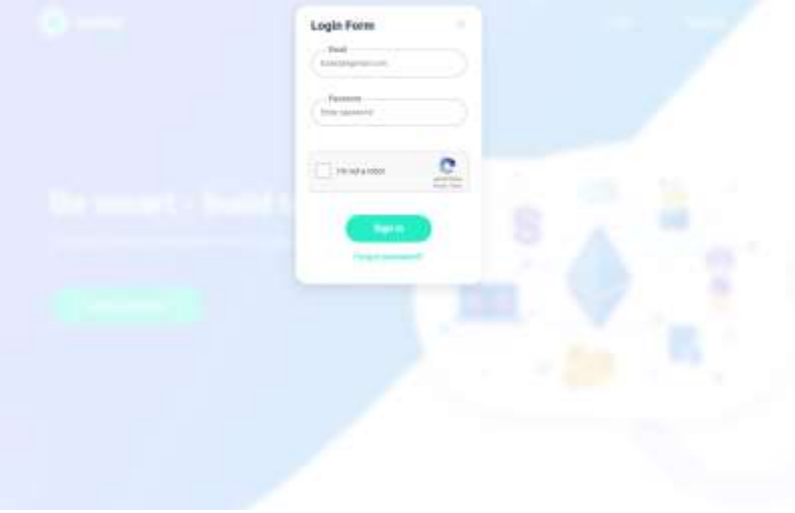


Рисунок 3.4 – Вікно реєстраційної форми

Після підтвердження реєстрації користувач матиме можливість зайти у власний кабінет і вибрати який саме тип контрактів він бажає задеплоїти (рисунок 3.5)

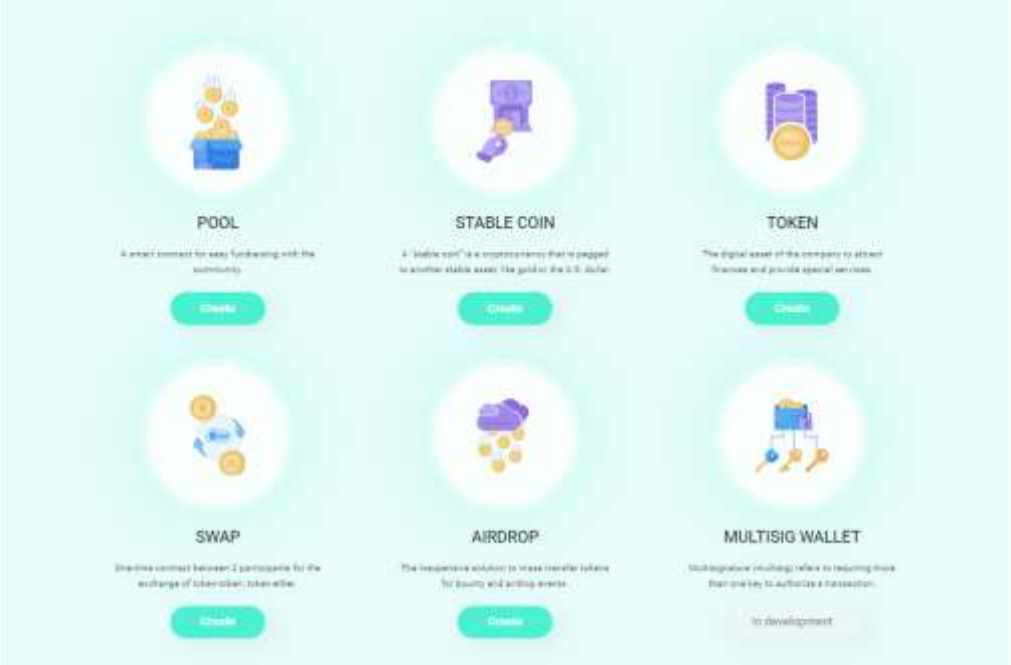


Рисунок 3.5 – Вікно форми вибору смарт-контрактів

Після вибору одного із предсталених варіантів, користувача перекине на сторінку заповнення даних, якщо вибрати варіант pool, тоді користувач побачить форму заповнення даних, яка предсталена на рисунку 3.6 – 3.8.

На цьому кроці користувач повинен заповнити такі дані, як:

- Адреса адміна, створеного контракту
- Максимальна кількість учасників
- При якій кілкості зібраних коштів важати збори успішними
- Комісія, котра відійде адміну
- Мінімальна сума, яку користувач може інвестувати
- Максимальна сума, яку користувач може інвестувати
- Назву пула, яка буде відображатися в персональному кабінеті
- Опис пула, яке буде відображатися в персональному кабінеті

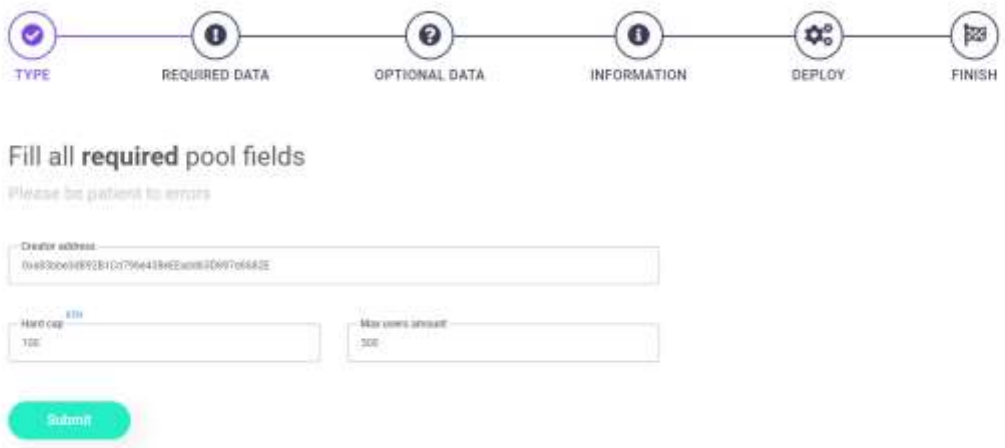


Рисунок 3.6 – Вікно заповнення даних

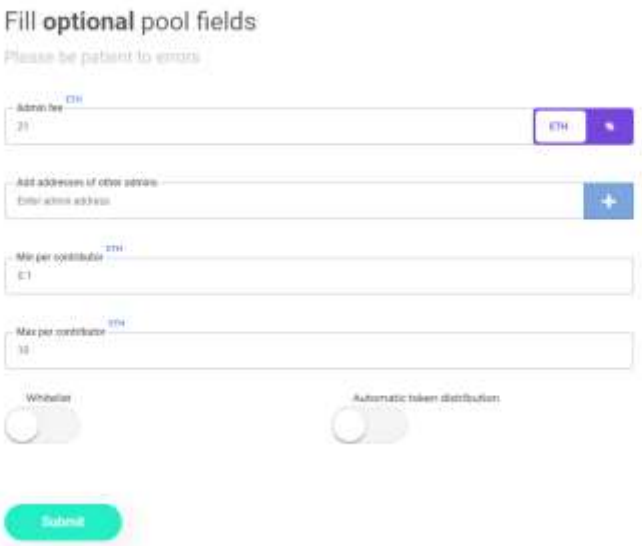


Рисунок 3.7 – Вікно заповнення даних

Після того, як всі дані заповнені, система перевіряє їх на коректність, в разі коректності відкриється вікно MetaMask, з інформацією про транзакцію, а саме скільки потрібно заплатити за розгортання даного контракту в мережі Ethereum (рисунок 3.8).

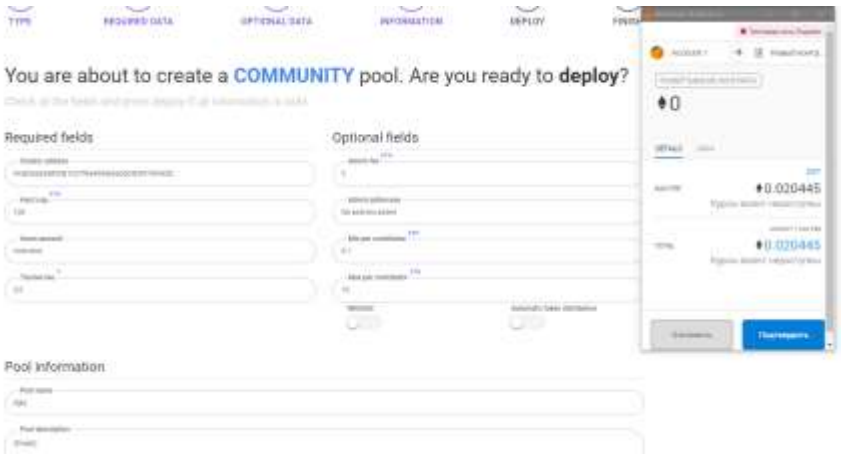


Рисунок 3.8 – Вікно заповнення даних

Після успішного розгортання контракту в мережі, уся інформація про нього зявиться в особистому кабінеті, перейшовши по вкладці my account, зображення 3.9

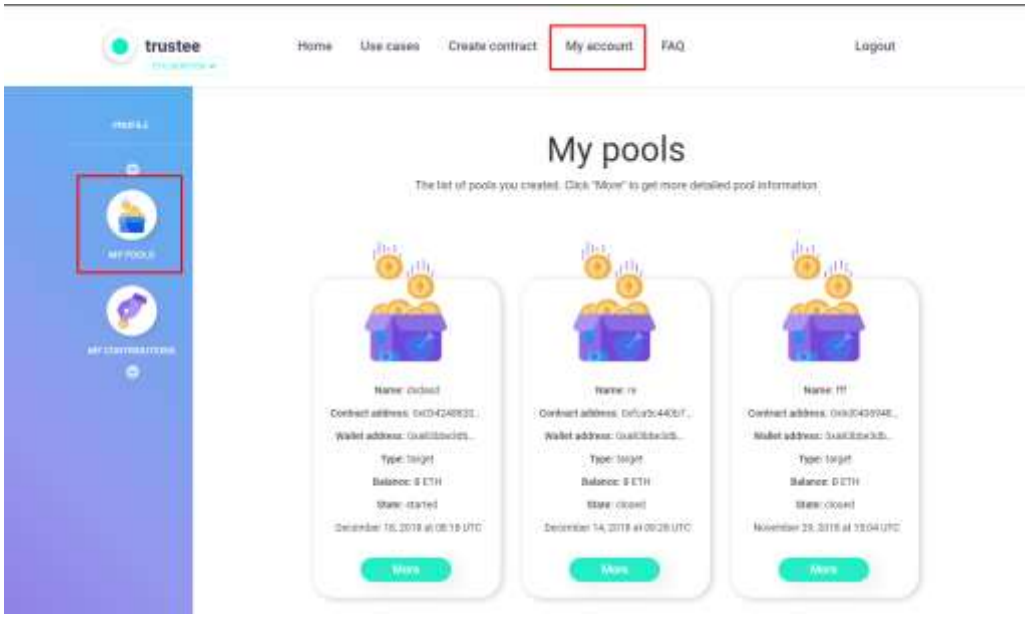


Рисунок 3.9 – Інформація про створені пули

Якщо користувач хоче отримати детальну інформацію, він повинен натиснути на кнопку More, тоді система перекине користувача на сторінку з детальною інформацією, рисунок 3.10. А саме, детальну інформацію про транзакції, депозити, адреси з яких були надіслані кошти.

Рисунок 3.10 – Детальна інформація про створений пул

3.4 Висновки

В даному розділі описано етап реалізації платформи на базі Blockchain Ethereum, а також його тестуваня.

Було провдено аналіз механізмів криптостійкості, що показав, використання алгоритми хешування SHA-3, приводить до збільшення ефективності роботи, а саме : збільшено розмір вхідного повідомлення, кількість операцій для знайдення колізій в сотні разів перевищує найближчого конкурента, але даний алгоритм являється найповільнішим з усіх представлених

Здійснено обґрунтування вибору мови програмування для розробки додатку. В результаті здійснення аналізу було обрано мови JS і Solidity.

Було протестовано розроблений додаток. При тестуванні проводилось декілька запусків web–додатке так, щоб додаток вірно проводила перекази коштів.

Web–додаток працює в режимі реального часу, процес переведення токенів відбувається на досить високому рівні у відповідності з задачами проектування. Результати, які отримано під час тестування роботи програми, відповідають результатам, які слідувало очікувати.

ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Сілагін та Бальзан.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	3
6	4	4
7	4	3
8	3	4
9	4	3
10	4	4
11	3	3
12	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 42$

Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5000	238,1	8	1904,8
Інженер-програміст	3500	166,66	45	7499,9
Всього:				9404,7

Розрахуємо додаткову заробітну плату:

$$З_{дод}= 0,1\cdot 9404,7 =940,47 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (785,7+ 7857)\cdot \frac{36,3}{100} = 3755,3 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	9000	25	3	562,5
Всього:				562,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	100	1	100
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				280,5

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,66 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0{,}8$).

$$В_е= 1{,}66 \cdot 0{,}6 \cdot 200 \cdot 0{,}8 = 163{,}2 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (9404,7 + 940,47) = 10345,17 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 9404,7+940,47+3755,3+562,5+280,5+163,2+10345,17 = 25481,54 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{25481,54}{1} = 25481,54$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{25481,54}{0,9} = 28279,82 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року

– на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1000 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 30 \cdot 1000 + (400 + 30) \cdot 200 = 116000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150) = 180500 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150 + 100) = 223500 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

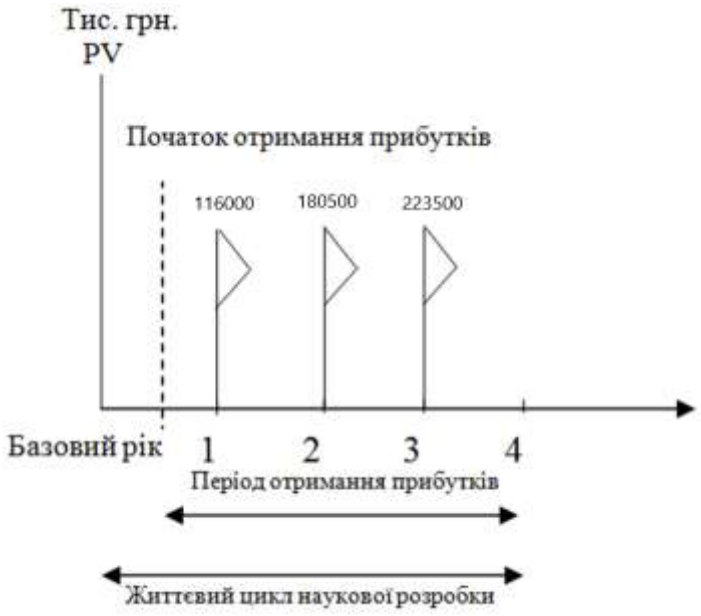


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{28279,82}{(1+0,1)^0} + \frac{116000}{(1+0,1)^2} + \frac{180500}{(1+0,1)^3} + \frac{223500}{(1+0,1)^4} = 412413,4 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 412413,4 - 28279,82 = 384133,58$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{384133,58}{28279,82}} - 1 = 1,44 \text{ або } 144\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 144% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{1{,}44} = 0{,}69 \text{ років}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновок

Під час виконання економічної частини магістерської кваліфікаційної роботи на основі обчислень було доведено, що розробка криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами є доцільною та економічно обґрунтованою.

Проведено аудит із залученням експертів у даній сфері розробки, який показав, що рівень комерційного потенціалу розробки є високим.

У даному розділі обчислено витрати на заробітну плату, на амортизаційні відрахування, витрати на комплектуючі, на силову електроенергію. Загальна вартість яких склала 28 279,82 грн.

Також спрогнозовано деякі комерційні ефекти від реалізації результатів розробки. Прогнозується, що протягом 3-х років буде отримано прибуток. Впровадивши розробку, приведена вартість всіх чистих прибутків підприємства за три роки становитиме 223 500 грн.

Також здійснено розрахунок ефективності інвестицій та періоду їх окупності. Оскільки життєвий цикл розробленого продукту становить 0.69 року.

Тому фінансування розробки є рентабельним для фінансування і може принести прибутки.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи розроблено криптовалютну інформаційну технологію підвищення безпеки роботи з фінансовими активами. Під час проведення аналізу предметної області, було визначено, що робота з фінансовими активами є дуже важливою на сьогоднішній день та її технічні рішення використовуються у багатьох сферах людської діяльності. Зокрема в роботі платіжних методів, фінансових систем, банків і т.д. Було проаналізовано існуючі задачі по роботі з фінансовими активами. Існуючі аналоги по залученню інвестицій потребують значних витрат коштів на впровадження та супроводження, а також у користувачів немає реального контролю над власними інвестиціями. Було визначено, що обраний метод робота з фінансовиим активами має високу криптостійкість, але з'являється складність роботи з Blockchain технологією.

У другому розділі проаналізовано та отримано чітке розуміння про технологої, які будуть використані в плануванні, реалізації та тестуванні програмого продукту. Мережею Blockchain було обрано Ethereum, так як на сьогоднішній день дана мережа являється найзручнішою системою для розгортання смарт-контрактів. Детально проаналізовано, як працює віртуальна машина Ethereum, а також, яким чином ми можемо підключити потрібні бібліотеки до нашого Web-додатку. Було спроектовано майбутню структуру проекту, створено діаграми діяльності підключення веб-додатку до мережі блокчейн, діаграма діяльності взаємодії клієнта, серверу з мережею блокчейн, діаграма класів й компонентів системи, ER-модель бази даних, а також схему-алгоритму роботи смарт-контрактів і Web-додатку. Проаналізовано математичні моделі криптографічних алгоритмів шифрування, що використовуються блокчейном для обчислень і захисту інформації від взлому: підпис, хеш-функція і еліптичні криві і т.д. Проведено аналіз програмних технологій, що є необіхдними для роботи з Blockchain Ethereum. Для організації взаємодії між смарт-контрактом і веб-додатком буде використана бібліотека Web3.js.

У третьому розділі проведено обґрунтування вибору мови програмування для розробки додатку. В результаті здійснення аналізу, було прийнято рішення, що смарт-контракти будуть розроблятися на мові Solidity, через те що вона має js-подібний синтаксис, а також дана мова має велике ком'юніті, яке модернізує її щодня. Для клієнтської частини було обрано мову JS, так як вона являється

доволі легкою для написання як клієнтських, так і серверних скріптів, на серверній частині буде використовуватися Node.js. Для написання сервера буде використовуватися фреймворк Sails.js, через те, що він як найкраще підходить для розробки легких, невеликих веб-додатків. Було провдено аналіз механізмів криптостійкості, що показав, використання алгоритми хешування SHA-3, приводить до збільшення криптостійкості роботи, а саме :

- Зменшено кількість раундів виконання хешування на 30% в порівнянні з алгоритмом SHA-2.
- збільшено розмір вхідного повідомлення на 6.25%
- кількість операцій для знайдення колізій 2^{250}, в порівняні зайпоширенішим алгоритмом хешування, у якого даний показник 2^{125}.

Єдиним недоліком знайденим при тестувані, являється збільшення часу роботи алгоритму на 67.7%

Web–додаток працює в режимі реального часу, процес переведення токенів відбувається на досить високому рівні у відповідності з задачами проектування. Результати, які отримано під час тестування роботи програми, відповідають результатам, які слідувало очікувати.

У четвертому розділі на основі обчислень було доведено, що розробка криптовалютної інформаційної технології підвищення безпеки роботи з фінансовими активами є доцільною та економічно обґрунтованою. Проведено аудит із залученням експертів у даній сфері розробки, який показав, що рівень комерційного потенціалу розробки є високим. У даному розділі обчислено витрати на заробітну плату, на амортизаційні відрахування, витрати на комплектуючі, на силову електроенергію. Загальна вартість яких склала 28 279,82 грн. Також спрогнозовано деякі комерційні ефекти від реалізації результатів розробки. Прогнозується, що протягом 3-х років буде отримано прибуток. Впровадивши розробку, приведена вартість всіх чистих прибутків підприємства за три роки становитиме 223 500 грн. Також здійснено розрахунок ефективності інвестицій та періоду їх окупності. Оскільки життєвий цикл розробленого продукту становить 0.69 року. Тому фінансування розробки є рентабельним для фінансування і може принести прибутки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. КОНФЕРЕНЦІЇ ВНТУ електронні наукові видання, XLVII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2019) [Електронний ресурс]. — Режим доступу:onferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/ view/2099

2. «ІНТЕРНЕТ-ОСВІТА-НАУКА-2019», Одинадцята міжнародна науково-практична конференція ІОН-2019, 22-25 травня, 2019 : Збірник праць. – Вінниця : ВНТУ, 2019 –343 с.

3. Райордан Р. ICO: сущность, проблемы, закон/Пер. с англ. – М.: Издательско–торговый дом “Русская редакция”, 2015 – 384 с.

4. Ashton K. That Internet of Things / K. Ashton // Thing. RFID Journal, 22 July 2009. [Electronic resource]. – Mode of access http://www.rfidjournal.com/articles/view?4986.

5. Gartner Says 6.4 Billion Connected "BLOCKCHAIN”, Up 30 Percent From 2015. [Electronic resource]. – Mode of access http://www.gartner.com/newsroom/id/3165317.

6. Shancang Li. ethereumbook/ Li Shancang, Li Da Xu, and Shanshan Zhao // Information Systems Frontiers 2015, 17.2. – Pp. 243–259.

7. Early Crowdfunding Player Indiegogo Brings On Famous New Investors – Venture CAPI-privatetal Dispatch – WSJ /Пер. с англ. – М.: Издательский дом “Вильямс”, 2017 – 1440 с.

8. Whitmore Andrew. The Internet of Things – A survey of topics and trends / Whitmore Andrew, Anurag Agarwal, and Li Da Xu // Information Systems Frontiers 17.2, 2015. – Pp. 261–274.

9. *Фаулер М., Скотт К.* UML. Основы. — Пер. с англ. — СПб: Символ–Плюс, 2002. — 192 с., и

10. Satoshi Nakamoto. Bitcoin: A Peer–to–Peer Electronic Cash System. [Electronic resource]. – Mode of access https://bitcoin.org/bitcoin.pdf.

11. Christidis Konstantinos, Michael Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. [Electronic resource]. – Mode of access

http://ieeexplore.ieee.org/iel7/6287639/6514899/07467408.pdf?arnumber=7467408.

12. Brody, Paul. Device democracy: Saving the future of the Internet of Things / Paul Brody, Pureswaran Veena // IBM, September, 2014.

13. Veena P. Empowering the Edge–Practical Insights on a Decentralized Internet of Things. Empowering the Edge–Practical Insights on a Decentralized Internet of Things / P. Veena, S. Panikkar, S. Nair, P. Brody.

14. Boohyung Lee. Blockchain–based secure firmware update for embedded devices in an Internet of Things environment / Lee Boohyung, Lee Jong–Hyouk. The Journal of Supercomputing, 2016. – Pp. 1–16.

15. Ferrer E.C. The blockchain: a new framework for robotic swarm systems. arXiv preprint arXiv:1608.00695, 2016.

16. Н. Коблиц Введение в эллиптические кривые и модулярные формы = Introduction to Elliptic Curves and Modular Forms. — Новокузнецк: ИО НФМИ, 2000. — С. 312.

17. Мельник А.О. Кіберфізичні системи: проблеми створення та напрями розвитку // Вісник Національного університету "Львівська політехніка". – Сер.: Комп'ютерні системи та мережі. – Львів : Вид–во НУ "Львівська політехніка". – 2014. – № 806. – С. 154–161.

Andreas M. Antonopoulos. Mastering Bitcoin: unlocking digital cryptocurrencies. "O'Reilly Media, Inc.", 2014. – 298 p.

18. Сэвитч У. С в примерах: Москва: ЭКОМ, 1997. 736с.

Шилдт Г. Самоучитель С: Санкт-Петербург: BHV-Санкт-Петербург, 1998. 620с.

19. Васильев А. Н. Java. / Объектно-ориентированное программирование: учебное пособие : базовый курс / Васильев А. Н.; . – Санкт-Петербург [и др.]: Питер, 2014. – 396 с.: ил. – ISBN 978-5-496-00044-4.