Fundamental and applied researches
in practice of leading scientific schools

ISSN 2313-7525

journal homepage: http://farplss.org

Fund&Appl Researches

Blockchain technology in the workflow system of higher education institutions

V. Vasiuta
ORCID 0000-0001-6209-1129

M. Buniak
ORCID 0000-0002-8457-6431

Poltava National Technical Yuri Kondratyuk University, Poltava, Ukraine

Article info

Accepted 28.02.2019

Poltava National Technical Yuri Kondratyuk University, Poltava, Ukraine

36011, Poltava, Pershotravnevyi avenue, 24, Poltava, Ukraine
Vasuta_V_B@meta.ua
+380953036940

Vasiuta, V., Buniak, M. (2019). Blockchain technology in the workflow system of higher education institutions. *Fundamental and applied researches in practice of leading scientific schools, 31 (1), 229–238.*

The article presents the results of a study of the theoretical foundations of the Blockchain technology. The analysis of the organization structure of its units and the principles of the system. Algorithms of smart contracts are considered and their advantages and disadvantages are analyzed. The purpose of the study is that based on the analysis of the theoretical foundations of building systems using blockchain technology and introducing them into various areas, to propose a system model using smart contracts on the Etherium platform for use in the workflow system in institutions of higher education. Convenient web interfaces for managing the creation of a contract and a module for the interaction of smart contracts with a Web application have been developed.

The management of an institution of higher education, namely, a department, faculty or institute, involves the correct and timely preparation of various documentation, in which the documentation on student learning outcomes holds a special place. On its basis various financial and reporting documents are being prepared. To this end, institutions of higher education are introducing workflow system and office automation systems to manage the processing of documents, as well as to organize control over the implementation of these processes. The paper proposes the introduction of a blockchain system on the Ethereum platform using smart contracts written in the Solidity language. Solidity is a tool that is used to create machine-level code that can be run on an Ethereum Virtual Machine. Solidity allows you to create and execute a smart contract without requiring centralized or trusted parties. The use of such technologies makes it possible to significantly reduce the number of paper-based information carriers and, to a significant extent, further automate the existing the workflow system in institutions of higher education.

The proposed system is a part of the workflow system of institutions of higher education, which is designed to automate the process of passing documents related to taking into account and submitting to the dean's office of the faculty or the institute the results of the final controls on academic disciplines or the results of retake of tests and exams. This process requires a large number of documents on paper. The proposed system is deployed in the local computer network of the Educational and Scientific Institute of Information Technologies and Mechatronics of Poltava National Technical Yuri Kondratyuk University and is being tested. The results allow us to draw conclusions about the reduction in the number of paper documents; time to obtain permits. In addition, the level of reliability and protection of information transmitted to the administration of the faculty or institute for further processing has increased. In the future, it is planned to introduce the proposed system for document flow automation for all institutes and faculties of the Poltava National Technical Yuri Kondratyuk University.

Key words: blockchain; smart contract; Ethereum, Soliditiy4 Cryptocurrenc; WEB application.

Васюта, В., Бунак М. (2019). Технологія Blockchain в системі документообігу закладу вищої освіти: Fundamental and applied researches in practice of leading scientific schools, 31 (1), 229–238.

Технологія Blockchain в системі документообігу закладу вищої освіти

В. Васюта
М. Бунак

Полтавський національний технічний університет імені Юрія Кондратюка, Полтава, Україна

У статті представлені результати дослідження теоретичних основ технології blockchain. Проведений аналіз структури організації її блоків та принципів роботи системи. Розглянуто алгоритми смарт-контрактів і проведено аналіз їх переваг та недоліків. Мета дослідження полягає у тому, що на основі проведеного аналізу теоретичних основ побудови систем з використанням блокчейн-технології запропонувати модель системи з використанням смарт-контрактів на платформі Etherium для використання в системі електронного документообігу закладів вищої освіти. Розроблено зручний веб-інтерфейс для керування створенням контракту та модуль взаємодії смарт-контрактів з Веб-додатком.

Управління закладом вищої освіти, а саме, кафедрою, факультетом чи інститутом передбачає правильне і своєчасне складання різної документації. На її підставі готуються фінансові та звітні документи. З цією метою у закладах вищої освіти впроваджуються системи електронного документообігу та автоматизації діловодства для управління процесами обробки документів. В роботі пропонується впровадження блокчейн-системи на платформі Ethereum з використанням смарт-контрактів, написаних на мові Solidity. Використання таких технологій дозволяє в значній мірі скоротити кількість паперових носіїв інформації та в значній мірі додатково автоматизувати існуючі системи документообігу в закладах вищої освіти.

Запропонована система уявляє собою частину електронного документообігу закладів вищої освіти, яка призначена для автоматизації процесу проходження документів, пов'язаних з обліком та поданням в деканат факультету чи дирекцію інституту результатів підсумкових контролів з навчальних дисциплін або результатів перескладання заліків та екзаменів. Запропонована система розгорнута в локальній комп'ютерній мережі Навчально-наукового інституту інформаційних технологій та механотроніки Полтавського національного технічного університету імені Юрія Кондратюка та проходить тестування. Отримані результати дають можливість зробити певні висновки про зменшення кількості паперових документів, часу для отримання дозвільних документів, підвищення достовірності та захищеності інформації, яка передається до адміністрації факультету чи інституту для подальшої обробки.

Ключові слова: блокчейн; смарт-контрак; Ethereum; Soliditiy; криптовалюта, Веб-додаток.

Вступ

Сучасний розвиток суспільства тісно пов'язаний з розвитком інформаційних технологій. У наш час інформаційні технології розвиваються стрімкими темпами. Увагу фахівців все більше привертає саме технологія blockchain. Десять років тому мало хто мав уявлення про таке явище у світі інформаційних технологій. Криптовалюта це одна з можливостей blockchain. На даному етапі розвитку blockchai-систем фахівці в IT галузі намагаються приділити більше уваги різним можливостям, які можна реалізувати через blockchain. На даний час смарт-контрактами цікавляться представники різних сфер людської життєдіяльності: банківської сфери, великого та малого бізнесу, закладів освіти та інші.

Мета статті

На основі проведеного аналізу теоретичних основ побудови систем з використанням блокчейн-технології та впровадження їх в різних сферах запропонувати модель системи смарт-контрактів для використання в системі документообігу закладів вищої освіти. Предметом дослідження є система смарт-контрактів на платформі Ethereum для не комерційного використання. Для досягнення зазначеної мети необхідно вирішити наступні задачі:

- розглянути поняття та можливості блокчейн-систем Bitcoin та Ethereum у криптовалютах та смарт-контрактах;
- проаналізувати дві найвідоміші блокчейн-системи для отримання інформації для подальшої розробки моделі;
- змоделювати систему смарт-контрактів на платформі Ethereum.

Результати та обговорення

Смарт-контракт – комп'ютерний протокол, основною метою якого є спрощення, верифікація та забезпечення умов виконання договору на програмному рівні, був запропонований вченим Ніком Сабо у 1996 році [1]. Можна сказати, що йдеться про врегулювання відносин сторін у формі певного коду, який придатний для зчитування комп'ютером. Процес проведення

комерційних та некомерційних операцій використовує у своїй основі поняття криптографічної довіри між сторонами, котра була описана Сатоші Накамото [2].

Blockchain – це послідовний набір блоків (або ж, в більш загальному випадку, орієнтований граф), і кожен наступний блок включає значення хеш-функції від попереднього блоку. Технологія blockchain використовується для організації журналів транзакцій, при цьому під транзакцією може розумітися що завгодно: фінансова транзакція, аудит подій аутентифікації і авторизації, записи про виконані роботи у компанії. При цьому подія вважається такою, що відбулася, якщо запис про неї включений в журнал.

Учасники завжди вважають оригінальною найдовшу версію blockchain і працюють над її подовженням. Якщо два вузла одночасно опублікують різні версії чергового блоку, то хтось із учасників отримає одну версію, а хтось – іншу. У такому випадку кожен почне працювати над своєю версією ланцюжка, зберігаючи іншу на випадок, якщо вона виявиться продовжена раніше. Двоїстість ланцюга зникне, як тільки буде отримано новий блок, який продовжить будь-яку з гілок, і ті вузли, що працювали над «конкуруючою» версією, переключаться на неї.

Сатоші Накомото запропонував систему електронних транзакцій, яка не заснована на довірі. Сильною стороною мережі є простота її структури. Всі вузли працюють самостійно, іноді обмінюючись інформацією. Немає необхідності в ідентифікації, оскільки повідомлення не йдуть по якомусь певному маршруту, а засновані на принципі «найменших витрат». Вузли можуть залишати мережу і знову підключатися, приймаючи найдовший ланцюжок блоків як підтвердження пропущеної історії транзакцій. Вони висловлюють свою згоду прийняти коректний блок в ланцюжок, використовуючи свої обчислювальні потужності для подовження цього ланцюга, або незгоду (якщо блок містить невірні дані), щоб не продовжувати цей ланцюжок. Будь-які правила протоколу можуть бути реалізовані через цей «механізм голосування» [2].

Як було зазначено вище, технологія blockchain використовується для організації журналів транзакцій, як комерційних так і не комерційних.

Такі системи мають три групи осіб:

1. Джерела подій (транзакцій);
2. Джерела блоків (фіксатори транзакцій);
3. Одержувачі (читачі) блоків і зафіксованих транзакцій.

Розглянемо три варіанти систем:

1. Централізований з довіреною центром;
2. Централізований з не довіреним центром;
3. Децентралізований варіант з використанням «доказу роботи».

Якщо є довірений центр: потрібно через певний проміжок часу (або ж через певний набір транзакцій) на програмному рівні сформувати новий блок, забезпечуючи його не тільки хеш-сумою, а й своїм електронним підписом. Кожен клієнт системи має можливість перевірити, що всі блоки в ланцюжку згенеровані саме довіреним центром. Довірений центр надає кожній транзакції відповідний час та номер, що забезпечує порядок і неможливість додавання (видалення) транзакцій з ланки і неможливість модифікації конкретних транзакцій.

Іноді користувачі звертаються до неповністю довіреного виділеного серверу. Через нього проводять фіксації транзакцій в журналі, але учасники бажають бути впевненими, що виділений центр не регенерує весь ланцюжок блоків, видаливши або додавши певні транзакції. Для цього можна використовувати, наприклад, наступні два методи.

Перший метод з використанням додаткового довіреного «сховища». Після створення чергового блоку центр повинен відправити у хеш-код від нового блоку. Це сховище не повинно приймати ніяких змін до хеш-кодів вже створених блоків. В якості такого сховища можна використовувати і децентралізовану базу даних системи, якщо така доступна в наявності. Розмір що зберігається може бути невеликим порівняно із загальним обсягом журналу.

Другий можливий метод полягає в доповненні кожного блоку міткою часу, згенерований довіреним центром тимчасових міток. Така мітка повинна містити час генерації мітки і електронний підпис центру, обчислений на підставі хеш-коду блоку і часу мітки. У разі, якщо не довірений центр захоче згенерувати частину ланцюжка блоків, з'явиться помітний розрив у мітках часу.

Найбільший інтерес являє децентралізована система blockchain без виділених центрів генерації блоків. Кожен учасник може взяти набір транзакцій, які очікують включення в журнал, і сформувати новий блок. Більш того, в системах типу BitCoin такий учасник (будемо його назвати «майнер») ще й отримає премію у вигляді певної суми або комісійних від прийнятих в блок транзакцій. Надійність таких систем ґрунтується саме на тому, що новий блок не можна сформувати швидше (в середньому) ніж за певний час. Це забезпечується механізмом, який отримав назву «proof-of-work», – доказ роботи.

У блокчейні Ethereum використовується вбудований цифровий токен, який носить назву «ефір» (від англ. Ether- «ефір»). Кожен раз, коли майнер обґрунтовує свій блок транзакцій, створюється новий токен або новий ефір, а майнер отримує винагороду за його створення. У системі не буває двох і більше коректних поточних станів. Щоб визначити, який з можливих шляхів є коректним і запобігти утворенню багатьох ланцюгів, в Ethereum застосовується метод протоколу GHOST – «Greedy Heaviest Observed Subtree». GHOST оголошує, що правильним блокчейном є той, на якому було виконано найбільше число обчислень. Завдяки такому підходу можна визначити загальне число блоків, що знаходяться у блокчейні.

Стан платформи Ethereum складається з великої кількості невеликих об'єктів – облікових записів, які взаємодіють між собою за рахунок парадигми обміну повідомленнями. У кожного облікового запису є певний стан і 20-байтова адреса. Адресою в Ethereum є 160-бітний ідентифікатор, який використовується для ідентифікації будь-яких облікових записів.

Всього існує два види облікових записів:

1. Зовнішні облікові записи, які контролюються закритими ключами;

2. Контрактні облікові записи, які контролюються спеціальним кодом, зазначеним в умовах контракту, і мають пов'язаний з ними код.

Для зовнішнього облікового запису передбачена можливість відправляти повідомлення іншим зовнішнім обліковим записам та контрактним обліковим записам. Для цього необхідно створити і зареєструвати нову транзакцію, використовуючи закритий ключ. Сполучення між двома зовнішніми обліковими записами є всього лише значенням для передачі. З іншого боку, повідомлення, відправлене від зовнішнього облікового запису до контрактного, є активацію коду контрактного облікового запису, при цьому з'являється можливість здійснення певних дій [3, 9].

За допомогою контрактних облікових записів самостійно ініціювати нові транзакції неможливо. Замість цього за допомогою контрактних облікових записів можна тільки запускати транзакції у відповідь на інші отримані транзакції.

Кожна дія в блокчейні Ethereum відбувається завдяки транзакціям, що ініціюються зовні контрольованими обліковими записами.

Стан кожного з облікових записів, незалежно від їх типу, може приймати одне з чотирьох значень:

1. nonce: якщо обліковий запис відповідає типу зовнішнього облікового запису, то отримане число являє собою кількість транзакцій, які були проведені з заданої адреси облікового запису. Якщо обліковий запис є контрактним обліковим записом, то елемент nonce - це кількість контрактів, створених конкретно цим обліковим записом;

2. balance: загальна кількість wei, якою володіє власник облікового запису;

3. storageRoot: хеш кореневого вузла префіксного хеш-дерева Меркла. Дерево Меркла кодує хеш вмісту цього облікового запису, при цьому за замовчуванням воно є порожнім.

4. codeHash: хеш ETHEREUM VIRTUAL MACHINE-коду (від англ. Ethereum Virtual Machine) облікового запису. Для контрактних облікових записів на цьому полі є кодом, який хешується і зберігається у вигляді codeHash.

Глобальний стан Ethereum – це зіставлення адрес облікового запису станів рахунку, що зберігається в структурі даних – префіксного дерева Меркла (рис.1).

Дерево Меркла (або «Merkle tree») являє собою тип файлу, який виконується та складається з набору вузлів, які включають:

1. Певну кількість вузлів, які розташовані в нижній частині дерева, що містить базові дані;

2. Набір проміжних вузлів, при цьому кожен вузол являє собою хеш двох його дочірніх вузлів;

3. Один кореневий вузол, також утворений з хеш двох дочірніх вузлів, який представляє вершину дерева.

Дані, що знаходяться в нижній частині дерева, створюються шляхом поділу тих даних які потрібно зберегти на окремі фрагменти. Далі такі фрагменти розміщуються в кошиках зберігання даних, після чого відбувається їх хешування і аналогічний процес повторюється до тих пір, поки загальне число хеш не дорівнюватиме одиниці або кореневого хешу. Для кожного значення, що зберігається всередині даного дерева потрібно ввести певний ключ. Для отримання відповідного значення, що зберігається в вузлі необхідно отримати команду ключа.

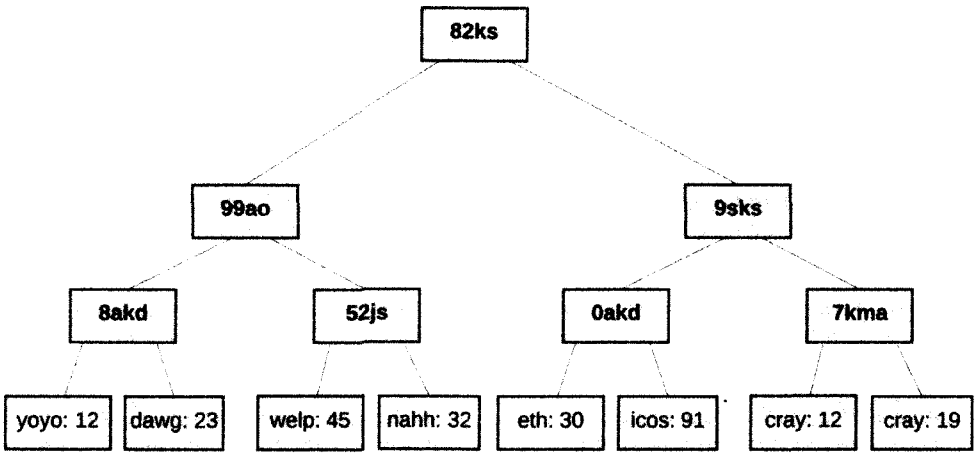

Рис. 1. Вигляд хеш-дерева Меркла

Що стосується Ethereum, то відображення ключа необхідного для дерева станів, знаходиться між адресами і пов'язаними з ними обліковими записами, в тому числі balance, nonce, codeHash, а також storage для кожного з облікових записів, при цьому storage є деревом (рис.2.).

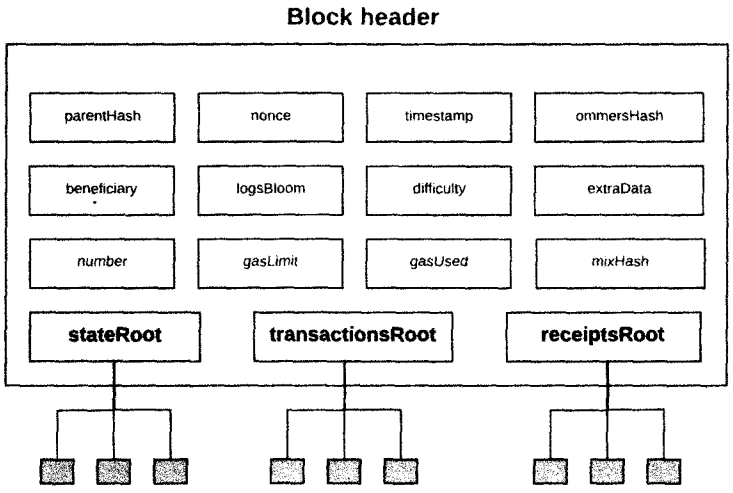

Рис. 2. Header блоку

Подібна структура префіксного дерева також може застосовуватися для зберігання, як транзакцій, так і сторінки прийому оплати. Якщо зупинятися на цьому більш детально, то кожен блок має так званий «header» або заголовок, в якому зберігається хеш кореневого вузла трьох різних структур дерева Меркла, в тому числі:

1. Стан префіксного дерева
2. Транзакції префіксного дерева
3. Сторінки прийому оплати для префіксного дерева

Можливість ефективного зберігання даної інформації в префіксному дереві Меркла є практичним рішенням для зберігання інформації у повних та неповних вузлах.

Повний вузол архіву виконує синхронізацію блокчейну за допомогою завантаження всієї ланки блоку від початкового стану до поточного, що містить заголовки, при цьому виконуючи транзакції, що розташовані в ньому. Як правило, майнери зберігають повний вузол архіву, оскільки без останнього у них не буде можливості брати участь в самому процесі майнінгу. Крім того, будь-який користувач також може завантажити повний вузол, при цьому у виконанні кожної окремої транзакції немає необхідності. Кожен повний вузол завжди містить повний ланцюжок.

Таким чином, можна зробити висновок, що перевага застосування префіксного дерева Меркле полягає в тому, що кореневий вузол даної структури є криптографічно залежним від даних, що зберігаються в дереві. Отже, хеш кореневого вузла може бути використаний в якості безпечного ідентифікатора для цих даних.

Одним з важливих моментів в системі Ethereum є процес оплати. За будь-яке обчислення, що здійснюється в результаті проведення операцій з транзакціями всередині мережі Ethereum існує певна плата – її номінал носить назву «пальне» (від англ. «Gas»).

Пальне – це одиниця виміру, яка використовується для визначення розміру оплати за конкретне обчислення. Ціна на пальне – це кількість «ефіру», яке ви здатні витратити на кожну одиницю пального. Вимірюється ціна на пальне в «gwei». Wei є найменшою одиницею ефіру, де 1018 Wei це всього 1 ефір. Один gwei дорівнює 1 000 000 000 Wei. Для проведення будь-якої транзакції відправник повинен встановити ліміт gas. Ціна і ліміт для неї - це максимальна сума в Wei, яку відправник готовий заплатити за проведення транзакції.

У випадку, якщо відправником не було надано необхідної кількості gas для проведення транзакції, остання буде вважатися недійсною. Таким чином, проведення транзакції переривається, а будь-які зміни стану анулюються, внаслідок чого система Ethereum повертає учасників угоди в початковий стан. Варто зазначити, що інформація про невдалі транзакції записується в системі, і завжди можна відстежити проведення яких саме транзакцій було перервано і на якому етапі стався збій.

Питання оплати за використання сховища, має деякі нюанси. Наприклад, оскільки збільшення місця в сховищі збільшує розмір бази даних станів Ethereum, і це відноситься до всіх вузлів, то це є стимулом зберігати відносно невеликий обсяг даних. Таким чином, якщо який-небудь з етапів транзакції передбачає видалення запису в сховище, то оплата за виконання цієї операції не стягується [4, 5, 8, 10].

Важливим аспектом роботи Ethereum є те, що будь-яка операція, яка виконується мережею, також одночасно виконується кожним повним вузлом. Проте, всі кроки, пов'язані з обчисленням на віртуальній машині Ethereum, дуже дорого коштують. Таким чином, для вирішення простих завдань можуть цілком згодитися смарт-контракти Ethereum, на відміну від тих випадків, коли потрібне виконання інших, більш складних, задач: зберігання файлів або електронної пошти, а також виконання завдань з області машинного

навчання, які можуть викликати надмірне завантаження мережі.

Транзакція є криптографічно підписаною частиною інструкції, яка спочатку задається зовнішньої обліковим записом, а потім упорядковується і передається в блокчейн. Всього існує два типи транзакцій: відправлення повідомлень і створення контракту в мережі Ethereum.

Усі транзакції згруповані в «блоки». Блокчейн містить кілька таких блоків, з'єднаних між собою.

Такі блоки складаються з:

1. Заголовка блоку;

2. Інформації про серію транзакцій, включених в даний блок;

3. Серії інших заголовків блоків для поточних оммерів.

Оммер – це блок, батьком якого є батьківський елемент поточного блоку. Їх наявність пов'язана з тим, що час блокування в Ethereum набагато нижче (приблизно 15 секунд), ніж для інших блокчейнів. Завдяки такій особливості швидкість проведення транзакцій збільшується. З іншого боку, однією з негативних сторін більш короткого часу блокування є те, що боротьба майнерів за чергове блочне рішення тільки посилюється.

Оммери були створені для того, щоб майнери могли отримати заслужену нагороду за включення блоків в основний ланцюжок. За Оммер можна отримати нагороду, меншу ніж за включення повного блоку.

У платформі Ethereum передбачена можливість вести журнали, мета яких записувати інформацію про різні транзакції і повідомленнях. Крім того, для контракту також існує можливість відкритого створення запису в такому журналі за допомогою оголошення «події», яке потрібно записати.

Запис журналу включає:

1. Адресу облікового запису реєстратора;

2. Ряд завдань, які відображають різні події, виконані для поточної транзакції;

3. Будь-які дані, які мають відношення до даних подій;

Смарт-контракти призначені мінімізувати участь третіх сторін у договорі, іноді повністю виключаючи центр прийняття рішень та для таких контрактів простіше проводити аудит. Під смарт-контрактом розуміють децентралізоване середовище і наявність функцій, які дозволяють проаналізувати базу даних і провести повний аудит виконання контрактів. Таким чином гарантується захист від змін даних «заднім числом», які спричинять за собою зміни у виконанні самого контракту. Оцифровка більшості процесів при створенні та запуску смарт-контракту часто спрощує технологію і вартість їх реалізації.

Для класифікації смарт-контрактів можна використовувати різні критерії[5].. На даний момент актуальними є чотири з них:

1. За середовищем виконання, яке може бути централізованим і децентралізованим. Децентралізовані контракти мають більшу незалежність і відмовостійкість при виконанні

2. По процесу завдання і виконання умов. Смарт-контракти можуть бути довільно програмован обмежено або чітко типізовані.

3. За способом ініціювання. Існують автоматизован смарт-контракти, які при виконанні певних умов виконуються автоматично та контракти для яки потрібно провести процедуру окремої ініціалізації.

4. За рівнем приватності. Смарт-контракти можуть бути: повністю відкриті; частково відкриті; повніст конфіденційні.

Для створення нової контрактного облікового запису потрібно оголосити адресу облікового запис який створюється за допомогою спеціальної формул Після цього відбувається створення нового облікового запису. Для виконання такої операції необхідно:

1. Виставити нуль в значенні nonce;

2. Налаштувати баланс облікового запису

3. Підрахувати розмір оплати;

4. Налаштувати хеш-код контракту в якості хе порожнього рядка

У момент, коли обліковий запис було створено, ін код автоматично відправляється на початку транзакці Існує декілька варіантів розвитку подій протяго виконання init-коду: оновлення сховища для облікового запису, створення іншого облікового запису для поточного контракту, відправка повідомлення і т.д.

Після виконання системою коду, який призначени для створення нового контракту, користувач не змож провести транзакцію якщо для неї потрібна кількість gas більша, ніж та, яка зберігається на балансі. Якщо ко ініціалізації успішно виконаний, то кошти, що бу необхідні для створення контракту, повинні бу внесені. У цю суму входить також вартість використання сховища, яка прямо збільшується з збільшенням розмірів створеного для контракту коду У випадку якщо коштів не вистачає для проведен операції, то транзакція припиняється.

Частина протоколу, який виконує обробк транзакцій в операційній системі Ethereum називаєтьс віртуальною машиною Ethereum (ETHEREUM VIRTUAL MACHINE). ETHEREUM VIRTUAL MACHINE, відмінність в від машини Тьюринга полягає в тому, що для її робот потрібен gas. Всі обчислення, які можуть бути виконан обмежені кількістю gas.

Для ETHEREUM VIRTUAL MACHINE передбаче певна область зберігання, таке сховище (або област зберігання) не змінюється і є частиною стану системи У ETHEREUM VIRTUAL MACHINE програмний ко зберігається в окремій. віртуальній ROM, доступ до як можна отримати за допомогою певних інструкцій. З ней точки зору така ETHEREUM VIRTUAL MACHINE відрізняється від типової архітектури фон Неймана.

Для віртуальної машини також передбаче спеціальна мова - байт-код ETHEREUM VIRTUAL MACHINE. Для створення смарт-контракту, як виконується в системі Ethereum, зазвичай використовують мову Solidity [6, 7].

До виконання обчислень процесор перевір валідність і доступність наведеної нижче інформації:

1. Стан системи;

2. Достатність коштів для виконання необхід операції;

3. Адресу облікового запису, якому належить код, який виконується;

4. Адресу відправника транзакції – ініціатора виконання поточної операції;

5. Адресу облікового запису - ініціатора коду, який виконується;

6. Інформація про достатність коштів для виконання транзакції;

7. Вхідні дані для виконання операції;

8. Кількість Wei, яка буде відправлена на рахунок цього облікового запису в результаті проведення поточної операції;

9. Інформація про машинний код, який виконується;

10. Інформація про заголовок блоку для поточного блоку;

11. Глибина виконання поточного повідомлення або створення контракту.

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

Після цього в ETHEREUM VIRTUAL MACHINE починається рекурсивне виконання транзакції: обчислення стану системи і стану машини для кожного циклу. Стан системи – це глобальний стан Ethereuma. Стан машини включає в себе:

1. Доступна кількість коштів;

2. Лічильник команд;

3. Вміст пам'яті;

4. Активна кількість слів у пам'яті;

Контент стека.

Існує три варіанти закінчення циклу:

1. Операції, що виконуються машиною, досягають виняткового стану;

2. Послідовність дій переходить до виконання наступного циклу;

3. Операції, що виконуються машиною, досягають логічного завершення.

Остаточне оформлення може відбуватися за двома варіантами, в залежності від того, створюється блок або він вже створений. У тому випадку якщо блок створюється, то остаточне оформлення означає процес Майнінг поточного блоку. З іншого боку, якщо блок уже створений, то таке визначення означає процес валідації поточного блоку. В обох з наведених вище випадках необхідно виконати чотири умови для остаточного оформлення блоку:

1. Валідація Оммерів: кожен блок Оммер, який знаходиться в заголовку блоку, повинен мати валідний заголовок блоку і бути шостим нащадком поточного блоку;

2. Валідація транзакцій: значення gasUsed для поточного блоку має бути рівним значенням загальної кількості пального, використаного для проведення всіх перерахованих в даному блоці транзакцій;

3. Призначення оплати: на адресу бенефіціарія призначається 5 одиниць ефіру за Майнінг кожного блоку (відповідно до пропозиції EIP-649 дана оплата буде зменшена до 3 одиниць ефіру). Більш того, за кожен Оммер, бенефіціар поточного блоку призначається оплата у вигляді додаткових 1/32 від загальної оплати за поточний блок. І останнє: бенефициарию блоку Оммер також призначається оплата у вигляді певної суми, для визначення якої існує спеціальна цифра.

4. Верифікація стану і значення nonce: для проведення даної процедури вам необхідно забезпечити виконання всіх транзакцій, а також зміна результуючих станів. Після чого буде потрібно задати новий блок після того, як оплата за даний блок була відправлена. Процес верифікації відбувається за допомогою порівняння завершального стану зі станом префіксного дерева, що зберігається в заголовку.

Сатоші Накамото, людина, яка вважається автором технології Bitcoin, зумів запропонувати децентралізований механізм, в якому і сама поведінка системи, і зміни до цієї системи проходять через явний або неявний механізм пошуку консенсусу між учасниками. Для отримання контролю над системою в цілому зловмисникові доведеться отримати контроль як мінімум над 50% всіх потужностей системи, а без цього можна лише спробувати обмежити можливість використання системи конкретними учасниками. Але в той же час створена технологія не позбавлена недоліків. Справа в тому, що система створена Накамото жорстко ґрунтується на криптовалютній моделі і з метою безпеки учасників мережі не може додати до функціоналу ті ж самі смарт-контракти. Існують оцінки, згідно з якими використання методу «доказ роботи» для системи Bitcoin призводить до витрат енергії, яку можна порівняти зі споживанням електрики цілими містами або країнами. Є проблеми і з пошуком консенсусу – складний механізм внесення змін. В даний час ведеться активна робота фахівців по поліпшенню системи – у 2017 році у системі Bitcoin запровадили новий протокол передачі даних SegWit2x. Він дозволив збільшити обсяг інформації, що зберігається в блоці до 2 мб.

З міркувань безпеки Bitcoin хоч і являє собою повну реалізацію моделі, яку у 1994 побудував Нік Сабо, не може дозволити собі внести в систему повний обсяг використання смарт-контрактів, тому і досі є лише поняттям кріптовалюти. Але слід зазначити, що навіть не дивлячись відсутність гнучкості системи у широкому розумінні даного поняття, у даній системі було реалізовано систему довіри, заснованої на криптографії, що дало сильній поштовх для стрімкого розвитку інших блокчейн систем кріптовалют та реалізації смарт-контрактів на базі інших платформ .

Саме тому, розробники блокчейн-системи Ethereum внесли корективи, і їх система є більш гнучкою. Смарт-контракти на платформі Ethereum вже користуються великою популярністю: купівля-продаж електронних документів, купівля-продаж нерухомості та інше.

З метою впровадження сучасних інформаційних технологій в систему управління закладом вищої освіти розроблена блокчейн-системи на платформі Ethereum з використанням смарт-контрактів написаних на мові Solidity. Використання таких технологій дозволяє в значній мірі скоротити кількість паперових носіїв інформації та в значній мірі додатково автоматизувати існуючі системи документообігу в ЗВО. Solidity - це інструмент, який застосовується для створення коду машинного рівня, який може виконуватися на Ethereum Virtual Machine. Solidity дозволяє створювати і

виконувати смарт-контракт не вимагаючи при цьому централізованих або довірених сторін.

Крок 1. Створення шаблону та функції-конструктора.

Функцію було використано для ініціалізації однієї єдиної змінної - address owner, в неї буде записана адреса того, хто створив контракт і відправив його в мережу.

```
pragma solidity ^0.4.0;
contract test {
//constructor
  function contructor() public {
    owner = msg.sender;
  }
```

Крок 2. Додавання базової інформації про автора

Додамо можливість зазначати базову інформацію про автора - ім'я, пошту, адреса і так далі. Для цього використовується звичайний mapping:

```
contract test {
  mapping (string => string) basic_data;
  address owner;
```

Крок 3. Створенння декількох структур для опису студента

Замість того, щоб описувати їх в поточному контракті, винесемо їх в окрему бібліотеку (в новому файлі), для структурування проекту.

Для цього в тій же директорії створюється новий файл structures.sol і бібліотеку Structures і в ній описується кожна з структур:

```
pragma solidity ^0.4.0;
library Struct {
  struct Project {
    string name;
    string link;
    string description;
  }
  struct Education {
    string name;
    string speciality;
    int32 year_start;
  }
  struct Exam {
    string name;
    string semister;
    string teacher;
  }
}
```

Крок 4. Імпортування структур

```
pragma solidity ^0.4.0;
import "/struct.sol";
contract test {
  mapping (string => string) basic_data;
  address owner;
  Struct.Project[] public projects;
  Struct.Education[] public educations;
  Struct.Exam[] public exams;
  function contructor() public {
    owner = msg.sender;
  }
```

Далі наведено код смарт-контракту мовою Solidity

```
pragma solidity ^0.4.0;
import "/struct.sol";
contract test {
    mapping (string => string) basic_data;
    address owner;
    Struct.Project[] public projects;
    Struct.Education[] public educations;
    Struct.Exam[] public exams;
    function contructor() public {
      owner = msg.sender;
    }
    modifier onlyOwner() {
      if (msg.sender != owner)
      _;
    }
    function setBasicData (string key, string value)
onlyOwner() public {
      basic_data[key] = value;
    }
    function editProject ()
      {
      bool operation;
      string name;
      string link;
      string description;
     onlyOwner(); {
      if (operation) {
        projects.push(Struct.Project(name, description,
link));
      } else {
        delete projects[projects.length - 1];
      }
    }
    function (
      bool operation,
      string name,
      string speciality,
      int32 year_start
    ) onlyOwner; {
      if (operation) {
        educations.push(Struct.Education(name,
speciality, year_start));
      } else {
        delete educations[educations.length - 1];
      }
    }
    function (bool operation, string name, string link,
string language) onlyOwner; {
      if (operation) {
        publications.push(Struct.Exam(name, semister,
teacher));
      } else {
        delete publications[publications.length - 1];
      }
    }
    function (string arg) constant returns (string)
      return basic_data[arg];

    function getSize(string arg) constant returns (uint) {
      if (sha3(arg) == sha3("projects")) { return
projects.length; }
      if (sha3(arg) == sha3("educations")) { return
educations.length; }
      if (sha3(arg) == sha3("exams")) { return
exams.length; }
```

```
    throw;
  }
}
```

Для полегшення опису моделі візьмемо до уваги наступних учасників системи: студент, викладач та представник деканату.

Кожному з учасників мережі потрібно створити власний обліковий запис, який надалі дозволить працювати у мережі. Кожен зі студентів та викладачів проходить процедуру реєстрації, після чого кожен з учасників мережі отримує свій публічний ключ, що дозволяє відправляти запити на проведення «транзакцій» у системі та ідентифікує кожного учасника.

У мережі існують облікові записи типу «Master», що будуть підтверджувати або відхиляти операції в мережі, додаючи їх у ланцюг блоків. Ці облікові записи потрібні для того, щоб реалізувати своєрідний протокол «доказу роботи», що має місце у системах Bitcoin та Ethereum.

Наприклад, студенту однієї з навчальних груп потрібно перескласти іспит з навчальної дисципліни X. Студент відправляє запит на перескладання до облікового запису Master. Якщо представник деканату вважає, що дана дія має право бути здійсненою – потрібно поставити цифровий підпис своїм приватним ключем. Студент не має права доступу до приватних ключів жодних з аккаунтів адміністрації або викладачів, тому це виключає можливість підробки контрактів у системі за допомогою дублювань цифрових підписів однаковими ключами. Наступним кроком студента буде відправка контракту зі вказаними умовами до Викладача. Останнім кроком стане цифровий підпис Викладача у разі успішного перескладання іспиту. Після того як умови контракту будуть виконані (пара цифрових підписів від Викладача та Деканату) – транзакція автоматично буде додана у загальну ланку блоків без можливості внесення змін.

Основною особливістю блокчейну є неможливість внести зміни у весь побудований ланцюг, окрім як провести повний back-up до конкретного блоку, але це буде суперечити всій філософії даного методу, тому на це потрібно звертати увагу всіх учасників.

При реалізації цієї системи потрібно враховувати те, що блокчейн повинен бути неперервним, адже існує ймовірність того, що зловмисник захоче заволодіти контролем над ситуацією і буде намагатися провести обчислення, що дозволять йому генерувати блоки або паралельно з оригінальним ланцюгом, або піти далі, і продовжити генерацію, додаючи свої блоки. Від цієї ситуації може врятувати поліпшена схема шифрування, що підвищить рівень безпеки і дозволить генерувати ланцюг лише в окремі періоди часу, але це потрібно робити мінімум раз на тиждень; або можливість Master-аккаунтів підтверджувати транзакції настільки часто, настільки це взагалі можливо: поточні оцінки або їх відсутність у блокчейні, що будуть додаватися допоможуть підвищити складність системи, що не дозволить зловмиснику провести швидкий перерахунок блоків для продовження генерації ланцюга своїми обчислюваними ресурсами. Питання що до безпеки, здебільшого пов'язане з витоком інформації, або прямим доступом до кожного з типів облікових записів наявних у мережі – Студента, Викладача та Master.

З розглянутих можливостей додавання подій у блокчейн фактультету в цілому є наступні варіанти розвитку подій:

1. Кожна з оцінок студента додається у блокчейн як окрема транзакція;

2. Оцінки студентів однієї групи являтимуть собою згрупований список та будуть додані у окремий момент часу, наприклад, після заповнення відповідної таблиці викладачем.

Варіант №1 не є бажаним, через те, що система буде отримувати занадто велику кількість транзакцій на обробку, створюючи чергу та ускладнюючи роботу обліковим записам типу Master, адже останні виступають у цій ланці валідаторами додання записів, проставляючи цифровий підпис. На відміну від №1, Варіант №2 значно полегшить роботу обліковим записам Master через відносно невелику кількість записів і можливість додавати їх впорядковано.

Після здачі модульної контрольної роботи у групі N, викладач заповнює таблицю та завантажує її у своєму особистому кабінеті на відповідну сторінку. «X» потрапляє у лист очікування операцій до деканату. Після отримання інформації Master-аккаунт отримує повну інформацію на підписання, для формування додаткових заміток (якщо це потрібно), перед додаванням блоку у загальну ланку.

Цікавою для деяких студентів може опинитися можливість за допомогою даної блокчейн-системи отримати дозвіл для перескладання іспиту уникаючи участі паперових документів. Кожен студент через обліковий запис може відправити запит на ім'я Master, щоб отримати дозвіл на перескладання іспиту або заліку за допомогою електронних запитів.

Наприклад, Студент Y навчальної групи N+1, який бажає перескласти іспит з певної дисципліни відправляє відповідний запит на проведення операції через свій обліковий запис. Запити будуть потрапляти у окремний лист очікування, у аккаунтів типу Master буде можливість створити окремий ланцюжок транзакцій боржників. Після того як запит буде додано до відповідної черги, студент матиме час, щоб потрапити до викладача та перескласти іспит. Якщо студент впорався з перескладанням іспиту у відведений час, Викладач підтверджує проведення операції, і Master-аккаунт додає запис у оригінальний блок.

Висновки

Сфера інформаційних технологій на сьогоднішній день захоплена ідеєю блокчейну, криптовалют, мікротранзакцій та смарт-контрактів. Це великий крок вперед, який дозволить підвищити автоматизацію майже усіх сфер життєдіяльності суспільства.

Модель даної системи є достатньо новою для використання в некомерційних цілях. Варто додати, що сама технологія смарт-контрактів має жорсткі правила формування умов, що може перешкодити зручності можливого використання на перших кроках, але при подальшій розробці програмного забезпечення їх

можливо буде уникнути додаючи нові інструменти керування та взаємодії.

В запропонованій системі було використано моделі напівавтоматичної реалізації смарт-контрактів. Основною перевагою використання системи буде можливість швидкого доступу до блоків, що містять інформацію про навчальну успішність студентів, не вимагаючи використання паперових документів та довідок. Також було уникнено використання поняття «gas», яке виступає оплатою за транзакції у системі Ethereum. Це було зроблено не випадково, так як система з самого початку є некомерційною і використання «пального» у даній системі є недоцільним. З метою подальшого впровадження запропонованої системи потрібно провести початкову підготовку спеціалістів для подальшої роботи з даною системою. Однозначними перевагами цієї системи стають:

1. Можливість використання даної моделі смарт-контрактів для розробки подальшої технічної документації та програмної реалізації на базі Інституту інформаційних технологій та механотроніки Полтавського національного технічного університету імені Юрія Кондратюка та у інших навчальних закладах на некомерційних засадах.

2. Наявність швидкого доступу до інформації щодо успішності студентів та формування необхідної документації.

3. Можливості продовження подальшої розробки даної системи некомерційних смарт-контрактів та реалізації на програмному рівні для спрощення роботи з електронним документообігом.

References / Література

1. Nick Szabo. Smart Contracts: Building Blocks for Digital Markets. URL: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html
2. Satoshi Nakamoto. (2008). Bitcoin.org: Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf
3. Bitcoin.org: Bitcoin Development. URL: https://bitcoin.org/en/development.
4. Ethereum Project: Ethereum Homestead Documentation. URL: http://www.ethdocs.org/en/
5. Gavin Wood. Ethereum: A secure decentralized generalished transaction ledger. *EIP-150 REVISION.* URL: https://gavwood.com/paper.pdf
6. Solidity: Docs. URL: http://solidity.readthedocs.io/en/v0.4.24/
7. Solidity: Solidity by Example. URL: https://solidity.readthedocs.io/en/v0.4.24/solidity-by-example.html
8. How does Ethereum work - Medium. URL: https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369
9. GitHub: Bitcoitcoin BIP'S. URL: https://github.com/bitcoin/bips
10. GitHub: Ethereum. URL: https://github.com/ethereum