МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ А.С. Савченко
«____» _____ 2020 р

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СПУПЕНЯ “МАГІСТРА”

ЗА СПЕЦІАЛІЗАЦІЄЮ “ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)”

Тема: “Інформаційна система з обробки транзакцій розподіленого реєстру”

Виконавець: студент-дипломник Боскін Костянтин Олександрович

Керівник: к. т. н., доцент Савченко Аліна Станіславівна

Нормоконтролер: _____ Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ

Завідувач кафедри

А.С. Савченко

“_____”_____2020 р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Боскіна Костянтина Олександровича

(прізвище, ім’я, по батькові)

1. Тема роботи: «Інформаційна система з обробки транзакцій розподіленого реєстру» затверджена наказом ректора № 1891/ст. від 02.10.2020р.

2. Термін виконання роботи: з 05.10.2020 по 31.12.2020 р.

3. Вихідні дані до роботи: розробка прикладного ПЗ для використання розподіленої інформаційної системи

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, розробка прикладного ПЗ для використання розподіленої інформаційної системи, оцінка якості технології, проект інтерфейсу прикладного програмування для розподіленої інформаційної системи з обробки транзакцій, висновки.

5. Перелік обов’язкового ілюстративного матеріалу: таблиці, рисунки, діаграми.

6. Календарний план-графік

№ з/п	***Завдання***	***Термін виконання***	***Підпис керівника***
1	Аналіз літератури та джерел за темою проекту.	05.10.20 – 10.10.20р.	
2	Розробка та затвердження плану дипломного проекту.	11.10.20 – 15.10.20р.	
3	Проведення консультації з науковим керівником щодо створення першого розділ.	16.10.20 – 18.10.20р.	
4	Аналітичний огляд і постановка задачі.	19.10.20 – 29.10.20р.	
5	Порівняльний аналіз існуючих програмних реалізацій систем валідації	30.10.19 – 10.11.20р.	
6	Створення серверної частини системи	11.11.20 – 21.11.20р.	
7	Створення користувацького інтерфейсу	22.11.20 – 02.12.20р.	
8	Висновки та оформлення пояснювальної записки дипломного проекту.	03.12.20 – 09.12.2020	
9	Підписання необхідних документів у встановленому порядку.	10.12.20	
10	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	11.12.20 – 20.12.20	

7. Дата видачі завдання: 05.10.2020 р.

Керівник дипломного проекту _____ Савченко А.С.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ _____

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Інформаційна система з обробки транзакцій розподіленого реєстру» містить 88 сторінок, 24 рисунків, 5 таблиць, 15 наукових джерел.

Об'єкт дослідження: процес проведення транзакцій в системах розподіленого реєстру

Предмет дослідження: система проведення транзакцій у P2P мережах

Мета роботи: Розробити розподілену інформаційну систему проведення транзакцій із зручним користувацьким інтерфейсом.

Методи дослідження, технічні та програмні засоби: порівняльний аналіз, розробка розподіленої системи за допомогою мови програмування Python та алгоритмів шифрування, порівняння ефективності роботи функціональних методів шифрування, розробка веб-інтерфейсу системи в середовищі PyCharm, обробка літературних джерел.

Отримані результати та їх новизна: розроблено розподілену систему управління транзакціями та веб-інтерфейсом для користувачів.

Матеріали дипломного проекту можуть бути використані при розробці розподілених систем управління транзакціями та веб-інтерфейсом для користувачів.

Ключові слова: P2P, ШИФРУВАННЯ, РОЗПОДІЛЕНІ СИСТЕМИ, ОБРОБКА ТРАНЗАКЦІЙ, SHA, РОЗПОДІЛЕНИЙ РЕЄСТР, PYTHON, FLASK

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

SHA – secure hash algorithm, алгоритм криптографічного шифрування

ІС – інформаційна система

РОС – розподілена обчислювальна система

TCP/IP – набір протоколів мережі Інтернет

ПЗ – програмне забезпечення

ОС – операційна система

P2P – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів

Блокчейн – розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшає.

SDK – набір інструментів розробки

URL – унікальний ідентифікатор ресурсу в інтернеті

API – програмний інтерфейс додатку.

ORM – об'єктно-реляційне відображення.

UI – інтерфейс користувача.

ЗМІСТ

ВСТУП

Обробка транзакцій в сучасному світі є невід'ємною складовою життя будь-якого сучасного жителя планети у цивілізованій країні. Фактично, більшу частину часу користувачі виконують транзакції - особливо під час розрахунків. Вживаючи термін «транзакція» - зазвичай вкладають такий зміст, що означає певну взаємодію об'єктів протягом виділеного відрізка часу, який включає певні етапи, такі як :

- запит
- виконання завдання
- відповідь.

У більш загальному та широкому значенні транзакції у банках — це різнородні операції, які є пов’язаними із фінансами і рахунками. У більш вузькому — це певного характеру операції власників банківських карт з використанням електронного рахунку, який користувачі отримують в банку (видача готівки, стипендії, переказ грошей, оплата рахунків)

Обробка транзакцій - це такий вид обробки інформації в комп'ютерній науці, при якому інформація ділиться на певні атомарні операції, які називаються транзакц

іями. Транзакція вважається виконаною повністю і успішно, якщо дотримується цілісність даних повністю незалежно від інших транзакцій, що виконуються паралельно, або не виконана зовсім. Таким чином — транзакція не може бути виконана частково.

З точки зору побудови системи або впровадження нового інноваційного рішення будь-яка компанія хоче, щоб її розробка відповідала критеріям, які диктує сучасний ринок і була конкурентоспроможною, а також, давалі гідну відповідь з точки зору викликів і завдань які ставить до неї базовий користувач.

Розділ 1. Аналітичний огляд розподілених інформаційних систем та постановка задачі

1.1 Області застосування інформаційних систем на базі технології обробки транзакцій розподіленого реєстру

Фактично, більшу частину часу ми виконуємо транзакції - особливо під час розрахунків. Можна також стверджувати, що будь-які процеси є транзакційними. Наприклад, якщо в логістичному центрі відправили машину, то в іншому логістичному центрі - її очікують. Можна сказати, що відбулася транзакція зі зміною стану (машина виїхала - машина їде - машина на місці). Точно так все відбувається і в фінансовому секторі, наприклад, переказ грошей - є транзакцією. Іншими прикладами можуть слугувати розрахунок в метро, купівля продуктів в супермаркеті, тощо.

Але, такий підхід має ряд недоліків, наприклад: централізація, непрозорість валідації транзакцій а також неможливість відслідкувати джерело. Для рішення цих проблем використовуються децентралізовані системи з обробки транзакцій, які власне і є повноцінною інформаційною системою. Відповідно, завдання, які така система має вирішувати - це децентралізація, підвищення стабільності та додаткова прозорість валідації, доступна будь-якому користувачу.

1.2 Вимоги до інформаційних систем

Інформаційна система (ІС) — це організований набір елементів, що збирає, обробляє, передає, зберігає та надає дані. ІС — складається із людей, обладнання, процесів, процедур, даних та операцій [14]. Кожна інформаційна система включає в себе наступні компоненти:

- структура системи;
- функції кожного елемента системи;
- вхід і вихід кожного елементу і системи в цілому;
- мета і обмеження системи та її окремих елементів.

Рис. 1.1. Загальна структура автоматизованих ІС

Інформаційна система не тільки відображає функціонування об’єкта управління, а й впливає на нього через органи управління. Вона є сукупністю інформаційних процесів для задоволення потреби в інформації різних рівнів прийняття рішень. Її метою є продукування інформації для використання (споживання) управлінським апаратом. Відповідно вона забезпеч ує нагромадження, передачу, збереження, оброблення та узагальнення інформації

“знизу вгору”, а також конкретизацію інформації “зверху донизу”. Призначення ІС полягає в описі економічного об’єкта, його станів, взаємодії, що виражається через економічні показники. Вона покликана своєчасно подавати органам управління необхідну і достатню інформацію для прийняття рішень, якість яких забезпечує високоефективну діяльність об’єкта управління та його підрозділів. [14]

До головних завдань належать:

- виявлення джерел інформації;
- збирання, реєстрація, обробка та видача інформації, що характеризує стан виробництва та управління;
- розподіл інформації між керівниками, підрозділами та виконавцями відповідно до їх участі в управлінні.

Сучасний прогрес інформаційних технологій тісно пов’язаний з розвитком програмованих інформаційних систем, які в економіці використовуються для автоматизації вирішення завдань. Загалом, для вирішення майже будь-якої задачі за ЕОМ слід виконати наступні дії :

1. необхідно створити інформаційне забезпечення (забезпечити розрахунки потрібнимиданими)
2. необхідно створити математичне забезпечення (створити мат. модель вирішення задачі, за якою складається програма для ЕОМ).

В свою чергу, інформація, що необхідна може надходити із зовнішніх джерел, або через систему інформаційного забезпечення, яка може доповнюва ти існуючі дані за рахунок нової інформації. Визначальною особливістю інформаційної системи є те, що вона забезпечує користувачів інформацією з кількох джерел даних, таким чином досягається максимальна достовірність.

1.3 Характеристики розподілених інформаційних систем

Загалом, метою створення ІС є – у визначені певні терміни створити систему обробки даних, яка має задані споживчі якості та відповідає поставленим умовам. Така система має ряд характеристик, зокрема, до них належать: [14]

Функціональна повнота – це властивість інформаційної системи, яка характеризує рівень автоматизації управлінських робіт. Коефіцієнт функціональної повноти:

$$Kf = Па * По$$

де Па–показники, отримані автоматизовано; По–загальна кількість показників.

Своєчасність – це властивість інформаційної системи, яка характеризує можливість отримання апаратом керівництва необхідної інформації.Коефіцієнт своєчасності:

$$Кс= Па - ПаПа$$

де Па–кількість показників, отриманих із затримкою щодо планового терміну подання

Функціональна надійність – це властивість інформаційної системи виконувати свої функції з обробки даних. Це сукупність надійностей програмного, інформаційного та технічного забезпечення.

Адаптивна надійність – це властивість інформаційної системи виконувати свої функції, якщо вони змінюються в межах умов, зумовлених розвитком системи керування об’єкта впродовж заданого проміжку часу.

Економічна ефективність інформаційної системи виявляється в покращенні економічних результатів функціонування об’єкта в результаті впровадження інформаційної системи. Створення інформаційної системи передбачає частковий чи повний перегляд методів і засобів функціонування інформаційної системи економічного об’єкта і виконання таких завдань: [14]

1) Виявлення його суттєвих характеристик.

2) Створення математичних і фізичних моделей досліджуваної системи та її елементів.

3) Встановлення умов взаємодії людини та комплексу технічних засобів.

4) Детальна розробка окремих проектних рішень.

5) Аналіз проектних рішень, практична апробація та впровадження

Розподіленою обчислювальною системою (РОС) називають множину незалежних ресурсів, що можуть взаємодіяти між собою з метою спільного рішення обчислювальних завдань. У більш вузькому сенсі, РОС - це множина автономних машин-обчислювачів (вузлів), об'єднаних мережею з установленим спеціалізованим програмним забезпеченням, які складають цільне, погоджене середовище для вирішення обчислювальних задач. У ряді випадків розподілені системи дозволяють забезпечити більш ефективне виконання задач у порівнянні з багатопроцесорними системами, основними перевагами при цьому являються гнучкість організації системи та масштабованість ресурсів.

Слід відзначити, що визначальним компонентом такої системи є відповідне прикладне програмне забезпечення (ПЗ), що виконує роль оркестратора роботи визначеної кількості ресурсів - фізичних обчислювачів (наприклад ЕВМ) та програмних додатків, які функціонують під керуванням операційних систем. Таке ПЗ базується на стандартах (representation and exchange) обміну інформацією. Особливостями ресурсів є можливість незалежного використання, скоординована робота (collaboration) з метою вирішення обчислювальної задачі, доступність до кінцевого користувача. Об’єднання ресурсів, при цьому, повинно здійснюватись у вигляді єдиної, когерентної системи. Слід зазначити, що одним із найголовніших складових вищезазначеного програмного забезпечення являється набір методів, підходів, засобів розподілення та управління вирішенням обчислювальних завдань , множина яких складає обчислювальну задачу. Безумовно ПЗ РОС можливо розглядати на різних рівнях, виділити додаткові шари, класи компонентів залежно

від їх специфіки. Але основу такого ПЗ складає ряд сервісів, які формують інтерфейс взаємодії компонентів різних програмних додатків та вирішують проблеми, пов’язані з відмінностями апаратних та програмних платформ.

Ключовими характеристиками розподілених систем є: прозорість, відкритість, паралельність виконання завдань, масштабованість, стійкість до помилок. Розглянемо ці характеристики. [13]

1) Прозорість (Transparency). Дана характеристика припускає приховування деталей реалізації, структури й функціонування системи. Щодо структурної складової системи - приховуються особливості представлення даних, наявність ряду різнорідних програмних і апаратних ресурсів, особливості їх конкурентного використання. Стосовно процесів − приховуються наявність паралельної обробки (concurrency), міграції (migration) або переміщення (reallocation) ресурсів. Міграція на відмінність від переміщення означає «жи-ве» переміщення ресурсу або обчислювального процесу на інший вузол РОС без зупинки процесу користування (наприклад, live migration в системі Microsoft Hyper-V користувальницьких віртуальних машин з одного вузла кластера на інший для збалансування потужності).

Важливою функцією розподіленої системи є приховування деталей та навіть факту розподілення обчислень між багатьма апаратними та програмними ресурсами, деталей координації їх роботи. Основні види прозорості, які забезпечуються розподіленими системами

Таблиця 1.1. Типи прозорості

Тип прозорості	**Пояснення**
Доступу (Access)	Приховує різноманіття в представленні даних та в питаннях доступу до ресурсів

Розміщення (Location)	Дозволяє забезпечити доступ до локальної та віддаленої інформації в уніфікованому вигляді незалежно від місця та часу взаємодії
Помилок (Failure)	Дозволяє автоматично виявити і виправити помилки, утворюючи при цьому у користувача враження безпомилкової роботи системи
Реплікації (Replication)	Дозволяє невидимо дублювати програмне забезпечення і дані на ряді машин з метою підвищення продуктивності
Міграції (Migration)	Приховує переміщення ресурсу на інше місце

2) Відкритість (Openness). Відкритість системи для розвитку(evolution) та використання означає її незалежність від специфіки ресурсів, доступність для використання програмними додатками користувача. Відкритість досягається шляхом введення специфікації програмної взаємодії. При цьому слід відзначити такі характеристики, як портируємість (portability) і інтероперабельність (interoperability) компонентів системи.

3) Портируємість - характеристика, що оцінює здатність переносу системи з однієї платформи (програмної або апаратної) на другу. Так програмний модуль називається портируємим у випадку, якщо його адаптація до нових умов (платформи) менша ніж ціна переробки даного продукту. Інтероперабельність - оцінює можливість взаємодії системи з іншими системами, незважаючи на можливі розходження, які пов'язані з їхньою реалізацією (мова програмування, середовище виконання).Обидві характеристики пов’язані зі спроможністю повторного використання продукту (reuse) у новому середовищі, або безпосередньо, або за новими умовами, відмінними від тих, за якими він був створений.

4) Паралельність обчислень (Concurrency). Паралельність виконання досягається шляхом паралельного розсилання запитів на виконання завдань на ряд зв'язаних обчислювачів.

5) Масштабованість (Scalability). Якщо при наявності певної кількості машин обчислювальна потужність системи недостатня, то можна її розширити шляхом додавання нових апаратних та програмних ресурсів.

6) Обробка помилок (Fault-tolerance). Робочі машини можуть приймати на себе роль резервних обчислювачів у випадку виходу з ладу інших машин (через збій апаратного чи програмного забезпечення). У такому випадку говорять, що збої і помилки можуть бути виявлені й оброблені іншими машинами. Головною ідеєю являється приховування від користувача та обробка виключних ситуацій: прогнозування виключних ситуацій, відновлення роботи процесів та їх станів, розподілення навантаження на інші вузли. [13]

1.4 Переваги та недоліки використання розподілених інформаційних систем

Оптимізація використання ресурсів, яка досягається використанням незадіяних потужностей (фізичні обчислювальні компоненти), ємностей накопичувачів (оперативна пам’ять або пам'ять вінчестера), графічних можливостей (відеопроцесори), периферійних пристроїв (принтери, плотери);

Підвищення швидкості доступу до даних за рахунок використання розподілених баз даних і широкомовних запитів/відповідей підвищення інформаційної ємності системи за рахунок збільшення кількості вузлів мережі;

Підвищення швидкості обробки даних, що обумовлює збільшення загальних обчислювальних можливостей системи щодо окремих її вузлів підвищення стійкості системи, яка досягається шляхом організації децентралізованої обробки.

Розподілені системи мають довгу історію розвитку (1966 рік), але пік розвитку може бути визначений широким поширенням мережі Інтернет, яке, в

свою чергу, обумовлено появою потужних персональних та серверних комп'ютерів, швидких локальних і глобальних мереж, відповідного програмного забезпечення.

Оскільки в наші дні розподілені системи отримали значне поширення, розробники та користувачі програмного забезпечення повинні знати особливості їх проектування. До недавнього часу всі великі системи бали централізованими, які запускались на одній головній обчислювальній машині з приєднаними до неї терміналами. Термінали практично не займалися обробкою інформації – всі обчислення проводились на головній машині. Розробники таких машин не ставили задач розподілених обчислень.

Всі сучасні системи можемо розділити на три великих класи.

1. Прикладні програмні системи, призначені для роботи тільки на одному персональному комп’ютері або робочій станції. До них відносимо текстові процесори, електронні таблиці, графічні системи та інше.
2. Вбудовані системи, призначені для роботи на одному процесорі або на інтегрованій групі процесорів. До них відносяться системи управління побутовими приладами, різними пристроями та приладами технічного призначення.
3. Розподілені системи, в яких програмне забезпечення слабко інтегрованій групі паралельно працюючих процесорів, об’єднаних через мережу. До них відносимо системи банкоматів, що належать одному банку, видавничі системи, системи ПЗ колективного користування.

В наш час між перерахованими класами програмних систем існують чіткі границі, які в подальшому будуть все більш стиратися. З часом, коли високошвидкісні безпровідні мережі стануть широкодоступний, появиться можливість динамічно інтегрувати пристрої з вбудованими програмними системами, наприклад електронні органайзери з більш загальними системами. [13]

Операційні системи розподілених систем можуть бути **розподіленими(distributed) і мережевими (networked).**

В першому випадку сама система розподілена по всіх вузлах, що для користувача практично приховує наявність мережі. В мережевих операційних системах на кожному вузлі міститься повна копія системи, при цьому сама мережа стає невидимою.

В розподілених обчисленнях можемо виділити три стратегії керування даними:

Централізована (centralized) - керування передбачає розміщення даних на одному вузлі, куди будуть маршрутизуватися всі запити. Особливістю цієї стратегії є те, що результат запиту розміщується в повідомленні і передається по мережі.

Реплікаційна(replicated) - керування копії даних створюється там, де в них найбільше виникає потреба. Це дозволяє уникнути двоетапного підтвердження, але може привести до того, що користувач буде працювати з застарілими даними, якщо не буде прийнято додаткових складних мір. Враховуючи, що доступ для читання запитується частіше, ніж доступ для запису, така стратегія не так і погана.

Розподілена(partitioned) - управління передбачає розподіл даних по вузлам, на основі ймовірності запитів до них. Вона потребує двоетапного підтвердження, але може виявитися ефективною в тому випадку, коли розподіл базується на природних законах належності даних. Об'єктно-орієнтована декомпозиція може виявитися навіть кращою, оскільки вона формується з урахуванням не тільки відношення належності, але і концепції стабільності.

Окремо від типів операційних систем і способі керування даними стоять різні типи розподіленої архітектури комп’ютерів.

Архітектура *клієнт/сервер* (client-server) передбачає наявність одного сервера і одного або декількох клієнтів, при цьому основний об’єм роботи буде виконуватись на комп’ютерах клієнтів. Часто сервер виступає просто сховищем файлів або даних. Це достатньо проста форма розподілення, в якій інтерфейс користувача і бізнес-логіка як правило перемішані.

Трирівнева архітектура клієнт/сервер передбачає виділення інтерфейсу користувача і бізнес-логіки в окремі рівні.

Багаторівнева може зробити це розбиття ще більш дрібнішим. Загальним рішенням Web-додатків є архітектура на основі так званого *тонкого*(thin) і *товстого* (thick) клієнтів, що передбачає винесення бізнес-логіки в клієнтську частину додатку (приклад Java-аплет, що виконується Web-браузером).

На наступному рівні складності знаходиться *мультіклієнтно/мультисерверна* (multiclient/multiserver) архітектура. В ній може існувати декілька серверів, однак вони не взаємодіють напряму. Інакше кажучи, вони не можуть поєднувати в собі роль клієнта і сервера. Таку архітектуру використовують деякі високонавантажені системи

Мережа рівноправних абонентів або з'єднання «точка-точка» (peer- to-peer) – вузли можуть являти собою процесори або образи задач в процесорі. Повідомлення при русі від вузла до вузла може бути розбите на частини, напрямлені різними маршрутами і в подальшому знову зібраними.

Зважаючи на наведені у попередньому пункті характеристики, стає очевидним, що, розподіленим системам властиві певні недоліки.

- *Складність*. Розподілені системи завжди на порядок складніші від централізованих. Стає набагато складнішим зрозуміти і оцінити властивості розподілених систем в цілому, а також тестувати ці системи, відповідно складність росте не прямолінійно. Наприклад, продуктивність системи залежить вже не від швидкості роботи одного процесора, а від шини мережі і швидкодії різних процесорів. Переміщаючи ресурси з однієї частини мережі в іншу, можемо радикально вплинути на продуктивність системи.
- *Безпека.* Як правило доступ до системи можемо отримати з декількох різних машин, повідомлення в мережі можуть переглядатися або перехоплюватися. Тому, в розподіленій системі набагато складніше підтримувати безпеку, відповідно до OWASP стандартів
- *Керованість.* Система може складатися з різнотипних комп'ютерів, на яких можуть бути встановлені різні версії операційних систем. Помилка однієї

машини не розповсюджується на інші машини з непередбачуваними наслідками. Тому необхідно значно більше зусиль, щоб керувати і підтримувати систему в робочому стані.

- *Непередбачуваність.* Як відомо всім користувачам Web-мережі, реакція розподілених систем на певні події непередбачувана і залежить від повного завантаження системи, її організації і мережевого навантаження. Оскільки ці всі параметри можуть постійно змінюватися, час , затрачений на виконання запиту користувача, в той чи іншій момент може суттєво різнитися.

При обговоренні переваг та недоліків розподілених систем визначено перелік критичних проблем проектування даних систем

Таблиця 1.2. Переваги та недоліки розподілених систем

Проблема проектування	Опис
Ідентифікація ресурсів	Ресурси в розподіленій системі розміщуються на різних комп'ютерах, тому систему імен ресурсів необхідно продумати так, щоб користувачі могли без труднощів відкривати необхідні їм ресурси і посилатися на них. Прикладом може бути система уніфікованого показника ресурсів URL, яка визначає адреси Web-сторінок. Без зручної і універсальної системи ідентифікації більша частина ресурсів виявиться недоступною користувачам системи
Комунікації	Універсальна працездатність Internet і ефективна реалізація протоколів TCP/IP в Internet для більшості розподілених систем є прикладом ефективного способу організації взаємодії між комп'ютерами. Однак там, де на продуктивність, надійність та інше накладаються спеціальні вимоги, можна скористатися альтернативними способами

	системних комунікацій
Якість системного сервісу	Якість сервісу, який надає система, відображає її продуктивність, працездатність і надійність. На якість сервісу впливає цілий ряд факторів: розподіл системних процесів, розподіл ресурсів, системні і мережеві апаратні засоби та можливість адаптації системи
Архітектура програмного забезпечення	Архітектура програмного забезпечення описує розподіл системних функцій за компонентами системи, а також розподіл даних компонент за процесорами. Якщо необхідно підтримувати високу якість системного сервісу, вибір правильної архітектури виявляється вирішальним фактором

Завдання розробників розподілених систем – створити таке програмне і апаратне забезпечення слідуючи стандартом, щоб реалізувати при цьому зберегти усі необхідні характеристики розподіленої системи.

В розподіленій системі різні системні компоненти можуть бути запрограмовані різними мовами програмування і виконуватись як окремі елементи навіть на різних типах процесорів. Моделі даних, подання інформації і протоколи взаємодії – все це не обов'язково є однаковим в подібній мережі. Отже для розподілених систем необхідне таке програмне забезпечення, яке могло би керувати цими різнотипними частинами та гарантувати взаємодію і обмін даними між ними. *Проміжне програмне забезпечення* відноситься власне до такого класу ПЗ. Воно знаходиться посередині між різними частинами розподілених компонент системи.

1.5 Визначення інформаційних систем блокчейн

Блокчейн або ланцюжок транзакцій - це технологія обробки, зберігання інформації та ідентифікації клієнтів. Дослівно з англійської блокчейн (blockchain) перекладається, як «ланцюжок блоків», а сама технологія була запропонована в 2008 році Сатоші Накамото.

Головні переваги запропонованої технології:

- прозорість - в блокчейні зберігаються дані про всі проведені операції за всю історію створення системи (криптовалюти);
- стабільність - ви не можете видалити або змінити інформацію «заднім числом», а тільки здійснити нову угоду;
- незалежність - інформація зберігається не на одному центральному сервері, а на безлічі комп'ютерів учасників мережі;

- незмінність даних.

Роботу блокчейну можна порівняти із іншою розподіленою системою Torrent - функціонування системи відбувається в режимі P2P.

Peer-to-peer, P2P — варіант архітектури системи [2], в основі якої стоїть мережа рівноправних вузлів. Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами, може надавати певні сервіси іншим. В правильно спроектованої мережі «peer-to-peer» не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером. Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P-мереж, які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі. В системі P2P автономні вузли взаємодіють з іншими

автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка може контролювати їх. В результаті автономії вузлів вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури. Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, що кожен вузол може надавати й отримувати ресурси, які надаються будь-якими іншими учасниками.

На прикладі P2P мережі торрент, коли користувач завантажує фільм з трекеру, центральний сервер не використовується. Файл безпосередньо завантажується з комп'ютера такого ж учасника мережі. Аналогічно і в blockchain. Усі транзакції проводяться між учасниками мережі безпосередньо. А здійснюються вони за рахунок того, що їхні комп'ютери під'єднані до однієї P2P мережі

Розглянемо технологічні механізми досягнення прозорості, стабільності, незмінності та незалежності.

Прозорість та незмінність. Усі інформація у системі блокчейн записується в транзакції, які згодом передаються до кожного учасника P2P мережі. Кожна попередня транзакція, згідно із протоколом має, вхідні дані про попередню, відповідно, завжди можна відслідкувати відправника. Оскільки зміна навіть одного біта інформації призводить до суттєвих змін в реєстрі - це означає, що дані залишаються незмінними.

Стабільність та незалежність. Оскільки мережа є P2P - відповідно, кожен учасник є валідатором. Навіть, якщо мережа втратить 99% потужностей, за рахунок протоколу, всі дані будуть збережені на залишених машинах і функціонування мережі може бути відновлене [2]

Це також означає, що інформацію про власника неможливо підробити, так як вона зберігається і регулярно оновлюється у багатьох учасників. Деяка частина комп'ютерів учасників системи постійно відключена, наприклад, в Лос-Анджелесі завжди ніч, коли картинна галерея в Києві здійснює угоди. Тому, в блокчейні перед проведенням кожної угоди, картинні галереї звіряють свої дані з мережею, і

визнають ту інформацію, яка є у більшості. Тобто, коли картинна галерея в Лос-Анджелесі відкриється - система блокчейн побачить, що у всіх картинних галереях Європи та Азії записана певна транзакція, і внесе цю інформацію й в свою базу.

На практиці в мережах блокчейн в якийсь момент часу має бути зареєстрована зразу кілька транзакцій, наприклад, які формуються системою в один блок (block). Послідовність кількох таких блоків називається ланцюгом (chain), при цьому такий ланцюг безперервний і нерозривний, так як кожен блок має посилання на попередній. Транзакції також неможливо видалити або змінити - тільки додати нові. Тому, завжди можна подивитися інформацію про перехід прав власності на актив з моменту його створення

Розглянемо принцип роботи технології, на прикладі проведення транзакції в системі.

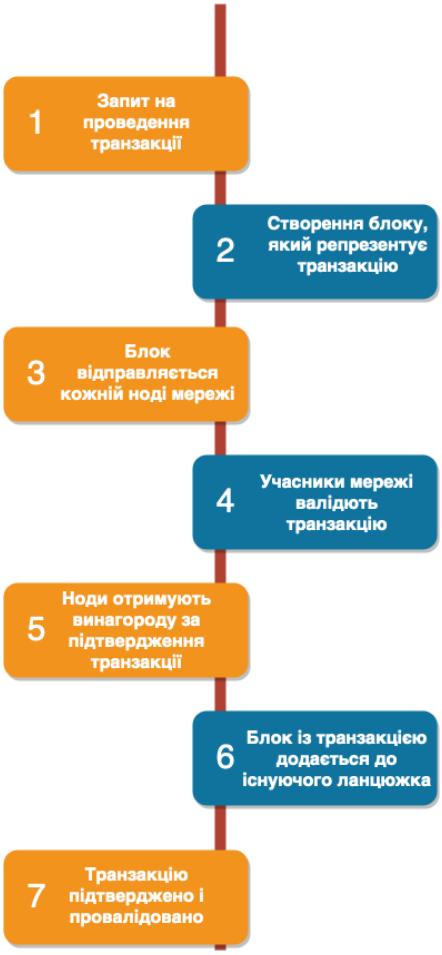


Рис. 1.2. Послідовність роботи технології “блокчейн”

Клієнт 1 хоче перевести Клієнту 2 одну одиницю даних

- Блокчейн формує цю операцію в блок разом з іншими аналогічними транзакціями. Новий блок містить номер і хеш попереднього блоку
- Сформований блок розсилається всім учасникам системи блокчейн
- Якщо немає помилок, кожен учасник записує блок інформації в свою базу даних
- Блок додається до ланцюжку попередніх блоків, таким чином, містить інформацію про всі попередні угоди
- Криптовалюта переходить від Клієнта 1 до Клієнта 2

Роль валідаторів в блокчейні

Системи блокчейн децентралізовані. Це означає, що транзакції обробляються великою кількістю спеціальних учасників системи - майнерами. Ними, як правило, може стати будь-який бажаючий, який має відповідне обладнання та програмне забезпечення.

Функції в системах блокчейн:

- зберігають копії даних, що захищає систему від втрати даних і підробки інформації
- підтверджують проведення транзакцій в системі
- перевіряють операції, що проводяться іншими майнерами.

Що ж мотивує майнерів допомагати системам блокчейн реєструвати і проводити операції? Зазвичай мотивацією виступає винагорода від системи за проведення транзакції, а також комісії, які платять майнерам учасники угоди за обробку транзакції.

Валідатору також потрібно ще проводити транзакцію швидше за всіх, адже винагороду може отримати тільки перший, хто запропонує код-шифрування

операції, тобто створить новий блок в ланцюжку. Цими щасливчиками стають ті майнери, у яких саме потужне обладнання для обробки операцій, тоді як інші учасники мережі «залишаються не при справах».

Блокчейн гаманець

Щоб здійснювати транзакції в блокчейн мережі - потрібно мати спеціальні гаманці (wallet) для зберігання даних. Інформація про гаманці і транзакції з ними захищена шифруванням. Учасники транзакції підтверджують операцію за допомогою криптографічних ключів - спеціальних унікальних цифрових кодів. Але все ж бували випадки, коли онлайн блокчейн гаманці зламували хакери, тому краще зберігати дані офлайн. Транзакція в блокчейні проводиться практично миттєво, але обробка і підтвердження операції може зайняти 10-15 хвилин.

Як правило, блокчейн гаманці припускають анонімність, тобто не можна встановити, хто здійснює операцію. Це допоможе зберегти ваші активи від сторонніх поглядів, але несе і додаткові ризики - адже якщо ви забудете реквізити доступу до гаманця - ви втратите активи.

Блок транзакцій — спеціальна структура для запису групи транзакцій. Транзакція при цьому здійснюється лише тоді, коли вважається підтвердженою. Це зручно і надійно, якщо йдеться про здійснення платежів чи передачу конфіденційних даних. Аби транзакція вважалася достовірною («підтвердженою»), її формат і підписи мають бути перевірені. Після цього групу транзакцій записують в спеціальну структуру (так званий блок). В цих блоках інформацію можна швидко перевірити. А ще в кожному наступному зберігається інформація про попередній. При операціях із криптовалютами, наприклад, у ланцюжку блоків міститься інформація про всі вчинені коли-небудь операції

В блок входять заголовок та список транзакцій. Заголовок блоку включає в себе свій хеш, геш попереднього блоку, геши транзакцій та додаткову службову інформацію. Першою транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок. Для проведення транзакцій в блоці використовують деревоподібне гешування, аналогічне формуванню геш-

суми файлу в протоколі BitTorrent. Транзакції, крім нарахування комісії за створення блоку, містять всередині атрибута input посилання на транзакцію, за якою на цей рахунок були отримані біткойни (або інші дані чи цифрові валюти). Комісійні операції можуть містити в атрибуті будь-яку інформацію (для них це поле носить назву Coinbase parameter), оскільки у них немає батьківських транзакцій. Створений блок буде прийнятий іншими користувачами, якщо числове значення гешу заголовка дорівнює або нижче певного числа, величина якого періодично коригується. [15]

Оскільки результат хешування (функції SHA-256) непередбачуваний, немає алгоритму отримання бажаного результату, окрім випадкового перебору. Якщо геш не задовольняє умову, то довільно змінюється блок службової інформації в заголовку — і геш перераховується. Після співпадіння варіантів вузол розсилає отриманий блок іншим підключеним вузлам, які перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок і наступний блок повинен включити в себе його геш. А тоді все починається спочатку.

Величина цільового числа, з яким порівнюється геш, коригується через кожні 2016 блоків. Заплановано, що вся мережа буде витрачати на генерацію одного блоку приблизно 10 хвилин, на 2016 блоків — близько двох тижнів. Якщо 2016 блоків сформовані швидше, то мета трохи зменшується і досягти її стає важче, в іншому випадку мета збільшується. Зміна складності обчислень не впливає на надійність потрібна лише для того, щоб система генерувала блоки майже з постійною швидкістю незалежно від потужності.

Найчастіше умисна зміна інформації в будь-якій з копій бази або навіть в досить великій кількості копій не буде визнана істинною, оскільки не відповідатиме правилам. Деякі зміни можуть бути прийняті, якщо будуть внесені в усі копії бази (наприклад, видалення кількох останніх блоків через помилку в їхньому формуванні). Як бачите, такий підхід робить blockchain дуже захищеним способом генерування та передачі даних чи коштів через будь-які канали інтернет-з’єднання.

1.6 Сфера використання блокчейну

Один із актуальних для України прикладів — реєстр права власності. Якщо його побудувати із залученням публічного blockchain, отримаємо найстійкіший цифровий реєстр планети при низькій вартості реалізації. Більше того — для забезпечення цілісності реєстру це є безальтернативним технічним рішенням. Це рішення доступне будь-кому і практично нічого не коштуватиме

Основна перевага крім вартості — непідконтрольність баз даних публічних блокчейнів жодній окремій організації, нації чи державі. Разом із цим активне впровадження blockchain знімає потребу в технічних посередниках та інституціях для регулювання. Перспектива значна, і недаремно Стенфорд, Гарвард, MIT почали вивчення можливостей blockchain із 2016 року.

Багато компаній вивчають можливості застосування технологій блокчейн всюди - від фінансів до виробництва.

- Корпорація Microsoft розвиває програми Blockchain-as-a-Service (BaaS) на своїй хмарній платформі Azure.
- IBM запустила власну BaaS пропозицію; передбачається його інтеграція з іншими продуктами компанії, такими як обчислювальна мережа IBM z Systems, система штучного інтелекту Watson для Інтернету речей та ін.
- Blockchain Foundry приділяє головну увагу заснованим на блокчейн сервісам для створення прототипів і випуску промислової продукції.
- Bigchain DB пропонує масштабовані сервіси блокчейн.
- Chain рекламує платформу блокчейн для фінансових сервісів.
- IBM і Samsung працюють над концепцією ADEPT, в якій технологія блокчейн буде використовуватися для формування основи децентралізованої мережі пристроїв - Інтернету речей. Блокчейн планують використовувати для реєстрації мільярдів пристроїв, які будуть автономно транслювати транзакції в системі з трирівневою архітектурою.

- Значну роль в просуванні блокчейна зіграла Європейська комісія, яка запустила в партнерстві з блокчейн-стартапом ConsenSys компанію EU Blockchain Observatory.

1.7 Аналіз існуючих інформаційних систем з обробки транзакцій

Концепція першого блокчейну була розроблена в 2008 році людиною (або групою людей), відомою як Сатоши Накамото. У 2008 році було описано протокол електронних платежів для однорангової мережі (peer-to-peer network, P2P). Це математичний алгоритм, який дозволяє безпечно і приватно обмінюватися цінностями через однорангові мережі. Протокол, створений Накамото, називають «протоколом довіри».Так була закладена основа для технології блокчейн. У 2009 році ця технологія була реалізована в рамках цифрової валюти —біткоїна. Таким чином, першою успішною практичною реалізацією блокчейн технології стала мережа біткойн. У 2015 році журнал The Economist опублікував статтю «Машина довіри», в якій йдеться про те, що технологія мережі біткойн, може повністю змінити економіку. Саме ця технологія стала першою, яка змогла вирішити інформаційну проблему, таку як забезпечення довіри між сторонами до отриманої інформації без залучення зовнішніх гарантів –банків, посередників тощо. Автори багатьох публікацій з блокчейну Дон Тапскотт та Алекс Тапскотт проаналізували програми, сервіси, бізнес-моделі, ринки, організації і навіть уряди, які оперують блокчейн

Консенсусний механізм дозволяє вузлам в одноранговій мережі працювати разом, не знаючи і не довіряючи один одному. Метою консенсусного алгоритму є виконання безпечного оновлення стану відповідно до деяких конкретних правил, при цьому право на виконання зміни стану розподіляється серед користувачів, яким надається право колективно виконувати ці зміни через алгоритм. Механізм консенсусу – це просто набір правил, який узгоджується вузлами в мережі, запускаючи програмне забезпечення мережі. Ці правила дозволяють мережі працювати за призначенням і залишатися синхронізованою.

Консенсусний протокол встановлює правила:

• яким чином блоки повинні бути додані в блокчейн;

• коли блоки вважаються дійсними;

• як вирішуються конфлікти.

Найбільш відомими механізмом консенсусу є доказ роботи (PoW), який використовується в мережі біткойн, доказ стану (PoS), який використовується в Coin, Proof of Elapsed Time –доказ минулого часу (PoeT), який використовується в проектах Hyperledger. Головним недоліком алгоритмів є те, що вони вимагають великих обчислювальних потужностей. Триває широка дискусія про те, які консенсусні механізми є кращими, також створюються нові алгоритми

Рис. 1.3. Логотип P2P мережі “Біткоін”

Ethereum (Ефіріум) – конструктор для створення рішень на блокчейн[7]. Дозволяє побудувати будь-який додаток з верифікацією на блокчейн. Основною ідеєю «Ефіріума» є використання розумних контрактів –записів, які містять умови виконання певних дій. Умовою може стати будь-яка дія, наприклад, передача товарів замовнику. Розробник, який використовує блокчейн Ethereum, може запрограмувати необхідні тригери і дії за допомогою вбудованої мови сценаріїв. При цьому кожен запис може бути перевірений всіма зацікавленими сторонами: реєстр даних залишається відкритим і децентралізованим. Завдяки високій гнучкості розумних контрактів саме Ethereum став однією з найбільш популярних платформ для створення нових блокчейн-проектів з використанням мови Solidity. Розробникам більше не потрібно придумувати власну реалізацію ланцюжка блоків: достатньо створити потрібну надбудову над вже існуючою системою.

Рис. 1.4. Логотип P2P мережі “Ефіріум”

На сьогоднішній день однією із найбільш популярних відкритих платформ для створення блокчейн проектів є **Hyperledger** від Linux Foundation, Intel, IBM та інших (більше 100 крупних фірм). Платформа пропонує набір інструментів для розробки:Fabric, Iroha, Sawtooth lake, blockchain explorer, Fabric chaintool (Caliper, Cello, Composer,Quilt), Fabric SDK Py, Corda. На сьогоднішній день однією із найбільш популярних інструментів є Hyperledger Fabric – це блокчейн-фреймворк, який призначений для створення основи для розробки рішень на блокчейні і заснований на модульній архітектурі. Fabric представляє собою модуль для розробки масштабованих блокчейн додатків з гнучким рівнем дозволів до якого в разі необхідності можуть бути приєднані різні компоненти, наприклад, алгоритми консенсусу. Основна вимога Hyperledger –це модульна структура. Різні служби повинні підключатися і відтворюватися, користувачі повинні мати можливість легко видалити і додати модуль відповідно до специфіки свого бізнесу.

Рис. 1.5. Логотип P2P мережі “Гіперледжер”

Проект Aragon реалізує концепцію децентралізованих організацій, існуючих виключно в межах блокчейна: жодних паперів і бюрократичних процедур, тільки цифрові дані. На сайті Aragon вже доступна альфа-версія програмного забезпечення, яка успішно справляється із завданнями, що виникають при створенні стартапів та інших приватних онлайн-проектів.

Рис 1.6. Логотип P2P мережі “Арагон”

Sia – проект децентралізованого хмарного сховища. На відміну від традиційних сервісів типу Google Drive або Amazon S3, що зберігають призначені для користувача дані на власних серверах, Sia пропонує механізм розподілу зашифрованої інформації на багатьох незалежних комп’ютерах. Перевага Sia перед традиційними хмарними сервісами полягає у вартості передплати: витрати на зберігання файлів у децентралізованому сховищі в 10-15 разів нижче, ніж у традиційних файлових хостингах. Крім того, зашифровані файли не можуть бути розкриті на вимогу поліції та інших державних структур. Ще один проект, практично ідентичний Sia –розподілене сховище Sto

Рис. 1.7. Логотип P2P мережі “Cia”

EOS - це платформа, для розробки децентралізованих додатків, головний конкурент Ethereum. Вона має простий і зручний інтерфейс, і дуже високу (до 100 000) швидкість обробки транзакцій. Внутрішня монета блокчейн-мережі EOS Token ERC-20, створений на базі ефіріума. Основні переваги платформи EOS це:
- Висока масштабованість система підтримує такі функції, як асинхронність зв'язку і паралельні виконання. У ній працюють тисячі комерційних додатків і транзакції виконуються майже миттєво.
- Зручність розробникам доступний великий набір інструментів і платформа веб-збірки. Тут можна зупинити і відрегулювати вийшло з ладу додаток, скористатися

готовими шаблонами для створення інтерфейсів і клієнтських баз даних. - Безкоштовний доступ до основних функцій. Якщо ви побажаєте запустити блокчейн-екосистему або децентралізоване додаток на платформі EOS, компоненти для створення доступні безкоштовно.

Рис. 1.8. Логотип P2P мережі “EOC”

Libra (у перекладі з англійської — Терези, попередньо відома як GlobalCoin, чи Facebook Coin) — криптовалюта, що у 2020 році планує випускати американська соціальна мережа Facebook. Її офіційно анонсували 18 червня 2019 року разом із випуском білої книги.

Планується, що на початковому етапі Libra буде використовуватися для грошових переказів, що здійснюються у різних валютах. Таким чином, вона складе пряму конкуренцію таким послугам, як Western Union, за користування якими знімається значна комісія. Оплата ж за перекази у новій криптовалюті повинна бути значно меншою. Крім того, якщо обліковий запис користувача Libra буде з'єднаний з його банківським рахунком, можна буде обміняти електронні гроші на іншу валюту безпосередньо на смартфоні.

Попередньо, Facebook заснував Libra Association, уклавши партнерську угоду з 28 великими компаніями, так чи інакше пов'язаними з електронними платежами, зокрема з MasterCard та Uber. Щоб уникнути волатильності, Libra забезпечуватиметься фінансовими активами, такими як казначейські цінні папери.

Рис. 1.9. Логотип P2P мережі “Лібра”

1.8 Постановка задачі

Усі із запропонованих інформаційних системи реалізують вирішення проблем, описаних у пункті 1.1, таких як:

- централізованість мережі
- швидкість проведення транзакцій
- можливості масштабування мережі

Усі ці проблеми приховано на глибокому або дуже глибокому рівні, хоча для звичайного користувача - це не несе високої цінності. Причина полягає в тому, що користувачі, фактично не можуть використовувати без технічних знань або спеціального апаратно-програмного комплексу подані ланцюги и виконувати транзакції в них. Відповідно, запропонована система має усунути зазначений недолік і відповідати наступним критеріям :

- Високий ступінь зрозумілості для будь-якого користувача
- Легка доступність з будь-якої точки
- Простий користувацький інтерфейс із мінімумом функцій, який допоможе ефективно використовувати всі можливості ІС з обробки транзакцій в розподілених реєстрах

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі було розглянуто області області застосування інформаційних систем на базі технології обробки транзакцій розподіленого реєстру. Було розглянуто загальні вимоги до інформаційних систем.

Визначено поняття розподілених інформаційних систем, їх головні характеристики (Функціональна повнота, Своєчасність, Функціональна надійність, Адаптивна надійність), а також переваги та недоліки їх використання. До переваг можна віднести децентралізованість системи а також швидкість обробки даних за рахунок багатьох учасників мережі

Надано визначення інформаційних систем на базі технології блокчейн - це технологія обробки, зберігання інформації та ідентифікації клієнтів, яка представляє собою «ланцюжок блоків».

Визначено сферу використання та проаналізовано існуючі рішення, представлені у вигляді існуючих проектів

Сформульовані задачі:

1. Високий ступінь зрозумілості для будь-якого користувача
2. Легка доступність з будь-якої точки світу
3. Простий користувацький інтерфейс із мінімумом функцій, який допоможе ефективно використовувати всі можливості ІС з обробки транзакцій в розподілених реєстрах

Розділ 2 Функції хешування для інформаційних систем

2.1 Поняття хеш та алгоритм хешування. Застосування хеш функції

Розглянемо алгоритми хешування і їх застосування в системах типу блокчейн. Загальну схему роботи подібних систем представлено на рис. 2.1.

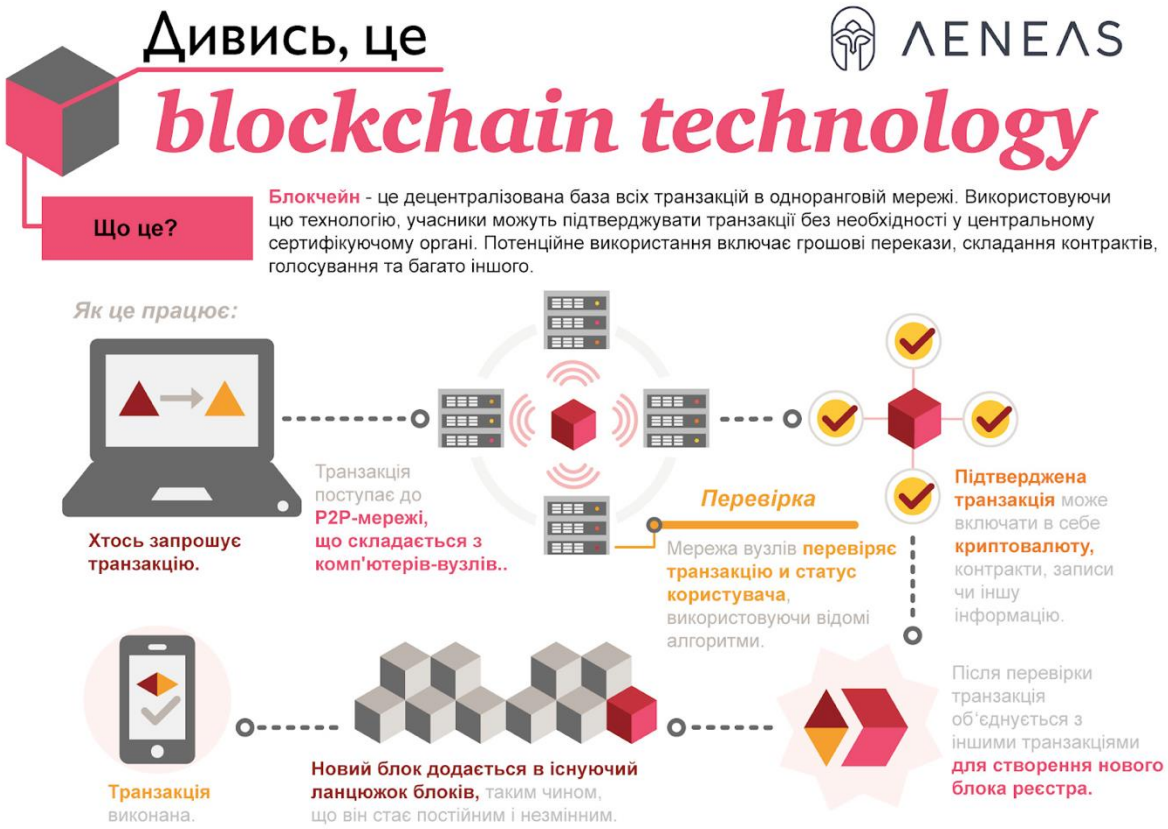


Рис. 2.1. Робота технології “блокчейн”

Алгоритми гешування в системах типу блокчейн [6] використовуються під час створення нового блоку і вал

ідації транзакції, оскільки, згідно із протоколом та ідеєю, кожна нова ланка має містити інформацію про попередню. Функції гешування чудово підходять для

вирішення цього завдання, оскільки вони досить ефективні і з точки зору інформації не займають багато фізичного місця

В ході аналізу, проведеного в другому розділі буде визначено найбільш відповідні технології для побудови власної системи, яка б вирішувала проблеми існуючих рішень, описаних в розділі 1 (реалізація більш ефективних алгоритмів на глибокому рівні для звичайного користувача - це не несе високої цінності, оскільки немає інтерфейсу для користування такими мережами)

Слово хеш походить від англійського «hash», яке можна перекласти як "плутанина", "мішанина", "фарш". Часто ще про сам процес говорять «хешування», від англійського "hashing" (рубати, подрібнювати).

З'явився цей термін в середині минулого століття серед людей, які займаються обробкою масивів даних. Спеціальна хеш-функція дозволяла привести будь-який масив даних до числа заданої довжини.

Наприклад, якщо байти з послідовності байт будь-якої довжини скласти, і від отриманої суми взяти залишок від ділення на 256, то цей залишок від ділення можна буде називати хешем. Для різних початкових байт залишок від ділення буде відрізнятись.

Але для різної послідовності байт по вищенаведеному алгоритму ми можемо отримати однаковий результат. Наприклад, якщо ми в результаті роботи хеш-функції отримали 1, то початкова послідовність довжиною 2 байти може бути однією з наступних:

0001, 0100, 02FF, 03FE, 04FD, ... FE03, FF02

Тобто для послідовності з 2 байт результат роботи хеш-функції буде однаковим у 256 випадках. І нескладним і відносно невитратним перебором можна отримати всі послідовності з 2 байт від яких хеш-функція видає однаковий результат.

Ситуації, коли хеш-функція видає однаковий результат для різних вхідних даних, називають колізіями.

Хеш-функція — функція, що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру.

Наведений вище приклад хеш-функції — це суто вигаданий автором приклад щоб продемонструвати принцип. Хеш-функції бувають різні, головне щоб результат їх виконання відповідають певним умовам:

1. функція повинна вміти проводити будь-який об'єм цифрових даних до числа заданої довжини (по суті — стискання даних до бітової послідовності заданої довжини хитрим способом).
2. найменша зміна (хоча б на один біт) вхідних даних має призводити до повної зміни хеша.
3. функція повинна бути стійкою у зворотній операції — ймовірність відновлення початкових даних по хешу повинна бути досить низькою.
4. функція повинна мати якомога нижчу ймовірність виникнення колізій.
5. хороша хеш-функція не повинна сильно навантажувати обчислювальні потужності. Тут часто важливим є компроміс між швидкістю роботи і якістю результату. Але існують випадки, де від хеш-функції якраз і "вимагається" складність і ресурсоємність.
6. алгоритм роботи функції має бути відкритим, щоб кожен бажаючий міг би оцінити її стійкість, тобто ймовірність відновлення початкових даних по результату її роботи.

2.1.1 Застосування хешування для інформаційних систем

Передача даних по комп'ютерним мережам.

Прості хеш-функції (ненадійні, але які просто і головне швидко обчислюються) застосовують для перевірки цілісності передачі пакетів по мережевому протоколу TCP/IP та інших для виявлення апаратних помилок — так зване "надлишкове кодування". Якщо хеш отриманого пакета даних співпадає з відправленим разом з пакетом даних (так званою "контрольною сумою), то це може означати, що втрати чи помилок при передачі пакета даних по мережі не сталось. Якщо ж хеші не співпадають, то можливо при передачі пакета відбулась втрата даних, у такому разі пакет пересилається знову. У даному випадку

використовується проста хеш-функція, тому що при передачі даних важлива швидкість.

Перевірка цілісності даних

Певно при завантаженні файлів з Інтернет ви стикались з тим, що там наводять деякі числа, які називають або хешем, або контрольною сумою, наприклад такі:

CRC32: 7438E546

MD5: DE3BAC46D80E77ADCE8E379F682332EB

SHA-1: 332B317FB97126B0F79F7AF5786EBC51E5CC82CF

Ці набори символів — результат роботи різних хеш-функцій (їх назви наведено перед числами), які застосували для усього вмісту файлу. За допомогою спеціалізованого програмного забезпечення обчислюється хеш завантаженого файлу і порівнюється його з тим, що було наведено на сайті. Якщо хеші збігаються — файл завантажено без помилок.

Криптографія

Великі по складності хеш-функції використовують у криптографії. Головна умова для них — неможливість по кінцевому результату (хешу) обчислити початковий масив даних. Ще одна головна умова — стійкість до колізій, тобто низька ймовірність отримати два однакових хеша з двох різних масивів даних при обробці їх такою функцією. Розрахунки по таким алгоритмам складні і ресурсоємні, але тут вже головне не швидкість, а надійність.

Для доступу до сайтів та сервісів по логіну і паролю часто використовують хешування. Зберігати паролі у відкритому вигляді для подальшої звірки тими, що ввів користувач, досить ненадійно з точки зору можливості їх викрадення. Тому зберігають хеші усіх паролів. Користувач вводить пароль, миттєво розраховується його хеш і звіряється з тим, що є у базі даних. Надійно і дуже просто. Як правило для такого типу хешування використовують складні функції з дуже високою криптостійкістю, щоб по хешу було неможливим відновити пароль.

Хешування використовують в технології електронного цифрового підпису. За допомогою хеша переконуються, що, наприклад, підписують саме той документ,

що потрібно. Також хеш гарантує, що документ підписано саме тою людиною, яка заявлена, тобто за допомогою можна засвідчувати особу.

Блокчейн

Хеш-функції використовуються в технології блокчейн, де хеш є гарантією цілісності ланцюжка транзакції (платежів) і захищає її від несанкціонованих змін

CRC32 — використовується для створення контрольних сум. Ця функція не є криптографічною. Існує багато варіацій цього алгоритма, число після "CRC" означає довжину отриманого хеша в бітах. Функція досить проста і не ресурсоємна. Використовується для перевірки цілісності пакетів в різних протоколах передачі даних.

MD5 — стара, але до цього часу дуже популярна версія криптографічного алгоритма, яка створює хеш довжиною 128 біт. Хоча стійкість цієї функції на сьогодні і не дуже висока, вона все одно часто використовується для шифрування паролів.

SHA-1 — криптографічна функція. Дає хеш довжиною 160 байт. Зараз відбувається активна міграція в бік SHA-2 — більш стійкої хеш-функції.

Хеш і алгоритми хешування – це ключові поняття, в сфері блокчейну і які завжди йдуть об руку з безпекою. Для підтримки децентралізованих мереж і консенсусних механізмів з тисячею вузлів, об'єднаних за допомогою p2p, необхідна “відсутність довіри” і ефективна система підтвердження. Цим мережам потрібні компактні способи шифрування інформації, які дозволяли б учасникам проводити швидку і безпечну перевірку.

Серед безлічі існуючих хеш-функцій прийнято виділяти криптографічно стійкі, які застосовуються в криптографії. Для того, щоб хеш-функція H вважалася криптографічно стійкою, вона повинна задовольняти трьом основним вимогам, на яких засновано більшість застосувань хеш-функцій в криптографії:

- Незворотність: для заданого значення хеш-функції m повинно бути обчислювально нездійсненно знайти блок даних X, для якого H(X) = m.

- Стійкість до колізій першого роду: для заданого повідомлення M повинно бути обчислювально нездійсненно підібрати інше повідомлення N, для якого H(N) = H(m)
- Стійкість до колізій другого роду: має бути обчислювально нездійсненно підібрати пару повідомлень, що мають однаковий хеш.

Для криптографічних хеш-функцій також важливо, щоб при найменшій зміні аргументу значення функції сильно змінювалося (лавинний ефект). Зокрема, значення хешу не повинно давати витоку інформації навіть для окремих бітів аргументу. Ця вимога є запорукою криптостійкості алгоритмів хешування, які хешують пароль користувача для отримання ключа.

2.1.2 Переваги застосування хеш-функцій в алгоритмах

Хеш-функції також використовуються в деяких структурах даних - хеш-таблицях, фільтрах Блума і декартових деревах. Вимоги до хеш-функції в цьому разі інші:

- хороша перемішуваність даних
- швидкий алгоритм обчислення
- Перевірка даних

У загальному випадку це застосування можна описати, як перевірка деякою інформацією на ідентичність оригіналу, без використання оригіналу. Для звірки використовується хеш-значення перевіреній інформації. Розрізняють два основні напрямки цього застосування:

Перевірка на наявність помилок.

Контрольна сума може бути передана каналом зв'язку разом з основним текстом. На приймальному кінці контрольна сума може бути розрахована наново і її можна порівняти з переданим значенням. Якщо буде виявлена розбіжність, то це означає, що при передачі виникли спотворення і можна запросити повтор.

Побутовим аналогом хешування в даному випадку може служити прийом, коли при переїздах у пам'яті тримають кількість місць багажу. Тоді для перевірки не потрібно згадувати про кожну валізу, а досить їх порахувати. Збіг буде означати, що жодної валізи не втрачено. Подібний метод використовується у піринговых мережах для перевірки скачаного файлу на помилки.

Даний метод легко доповнити до захисту від фальсифікації переданої інформації (метод MAC). У цьому випадку хешування проводиться криптостійкою функцією над повідомленням, об'єднаним з секретним ключем, відомим тільки відправнику і одержувачу повідомлення. Таким чином, криптоаналітик не зможе відновити код, за перехоплених повідомлень і значенням хеш-функції, тобто, не зможе підробити повідомлення.

Прискорення пошуку даних.

Під час запису текстових полів у базі даних може розраховуватися їх хеш-код і дані можуть міститися в розділі, що відповідає цій хеш-функції. Тоді при пошуку даних треба буде спочатку обчислити хеш-код тексту і відразу стане відомо, в якому розділі його треба шукати, тобто, шукати треба буде не серед усієї бази, а лише в одному її розділі (це сильно прискорює пошук).

Побутовим аналогом хешування в даному випадку може служити розміщення слів у словнику за алфавітом. Перша літера слова є його хеш-кодом і при пошуку ми переглядаємо не весь словник, а тільки потрібну літеру.

2.2 Огляд сучасних алгоритмів хешування:

JH. Алгоритм складається загалом із чотирьох криптографічних функцій [13], скорочено : JH-224, JH-256, JH-384 і JH-512. Побудований за схемою Меркле-Дамгарда, таким чино що, вхідне слово розбивається на блоки по 512 біт. Але також можливі і інші розміри хеш-значень - 224, 256, 384 і 512 біт. Незалежно від розміру вихідного значення алгоритму – буде використовуватися одий і той самий

алгоритм сстиснення, що формує після обробки останнього блоку повідомлення 1024-бітове значення, яке просто складуюється до потрібного розміру.

HAVAL — Для будь-якого вхідного значення алгоритм генерує хеш-значення, яке називається «дайджестом повідомлення», і може мати довжину 128, 160, 192, 224 або 256 біт. Кількість ітерацій — змінна, від 3 до 5. Кількість раундів на кожній ітерації — 32. Є модифікацією MD5. [13]

Keccak – вид алгоритму хешування зі змінною розрядністю [13]. Вхідне повідомлення розбивається на блоки по 1024 біт кожний, при цьому, використовується спосіб так званої «криптографічної губки»

MD2 – алгоритм, що виконує функцію згортки для дани змінного розміру але при цьому, гарантовано - з отриманням 128-бітного вихідного слова. Перш за все, виконується операція «доповнення» вхідних даних до розміру, так, щоб вхідне слово було кратним 16 байтам. Потім обчислюється контрольна сума повідомлення, яка також доповнює повідомлення перед його подальшою обробкою. [13]

MD4 – Вхідне повідомлення, яке потрібно зашифрувати, розбивається на шматки розміром по 512 бітів. Останній блок завжди доповнюється до 512-бітового розміру, якщо не вистачає даних

MD5 — 128-бітовий алгоритм хешування. З точки зору безпеки – отримав надлаштування над його попередником – MD4

MD6 — алгоритм хешування змінної розрядності. Пропонується на зміну менш досконалому MD5.

N-Hash - криптографічна хеш-функція на основі циклічної функції FEAL. В даний час вважається небезпечною. [13]

RIPEMD-128 — хеш-функція з розміром хешу 128-біт. Хешування RIPEMD-128 потребує достатньо багато часу для виповнення и не гарантує високий рівень безпеки. [13]

RIPEMD-160 - покращена версія RIPEMD, яка, в свою чергу, використовувала принципи MD4 і по продуктивності її можна порівняти з більш популярною SHA-1.[13]

RIPEMD-256 – покращена версія RIPEMD-128 і призначена для додатків, які вимагають довшого хеша і не потребують в більшій безпеці, ніж додатки рівня RIPEMD-128. [13]

RIPEMD-320 – покращена версія RIPEMD-160, и призначена для додатків, які потребують більш довгого хешу и не потребують більшої безпеки, ніж додатки рівня RIPEMD-160. [13]

Secure Hash Algorithm 1 (SHA-1) – дуже розповсюджений «класичний» алгоритм шифрування, який дуже багато де використовується. Також рекомендований в якості основного для державних установ у США. Принципи є збіжними із принципами хешування MD4.

SHA-2* - збірна назва односпрямованих хеш-функцій SHA-224, SHA-256, SHA-384 і SHA-512. Хеш-функції призначені для створення «відбитків» або «дайджестів» повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов'язаних із захистом інформації.

Skein - алгоритм хешування змінної розрядності. Хеш-функція Skein виконана як універсальний криптографічний примітив, на основі блочного шифру Threefish, працюючого в режимі UBI-хешування. [13]

Snefru - криптографічна односпрямована хеш-функція. Функція Snefru перетворює повідомлення довільної довжини в хеш-довжину.

Tiger — хеш-функція, призначена для особливо швидкого виконання на 64-розрядних комп'ютерах. [13]

Алгоритми сімейства MD (Message Digest)

До алгоритмів сімейства MD (Message Digest) входять алгоритми MD2, MD4, MD5 та MD6. Перші три з них широко використовувалися і продовжують широко використовуватися, незважаючи на те, що існують відомі криптоаналітичні атаки проти кожного з них.

Алгоритми MD мають різні характеристики; вони розроблені в різні роки відомим криптологом Рональдом Ривестом (Ronald Rivest) за участю деяких інших фахівців RSA Laboratories - наукового підрозділу компанії RSA Data Security Inc.

2.1.1 Алгоритм MD5

Алгоритм MD5 є найбільш широко відомим і використовуваним алгоритмом хешування серед алгоритмів MD.

Алгоритм MD5 був запропонований в 1992 р., тобто всього через два роки після опублікування алгоритму MD4. Передумови розробки MD5 були такі:

- при розробці алгоритму MD4 основним критерієм була швидкодія даного алгоритму, його криптостійкість була дещо менш важлива; в результаті, вже в 1991 р. була опублікована перша атака на усічений алгоритм MD4);
- оскільки алгоритм MD4 вийшов виключно швидким, особливо на 32-бітних платформах, передбачалося, що саме тому він буде в центрі уваги криптоаналітиків, що, рано чи пізно, призведе до виявлення вразливостей і до успішних криптоаналітичних атак на даний алгоритм.

В результаті Рональд Рівест розробив алгоритм MD5 - дещо повільніший, ніж MD4, але маючи більш консервативний дизайн і враховуючий поточні результати і тенденції в криптоаналізі, що зумовило дещо вищу, порівняно з MD4, криптостійкість даного алгоритму.

Як і алгоритм MD4, MD5 виробляє 128-бітове хеш-значення для повідомлень перемінної довжини.

Перш за все, аналогічно алгоритму MD4, вхідне повідомлення розбивається на блоки по 512 бітів; останній з блоків завжди доповнюється до 512-бітового розміру таким чином: [14]

- додається одиничний біт;
- потім - нульові біти до розміру 448 бітів; нульові біти не додаються, якщо після додавання одиничного біта блок має 448-бітовий розмір;
- до послідовності додається 64-бітове значення, рівне розміром в бітах вихідної послідовності вхідних даних до доповнення;
- якщо розмір повідомлення перевищує 2^64, то записується значення розміру по модулю 2^64.

Аналогічно алгоритму MD4, для зберігання проміжних значень після обробки кожного блоку повідомлення використовуються 32-бітові регістри A, B, C і D. Вектор ініціалізації алгоритму MD5 еквівалентний співвідношенню в MD4. 512-бітові блоки доповненого повідомлення M почергово обробляється таким чином:

1. Блок вхідних даних представляється у вигляді 16 слів, кожне з яких є 32-бітовим.
2. Вміст регістрів A..D копіюється в тимчасові змінні a..d.

3. Виконуються 64 ітерації перетворень, в кожній з яких модифікуються змінні $a..d$ за участю одного із слів блоку повідомлення M[0]..M[15]:

1. t = b + ((a + f(b, c, d) + M[xi] + T mod 232<<< s)mod 232)
2. a = d;
3. d = c;
4. c = b;

5. b = t, де:

j – номер ітерації, j = 1..64(кожні 16 ітерацій формують один з чотирьох раундів алгоритму);

t – тимчасова 32-бітна змінна;

f()– одна з наступних раундових функцій:

Номер ітерації j , f()	
1…16	f(x, y, z) = (x & y) | (x & z)
17…32	f(x, y, z) = (x & y) | (y & z)
33…48	f(x, y, z) = (x y z)
49…64	f(x, y, z) = (x | z)

Походження псевдовипадкових констант K таке:

K = trunc(abs(sin(j))*232, де j береться в радіанах.

<<< - Операція побітового циклічного зсуву вліво; кількість бітів зсуву s визначається в залежності від номера ітерації.

4. Остання ітерація при обробці кожного блоку повідомлення відрізняється від інших тим, що в ній виконується тільки наступна операція:

a = b + ((a + f(b, c, d) +M[xi] + Timod 232)<<<s) mod 232.

Значення регістрів A..D складаються по модулю 2^32 з отриманими значеннями змінних a..d відповідно.

Вихідне значення алгоритму, тобто 128-бітове хеш-значення повідомлення - результат конкатенації регістрів A..D після обробки останнього блоку розширеного повідомлення.

Основні відмінності алгоритму MD5 від MD4 такі:

- в алгоритмі виконується 4 раунди замість трьох;
- змінені раундові функції f(); зокрема, більше не використовується функціяf(x, y, z) = (x & y) | (x & z) | (y & z), властивості якої викликали сумніви у криптологів ще в 1990 р.

- в кожну ітерацію додана додаткова операція додавання зі змінною b (саме змінна b модифікується в кожній попередній ітерації, за винятком першої) по модулю 2^32, яка прискорює вплив значень бітів вхідних даних на біти вихідного значення;
- в кожному раунді тепер використовуються унікальні модифікуючі константи;
- змінений (на менш упорядкований) порядок обробки вхідних слів блоку повідомлення, а також оптимізовані величини s.

Алгоритм MD5 не виглядає більш складним в реалізації, ніж алгоритм MD4. Всі зміни в алгоритмі є досить локальними, але в сукупності автору алгоритму вдалося вирішити завдання його посилення за рахунок деякого погіршення швидкісних якостей алгоритму (MD4 приблизно в 1,5 рази швидше, ніж MD5).

Алгоритм безпечного хешування (Secure Hash Algorithm, SHA)

NIST, разом з NSA, для Стандарту цифрового підпису (Digital Signature Standard, див. Розділ 20.2) розробив Алгоритм безпечного хешування Secure Hash Algorithm (SHA). (Сам стандарт називається Стандарт безпечного хешування (Secure Hash Standard, SHS), SHA - це алгоритм, використовуваний в стандарті) у відповідності з Federal Register:

Пропонується Федеральний стандарт обробки інформації (Federal Information Processing Standard, FIPS) для Стандарту безпечного хешування (Secure Hash Standard, SHS). Ця пропозиція визначає Алгоритм безпечного хешування (Secure Hash Algorithm, SHA) для використання разом зі Стандартом цифрового підпису (Digital Signature Standard).

Крім того, для додатків, в яких не вимагається цифровий підпис, SHA повинен використовуватися у всіх Федеральних додатках, в яких знадобиться алгоритм безпечного хешування.

Цей Стандарт визначає Алгоритм безпечного хешування (Sccu.ro Hash Algorithm, SHA), необхідний для забезпчення безпеки Алгоритму цифрового підпису (Digital Signature Algorithm, DSA). Для будь-якого вхідного повідомлення довжиною менше 264 бітів SHA видасть 160-бітовий результат, званий коротким змістом повідомлення. Далі, короткий зміст повідомлення стає входом DSA, який обчислює підпис для повідомлення. Підписування короткого змісту замість всього повідомлення часто підвищує ефективність процесу, так як короткий зміст повідомлення набагато менше, ніж саме повідомлення. Той же короткий зміст повідомлення повинен бути отриманий тим, хто перевіряє підпис, якщо прийнята їм версія повідомлення використовується як вхід в SHA. SHA називається безпечним, оскільки він розроблений так, щоб було обчислювально неможливо знайти повідомлення, відповідне даному короткому змісту повідомлення або знайти два різних повідомлення з однаковим коротким змістом повідомлення. Будь-які зміни, що відбулися при передачі повідомлення, з дуже високою ймовірністю призведуть до зміни короткого змісту повідомлення, і підпис не пройде перевірку. Принципи, що лежать в основі SHA аналогічні використаним професором Рональдом Л. Рівестом з MIT при проектуванні алгоритму короткого змісту повідомлення MD4 [20]. SHA розроблений за зразком згаданого алгоритму.

SHA видає 160-бітове хеш-значення, більш довге, ніж у MD5.

Опис SHA

По-перше, повідомлення доповнюється, щоб його довжина була кратна 512 бітам. Використовується те же доповнення, що і в MD5: спочатку додається 1, а потім нулі так, щоб довжина отриманого повідомлення була на 64 біта менше числа, кратного 512, а потім додається 64-бітове представлення довжини оригінального повідомлення. [14]

Ініціалізуються п'ять 32-бітових змінних (в MD5 використовується чотири змінних, але розглянутий алгоритм повинен видавати 160-бітове хеш-значення):

A = 0x67452301

B = 0xefcdad89

C = 0x98badcfe

D = 0x10325476

E = oxc3d2elf

Потім починається головний цикл алгоритму. Він обробляє повідомлення 512-бітовими блоками і продовжується, поки не вичерпаються всі блоки повідомлення.

Спочатку п'ять змінних копіюються в інші змінні:

A в a, B в b, C в c, D в d і E в e.

Головна ітерація має в собі чотири етапи по 20 операцій в кожному (у MD5 чотири етапи по 16 операцій для прикладу). Кожна операція являє собою нелінійну функцію над змінними a, b, c, d і e, а потім виконується зсув і складання аналогічно MD5. У SHA використовується наступний набір нелінійних функцій:

$$f_{i=0...19}(XYZ) = (X \hat{} Y) \vee ((\neg X) \wedge Z)$$

$$f_{i=20...39}(XYZ) = (X \oplus Y \oplus Z)$$

$$f_{i=40...59}(XYZ) = (X \hat{} Y) \vee (X \hat{} Z) \vee (Y \hat{} Z)$$

$$f_{i=60...79}(XYZ) = (X \oplus Y \oplus Z)$$

в алгоритмі використовуються наступні чотири константи:

K = 0x5a827999, t=0..19

K = 0x6ed9eba1, t=20..39

K = 0x8f1bbcdc, t=40..59

K = 0xca62c1d6, t=60..79

Блок повідомлення перетворюється з 16 32-бітових слів в 80 32-бітових слів за допомогою наступного алгоритму:

$$W_i = M_i,\ 0 <= t\ <= 15$$

$$W_i = (W_{i-3} \oplus W_{i-6} \oplus W_{i-14} \oplus W_{i-16}) <<< 1,\ 16\ <= t <= 79$$

(В якості цікавого зауваження, в первісній специфікації SHA не було циклічного зсуву вліво. Зміна "виправляє технічну ваду, яка робила стандарт менш безпечним, ніж передбачалося" . NSA відмовилося уточнити справжню причину вади.) [13]

Якщо t - це номер операції (від 1 до 80),W являє собою t-ий підблок розширеного повідомлення, а <<<s - це циклічний зсув вліво на s бітів, то головний цикл виглядає наступним чином:

```
FOR t = 0 to 79

TEMP = a <<< 5 + f(bcd) + W + K

e = d

d = c

c = b <<< 30

b = a

a= TEMP
```

На рисунку показана одна операція. Операція по зрушенню змінних виконує ту ж функцію, яку в MD5 виконує використання в різних місцях різних змінних.

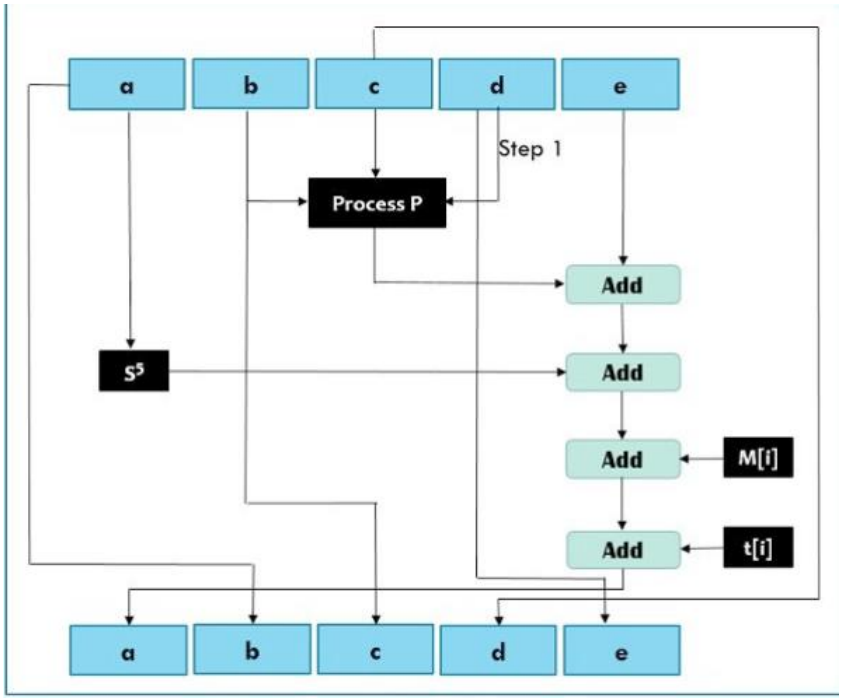


Рис.2.2. Одна операція SHA.

Після всього цього a, b, c, d і e додаються до A, B, C, D і E, відповідно, і алгоритм продовжується для наступного блоку даних. Остаточним результатом служить об'єднання A, B, C, D і E

Безпека SHA

SHA дуже схожа на MD4, але видає 160-бітове хеш-значення. Головною зміною є введення розширюючого перетворення і додавання виходу попереднього кроку в наступний з метою отримання більш швидкого лавинного ефекту. Рон Рівест опублікував цілі, переслідувані ним при проектуванні MD5, але розробники SHA цього не зробили. Ось поліпшення, внесені Рівестом в MD5 щодо MD4, і їх порівняння з SHA:

- Додався четвертий етап." У SHA також. Однак у SHA на четвертому етапі використовується та ж функція f, що і на другому етапі.
- Тепер в кожній дії використовується унікальна константа, що додається." SHA дотримується схеми MD4, повторно використовуючи константи для кожної групи їх 20 етапів.
- Функція G на етапі 2 з (X ^Y) v (X ^Z) v (Y ^Z)
- була змінена на (X ^Z) ∨ (Y ^Z), щоб зробити G менш симетричною. У SHA використовується версія функції з MD4: (X ^Y) ∨ (X ^Z) ∨ (Y ^Z).
- Тепер кожна дія додається до результату попереднього етапу. Це забезпечує більш швидкий лавинний ефект. Ця зміна була внесена і в SHA. Відмінність полягає в тому, що в SHA додана п'ята змінна до, b, c і d, які вже використовуються в f. Ця незначна зміна робить застосування злому MD5 ден Боєром і Босселаєрсом неможливим по відношенню до SHA.
- Змінився порядок, в якому використовувалися підблоки повідомлення на етапах 2 і 3, щоб зробити шаблони менш схожими. SHA в цьому місці зовсім відрізняється, так як використовує циклічний код виправлення помилок.
- Значення циклічного зсуву вліво на кожному етапі були наближено оптимізовані для прискорення лавинного ефекту. Чотири зсуву, використовувані на кожному етапі, відрізняються від значень, використовуваних на інших етапах. SHA на кожному етапі використовує постійне значення зсуву. Це значення - взаємно просте чисте розміром слова, як і в MD4.

Це призводить до наступного висновку: SHA - це MD4 з додаванням розширюючого перетворення, додаткового етапу і поліпшеним лавинним ефектом. MD5 - це MD4 з поліпшеним бітовим хешуванням, додатковим етапом і поліпшеним лавинним ефектом.

Відомості про успішні криптографічні зломи SHA відсутні. Так як ця односпрямована хеш-функція видає 160-хеш-значення, вона стійкіша до злому грубою силою (включаючи злом методом дня народження), ніж інші 128-бітові хеш-функції.

HAVAL

HAVAL - це односпрямована хеш-функція змінної довжини [21]. Вона є модифікацією MD5. HAVAL обробляє повідомлення блоками по 1024 біта, в два рази більшими, ніж у MD5. Використовується вісім 32-бітових змінних зчеплення, в два рази більше, ніж в MD5, і змінне число етапів, від трьох до п'яти (в кожному 16 дій). Функція може видавати хеш-значення довжиною 128, 160, 192,224 або 256 бітів.

HAVAL замінює прості нелінійні функції MD5 на сильно нелінійні функції 7 змінних, кожна з яких задовольняє суворому лавинному критерію. На кожному етапі використовується одна функція, але при кожній дії вхідні змінні переставляються різним чином. Використовується новий порядок повідомлення, і при кожному етапі (крім першого етапу) використовується своя константа, що додається. В алгоритмі також використовується два циклічних зсуву.

Блок – одна з основних складових будь-якого блокчейну. Блок містить інформацію про транзакції, тимчасові мітки та інші істотні метадані. Величезну роль у забезпеченні безпеки відіграє можливість стискати великі обсяги інформації про глобальний стан мережі в короткі стандартні повідомлення, які при необхідності можна легко перевірити. Ці повідомлення і є хеш. Відповідно, якщо змінити хоча б один символ вхідного значення, вийде зовсім інший хеш!

Криптографічні хеші використовуються скрізь, починаючи із зберігання паролів і закінчуючи системами перевірки. Їх суть полягає у використанні детермінованого алгоритму, який бере вхідні дані і створює рядок з фіксованою довжиною. Таким чином, ті ж вхідні дані завжди будуть перетворені в однакові вихідні дані.

Детермінізм важливий не тільки для хешів, адже навіть якщо змінити один біт вхідних даних, вийде зовсім інший хеш. [13]

Проблема алгоритмів хешування полягає в неминучості колізій. Так як довжина рядка такого хешу фіксована, то можуть існувати і інші вхідні дані, що утворюють той же хеш. Колізії — це погано. Це означає, що зловмисник може створювати колізії за запитом, може передавати шкідливі файли або дані з коректним хешем і видавати ці дані за правильні. Хороша функція хешування повинна максимально ускладнювати для зловмисників процес пошуку шляхів створення вхідних даних з тим же значенням хешу.

Процес розрахунку хешу не повинен бути занадто ефективним, так як в цьому випадку зловмисники легко можуть обчислити колізії. Алгоритми хешування повинні бути стійкими до атак знаходження прообразу. Необхідно максимально ускладнювати процес розрахунку кроків для знаходження вихідного значення, з якого був створений хеш (наприклад, прообразу).

Необхідно, щоб спосіб обчислення x для s = hash(x) був практично неможливим.

Отже, «гідні» алгоритми хешування повинні мати три властивості:

- якщо змінити один біт вхідних даних, повинен утворитися лавинний ефект і вийти зовсім інший хеш;
- невелика ймовірність колізій;
- ефективність, яка не жертвує колізійною стійкістю.

Різноманітність і еволюція алгоритмів хешування

SHA1 і SHA2

АНБ вже протягом довгого часу є піонером по створенню стандартів алгоритмів хешування. Саме АНБ запропонував алгоритм Secure Hashing Algorithm або SHA1 з фіксованою довжиною вихідного значення в 160 біт. На

жаль, SHA1 набагато перевершував MD5 завдяки збільшенню значення, кількості односторонніх операцій та їх складності, але цей хеш не пропонував грунтовних поліпшень захисту від більш потужних машин, що створюють різні вектори атаки.

SHA3

У 2006 році Національний інститут стандартів і технологій США (NIST — National Institute of Standards and Technology) оголосив конкурс на створення альтернативи алгоритмом SHA2, яка повинна була фундаментально відрізнятися за своїм дизайном і стати новим стандартом. З схеми алгоритмів хешування під назвою KECCAK (вимовляється як «кечак») з'явився алгоритм SHA3.

Хоча SHA3 і має схожу з попередніми алгоритмами назву, він сильно відрізняється за своєю внутрішньою структурою механізмом криптографічної губки. Цей механізм здійснює випадкові перестановки для поглинання і створення даних, служачи джерелом випадковості для вхідних даних, які будуть потрапляти в алгоритм хешування в майбутньому.

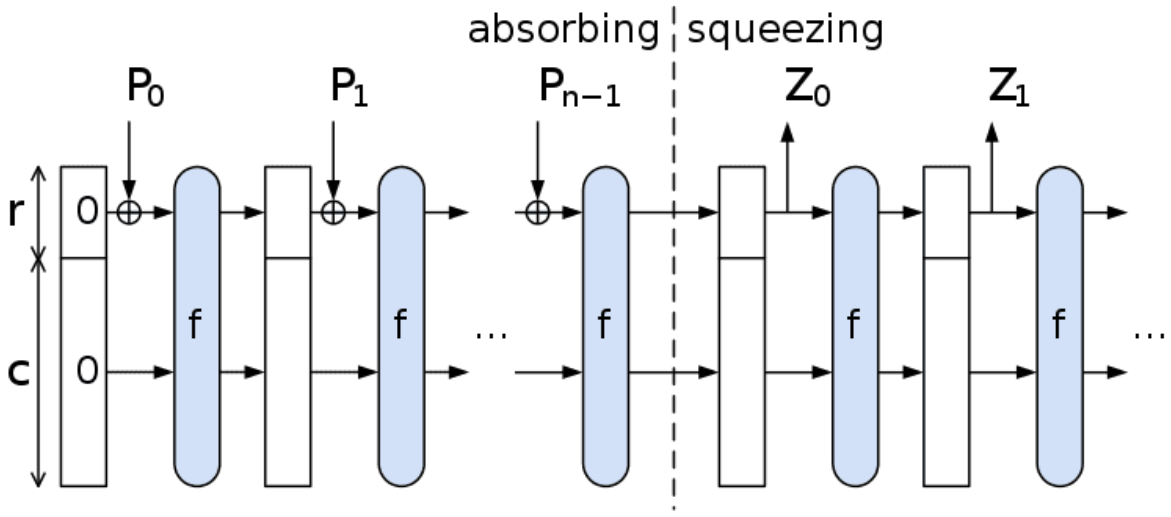


Рис 2.3. Ітерація SHA-3

Обробка вхідних даних за допомогою криптографічної губки KECCAK256

SHA3 зберігає початковий стан з великою кількістю бітів інформації, ніж у вихідному значенні, і тим самим перевершує обмеження колишніх алгоритмів. Цей алгоритм став стандартом NIST в 2015 році.

Хешування і доказ виконаної роботи

В рамках впровадження алгоритму хешування в протокол блокчейну для алгоритму доказу виконаної роботи Bitcoin використовував SHA256, а пізніше Ethereum — модифіковану версію SHA3 (KECCAK256). При виборі хеш-функції для блокчейну з доказом виконаної роботи дуже важлива ефективність обчислення хешу.

SHA256 біткойн може бути вкрай ефективно обчислений за допомогою спеціального апаратного забезпечення – спеціалізованих інтегральних мікросхем (ASIC-Application Specific Integrated Circuits). Про використання ASIC в майнінгових пулах та про те, як вони призводять до централізації обчислення, написано дуже багато. Доказ виконаної роботи провокує створення пулів з груп ефективних в обчислювальному відношенні машин, завдяки чому збільшується так звана «хеш-потужність» або кількість хешів, які може обчислювати машина за певний період часу.

Подвійний SHA256

Біткойн хешує дані за допомогою SHA256 незвичайним способом: він запускає в протоколі два види алгоритму. Потрібно розуміти, що це не міра проти атак «днів народження», тому що якщо hash(x) = hash(y), то hash(hash(x)) = hash(hash(y)). Подвійний алгоритм SHA256 використовується для боротьби з атакою подовженням повідомлення.

Якщо зловмисники знають довжину вхідних даних хешу, то можуть змусити хеш-функцію взяти певну частину внутрішнього стану, додавши секретний рядок до значення хешу. Через цю нестачу алгоритму SHA256 з сімейства алгоритмів SHA2 Біткойн обчислює хеш двічі.

Ethereum 2.0 і BLAKE

SHA3 – не єдиний винахід, що з’явився в результаті конкурсу NIST в 2006 році. Друге місце зайняв алгоритм BLAKE. Для сегментування Ethereum 2.0 більш ефективне хешування — це практично обов’язкова вимога, яка серйозно досліджується командами розробників. Ретельному аналізу піддається алгоритм

хешування BLAKE2b — сильно вдосконалена версія BLAKE,через свою фантастичну ефективність в порівнянні з KECCAK256 і високого рівня безпеки.
На сучасному процесорі обчислення за допомогою BLAKE2b проводиться в три рази швидше, ніж за допомогою KECCAK. [13]

Отже, використовуючи отримані знання можна зробити висновок про те, який алгоритм буде використано в створюваній інформаційній системі. Оскільки планується мати як мінімум одну ноду, яка буде постійно підтримувати блокчейн у включеному стані і не надавати звичайним користувачам доступу до валідації - найбільш підходящим буде рішення з використання алгоритму на базі SHA-256 замість -1, для зменшення кількості колізій та покращеної відмовостійкості

2.3 Найрозповсюджені способи криптографічного взлому алгоритмів

«Brute force» перекладається з англійської як «груба сила», що описує суть брутфорс-атаки. Під час неї відбувається підбір всіх можливих варіантів пароля і триває це до тих пір, поки не вийде його вгадати. Однак на вгадування по-справжньому складних паролів можна витратити кілька років, ніж навряд чи хтось буде займатися. Тому вони практично на 100% відсотків зможуть захистити вас від успішних спроб перебору.

Брутфорс-атаці легко піддаються паролі, що складаються з простих комбінацій знаків, розташованих по сусідству на клавіатурі. Це може бути ряд цифр («123456») або комбінацій букв («qwerty»). До ненадійним паролів також відносяться і ті, які представляють собою дату («18051991») або осмислене слово, особливо «admin» і «password». Навіть якщо ви виберете в якість пароля російське слово, набране з англійською розкладкою («gfhjkm»), його теж можна буде вгадати.

Щоб зламати акаунт з паролем середньої складності, може знадобитися всього 2-3 години. Чим простіше пароль, тим швидше атака досягне своєї мети. Для

перебору паролів використовується спеціальне програмне забезпечення, яке або створюється самими кіберзлочинцями, або запозичується ними у своїх колег. А в якості потужностей використовують уже раніше зламані комп'ютери і сервери - і, можливо, зламані вони були також за допомогою брутфорса.

Аккаунт з обчисленим паролем можна використовувати не тільки для злому нових акаунтів, але і для організації розсилки спаму. А якщо мова про акаунт адміністратора сайту, але після його злому зловмисники можуть розмістити на сайті шкідливий код. Крім того, метод повного перебору використовується для отримання доступу до секретних файлів або до конфіденційної інформації користувача.

Сучасні системи мають досить надійні способи захисту від брутфорс-атак на облікові записи. Ефективно боротися з ними допомагає обмеження спроб входу в аккаунт: наприклад, якщо користувач три рази поспіль набрав невірну пару логін-пароль, наступну спробу він зможе здійснити тільки через 15 хвилин. Також нерідко при невдалій спробі входу користувачеві пропонують пройти капчу, що шкідливе ПО для перебору паролів вже навряд чи зможе зробити.

Широко застосовується і двухфакторная аутентифікація. При її використанні юзера просять ввести не тільки логін і пароль, а й, наприклад, код, який відправляється на його номер телефону. До речі, ми реалізували свій власний механізм двофакторної аутентифікації за допомогою програми «Джіно.Ключ». Також для додаткового підтвердження особи користувача враховується пристрій, з якого здійснюється вхід, і місце розташування користувача.

Однак в деяких системах і додатках таких заходів захисту може і не бути. І тоді брутфорс-атаку можна легко провести, а зафіксувати її зазвичай буває непросто. Адміністратора сайту повинен насторожитися, коли невідомий користувач з одного і того ж IP-адреси починає наполегливо намагатися увійти в систему.

Відстежити це можна, перш за все, за допомогою лог-файлів. Але краще зробити так, щоб таке зовсім не відбувалося.

Злом хешів

Один з перших стандартів алгоритмів хешування — хеш MD5. Цей алгоритм користувався популярністю для перевірки цілісності файлів (контрольні суми) та зберігання хешованих паролів в базах даних веб-додатків. Його функціонал досить простий — він видає рядок з фіксованою довжиною 128 біт для кожного вхідного значення і використовує стандартні односторонні операції в декількох циклах для обчислення детермінованого вихідного значення. Через коротку довжину вихідного значення і простоту операцій хеш MD5 дуже легко піддається злому, зокрема атаці “днів народження”.

Атака «днів народження»

Існує відоме твердження про те, що якщо в кімнаті знаходяться 23 людини, то шанс того, що двоє з них народилися в один день, дорівнює 50%. Якщо в кімнаті знаходяться 70 осіб, то ця ймовірність збільшується до 99,9%. Це відбувається згідно з принципом Діріхле, який стверджує, що якщо помістити 100 голубів в 99 коробок, то в одній з коробок виявиться два голуба. Обмеження фіксованої довжини вихідного значення означає, що існує фіксований рівень комбінацій для колізій.

Хеш MD5 настільки вразливий до колізій, що навіть на простому процесорі Pentium з частотою 2,4 ГГц, можна обчислити колізії хешу всього за кілька секунд. Більш того, широке застосування цього хешу на зорі існування мережі створило мільйони прообразів MD5, які можна було знайти за допомогою звичайного пошуку Google по їх хешу.

2.4 Опис алгоритму PoW, переваги та недоліки

Для підтвердження проведення транзакцій криптовалют різні алгоритми, проте одним з із найпоширеніших залишається Proof of Work [2] (PoW, "пруф оф ворк", доказ виконаної роботи) - на ньому засновані основні блокчейн-мережі.

Водночас алгоритм PoW виконує відразу кілька найважливіших для існування цифрових активів функцій, деякі з яких не зовсім очевидні.

Майнінг - підбір (пошук) хешу (ключа) до блоку, для його створення.
Ось як виглядає типовий блок:

* Hash: 00000000000001c21dbf4715d5da1a288061faa21e950dd8df6ae25c8b55d868
* Previous block:
000000000000056a7dcf283f627c2a17c55ffe1937a6ed2bc467d9c524311da2
* Difficulty: 1 690 895.803052 (Bits: 1a09ec04)
* Transactions: 184
* Total BTC: 4251.63216933
* Size: 58.913 kilobytes
* Merkle root:
98c5d975bf556f0344770eee7ab31688a1c108223c14cea908ff99b0ab8fe947
* Nonce: 3723473450

Для вирішення завдання системою видається хеш, в який валідатор повинен хешувати блок.

Валідатор робить першу спробу, в ході неї у нього обчислюється хеш, який не збігається з заданим, що означає, що система не підтверджує його, і триває подальший пошук ключа.

Складність видобутку блоку (знаходження збігаючого хешу) залежить від кількості валідаторів і їх потужностей. Складність виражається, наприклад в блокчейні bitcoin, в кількості нулів в хеші.

Кожен блок містить хеш попереднього блоку, утворюючи ланцюг. Блок змінити неможливо - можливо лише створити блок на тій же "висоті", який буде містити в собі hash попереднього блоку. Для проведення такого процесу потрібно виконати роботу по знаходженню всіх попередніх блоків. Висока складність (практична неможливість) цього процесу захищає ланцюжок блоків.

Алгоритм proof of work захищає від double-spending атаки (Подвійна трата або Double-spending називається вдале використання одних і тих же коштів двічі), тим, що верифікує кожну транзакцію, яка додається в ланцюжок блоків, на предмет того, що одиниці, що містяться в транзакції, не були використані раніше.

У вас на рахунку було х умовних одиниць, була виконана операція з їх витрати, транзакція підтверджена, а це означає що інформація про вашу витрату була записана в блокчейн і у вас не буде можливості витратити суму другий раз, тому що за даними в блокчейні ви суму вже витратили.

Переваги Proof of Work

Важливим фактором є неможливість влаштувати DDos-атаку. При DDoS-атаці тисячі користувачів підключаються до сервера і через неможливість сервера обробити операції всіх користувачів сайт починає зависати, в цей час нові користувачі не можуть здійснювати ніяких операцій. [2]

У випадку з блокчейном DDoS-атаку можливо влаштувати єдиним способом: необхідно створювати величезну кількість дрібних транзакцій. Блок в стандартних блокчейнах обмежений 1МБ, а це означає що на обробку великої кількості дрібних транзакцій майнери будуть витрачати дуже багато часу і "реальні" транзакції не будуть потрапляти в блоки, через це утворюється велика кількість непідтверджених транзакцій, а це означає що кожна угода може очікувати підтвердження добу.

Недоліки консенсусу

- Енергетичні витрати - велика кількість вузлів виробляють обчислення, але в реальності тільки один (перший) проводить успішну роботу і отримує винагороду.
- Вразливість до атаки “51” Якщо валідатор або пул валідатор контролює більше половини потужностей, то з'являється можливість

повністю контролювати мережу: вони можуть додавати нові блоки, маніпулювати двосторонніми операціями і не підтверджувати нові транзакції. Також "Атака 51%" може призвести до того, що недобросовісні майнери зможуть використовувати одну і ту ж монету кілька разів, відкликаючи здійснені з нею транзакції, що є double-spending або "подвійною тратою". При цьому атакуюча сторона не може змінювати інформацію в вже доданих блоках і генерувати нові криптовалюти.

За діаграмою можна сказати, що кожному з них далеко до 51% володіння всіх потужностей, однак це предметна картинка: пулами BTC.com і AntPool володіє одна і та ж компанія Bitmain, що займається виробництвом майнінг-устаткування ASIC.

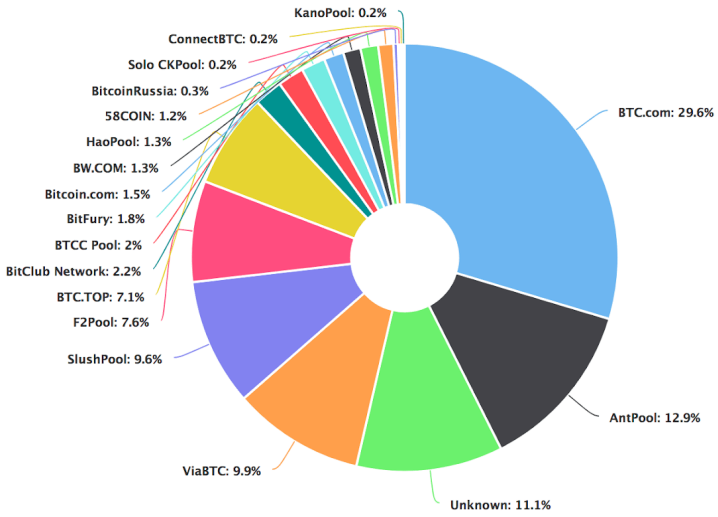


Рис. 2.4. Розподілення потужностей поміж компаній

2.5 Опис алгоритму PoS, переваги та недоліки

Proof-of-stake (PoS, "доказ володіння", "пруф оф стейк") - це метод захищеного шифрування за допомогою запитування у користувачів показати

власність певної кількості одиниць в розпорядженні. Він відрізняється від систем proof-of-work, які використовують алгоритми хешування для підтвердження електронних транзакцій

Принцип роботи

Proof-of-stake повинен мати спосіб визначати наступний дійсний блок в будь-якому блокчейні. Вибір за балансом рахунку в результаті призвів би до (небажаної) централізації, оскільки єдиний найбагатший член мав би постійну перевагу. Замість цього, були розроблені кілька способів відбору.

Можна використовувати рандомізацію для прогнозування наступного валідатора, використовуючи формулу, яка шукає найнижче значення хеша в комбінації з розміром ставки.

Переваги Proof-of-Stake

Блокчейни Proof of Stake можуть бути більш енергоефективними, ніж Proof of Work, здебільшого спирається на використанні енергії. генератора. У Proof-of-Stake ті, хто "охороняють" монети, завжди є тими, хто ними володіє (хоча деякі криптовалюти дозволяють або забезпечують надання потужності ставок в інші ноди). [2]

Недоліки PoS

Деякі автори сперечаються, що proof-of-stake не є ідеальним варіантом для розподіленого консенсус-протоколу. Одну проблему зазвичай називають проблемою "нічого поставленого", коли (в разі невдачі консенсусу) генераторам нічого втрачати, голосуючи за множинні історії блокчейна, що перешкоджає вирішенню консенсусу. Оскільки потрібно мало витрат при роботі в декількох ланцюгах (на відміну від систем proof-of-work), будь-хто може зловжити цією проблемою, щоб спробувати витратити двічі (у разі реорганізації блокчейна) "безкоштовно". Запропоновані рішення цієї проблеми: [2]

- Запропонований Ethereum протокол Slasher дозволяє користувачам додавати валідатора, який форжить на вершині більше однієї гілки блокчейна примусово лишати винагороди.
- Peercoin на ранніх етапах централізовано використовував контрольні трансляційні пункти (підписані особистим ключем розробника). Ніяка реорганізація блокчейна глибше, ніж останні відомі контрольні пункти, не була дозволена. Контрольні пункти зараз у коді згідно v0.6 і не застосовуються тепер, коли мережа досягла відповідного рівня поширення.

- Протокол Nxt дозволяє реорганізацію тільки останніх 720-ти блоків. Але, все ще: валідатор може слідувати форку з 721 блоку, незалежно від того, чи є це найвищим блокчейном, що попереджає консенсус.
- Гібрид "Докази згоряння" ("Proof of burn") і proof of stake. Блоки Proof of burn виступають в якості контрольних пунктів, не містять транзакцій, більш безпечні і прив 'язуються один до одного і ланцюги PoS, але є більш дорогими.
- Гібрид proof-of-work і proof-of-stake від Decred 's. Proof-of-stake як розширення залежить від формування часових міток proof-of-work, ґрунтуючись на пропозиції "докази активності", що націлене на вирішення проблеми "нічого поставленого", якщо майнери proof-of-work майнитимуть блоки, а proof-of-stake виступатимуть як другий механізм аутентифікації.

Статичні симуляції показали, що одночасний форжинг на декількох ланцюжках можливий, і навіть прийнятний. Однак, захисники proof-of-stake вважають, що більшість описаних сценаріїв атак неможливі, або так непередбачувані і існують тільки в теорії.

Валідація PoS

- Блок, що досягає мети складності валідується валідатором. Мета складності періодично коригується, щоб 1 блок PoW з'являється кожні 10 хвилин.
- Робоче представлення хешується 10 разів поспіль. Кожен наступний хеш відображається на індивідуальному не втраченому обсязі в блокчейні. Насправді, це лотерея з двома наборами з п 'яти переможців. Перші п'ять хешів належать до обов' язкових підписів, інші п 'ять - до добровільних.
- Якщо обов'язкові підписи відносяться до активних відкритих ключів, блок може потенційно вважатися дійсним. В іншому випадку, блок недійсний і повинен бути скинутий.
- Якщо майнер PoW знаходить потенційно дійсний блок, він відправляє наступний хеш в мережу. Якщо робоче подання досягає мети складності і відноситься до активних сторін, то блок передається через мережу. В іншому випадку, повідомлення позначається як спам.
- Перші п'ять обраних сторін послідовно підписують цей хеш і відправляють його далі.
- Після того, як послідовність обов 'язкового підпису завершена, остання сторона публікує блок PoW, а також свій власний блок PoS.
- Останні п'ять хешів належать до добровільних підписів. Ці добровільні підписи можуть бути вставлені в будь-який блок серед наступних 6 блоків як особливі txn. Вони не вимагають оплати.
- Перехід до кроку 1

Примітка: цей процес є одночасним, щоб численні хеші блоків могли циркулювати в мережі, намагаючись зібрати п 'ять підписів і створити пари блоків PoW/PoS. Пари блоків, що програють цю гонку, стають недійсними.

Неприпустимість стандартних векторів атаки

Поки зловмисники не володіють більшою часткою ставки, всі типи атак PoW є обчислювально нездійсненними.

Є два типи відомих атак:

1) подвійна трата

2) Відмова в обслуговуванні.

Розгляньмо приблизні погляди нижче. Числа настільки сприятливі, що розгляд точної статистики не особливо цікавий.

Подвійна трата

Подвійні витрати спираються на секретність. Щоб добувати блоки таємно, валідатор PoW повинен вибрати 5 зі своїх відкритих ключів у лотереї. Якщо майнер PoW володіє часткою 0 < s < 1 всіх монет, ймовірність того, що блок, який досягає мети складності, вибере монети майнера - (1/s) ^ 5. Для s = 0.01, з 10 мільярдів блоків один буде задовольняти заданому критерію. Навіть для вкрай малих значень сукупності хешів, непрактично в приватному порядку майнити в десять мільярдів разів швидше, ніж всі інші майнери разом. Для For s = 0.1, 1 з 100,000 блоків будуть задовольняти даному критерію (тобто, атака вимагає приблизно 99.999% всієї потужності хешування). Для s = 0.5, зловмисник матиме успіх, якщо контролюватиме 51% значень сукупності хешів.

Відмова в обслуговуванні

Зловмисник, який добуває блок відкрито, може просто виробляти порожні блоки PoW. Однак це не призведе до відмови в обслуговуванні. 50% всіх блоків у випадковому порядку будуть видобуті через PoS. Зловмисник не може змусити майнерів PoS створювати порожні блоки. Тому, він не може відмовити в обслуговуванні, незалежно від того, яке значення хеша він контролює.

2.6 Порівняння PoW та PoS

Спільного у Proof-of-Work і Proof-of-Stake не так багато. Хіба що у них є власна атака 51%, яка в підсумку призводить до краху мережі.

В цілому у Proof-of-Stake здається є ряд очевидних переваг: більш висока швидкість валідації, менші витрати ресурсів для захисту, менші комісії.

Але при цьому атакувати мережу з алгоритмом Proof of Work фактично нереально - для цього необхідно суперкомп 'ютер і кілька електростанцій для його обслуговування.

У Proof-of-Stake все вибудувано таким чином, що учасники прагнуть захопити якомога більшу частку коїнів, щоб отримувати більшу винагороду за комісію. Через це виникає централізація. Але навіть якщо дестабілізація мережі і не принесе нічого власникам мажоритарної Stake, все одно у цього алгоритму консенсусу є один недолік.

Йдеться про атаку Nothing-at-Stake - це коли створюється ланцюжок порожніх блоків групою користувачів, що в підсумку може призвести до подвійного витрачання, конфлікту версій блокчейна і неминучого форку. Саме над усуненням цієї проблеми займаються розробники нового протоколу Casper, який в майбутньому буде впроваджений в платформу Ethereum. Поки ніякої конкретики немає: ні дати переходу, ні технічних деталей. За однією з версій, у платформі учасники будуть ставити свої частки заради отримання винагород - але це поки не підтверджено.

У обох протоколів є свої переваги і недоліки. Proof-of-Stake економічно вигідніший і раціональніший з технічної точки зору, але в таких глобальних платформах як блокчейн Bitcoin або інших здається більш надійним варіантом.

Огляд альтернатив Proof of Work і Proof of Stake

Зі становленням криптовалют, і все більш глибоким розробок в області блокчейн були запропоновано й інші алгоритми, крім механізмів доказів роботи і частки. Деякі з них вже були реалізовані в нових криптовалютах, інші тільки на етапі проекту.

Таблиця 2.1. Порівняння протоколів підтвердження транзакцій

Назва протоколу	Суть
Proof-of-	Гібридний протокол між алгоритмом доказу роботи і частки.

Activity	Зазвичай використовується наступна схема: на початковому етапі видобуваються всі монети без запису транзакцій в блокчейн (PoW), а потім використовується PoS з майстернями. Класичний приклад – криптовалюта DASH.
Proof of Delegated Stake	Модифікований варіант POS, в якому відбувається делегація підтвердження частки. Учасники мережі можуть вибирати, хто з нод буде підтверджувати транзакції і голосувати за різні рішення в мережі. Застосовується до Bitshares.
Proof of Leased Stake	Можна перевести як доказ орендованої частки. Цей протокол впроваджено в платформу Waves. Суть полягає в тому, що в класичному PoS тільки ноди з великим стеком можуть підтверджувати транзакції і отримувати винагороди. У PoLS учасники з невеликими частками можуть здавати їх оренду нодам і отримувати винагороду теж. Схема нагадує пули зі звичайним майнінгом.
Proof-of-Burn	У цьому протоколі використовується спалювання монет. Учасники отруюють їх на якусь спеціальну адресу, де вони стають неактивним. Натомість вони отримують право майнити нові монети. Протокол застосовується в Slimcoin.
Proof-of-Signature PoSign	абсолютно новий механізм, який ще навіть не до кінця доопрацьований. Використовується у блокчейні криптовалюти XTRABYTES. Ідея полягає в тому, що кожен зі статистичних нодів мережі підписує нові блоки. Якщо нода спробує провести атаку, то вона потрапляє в чорний список.
Proof-of-	Тут для доказу використовується простір для зберігання даних. Чим

Capacity	більше його - тим більше ти майниш. Піонер PoC - криптовалюта Burst.
Proof-of-Brain	Іноді цей термін використовують, щоб описати принцип роботи Steemit і Golos. Тут учасникам для "майнінгу" потрібно створювати контент, чи то пак, вмикати мізки.
Proof of Importance	Доказ важливості - це алгоритм консенсусу в мережі криптовалюти NEM. Важливість "вираховується" як комбінація поточного балансу і транзакційної активності учасника.

2.7 Рішення масштабування другого рівня

Оскільки записи є по суті частиною блокчейну, відповідно – є записами хоч і до розподіленої але все ж бази даних. Відповідно, для масштабування можна використовувати стандартні методи та засоби

ROLAP — OLAP-системи, які мають прямий доступ до існуючих БД або використовують дані, завантажені у власні локальні таблиці.

MOLAP - розшифровується як багатовимірна онлайн-аналітична обробка. Це тип OLAP-процесу, який використовує багатовимірну модель даних, завдяки чому аналіз даних можна зробити легко. Дані в MOLAP попередньо обчислюються, попередньо узагальнюються і зберігаються в MOLAP. MOLAP має можливість зберігати різні перестановки та комбінації даних, які вже зберігаються у багатовимірному масиві. Це означає, що він обробляє дані, які вже зберігаються у визначеному багатовимірному масиві. До всіх комірок наявних даних можна отримати доступ безпосередньо з масиву. Як результат, MOLAP працює швидше і дає відповіді на аналітичні дані

ROLAP та MOLAP

ROLAP є альтернативою технології MOLAP (багатовимірна OLAP). Хоча обидві технології, ROLAP і MOLAP, є аналітичними інструментами, призначеними для аналізу даних за допомогою багатовимірної моделі даних, однак істотною відмінністю ROLAP є те, що вона не вимагає попереднього обчислення і зберігання інформації. Замість цього, ROLAP-інструменти надають доступ до даних реляційної бази даних та генерують SQL-запити для надання інформації на відповідному рівні за запитом користувача.

Обговорення переваг і недоліків ROLAP, загалом зосереджується на тих речах, які відноситься до найбільш широко використовуваних ROLAP і MOLAP інструментів, доступних сьогодні. У деяких випадках будуть інструменти, які є виключеннями для будь-яких зроблених узагальнень.

Переваги ROLAP

- ROLAP вважається більш масштабованою для обробки великих обсягів даних, особливо для моделей дуже великої розмірності (тобто з мільйонами елементів).
- Завдяки різноманітності доступних інструментів для завантаження даних, а також можливості для точного налаштування ETL коду для конкретної моделі даних, час завантаження даних, як правило, є набагато меншим, ніж у MOLAP-системах.
- Дані зберігаються в стандартній реляційній БД і доступ до них можна отримати, використовуючи будь-який SQL-інструмент створення звітності (окрім OLAP-інструментів).
- ROLAP-інструменти краще обробляють непоєднувані факти (наприклад, текстові описи). В MOLAP-інструментах, як правило, знижується швидкодія про подібній обробці.
- При змінній розмірності задачі, коли зміни в структуру вимірювань вносяться досить часто, ROLAP-системи з динамічним представленням

розмірності постають найкращим рішенням, оскільки в них такі маніпуляції не вимагають фізичної реорганізації БД.

- ROLAP-системи можуть використовувати елементи управління авторизацією БД, такі як безпека на рівні рядка, в результаті чого результати запиту фільтруються в залежності від заданих критеріїв, що встановлюються, наприклад, для певного користувача чи групи користувачів (оператор SQL WHERE).

Недоліки ROLAP

- Існує твердження, що ROLAP-інструменти мають меншу продуктивність, ніж MOLAP.
- Завантаження зведених таблиць повинно управлятися користувацьким ETL кодом. ROLAP-інструменти не допомагають у вирішенні цього завдання, що передбачає витрачання більше часу на розробку і більше коду для підтримки даної функції.
- Якщо пропустити етап створення зведених таблиць, то продуктивність запитів страждатиме через звернення даних запитів до більш деталізованих таблиць. Це можна частково вирішити долученням додаткових зведених таблиць, однак як і раніше є недоцільним створення таких таблиць для всіх комбінацій розмірності/атрибути.
- ROLAP-інструменти покладаються на БД загального призначення для запитів і кешування, отже, кілька спеціальних методів, що використовуються в MOLAP-інструментах, є недоступними (наприклад, спеціалізована ієрархічна індексація). Тим не менш, сучасні ROLAP-інструменти використовують останні поліпшення в мові SQL, такі як CUBE і ROLLUP оператори, DB2 Cube Views, а також інші SQL розширення OLAP. Ці вдосконалення SQL можуть частково усунути даний недолік.

Оскільки ROLAP-інструменти використовують SQL для всіх обчислень, то вони не придатні для моделей, де необхідно робити складні обчислення, які

неможливо здійснити завдяки SQL. Прикладами таких моделей є бюджетування, фінансова звітність та інші.

ВИСНОВКИ ДО РОЗДІЛУ 2

Зважаючи на недоліки реалізації мереж, такі як :

- Відсутність користувацького інтерфейсу

- Складність у використанні для звичного користувача

доцільно сфокусуватися на вирішення цих проблем, для того щоб зробити технологію більш доступною для широкого користувача. Відповідно, оскільки зараз є популярними веб-інтерфейси, для користувача буде зручним мати такого роду “місток” між складними технологіями і простими операціями.

Очевидно, що оскільки такі мережі з обробки транзакцій є розподіленими, то в них має бути серверна частина, спроектована таким чином, щоб не мати прив’язки до ролі користувача, оскільки це буде грубим порушенням самої суті таких P2P систем. Також, оскільки однією із задач є реалізація зручного інтерфейсу - слід також орієнтуватися на широкого користувача. Відповідно, для кодування серверної частини доцільно обрати мову програмування Python завдяки її легкості портування на різні платформи, великій кількості розробників, які використовують мову, а також гарному набору інструментів для реалізації класичних завдань по шифруванню/дешифруванню а також для розв’язання задач у сфері мережевої взаємодії. З точки зору користувацьких інтерфейсів - було обрано html/css/js для реалізації веб-додатку для користувачів. Таким чином, буде забезпечена максимальна зручність для користувачів.

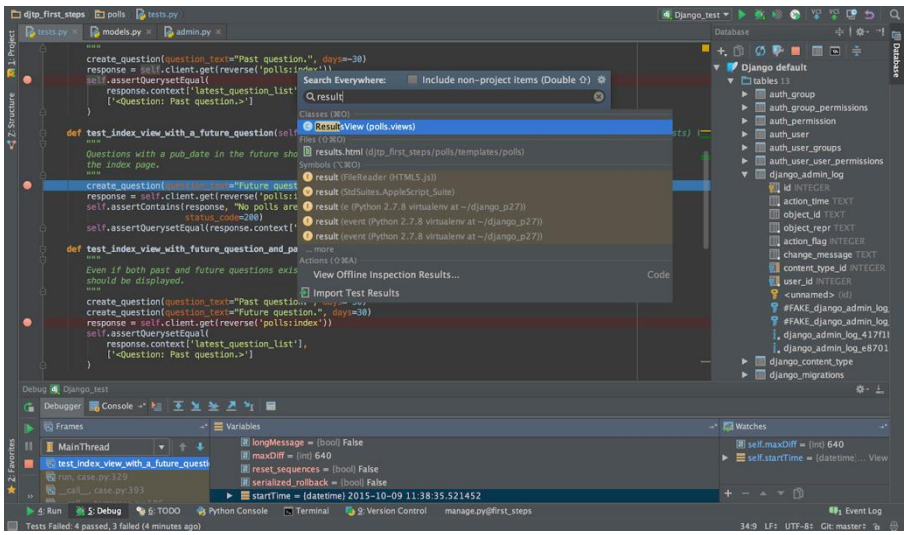
РОЗДІЛ 3. Розробка реалізації розподіленої інформаційної системи на базі блокчейн

3.1 Інтерфейс IDE

В ході аналізу було виявлено певну потребу для користувачів в наявності розподіленої інформаційної системи, серед основновних вимог до якої можна зазначити:

- реалізація функціонал обміну даними згідно із протоколами PoW/POS
- збереження децентралізованості як основної частини P2P систем
- ергономічний та ефективний, зрозумілий інтерфейс користувацький інтерфейс

Для вирішення поставленої задачі, було обрано мову програмування Python в якості платформи для описання серверної частини даної інформаційної системи, зважаючи на легкість у використанні, розвинене співтовариство, а також велику кількість фреймворків, які значно полегшують використання мережевих ресурсів. Також, досить важливим є фактор наявності великої кількості бібліотек, які беруть на себе реалізацію складних алгоритмів. Для написання фінального користувацького інтерфейсу обрано стандартну зв'язку, HTML+CSS+JS з використанням Ajax - запитів, для додаткової оптимізації перезавантаження сторінок. В якості інструмента для розробки дипломного проекту було обрано відповідний інструмент : Pycharm IDE [9], який є офіційною рекомендацією від великої корпорації JetBrains



Кафедра КІТ (47)				НАУ 20 06 03 000 ПЗ				
Виконав	Боскін К.О			Розробка реалізації розподіленої системи на базі блокчейн	Літера		Аркуш	Аркушів
Керівник	Савченко А.С				Д		75	11
Консульт.					УС-211М 122			
Н-контроль	Райчев І.Е.							

Рис 3.1. Інтерфейс PyCharm

PyCharm IDE [8] було обрано також через те, що даний інструмент у порівнянні з іншими має дуже серйозний арсенал різних додаткових модулів які спрощують життя розробникам. Так, наприклад, вбудований інструмент моніторингу кількості одночасних процесів допомагає аналізувати кількість енергії, які певними процесами поглинаються, як наслідок, ведуть до оптимізації витрати батареї та покращення продуктивності.

3.2 Структура проекту

Під час реалізації проекту було створено наступну структуру директорій, для більш зручного користування і для реалізації нагальних потреб проекту

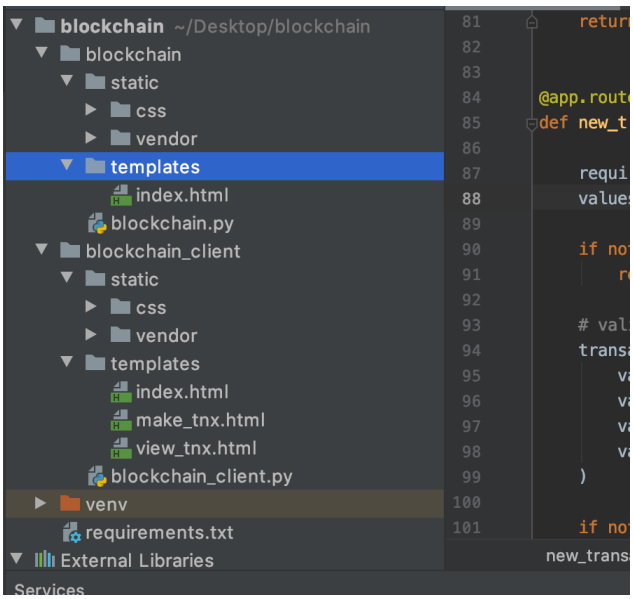


Рис 3.2. Структура проекту

3.2.1 Директорія Static

До директорії static входять статичні файли – це такі файли, які статично будуть використовуватися веб-додатком. Наприклад, субдиректорія Vendors містить всі бібліотеки і їх статичні файли які були чи будуть використовуватися проектом

Наприклад, для даного проекту використовуються наступні додаткові модулі :

- Bootstrap
- FontAwesome
- jquery
- datatables

Bootstrap — це опенсорсний набір інструментів розробників, який має відкритий код і призначений для створення веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Font Awesome - це шрифт з піктограмами, які можуть додаватися до будь-яких мегментів веб-сторінки, щоб підвищити їх наочність і поліпшити дизайн. Розробник Дейв Ганді зібрав істотний набір, який налічує з декілька сотень іконок і підходить під будь-які завдання.

jQuery — популярна JavaScript-бібліотека з відкритим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день

3.2.2 Директорія template

До теки templates входить програмний код html, який відправляється сервером у відповідь на певні запити. Наприклад, index.html є назвою-галузевим стандартом і означає сторінку, яка є репрезентацією сайту

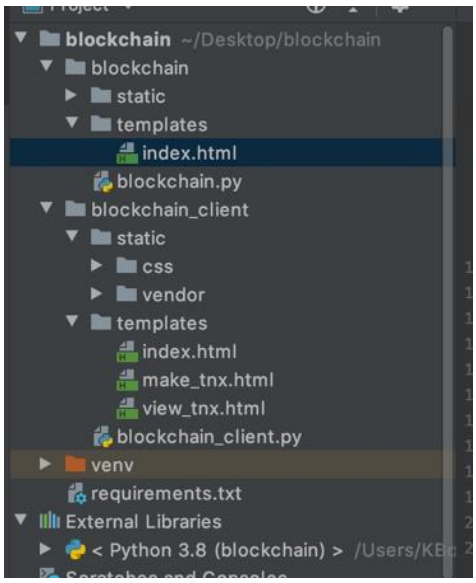


Рис 3.3. приклад index.html

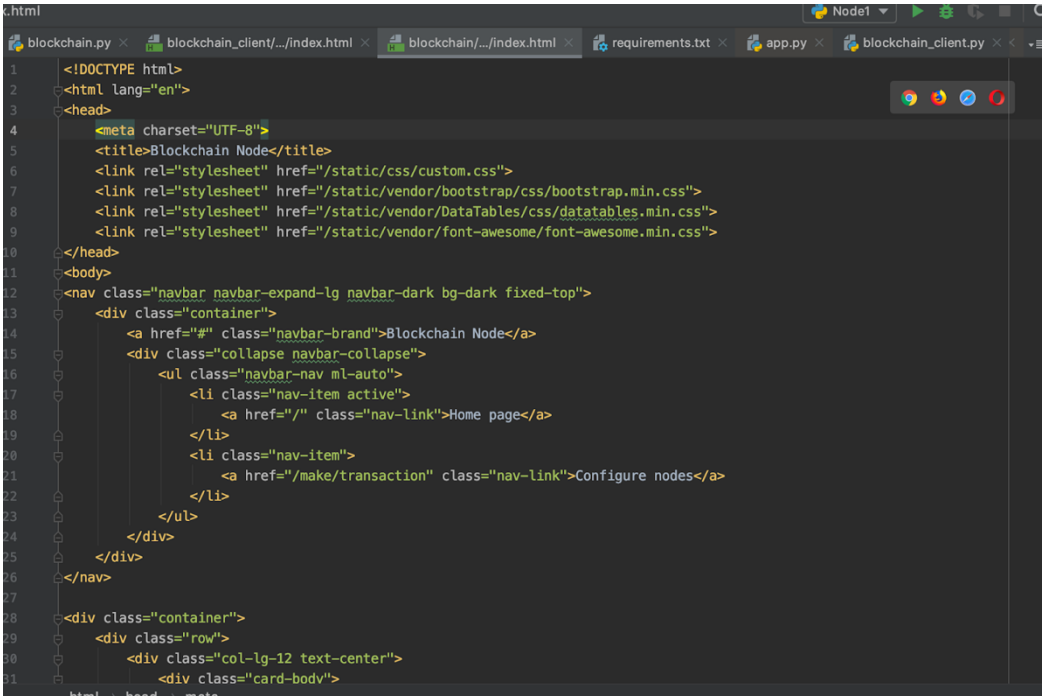


Рис 3.4. приклад index.html

3.2.3 Модулі blockchain та blockchain client

Python дозволяє розмістити класи, функції або дані в окремому файлі і використовувати їх в інших програмах. Файл, який містить початковий код на Python, є модулем. Об'єкти з модуля можуть бути імпортовані в інші модулі.

Ім'я файла утворюється додаванням до імені модуля розширення .py. При імпорті модуля інтерпретатор шукає файл спочатку у поточному каталозі, потім у каталогах, вказаних у змінній середовища PYTHONPATH, потім у залежних від платформи шляхах за замовчуванням. Ми можемо внести зміни в PYTHONPATH додавши туди свій шлях. Каталоги, в яких здійснюється пошук, можна подивитись у змінній sys.path.

Модулі blockchain та blockchain_client представляють собою відповідно мастерноду та звичайного користувача даної мережі. Тобто, фактично, кожний екземпляр користувача буде репрезентуватися кодом, який виконується в blockchain_client. Фактично, це дуже подібно до мікросервісної архітектури, хоча в даному випадку більшість мікросервісів навіть не будуть знати про існування одне одного, крім операцій, які їх зводять одна до одної

3.2.4 Діаграма послідовності дії системи

Діаграма послідовності — різновид діаграми в мові моделювання UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень

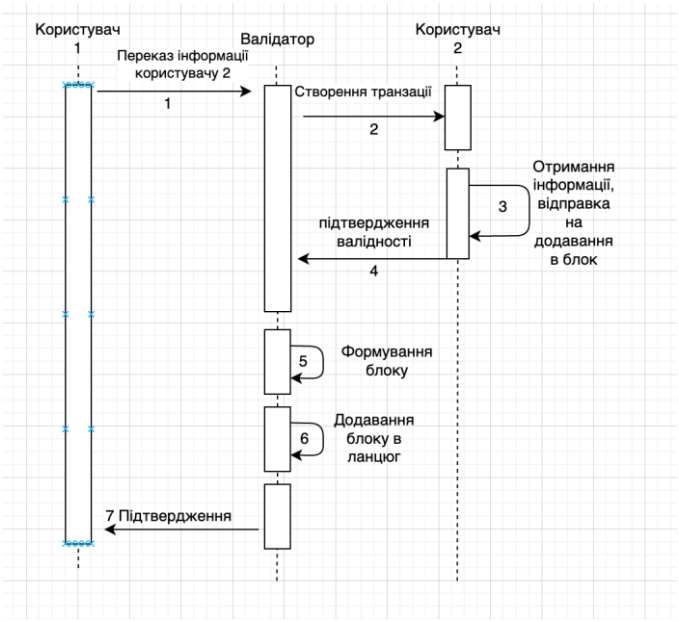


Рис 3.5. Діаграма послідовності дії системи

3.3 Створення гаманця

В залежності від типу браузера та версії ОС (Ubuntu, OSX, Windows) візуальне представлення системи може відрізнятися від наданого, але, при цьому, послідовність дій не буде змінюватися

Для запуску додатку його можна скомпілювати прямо з вихідного коду або ж зі згенерованого виконавчого файлу, прямо із робочого столу ОС

На екрані користувача розміщено генератор пар публічний-приватний ключ (власне, скринька та підпис), а також розміщено елементи для переходу на сторінки генерування транзакції (тобто переведення на скриньку іншого користувача), а також огляд транзакцій

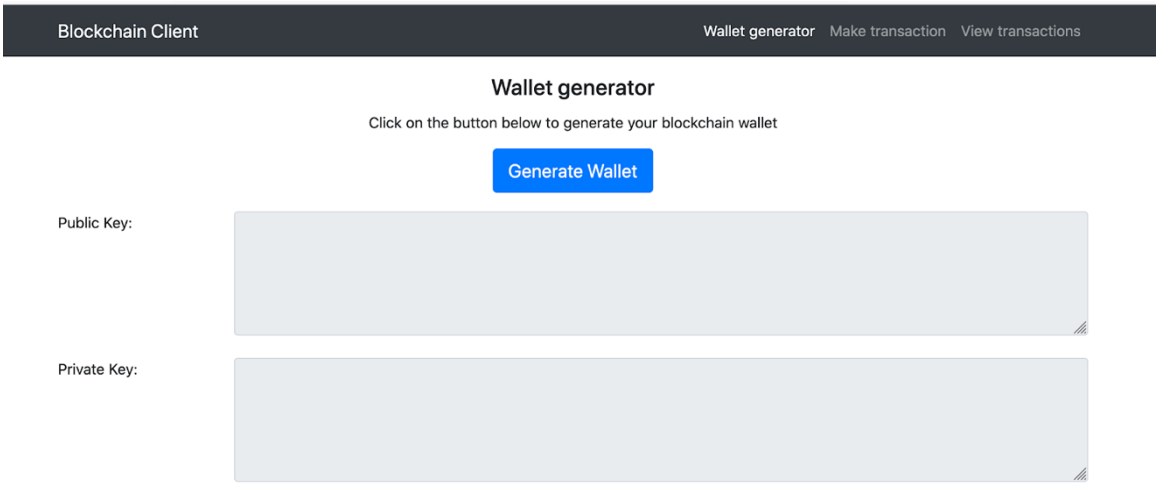


Рис 3.6. Екран користувача

Після натискання на кнопку “Generate Wallet”, відповідно дії генерації гаманця, екран буде виглядати наступним чином :

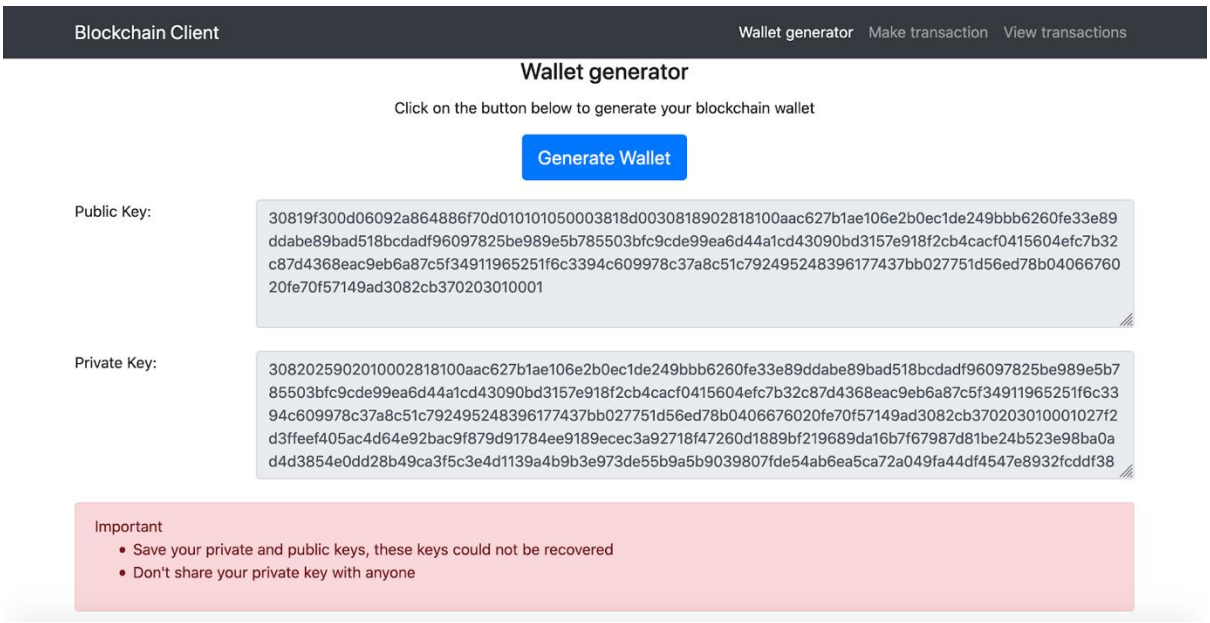


Рис 3.7. Згенерований гаманець

Знизу, для користувача буде відображено вікно із застереженням щодо безпеки а також надійного збереження сенситивних даних, коли користувач хоче виконати переведення, він може перейти на екран генерування транзакції

На екрані генерування транзакції розміщено форму, які відправляє POST запит на серверну частину. З точки зору користувацього інтерфейсу присутня інформаційна части (заголовок і підзаголовок) а також 4 поля вводи. Перше — призначено для введення адреси з якої слід зробити переказ, друге - приватний ключ користувача, необхідний для створення транзакції і її підпису. Третє — призначено для введення адреси на яку слід зробити переказ, друге — приватний ключ користувача, необхідний для створення транзакції і її підпису. І останнє поле є кількісною репрезентацією

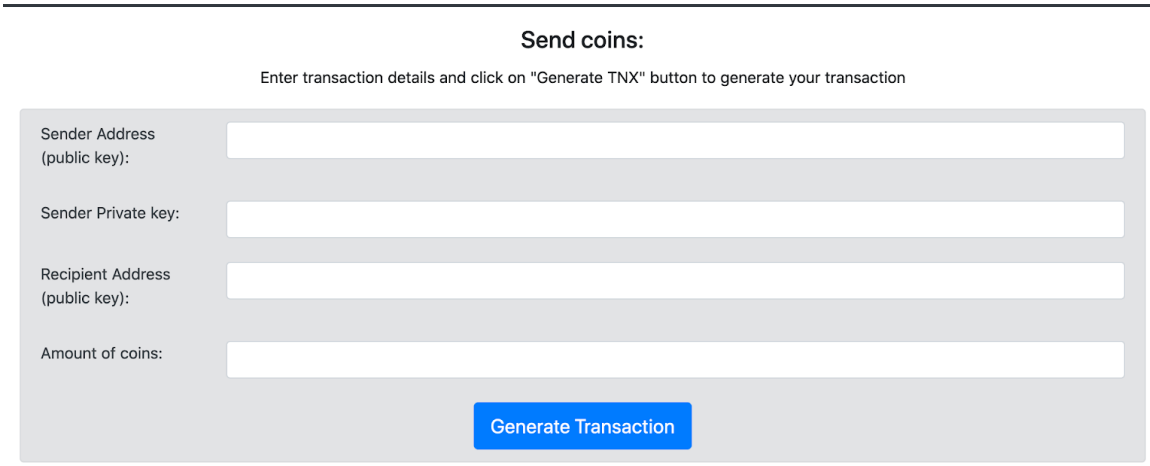


Рис 3.8. Генерування транзакцій

Після введення даних, екран виглядає наступним чином

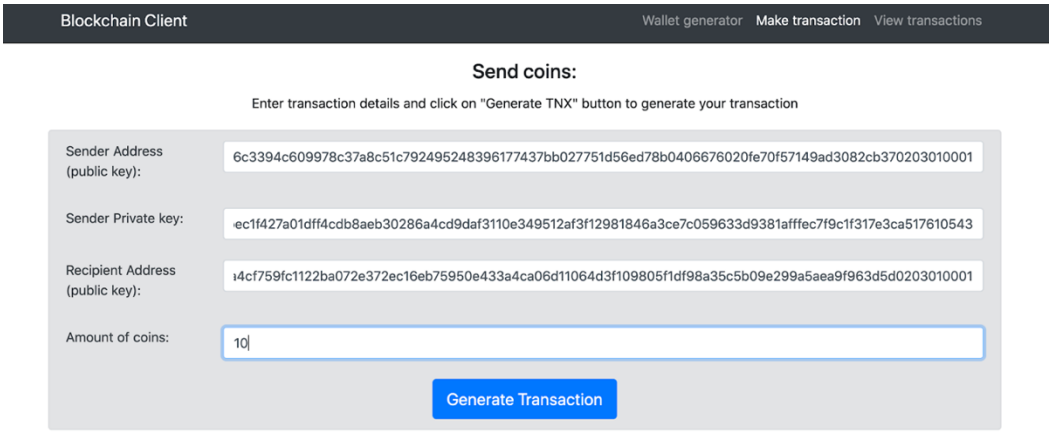


Рис 3.9. Генерування транзакцій

Після того, як користувач натискає на кнопку “Згенерувати транзакцію”, з’являється модальне вікно для остаточного коригування і підтвердження даних

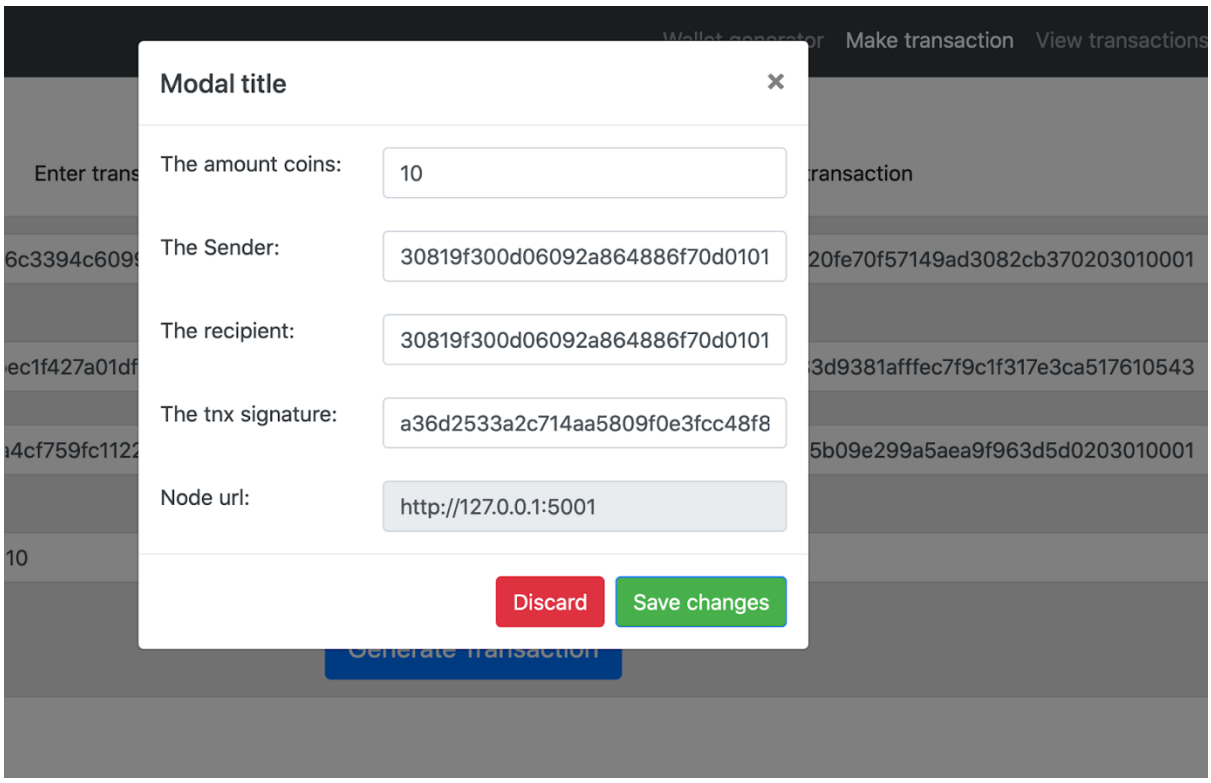


Рис 3.10. Генерування транзакцій

Форма також містить додаткові поля, такі як :

- підпис транзакції (це її геш)
- URL ноди валідатора, на який буде відправлена робота із валідування транзакції і створення блоку

За замовчуванням буде надано адресу мережі, в якій розгортається система, хоча, користувачам надається можливість персонального вибору валідатора, на користь якого буде відправлена комісія під час створення транзакції

Після того, як користувач підтвердить введення інформації - буде показане модальне вікно

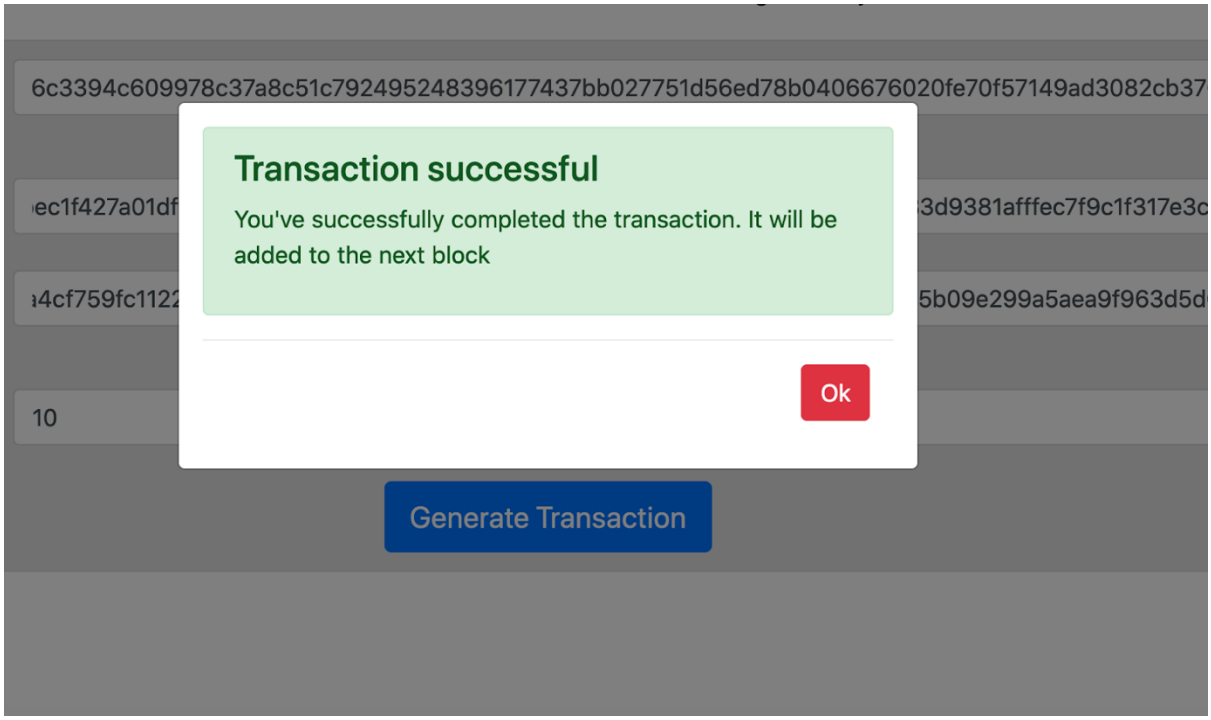


Рис 3.11. Генерування транзакцій

ВИСНОВКИ ДО РОЗДІЛУ 3

Таким чином у IDE PyCharm було розроблено мобільну інформаційну систему з обробки транзакцій розподіленого реєстру на базі мови програмування Python для серверної частини і використанням стандартного стеку технологій HTML+CSS+JS для реалізації веб-додатку системи. Додаток імпелемнтує функціонал із надання інтерфейсу користувачам для створення транзакцій, їх валідування, верифікації та остаточного підтвердження в рамках створеної P2P мережі.

Для реалізації серверної частини було обрано фреймворк flask, за рахунок функціонального підходу дуже зручно оперувати різними сутностями. Було реалізовано декілька конфігурацій запуску додатку для того, щоб продемонструвати у відповідних ролях, а також виконано відповідні налаштування IDE для подальшого запуску і використання реалізованого програмного забезпечення

IDE PyCharm було обрано для реалізації даної розподіленої інформаційної системи за рахунок ряду переваг, таких як зручність та велика кількість розробників, що автоматично означає біль широку аудиторію використання і як наслідок більші можливості із отримання інформації

ВИСНОВКИ

Під час дипломного проектування було надано відповіді на такі питання як доцільність існування розподілених систем, їх переваги та недоліки, а також розроблено функціональний застосунок, який імплементує переваги, необхідні для сучасних P2P систем

Визначено поняття розподілених інформаційних систем, їх головні характеристики (Функціональна повнота, Своєчасність, Функціональна надійність, Адаптивна надійність), а також переваги та недоліки їх використання. До переваг можна віднести децентралізованість системи а також швидкість обробки даних за рахунок багатьох учасників мережі

В результаті аналізу розподілених інформаційних систем можна зробити висновок, що розглянуті системи реалізують вирішення таких проблем:

- централізованість мережі
- швидкість проведення транзакцій
- можливості масштабування мережі

Розроблена у дипломній роботі система не має таких недоліків і відповідає наступним критеріям :

- Високий ступінь зрозумілості для будь-якого користувача
- Легка доступність з будь-якої точки
- Простий користувацький інтерфейс із мінімумом функцій, який допоможе ефективно використовувати всі можливості ІС з обробки транзакцій в розподілених реєстрах.

Під час проектування створено розподілену інформаційну систему на базі мови Python для кодування серверної частини а також використано мови

HTML/CSS/JS для створення веб-застосунку і надання користувачам веб-інтерфейсу для легкого доступу до функцій мережі

ДЖЕРЕЛА БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Корпоративные сети // Intuit.ru: Национальный Открытый Университет "ИНТУИТ": режим доступу: http://www.intuit.ru/studies/courses/1/1/lecture/20 (дата звернення: 17.11.2020). – Корпоративные сети.

2. Алгоритм PoW та PoS // bizlato.com: Відкритий блог: режим доступу: https://bitzlato.com/ru/blog/algoritm_proof_of_work_chto_takoe_pow_i_princip_raboty_bitzlato (дата звернення: 14.11.2020) - Алгоритмы P2P систем

3. Структура інформаційних систем // studfile.net: Лекція №3: режим доступу: https://studfile.net/preview/5064248/page:9/ (дата звернення: 14.11.2020) - Назва з екран

4.HLRLookup, // https://www.amdtelecom.net/services/numbers-insight/hlr-lookup/ : HLR Lookup, дата зверн. 12.11.2020

5. JavaScript, //https://developer.mozilla.org/en-US/docs/Web/JavaScript: JavaScript , дата зверн. 10.11.2020

6. Що таке блокчейн // aeneas.pm: Що таке блокчейн простими словами статті http://aeneas.pm/uk/2019/05/07/what-is-blockchain-infographics-2019/ (дата звернення 11.11.2010) - Назва з екрану

7. Top Code Editors and IDE for Python Development of 2020 [Електронний ресурс].Режим:https://www.cloudways.com/blog/top-ide-and-code-editors-python-development/ (дата звернення 26.05.2020) – The Difference Between IDE and Code Editors.

8. NetBeans IDE – Reviews, Pros & Cons [Електронний ресурс]. Режим доступу: https://stackshare.io/netbeans (дата звернення 08.11.2020) – NetBeans IDE.

9. Pros and Cons of Eclipse 2020 [Електронний ресурс]. Режим доступу: https://www.trustradius.com/products/eclipse/reviews?qs=pros-and-cons (дата звернення 06.11.2020) – Reviews.

10. Pros and Cons of Visual Studio IDE 2020 [Електронний ресурс]. Режим доступу: https://www.trustradius.com/products/visual-studio-ide/reviews?qs=pros-and-cons (дата звернення 11.11.2020) – Reviews.

11. Features PyCharm IDE [Електронний ресурс]. Режим доступу: https://www.jetbrains.com/pycharm/features/ (дата звернення 05.11.2020) – Features.

12. Що таке інформаційна система [Електронний ресурс]. Режим доступу : https://pidru4niki.com/1222090547713/informatika/informatsiyni_sistemi (дата звернення 11.11.2020) - Інформаційні системи

13. Аналіз хеш функції [Електронний ресурс]. Режим доступу : https://studfile.net/preview/5384017/page:3/ (дата звернення 05.11.2020) – Алгоритми хешування

14. Інформаційні системи та їх роль в управлінні економікою [Електронний ресурс]. Режим доступу : https://www.uzhnu.edu.ua/uk/infocentre/get/6742 (дата звернення 05.11.2020) - Поняття інформаційної системи

15. Перспективи розвитку мережі «Блокчейн» [Електронний ресурс]. Режим доступу :http://elibrary.kubg.edu.ua/id/eprint/25667/1/%D0%A1%D0%BF%D0%B0%D1%81%D1%96%D1%82%D1%94%D0%BB%D1%94%D0%B2%D0%B0.pdf (дата звернення (05.11.2020)