МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу
Кафедра інженерії програмного забезпечення

Пояснювальна записка до дипломного проекту

магістра
(освітній ступінь)

на тему «Програмне забезпечення захищеного сховища інформації на базі технології блокчейн»

ХАІ.603.667П1.121.156326.20О

Виконав: студент 6 курсу групи №667П1
Спеціальність 121 – Інженерія програмного забезпечення
(код та найменування)
Освітня програма Хмарні обчислення та Інтернет речей
(найменування)
Якушев Д.О.
(прізвище й ініціали студента)
Керівник: Туркін І. Б.
(прізвище й ініціали)
Рецензент: Ткачов В.М.
(прізвище й ініціали)

Харків – 2020

Міністерство світи і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу
(повне найменування)
Кафедра інженерії програмного забезпечення
(повне найменування)
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – інженерія програмного забезпечення
(код та найменування)
Освітня програма хмарні обчислення та Інтернет речей
(найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали та прізвище)
"_____" _____ 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

_____Якушеву Денису Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема дипломного проекту Програмне забезпечення захищеного сховища інформації на базі технології блокчейн
керівник дипломного проекту Туркін Ігор Борисович, д.т.н., проф., зав. каф. 603
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № _____ від "____" _____ 2020 року
2. Термін подання студентом проекту _____
3. Вихідні дані до проекту вимоги до програмного забезпечення захищеного сховища інформації на базі технології блокчейн.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз проблемної області та постановка завдання
2. Моделі і методи захисту даних на основі технології блокчейн
3. Розроблення прототипу програмного забезпечення захищеного сховища інформації на базі технології блокчейн
4. Перелік графічного матеріалу Пояснювальна записка – 1(усього –61 сторінка), рисунків, рисунків – 12, таблиць – 4, додатків – 1.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Туркін І. Б., д. т. н.,проф., зав. каф. 603		
2	Туркін І. Б., д. т. н.,проф., зав. каф. 603		
3	Туркін І. Б., д. т. н.,проф., зав. каф. 603		

Нормоконтроль _____ _____ «___»_____20__р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «___»_____20_____ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз проблемної області та постановка завдання	24.12.2019	
2	Моделі і методи захисту даних на основі технології блокчейн	26.05.2020	
3	Розроблення прототипу програмного забезпечення захищеного сховища інформації на базі технології блокчейн	15.11.2020	
4	Оформлення пояснювальної записки	15.11.2020–01.12.2020	
5	Предзахист дипломного проекту	30.11.2020	
6	Захист дипломного проекту	09.12.2020	

Студент _____ Якушев Д.О.
(підпис) (прізвище та ініціали)

Керівник проекту ____ Туркін І. Б.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломний проект магістра на тему «Програмне забезпечення захищеного сховища інформації на базі технології блокчейн»: 61 сторінка., 12 рисунків, 30 джерел.

Технології завдали відчутного удару по захисту нашої конфіденційності. Велика частина того, чим займаються окремі особи або організації, тепер надбання громадськості. Сторонні особи контролюють, зберігають і використовують особисті та корпоративні дані, шаблони, переваги і заняття. Багато нові бізнес-моделі ґрунтуються на зборі, організації та перепродажу наших особистих даних.

Технології також спростили пошук конкретної особи за його даними, навіть якщо цей користувач відмовився від соціальних мереж. Наприклад, зрушення в технології розпізнавання осіб знайшли широке застосування в сфері торгівлі і забезпечення безпеки.

Технологія блокчейн потенційно здатна обмежити вплив цієї ерозії конфіденційності, надаючи при цьому особисту інформацію, якщо та необхідна для справи. Наприклад, користувач може зберігати особисту інформацію в блокчейні і тимчасово оприлюднити частину її для отримання тієї чи іншої послуги. Біткойн і інші цифрові валюти, в основі яких лежить технологія блокчейн, продемонстрували, що з використанням децентралізованої точок доступу та розподіленого публічного журналу транзакцій можливо виробляти надійні і прозорі розрахунки.

БЛОКЧЕЙН, ЗАХИСТ ІНФОРМАЦІЇ, ХЕШУВАННЯ, ШИФРУВАННЯ

РЕФЕРАТ

Дипломный проект магистра на тему «Программное обеспечение защищенного хранилища данных на базе технологии блокчейн»: 61 страница, 12 рисунков, 30 источников.

Технологии нанесли ощутимый удар по защите нашей конфиденциальности. Большая часть того, чем занимаются отдельные лица или организации, теперь достояние общественности. Посторонние лица контролируют, хранят и используют личные и корпоративные данные, шаблоны, предпочтения и занятия. Многие новые бизнес-модели основываются на сборе, организации и перепродаже наших личных данных.

Технологии также упростили поиск конкретного лица по его данным, даже если этот пользователь отказался от социальных сетей. Например, подвижки в технологии распознавания лиц нашли широкое применение в сфере торговли и обеспечения безопасности.

Технология блокчейн потенциально способна ограничить воздействие этой эрозии конфиденциальности, предоставляя при этом личную информацию, если та необходима для дела. Например, пользователь может хранить личную информацию в блокчейне и временно обнародовать часть её для получения той или иной услуги. Биткойн и другие цифровые валюты, в основе которых лежит технология блокчейн, продемонстрировали, что с использованием децентрализованной одноранговой сети и распределённого публичного журнала транзакций возможно производить надёжные и прозрачные расчёты.

БЛОКЧЕЙН, ЗАЩИТА ИНФОРМАЦИИ, ХЕШИРОВАНИЕ, ШИФРОВАНИЕ

ABSTRACT

Master's thesis on "Secure information storage software based on blockchain technology ": 61 pages, 12 figures, 30 sources.

Technology has dealt a significant blow to protecting our privacy. Much of what individuals or organizations do is now in the public domain. Unauthorized persons control, store and use personal and corporate data, templates, preferences and activities. Many new business models are based on the collection, organization and resale of our personal data.

Technology has also simplified the search for a specific person by his data, even if this user has abandoned social networks. For example, advances in face recognition technology have found wide application in the field of trade and security.

Blockchain technology has the potential to limit the impact of this erosion of privacy, while providing personal information if necessary. For example, a user can store personal information on the blockchain and temporarily release part of it to receive a particular service. Bitcoin and other digital currencies based on blockchain technology have demonstrated that using a decentralized peer-to-peer network and a distributed public transaction log, it is possible to make reliable and transparent calculations.

BLOCKCHAIN, INFORMATION PROTECTION, HASHING, ENCRYPTION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

DAO – Decentralized Autonomous Organization
DHT – Distributed Hash Table
ECDSA – Elliptic Curve Digital Signature Algorithm
FUSE - filesystem in userspace
IBM – International Business Machines
IPFS – InterPlanetary File System
MuSig – протокол для агрегації відкритих ключів та підписів для алгоритму цифрового підпису Шнорра
NAT – Network Address Translation
Peer-To-Peer – однорангова мережа
PoS – принцип доказу зберігання коштів користувача за певний термін
RSA – криптографічний алгоритм з відкритим ключем, який базується на обчислювальної складності задачі факторизації великих цілих чисел
SPV – Simplified Payment Verification
TCP – Transmission Control Protocol
UDP – User Datagram Protocol
UTXO – Unspent Transaction Output
ПЗ – програмне забезпечення
ПК – персональний комп'ютер

ВСТУП

Інформація постійно приносила перевагу в боротьбі за благополуччя і влада, проте в сучасних умовах, в інформаційний століття, вона стала основною зброєю. З розвитком інформаційних технологій і загальнодоступних засобів масових комунікацій збільшилася ймовірність зловживань, пов'язаних із застосуванням зібраних і накопичених даних про людину. Виникли і результативно застосовуються правопорушниками кошти швидкої обробки персональних даних, формують небезпеку правам і законним інтересам людини.

В умовах глобалізації діяльність людини все більше і більше знаходить взаємозв'язок з всесвітньою мережею Інтернет, за минулі десятиліття кількість її користувачів збільшилася неодноразово.

Високі технології розвиваються з величезною швидкістю, створюючи інструменти вирішення повсякденних завдань. В даний час людство не уявляє своє життя без інтернету, більшість угод проходять за допомогою інтернету.

Працюючи в мережі покупці отримують безліч плюсів, за рахунок мобільності, зручностей, швидкості здійснення різних операцій купівлі-продажу, але персональні відомості потрапляють під величезну небезпеку.

Проблема збереження персональної інформації є актуальним, важливою вважається охорона персональних даних, що виявляються в мережі інтернет і їх інформаційна захищеність.

Безсумнівно, широко застосовуючи ПК і мережі з метою обробки та передачі даних, дані сфери повинні бути грунтовно захищеними від можливості доступу до неї сторонніх осіб. Її втрати або спотворення. Відповідно до статистичних відомостей, більше вісімдесяти відсотків фірм несуть фінансові збитки через порушення цілісності і конфіденційності використовуваних даних.

Більш популярним джерелом загроз персональної інформації вважається Інтернет. У сучасному суспільстві майже будь-яка людина має електронну пошту, в деяких випадках акаунтів кілька (Індивідуальний і робочий електронну поштову скриньку), і профілі в різноманітних соціальних мережах, до того ж і в професійних соціальних мережах. В будь-якому випадку злом профілю здатний спричинити втрату персональної інформації розміщених або на сторінці аккаунта, або колись пересилаються за допомогою сервісу, але ж нерідко за допомогою пошти та соціальних мереж відправляються в тому числі паспортні дані та інші істотні дані. Всілякі незаконні дії, які призвели до втрати індивідуальних відомостей, не дотримуються основний закон країни - Конституцію. Окрема проблема охорони персональних відомостей в мережі Інтернет з'являється в тому випадку, якщо зосередити увагу на електронну торгівлю, так як онлайн придбання стали багатьох людей природним феноменом.

При здійсненні цих дій, потрібно найбільш ретельно досліджувати веб-сайт, на якому купується продукт, на предмет суворого відповідності нормам закону і не слід прив'язувати власну банківську карту до платіжної системи сайту, дана операція несе за собою додатковий ризик. Ще одним джерелом загрози для індивідуальної інформації в мережі Інтернет можуть бути сайти з

пошуку роботи та портали персоніфікованих (тобто спеціалізованих для певного громадянина і включають його персональні відомості) послуг населенню.

Цифрові технології дають можливість вивчити мільярди особистих взаємодій, в процесі яких, суспільство обмінюються думками, засобами, продуктами і плітками. У новий цифровий період буде потрібно регулювати людство по-новому. знадобиться починати тестування зв'язків в цьому світі значно раніше і набагато більше, ніж раніше. Необхідно формувати так звані «живі лабораторії», де можливо відпрацьовувати ідеї з метою зведення суспільства, яке підлягає контролю індивідуальними відомостями. Збільшувати творчий потік ідей можна було б усім людям, анонімно, з відсутністю побоювання ділитися персональною інформацією. У постіндустріальному суспільстві конфіденційність початку означати, що деякі дані, легкодоступні для одних, недосяжні для інших. Конфіденційність має на увазі потребу запобігання розголошенню індивідуальних відомостей, а приватність - це арбітр, який вирішує, якому об'єкту бути більш контрольованим. Вельми важливо зберігати непросте рівновагу серед конфіденційності та відкритості, про даному мають можливість подбати самі користувачі за допомогою установки браузера або соціальних мереж. Здібності обміну даними в інтернеті на сьогоднішній день досить безмежні і продовжують вдосконалюватися. Інтернет в даний період вважається активної суспільним середовищем, яка пов'язує колосальне число людей.

Мета досліджень – розробка програмного забезпечення сховища інформації на базі технології блокчейн.

Об'єкт дослідження – захищеність технології блокчейн та хмарних сховищ.

Предмет дослідження – методи та способи захисту даних в технологіях блокчейн та хмарних сховищ.

Наукова новизна отриманих результатів полягає в оцінені можливостей створення унікального ПЗ, яке буде позволяти залишатися захищеним у системі обміну файлами.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд технології блокчейн

Блокчейн (від англ. Block chain) - це ланцюжок блоків або якщо бути точніше розподілена база даних. Вперше цей термін був застосований як назва для розподіленої бази даних криптовалюти біткоін. Однак дану технологію можна використовувати не тільки для криптовалюти[1].

Технологія зберігання інформації блокчейн дозволяє вирішити існуючу проблему, пов'язану з відсутністю гарантій з боку посередників, які виступають в якості третіх осіб при здійсненні тих чи інших дій. Наприклад, немає ніяких гарантій, що банк, який виступає гарантом вашого платежу, не збанкрутує.

Блокчейн - це база даних, у якій немає керуючого, самі учасники є керуючими. Так звані Майнери. Вона заснована на тимчасовій (p2p) мережі, загальному реєстрі і криптографії публічного і приватного ключа. Увійшовши в блокчейн-мережу, користувач підключається до інших комп'ютерів мережі для того, щоб обмінюватися з ними даними: блоками і записами. Отримавши нові дані, кожен користувач перевіряє їх коректність, і, переконавшись у достовірності, зберігає їх у себе, а також передає коректні дані далі по мережі.

Учасники мережі діляться на дві групи: звичайні користувачі, які створюють нові записи, і Майнери, які створюють блоки. Звичайні користувачі створюють і поширюють через мережу записи, наприклад, про грошові перекази або про передачу прав власності. Майнери збирають записи, перевіряють їх і записують в блоки, а потім розсилають ці блоки по мережі. Після чого звичайні користувачі отримують блоки і зберігають їх у себе, щоб можна було коректно створювати свої і достовірно перевіряти чужі нові записи. Діяльність Майнерів називається Майнінг. Так само існують інші способи перевірки достовірності даних, наприклад PoS[2].

Як випливає з назви технології, в її основі лежить ланцюжок послідовно пов'язаних блоків. Нові блоки завжди додаються строго в кінець ланцюжка. Блок складається з заголовка і тіла, що містить записи. Блоки пов'язані з допомогою ключів, оскільки в заголовку кожного блоку зберігається ключ попереднього блоку. Це забезпечує захищеність мережі.

Ключ кожного блоку розрахований на дані всього блоку і ключ попереднього блоку. А це значить, що в ключі будь-якого блоку закодовані не тільки записи цього блоку, але і всі попередні блоки. При цьому ключ блоку повинен задовольняти правилам безпеки, встановлює рівень захищеності мережі. Наприклад, в біткоіни ключі перших блоків починалися з десяти нулів, що встановлювало ступінь складності створення нового блоку.

Майнер - це користувач блокчейн-мережі, який крім перевірки і поширення даних займається і створенням нових блоків. Це також може бути спеціалізована програмна платформа. Отримавши нові записи від інших учасників мережі, майнер збирає їх разом, формує заголовок майбутнього блоку і розраховує ключ блоку. Щоб знайти підходяще значення ключа, Майнеру доводиться робити

величезну кількість перерахунків. Коли відповідний ключ знайдений, Майнер зберігає блок і відправляє його іншим учасникам мережі. Тепер всі записи в блоці підтверджені і захищені кодом, який вельми нелегко підробити. Причому, в ключі блоку закодований і ключ попереднього блоку, який тепер підробити не можна. Така витончена процедура розрахунку ключів ускладнює створення блоку, але ще більше вона ускладнює створення підроблених блоків, роблячи це майже неможливим.

Записи в тілі блоку також захищені шляхом зв'язування в ланцюжок. Кожен запис містить посилання на попередній запис-джерело, а також блокуючу умова, щоб розблокувати правило. Для опису правил і умов використовується мова програмування, який дозволяє задавати складну логіку і правила взаємодії учасників. Джерел і результатів в кожному записі може бути кілька, тобто запис може перетворити кілька записів-джерел в кілька записів-результатів. Таким чином, блокчейн призводить нас до «розумним» контрактами, що дозволяє формалізувати відносини не тільки між людьми, але і між роботами і програмами, що створює передумови для використання технології в Інтернеті речей. Наприклад, в концепції «розумного» будинку, який контролює витрату електрики, газу, води, кількості продуктів в холодильнику, автоматично укладає контракти на поставки всього необхідного і оплачує їх. Блокчейн в своїй основі відкрита і публічна, і переглянути її вміст можна без проблем. Для цього є програми-аналізатори та онлайн-сервіси.

Основою технології блокчейн є електронний реєстр всіх фінансових транзакцій. Як правило, такі електронні «бухгалтерські книги» є однією з головних точок уразливості. Якщо зловмисники отримають доступ до головного реєстру записів, то це може привести до цілковитого «падіння» системи, адже, отримавши доступ до записів, зловмисник може вкрасти необмежену суму грошей або опанувати будь-якої особистої інформації, просто переглянувши список транзакцій.

У блокчейні реєстр записів є децентралізованим - це означає, що одиночний комп'ютер або система не можуть отримати контроль над всією «бухгалтерської книгою». А значить, для того, щоб отримати доступ до головного реєстру записів, знадобилося б організувати неймовірно складну і чітко скоординовану операцію, згідно з якою в один момент тисячі пристроїв були б атаковані одночасно.

Іншим принципом, що забезпечує виняткову безпеку, є сама ланцюг транзакцій. Головний реєстр записів є довгий ланцюжок послідовних блоків. Кожен блок, що входить до даний ланцюг, є лише частиною загальної структури, яка бере свій початок від найпершої виробленої в даній системі операції.

Це означає, що будь-кому, хто вирішить змінити інформацію про однієї транзакції, перед цим доведеться вкрай акуратно і точно змінити всі записи, що ведуть до цієї транзакції. Виходячи з чого, передбачуване втручання виглядає вкрай складним процесом, що також є одним з плюсів в побудові безпеки блокчейна.

Крім того, є також і інші елементи, що забезпечують захист блокчейна.

Більше двох користувачів підтверджують і забезпечують безпеку

проведеної транзакції. Навіть в більшості сучасних процесингових систем в перевірці задіяні тільки кілька рівнів верифікації - як правило, це продавець, покупець і якісь треті особи (найчастіше банк або кредитна агенція).

Однак в системі блокчейна існує від декількох сотень до декількох тисяч різних вузлів, на кожному з яких зберігається повна копія реєстру записів. Тому будь-який з цих вузлів також може брати участь в перевірці транзакції, і якщо вузол з якихось причин не приймає транзакцію, то вона буде скасована. Подібний розклад майже до мінімуму знижує можливість створення помилкової або шахрайської транзакції.

Криптографічні ключі, які використовуються системою в обмінних процесах, також є дивом сучасної кібербезпеки. Кожен зашифрований ключ являє собою довгу, складну послідовність даних, практично не піддається розшифровці. А якщо врахувати, що для підтвердження потрібні два таких унікальних ключа, то система починає виглядати мало не неприступною фортецею. При цьому вважається, що блокчейн володіє унікальною системою безпеки, тому що при подібному рівні захисту в ньому вдається зберегти майже повну прозорість транзакцій.

Але, як вже говорилося, навіть блокчейн не ідеальний. У нього, як і у будь-який інший системи, є слабкі місця. Так що, якщо ви плануєте використовувати криптовалюта і вкладати в неї свої кошти, або якщо вам в майбутньому доведеться мати справу з блокчейном, то просто необхідно знати і розуміти потенційно вразливі місця технології. Тому постарайтеся запам'ятати такі особливості, що стосуються безпеки даної технології:

Якщо ви вирішите створити систему на основі технології блокчейн з нуля, то одна невелика помилка може стати фатальною і «покласти» всю вашу розробку. Звичайно ж, це не можна вважати недоліком самого блокчейна - це, скоріше, стосується особливостей його використання. Крім того, середньостатистичній людині набагато важче розібратися в блокчейне через його складності, що, в свою чергу, означає, що багато хто не до кінця розуміють ризики, пов'язані з використання системи, а також в повному обсязі використовують доступний функціонал.

Для роботи блокчейна необхідні як мінімум кілька сотень, а ще краще кілька тисяч узгоджено працюють вузлів. Саме через це система є вкрай вразливою до атак на початкових етапах роботи. Наприклад, якщо який-небудь користувач зможе отримати контроль над 51% вузлів системи, то він зможе повністю контролювати результат роботи. А якщо в системі всього 20 вузлів, то подібний варіант розвитку подій більш ніж можливий.

Структура блокчейна - це також одна з причин, по якій може бути порушено нормальне функціонування системи. Так, якщо система отримає надто широке поширення, а інфраструктура блокчейна виявиться не готовою до такого обсягу операцій, то в результаті може знизитися швидкість проведення транзакцій, можуть з'явитися проблеми зі зберіганням даних, а це все не кращим чином вплине на ефективність мережі.

Хоч і не можна сказати, що даний пункт безпосередньо пов'язаний з безпекою блокчейна, але політика системи може вплинути на її застосування і

подальший розвиток. З огляду на те, що валюта в системі блокчейн є міжнародною і децентралізованої, це, по суті, знецінює національну валюту, контрольовану державою. І, природно, на даний момент керівні органи деяких держав прагнуть ввести більш суворі обмеження на використання блокчейна. Уряди різних країн сподіваються взяти систему під контроль до того, як вона стане серйозним конкурентом і почне загрожувати їхній економіці. Непрямим чином це також є суттєвою загрозою безпеки блокчейна, яка може значно уповільнити поширення технології.

Наприклад, NiceHash - сторонній ринок для Майнінг біткоіни - був не так давно зламаний, в результаті чого було вкрадено криптовалюта на більш ніж 60 мільйонів доларів. Як виявилося, дана платформа була небезпечною. Тобто, це не помилка безпеки самої системи блокчейн. Швидше, навпаки, кіберзлочинці отримали доступ до системи NiceHash за допомогою блокчейна[3].

При транзакціях в системі blockchain використовуються публічні і приватні криптографічні ключі. Самі по собі такі ключі зламати майже неможливо, проте кіберзлочинець може отримати їх більш простим і звичним способом. Наприклад, ключі можна дістати в тому випадку, якщо ви зберігаєте їх на небезпечній або слабкозахищеній платформі. Так, якщо хтось зламає ваш поштовий ящик, то він зможе отримати доступ до всіх ваших листів, а значить, і до ключів вашого профілю в блокчейне. В такому випадку зловмисник зможе заволодіти вашими коштами, видавши себе за вас. І це одне з головних питань, що стосуються безпеки системи.

Користувачі системи також можуть попастися на інші, більш традиційні прийоми шахраїв. По суті, такі шахрайські схеми не вважаються слабким місцем в системі безпеки блокчейна. Так, наприклад, ви можете отримати електронного листа, в якому незнайомий вам людина буде переконувати вас, що саме ви стали тим щасливчиком, який виграв щось значне. Або ж шахраї можуть запропонувати вам витратити вашу криптовалюта на якийсь товар або послугу, яку ви ніколи не отримаєте.

1.2 Безпека інформаційної технології блокчейн та її похідних

На даний момент головною загрозою для блокчейна, щодо гіпотетичної, є «атака 51%», коли зловмисник може зробити відкат транзакцій, друкуючи альтернативні блоки на бічній ланцюжку (гілці) і гарантовано спростовуючи те, що відбувається в основний ланцюжку блокчейна. По суті, це схоже на «човниковий біг». Однак з огляду на ресурсомісткість рішення хеш-функції і емісії нових біткоіни, поки що такий варіант здається малоймовірним. Змова власників найбільших майнінгових пулів теж виглядає непереконливим (якщо не враховувати статистику найбільших виробників біткоіни). Але подібні приклади вже були: один з пулів - ghash.io - набрав потужність, близьку до 50%, після чого власники припинили прийом нових користувачів, щоб не створювати компрометуючу ситуацію[4].

Розглянемо сценарій, коли атакуючий намагається генерувати альтернативну ланцюжок швидше ніж чесні вузли. Навіть якщо це вдасться, він

не зможе робити в системі будь-які зміни, наприклад створювати монети з повітря або брати монети, які йому ніхто не перекладав. Вузли не приймуть неправильну транзакцію в якості платежу, а чесні вузли ніколи не приймуть блок з неправильною транзакцією. Атакуючий може лише змінити одну зі своїх власних транзакцій повернувши собі платіж, який він нещодавно здійснив. Гонка між чесної ланцюжком і ланцюжком атакуючого може бути описана в термінах біноміального випадкового блукання. Вдале подія це зростання чесної ланцюжка на один блок з наближенням до мети на +1, невдале подія це зростання на один блок ланцюжка атакуючого зі зменшенням відриву на -1. Можливості атакуючого в гонці в умовах обмежень, схоже на опис проблеми Gambler's Ruin (розорення картяра). І так, картяр з необмеженим кредитом починає гру в умовах обмежень і може потенційно провести необмежене число партій щоб спробувати досягти беззбитковості. Ми можемо порахувати вірогідність досягнення нею беззбитковості або те ж саме, що атакуючий обжене чесних будівельників ланцюжка (формула 1.1).

Нехай:

p - ймовірність що чесний хост знайде наступний блок;

q - ймовірність, що атакуючий знайде наступний блок;

q_z - ймовірність, що атакуючий виграє гонку якщо він відстав на z блоків.

$$q_z = \begin{cases} 1, & p \leq q \\ (q/p)^z, & p > q \end{cases} \qquad (1.1)$$

Припустимо, що p> q, тоді ймовірність експоненціально зменшується при збільшенні числа блоків, на яке відстав атакуючий. Таким чином, якщо йому не вдасться вирватися вперед на самому початку, то його шанси перемогти в подальшому стають зникаюче малі. Розглянемо тепер, як довго одержувач платежу повинен чекати, щоб бути впевненим що відправник не зможе змінити транзакцію. Припустимо, що відправник - це атакуючий, який хоче щоб одержувач повірив що платіж проведено, але через деякий час повернути платіж собі назад. Одержувач буде сповіщений коли таке трапиться, але відправник сподівається, що буде вже пізно. Одержувач генерує нову пару ключів і віддає публічний ключ відправника незабаром після свого підпису. Це не дає можливості відправнику підготувати ланцюжок блоків заздалегідь працюючи над нею з випередженням для виконання транзакції в даний момент. Тільки коли транзакція послана, нечесний відправник може почати в секреті працювати над паралельною ланцюжком, що містить альтернативну версію цієї транзакції. Одержувач чекає, поки транзакція буде додана в блок і Z блоків буде додано після цього. Він не знає на якому етапі будівництва знаходиться атакуючий, але вважаючи що чесні блоки будувалися з однаковим середнім часом на один блок, очікуване значення приросту атакуючого може бути знайдено через розподіл Пуассона (формула 1.2).

$$\lambda = z\frac{q}{p} \tag{1.2}$$

Для отримання ймовірності, з якою атакуючий все ще може вийти вперед (формула 1.3), помножимо щільність пуассоновського розподілу кожного значення прогресу атакуючого на ймовірність того, що він вийде вперед з даної точки (формула 1.4).

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} * \begin{cases} (q/p)^{(z-k)} & k \le z \\ 1 & k > z \end{cases} \tag{1.3}$$

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} * (1 - (q/p)^{(z-k)}) \tag{1.4}$$

Чисельно проаналізувавши отримане вираз, можна легко переконатися, що ймовірність експоненціально зменшується зі зростанням z.

У різних продуктах технології блокчейн використовуються різні методи підтвердження блоків. Так, в біткоіни застосовується метод proof-of-work - Підтвердження блоків обчислювальною потужністю. Інший варіант закриття блоків - proof-of-stake, коли блоки друкувати не обчислювальною потужністю, а за допомогою грошей, що перебувають у людей на руках. У цьому випадку, щоб провести «атаку 51%», необхідно мати 51% монет, що обертаються в системі. Як і у випадку з «човниковим бігом», якщо зловмисник володіє більш ніж 51% монет, він також зможе створити альтернативну ланцюжок, яка перетвориться в основну. Ця ситуація нагадує голосування на зборах акціонерів, коли у одного з власників є на руках контрольний пакет, що блокує голоси інших власників[5].

Якщо безпеку блокчейна викликає мінімум побоювань, то збереження біткоіни, навпаки, породжує багато запитань, адже, як і звичайні паперові гроші, криптовалюта теж можна викрасти. Ключ записи в блокчейне - це хеш-функція від публічного ключа. Невпевнено або недбале зберігання закритого ключа може призвести до крадіжки або втрати біткоіни. За даними Harvard Business Review, вартість втрачених біткоіни вже становить близько $ 950 млн. Найпростіший спосіб убезпечити себе - створити пароль гаманця. Але якщо хакер викраде і ваш гаманець, і ваш пароль, відновити викрадені біткоіни буде практично неможливо, оскільки транзакції, подані з вкраденими ключами, здаються перевіряючим вузлів відрізнятись від законних транзакцій. Деякі скептики стверджують, що хакери зможуть зламати ключ, використовуючи сервіси, що обчислюють паролі по хешу. Однак, з огляду на нинішні обчислювальні потужності, це виглядає малоймовірним. Але якщо раптом з'явиться алгоритм, що дозволяє ефективно факторизовать еліптичні криві, то виникне ймовірність того, що до адрес-гаманцях, з яких витрачалися гроші, легко можна буде підібрати закриті ключі.

Чи не менше запитань викликає і надійність бірж, що зберігають

криптовалюта. У серпні 2016 року зі гонконгської біржі Bitfinex - однієї з чотирьох найбільших в світі майданчиків для кріптовалютних торгів - викрали 119 756 біткоіни (близько $ 65 млн). За Bitfinex закріпилася репутація одного з найбільш надійних і безпечних організацій: більшість призначених для користувача засобів зберігалися в гаманцях з мультіподпісью і в «холодних сховищах». Незважаючи на це, зловмисникам вдалося обійти захист Bitgo, в тому числі двухфакторну аутентифікацію і механізм мультіподпісі, і зробити масову крадіжку з індивідуальних гаманців користувачів. Деталі злому так і не були донесені до широкого загалу, проте в ЗМІ тиражувалося думку про можливу причетність співробітників Bitfinex до злому, що знову ставить питання про людський фактор.

За даними blockchain.info, Майнер, видобувні біткоіни, виробляють 450 тисяч трлн обчислень в секунду. Кожна спроба вимагає енергії. Щорічно на виробництво біткоіни йде від 2 терават-годин (це більше, ніж споживає 150-тисячне місто в Каліфорнії) до 40 терават-годин (це дві третини від споживання 10-мільйонного округу Лос-Анжелес).

У США кожна транзакція біткойнов коштує близько 6 доларів, і в результаті більшість Майнерів відкривають «ферми» в країнах з дешевою електроенергією. За даними bitcoinity.org, в лютому 2016 року ТОП-виробників біткоіни виглядав наступним чином:

1) Antpool - 25,90%;
2) F2pool - 22,60%;
3) Bitfury - 15,09%;
4) BtcChina - 12,78%;
5) bw.com - 6,34%;
6) інші - 17,29%.

Чотири з п'яти перерахованих Майнерів, за винятком Bitfury, мають китайську «прописку». Це породжує ризик «картельних змов» та «загрози 51%», а також ставить питання про мотивацію компаній, в дійсності контролюючих систему, - їх інтереси можуть відрізнятися від інтересів людей, які тримають активи в біткоіни. Нарешті, ніхто не відміняв політичні ризики і вплив китайських регуляторів на інтернет-галузь.

У 2017 р криптовалюта Ethereum, побудована на мові Тьюринга Віталіком Бутеріном, постраждала від «контрагента» - Decentralized Autonomous Organization (децентралізованої автономної організації). DAO - цифрова компанія або віртуальний хедж-фонд, частку в якому можна отримати шляхом вкладення особистих коштів і покупки на них DAO-токенів - стала жертвою власних недоробок. У коді DAO виявили уразливості, які дозволили викрасти токени. Ethereum довелося здійснити хардфорк і провести зворотний повернення токенов. Хардфорк передбачає поділ і поява іншої мережі, яка містить в собі всю інформацію і ланцюжок блоків блокчейн первинної мережі. Після поділу нова мережа починає жити своїм життя і створювати власну історію транзакцій. Зазвичай хардфорк роблять в тому випадку, коли в співтоваристві з'являються розробники, яких не влаштовують поточні можливості мережі, вони хочуть відокремитися і поліпшити мережу на свій розсуд[6].

Проблем з безпекою у Ethereum не було проте проект зазнав репутаційні втрати через недоліки DAO. Одночасно з блокчейном розвиваються і інші технології, і ступінь їх впливу і небезпеки поки не до кінця очевидна. Так, дослідники з Університету Ньюкасла представили механізм управління ботнетом для відправки повідомлень ботам в мережі біткоіни. Ботнет (англ. Botnet, походить від слів robot і network) - комп'ютерна мережа, що складається з певної кількості хостів, з запущеними ботами - автономним програмним забезпеченням. приховано встановлюються на пристрій жертви і дозволяє зловмиснику виконувати якісь дії з використанням ресурсів зараженого комп'ютера. Не виключено, що мало-помалу боти відсунутий людей від Майнінг біткоіни, як це сталося у випадку з ботнетом інтернету речей Mirai. У квітні 2017 року фахівці IBM виявили, що Mirai активно встановлює код Майнінг біткоіни на комп'ютери деяких зі своїх жертв. Так що «ботізація» Майнінг - поки не до кінця очевидна, але цілком відчутна перспектива.

Залишаться і звичайні проблеми інформаційної безпеки. Ніхто нічого не зможе зробити, якщо з вашого комп'ютера вкрадуть не тільки гаманець, але і пароль до нього. І тут блокчейн не панацея, а додаткова вразливість. Адже випадок з DAO все ж винятковий, і заради однієї людини ніхто відкочувати блоки не буде. Хоча розслідувати крадіжку грошей, можливо, буде простіше. У відкритій системі відразу буде видно, коли гроші зняли і куди перевели. Але ось розплутувати подальшу ланцюжок вже складно. Правоохоронні органи, як правило, не справляються і зі звичайною крадіжкою грошей через Інтернет, а про те, щоб розслідувати факт крадіжки в блокчейн-ланцюжку, і говорити нема чого. Їм просто не вистачить компетенції.

Коли говорять про «успішні атаки на біткоіни», найчастіше мають на увазі успішні атаки на біржу, яка торгує біткоіни, що, звичайно, в принципі невірно. Адже сама ланцюжок блоків при цьому не компрометується. Але постраждалим від цього не легше. Останній гучний випадок стався в серпні 2016 року. З гонконгської кріптовалютной біржі Bitfinex викрали 119 756 біткоіни (близько 65 млн дол.). Bitfinex є однією з чотирьох найбільших майданчиків для кріптовалютних торгів і належить фінансово-технологічної компанії iFinex. Остання була заснована в 2012 році зі штаб-квартирою в Гонконгу. За п'ять років існування біржа тільки один раз піддалася злому «гарячого гаманця» - в травні 2015 го, проте втрати були незначні.

До сих пір Bitfinex зберігала репутацію однієї з найбільш надійних і безпечних кріптовалютних бірж. Більшість призначених для користувача засобів зберігалися в гаманцях з мультіподпісью і в «холодних сховищах». Це особливо підігріло інтерес до проведеної крадіжці, адже «холодне зберігання» вважалося найнадійнішим способом криптовалюта і використовувалося для заощадження.

1.3 Хмарне сховище даних

День у день виникає гостра необхідність в надійному зберіганні і швидкому зручному обміні великими обсягами найрізноманітніших даних між користувачами мережі, що знаходяться в самих різних куточках нашої планети.

Одним з найбільш оптимальних, ефективних, зручних і тому затребуваних рішень стали так звані хмарні сервіси для зберігання даних, що дозволяють зберігати, оперативно редагувати і швидко передавати значні обсяги найрізноманітніших файлів. Причому доступ до тих або інших даних можуть одночасно мати багато користувачів, або ж інформація може носити конфіденційних характер і бути призначеною тільки для особистого користування[7].

У загальних рисах хмарне сховище даних являє собою віртуальну мега-флешку, віртуальний файлообмінник або сервер з даними, доступ до яких можна отримати з будь-якого пристрою, підключеного до мережі і використовуваному хмарного сервісу.

Однак на відміну від традиційного сервера, що завантажується в хмарні сховища інформація розміщена одночасно на безлічі мережевих серверів, в тому числі, що знаходяться за тисячі кілометрів на іншому кінці Землі.

Маючи рахунок у сервісі, що надає «хмара» певного, як правило, відносно невеликого обсягу, через сайт або встановлюється на комп'ютер або інший пристрій (планшет, смартфон) клієнт-програму, можна автоматично копіювати інформацію з жорсткого диска в «хмару».

Причому, синхронізувавши пристрій і віртуальне сховище, можна автоматично мати під рукою найбільш актуальну інформацію. Це означає, що редаговані файли на комп'ютері або прямо в «хмарі» будуть автоматично оновлюватися або переноситься у відредагованому варіанті туди, де інформація не зазнала зміни. Це зручно і дозволяє істотно знизити ризик випадкової втрати цінних даних.

Безкоштовно хмарні сервіси надають небагато місця, проте платний аккаунт істотно розширює обсяг і функціонал «хмари». Також стежачи за програмою лояльності і проведеними хмарними сервісами акціями, можна абсолютно безкоштовно розширити обсяг свого віртуального сховища до значних розмірів.

1.3.1 Переваги хмарного сховища даних

Основні переваги використання хмарних сховищ даних, на відміну від сумного жорсткого диска, свого комп'ютера або навіть традиційного сервера очевидні.

«Хмари» дозволяють заливати, зберігати і обмінюватися файлами в великих обсягах (понад 20 МБ). Передати важкий файл по електронній пошті складно або зовсім неможливо. На флешці - довго, незручно, а часто не представляється можливим на увазі фізично велику відстань між відправником і отримувачем. Рішення - завантажити інформацію в хмарне сховище, створивши там папку, після чого поділиться з адресатами посиланням поштою.

Доступ до збережених в «хмарі» файлів можна отримати де і коли завгодно з будь-якого пристрою, головне мати вихід у всесвітню павутину. Це особливо зручно, наприклад, при необхідності звернення до будь-яким загальним робочим документам, які, при відповідних настройках, що проводяться власником облікового запису, можна редагувати в режимі онлайн.

Спеціальні алгоритми резервування даних, що зберігаються, що застосовуються на хмарних сервісах, практично повністю виключають ймовірність втрати важливих даних через прикрі збої або пошкоджень і виходу з ладу жорсткого диска або проблем з традиційним сервером.

І це тільки лише можливості безкоштовного аккаунта. Користування хмарним сервісом на платній основі розширює список переваг.

1.3.2 Недоліки хмарного сховища даних

Однак в роботі з «хмарами» справедливості заради можна відзначити і кілька основних мінусів.

Перший і головний - це недостатнє опрацювання питання забезпечення безпеки. Незважаючи на пропаговану політику конфіденційності, до інформації, що завантажується залишається відкритим доступ співробітників сервісу і його програмного забезпечення.

Другий недолік випливає частково з першого - при зломі сховища або сервісу, наприклад, в результаті хакерської атаки, конфіденційні файли можуть потрапити під загальний доступ або в руки до зловмисників.

Третій недолік пов'язаний з необхідністю очікування повної синхронізації пристрою і хмарного сховища, якщо така опція використовується. У свою чергу, тривалість очікування при зверненні до сервісу залежить від швидкості доступу в мережу. Переривати же процес синхронізації не рекомендується через високу ймовірність виникнення помилок і збоїв.

1.4 Аналіз сучасних систем обміну файлів заснованих на технології блокчейн

1.4.1 Огляд системи “Decentralized Cloud Storage — Storj.io”

Storj - це платформа для хмарного зберігання даних, яку не можна піддавати цензурі, контролювати або мати простої. Це децентралізована, повністю зашифрована платформа хмарного зберігання, що використовує криптографію для захисту ваших файлів.

Storj - це платформа, кріптокоррекція і набір децентралізованих додатків, які дозволяють вам зберігати дані в безпечному і децентралізованому вигляді. Ваші файли зашифровані, розбиті на шматки і зберігаються в децентралізованої мережі комп'ютерів і серверів по всьому світу. Ніхто, крім Вас, не має доступу до Ваших файлів, навіть в зашифрованому вигляді.

Завдяки цьому Storj також швидше, дешевше і безпечніше, ніж традиційні платформи хмарного зберігання даних.

Швидше – кілька машин одночасно обслуговують частини ваших файлів.

Дешевше – ви здаєте в оренду людям вільне місце на жорсткому диску, замість того, щоб платити за спеціально побудований центр обробки даних.

Безпечніше – ваші файли зашифровані і подрібнені. Немає необхідності довіряти свої файли корпорації, вразливим серверів або співробітникам. Storj повністю виключає довіру з рівняння.

Storj використовує шифрування з відкритим/закритим ключем і криптографічні хеш-функції для забезпечення безпеки. Для кращого захисту ваших даних, файли шифруються на клієнтській стороні на вашому обчислювальному пристрої перед їх завантаженням. Кожен файл розбивається на шматки, які спочатку шифруються, а потім поширюються по мережі Storj. Мережа складається з вузлів Storj, керованих користувачами по всьому світу, які орендують невикористане місце на жорсткому диску в обмін на STORJ Token (STORJ)[8].

1.4.2 Огляд системи “Blockchain-based file sharing network MediaCoin”

Mediacoin - це конфіденційна децентралізована мережа обміну файлами, вільна від реклами і цензури, зі своєю власною кріптоекономікой. Мережа налічує понад 25 000 користувачів і 450 000 файлів, працює більше року і має англійську і російську версії. Монета MDC продається на крипто-валютній біржі Livecoin.

Mediacoin (MDC) - це цифрова валюта, в якій учасники мережі нагороджують один одного за наданий контент. Користувачі скачують між собою файли в обмін на віртуальні монети. Мережа дозволяє заробляти монети, поширюючи файли, розміщуючи унікальний контент, а також залучаючи рефералів[9].

1.4.3 Огляд протоколу зв'язку “IPFS”

IPFS - контентно-адресуємий, спеціальний робочий гіпермедійний протокол зв'язку. Вузли IPFS-мережі формують розподілену файлову систему. IPFS є проектом з відкритим вихідним кодом, розробленим Protocol Labs за сприяння open-source спільноти.

IPFS є однорангову розподілену файлову систему, яка з'єднує всі обчислювальні пристрої єдиною системою файлів. У певному сенсі IPFS схожа зі всесвітньою павутиною. IPFS можна уявити як єдиний BitTorrent-рій, обмінюються файлами єдиного Git-сховища. Іншими словами, IPFS забезпечує контентно-адресуемую модель блочного сховища з контентно-адресованими гіперпосиланнями і високу пропускну здатність.

Це формує узагальнений деревовидний спрямований граф.

IPFS поєднує в собі розподілену хеш-таблицю, децентралізований обмін блоками, а також самосертіфіцірующееся простір імен. При цьому IPFS не має точок відмови, і вузли не зобов'язані довіряти один одному.

Доступ до файлової системи може бути отриманий різними способами:

- через FUSE
- поверх HTTP.

Локальний файл може бути доданий в файлову систему IPFS, що робить його доступним всьому світу. Файли ідентифікуються по їх мультіхешам, що спрощує кешування. Вони поширюються через протокол, заснований на протоколі BitTorrent. Переглядають контент, допомагають в доставці контента для інших користувачів мережі. IPFS має сервіс імен під назвою IPNS, глобальний простір імен на основі відкритих ключів, сумісний з іншими просторами імен і має можливість інтегрувати DNS, .onion, .bit і інші в IPNS[10].

1.4.4 Огляд системи “Axel.org”

AXEL - безпечна, зашифрована платформа для зберігання та обміну файлами, яку може використовувати будь-який бажаючий.

AXEL - це хмарне сховище і додаток нового покоління для обміну файлами, що поєднує в собі простий, інтуїтивно зрозумілий інтерфейс і потужні можливості.

Захистіть свої файли за допомогою нашого протоколу передачі IPFS і надійного шифрування паролів. На відміну від стандартного HTTP, кожен раз, коли ви відправляєте документ одного або колезі через нашу IPFS-підтримувану мережу, він ділиться на безліч невеликих шматочків, перш ніж відправити його за призначенням. Це робить його набагато більш безпечним під час транспортування, так як хакери не можуть перехопити весь файл. Крім того, завжди є можливість включити AES 256-бітове шифрування пароля. Цей практично не зламуємий алгоритм пропонує сильний 2-й рівень захисту.

Контролюйте свої дані за допомогою параметрів, що настроюються безпеки. Виберіть, коли закінчиться термін придатності вашого загального контенту і чи зможе одержувач завантажити файли[11].

1.5 Висновки з розділу 1

В цьому розділі був зроблен аналіз проблемної області. Було переглянуто основні можливості блокчейну для захисту файлів, та користувача у системі в цілому. Переглянуто можливість використання хмарного сховища для таких цілей. Також були переглянуті аналоги створюваного програмного забезпечення.

2 МОДЕЛІ І МЕТОДИ ЗАХИСТУ ДАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

2.1 Блокчейн

2.1.1 Peer-To-Peer мережа

Блокчейн - це однорангова мережа. Однорангова мережа відноситься до неієрархічної мережі, свого роду архітектури в межах інформатики, де участь та завдання поділяються рівними між рівнями. Що стосується інформатики, кожен системний компонент називається комп'ютерним вузлом. Вузол являє собою або пристрої, або точки даних. Комп'ютер діє як вузол, оскільки має IP-адресу, але також кожне посилання, на яке натискається, наприклад, на веб-сторінці компанії, оскільки воно є частиною більшої структури даних. Однорангова мережа - це приклад вузлів, що діють в децентралізованій архітектурі програмного забезпечення. Децентралізована архітектура системи - це система, де влада або відповідальність розподіляються на кожен окремий вузол[12]. Навпаки централізована системна архітектура - де функції здійснюються через центральний елемент, дивіться рисунок 2.1.

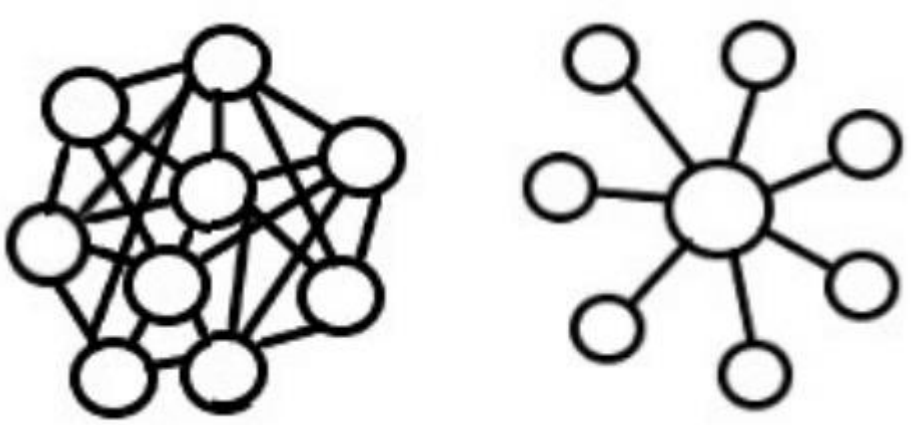


Рисунок 2.1 - Різниця децентралізованої (ліворуч) та централізованої (праворуч) архітектури програмного забезпечення

У блокчейні кожен вузол представляє користувача мережі. Жоден користувач не має конкретної ролі та всі користувачі взаємодіють на одних і тих же умовах, тобто вони є і постачальниками, і споживачами ресурса.

2.1.2 Асиметричне шифрування

Більшість систем криптовалют або систем, заснованих на технології блокчейн, зараз використовують алгоритм ECDSA[13].

На відміну від RSA, ECDSA базується на набагато більш складних математичних обчисленнях. А використовується ECDSA тільки для цифрових підписів. Вперше використання алгебраїчних властивостей еліптичних кривих в криптографії було запропоновано в 1985 році. ECDSA як стандарт цифрового підпису почав застосовуватися з кінця 1990-х років.

Крипостійкість ECDSA заснована на математичній проблеми дискретного логарифма в групі точок еліптичної кривої. Навіть з назви видно, що ця математична задача досить складна. Набагато складніше, ніж математична задача факторизації числа, на якій заснований RSA. Еліптичної крива - це безліч пар точок (X і Y), які задовольняють рівняння: $\mathbf{y^2 = ax^3} + bx + c$, рисунок 2.2.

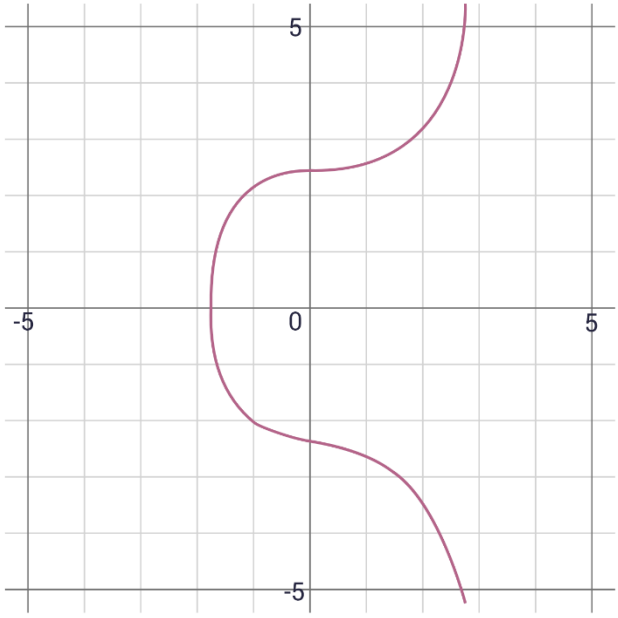


Рисунок 2.2 – Еліптична крива біткоіну, описана рівнянням $y^2 = x^3 + 7$

ECDSA використовує не тільки алгебраїчні властивості еліптичних кривих. А ще й кінцеві поля. Кінцеве поле - це заданий діапазон позитивних чисел, в рамках якого лежать результати алгебраїчних обчислень.

Еліптичні криві в рамках кінцевого поля змінюються до невпізнання. Але все математичні властивості кривої залишаються колишніми.

Завдяки всім цим чудовим математичним властивостям до сих пір не вдалося зламати ECDSA за допомогою обчислювальних потужностей. Вийшло лише зламати еліптичну цифровий підпис на смартфонах. Та й то за допомогою так званої "сторонньої атаки". Справа в тому, що смартфони при генерації цифрового підпису ECDSA споживають певну кількість енергії. І генерують при цьому електромагнітні хвилі. Кіберзлочинець може виміряти довжину хвилі. І

потім витягти секретний ключ. Але для цього треба перебувати в безпосередній близькості від пристрою.

У чому перевага ECDSA перед RSA? Для досягнення приблизно однакового рівня захисту для еліптичного шифрування потрібно ключ значно меншої довжини, ніж у RSA.

Як було написано вище, ECDSA використовується тільки в цифрових підписах. Цифровий підпис - це зашифрований хеш повідомлення. Процес виглядає так. Нам потрібно спочатку отримати хеш повідомлення. Оригінал тексту обробляється за допомогою хеш-функції (хешіруют). А потім отриманий необоротний хеш (цифровий відбиток повідомлення) перетворять в бітове число. І вже це число шифрується за допомогою ECDSA.

Для створення цифрового підпису ECDSA використовується приватний ключ. А ось перевірити справжність цифрового підпису може будь-хто за допомогою публічного ключа.

2.1.3 Хешування

Хешування відноситься до процесу створення певного висновку з вхідних даних різного розміру. Це робиться за допомогою математичних формул, також відомих як хеш-функції (реалізовані у вигляді алгоритмів хешування)[14].

Не всі хеш-функції припускають використання криптографії, а тільки ті, які спеціально призначені для цього, так звані криптографічні хеш-функції, які лежать в основі криптовалюти. Завдяки їх роботі, блокчейни і інші розподілені системи здатні досягти високого рівня цілісності даних і безпеки.

Як звичайні, так і криптографічні хеш-функції є детермінованими. Бути детермінованим означає, що до тих пір, поки вхідні дані не змінюються, алгоритм хешування завжди буде видавати один і той же результат (також відомий як дайджест або хеш).

Алгоритми хешування в криптовалютах розроблені таким чином, що їх функція працює в односторонньому порядку, це означає, що дані не можуть бути повернуті в зворотному порядку без вкладення великої кількості часу і ресурсів для здійснення обчислень. Іншими словами, досить легко створити вихід з вхідних даних, але відносно важко здійснити процес в зворотному напрямку (згенерувати висновок на основі вхідних даних). Чим складніше знайти вхідний значення, тим більш безпечним вважається алгоритм хешування.

Різні види хеш-функцій виробляють висновок різної величини, але можливий розмір даних на виході для кожного з алгоритмів хешування завжди є постійним. Наприклад, алгоритм SHA-256 може виробляти висновок виключно в форматі 256-біт, в той час як SHA-1 завжди генерує 160-бітний дайджест.

Щоб проілюструвати це, давайте припустимо слова "Binance" і "binance" через алгоритм хешування SHA-256 (той, який використовується в біткоіні), рисунок 2.3.

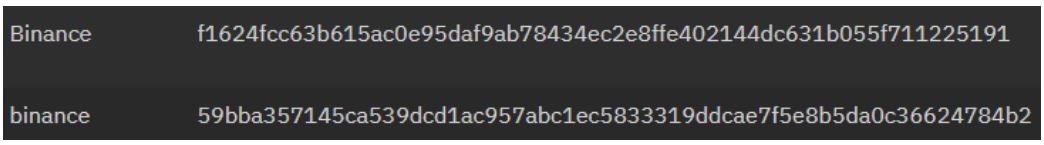


Рисунок 2.3 - Результати хешування

Зверніть увагу, що незначна зміна (реєстр першої літери) призвело до зовсім іншого значення хеша. Оскільки ми використовуємо SHA-256, дані на виході завжди матимуть фіксований розмір в 256 біт (або 64 символу), незалежно від величини введення. Крім цього, не має значення скільки разів ми пропустимо ці два слова через алгоритм, два виходу не будуть видозмінюватися, оскільки вони є постійними.

Звичайні хеш-функції мають широкий спектр варіантів використання, включаючи пошук по базі даних, аналіз великих файлів і управління даними. У свою чергу, криптографічні хеш-функції широко використовуються в додатках пов'язаних з інформаційною безпекою для аутентифікації повідомлень і цифровий дактилоскопії. Коли мова заходить про біткоіни, криптографічні хеш-функції є невід'ємною частиною в процесі Майнінг, а також займають основну роль в генерації нових ключів і адрес.

Хешування демонструє весь свій потенціал при роботі з величезною кількістю інформації. Наприклад, можна пропустити великий файл або набір даних через хеш-функцію, а потім використовувати висновок для швидкої перевірки точності і цілісності даних. Це можливо завдяки детермінованою природі хеш-функцій: вхід завжди буде призводити до спрощеного стислому виходу (хешу). Такий метод усуває необхідність зберігати і запам'ятовувати великі обсяги даних.

Хешування є особливо корисним у відношенні технології блокчейн. У блокчейні біткоіну здійснюється кілька операцій, які включають себе хешування, велика частина якого полягає у майнінгу. За фактом, практично всі кріптовалютні протоколи покладаються на змішування для зв'язування і стиснення груп транзакцій в блоки, а також для створення криптографічного взаємозв'язку і ефективної побудови ланцюжка з блоків.

Знову ж звертаємо увагу на те, що хеш-функція, яка використовує криптографічні методи, може бути визначена як криптографічний хеш-функція. Для того, щоб її зламати потрібно безліч спроб грубого підбору чисел. Щоб реверсировать криптографічний хеш-функцію, потрібно підбирати вхідні дані методом проб і помилок, поки не буде отримано відповідний висновок. Проте, існує можливість того, що різні входи будуть виробляти однаковий висновок, в такому випадку виникає колізія.

З технічної точки зору, криптографічна хеш-функція повинна відповідати трьом властивостям, щоб вважатися безпечною. Можемо описати їх як: стійкість до колізії, і стійкість до пошуку першого і другого прообразу.

Перш ніж почати розбирати кожну властивість, узагальнимо їх логіку в трьох коротких пропозиціях.

Стійкість до колізії: неможливо знайти два різних входу, які виробляють хеш, аналогічний висновку.

Стійкість до пошуку першого прообразу: відсутність способу або алгоритму зворотного відновлення хеш-функцію (знаходження входу по заданому виходу).

Стійкість до пошуку другого прообразу: неможливо знайти будь-який другий вхід, який би перетинався з першим.

2.1.3.1 Стійкість до колізії

Як згадувалося раніше, колізія відбувається, коли різні вхідні дані виробляють однаковий хеш. Таким чином, хеш-функція вважається стійкою до колізій до тих пір, поки хто-небудь не виявить колізію. Зверніть увагу, що колізії завжди будуть існувати для будь-якої з хеш-функцій, в зв'язку з безліччю вхідних даних і обмеженою кількістю висновків.

Таким чином, хеш-функція стійка до колізії, коли ймовірність її виявлення настільки мала, що для цього будуть потрібні мільйони років обчислень. З цієї причини, незважаючи на те, що не існує хеш-функцій без колізій, деякі з них на стільки сильні, що можуть вважатися стійкими (наприклад, SHA-256).

Серед різних алгоритмів SHA групи SHA-0 і SHA-1 перестають бути безпечними, оскільки в них були виявлені колізії. В даний час тільки групи SHA-2 і SHA-3 вважаються найбезпечнішими і стійкими до колізій[15].

2.1.3.2 Стійкість до пошуку першого прообразу

Дана властивість тісно взаємопов'язано з концепцією односторонніх функцій. Хеш-функція вважається стійкою до пошуку першого прообразу, до тих пір, поки існує дуже низька ймовірність того, що хтось зможе знайти вхід, за допомогою якого можна буде згенерувати певний висновок.

Зверніть увагу, що це властивість відрізняється від попереднього, оскільки зловмисникові потрібно вгадувати вхідні дані, спираючись на певний висновок. Такий вид колізії відбувається, коли хтось знаходить два різних входу, які виробляють один і той же код на виході, не надаючи значення вхідних даних, які для цього використовувалися.

Властивість стійкості до пошуку першого прообразу є цінним для захисту даних, оскільки простий хеш повідомлення може довести його справжність без необхідності розголошення додаткової інформації. На практиці багато постачальників послуг і веб-додатки зберігають і використовують хеші, згенеровані з паролів замість того, щоб користуватися ними в текстовому форматі.

2.1.3.3 Стійкість до пошуку другого прообразу

Для спрощення розуміння, можна сказати, що даний вид стійкості знаходиться десь між двома іншими властивостями. Атака знаходження другого прообразу полягає в знаходженні певного входу, за допомогою якого можна

згенерувати висновок, який спочатку створювався за допомогою інших вхідних даних, які були заздалегідь відомі.

Іншими словами, атака знаходження другого прообразу включає в себе виявлення колізії, але замість пошуку двох випадкових входів, які генерують один і той же хеш, атака націлена на пошук вхідних даних, за допомогою яких можна відтворити хеш, який спочатку був згенерований за допомогою іншого входу .

Отже, будь-яка хеш-функція, стійка до колізій, також стійка і до подібних атак, оскільки остання завжди має на увазі колізію. Проте, все ще залишається можливість для здійснення атаки знаходження першого прообразу на функцію стійку до колізій, оскільки це передбачає пошук одних вхідних даних за допомогою одного висновку.

2.1.4 Дерево Меркла

Вузли в блокчейн-мережі анонімні і працюють в умовах відсутності довіри. У цій ситуації постає проблема верифікації даних: як перевірити, що в блоці записані коректні транзакції? Для оцінки кожного блоку знадобиться багато часу і обчислювальних ресурсів. Вирішити проблему і спростити процес допомагають дерева Меркла.

Блоки в біткойн-блокчейне - це перманентно записуємі файли, які містять інформацію про проведені користувачами транзакціях. Додатково кожен блок містить Generation Transaction (або Coinbase Transaction) - це транзакція з інформацією про адресу з нагородою за рішення блоку, яка завжди стоїть першою в списку.

Всі транзакції в блоці представлені як рядки в шістнадцятковому форматі, які хешіруются для отримання ідентифікаторів транзакцій. На їх основі будується хеш блоку, який враховується наступним блоком, забезпечуючи незмінність і зв'язність реєстру. Єдине хеш-значення блоку збирається за допомогою дерева Меркла, концепція якого була запатентована Ральфом Мерклом (Ralph Charles Merkle) в 1979 році.

Дерево Меркла, або хеш-дерево, - це двійкове дерево, кінцеві вузли якого - це хеші транзакцій, а внутрішні вершини - результати складання значень пов'язаних вершин. Ось приклад хеш-дерева з трьома транзакціями, рисунок 2.4.

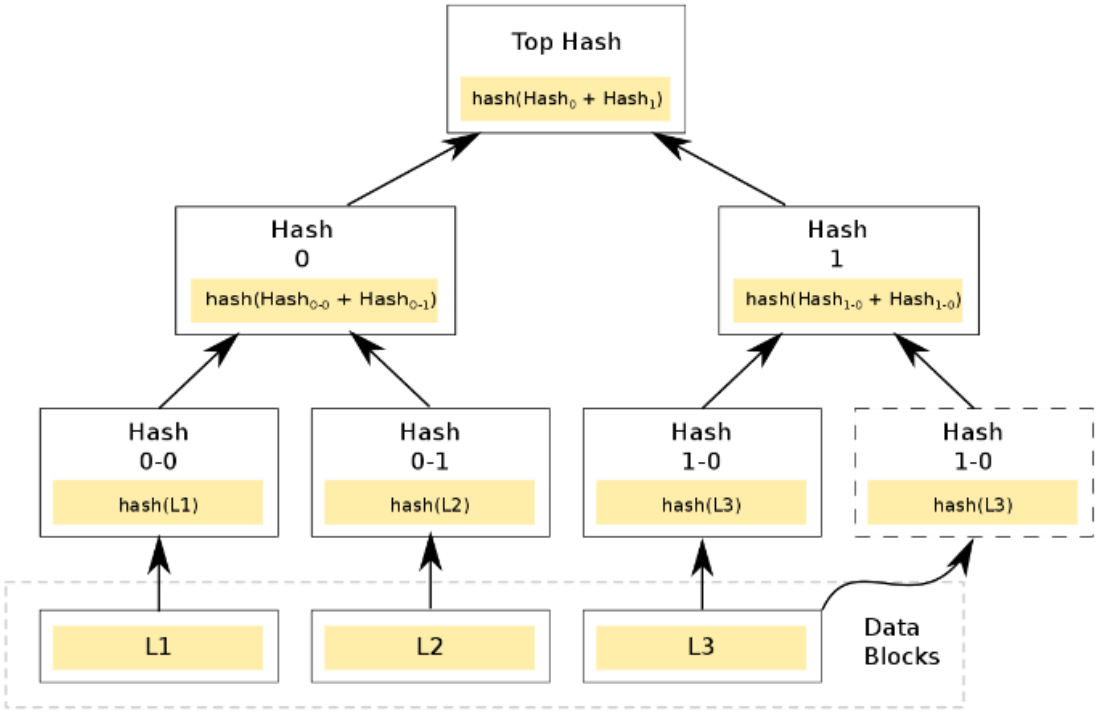


Рисунок 2.4 – Приклад хеш-дерева з трьома транзакціями

Побудова дерева відбувається наступним чином.

Обчислюються хеш-кодування транзакцій, розміщених в блоці: hash (L1), hash (L2), hash (L3) і так далі.

Обчислюються хеш-кодування від суми хешів транзакцій, наприклад hash (hash (L1) + hash (L2)). Так як дерево Меркла є бінарним, то число елементів на кожній ітерації має бути парним. Тому якщо блок містить непарну кількість транзакцій, то остання дублюється і складається сама з собою: hash (hash (L3) + hash (L3)).

Далі, знову обчислюються хеш-кодування від суми хешів. Процес повторюється, поки не буде отримано єдиний хеш - корінь дерева Меркле (merkle root). Він є криптографічним доказом цілісності блоку (тобто того, що всі транзакції знаходяться в заявленому порядку). Значення кореня фіксується в заголовку блоку[16].

2.1.5 Фільтр Блума

Фільтр Блума - це фільтруючий механізм імовірнісного пошуку, спосіб опису необхідного шаблону без його точного визначення. Фільтр Блума пропонує ефективний спосіб формування шаблону пошуку із забезпеченням захисту приватності. Такі фільтри використовуються SPV-вузлами при зверненні до партнерів із запитами транзакцій, відповідних заданим шаблоном, при цьому точно не вказується, які саме адреси, ключі або транзакції необхідно шукати.

Фільтр Блума виконує свою функцію, дозволяючи SPV-вузлу визначити зразок пошуку для транзакцій з можливістю настройки збереження більшої точності або секретності. Більш конкретно визначений фільтр Блума видасть точні результати, але платою за це будуть відкриті шаблони, якими цікавиться цей SPV-вузол, отже, і адреси, що належать гаманцю цього користувача. Менш конкретно визначений фільтр Блума видає більше даних про більшу кількість транзакцій, багато з яких не потрібні цьому вузла, але такий підхід дозволяє вузлу зберегти більшу ступінь секретності[17].

Фільтр Блума реалізований як масив змінного розміру, що складається з N бінарних цифр (бітових полів) і змінного кількості M хеш-функцій. Хеш-функції підбираються таким чином, щоб завжди генерувати результат між 1 і N, відповідний масиву бінарних цифр. Хеш-функції генеруються детерминированно, тому будь-який вузол, що застосовує фільтр Блума, завжди буде використовувати одні і ті ж хеш-функції і отримувати одні і ті ж результати для певних вхідних даних. Фільтр Блума можна налаштовувати, вибираючи різну довжину (N) фільтра і різну кількість (M) хеш-функцій, тим самим змінюючи рівень точності і відповідний рівень секретності.

На рисунку 2.5 показаний приклад дуже маленького масиву з 16 бітів і набір з трьох хеш-функцій для наочної демонстрації роботи фільтра Блума.

Рисунок 2.5 – Приклад спрощеного фільтра Блума з 16-бітовим масивом і трьома хеш функціями

Фільтр Блума инициализируется нулями у всіх бітових полях. Як додати шаблон в фільтр Блума цей шаблон хешіруется по черзі кожною хеш-функцією.

Результатом застосування першої хеш-функції до вхідних даними є число між 1 і N. Для відповідного біта в масиві (Проіндексований від 1 до N) встановлюється значення 1, тобто записується результат виконання першої хеш-функції. Потім виконується наступна хеш-функція і встановлюється відповідний її результату біт і т.. Після застосування всіх M хеш-функцій шаблон пошуку вважається записанним в фільтрі Блума як M бітів, значення яких змінені з 0 на 1.

На рисунку 2.6 показаний приклад додавання шаблону в простий фільтр Блума.

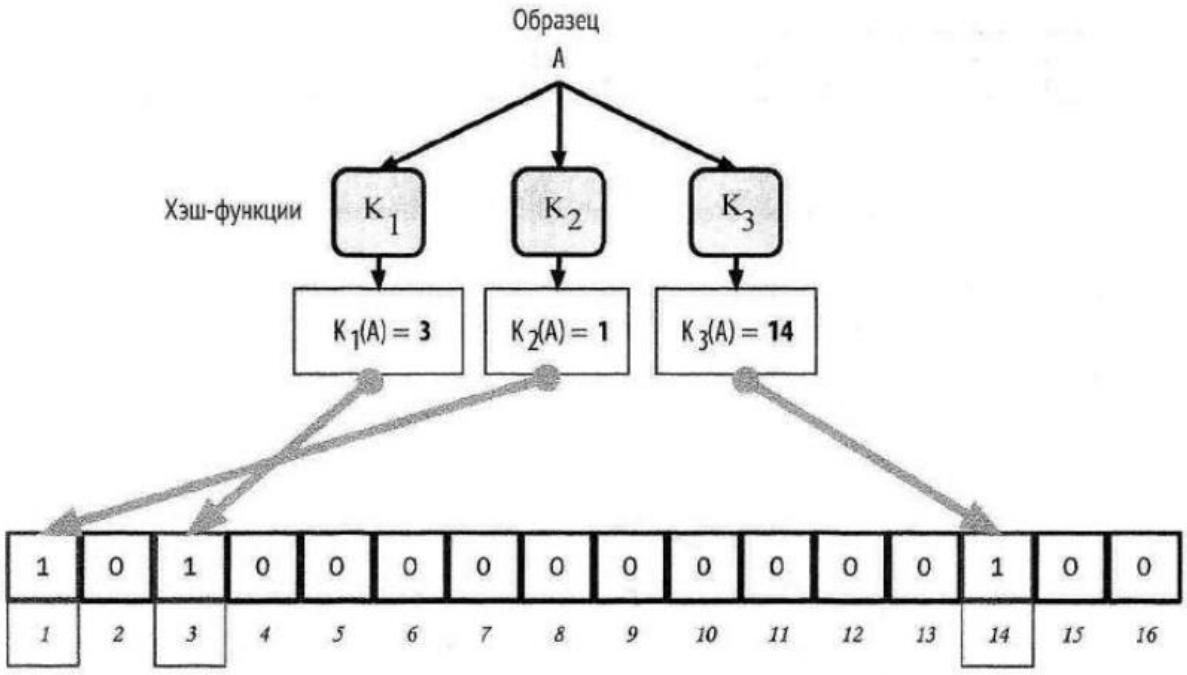


Рисунок 2.6 – Додавання шаблону «A» в простий фільтр Блума

Додавання другого шаблону виконується дуже просто, оскільки повторюється вищеописаний процес. Шаблон хешируется по черзі кожною

хеш-функцією, результати роботи фіксуються за допомогою установки для відповідних бітів значення 1. Оскільки фільтр Блума заповнюється декількома шаблонами, результат виконання хеш-функції може відповідати біту, для якого вже встановлено значення 1, і в цьому випадку біт не змінюється. По суті, чим більше шаблонів записується в перекриваємі бітові поля, тим більш насиченим одиничними бітами стає фільтр, при цьому точність фільтра знижується. Саме тому фільтр Блума є ймовірнісної структурою даних - він стає менш точним, якщо додається все більше шаблонів. точність залежить від кількості додаються шаблонів, що зіставляється з розміром бітового масиву (N), і числа хеш-функцій (M). Бітовий масив більшого розміру і більша кількість хеш-функцій дозволяють записати більше шаблонів з високою точністю. Бітовий масив малого розміру і невелика кількість хеш-функцій дозволяють записати менше шаблонів і забезпечують меншу точність.

Фільтри Блума використовуються для фільтрації транзакцій (і містять їх блоки), які SPV-вузол отримує від партнерів, вибираючи тільки цікавлячи його транзакції без розкриття відповідних адресів або ключів. SPV-вузол инициализирует фільтр Блума як «порожній», в цьому стані фільтру не відповідають ніякі шаблони. Потім SPV-вузол формує список всіх адрес, ключів і хеш-значень, в яких він зацікавлений. Це робиться за допомогою вилучення хеш-значення відкритого ключа, хеш-значення скрипта і ідентифікаторів транзакцій з будь-яких даних UTXO, керованих гаманцем цього вузла. Потім SPV-вузол додає кожен з обраних об'єктів в фільтр Блума, так що «відповідність» фільтру буде досягнуто, якщо задані шаблони присутні в транзакції, при цьому самі шаблони не розкриваються.

Після цього SPV-вузол відправляє партнеру повідомлення fi.lterload, що містить фільтр Блума, для використання на цьому з'єднанні. На стороні партнера фільтри Блума звіряються з кожної що входить транзакцією. Повноцінний вузол перевіряє кілька частин транзакції на відповідність фільтру Блума, виконуючи пошук збігів в наступних елементах:

1) ідентифікатор транзакції;
2) компоненти даних з блокуючих скриптів кожного фрагмента вихідних даних транзакції (кожен ключ і кожне хеш-значення в скрипті);
3) кожен фрагмент вхідних даних транзакції;
4) кожен компонент даних підписи вхідних фрагментів (або скриптів доказу).

При перевірці всіх перерахованих елементів фільтри Блума можна використовувати для пошуку збігів в хешах відкритих ключів, в скриптах, в значеннях, в відкритих ключах в підписах або в будь-якому компоненті смарт-контракту або складного скрипта, в тому числі і в тих компонентах, які з'являться в майбутньому.

Після установки фільтра партнер перевіряє з його допомогою вихідні дані кожної транзакції. Запитувачу вузлу відправляються тільки ті транзакції, які відповідають фільтру.

У відповідь на повідомлення getdata від запитувача вузла партнери відправляють повідомлення, що містить заголовки тільки тих блоків, які відповідають фільтру, і шлях в дереві Меркла для кожної знайденої транзакції. Потім партнери починають відправку повідомлень, що містять транзакції, відповідні фільтру.

Після початку передачі транзакцій з повноцінного вузла на запитувач SPV-вузол цей SPV-вузол відкидає всі помилкові збіги і використовує транзакції з «правильними» збігами для поновлення власного набору даних UTXO і балансу гаманця. Після поновлення локального уявлення набору даних UTXO SPV-вузол змінює фільтр Блума, визначаючи в ньому шаблони для відповідності всім майбутнім транзакціях, що посилаються на тільки що отримані дані UTXO. Потім повноцінний вузол використовує новий фільтр Блума для пошуку збігів в нових транзакціях, і весь процес повторюється.

Вузол, що встановлює фільтр Блума, може в інтерактивному режимі додавати шаблони в цей фільтр, посилаючи повідомлення для повної очищення фільтра Блума відправляється повідомлення оскільки немає можливості видаляти шаблони з фільтра, вузол повинен повністю очистити фільтр, потім відправити нові шаблони, якщо старий шаблон більше не потрібен.

2.2 Протокол BitTorrent

BitTorrent - пірінговий (Peer-To-Peer) мережевий протокол для кооперативного обміну файлами через Інтернет.

Файли передаються частинами, кожен torrent-клієнт, отримуючи (викачувавши) ці частини, в той же час віддає (закачує) їх іншим клієнтам, що знижує навантаження і залежність від кожного клієнта-джерела і забезпечує надмірність даних[18].

2.2.1 Принцип роботи протоколу

Перед початком скачування клієнт під'єднується до трекера за адресою, вказаною в торрент-файлі, повідомляє йому свою адресу і хеш-суму торрент-файлу, на що у відповідь клієнт отримує адреси інших клієнтів, що викачують або роздають цей же файл. Далі клієнт періодично інформує трекер про хід процесу і отримує оновлений список адрес. Цей процес називається оголошенням.

Клієнти з'єднуються один з одним і обмінюються сегментами файлів без безпосередньої участі трекера, який лише зберігає інформацію, отриману від підключених до обміну клієнтів, список самих клієнтів і іншу статистичну інформацію. Для ефективної роботи мережі BitTorrent необхідно, щоб якомога більше клієнтів були здатні приймати вхідні з'єднання. Неправильне налаштування NAT або брандмауера можуть цьому перешкодити.

При з'єднанні клієнти відразу обмінюються інформацією про наявні у них сегментах. Клієнт, що бажає викачати сегмент (лічер), надсилає запит і, якщо другий клієнт готовий віддавати, отримує цей сегмент. Після цього клієнт перевіряє контрольну суму сегменту. Якщо вона збіглася з тією, що записана в торрент-файлі, то сегмент вважається успішно скачаним, і клієнт оповіщає всіх приєднаних бенкетів про наявність у нього цього сегменту. Якщо ж контрольні суми розрізняються, то сегмент починає скачиватися заново. Деякі клієнти банять тих бенкетів, які занадто часто віддають некоректні сегменти.

Таким чином, обсяг службової інформації (розмір торрент-файлу і розмір повідомлень зі списком сегментів) безпосередньо залежить від кількості, а значить, і розміру сегментів. Тому при виборі сегмента необхідно дотримуватися балансу: з одного боку, при великому розмірі сегмента обсяг службової інформації буде менше, але в разі помилки перевірки контрольної суми доведеться заново завантажувати більше інформації. З іншого боку, при малому розмірі помилки не так критичні, так як необхідно заново завантажити менший

обсяг, але зате розмір торрент-файлу і повідомлень про наявні сегментах стає більше.

2.2.2 Алгоритм обміну даними

Кожен клієнт має можливість тимчасово блокувати віддачу іншому клієнтові. Це робиться для більш ефективного використання каналу віддачі. Крім того, при виборі - кого розблоковувати, перевага віддається бенкетам, які самі передали цьому клієнтові багато сегментів. Таким чином, бенкети з хорошими швидкостями віддачі заохочують один одного за принципом «ти - мені, я - тобі».

Обмін сегментами ведеться за принципом «ти - мені, я - тобі» симетрично в двох напрямках. Клієнти повідомляють один одному про наявні у них сегментах при підключенні і потім при отриманні нових сегментів, і тому кожен клієнт може зберігати інформацію про те, які сегменти є у інших підключених бенкетів. Порядок обміну вибирається таким чином, щоб спочатку клієнти обмінювалися найбільш рідкісними сегментами: таким чином підвищується доступність файлів в роздачі. У той же час вибір сегмента серед найбільш рідкісних випадковий, і тому можна уникнути ситуації, коли всі клієнти починають завантажувати один і той же самий рідкісний сегмент, що негативно б відбилося на продуктивності.

Обмін даними починається, коли обидві сторони в ньому зацікавлені, тобто, кожна зі сторін має сегменти, яких немає в іншої. Кількість переданих сегментів підраховується, і якщо одна зі сторін виявляє, що передає в середньому більше, ніж приймає, вона блокує на деякий час віддачу іншій стороні. Таким чином, в протокол закладена захист від лічерів.

Сегменти поділяються на блоки розміром 16-4096 кілобайт, і кожен клієнт запитує саме ці блоки. Одночасно можуть запитуватися блоки з різних сегментів. Більш того, деякі клієнти підтримують скачування блоків одного сегмента у різних бенкетів. В цьому випадку описані вище алгоритми і механізми обміну застосовні і до рівня блоків.

2.2.3 Протоколи і порти

Клієнти з'єднуються з трекером по протоколу TCP. Найбільш часто використовуваний входить порт трекера: 6969. Найбільш часто використовуваний діапазон вхідних портів клієнтів: 6881-6889.

Номери портів не фіксовані в специфікації протоколу і можуть змінюватися при необхідності. В даний момент більшість трекерів використовують звичайний HTTP порт 80, а для клієнтів рекомендується вибрати випадковий вхідний порт. Більш того, деякі трекери не допускають використання портів клієнтів з стандартного діапазону 6881-6889, так як деякі провайдери забороняють використання цього діапазону портів.

DHT-мережу в BitTorrent-клієнтах використовує протокол UDP.

Крім того, протокол UDP використовується UDP-трекерами (підтримується не всіма клієнтами і не є офіційною частиною протоколу) і для з'єднання клієнтів один з одним через UDP NAT Traversal[19].

2.3 Новітні технології для поліпшення існуючих платформ

2.3.1 Схема Шнорра

Схема Шнорра - це алгоритм цифрових підписів. Алгоритм цифрових підписів, серед іншого, визначає відносини між громадськими структурами і приватними ключами («адресою» і «паролем») - а значить, його вибір вкрай важливий для безпеки.

Крім того, оскільки цифрові підписи складають значну частину даних, з яких складається транзакція, вибір алгоритму цифрових підписів вкрай важливий для конфіденційності та ефективності.

У разі прийняття підписи Шнорра стануть альтернативою поточним алгоритмом підписів на основі еліптичних кривих в біткоіні.

Перш за все, підписи Шнорра привертають тим, що легко обчислюються і вважаються надзвичайно безпечними. Однак головні переваги підписів Шнорра пов'язані з їх властивістю агрегування.

Підписи Шнорра здатні агрегувати кілька окремих підписів в єдину підпис. Така властивість агрегування підписів особливо цінно в світлі того, який обсяг пам'яті займають дані підписів в транзакції біткоіни, рисунок 2.7.

Рисунок 2.7 - Стандартна транзакція біткоіна. Кількість символів, які необхідні для підписів (виділені жовтим)

Агрегування підписів Шнорра потенційно корисно в декількох аспектах, що дають переваги мережі біткоіни і її користувачам.

По-перше, здатність агрегувати кілька підписів в єдину підпис особливо цінна для «транзакцій з мультіподпісью» - тобто транзакцій біткоіни, що вимагають кілька підписів, щоб мережа визнала їх дійсними. При поточній структурі біткоіни ці транзакції з мультіподпісью мають набагато більший розмір, ніж стандартні транзакції з одним підписом, - що несе негативні наслідки для ефективності та конфіденційності.

По-друге, виявляється, можливо розширити концепцію агрегування підписів Шнорра - за допомогою схеми, відомої як MuSig, - для агрегування

підписів кількох UTXO (невитрачених виходів транзакцій) в єдину підпис. Теоретично це той же процес, що і в наведеному вище прикладі, тільки замість того, щоб просто агрегувати кілька підписів, необхідних для витрачання одного UTXO, ми розширюємо концепцію, включаючи підписи кількох UTXO[20].

Що стосується ефективності, головною перевагою є скорочення розміру транзакцій - що означає зниження витрат на зберігання і обчислення. Транзакції з мультіподпісью Шнорра навіть ще більш компактні й ефективні, ніж нинішні транзакції біткоіни з одним підписом. Це важливо, тому що знижуються комісії транзакцій для користувачів і мінімізуються ресурсні вимоги для учасників мережі.

Крім того, так як розмір і витрати транзакцій з мультіподпісямі Шнорра такі ж, як у транзакцій з одним підписом, прийняття підписів Шнорра має сприяти збільшенню різноманітності - і, можливо, складності - транзакцій з мультіподпісью в мережі. Це подвійний виграш: користувачі зможуть створювати складніші схеми транзакцій, не перевантажуючи мережу і не несучи додаткових витрат.

Що стосується конфіденційності, підписи Шнорра (або схеми на основі підписів Шнорра, такі як MuSig) мають дві переваги. По-перше, транзакції з мультіподпісью відрізняються від транзакцій з одним підписом. По-друге, агрегована мультіподпісь Шнорра не розкриває індивідуальні публічні ключі входів (учасників контракту з мультіподпісью).

Іншими словами, підписи Шнорра допомагають уникнути витоку інформації про власників публічних ключів контракту з мультіподпісью і навіть допомагають приховати, чи має транзакція один підпис або кілька.

Нарешті, схема MuSig на основі підписів Шнорра також може запропонувати непряме перевага для конфіденційності, поліпшивши економіку контрактів з мультіподпісью: якщо ми можемо агрегувати підписи кількох UTXO, то MuSig може сприяти використанню функцій, що поліпшують конфіденційність, таких як «coinjoin». Тобто, з MuSig користувачі можуть досягти менших транзакційних витрат, агрегируя свої транзакції з іншими (фактично розділяючи витрати на розмір транзакції з іншими), - що може поліпшити конфіденційність мережі в цілому.

2.4 Висновок з розділу 2

У цьому розділі ми оглянули усі основні принципи роботи таких технологій як блокчейн та BitTorrent. Описали методи, які будуть використовуватись при розробці майбутнього ПЗ.

Також оглянули нові технології, які зараз не користуються популярністю, але можуть нам дуже допомогти при розробці ПЗ.

3 РОЗРОБЛЕННЯ ПРОТОТИПУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАХИЩЕНОГО СХОВИЩА ІНФОРМАЦІЇ НА БАЗІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

3.1 Архітектура та протокол BitTorrent

BitTorrent - це мережевий протокол прикладного рівня, на якому ви отримали файлів. Він використовує однорангову (P2P) мережеву архітектуру, в якій багато однорангових користувачів виступають в якості клієнта і сервера, завантажуючи від peer-ів в той же самий час, коли вони завантажуються іншими. Продуктивність збільшується в міру того, як кількість загрузчиков збільшується, що робить систему масштабуєму. Вона також використовує клієнт-серверну архітектуру, в якій peer-и зв'язуються з сервером, щоб знайти інших peer-ів, до яких вони можуть підключитися.

3.1.1 Архітектура BitTorrent

BitTorrent - це гібридна мережа, що використовує архітектуру клієнт сервер, так і архітектуру однорангової мережі. Централізований сервер називається трекером. Відповідальність трекера - допомогти peer-ам знайти інших peer-ів. Трекер складається з безлічі торрент-сесій з кожним сеансом, який він зберігає відстеження всіх peer-ів, які беруть участь у конкретному торренті. Peer контактує з трекером і трекер відповідає списком peer-ів, до яких він може підключитися. Трекер не несе відповідальності за фактичний розподіл вмісту взагалі. Пропускна здатність трекера дуже низька, оскільки вона є простим протоколом, до якого peer-и підключаються лише під час запуску та через визначені інтервали часу, зазвичай 30 хвилин. Peer знає URL-адресу трекера, оскільки вона визначена в торрент файлі.

Торрент-файл - це статичний файл "metainfo", який представляє розподілений сеанс вмісту. Торрент-файл створюється з URL-адресою трекера та фактичним файлом або файлами, які будуть частиною цього потоку. Формат торрент-файлу - це кодування. Bin-кодування складається з вкладених словників та списків. Ці словники та списки можуть містити рядки та цілі числа. Потік файл - це закодований словник, що містить два ключі, оголошення та інформацію. Ключ оголошення - URL-адреса трекера. Інформаційна клавіша відображається у словнику, описаному нижче. Інформаційна клавіша - це словник із такими клавішами: ім'я, довжина шматка, шматки та довжина або файл. Ключ імені - це рядок, який є запропонованою назвою для завантаження файлу. Запропоновано означати, що завантажувач може вибрати ім’я, яке бажає. Довжина шматка відповідає кількості байтів на кожен фрагмент, на який розділений файл. Файл розділений на шматочки однакового розміру, за винятком можливо останнього шматка. Ключ фрагментів відображається в рядку SHA1 хешу кожного шматка. Кожен хеш SHA1 - це рядок довжиною 20 символів, наприклад, хеш значення фрагменту, якого 4 буде підрядком фрагментів із символів 60 - 79, якщо

припустити, що рядок починається з індексу 0. Це використовується, щоб завантажувач міг перевірити дані, перевіривши що вони завантажуються з цими хеш-значеннями. Наступна клавіша може бути довжиною або файлами, залежно від того, чи цей торрент - це один файл або каталог файлів. Для одного файлу ключовою є довжина, і це просто довжина файлу в байтах. Якщо Торрент представляє каталог, в якому використовуються ключові файли. Цей ключ відображається у списку файлів і містить список словників, що містять такі ключі, довжину та шлях. Довжина - це число байтів файлу і шлях - це список рядків, які відповідають іменам підкаталогів.

Тепер, коли пір має торрент-файл і підключається до трекера для отримання списку peer-ів, до якого він може приєднатися. Співрозмірник спочатку повинен виділити місце для файлу або файлів, які він отримує з торрент-файлу. Це необхідно, оскільки peer не завантажує фрагменти файлу для того, щоб BitTorrent було потрібно зібрати ці частини, щоб отримати їх. Однорангова мережа - це мережа, в якій peer-и виступають як клієнти, сервери та сівалки виступають просто як сервери. Peer-и розподіляють файл між собою за допомогою техніки розподілення. Peer-и завантажують шматки від кількох peer-ів одночасно.

3.1.2 Tracker Протокол

Трекер відіграє важливу роль у BitTorrent, тому що без нього його не було б способу для peer-ів знайти інших peer-ів для завантаження. Трекери використовують простий протокол, накладений на верхню частину HTTP. Трекер отримує HTTP-запит GET, і він надсилає кодовані повідомлення на запит peer-а. Запити GET трекера містять такі ключі; info_hash, peer_id, ip, port, uploaded, downloaded, left, and event . Поле info_hash визначає спосіб відстеження до якого торрент-сеансу клієнт входить або приєднується. Цей ключ - 20-байтовий хеш SHA1 із торрент-файлу. Peer_id - це ідентифікатор, який клієнт випадково створив напочатку завантаження. Це рядок довжиною 20 символів. Наступний ключ, ip, є необов’язковим, тобто зазвичай використовується для джерела, якщо воно знаходиться на тому самому комп’ютері, що і трекер. Ключ порту - це порт номер, який слухає peer. Uploaded, downloaded, left відповідає загальній кількості завантаження, загальна кількість завантаження у певний час і кількість байтів, що залишилось для завантаження відповідно. Вони використовуються, щоб трекер міг відстежувати статистику. Наприклад, це може зараз скільки сівалок і п’явок на торрент-сесії або скільки разів має певний вміст було завантажено. Наступне поле event необов’язкове, яке може мати такі значення, як started, completed, stopped, або empty. Started використовується, коли завантаження тільки починається. Completed використовується коли завантажувач закінчив завантаження. Stopped, коли рівний пристрій припиняє завантаження. Empty використовується під час анонсів, які робітники роблять через рівні проміжки часу. Відповіді, які трекер надсилає peer-ам, - це закодовані словники. Цей словник повинен містити два ключі, інтервальний та одноранговий. Інтервальна клавіша - це кількість секунд, що відповідає, скільки peer-у слід чекати між регулярними запитами. Ключі-партнери відображають

список словників, який представляє список peer-ів, який містить ключі, ідентифікатор однорангової мережі, ip та порт. Ідентифікатор однорангової мережі - це ідентифікатор мережі надісланий трекеру в запиті трекера. IP і порт - це IP-адреса та номер порту peer-ів. Зазвичай у цій відповіді трекер повертає 50 випадкових peer-ів. Зверніть увагу, як це може відстежувати статистику, записуючи всю інформацію, яку вона отримує від запитів peer-ів. пропускна здатність трекера дуже низька, оскільки peer-и підключаються до трекера лише на дуже короткий час через тривалі проміжки часу (зазвичай 30 хвилин). Загальна кількість смуги пропускання, яку використовує трекер в даний час близько тисячної загальної кількості використовуваної смуги пропускання[21].

3.1.3 BitTorrent Peer Protocol

Peer-и спілкуються між собою, надсилаючи повідомлення безпосередньо один одному за допомогою BitTorrent Peer Protocol. Цей протокол працює через TCP. Для того, щоб відправити двох peer-ів повідомляючи один одному, спочатку вони повинні з’єднатися між собою, надіславши повідомлення про рукостискання. Це повідомлення про рукостискання починається з рядка "19 BitTorrent Protocol". 19 – довжина префіксу. Після цього рядка є вісім зарезервованих байтів, які наразі не використовуються, але вони є додані, щоб дозволити розширювати протокол. Наступні 20 байт - це хеш-значення SHA1 інформаційне значення торрент-файлу. Це те саме значення, яке використовується в запитах трекера. Ця величина становить необхідний, оскільки peer може брати участь у багатьох потокових роях, і коли рівний посилає повідомлення про рукостискання, він повинен знати, до якого конкретно рою він приєднується. Наступні 20 байт – це 20-байтовий ідентифікатор однорангової мережі, що є тим самим значенням, яке надсилається трекеру в запитах. Це завершує підключення, і тепер peer-и можуть почати надсилати інші типи повідомлень до цього і всі їхні зв’язані peer-и у своєму наборі peer-ів. Для всіх зв’язків, які має peer, він також повинен зберігати конкретну інформацію про них. Всі з'єднання знаходяться або в задушеному, або в незамкненому стані. Якщо peer задихається, тоді не дозволяється завантажувати дані у цьому зв’язку. Усі зв’язки також повинні бути зацікавленими. Колектив зацікавлений у іншому партнері, якщо у нього є частини вмісту, якого він не має. Тільки коли одноранговий інтерес зацікавлений у іншому peer-і та його відключення він може завантажити з з'єднання. Оскільки peer повинен надсилати повідомлення, щоб заявити, що він зацікавлен, всі peer-и також повинні знати, які фрагменти вмісту вже завантажив peer. Окрім повідомлення про рукостискання, є ще 9 повідомлень які можна надіслати іншим peer-ам. Ці повідомлення відрізняються один від одного першим байтом, який вказує, що це за тип повідомлення[22]. Ці повідомлення:

1) choke;
2) unchoke;
3) interested;
4) not interested;

5) have;
6) bitfield;
7) request;
8) piece;
9) cancel.

Накладні витрати на завантаження та завантаження дуже низькі. Легаут визначив що накладні витрати в його експериментах становлять менше 2% . Ці повідомлення будуть пояснені в приклад Peer A, який приєднується до торрент-сесії та описує, які повідомлення він надсилає та отримує у складі рою. Перший Peer A надсилає запит на трекер, щоб отримати список peer-ів, до яких потрібно підключитися. Peer A продовжує ініціювати з'єднання BitTorrent з підмножиною цього списку, надсилаючи повідомлення рукостискання. Зазвичай ініціюється не більше 40 з'єднань. Після того, як Peer A підключається до іншого peer-у, він очікує повідомлення бітового поля. Зазвичай це повідомлення надсилається лише в тому випадку, якщо peer вже завантажив принаймні один шматок. Це повідомлення є бітовим полем, яке відповідає фрагменти файлу, який peer уже завантажив. Це потрібно, тому що всі peer-и в рої мусять знати те, що є у всіх інших peer-ів. Peer A повинен підтримувати свій активний набір peer-ів. Цей набір складається з нерівнених peer-ів, які зазвичай містять лише чотирьох peer-ів. Цей набір повинен постійно змінюватись, щоб збільшити швидкість завантаження. Це робиться за допомогою алгоритму choke, що буде обговорено пізніше, але його не можна викликати, доки peer A не має хоча б одного повного твору. Після того, як Peer A має зв’язки з іншими peer-ми і він знає, які шматки у них є, він може тепер надсилайте зацікавлені повідомлення цим колегам, у яких є шматки, які він хоче. Це повідомлення не має зайвих даних, оскільки для цього не потрібно вказувати нічого конкретного. Коли peer отримує це повідомлення від Peer A, вони оновлюють стан цього peer-у до зацікавленого. Коли Peer A отримує ці повідомлення, він може продовжити, надсилаючи повідомлення запиту рівному користувачеві, який їх відключив. Запитувати повідомлення містять три параметри; покажчик, початок і довжина. Індекс посилається на фрагмент файлу та початок і довжина використовуються для вказівки, яку частину цього фрагмента вони хочуть називати блоками. Peer отримує поштучні повідомлення від peer-ів, які надсилають повідомлення про запит, які містять фактичні дані. Коли Peer A завантажує повний фрагмент (а не блок), він перевіряє цілісність цього фрагмента обчислюючи хеш цього фрагмента SHA1 і перевіряючи це значення порівняно із значенням у торрент-файлі. Якщо значення однакові, він надсилає повідомлення всім peer-ам, до яких підключений Peer A вказуючи, яку частину вона щойно завантажила. Також Peer A отримуватиме повідомлення від інших peer-ів, коли вони завершили завантаження твору. Peer A повинен підтримувати, які фігури всі peer-и мають та розсилають зацікавлені та незацікавлені повідомлення всім peer-ам, коли щось змінюється, або вони отримують повідомлення про наявність, або вони завантажили повну частину. Peer A не може завантажувати нічого до peer-ів, поки не завершить свою першу частину. Peer вирішає який з peer-ів повинен вибратидля завантаження за допомогою алгоритму choke.

3.1.4 Алгоритм Choke

Алгоритм choke використовується для зміни активного набору peer-ів, які є одноранговими вузлами. Це робить BitTorrent справедливим, оскільки peer-и завантажують їх до інших користувачів, які надають їм найшвидшу швидкість завантаження. Справедливості заради не слід дозволяти peer-ам завантажувати набагато більше вони завантажили. Оскільки однорангові завантаження здійснюються для peer-ів, які дозволяють завантажувати, цей алгоритм надає перевагу peer-ам, які завантажують, ніж ті, хто цього не робить. Цей алгоритм намагається максимально використовувати завантаження для, наприклад, якщо два peer-и отримують низький рівень завантаження, вони можуть почати завантаження один до одного і обоє peer-ів отримають кращу швидкість завантаження, ніж раніше. Спосіб, яким peer з'ясовує, чи може він отримати кращі швидкості передачі даних - це випробування невикористаних з'єднань на пробній основі. Алгоритм choke робить різницю між станом сівалки та пікави. Коли в стані вилучення, одноранговий сервер повинен підтримувати поточну швидкість завантаження для своїх однорангових з'єднань. Це алгоритм викликається кожні 10 секунд, або всякий раз, коли незацікавлений peer цікавиться або становиться незацікавленим. Час виклику цього алгоритму є важливим, оскільки ресурси витрачаються, якщо вони швидко задихаються і відбивають peer-ів. Десять секунд дозволяють з'єднанню TCP отримати повну інформаційну місткість. Цей алгоритм замовляє peer-ів, які зацікавлені та шлють блок через фрагмент повідомлення за останні 30 секунд. Ці peer-и сортуються за швидкістю завантаження. Якщо peer немає надісланих блоків за останні 30 секунд, peer вважається ослабленим. Охайні peer-и залишаються осторонь, цей алгоритм призначений для того, щоб гарантувати відключення активних peer-ів. Трійка найшвидших peer-ів відключені. Кожні три раунди, що становить 30 секунд, один peer вибирається випадковим чином. Цього peer-а називають запланованим оптимістичним невстановленим рівнем. Якщо цей peer є частиною трьох найшвидших peer-ів, інший - вибраний навмання. Якщо peer швидкий та зацікавлений, його можна вибрати як оптимістичний. Оптимістичне розстібання є необхідним, оскільки без нього peer-и не мали б способу виявити краще зв'язки, ніж ті, які вони мають зараз. Також без цього оптимістичного відхилення, нового peer-у, у яких немає штук, ніколи не зможуть отримати свою першу частину. Choke алгоритм відрізняється в стані затравки. У попередніх версіях BitTorrent алгоритм choke був однаковим у стані викіду як насіннєвий стан, за винятком того, що насіннєвий стан використовує швидкість завантаження, щоб визначити, яким peer-ам потрібно розстібнути. Причина, чому цю стару версію було змінено, оскільки вона надає перевагу peer-ам з високим швидкість завантаження. Це може дозволити peer-у домінувати над усіма ресурсами насіння, якщо воно має найвищу швидкість завантаження. Це може дозволити рівному користувачеві, який нічого не завантажує (вільний вершник), отримати висока швидкість завантаження. Це зробить протокол несправедливим, оскільки це можливо для peer-а завантажувати, ніколи не вносячи нічого назад.

Викликається поточний алгоритм choke кожні 10 секунд, ідля визначення того, що саме, використовуються лише незацікавлені та зацікавлені peer-и. Алгоритм впорядковує peer-ів за часом останнього відключення для всіх peer-ів, які були відключені нещодавно. Потім швидкість завантаження використовується для вибору між peer-ми з однаковим останнім незахопленим часом. Через цей крок peer-и не впорядковуються відповідно до швидкості завантаження, але не час їх останнього відштовхування. Це вигідніше, оскільки активний набір peer-ів змінюється частіше. Також кожні два з коли-небудь трьох раундів є першими трьома peer-ами unchocked та інший зацікавлений peer навмання. Третій тур, перші чотири peer-и unchocked. Дослідження Бхарамбе щодо аналізу та поліпшення роботи BitTorrent встановив, що цей сучасний алгоритм не запобігає несправедливості. Особливо в неоднорідних налаштування, коли однорангові з високою пропускною здатністю підключаються до однорангових з низькою пропускною здатністю. Він придумав а вирішення цієї проблеми шляхом включення трекера, що відповідає смузі пропускання. Хоча це Алгоритм не є досконалим, він ефективно виконує роботу з максимального використання та робить кращу роботу ніж інші програми обміну файлами P2P, намагаючись запобігти несправедливості[23].

3.1.5 Алгоритм відбору фрагментів

BitTorrent використовує найрідкісніший перший алгоритм, щоб визначити, який фрагмент йому слід наразі завантажити. Метою цього алгоритму є максимізація різноманітності доступних частин, які робить кількість копій кожного фрагмента максимально рівною. Це робить це більш малоймовірним peer-и матимуть проблеми із завантаженням фрагментів через “рідкісні” блоки, які важко знайти. Крім того, використовуючи цей алгоритм, більш вірогідно, що peer-у завжди буде що запропонувати інші peer-и. Оскільки кожен peer зберігає, які шматки є у всіх їхніх peer-ів у їхньому наборі peer-ів, це може тоді визначити, скільки копій штук можна завантажити. Він використовує цю інформацію для визначення, яку частину завантажити першою, відстежуючи найрідкісніші набори штук. Цей набір оновлено кожного разу, коли він отримує повідомлення про наявність або коли такий набір зник. Коли peer щойно приєднався до торренту і в даний час не має частин, він спочатку використовує випадкову політику. Ця політика застосовується до тих пір, поки peer не завантажить 3 повні частини. Мета випадкової першої політики полягає в тому, що випадковий шматок є більш вірогідним, ніж рідкісні твори, тому час завантаження буде швидшим. Це важливо для peer-у, завантажити свій перший шматок, тому що він не може виконати алгоритм choke, якщо у нього немає шматка, який цікавить іншого peer-а. Крім того, кожен раз, коли колега вибирає фрагмент для завантаження, він використовує жорстку політику пріоритетів, яка є на рівні блоків. Щоразу, коли peer завантажує блок твору, він обирає завантажити інші блоки того самого шматка. Це дозволить рівному завантажувачу завантажити повністю замість того, щоб мати купу блоків рідкісних шматків, не маючи цілих частин. це є важливо мати цілі фрагменти, оскільки peer не може завантажувати

блоки фрагмента, доки їх немає завантажив повну частину. Також коли викідник майже закінчує завантаження, він переходить у режим завершення гри. Цей режим запускається після того, як одноранговий запит запитує всі блоки. У цьому режимі колега запитує всі блоки, які він не отримав для всіх підключених peer-ів. Кожного разу, коли він отримує блок, він надсилає скасувати повідомлення для всіх інших підключених peer-ів. Ці повідомлення про скасування потрібні, щоб переконатися що не надто багато пропускної здатності витрачається на надлишкові штучні повідомлення. Доведено, що це період не витрачає занадто велику пропускну здатність, оскільки цей період дуже короткий і закінчується файлом, який завжди швидко завантажується.

3.2 Blockchain

Блокчейн можна визначити як децентралізовану база даних, структуровану в блоки, кожен з яких містить певний обсяг інформації та розповсюджується через ланцюжок (тобто книгу) над мережею. Отже, це а цифровий спосіб зберігання будь-яких даних через мережу. Зокрема, блокчейн - це розподілена мережа, означає, що дані, що містяться в книзі, є постійно ділиться та синхронізується через учасників, навіть якщо вони розподілені по кількох сайти, установи чи географії. Кожен учасник в мережа може отримати доступ до записів, якими спільно користуються книги та їх тиражувати. Якщо відповідає правилам встановлені в протоколі та підтверджені будь-яке доповнення до книги автоматично відображається у всіх його копіях. Основоположним каменем усієї технології блокчейн є так званий механізм консенсусу, який забезпечує це інформація, введена в блоки, є правильною і відповідно до правил, встановлених у протоколі. Практично це дозволяє обмінюватися інформацією між двома учасниками, що належать до мережі без необхідності транзиту для центрального утворення спочатку перевірити зміст.

3.2.1 Слої архітектури Blockchain

Перший крок для розуміння ділові та організаційні наслідки блокчейн додаток - це розуміння його IT-архітектури.Блокчейн можна розглядати як IT-архітектуру складається з трьох шарів:

1) верхній шар - додаток Blockchain. Фінал послуга, розроблена компанією з використанням блокчейн;

2) середній шар - книга блокчейнів. Розподілений книга, на якій побудований додаток блокчейн;

3) нижній шар - апаратне забезпечення / мережа блокчейну.

Мережа представлена сумою всіх вузлів використовуючи їх обчислювальну здатність до брати участь у механізмі консенсусу - підтвердження або відхилення нових транзакцій - і до зберігати всю історію транзакцій відбулося в цьому конкретному блокчейні.об'єкта.

3.2.2 Моделі управління блокчейном

Модель управління блокчейном може бути класифікованау двох вимірах: “дозволено / без дозволу” та “державний / приватний”. Перший вимір відноситься до можливість брати участь у механізмі консенсусу тоді як другий пов'язаний з можливістю для а користувачеві отримати доступ до належного додатку блокчейну.

У бездозвольних блокчейнах будь-хто, в тому числі злісні актори, можуть брати участь у консенсусі процес. Таким чином, кожен може вільно брати активну участь у мережі. Витрати вищі, а швидкість повільніша ніж на дозволеному ланцюжку.

Дозволені блокчейни зберігаються централізовано один (або кілька) авторизованих користувачів. У цьому випадку уповноважені користувачі перевіряють кожну транзакцію. Прочитайте дозволи можуть бути загальнодоступними або обмежені довільний обсяг. З іншого боку, як дозволені, так і без дозволу блокчейн може бути як загальнодоступним, так і приватний.

У публічній екосистемі кожен може приєднатися до мережі і використовувати додаток, увімкнений блокчейном технології. Користувач може отримати доступ до певної послуги без дозволу постачальника послуг. В публічного без дозволу блокчейну немає центральний орган влади та всі, хто має Інтернет з'єднання можна скористатися послугою (написати), прочитати історія транзакції (читати) і, врешті-решт, взяти участь у механізмі консенсусу (здійснити). Наприклад, біткойн - це публічна екосистема в Росії які кожна людина може надіслати / отримати а транзакція[24].

У приватному блокчейні відомі кінцеві і перевірено, і вони зможуть отримати доступ до послуга блокчейну, лише якщо постачальник послуг дозволяє їм. Тим самим проводяться учасники підлягає підзвітності на законних умовах (поза блокчейном) і стимулюються до поводитися чесно, щоб уникнути судового переслідування або наслідки. Наприклад, смарт-контракти а децентралізований додаток, написаний на Ethereum читати кожен, але доступ до сама програма може бути обмежена його розробники.

3.3 Розроблення прототипу программного забезпечення

3.3.1 Формування цілей і постановка задачі для розробки програмного забезпечення

Проектування програмного забезпечення варто почати з визначення мети, яку він повинен досягти. Для проекту, створеного в рамках даної роботи необхідно створити програмне забезпечення, яке дозволяє загружати та скачувати файли за допомогою таких технологій як BitTorrent та Blockchain.

На наступному етапі важливо декомпозувати завдання по розробці програми. Традиційно, розробка додатків складається з декількох незмінних етапів:

1) проектування і розробка дизайну користувальницького інтерфейсу, робота над досвідом взаємодії (UI/UX);

2) розробка користувальницького інтерфейсу і клієнтської частини додатку;

3) проектування структури бази даних, розробка серверної частини веб-додатку, інтеграція з базою даних.

3.3.2 Побудова діаграми варіантів використання

Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделю програмного забезпечення в процесі його проектування і розробки[25].

Розробка діаграми варіантів використання переслідує наступні мети:

− визначити спільні кордони і контекст модельованої предметної області на початкових етапах проектування програмного забезпечення;

− сформулювати загальні вимоги до функціональної поведінки проектованого програмного забезпечення;

− розробити вихідну концептуальну модель програмного забезпечення для її подальшого деталізації у формі логічних і фізичних моделей;

− підготувати вихідну документацію для взаємодії розробників програмного забезпечення.

Суть даної діаграми складається в наступному: проектоване програмне забезпечення представляється у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять програмним забезпеченням при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з програмним забезпеченням.

У системі є лише один вид користувача: користувач програмного забезпечення.

Користувач програмного забезпечення має функції (рис. 3.1):

- Завантаження файлу до системи
- Скачування потрібного файлу з системи
- Перегляд усіх файлів
- Пошук певного файлу
- Перегляд швидкості при скачуванні файлу

Рисунок 3.1 – Діаграма варіантів використання користувача

3.3.3 Вимоги до ПЗ

3.3.3.1 Функціональні вимоги

Функціональні вимоги до розроблювального програмного забезпечення такі.

Користувач додатку повинен мати доступ до наступного функціоналу:

- Завантаження файлу до системи
- Скачування потрібного файлу з системи
- Перегляд усіх файлів
- Пошук певного файлу
- Перегляд швидкості при скачуванні файлу

3.3.3.2 Нефункціональні вимоги

До нефункціональних вимог відносяться такі:

мінімальні системні вимоги для роботи програми на стороні клієнта наведені в таблиці 3.1;

Таблиця 3.1 – Конфігурація ПК для роботи програми на стороні клієнта

Операційна система	Windows 7 або вище
Процесор	Intel core i3 або краще
Об'єм оперативної пам'яті	4 ГБ
Графічний адаптер	ПЗ не використовує
Інтернет	20 мб/с

Форма побудови меню додатку на екрані:

- на головній сторінці повинен бути список з доступними файлами
- біля списку повинен бути “input” для пошуку певного файлу
- на головній сторінці повинна бути кнопка для завантаження файлу до системи
- після натискання на кнопку завантаження, повинно відкриватися діалогове вікно для вибору файлу
- при завантаженні файлу, на формі повинна відображатися швидкість завантаження.

3.3.4 Специфікація варіантів використання

Сценарії використання, варіанти використання або прецеденти - специфікація послідовностей дій (варіанти послідовностей і помилкові послідовності) які може здійснювати програмне забезпечення, система, підсистема або клас, взаємодіючі з зовнішніми дійовими особами.

Специфікація варіантів використання виконана у вигляді таблиць з описом прецедентів і сценаріїв і наведена в таблицях 3.2-3.4

Таблиця 3.2 – Сценарій завантаження файлу

Ім'я	File_upload
Назва	Завантаження файлу
Опис	Завантаження файлу до системи, где його потім можуть скачати інші користувачі
Передумова	Програмне забезпечення запущене
Постумова	Файл успішно появляється у списку
Основний потік	1 Користувач натискає на кнопку завантаження 2 Відкривається діалогове вікно 3 Користувач вибирає файл та натискає «окей» 4 Користувач отримує повідомлення про успішне завантаження
Альтернативний потік	1 Користувач натискає на кнопку завантаження 2 Відкривається діалогове вікно 3 Користувач закриває діалогове вікно 4 Користувач отримує повідомлення про те, що файл не був обран

Таблиця 3.3 – Сценарій скачування файлу

Ім'я	File_download
Назва	Скачування певного файлу
Опис	Скачування файлу, який є в системі
Передумова	Програмне забезпечення запущене
Постумова	Скачування файлу
Основний потік	1 Користувач вибирає зі списку файл 2 Користувач натискає кнопку завантаження

Таблиця 3.4 – Сценарій пошуку певного файлу

Ім'я	File_search
Назва	Пошук файлу
Опис	Пошук файлу для подальшого його скачування
Передумова	Програмне забезпечення запущене
Постумова	Потрібній файл є у списку
Основний потік	1 Користувач набирає у строчці пошуку певне ім’я 2 Користувач бачить у списку необхідний файл
Альтернативний потік	1 Користувач набирає у строчці пошуку певне ім’я 2 Користувач бачить пустий список, бо такого файлу нема

3.4 Розробка макетів екранних форм

На рисунку 3.3 надано макет головної сторінки.

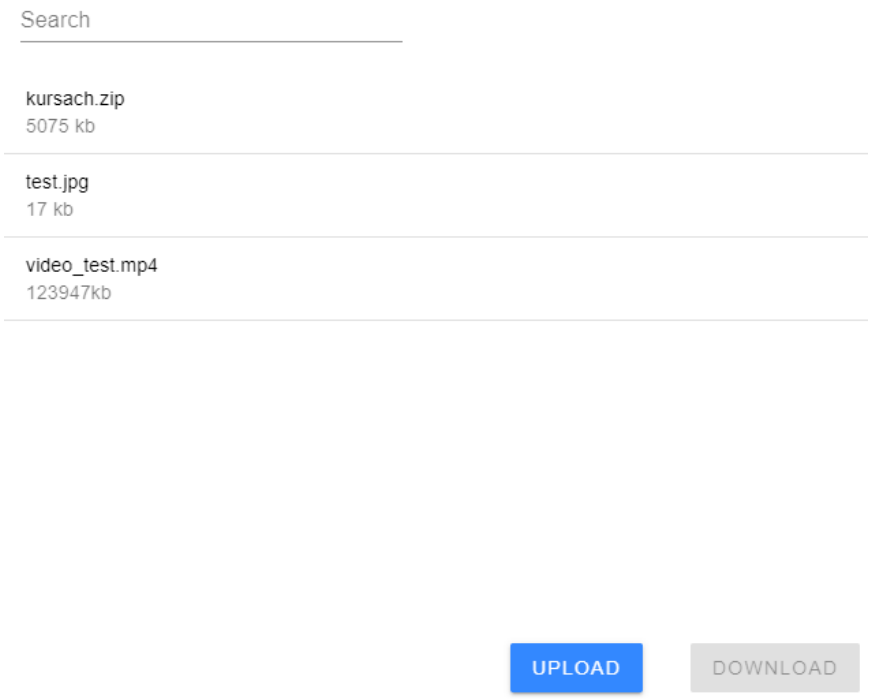


Рисунок 3.3 - Макет головної сторінки

На рисунку 3.4 надано макет пошуку файлу

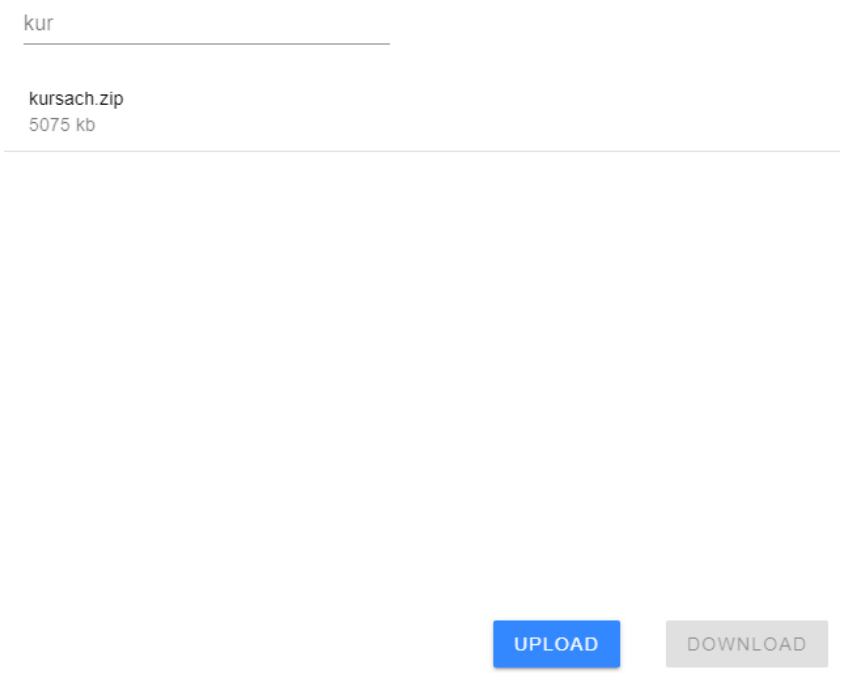


Рисунок 3.4 - макет пошуку файлу

На рисунку 3.5 надано макет вибору файлу зі списку

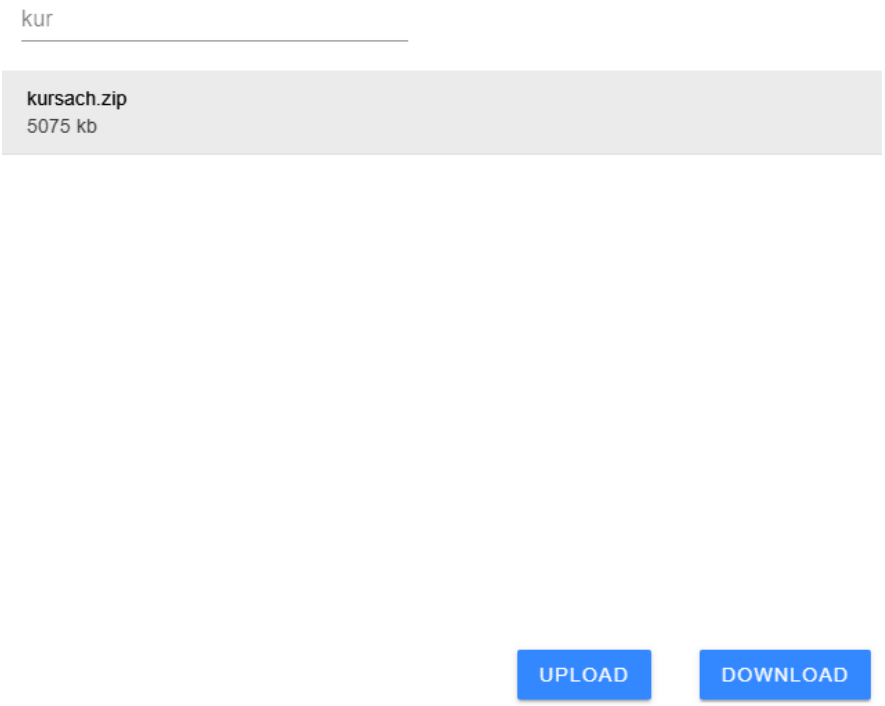


Рисунок 3.5 - Макет вибору файлу з списку

На рисунку 3.6 надано макет скачування файлу.

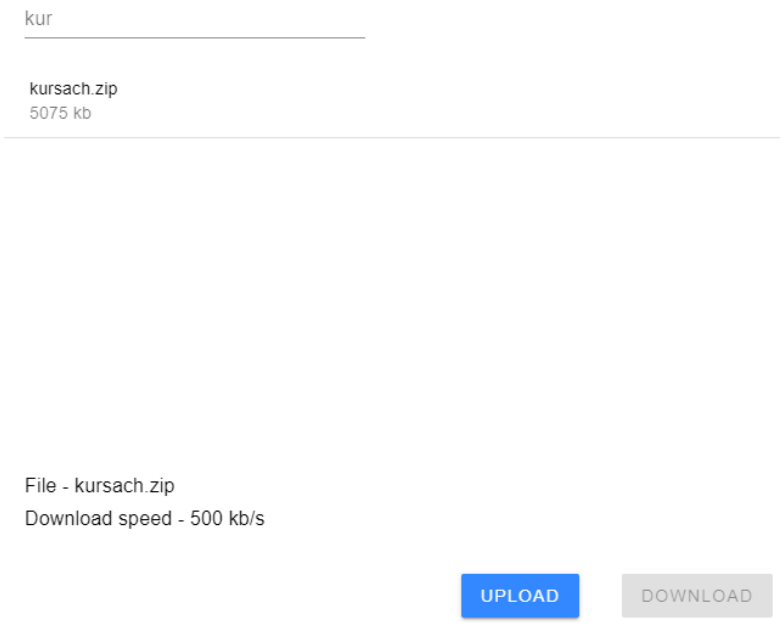


Рисунок 3.6 – Макет скачування файлу

3.5 Початкові дані

Для того, щоб забезпечити прозорий і простий інтерфейс взаємодії програмних модулів додатку, які були виділені в попередньому розділі, необхідно забезпечити поділ функціональних можливостей між модулями.

Оскільки рішення задачі вимагає розробки інтерактивного інтерфейсу для взаємодії з користувачем, клієнтської частини додатку знадобляться потужні інструменти для відображення інтерфейсу.

Відповідно, для того, щоб ізолювати цю функціональність в одному модулі, необхідно сконцентрувати всі дії по формуванню відображення інтерфейсу в клієнтському додатку. Односторінковий додаток забезпечує винос здебільшого логіки в модуль клієнтського додатку: обробка даних для відображення, відображення, обробка користувальницьких подій, відправка даних на сервер.

Таким чином, інтерфейс для клієнтського модуля представляє із себе методи по прийому й відправленню необроблених даних без відображення.

Для вирішення завдань клієнтського додатка було обрано такі технології:

– .NetCore - це горизонтальний розвиток програмної платформи .NET в інші операційні системи, в яких вона забезпечує можливість використання додатків, розроблених для Windows[26];

Основними завданнями серверного додатку при реалізації підходу односторінкового додатку є:

– процедура обробки запитів від клієнтської частини додатку і виконання бізнес-логіки додатка;

– формування сторінки для першого користувальницького запиту.

Для вирішення завдань серверного додатку було обрано такі технології:

- .NetCore
- Azure Function
- Azure WebApp
- Azure CosmosDb

Взаємодію бази даних і сервера слід організувати, дотримуючись принципу, що бізнес логіка знаходиться в коді серверного додатку, а не в базі даних. В цьому випадку база даних зберігає дані, і надає прямий доступ до даних, тоді як вся бізнес-логіка реалізована в підсистемах серверного додатка[27]. База даних дозволяє виконувати транзакції для проведення атомарних операцій над даними.

Ізоляція бізнес-логіки в підсистемах серверного додатку дозволяє забезпечити прозорий інтерфейс взаємодії сервера з базою даних.

Для забезпечення такої ізоляції, можна визначити наступну ієрархічну структуру завдань:

– розробка модуля клієнтської частини веб-додатку;
– розробка модуля серверної частини веб-додатку;

3.6 Архітектура додатку

Peer-to-peer - це архітектура передачі даних, заснована на ідеї рівноправності всіх учасників мережі. У мережі відсутні виділені сервери. На відміну від традиційних мереж, кожен з учасників є як клієнтом, так і сервером. Основною перевагою такої мережі є практично повна незалежність працездатності мережі від її розміру. У той час, як побудувати і підтримувати сервер на сотні тисяч користувачів є непростим завданням, peer-to-peer мережі відмінно з цим справляються[28].

Ідея peer-to-peer спілкування полягає в тому, що кожен peer знає і підтримує інформацію про інших учасників. Коли новий клієнт підключається до мережі, він може дізнатися у будь-якого бенкету інформацію про те, де і які файли зараз доступні. Коли клієнт починає викачує файл собі на комп'ютер, то викачані частини цей файлу відразу стають доступні для скачування іншим користувачам. Ніхто не дає гарантію, що кожен сервер буде знаходитися тривалий час в мережі і давати скачувати інформацію, навпаки - ситуація, коли сервер пропадає в процесі завантаження, є природною. В даному випадку буде знайдений новий сервер, який може продовжити передачу даних. Для підтримки списку активних peer-ів кожен сервер посилає іншим серверам heartbeat[29]. Heartbeat - це повідомлення, яке один сервер посилає іншому, щоб сказати йому, що він живий. Відповідно, якщо heartbeat довго не приходить, значить цей сервер потрібно видалити зі списку активних peer-ів. Постійно обмінюватися heartbeat-ами з великою кількістю серверів є трудомісткою оперцією. Тому у кожного сервера є два параметри - нижня і верхня межа на розмір списку активних серверів. Коли ця кількість стає нижче нижньої межі, запускається пошук нових учасників. Сервер запитує у інших серверів список активних peer-ів і додає деяких з них в свій список, але при цьому стежить за тим, щоб розмір списку не перевищив верхню межу.

3.7 Висновки з розділу 3

В даному розділі проведено аналіз, вибір і обґрунтування технологій розробки програмного забезпечення.

Були розглянуті вимоги до ПЗ, в результаті розгляду вимог було описано специфікації варіантів використання. Для реалізації ПЗ було обрано peer-to-peer архітектуру та мову програмування C#. В результаті розробки прототипу було спроектовано та реалізовано серверну та клієнтську частину програмного забезпечення. В зв‘язку з тенденцією збільшення навантаження на клієнтську частину і створення нових інструментів, у розділі було розглянуто та проаналізовано необхідний набір засобів[30].

ВИСНОВКИ

В першому розділі був зроблен аналіз проблемної області. Були переглянуті основні можливості блокчейну для захисту файлів, та користувача у системі в цілому. Переглянуто можливість використання хмарного сховища для таких цілей. Також були переглянуті аналоги створюваного програмного забезпечення.

В другому розділі були оглянуті усі основні принципи роботи таких технологій як блокчейн та BitTorrent. Описали методи, які будуть використовуватись при розробці майбутнього ПЗ.

Також оглянули нові технології, які зараз не користуються популярністю, але можуть нам дуже допомогти при розробці ПЗ.

В третьому розділі проведено аналіз, вибір і обґрунтування технологій розробки програмного засобу.

Були розглянуті вимоги до ПЗ, в результаті розгляду вимог було описано специфікації варіантів використання. Для реалізації ПЗ було обрано peer-to-peer архітектуру та мову програмування C#. В результаті розробки прототипу було спроектовано та реалізовано серверну та клієнтську частину програмного забезпечення. В зв‘язку з тенденцією збільшення навантаження на клієнтську частину і створення нових інструментів, у розділі було розглянуто та проаналізовано необхідний набір засобів.

ПЕРЕЛІК ПОСИЛАНЬ

1. BlockChain [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/b/blockchain.asp.
2. PoS [Електронний ресурс] – Режим доступу до ресурсу: http://www.marketch.ru/marketing_dictionary/marketing_terms_p/point_of_sale/.
3. NiceHash [Електронний ресурс] – Режим доступу до ресурсу: https://www.nicehash.com/.
4. Атака 51% [Електронний ресурс] – Режим доступу до ресурсу: https://academy.binance.com/ru/articles/what-is-a-51-percent-attack.
5. Proof-of-work [Електронний ресурс] – Режим доступу до ресурсу: https://forklog.com/chto-takoe-proof-of-work-i-proof-of-stake/.
6. Ethereum – глобальна платформа [Електронний ресурс] – Режим доступу до ресурсу: https://ethereum.org/ru/.
7. Хмарне сховище даних [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cloud_storage.
8. Storj [Електронний ресурс] – Режим доступу до ресурсу: https://storj.io/.
9. Mediacoin [Електронний ресурс] – Режим доступу до ресурсу: https://mediacoin.pro/ru/.
10. IPFS [Електронний ресурс] – Режим доступу до ресурсу: https://ipfs.io/.
11. Axel [Електронний ресурс] – Режим доступу до ресурсу: https://www.axel.org/.
12. Peer-to-peer [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Peer-to-peer.
13. Elliptic Curve Digital Signature Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm.
14. Хешування [Електронний ресурс] – Режим доступу до ресурсу: https://ssl.com.ua/blog/hashing-coding-encryption/.
15. Стійкість до колізії [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Collision_(computer_science).
16. Merkle Tree [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/455260/.
17. Фільтр Блума [Електронний ресурс] – Режим доступу до ресурсу: https://otus.ru/nest/post/972/.
18. BitTorrent [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/BitTorrent.
19. Transmission Control Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://networkguru.ru/protokol-transportnogo-urovnia-tcp-chto-nuzhno-znat/.
20. Схема Шнорра [Електронний ресурс] – Режим доступу до ресурсу: https://forklog.com/chto-takoe-podpisi-shnorra-chto-takoe-taproot/.
21. Tracker [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tracker.
22. BitTorrent Peer Protocol [Електронний ресурс] – Режим доступу до

ресурсу: https://wiki.theory.org/BitTorrentSpecification.

23. Algorithm choke [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/figure/Implementation-of-the-choking-algorithm_fig3_223808116.

24. Моделі управління блокчейном [Електронний ресурс] – Режим доступу до ресурсу: https://cryptor.net/kriptovalyuty/modeli-upravleniya-v-blokcheyn-protokolah.

25. UML [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/458680/.

26. .Net Core [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/aspnet5/1.1.php.

27. Azure [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/hub/azure/.

28. Пірінгові мережі [Електронний ресурс] – Режим доступу до ресурсу: http://artismedia.by/blog/chto-takoe-peer-to-peer/.

29. Heartbeat [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Heartbeat_(computing).

30. C Sharp [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/C_Sharp.

ДОДАТОК А

Слайди презентації

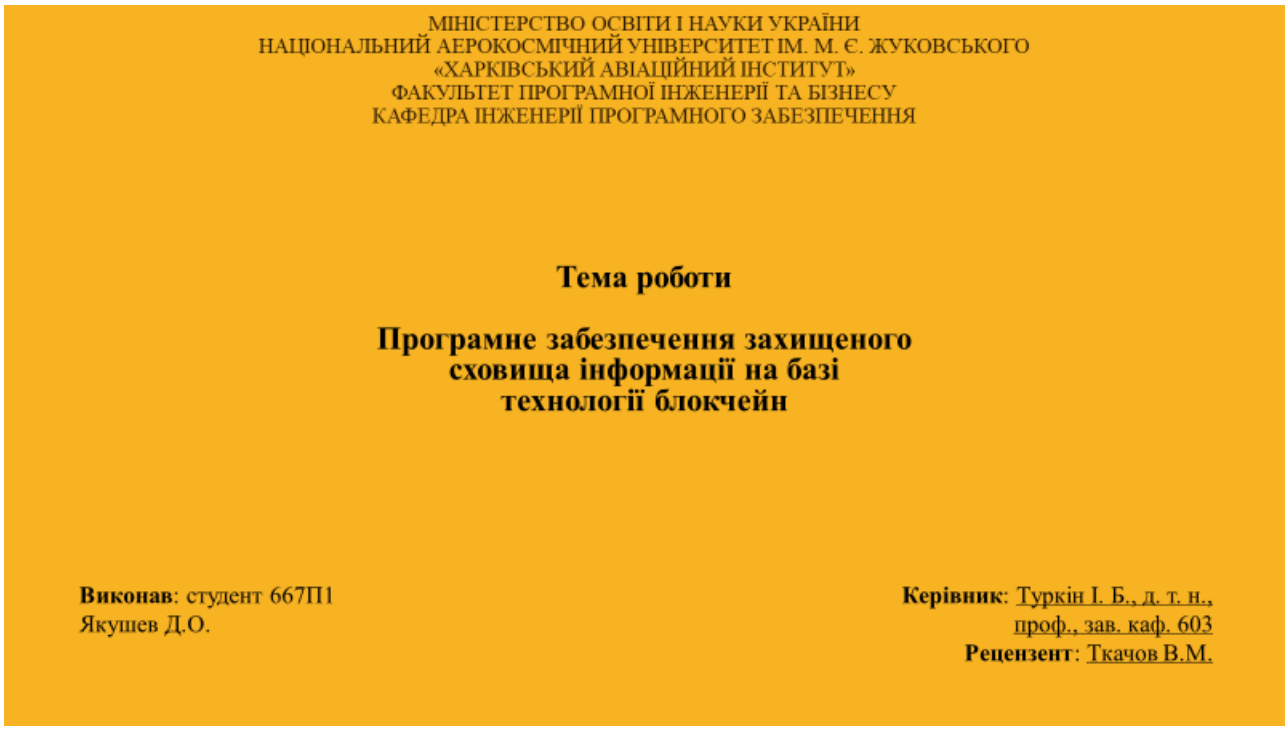


Рисунок А.1 – Титульний слайд

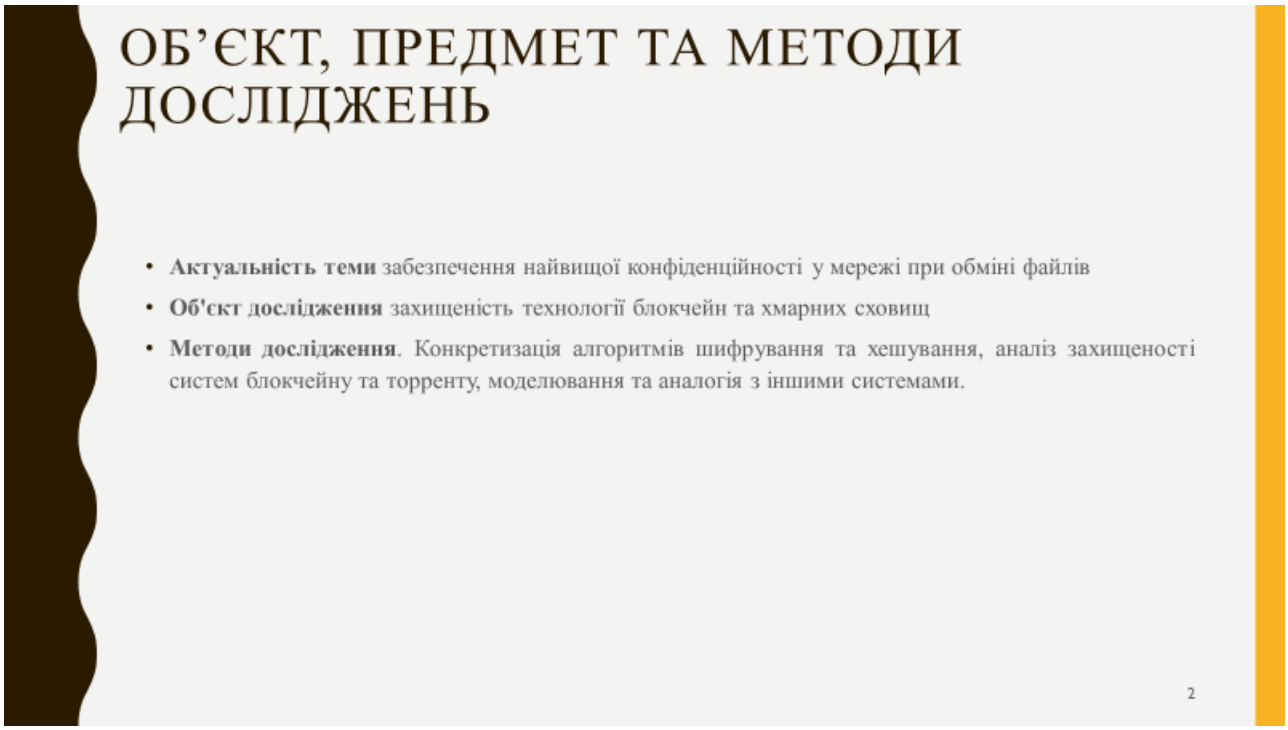


Рисунок А.2 – Об’єкт, предмет та методи досліджень

Рисунок А.3 – Мета роботи

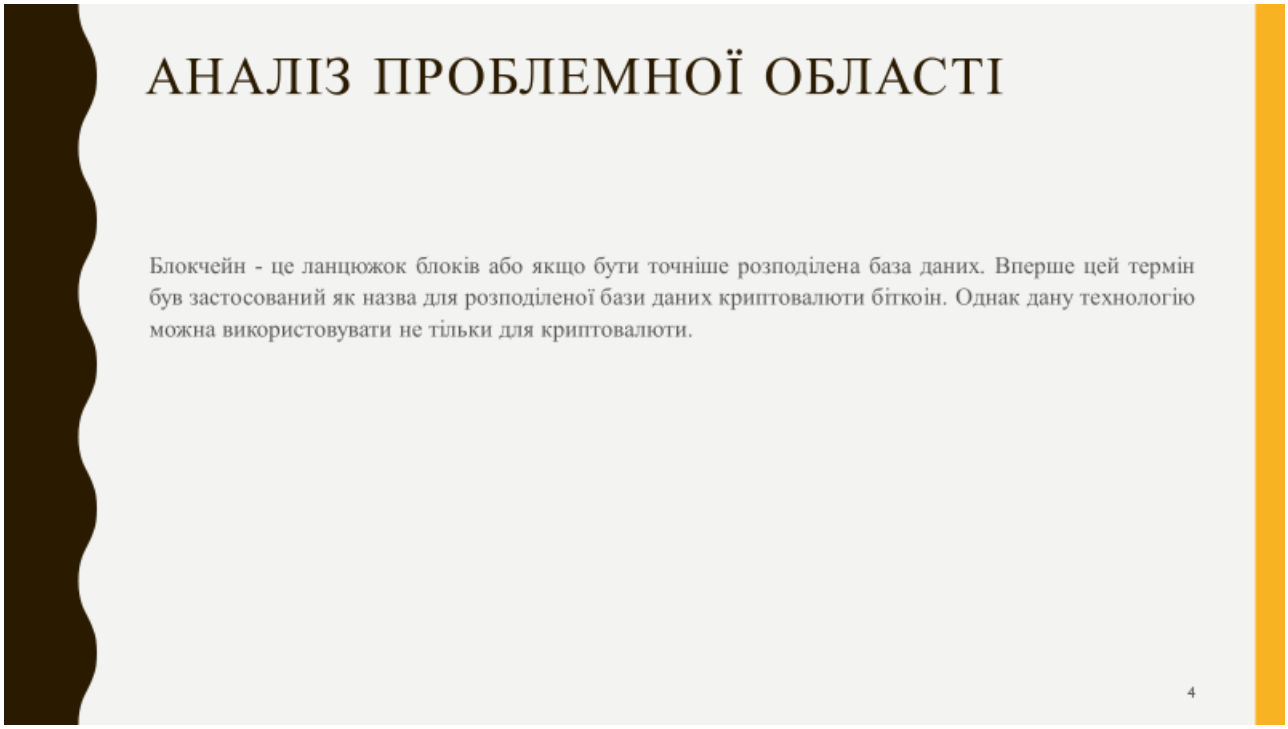


Рисунок А.4 – Аналіз проблемної області

АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

На даний момент головною загрозою для блокчейна, щодо гіпотетичної, є «атака 51%», коли зловмисник може зробити відкат транзакцій, друкуючи альтернативні блоки на бічному ланцюжку (гілці) і гарантовано спростовуючи те, що відбувається в основний ланцюжку блокчейна. По суті, це схоже на «човниковий біг». Однак з огляду на ресурсомісткість рішення хеш-функції і емісії нових біткоінів, поки що такий варіант здається малоймовірним.

5

Рисунок А.5 – Аналіз проблемної області

АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

Ми можемо порахувати вірогідність досягнення беззбитковості або те ж саме, що атакуючий обжене чесних будівельників ланцюжка

p - ймовірність що чесний хост знайде наступний блок;

q - ймовірність, що атакуючий знайде наступний блок;

q_z - ймовірність, що атакуючий виграє гонку якщо він відстав на z блоків.

$$q_z = \begin{cases} 1, & p \leq q \\ (q/p)^z, & p > q \end{cases}$$

Припустимо, що p> q, тоді ймовірність експоненціально зменшується при збільшенні числа блоків, на яке відстав атакуючий. Таким чином, якщо йому не вдасться вирватися вперед на самому початку, то його шанси перемогти в подальшому стають зникаюче малі.

6

Рисунок А.6 – Аналіз проблемної області

МОДЕЛІ ТА МЕТОДИ

Блокчейн - це однорангова мережа. Однорангова мережа відноситься до неієрархічної мережі, свого роду архітектури в межах інформатики, де участь та завдання поділяються рівними між рівнями. Що стосується інформатики, кожен системний компонент називається комп'ютерним вузлом. Вузол являє собою або пристрої, або точки даних. Комп'ютер діє як вузол, оскільки має IP-адресу, але також кожне посилання, на яке натискається, наприклад, на веб-сторінці компанії, оскільки воно є частиною більшої структури даних. Однорангова мережа - це приклад вузлів, що діють в децентралізованій архітектурі програмного забезпечення. Децентралізована архітектура системи - це система, де влада або відповідальність розподіляються на кожен окремий вузол. Навпаки централізована системна архітектура - де функції здійснюються через центральний елемент

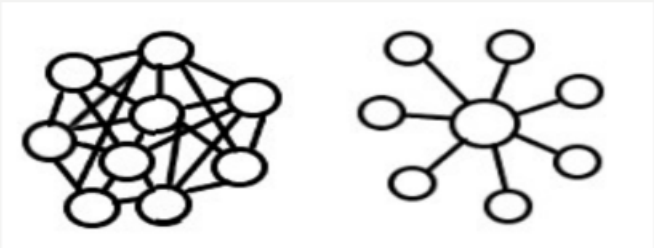


Рисунок А.7 – Моделі та методи

МОДЕЛІ ТА МЕТОДИ

Більшість систем криптовалют або систем, заснованих на технології блокчейн, зараз використовують алгоритм ECDSA.

На відміну від RSA, ECDSA базується на набагато більш складних математичних обчисленнях. А використовується ECDSA тільки для цифрових підписів. Вперше використання алгебраїчних властивостей еліптичних кривих в криптографії було запропоновано в 1985 році. ECDSA як стандарт цифрового підпису почав застосовуватися з кінця 1990-х років.

Рисунок А.8 – Моделі та методи

МОДЕЛІ ТА МЕТОДИ

Хешування відноситься до процесу створення певного висновку з вхідних даних різного розміру. Це робиться за допомогою математичних формул, також відомих як хеш-функції (реалізовані у вигляді алгоритмів хешування).

Не всі хеш-функції припускають використання криптографії, а тільки ті, які спеціально призначені для цього, так звані криптографічні хеш-функції, які лежать в основі криптовалюти. Завдяки їх роботі, блокчейни і інші розподілені системи здатні досягти високого рівня цілісності даних і безпеки.

Як звичайні, так і криптографічні хеш-функції є детермінованими. Бути детермінованим означає, що до тих пір, поки вхідні дані не змінюються, алгоритм хешування завжди буде видавати один і той же результат (також відомий як дайджест або хеш).

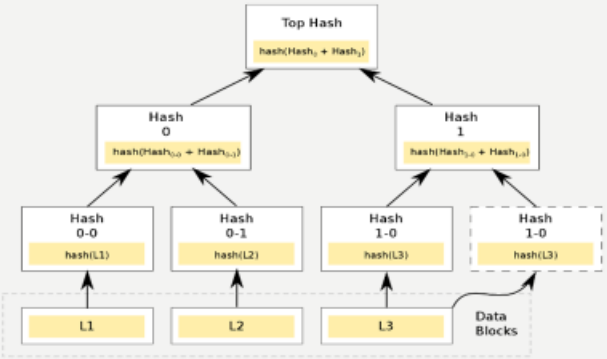
9

Рисунок А.9 – Моделі та методи

МОДЕЛІ ТА МЕТОДИ

Вузли в блокчейн-мережі анонімні і працюють в умовах відсутності довіри. У цій ситуації постає проблема верифікації даних: як перевірити, що в блоці записані коректні транзакції? Для оцінки кожного блоку знадобиться багато часу і обчислювальних ресурсів. Вирішити проблему і спростити процес допомагають дерева Меркла

Дерево Меркла, або хеш-дерево, - це двійкове дерево, кінцеві вузли якого - це хеші транзакцій, а внутрішні вершини - результати складання значень пов'язаних вершин. Ось приклад хеш-дерева з трьома транзакціями



10

Рисунок А.10 – Моделі та методи

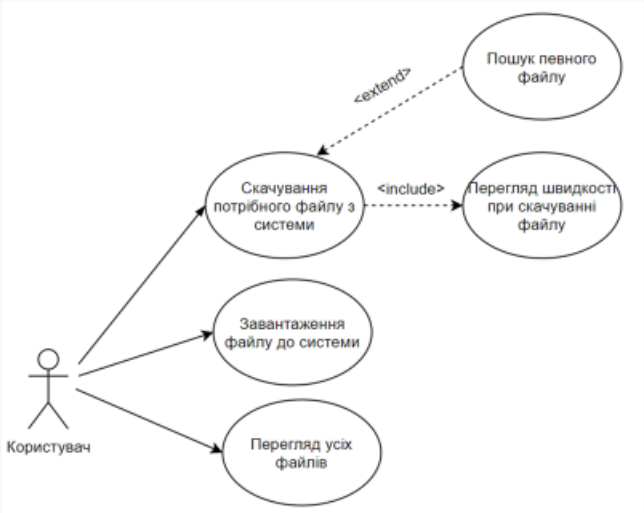
МОДЕЛІ ТА МЕТОДИ

BitTorrent - пірінговий (Peer-To-Peer) мережевий протокол для кооперативного обміну файлами через Інтернет.
Файли передаються частинами, кожен torrent-клієнт, отримуючи (викачувавши) ці частини, в той же час віддає (закачує) їх іншим клієнтам, що знижує навантаження і залежність від кожного клієнта-джерела і забезпечує надмірність даних

11

Рисунок А.11 – Моделі та методи

РОЗРОБКА ПЗ



12

Рисунок А.12 – Розробка ПЗ

Рисунок А.13 – Розробка ПЗ

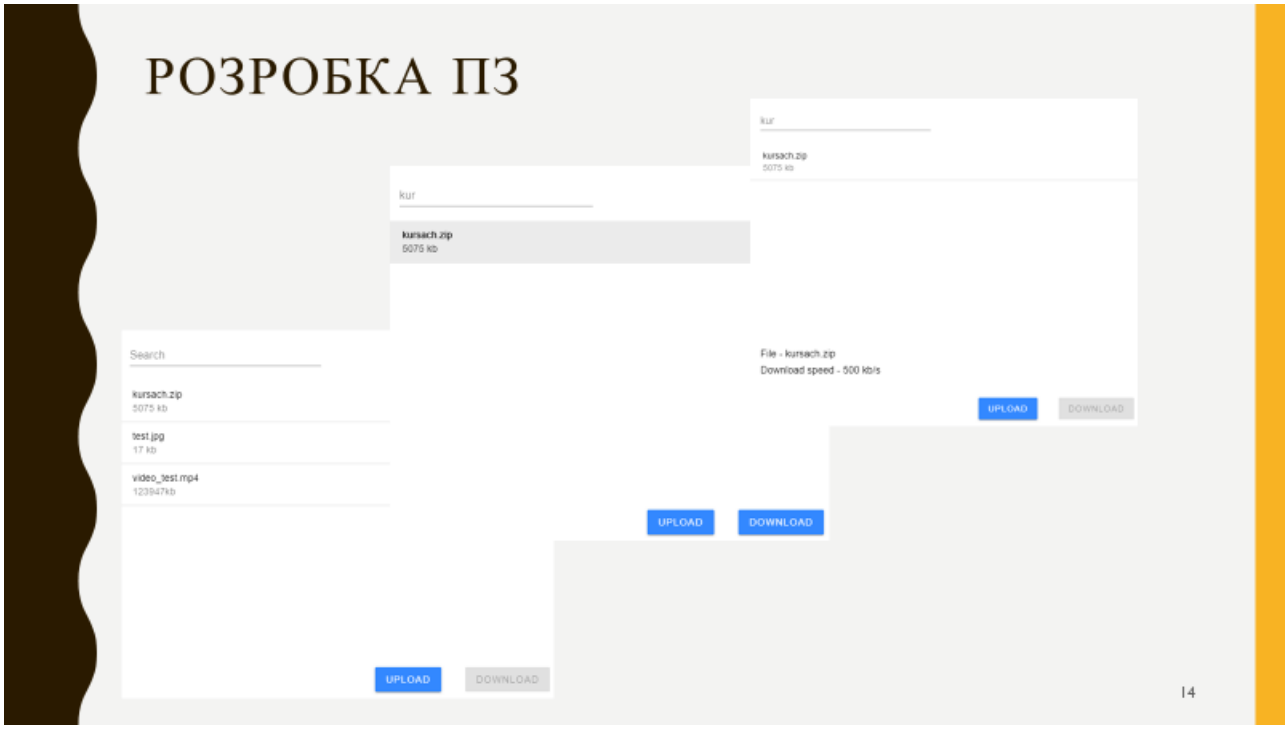


Рисунок А.14 – Розробка ПЗ

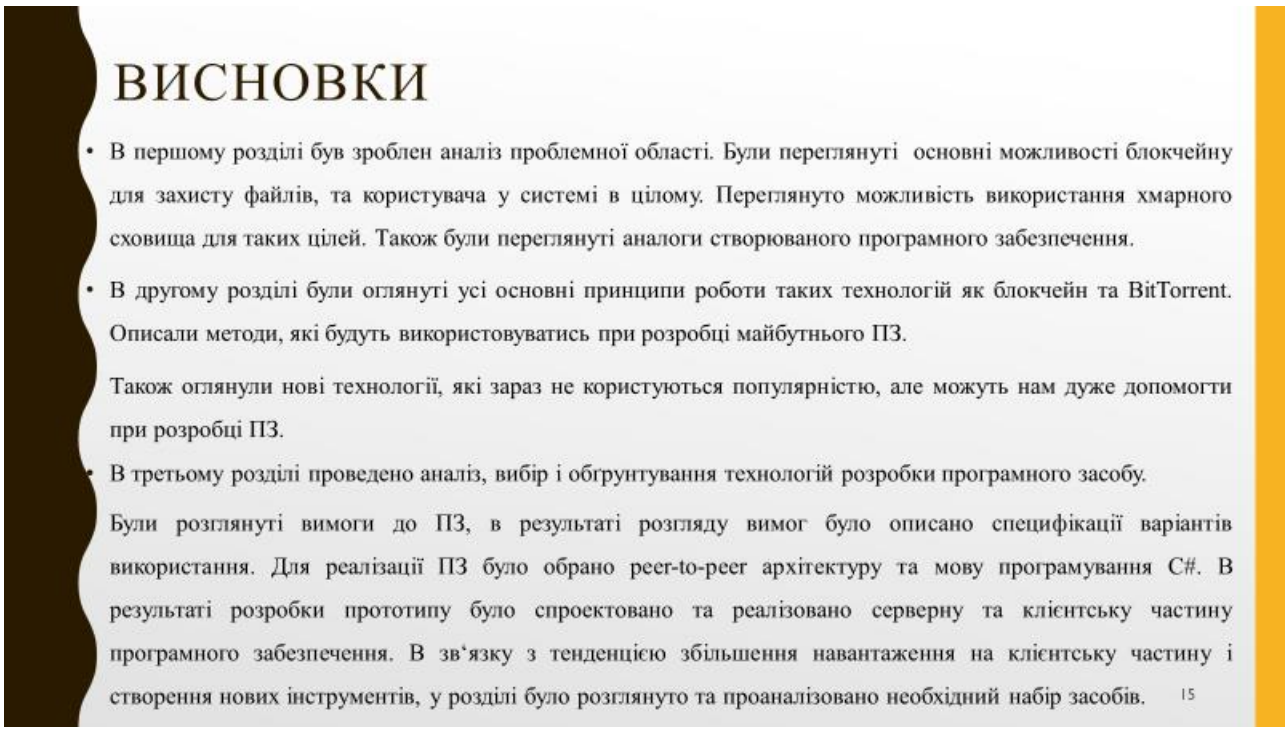


Рисунок А.15 – Висновки