Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК _____

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)
“___”_____2018_р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка,
(код і назва)

спеціалізація Апаратно-програмні засоби електронних комунікацій

на тему: Побудова VPN мереж на базі технології Blockchain

Виконала: студентка 2 курсу, групи ТЗ-71мп
(шифр групи)

Кукліна Анна Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник
професор кафедри телекомунікацій, д.т.н. Романов О.І. _____
(прізвище, ім’я, по батькові) (підпис)

Рецензент
доцент кафедри інформаційно телекомунікаційних систем, к.т.н._____
(посада, науковий ступінь, вчене звання, науковий ступінь)

Скулиш М.А. _____
(прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018_ рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Спеціалізація Апаратно-програмні засоби електронних комунікацій

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____Явіся В.С.
(підпис) (ініціали, прізвище)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Куклініій Анні Сергіївні

1. Тема дисертації : Побудова VPN мереж на базі технології Blockchain
науковий керівник дисертації Романов О.І., д.т.н., професор ,
затверджені наказом по університету від «_06_» «_11_» 2018р. № _4095-с_
2. Строк подання студентом дисертації 11 грудня 2018 р.
3. Об’єкт дослідження: технологія Blockchain, як засіб розгортання VPN мережі
4. Предмет дослідження є принцип побудови VPN мережі на базі технології Blockchain.
5. Перелік завдань, які потрібно розробити:
   1) Визначення компонентів роботи технології Blockchain, аналіз можливості реалізації захищеної комунікації вузлів мережі.
   2) Розробка методу реалізації мережі VPN на базі технології Blockchain та його опис.
   3) Розробка методичних матеріалів, покрокового опису реалізацію макету мережі VPN на базі технології Blockchain.
6. Орієнтовний перелік ілюстративного матеріалу _____
7. Орієнтовний перелік публікацій _____

1) Аналіз методів побудови корпоративних мереж на основі VPN-технологій Нестеренко М.М., Саєнко Б.В., Кукліна А.С. Інститут телекомунікаційних систем НТУУ «КПІ ім. Ігоря Сікорського»
2) Использование механизма Explicit Congestion Notification (ECN) для защиты сетей от перегрузки Куклина А. С.

9. Дата видачі завдання 1 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Написання першого розділу, постановка завдання на магістерську дисертацію.	20 вересня 2018р.	
2	Розгляд технології Blockchain, визначення її складових.	1 жовтня 2018р.	
3	Аналіз існуючих рішень на базі технології Blockchain.	15 жовтня 2018р.	
4	Опис складових, на яких базується Blockchain. Другий розділ.	15 листопада 2018р.	
5	Огляд програм, для застосування в створенні тестового оточення.	30 листопада 2018р.	
6	Створення тестового оточення для передачі даних у локальній Blockchain мережі.	1 грудня 2018р.	
7	Оформлення роботи. Підготовка ілюстративного матеріалу та доповіді.	5 грудня 2018р.	

Студент _____ _____ _Кукліна А.С._
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____Романов О.І.___
(підпис) (ініціали, прізвище)

Пояснювальна записка
до магістерської дисертації

на тему: Побудова VPN мереж на базі технології Blockchain

Київ – 2018 року

УДК 004.056.5

РЕФЕРАТ

Дипломна робота містить 106 с., 1 табл., 47 рис., 14 джерел.

ВІРТУАЛЬНА ПРИВАТНА МЕРЕЖА, ТОПОЛОГІЯ VPN, ХЕШ-ФУНКЦІЯ, ХЕШ-ТАБЛИЦЯ, СМАРТ-КОНТРАКТ, ОБМІН ДАНИМИ, ПРИВАТНІСТЬ ІНФОРМАЦІЇ, ТРАНЗАКЦІЯ, АСИМЕТРИЧНІ КРИПТОСИСТЕМИ, ETHEREUM, BLOCKCHAIN

Об’єктом дослідження є технологія Blockchain, як засіб розгортання VPN мережі.

Мета роботи: формування принципів роботи VPN мережі на базі Blockchain, опис процесів, що мають відбуватися при передачі даних та реалізація макету, що демонструватиме передачу даних. В ході виконання даної дипломної роботи проведено аналіз існуючих топологій побудови VPN мереж. Досліджено принцип роботи технології Blockchain та її складових. В результаті, сформовано перелік складових, та кроків, що формують принципи роботи VPN мережі на базі технології Blockchain. Результати можуть бути використані для подальшого розгортання мережі.

ABSTRACT

The work contains 106 pages, 1 tables, 47 figures, 14 sources.

VIRTUAL PRIVATE NETWORK, VPN TOPOLOGY, HASH FUNCTION, HASH TABLE, SMART CONTRACT, DATA EXCHANGE, INFORMATION PRIVACY, TRANSACTION, ASSYMETRIC CRYPTOGRAPHY, ETHEREUM, BLOCKCHAIN

The object of the study become Blockchain technology as a means of deploying a VPN network.

The purpose of the work: the formation of the principles of the VPN network based on Blockchain, description of the processes to occur in the transmission of data and implementation of a layout that demonstrates the transmission of data. During the execution of this thesis, an analysis of existing topologies for constructing VPN networks were conducted. The principle of Blockchain technology and its components were investigated. As a result, a list of components and steps that form the principles of VPN network operation based on Blockchain technology has been generated. The results could be used to further deploy the network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ABI – Application Binary Interface (Двійковий інтерфейс програми)

AH – Authentication Header (Заголовок автентифікації)

ATM – Asynchronous transfer mode (Асинхронний режим передачі даних)

CLI – Command-line interface (Інтерфейс командного рядка)

CPE – Customer premises equipment (Абонентське обладнання)

EIP – Ethereum Improvement Proposal (Пропозиція з покращення Ethereum)

ESP – Encapsulating Security Payload (шифрування даних)

ETH – Ether (Ethereum токен)

EVM – Ethereum Virtual Machine (Віртуальна машина Ethereum)

GRE – Generic Routing Encapsulation (загальна інкапсуляція маршрутів)

GUI – Graphical User Interface (графічний інтерфейс користувача)

IKE – Internet Key Exchange (обмін ключами доступу для SA)

IP – Internet Protocol address (Інтернет протокол)

IPFS – InterPlanetary File System (Міжпланетна файлова система)

IPO – Initial Public Offering (‘Первинне розміщення монет’)

IPSec – Internet Protocol Security (Безпека інтернет протоколу)

ISDN – Integrated Services Digital Network (цифрова мережа з інтегрованими службами)

L2TP – Layer 2 Tunneling Protocol (протокол тунелювання 2 рівня)

MD – Message digest (“представлення повідомлення”)

MPLS – Multiprotocol Label Switching (комутація з мітками)

NAT – Network Address Translation (перетворення мережних адрес)

P2P – Point-to-point (точка-точка)

POC – Proof Of Concept (Доказ концепції)

PoW – Proof Of Work (Доказ роботи)

PPTP – Point-to-Point Tunnelling Protocol (тунельний протокол типу P2P)

RPC – Remote procedure call (Виклик віддаленої процедури)

RSA – Rivest, Shamir та Adleman (криптографічний алгоритм)

SA – Security Association (асоціація безпеки)

SHA – Secure Hash Algorithm (“безпечний алгоритм хешування”)

TCP – Transmission Control Protocol (протокол керування передачею)

TOR – The Onion Router (‘Цибулева маршрутизація’)

UDP – User Datagram Protocol (Протокол датаграм користувача)

URL – Uniform Resource Locator (Уніфікований локатор ресурсів)

UTC – Coordinated Universal Time (Всесвітній координований час)

VC – Virtual Channel (віртуальний канал)

VPN – Virtual Private Network (Віртуальна приватна мережа)

БД – База даних

ОС – Операційна система

ПЗ – Програмне забезпечення

ПК – Персональний комп’ютер

СВК – Система з відкритим ключем

ВСТУП

Актуальність. Virtual Private Network (VPN; укр. Віртуальні приватні мережі) – програмно-апаратне рішення метою якого є відділення певної частини трафіку від всього, що транслюється в мережі по якій здійснюється передача.

Сучасні корпоративні мережі використовують глобальну мережу для комунікації віддалених сайтів корпорації, співпраці з компаніями постачальниками та партнерами. Користувачі мережі Інтернет мають потребу в захищеному з'єднанні, очікують на безпечну передачу інформації, а, іноді, потребують доступ до обмежених ресурсів. VPN технологія – це рішення, що задовольняє потреби сучасних користувачів мережі Інтернет. Зазвичай, VPN будується за централізованою схемою, а саме: існує VPN сервер на якому проходять процеси шифрування та фільтрації трафіку; до сервера підключаються користувачі зі встановленим програмним забезпеченням, або знаходячись в налаштованій підмережі (наприклад сайти компаній).

Відповідно до потреб користувачів може бути обрано рішення, що реалізує потреби з врахуванням доступності обладнання, вартості реалізації та конфігурації мережі.

У даній роботі планується розглянути реалізацію захищеного з'єднання на базі технології яка добре себе зарекомендувала у сфері криптовалют – Blockchain. В її основі лежать алгоритми, що дозволяють запобігти підміні інформації та несанкціонованому доступу. Ці особливості Blockchain можуть створити нові принципи взаємодії з інформацією та дати розвиток новому поколінню VPN мереж.

Мета роботи.

Для досягнення мети роботи було поставлено та вирішено такі задачі:

1) Виконання аналізу існуючих класифікацій та топологій віртуальних приватних мереж;
2) Вивчення основних компонентів технології Blockchain;

3) Дослідження роботи мережі Blockchain, принципів передачі, обробки та зберігання даних;
4) Здійснити підбір програмного забезпечення, що допоможе реалізувати тестову мережу Blockchain;
5) Створити тестове оточення, у якому буде захищено передано дані з подальшим записом та зберіганням у мережі Blockchain;
6) Розробити покрокові методичні матеріали, що допоможуть у подальшому впровадженні та розвитку безпечної передачі інформації у мережі Blockchain VPN.

1. ЗАВДАННЯ І ЦІЛІ ВИКОРИСТАННЯ ВІРТУАЛЬНИХ ПРИВАТНИХ МЕРЕЖ. КЛАСИФІКАЦІЯ VPN. ПОСТАНОВКА РОБОЧОГО ЗАВДАННЯ З УРАХУВАННЯМ ІСНУЮЧИХ ВИМОГ

Метою даного розділу є аналіз актуальності використання віртуальних приватних мереж та основних сфер їх призначення. В рамках даного розділу сформулюємо визначення основної мети та завдання мереж VPN у сучасному світі інформаційних технологій. Створення актуальної класифікації мереж та їх цілей має стати гарним підґрунтям знань, що необхідні для повноцінного аналізу функціональності та вимог до сучасних рішень у даній сфері.

1.1 Цілі і завдання мереж VPN

Сьогодні діяльність сучасних компаній пов’язана з активністю в мережі Інтернет. Передача даних, робота віддалених сайтів компанії, фінансові операції та укладення угод, збереження даних та робота з корпоративними даними все обумовлює необхідність постійної роботи з мережею Інтернет. Безпека є необхідним фактором для успішної активності компанії. Витік інформації здатний спричинити ряд наслідків, що можуть катастрофічно вплинути на компанію: від зменшення прибутку до втрати репутації та ліквідації бізнесу.

Віртуальні приватні мережі (VPN , вже давно розглядаються як сервіс-орієнтовані мережі. Вони використовуються для безпечного зберігання даних корпоративних мереж у випадках, коли співробітники працюють віддалено. Вони також використовуються окремими особами, які бажають уникнути інтернет-цензури і отримати доступ до регіонально обмеженого контенту. VPN набувають все більшої популярності як інструмент підвищення безпеки при підключенні до Інтернету. Також використання VPN обумовлене тим, що принциповим фактором реалізації мережі є вартість робіт для побудови та вартість утримання корпоративних мереж. На рис. 1.1 зображено загальну

схему компонентів мережі VPN: віддалені сайти, що підключені до головного, віддалені користувачі, що також мають доступ до корпоративної мережі.

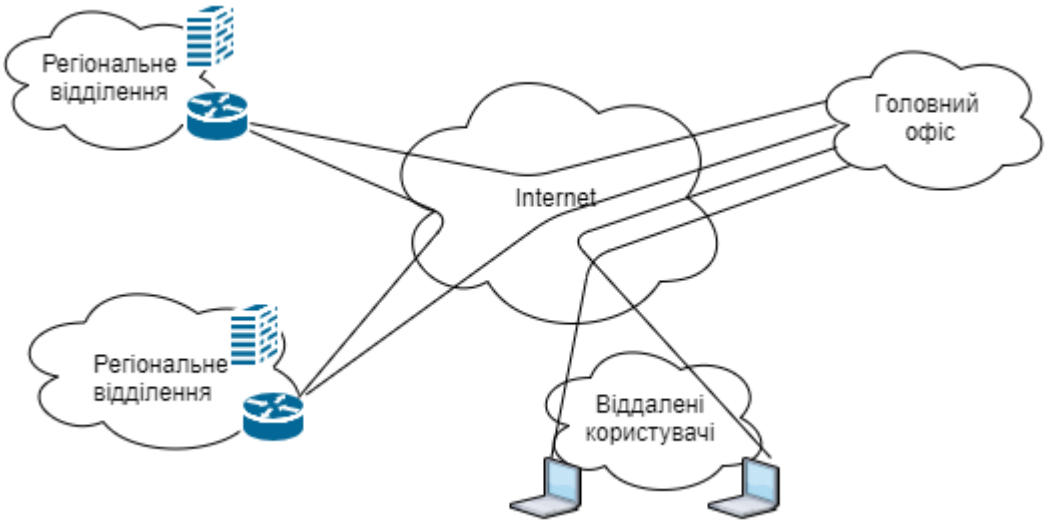


Рис. 1.1 Загальна схема VPN мережі

Різниця у вартості послуг по об'єднанні віддалених сайтів мереж, наприклад Frame Relay та мережі на базі Internet є доволі суттєвою. Проте, з'єднання двох точок у мережі Internet повинно мати відповідний рівень безпеки, при передачі даних. На сьогоднішній час існує значна кількість механізмів та рішень, що забезпечують конфіденційність і цілісність інформації, що передається мережею.

Однією з важливих сфер, потребуючих такого захищеного каналу є бізнес процеси та комунікація з замовниками, доступ до даних у мережі, перебуваючи за межами офісу, чи країни.

Конфігурація VPN для корпоративної мережі покликана забезпечити збереження цілісності та безпеку даних, тобто захист інформації.

Під поняттям захист інформації, в сучасному світі, розуміється сукупність дій та заходів, що покликані забезпечити конфіденційність та цілісність даних у процесах збору, передачі, обробки та зберігання.

Цілями захисту інформації є:

- Запобігання витоку, розкрадання та несанкціонованого поширення, спотворення, підробки.
- Запобігання загрозам безпеки особистості, суспільства, держави.
- Запобігання протиправним діям зі знищення, зміни, копіювання, попередження інших форм незаконного проникнення в інформаційні системи, забезпечення правового режиму як об'єкта власності. [1]

Персональні дані є найбільш привабливою метою крадіжок. Для їх здобуття використовують спам розсилки, фішинг, прослуховування трафіку.

Наступним об’єктом злочинів у сфері інформаційної безпеки є особливості та умови проведених фінансових операцій та угод. Внутрішні фінансові звітності підприємства можуть бути викрадені та використані для виведення компанії з ринку. Недоліки в захисті інформації призводять до передачі бізнес-планів, інтелектуальної власності та інших конфіденційних відомостей.

До технічних засобів інформації належать:

- Резервне копіювання і віддалене зберігання масивів важливих даних додаткових серверах.
- Дублювання та резервування всіх підсистем мережі, що мають значення для цілісності даних.
- Створення можливості перерозподілу ресурсів в мережі у випадках виходу з ладу елементів мережі.
- Можливість використання резервних джерел живлення.
- Встановлення програмного забезпечення, що забезпечує захист баз даних та конфіденційної інформації від несанкціонованого доступу.

Основною метою віртуальної мережі є максимально можливе відокремлення потоку даних компанії (певної групи користувачів) від потоку даних інших користувачів загальної мережі.

Відповідно, глобальним завданням віртуальної мережі є забезпечення зазначеної відособленості даних, яку можна розділити на наступні під задачі, які вирішуються засобами VPN:

- Конфіденційність - гарантія того, що дані не будуть переглянуті третіми особами.
- Цілісність - забезпечення схоронності переданих даних.
- Доступність - санкціоновані користувачі повинні мати можливість підключення до VPN постійно.

Переваги VPN полягають в тому, що оверлей побудована мережа є набагато ефективнішою у технологічному та фінансовому плані, реалізована на базі маршрутизаторів зі спеціальною операційною системою, або ж програмних клієнтів мережа, не потребує значних вкладень на розвиток інфраструктури. Таким чином, трафік передається публічною мережею, використовуючи незахищені протоколи, алгоритми шифрування, що працюють, при передачі через VPN, дозволяють об’єднати два сайти корпорації захищеним каналом в єдину мережу. При використанні VPN ми користуємося багатьма властивостями, що характерні виділеній лінії, завдяки тунелюванню трафіку, як при звичайному з’єднанні точка-точка. Кожна пара «відправник-одержувач» відбувається встановлення тунелю- безпечного логічного з’єднання, при якому можлива інкапсуляція даних одного протоколу в пакети іншого протоколу

Основна методика роботи захищеної приватної мережі залишається майже незмінною, не залежно від використовуваних протоколів та рівня їх роботи. Зазвичай використовується пара протоколів, один з яких бере участь у встановленні з’єднання, а інший – в інкапсуляції для подальшої передачі по тунелю.

1.2 Класифікація VPN і їх характеристика

VPN мережі будуються з застосуванням протоколів тунелювання даних та передачі їх через загальнодоступну мережу Інтернет, саме протоколи тунелювання забезпечують шифрування даних.

На рис. 1.2 наведена класифікація VPN мереж. Для класифікації протоколів у VPN мережах можна поділити їх на групи відповідно до рівня, на якому вони працюють. Згідно моделі взаємодії відкритих систем : канальний, мережевий, транспортний рівні.

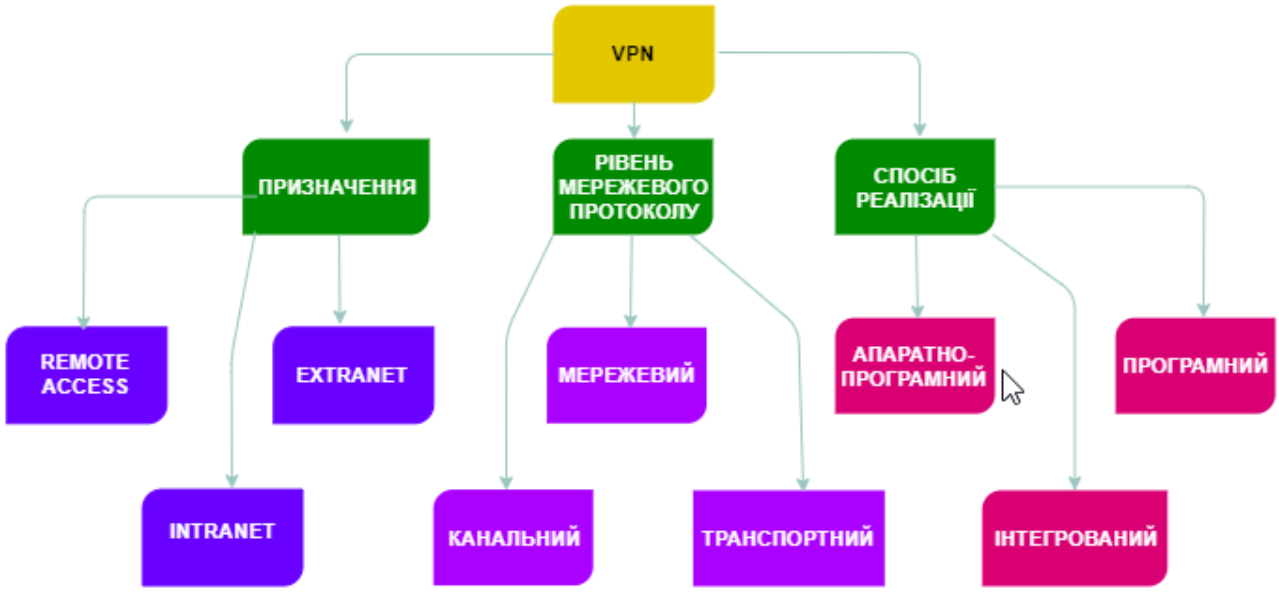


Рис.1.2 Класифікація VPN мереж

Протоколи канального рівня

Протоколами канального рівня є протоколи L2TP та PPTP, що використовують автентифікацію та авторизацію.

Протокол двоточкового тунельного зв'язку (Point-to-Point Tunnelling Protocol – PPTP) використовує в процесі роботи відкриті стандарти TCP/IP, в основі протоколу лежить протокол двоточкового зв'язку PPP. PPTP

використовує інкапсуляцію IP пакетів. PPTP клієнти використовують порт призначення для створення керуючого тунелем з'єднання. Даний процес проходить на транспортному рівні моделі взаємодії відкритих систем. Після моменту створення тунелю клієнт та сервер починають обмін службовою інформацією. Після встановленого керуючого з'єднання, створюється з'єднання для пересилання даних по тунелю.

Під інкапсуляцією даних, що передує відправці в тунель розуміють два етапи: спочатку створюється інформаційна частина PPP. Дані проходять від прикладного рівня моделі взаємодії відкритих систем до канального. [2]

Другим етапом є відправка даних вгору по моделі взаємодії відкритих систем та інкапсуляція протоколами верхніх рівнів. PPTP шифрує поле корисного навантаження пакету та бере на себе функцію канального рівня , саме додає до PPTP-пакету PPP-заголовок та кінцевик, що завершує формування кадру на канальному рівні. Після цього PPTP здійснює інкапсуляцію PPP-кадра в пакет GRE (Generic Routing Encapsulation), що належить мережевому рівню Після інкапсуляції кадру PPP з заголовком GRE, пакет інкапсулюється в IP пакет, в якому IP заголовок містить адресу відправника та отримувача пакету. На фінальному етапі PPTP додає PPP заголовок та кінцевик.

L2TP – Layer 2 Tunneling Protocol. Протокол використовує в якості транспорту протокол UDP, при цьому формат повідомлень для встановлення службового з’єднання та пересилання даних однаковий. L2TP, що працює на базі IPSec надає вищий рівень безпеки та гарантує майже високий рівень безпеки даних.

MPLS Multiprotocol Label Switching – багато-протокольна комутація по міткам – механізм передачі даних, що емулює особливості мереж з комутацією каналів на базі мереж з комутацією пакетів. Функціонування протоколу MPLS відбувається на рівні, що знаходиться між канальним та мережевим рівнями моделі взаємодії відкритих систем. Протокол було розроблено для створення

універсальної служби передачі даних для мереж з комутацією як пакетів так і каналів. Рішення, щодо організації VPN на канальному рівні мають область дії в рамках домену інтернет сервіс провайдера.

Протоколи мережевого рівня

На мережевому рівні використовується протокол IPSec, що реалізує шифрування та конфіденційність даних, а також автентифікацію. Застосування протоколу IPSec дозволяє реалізацію повнофункціонального доступу, що є еквівалентним фізичному під'єднанню до корпоративної мережі. Для того щоб встановити VPN з'єднання кожен з клієнтів мережі повинен здійснити конфігурацію набору параметрів IPSec, тобто – мати налаштоване програмне забезпечення.

IPSec – це протокол, в який закладено алгоритми шифрування, що сприяють захисту інформації у каналах VPN мереж. IPSec або Internet Protocol Security стандарт, для роботи якого потрібна його підтримка лише між відправником та одержувачем, всі інші вузли, що знаходяться на маршруті просто забезпечують передачу IP трафіку. Спосіб взаємодії двох вузлів, що використовують IPSec визначають, як захищена асоціація

Security Association (SA). Захищена асоціація функціонує на основі згоди, яку заключають учасники, що використовують засоби IPSec. Дана згода погоджує ряд параметрів: IP-адреса одержувача та відправника, криптографічний алгоритм, порядок обміну ключами, розмір ключів, строк їх служби, алгоритм автентифікації.

Основу стандарту складають три протоколи: AH або Authentication Header – заголовок автентифікації, що гарантує цілісність та автентичність даних, основне призначення протоколу AH - впевнитися стороні отримувача у тому, що пакет було відправлено стороною, з якою було встановлено безпечну асоціацію; у тому, що зміст пакету не було спотворено в процесі передачі по

мережі; у тому, що пакет не є дублікатом вже отриманого пакету. Для виконання даних функцій протокол AH використовує заголовок.

ESP або Encapsulating Security Payload – інкапсуляція зашифрованих даних. Протокол вирішує дві задачі - забезпечення автентифікації та цілісності даних на основі дайджесту повідомлення та захист даних, що передаються, від несанкціонованого доступу.

Протоколи AH та ESP можуть працювати в двох режимах: транспортному та тунельному. В транспортному режимі передача даних проходить з оригінальними заголовками, а в тунельному – початковому пакету надається новий заголовок, шляхом інкапсуляції в інший пакет.

Функції протоколів AH і ESP частково повторюються: протокол AH грає роль забезпечення цілісності та автентифікації даних, протокол ESP може шифрувати дані та виконувати функції протоколу AH (в обмеженому вигляді). ESP може підтримувати функції шифрування і автентифікації / цілісності тобто або всю групу функцій, або тільки автентифікацію / цілісність даних, або лише функції шифрування.

IKE або Internet Key Exchange – процес обміну ключами Internet у процесі функціонування вирішує додаткове завдання автоматичного видавання секретних ключів кінцевим точкам захищеного каналу, що необхідні для роботи протоколів автентифікації та шифрування даних.

Транспортний рівень

На транспортному рівні працює протокол SSL/TLS або Secure Socket Layer/Tгansport Layer Security, що реалізує шифрування та автентифікацію між транспортним рівнями відправника та одержувача. SSL/TLS може застосовуватись для захисту трафіку TCP, протокол не може бути використаний у цілях захисту трафіку UDP. При побудові VPN мережі, що працює на базі SSL/TLS не потрібно встановлювати спеціальне програмне забезпечення,

оскільки всі браузери та поштові клієнти підтримують протоколи. Процес роботи SSL/TLS описується трьома фазами: встановлення діалогу між сторонами, у процесі якого узгоджується алгоритм шифрування, обмін ключами за алгоритмом з відкритим ключем або автентифікація на основі сертифікатів, і нарешті передача даних, що проходять процес шифрування за допомогою симетричних алгоритмів шифрування. [3]

Класифікація за способом реалізації

1. Програмно-апаратні засоби

VPN мережі можуть бути реалізовані у вигляді програмно апаратного забезпечення. Реалізація VPN мережі базується на спеціалізованому комплексі програмно- апаратного рішення, наприклад маршрутизатор з над лаштованим інтерфейсом, яким можна управляти з програмного забезпечення. Така реалізація характеризується високим рівнем захищеності та продуктивності.

VPN мережі, реалізовані на програмному забезпеченні потребують лише персональний комп’ютер з встановленим відповідним ПЗ.

VPN мережі з інтегрованим рішенням. Забезпечують функціональність, що вирішує питання фільтрації трафіку, мережевий екран, забезпечення якості обслуговування.

Одним зі способів створення захищених каналів VPN є використання маршрутизаторів. Вся інформація, що виходить з локальної мережі проходить через маршрутизатор, тож на нього покладаються функції шифрування трафіку. Таким чином, операційні системи маршрутизаторів можуть підтримувати протоколи L2TP и IPSec та функції ідентифікації при встановленні з’єднання, обмін ключами. Наприклад, у Cisco System існує спеціалізований пристрій , що має назву Cisco 1720 VPN Access Router, призначений для встановлення в компаніях малого та середнього розміру.[2]

2. Програмні засоби

У випадку реалізації VPN мережі на базі лише програмного рішення, виробником постачається спеціалізоване, яке працює на окремому комп'ютері. Більшість таких реалізацій – це ПЗ, що виконує роль проксі-сервера. Комп'ютери з таким ПЗ можуть знаходитись за брандмауером. Прикладом такого рішення є AltaVista Tunnel компанії Digital. При використанні такого програмного комплексу клієнт здійснює підключення до сервера Tunnel, проходить автентифікацію та обмін ключами. Шифрування відбувається з використанням ключів 56 або 128 біт, якими обмінялися клієнт з сервером на початку з'єднання. Далі відбувається процес інкапсуляції зашифрованих пакетів у пакети IP, що надсилаються на сервер. Однією з особливостей такого програмного рішення є генерація нових ключів кожні 30 хвилин.

Простота встановлення та зручна конфігурація програмно реалізованих VPN мереж є перевагою. Недолік- це нестандартна архітектура, а саме : власний алгоритм обміну ключами та не завжди найвищу продуктивність.

3. Апаратні засоби

У випадках, коли необхідна висока ефективність та продуктивність роботи мережі VPN, може бути використано апаратні засоби.

VPN може бути побудована на спеціальних пристроях та може бути використано в мережах, що вимагають високої продуктивності. Такі рішення використовують апаратне шифрування переданої інформації і мають можливість пропускати потік даних від 100 Мбіт/с. Оснащені підтримкою протоколів різних рівнів і механізмами утворення та обміну ключами, апаратні рішення мають високий рівень захисту інформації. Зазвичай можуть бути доповнені апаратно, та підтримувати додаткові функції, наприклад брандмауеру. VPN класифікують за призначенням. Remote access, Extranet, Intranet мережі.

Remote access VPN – працює таким чином, що тунель організовується між програмним клієнтом на комп'ютері користувача і будь-яким пристроєм, який виступає в якості сервера і організовує підключення від різних клієнтів. Структурна схема мережі зображена на рисунку 1.3

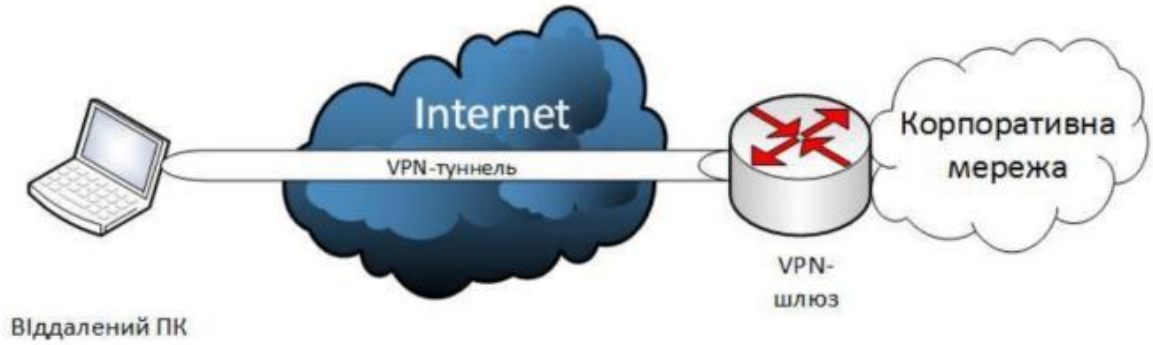


Рис. 1.3 Remote access VPN

Між корпоративні мережі VPN (extranet-VPN) створюються для забезпечення представникам компаній захищений обмін зі стратегічними партнерами по бізнесу, замовниками, постачальниками, клієнтами і тому подібне. Такий тип мережі надає прямий доступ з мережі однієї компанії у мережу іншої, тим самим сприяючи захищеному обміну інформацією. Структурна схема мережі зображена на рис. 1.4

Рис. 1.4 Extranet VPN

Внутрішньо корпоративні мережі VPN (intranet-VPN) – це мережі такого типу, що забезпечують зв'язок територіально розподілених філіалів однієї організації. Структурна схема мережі зображена на рисунку 1.5

Рис. 1.5 Intranet VPN

1.3 Постановка завдань на магістерську дисертацію

На даний період часу актуальність захисту інформації та захищеного обміну інформацією має пріоритетне значення. Корпоративні мережі, міжкорпоративні комунікації, віддалений доступ, доступ до територіально обмежених ресурсів є необхідною умовою багатьох сучасних рішень побудови каналів зв’язку. Інформація, що передається має бути забезпечена умовами, при яких вона не може бути змінена, підроблена, видана за чиюсь власність. На даний час більшість реалізованих VPN сервісів є централізованими, тобто мають точку, через яку проходить більша кількість трафіку. З одного боку, вона конфігурується і централізовані засоби шифрування та захисту інформації можуть бути збережені в її конфігураційних файлах. З іншого боку – при отриманні доступу до центрального вузла, можна знизити рівень безпеки системи, або ж взагалі нанести шкоду корпоративним даним.

Однією з альтернатив є побудова такої VPN мережі, що буде децентралізованою з можливістю контролю усіх попередніх фактів передачі даних, що є захищеними від зміни, контролю відправників мережі, тобто,

інформація, що дані були передані між двома учасниками, відома визначеній кількості учасників мережі, при цьому доступ до неї обмежений та може бути наданий лише тим, хто здійснював обмін.

Тож у дипломній роботі поставлено задачі:

1. Розгляду технології Blockchain, що володіє зазначеними вище особливостями.
2. Визначення компонентів роботи технології Blockchain, аналіз можливості реалізації захищеної комунікації вузлів мережі.
3. Розробка методу реалізації мережі VPN на базі технології Blockchain та його опис.
4. Розробка методичних матеріалів, покрокового опису реалізацію макету мережі VPN на базі технології Blockchain.

Висновки до розділу

1. У розділі було проаналізовано основні області та цілі використання мереж VPN. Результатом цього можемо дійти до висновків, що застосування додаткових засобів захисту інформації в корпоративних мережах та персональних цілях є обов’язковою складовою будь-якої сучасного каналу передачі даних.

2. Наведена класифікація мереж за способом реалізації, призначенням та рівнем мережевого протоколу вказує на різноманітність сучасних способів реалізації приватного простору у випадках, де це необхідно. Сучасні приватні мережі представляють собою досить гнучкі структури, що, тим не менш, мають ряд своїх особливостей та потребують додаткових засобів приватності.

3. Вважаючи на вищезазначені особливості, застосування нових технологій в даній сфері з використанням Блокчейн структур може спробувати вирішити існуючі проблеми сучасних VPN мереж.

2. ПРИНЦИПИ СТРУКТУРНОЇ ПОБУДОВИ МЕРЕЖ VPN. ВВЕДЕННЯ ДО ПОНЯТТЯ BLOCKCHAIN ТА ВИЗНАЧЕННЯ ОСНОВНИХ ЕЛЕМЕНТІВ ДАНОЇ МЕРЕЖІ

Завданням першої частини даного розділу є визначення основних видів мережевих топологій в класичних VPN структурах, опис їх принципів роботи. Ця інформація має вказати на основні особливості роботи мережевих клієнтів, вказати на їх основні переваги та недоліки. Друга частина даного розділу має на меті створення уявлення про роботу альтернативних мереж комунікації, таких як The Onion Router та Blockchain структури. На останній буде зосереджено основну увагу, розкрито принципові особливості та механізми в її складі. Дана інформація має слугувати для створення власної концепції й роботи розподіленої VPN мережі.

2.1 Топології мереж VPN і їх характеристика

Сучасні VPN мережі та їх топології визначаються вимогами, що ставлять підприємства для рішення своїх бізнес проблем. Однак, кілька відомих топологій використовуються найбільш часто.

Одні і ті ж топології можуть вирішувати різні проблеми бізнесу на різних вертикальних ринках та сферах. Розділимо топології VPN мереж на дві категорії:

1. Топології, що побудовані на вже існуючих мережах – модель overlay (укр. нашаровування) VPN, сюди можна віднести радіальну топологію, частково або повно зв’язну mesh топологію та гібридну топологію.

2. Extranet топології, сюди відносяться з’єднання кожний-з-кожним та топології з центральним вузлом.

2.1.1 Радіальна топологія

Радіальна топологія найбільш часто зустрічається та представляє собою топологію, в якій кілька віддалених офісів підключені до центрального вузла, як зображено на рис. 2.1.

Топологія Hub-and-spoke зазвичай використовується в організаціях зі строгими ієрархічними структурами, наприклад, в банках, урядах, роздрібних магазинах, міжнародних організаціях з невеликими внутрішньодержавними офісами.

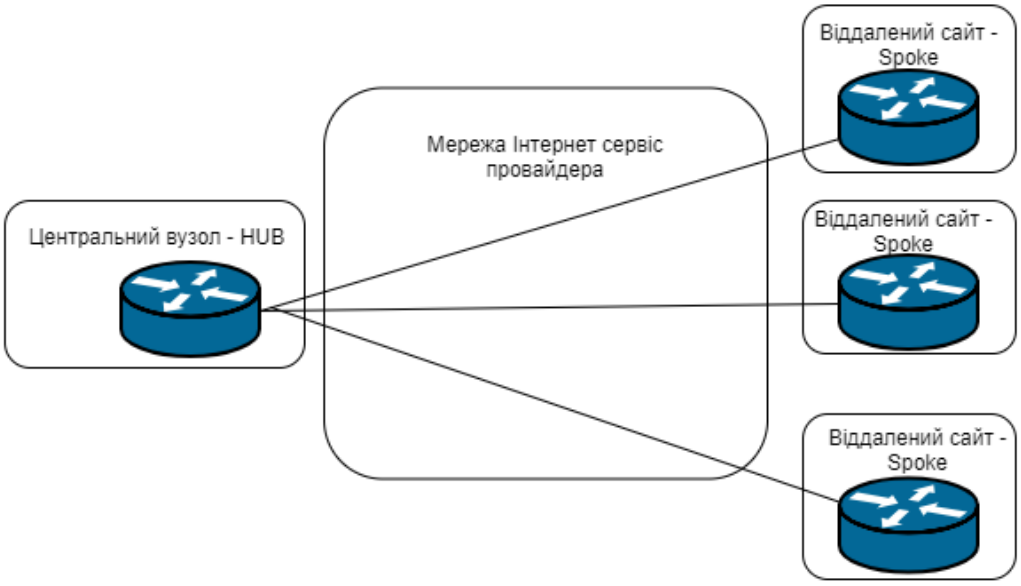


Рисунок 2.1. Hub-and-spoke(радіальна) топологія

Під час розгортання мереж VPN на основі технологій канального рівня, таких як Frame Relay або ATM, доволі часто використовується радіальна топологія, що обумовлено найнижчою вартістю реалізації та закупки обладнання для радіальної топології, порівняно з іншими видами топологій.

Іншими словами, у багатьох випадках замовник мав би ряд інших переваг, обравши іншу топологію, але обирається саме радіальна, виходячи з міркувань нижчої складності та вартості.

У випадку підвищених вимог до надлишковості системи, топологія hub-and-hub, зазвичай, удосконалюється за допомогою використання додаткового маршрутизатора в точці комутації, що зображено на рис. 2.2, або за допомогою резервуючої точки, яка в даному випадку має бути зв’язана з головним вузлом через високошвидкісне з’єднання, що показано на рис. 2.3

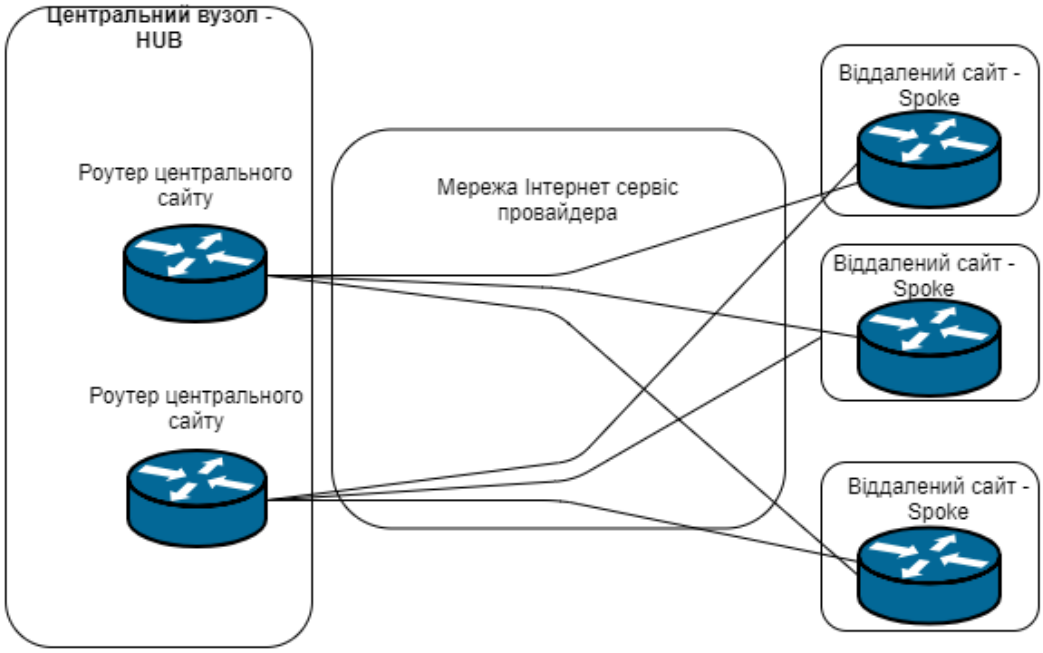


Рисунок 2.2. Hub-and-spoke топологія з двома центральними маршрутизаторами

Реалізація надлишкової радіальної топології з VPN моделлю, що базується на віртуальних каналах (VC), завжди створює ряд проблем. Кожен вузол сайту вимагає VC, принаймні, для двох центральних маршрутизаторів. Такі VC можуть бути враховані у основній та резервній конфігураціях, або конфігураціях розподілу навантаження, з рядом недоліків:

- У основній та резервній конфігураціях, резервуючий VC не використовується, коли основний VC в роботі, тож користувач несе непотрібні витрати.

- У конфігурації розподілу навантаження, точка комутації – центральний сайт зазнає зниження пропускної здатності, якщо один з VC, або один з центральних маршрутизаторів виходить з ладу. Конфігурація розподілу навантаження зазвичай є доцільною для топології з резервною точкою комутації, як зображено на рис. 2.3.

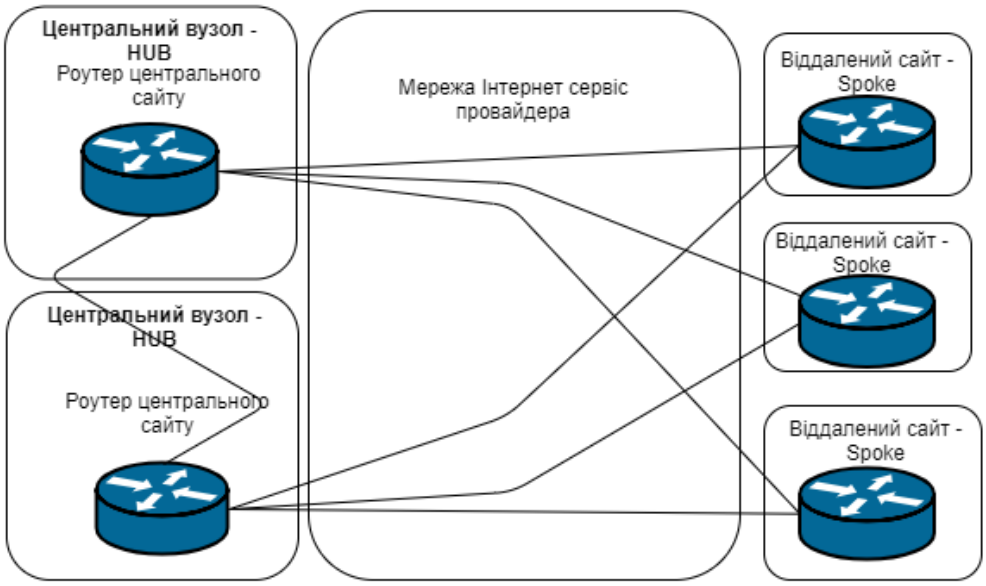


Рисунок 2.3. Hub-and-spoke (радіальна) топологія з двома роздільними центральними маршрутизаторами.

Постачальники, що прагнуть надати клієнтам більш якісні послуги, враховують вимоги надлишковості та пропонують клієнтам покращений сервіс: приватний віртуальний канал. Користуючись таким сервісом користувач отримує дві віртуальні схеми за ціною однієї при умові, що для передачі трафіку може використовуватись лише один VC за раз, при цьому невелика кількість даних може передаватися другим VC, для того, щоб у разі необхідності дозволити обмін протоколами маршрутизації через другий VC.

Вимоги щодо резервування можуть ускладнювати будову радіальної топології Hub-and-spoke з введенням функції резервного копіювання. Резервна

копія, що реалізується в межах мережі постачальника послуг (наприклад, з'єднання ISDN, резервує Frame Relay лінію, як показано на рис.2.4), є прозорим для клієнта, однак не надає наскрізного резервування, оскільки не може виявляти усі можливі виходи з ладу (наприклад проблеми протоколу CPE або протоколу маршрутизації). Наскрізне резервування в мережах VPN, побудованих поверх мережі Інтернет сервіс провайдера може бути досягнуте лише за допомогою встановлення dial-up з'єднання за межами простору VPN мережі.

Зазвичай проста радіальна топологія перетворюється в багаторівневу топологію в міру зростання мережі. Багаторівнева топологія може бути рекурсивною радіальною топологією, що показана на рис 2.5, або гібридною топологією, яка розглядається далі в цьому розділі. Реорганізація мережі може бути викликана обмеженнями масштабованості протоколів IP-маршрутизації або проблемами масштабованості на рівні додатків (наприклад, впровадження трирівневого підходу клієнт-сервер).

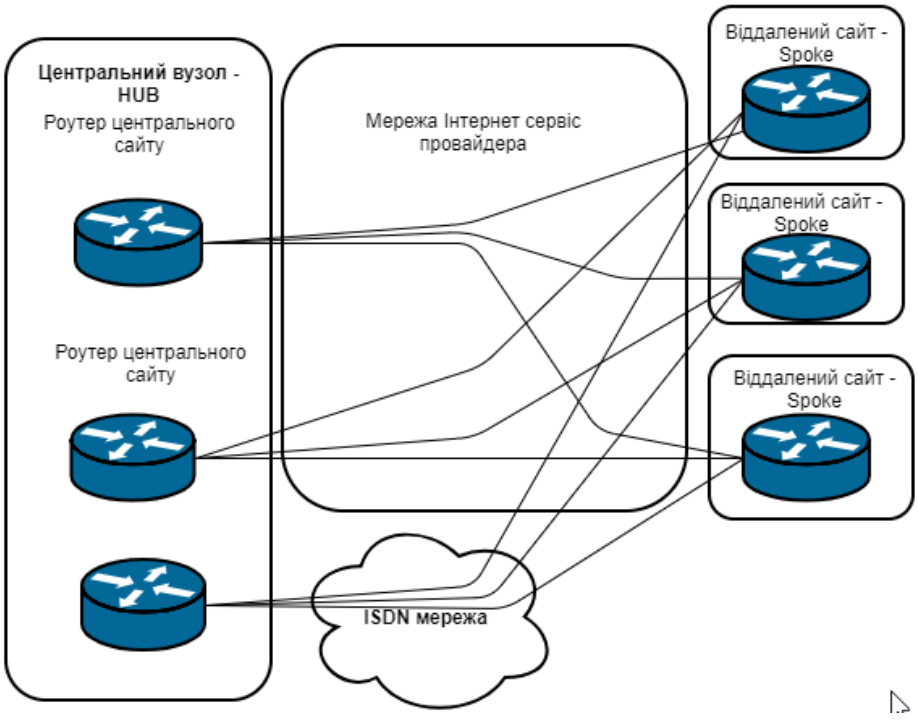


Рисунок 2.4. Dial Backup рішення в межах мережі сервіс-провайдера

Топологія Hub-and-spoke реалізована за допомогою оверлейної моделі VPN, добре підходить для середовищ, в яких віддалені офіси в основному обмінюються даними з центральними сайтами, а не один з одним, оскільки дані, якими обмінюються віддалені офіси, завжди транспортуються через центральний сайт. Якщо переважний обсяг даних, передається саме між віддаленими офісами, то більш доцільно використовувати з частково зв'язну або повно зв'язною mesh топологію.

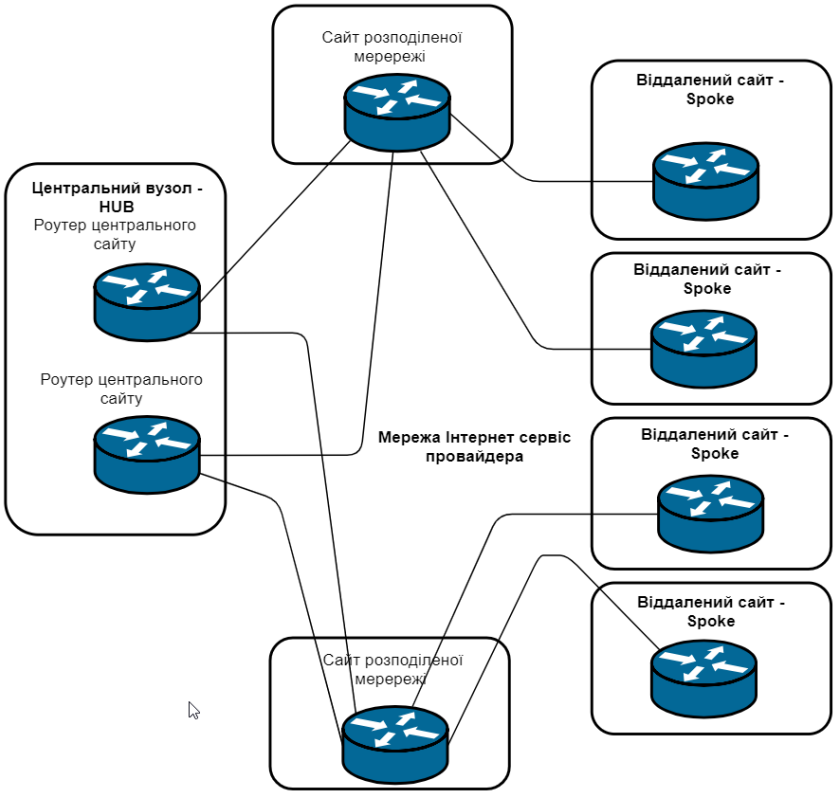


Рисунок 2.5. Багаторівнева Hub-and-spoke топологія

2.1.2 Частково або повно зв'язні mesh топології.

Не завжди замовникам вигідно реалізовувати мережі у вигляді радіальної топології, з огляду на ряд причин:

1. Підприємство може мати менш ієрархічну структуру та потребувати обміну даними між рівноправними точками.

2. Підприємство може використовувати програмне забезпечення , що потребує одноранговий зв’язок між клієнтами у процесі роботи (наприклад, системи обміну повідомленнями або спільної роботи).

3. Для деяких багатонаціональних корпорацій вартість радіальної топології може бути надмірною через високу вартість міжнародних каналів.

У таких умовах модель оверлей VPN, що найкраще підходить для організаційних потреб – це частково зв’язна mesh топологія, де сайти організації з’єднані за допомогою віртуальних каналів, що визначаються вимогами до трафіку (які продиктовані потребами організації). Якщо не всі сайти мають прямий зв'язок з усіма іншими сайтами (приклад на рис. 2.6), топологія називається частково зв’язною mesh топологією; якщо кожен сайт має пряме сполучення з кожним іншим сайтом, топологія називається повно зв’язною mesh топологією.

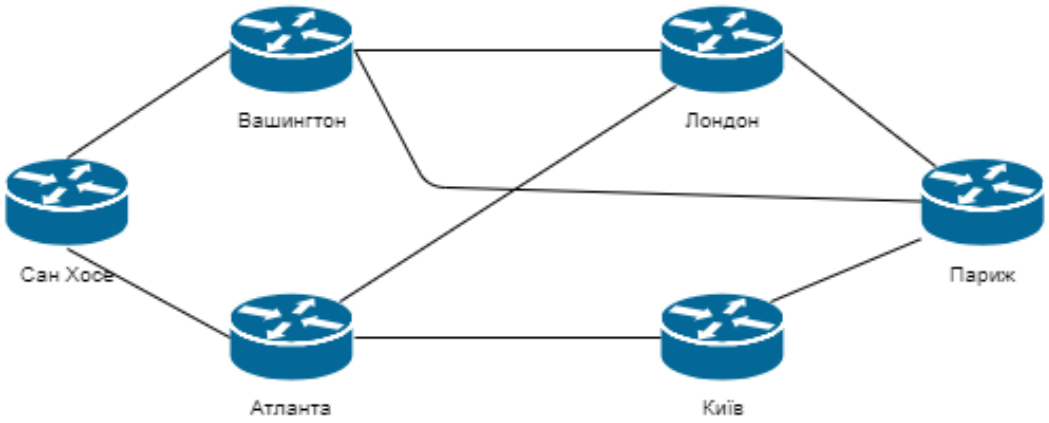


Рисунок 2.6. Частково зв’язна mesh топологія

Мережі з повно зв’язною mesh топологією не так часто реалізуються, через дуже високу вартість цього підходу і складності, причиною чого стали масові віртуальні канали. При такому типі топології число каналів $C = [(n - 1)\ x\ n)\ \div\ 2]$, де n дорівнює числу підключених пристроїв.

Більшість клієнтів змушені погоджуватися на не повно зв’язну mesh топологію, на яку зазвичай впливають такі зовнішні параметри, як доступність та вартість каналів.

Топологія з повно зв’язною mesh топологією реалізується досить просто: складається матриця трафіку, яка враховує пропускну здатність, необхідну для роботи двох сайтів VPN, у постачальника послуг замовляються віртуальні канали. З іншого боку, реалізація не повно зв’язної mesh топології може бути складним завданням, оскільки необхідно:

1. Проаналізувати потреби підприємства та розрахувати матрицю трафіку.

2. Запропонувати топологію з частково зв’язною mesh топологією на основі вимог резервування та матриці трафіку (наприклад, замовити віртуальні канали тільки між сайтами з високими вимогами до трафіку).

3. Визначити, через які саме віртуальні канали буде направлятися трафік між будь-якими двома сайтами. Цей крок також може включати налаштування протоколу маршрутизації, щоб переконатися, що трафік протікає через відповідні віртуальні канали.

4. Розрахувати ємність віртуального каналу відповідно до матриці трафіку і агрегації трафіку.

2.1.3 Гібридна топологія

Великі VPN-мережі, побудовані з використанням оверлейної VPN-моделі, як правило, об'єднують радіальну топологію з частково зв’язною mesh топологією. Наприклад, велика багатонаціональна організація може мати мережі доступу в кожній країні, реалізовані радіальною топологією, тоді як мережа всередині країни буде реалізована частково зв’язною mesh топологією. На рисунку 2.7 показаний приклад такої організації.

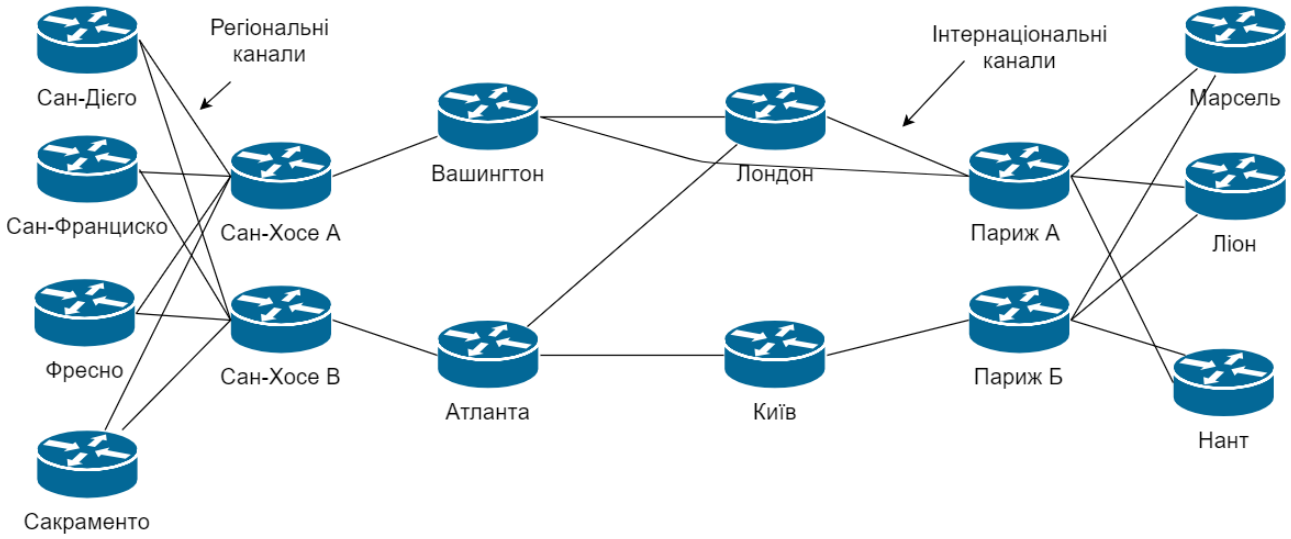


Рисунок 2.7. Приклад гібридної топології

Найкращий підхід до побудови гібридної топології полягає в наступному:

- Розділити загальну мережу на основні, розподілені та мережі доступу.
- При конструюванні топології необхідно ізолювати ядро та мережі доступу через рівень розподілу (наприклад, локальний збій у віддаленому офісі не розповсюдиться по всій мережі).
- Аналогічним чином, маршрутизатори віддаленого офісу не мають фіксувати поломки на міжнародних каналах).

2.1.4 Проста Extranet топологія

Intranet топології стосуються в основному фізичної або логічної топології VPN, як це диктується технологією віртуальних каналів, за допомогою яких реалізована оверлейна модель VPN. У топологіях Extranet зазвичай увага приділяється вимогам безпеки VPN-мережі, котрі потім можуть бути реалізовані з використанням ряду різних топологій, як за допомогою оверлей моделі, так і однорангової VPN мережі.

Традиційна топологія Extranet дозволяє ряду компаній проводити обмін даними кожний-з-кожним. Як приклад можна навести групи підприємств

об’єднаних специфікою роботи: авіаційні компанії, виробники авіаційної техніки. Або ланцюжки постачальників (наприклад, виробник автомобілів та постачальники комплектуючих.

Дані в Extranet топології такого типу можуть бути передані між будь яким числом сайтів. Сама Extranet топологія не накладає обмежень на обмін даними. Зазвичай кожен сайт відповідає за власну безпеку, фільтрацію трафіку та між мережевий екран. Єдиною причиною використання Extranet замість звичного Інтернету є якість гарантій обслуговування, з огляду на чутливість та цінність даних, що передаються по такій мережі.

Якщо Extranet топологія реалізована на основі однорангової VPN-моделі (приклад Extranet на рисунку 2.8), кожна організація вказує лише кількість трафіку яку він збирається отримати та надіслати з кожного із своїх сайтів. Тож планування зі сторони клієнта та провайдера послуг є простим та ефективним.

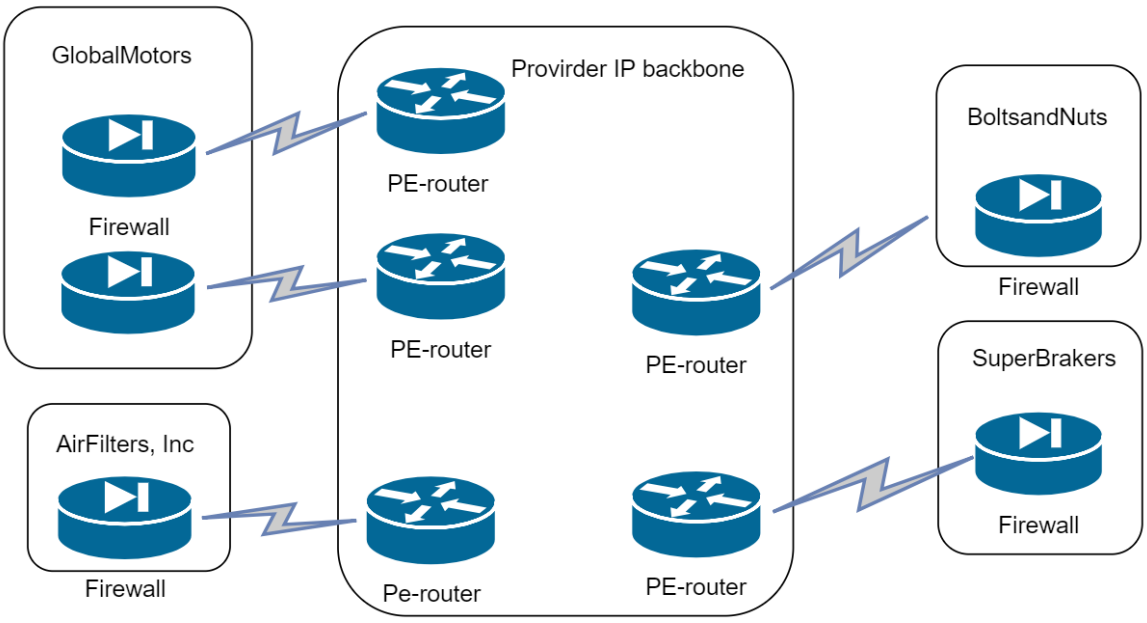


Рисунок 2.8. Приклад VPN моделі з вбудованим Peer-to-peer

Але в моделі оверлейної VPN мережі трафік між сайтами передається по віртуальним каналам точка-точка, приклад зображено на рис. 2.9.

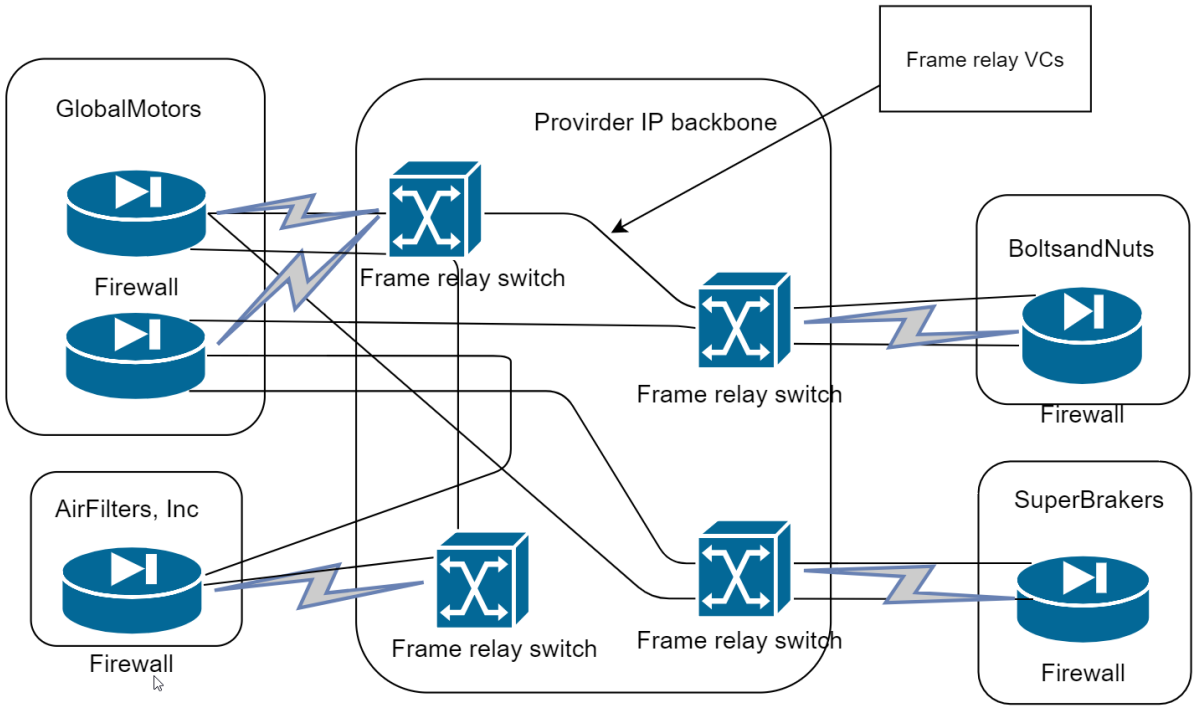


Рисунок 2.9. Приклад Extranet топології, реалізованої з використанням моделі Overlay VPN

В топології Extranet, що зображена на рисунку 2.9, кожна організація, що є в складі мережі, сплачує за канали, якими користується. Очевидно, що для мінімізації витрат встановлюються лише самі необхідні віртуальні канали. Крім того, учасники такої VPN мережі можуть перешкоджати транзитному трафіку між іншими учасниками, що проходять через цей самий віртуальний канал. Це часто приводить до проблем маршрутизації та часткової втрати зв'язку. Таким чином, однорангова VPN-модель є найкращим способом реалізації однорангової Extranet топології. [4]

2.1.5 Central-services Extranet

Організації, що працюють у спільній сфері, що поєднані в одну мережу за допомогою Extranet топології часто є досить відкритими, що дозволяє з'єднання кожний-з кожним між організаціями. Extranet топології спеціального призначення (наприклад, мережі управління ланцюгами поставок, що

пов’язують велике підприємство з його постачальниками), як правило, більш централізовані та забезпечують з’єднання з головним підприємством усіх постачальників, або дочірніх компаній, приклад на рис. 2.10.

Інші приклади такої Extranet топології включають біржові мережі, де кожен брокер може спілкуватися з фондовою біржою, але не з іншими брокерами або фінансовими мережами, побудованими в деяких країнах між центральним банком і комерційними банками. Хоча цілі таких Extranet типів мереж можуть широко варіюватися, всі вони мають загальну концепцію: кілька різних користувачів отримують доступ до центральної службі (додаток, сервер, сайт, мережа).

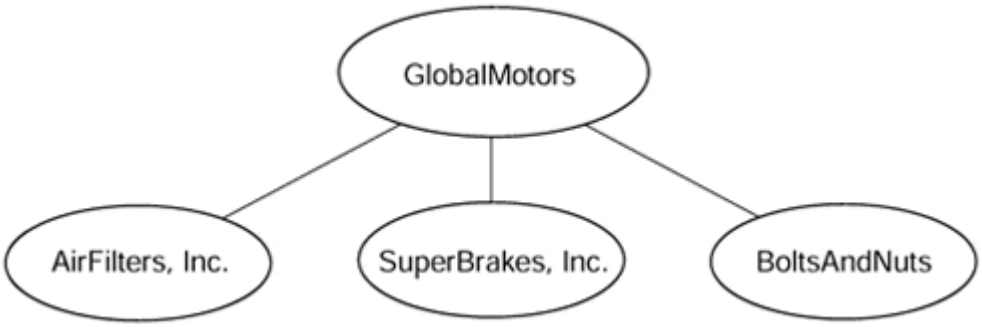


Рисунок 2.10. Extranet топологія, як ланцюг поставок.

Безпека в Extranet топології з централізованим з’єднанням підприємств, зазвичай розробляється та впроваджується центральною організацією, яка зв’язує та оплачує Extranet топології. Інші учасники з критично важливими внутрішніми мережами (наприклад, біржові брокери або комерційні банки) також можуть захотіти реалізувати свої власні заходи безпеки (наприклад, між мережевий екран між своєю внутрішньою мережею і Extranet).

Подібно будь-який інший мережі VPN, Extranet топологія з централізованим з’єднанням підприємств може бути реалізована або за допомогою однорангової, або оверлейной моделі VPN.

Реалізація Extranet топології центральних служб за допомогою оверлейной моделі VPN надзвичайно проста: створюються віртуальні канали між усіма учасниками і центральним сайтом. Розмір кожного віртуального каналу відповідає вимогам до трафіку між учасником і центральним сайтом.

Центральний сайт оголошує під-мережі на центральному сайті, до яких матимуть доступ інші підприємства, підключені до головного. Центральний сайт фільтрує трафік, отриманий іншими учасниками, щоб переконатися в тому, що проблема маршрутизації або цілеспрямована атака на крадіжку служби не впливають на стабільність VPN. Після цих трьох кроків VPN-мережа з рис. 2.10 перетворюється в топологію з віртуальними каналами на рис. 2.9.

В рамках будь-якої моделі Extranet топології дана мережа матиме обмежену кількість VC (що призведе до перевантаженої радіальної топології) через обмеження витрат. В рамках моделі Extranet топології центральних служб одна і та ж VPN матиме таку ж кількість VC через обмеження безпеки. Таким чином, цей приклад представляє цікавий випадок, коли ряд різних вимог може диктувати ту ж топологію VC.

Трохи складніша Extranet топологія з централізованим з’єднанням підприємств може містити кілька серверів, розподілених по декількох сайтах і кілька клієнтських сайтів, які звертаються до цих серверів аналогічно налаштуванню на рис. 2.11. Типовими прикладами, які будуть потрібні для цієї топології, є мережі передачі голосу по IP-мереж, де ряд користувачів отримують доступ до загальних шлюзів в різних містах (або країнах), але не можуть бачити один одного.

Така Extranet топологія також може бути реалізована або за допомогою однорангової VPN-моделі, або за допомогою оверлей VPN-моделі. Кількість VC, необхідних для моделі оверлей VPN (потрібен окремий VC з кожного сайту клієнта на кожен серверний сайт), і відповідна складність ініціалізації зазвичай перешкоджає розгортанню оверлей моделі VPN в цих сценаріях.

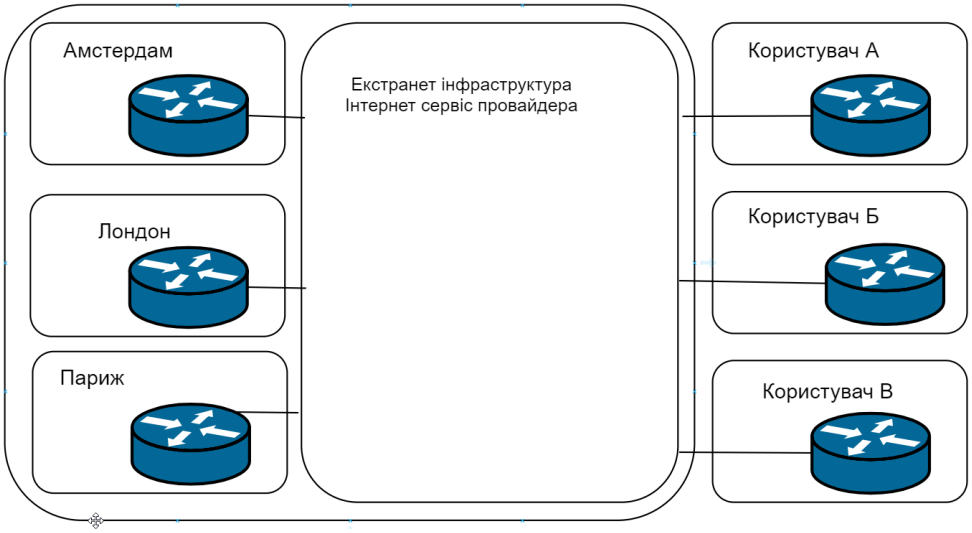


Рисунок 2.11. Extranet топологія з централізованим підключенням підприємств з великою кількістю сайтів.

Більш керована настройка буде використовувати або однорангову модель, або комбінацію обох моделей, як показано на малюнку 2.12.

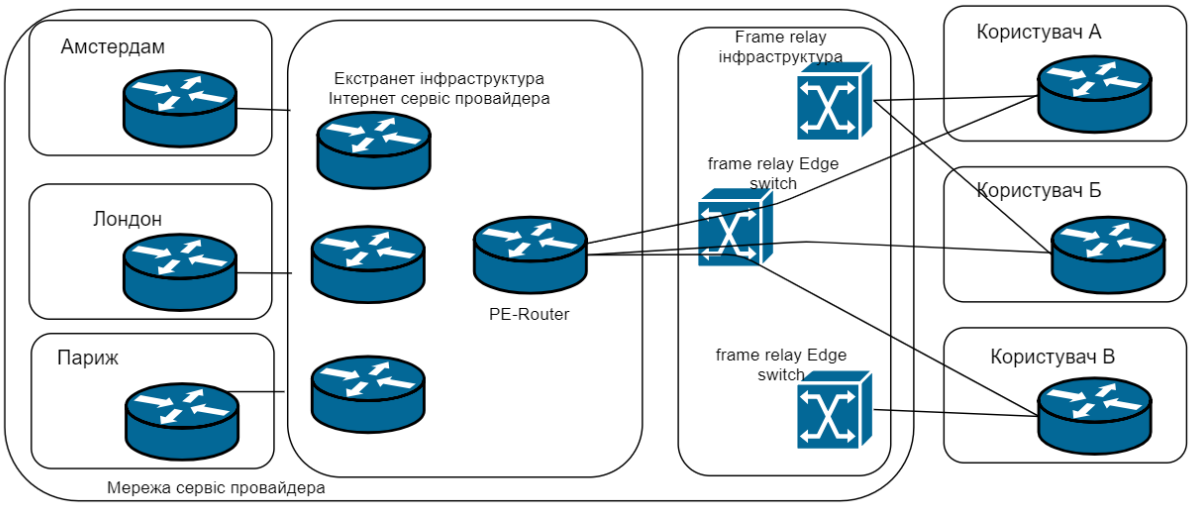


Рисунок 2.12. Combination of Peer-to-peer VPN with Overlay VPN

2.2 Технологія багатошарової маршрутизації Tor

Однією з сучасних технологій захищеного обміну інформацією та забезпечення високого рівня приватності в мережі Інтернет – це Tor

маршрутизація (англ. The Onion Router; укр. ‘Цибулева маршрутизація’). Даний тип маршрутизації є інфраструктурою загального призначення для приватної комунікації в публічній мережі. Tor працює завдяки динамічній побудові анонімних з’єднань. Мережа побудована з деякої кількості маршрутизаторів являє собою розподілену та контрольовану кількома адміністративними доменами, таким чином відсутня єдина точка відмови і один з маршрутизаторів мережі не може, вийшовши з ладу, зруйнувати мережу або порушити анонімність користувачів.

Для встановлення з’єднання необхідно пройти три фази: встановлення з’єднання, транзит трафіку, завершення з’єднання. Початком встановленням з’єднання вважається момент, коли ініціатор створює структуру даних, що є рекурсивною та багатошаровою, вона визначає шлях трафіку через мережу. Така структура даних визначає властивості з’єднання в кожному з вузлів, криптографічний контроль інформації (рис. 2.13 - 2.14).

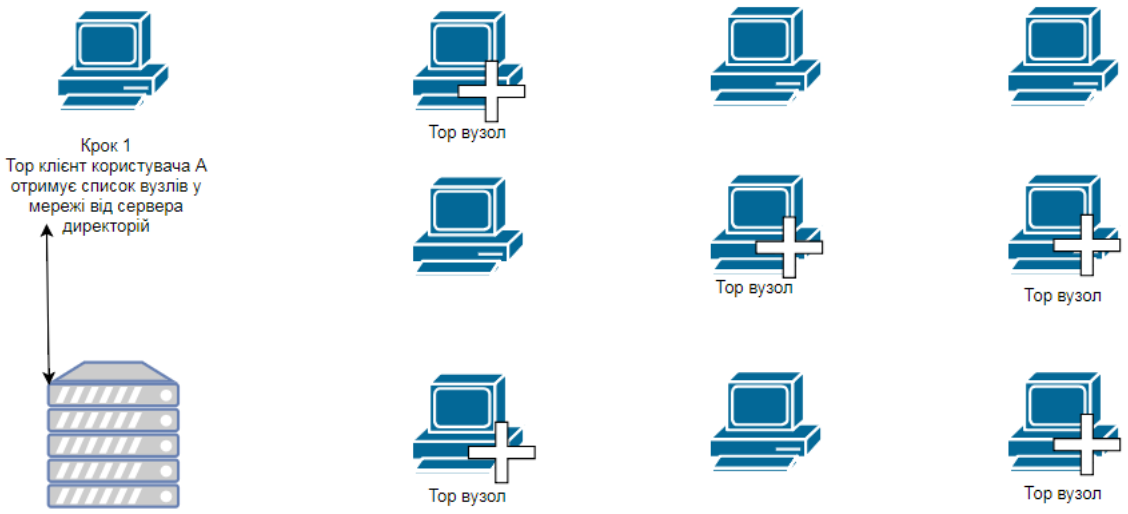


Рисунок 2.13. Крок 1, отримання списку Tor вузлів

Кожен маршрутизатор мережі, що знаходиться на маршруті використовує власний публічний ключ для розшифрування структури даних, яку було

сформовано ініціатором з'єднання. Цей процес дозволяє виявити наступний маршрутизатор, або наступну вбудовану структуру даних про маршрут.

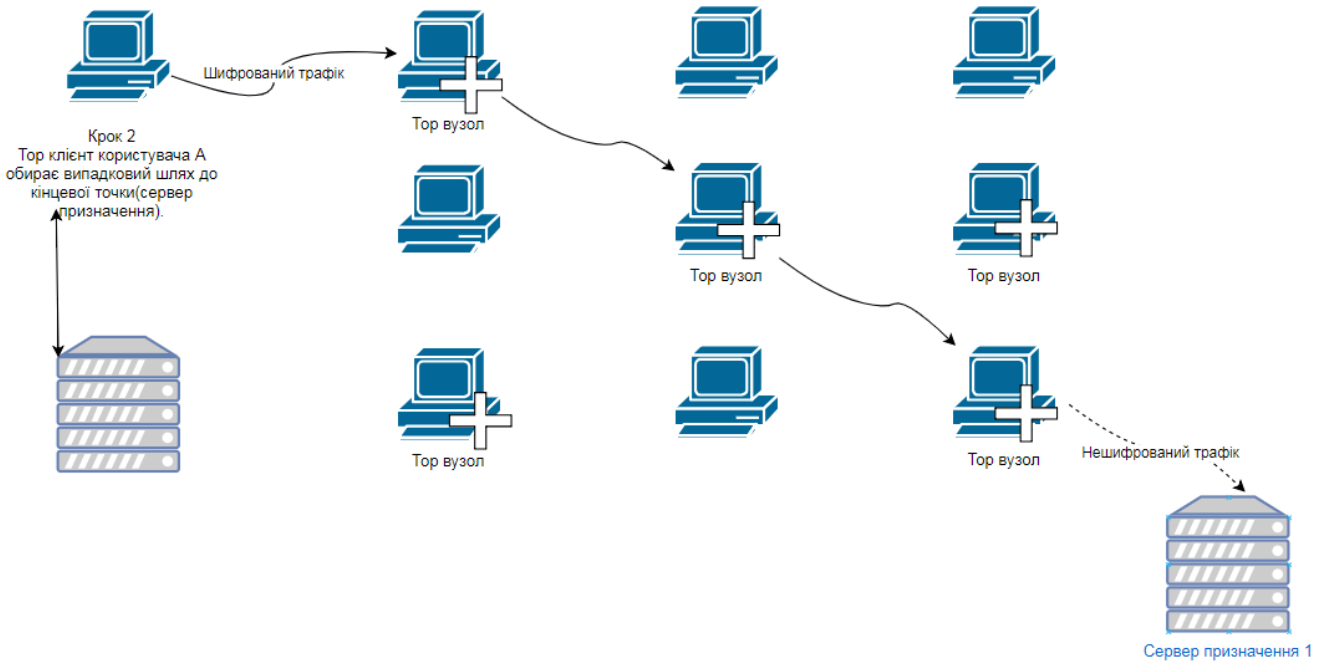


Рисунок 2.14. Крок 2, вибір маршруту до серверу призначення 1

Після встановлення з’єднання передача даних може здійснюватися в обидві сторони. Дані, що передаються ініціатором кожного разу повторно шифруються з використанням алгоритмів, що описані в початковій структурі даних. Під час транзиту трафіку через мережу Tor, кожен з маршрутизаторів видаляє один з шарів шифрування, заданий в криптографічному контролі на початку встановлення сеансу зв’язку. Таким чином, адресат отримує розшифровані дані. Мережа Tor являє собою групу добровільних серверів. Клієнти Tor мережі використовують її завдяки підключенню до серії віртуальних тунелів, що дозволяє ділитися інформацією через публічні мережі без компрометації приватності. [5]

Мережа Tor побудована на оверлей моделі. Користувач запускає програмне забезпечення, що називається Tor-проксі, що призначене для

встановлення ланцюга з'єднань та визначення директорій. Трафік проходить через мережу, формуючись в повідомлення фіксованого розміру, а саме повідомленнями по 512 байт – заголовок та корисна інформація. Заголовок містить ідентифікатор ланцюжка, що визначає приналежність ланцюга. В Tor мережі кожен ланцюжок передачі може бути поділений між кількома TCP-потоками. Для створення приватного маршруту проходження через мережу за допомогою Tor маршрутизації, користувацьке програмне забезпечення, або клієнт послідовно будують ланцюжки захищених з'єднань, через проміжні вузли – ретранслятори трафіку. Кожен з ретрансляторів має доступ до інформації про те, хто передав йому пакет і кому він має передати його далі. Немає такого вузла в мережі, який повністю мав дані про маршрут, який проходить пакет всередині мережі. Далі зображено приклад встановлення з'єднання, передачі трафіку та отримання його у вигляді не зашифрованої інформації отримувачем - сервером призначення (рис. 2.15):

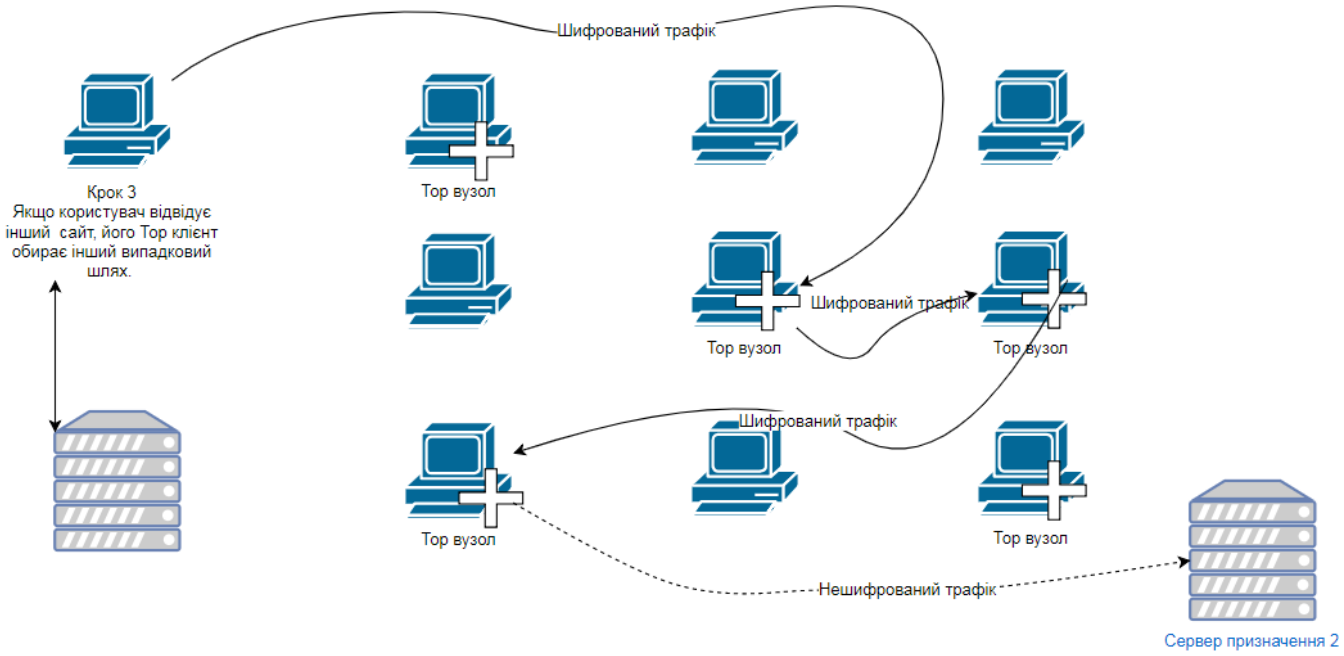


Рисунок 2.15. Крок 3, вибір іншого маршруту до серверу призначення 2

2.3 Аналіз принципів структурної побудови блокчейн мереж

Блокчейн, тобто ланцюжок блоків транзакцій (англ. Blockchain, від block – блок, chain – ланцюг) – розподілена база даних, яка підтримує перелік записів, так званих блоків, що постійно зростає. База захищена від підробки та переробки. Кожен блок містить мітку часу та посилання на попередній блок хеш дерева. [6]

2.3.1 Визначення складових “Blockchain”. Особливості роботи технології, основні принципи функціонування мережі та технологій в її складі.

Технологія Blockchain (укр. Блокчейн) стала дійсно винахідливим творінням думки людини, або групи людей, що працювали під псевдонімом Satoshi Nakamoto. З моменту винайдення та формулювання принципів роботи мережі ця технологія зазнала чималих змін та популярності в світі. Що ж саме нового привнесла ця технологія?

Можливість поширювати інформацію по мережі без її копіювання між учасниками мережі – так Блокчейн створив нову основу для нового типу всесвітнього інтернету. Оригінальна розробка технології була спрямована на винайдення нового слова в сфері цифрових валют – криптовалют, таких як Bitcoin (укр. (дослівно) Цифрова бітова монета), ETH (Ethereum), та інших. Але, з часом, спеціалісти в технічній сфері почали винаходити нові варіації і потенціали такого методу.

Блокчейн це незламний цифровий кластер для запису змін в мережі, що може бути запрограмовано не тільки на запис фінансових транзакцій, але й будь яких інших існуючих значень – будь-якої інформації в світі. (Don & Alex Tapscott, автори “Blockchain Revolution” (“Революція Блокчейн”, 2016 р.))

В основу технології блокчейн – роботу всіх механізмів закладено використання таких технологій та методів роботи і шифрування даних:

- Асиметричні алгоритми шифрування або «асиметричні криптосистеми» (пари “приватних” та “публічних” ключів);
- Хеш-функції або “хешування” даних (функції MD та SHA);
- Хеш-таблиці для запису результатів хешування – операцій в блоках транзакцій (використання хеш-дерева типу “Дерево Меркла”);
- Смарт-контракти (англ. Smart Contracts) – метод передачі даних (цифрових цінностей) від однієї особи до іншої;
- Токени та реалізація механізму Proof of concept (POC) – доказ концепції, як методу верифікації події (затвердження угоди) в системі.

Визначимо поняття кожного вище зазначеного терміну з переліку.

2.3.2 Асиметричні алгоритми шифрування – захист даних користувача, що передаються в мережу Blockchain

Асиметричні криптосистеми – ефективні системи криптографічного захисту даних, які також називають криптосистемами з відкритим ключем. В таких системах для зашифровування даних використовують один ключ, а для розшифровування – інший (звідси і назва – асиметричні). Перший ключ є відкритим і може бути опублікованим для використання усіма користувачами системи, які шифрують дані. [7]

Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних отримувач зашифрованої інформації використовує другий ключ, який є секретним (закритим). Ключ розшифровування не може бути визначеним з ключа зашифровування.

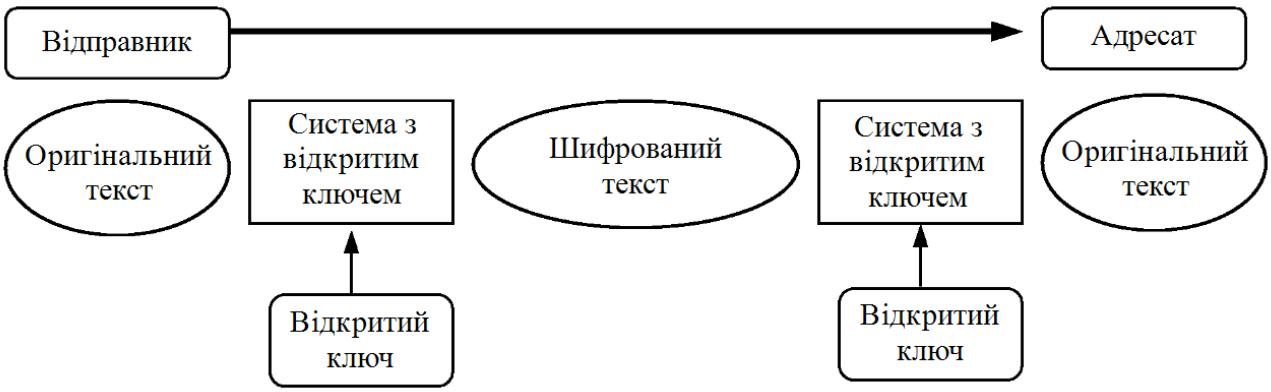


Рисунок 2.16. Схема передачі даних в асиметричних криптосистемах

Щоб гарантувати надійний захист інформації, системи з відкритим ключем (СВК) обов'язково мають відповідати двом очевидним та важливим правилам:

1. Перетворення вихідного тексту повинно бути незворотнім і виключати можливість відтворення зашифрованої інформації за допомогою відкритого ключа;
2. Визначення закритого ключа на основі відкритого повинно бути неможливим з врахуванням сучасних досягнень та можливостей обчислювальної техніки. При цьому, обов'язковою є точна оцінка складності (кількості операцій та часу) для зламу шифру.

Ідея криптографії з відкритим ключем тісно пов'язана з ідеєю односторонніх функцій, або таких функцій f(x), що знаючи значення аргументу 'x' достатньо легко знайти значення функції, тоді як визначення аргументу з функції досить складне в сенсі теорії. Тому, фактично, для знаходження аргументу з функції користувачеві необхідно мати додатковий спосіб облегшити розшифрування і мати спосіб легко відтворити початкове значення. Цим способом і виступає ключ користувача та на даному прикладі виступає значенням функції так, що f(x) = y, де 'y' – закритий ключ в системі СВК.

В цілому, всі сучасні криптосистеми з відкритим ключем використовують один з даних типів незворотних перетворень:

1. Розбиття великих чисел на прості множники;
2. Розрахунок логарифмічної функції в скінченному просторі;
3. Розрахунок коренів алгебраїчних рівнянь.

Також, кажучи про практичну цінність СВК, слід зазначити можливі застосування таких алгоритмів:

1. Як самостійні засоби захисту передавання та зберігання даних;
2. **Як засіб для розподілення ключів.** Алгоритми СВК достатньо складні у порівнянні з іншими традиційними криптосистемами і на практиці за допомогою СВК зручно поширювати ключі, об’єм інформації котрих незначний. А, згодом, за допомогою хмарних технологій здійснювати обмін великими інформаційними потоками.
3. **Як засіб автентифікації користувачів.**

Системи з відкритим ключем можуть використовувати найрізноманітніші алгоритми криптування. Одними з найпопулярніших при цьому є криптосистеми RSA, Ель-Гамаля або Діффі-Геллмана та криптосистеми на основі еліптичних рівнянь.

Фактично, вибір алгоритму лише визначає спосіб та складність шифрування ключа, а не принципову різницю між методом передачі інформаційних потоків.

Цифрова сигнатура або електронний підпис

Для перевірки того, що повідомлення або інформація дійсно належить тому чи іншому учаснику мережі, до повідомлень мають бути приписані так звані цифрові сигнатури.

Цифрова сигнатура або електронний підпис (англ. signature – підпис) – це рядок символів, що залежить як від відправника так і від змісту повідомлення (рис. 2.17).

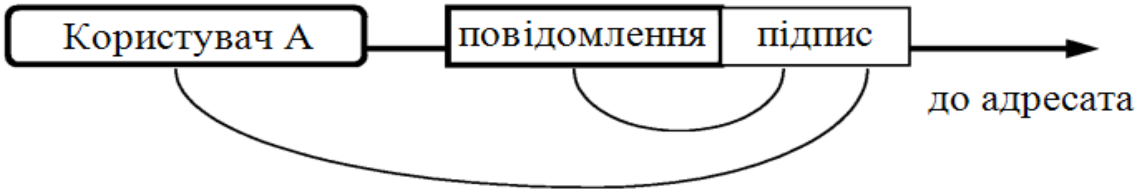


Рисунок 2.17. Цифрова сигнатура як частина самого повідомлення

Жоден учасник мережі крім користувача А (відправника) не може визначити формат підпису для кожного конкретного повідомлення. Жоден, включаючи самого користувача, не може змінити змісту повідомлення так, щоби підпис залишався незмінним. Хоч і отримувач повідомлення мусить мати змогу перевірки підпису на належність до відправника. Для перевірки валідності цифрового підпису користувач В (отримувач), має надати інформацію третій особі С (мережа або сервер верифікації підписів) про те, які самі дані було використано для перевірки сигнатури. Якщо повідомлення передається безпосередньо від відправника до адресата, виключаючи третю сторону, то в такому випадку мова йде про “самобутній цифровий підпис”.

Ряд недоліків наведеної вище моделі:

- В ланцюжку передбачається наявність третьої особи – клієнта котрому однаково довіряють як відправник так і отримувач;
- Відправник, отримувач та клієнт верифікації мають обмінятися істотною кількістю службової інформації до того, як буде передано саме повідомлення;
- Передача такої інформації має відбуватися у закритому вигляді, її використання в даному випадку малоефективне.

Тим не менш, навіть така схема цифрового підпису успішно використовується в цифрових системах, де має виконуватися два простих правила: необхідність автентифікації / інформаційного впізнання та обов'язкове шифрування повідомлень, що передаються в мережі.

2.3.3 Хеш-функції – дані Blockchain мережі; хешування – перетворення первинних даних у дані мережі

Використання цифрової сигнатури передбачає використання певних функцій шифрування:

$$S = H(k, T),$$

де S – сигнатура, k – ключ, T – оригінальний текст.

Функція H(k, T) при цьому є хеш-функцією якщо вона відповідає наступним вимогам:

1. Оригінальний текст може бути довільного розміру або довжини;
2. Значення H(k, T) має фіксовану довжину;
3. Значення функції H(k, T) легко розраховується для будь-якого аргументу;
4. Відтворити аргумент по значенню з точки зору розрахункової потужності та прикладених сил – майже неможливо;
5. Функція H(k, T) – однозначна.

Однозначність розділяють на слабку та сильну. При слабкій однозначності для заданого значення T майже неможливо знайти інший текст T′, для якого H(k, T) = H(k, T′). При сильній однозначності для будь-якого тексту T неможливо знайти інший задовольняючий текст, що мав би те саме значення хеш-функції.

Хешуванням (англ. hashing) при цьому називають перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями або функціями згортання,

а їхні результати називають хешем, хеш-кодом, хеш-сумою, або дайджестом повідомлення (англ. message digest), рис. 2.18.

З визначення хеш-функції випливає, що для будь-якої функції є тексти-близнюки, що мають однакове значення хеш-функції, тому що потужність безлічі аргументів незрівнянно більша потужності кількості значень.

Хеш-функції слугують з метою оптимізації даних за рахунок того, що у однакових значень (записів в базі даних) однакові значення хеш-функції. Такий підхід пошуку дублікатів ефективний для файлів великого розміру. Криптографічна хеш-функція дозволяє перевірити, що певні вхідні дані зіставляються із заданим значенням хешу, але, якщо вхідні дані невідомі, то відновити вхідне значення видається майже неможливим за умови знання лише про збережене значення хеш-функції. Даний механізм використовується для перевірки та забезпечення цілісності переданих даних, і є основним блоком для автентифікації повідомлень з використанням хешу.

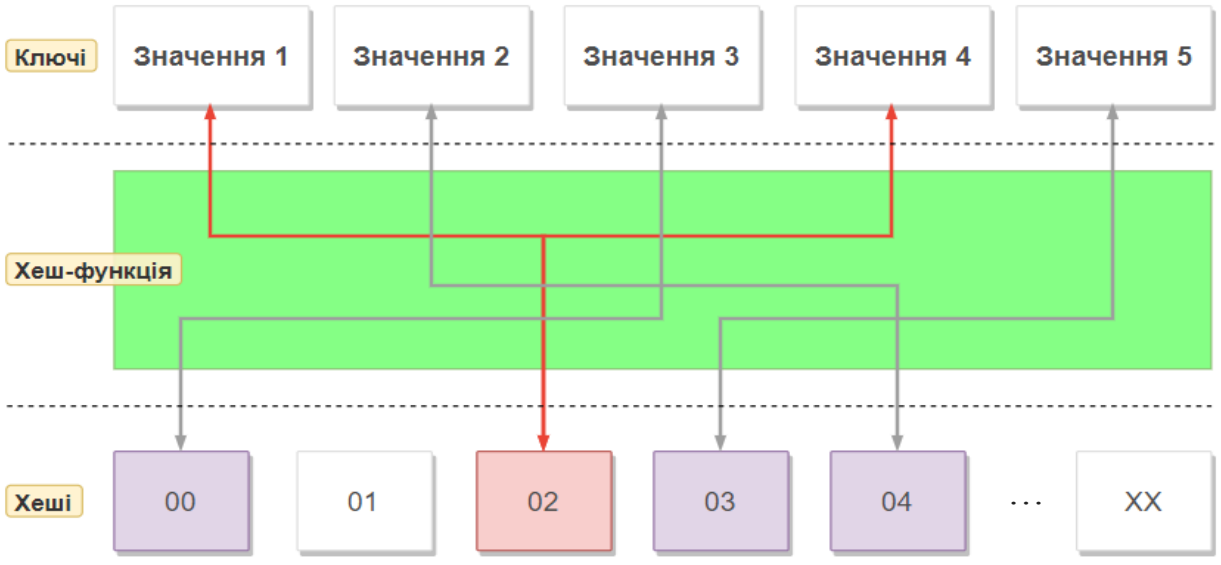


Рисунок 2.18. Хеш-функція ставить у відповідність значенням ціле число від 0 до XX. Є суперечність (колізія) між Значенням 1 та Значенням 4, яким відповідає однакове значення даних

Найбільш відомими хеш-функціями є MD2/4/5 та SHA.

Три алгоритми серії MD працюють за принципом перетворення тексту довільної довжини в 128-бітну сигнатуру. Вони отримали широке поширення в сучасних мережах як спосіб перевірки цілісності файлу або посилання за допомогою порівняння даних з розрахованим попередньо хешем.

Алгоритм MD5 передбачає:

1) доповнення тексту до довжини, що дорівнює 448 біт по модулю 512;
2) додавання довжини тексту в 64-бітному представленні;
3) 512-бітні блоки мають пройти процедуру Damgard-Merkle (цей клас перетворень передбачає розрахунок аргументів фіксованої довжини для фіксованих по довжині значень), при чому кожний блок проходить цю операцію в чотирьох різних циклах.

Сімейство алгоритмів SHA

SHA-256 – це алгоритм, або, іншими словами, криптографічна хеш-функція, яка була розроблена Агентством національної безпеки Сполучених Штатів Америки. Технічні параметри SHA-256:

- Показник розміру блока – 64.
- Максимально допустима довжина повідомлень (байт) – 33.
- Характеристика розміру дайджесту повідомлення (байт) – 32.
- Стандартний розмір слова (байт) – 4.
- Параметр довжини внутрішнього положення (байт) – 32.
- Число ітерацій в одному циклі – всього 64.
- Швидкість при стандартних умовах (MiB/s) – близько 140.

Робота алгоритму SHA-256 базується на принципі процедури Damgard-Merkle (так само, як і в випадку з алгоритмом MD5), в згідності з яким початковий показник відразу після внесення правок розділяється на блоки, а ті в свою чергу на 16 слів.

Даний протокол працює з даними, що розподілені на блоки по 512 біт (64 байти). Він виконує їх криптографічне “змішування” та на виході видає 256-бітний хеш-код.

Набір даних проходить через цикл, що нараховує 80 або 64 ітерації. Кожен етап характеризується запуском хешування зі слів, що складають блоки. Пара з них оброблюється за допомогою інструментів функції. Надалі, результати перетворення утворюють суму, що в результаті видають правильний показник хеш-коду. Для генерації кожного наступного блоку використовується значення попереднього. Перетворення цих блоків ізольовано – недопустима операція.

Основна робота даної хеш-функції полягає в хешуванні даних. Але для повного занурення в сенс та мету роботи даної хеш-функції слід визначити поняття терміну “майнінг” (англ. mining – видобуток).

Майнінг – головна компонента механізму захисту будь-яких цифрових валют. Принцип дії полягає в об’єднанні “майнерами” здійснених операцій в єдиний блок, який вже було перетворено сотні разів з метою встановлення виключно рідкісного хеш-коду, що відповідає певним встановленим вимогам мережі. Якщо подібне значення визнають знайденим, то блок “добувається” (тобто, потребує знаходження) і додається до блокчейну цифрової валюти. Така розрахункова діяльність не дає будь-якої користі окрім підвищення складності розрахунку та генерації необхідного блоку в подальшому. З іншої сторони, тільки завдяки їй користувачі даної системи можуть бути впевнені в тому, що їх платформу (блокчейн) не буде взято під контроль зловмисниками та централізовано. Мережа не має єдиного центру і розподілена, зменшується кількість переходів від одної точки до іншої, що також діє на благо усієї структури (рис. 2.19).

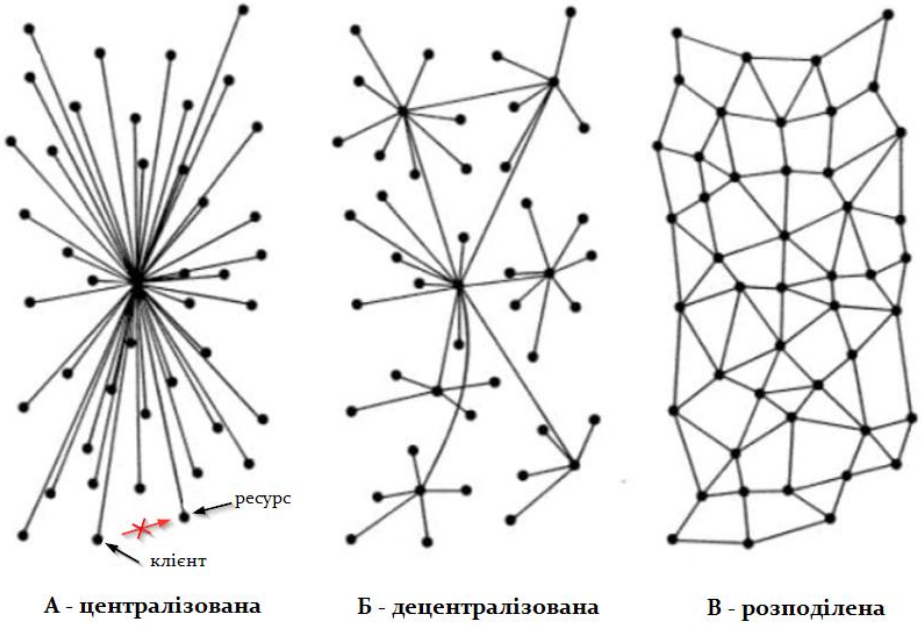


Рисунок 2.19. Види мереж за структурою з’єднань

Стандартна хеш-функція приймає на вхід блок з певною інформацією, видаючи на виході випадкове та непередбачуване значення. Вона розроблена таким чином, що не існує оптимального та однозначно вірного методу знайти необхідний показник без продовження перебору значень знову і знову до тих пір, доки не буде знайдено відповідний хеш-код.

Одним з найпопулярніших алгоритмів розрахунку блоків є саме SHA-256. Саме цим алгоритмом користується найдорожча криптовалюта в світі – Bitcoin. Причому, для підвищення рівню безпеки даний алгоритм задіюється два рази та іменується в даному випадку подвійним.

В Bitcoin критерієм оцінки правильності хешу вважають кількість “0” записаних на початку хешу. Віднайти таке значення також вкрай важко як і, наприклад, знайти банківський рахунок що містить декілька нулів наприкінці. Зрозуміло, що для хеш-функції дана операція в багато разів ускладняється і приблизному еквіваленті на даний момент дорівнює ймовірності знайти

необхідне значення в масиві з значень, що приблизно дорівнює 1,4 помножене на 10 у 21 ступені.

2.3.4 Хеш-таблиці як структури даних в блокчейні

Хеш-таблиця – структура даних, що реалізує інтерфейс асоціативного масиву, вона дозволяє зберігати пари (ключ та значення) і здійснювати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення за ключем.

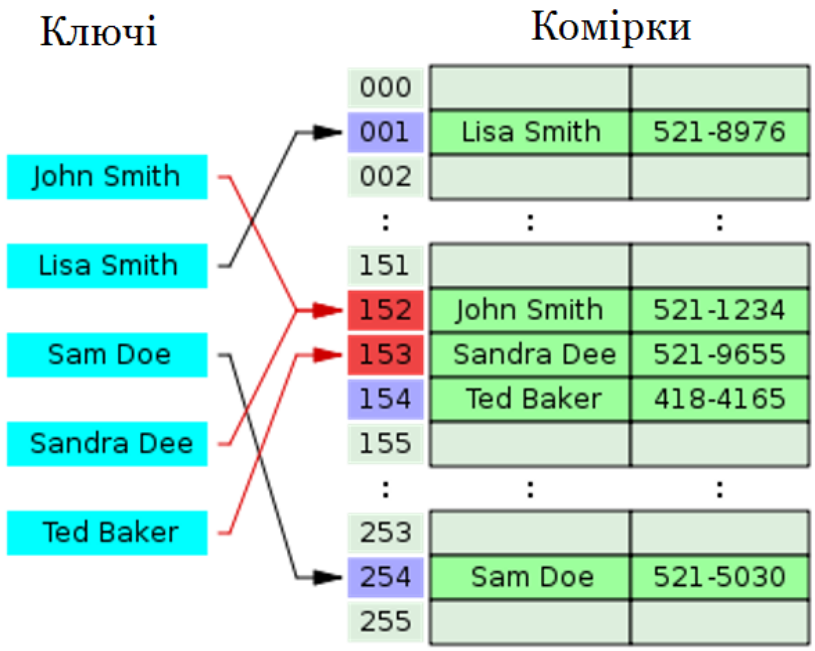


Рисунок 2.20. Розв’язання колізій хеш-таблиці методом лінійного перебору

Дерево Меркла (англ. Merkle tree) представляє собою особливу структуру даних (хеш-дерево), яка зберігає підсумкову інформацію про певний більший обсяг даних. Використовується для перевірки цілісності даних і застосовується для збереження хешів транзакцій у єдиний блок з мульти-хеш значенням. [8] Принцип, за яким це відбувається наведено на рис. 2.21.

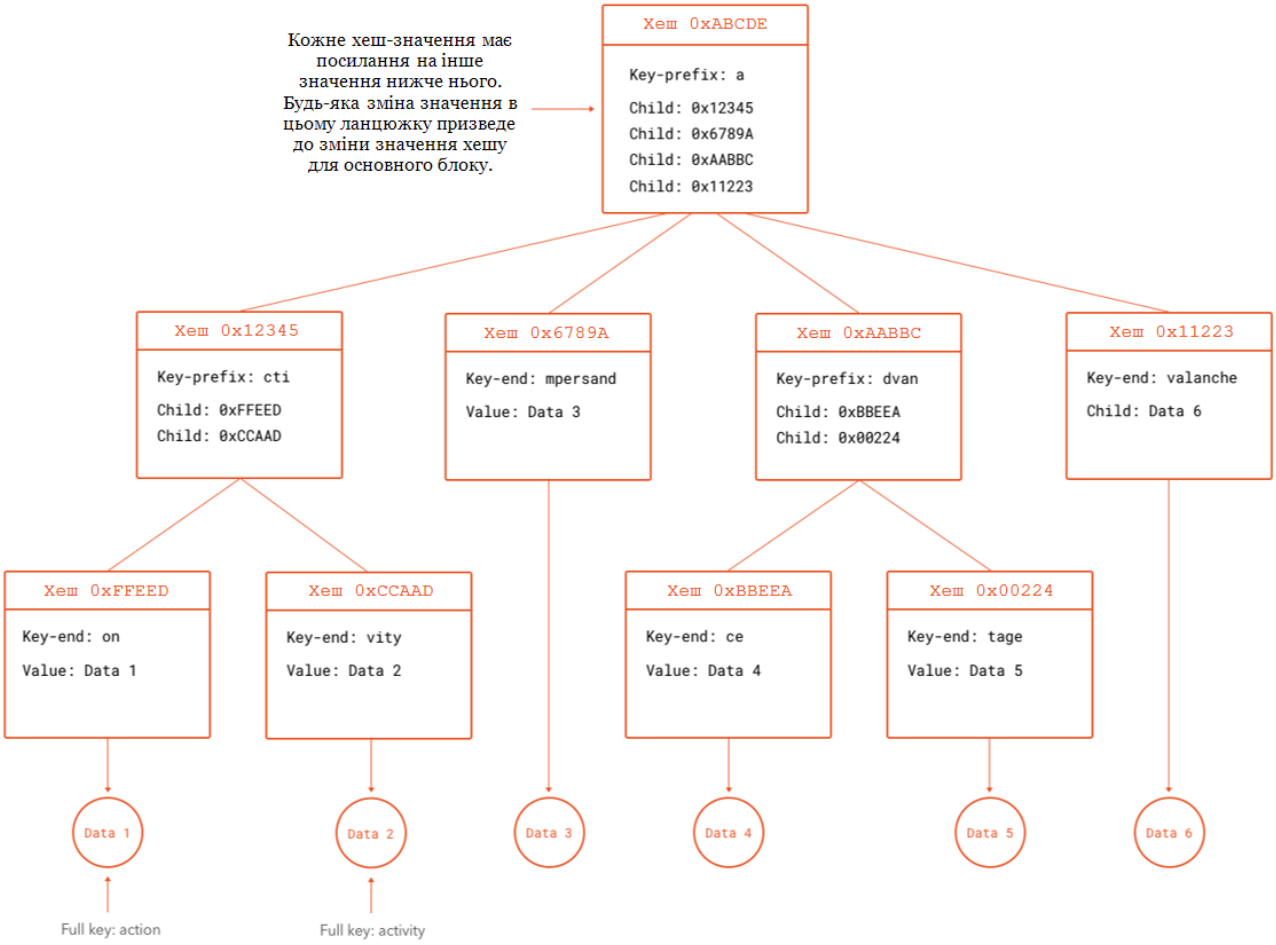


Рисунок 2.21. Побудова мульти-хеш блоку з дочірніх хеш-значень

2.3.5 Смарт-контракти як спосіб обміну цифровими цінностями (даними) в блокчейні

Усі криптовалюти успішно застосовуються для проведення щоденних валютних транзакцій. Але, ті ж самі мережі, де вони реалізуються, можливо використовувати в цілях розподіленої роботи програмного забезпечення та розповсюдження даних за принципами анонімності та простих договорів.

Для даних цілей в блокчейні створюється спеціальний об'єкт – смарт-контракт. Такі програми записуються в мережі та залишаються дійсними назавжди. Крім того, у всіх учасників мережі залишається копія проведеної операції. При цьому роботу контракту можна зіставляти з управлінням

грошовими операціями: створенням аукціону, гри з грошовою винагородою, парі, тощо.

Також, смарт-контракти відмінно пасують для автоматизації бухгалтерського обліку: контракт може записувати у собі, від кого і скільки прийшло грошей та планувати на цій основі фінансові прогнози на прибутки та втрати підприємства. Усі учасники мережі мають доступ до кількості операцій та їх призначення – блокчейн захищає від несанкціонованих та прихованих спекуляцій.

Розглянемо реалізацію смарт-контрактів на базі блокчейну Ethereum. В проектування цієї мережі, на відміну від блокчейну Bitcoin-у, з самого початку були закладені можливості впровадження даної технології та застосування її для розподіленої роботи програм на основі блокчейну, зокрема, як єдиної децентралізованої віртуальної машини Turing Complete. Фактично, Etherium це платформа для створення практично будь-яких децентралізованих онлайн-сервісів на базі блокчейну, що працюють на базі розумних контрактів. Зацікавленість даним проектом вже виявили такі компанії як Microsoft, IBM, Acronis та багато організацій-стартапів.

Смарт-контракт – це програмний код, який містить інформацію про транзакцію (або, простіше, угоду) у форматі “якщо … тоді …”. Наприклад, “Якщо користувачем X буде занесено в систему 100 ETH, тоді він отримає 10 токенів N від користувача Y”.

Токен – це внутрішня валюта компанії або особи, яка випускається та розповсюджується за методом залучення інвестицій. За дану валюту, в кожному конкретному випадку, інвестор може отримати різноманітні блага, що пропонує ініціатор компанії. Процедура проводиться на манер первинної публічної пропозиції акцій, за виключенням відсутності юридичних прав або державного регулювання поширення ICO (Initial Coin Offering, укр. “Первинне розміщення

монет”, маючи на увазі новий підвид цифрової валюти в котру вкладено певний процент прав та бонусів інвесторами проекту).

Тобто, повертаючись до вищевикладеного прикладу, якщо X та Y виконують свої зобов'язання, то кожен з них отримує попередньо визначений ресурс. Якщо хтось з учасників даної процедури спробує уникнути виконання своїх зобов’язань, то угода буде вважатися не завершеною і сторони залишаться при своїх інтересах та майнових правах.

Для мережі Etherium принцип роботи смарт-контрактів достатньо прозорий: актив потрапляє у програму і вона сама слідкує за виконанням умов угоди. Коли вони будуть виконані, то компанія (продавець) отримає крипто валюту у встановленому еквіваленті до товару, що перейде у власність інвестора (покупця). Основними атрибутами будь-якого смарт-контракту є:

- **Підписанти** – сторони домовленості, що приймають оговорені умови (для цього використовуються адреси облікових записів та цифрова сигнатура, або цифровий мульти-підпис, якщо підписантів контракту більше ніж два);
- **Предмет домовленості –** власне, ресурси для обміну (значення). При цьому, вони мають бути частиною системи, в рамках якої реалізується контракт. Якщо X та Y бажають скласти домовленість в мережі Etherium, то X повинен мати оговорену суму на рахунку, а Y – розмістити обмовлену кількість токенів в межах платформи;
- **Умови домовленості** – математично підтверджений опис умов (станів та функцій даного смарт контракту), за яких контракт буде вважатися можливим для виконання та функціональним. [9]

Графічне порівняння смарт-контракту та звичайного паперового смарт контракту наведено на рис. 2.22. Звичайний контракт не підкріплено жодним іншим доказом аніж папером, смарт-контракт – точно перевірена та прозора одиниця в складі мережі блокчейн.

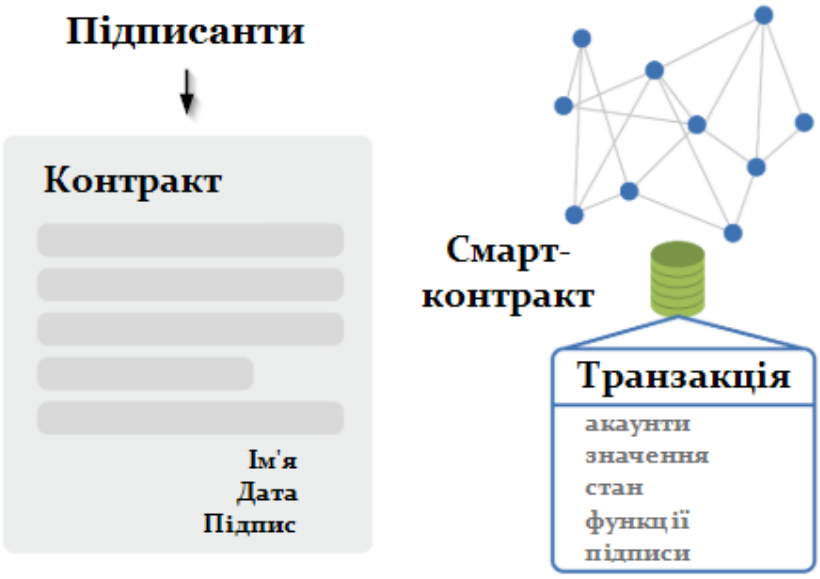


Рисунок 2.22. Звичайний контракт та смарт-контракт в спрощеному вигляді

Ключові переваги смарт контрактів

Доказ концепції (англ. Proof of concept, PoC) є реалізацією методу або ідеєю демонстрування здійсненності або демонстрації можливості виконання домовленості в принципі з метою перевірки того, що певна угода (інформація та дані, в широкому розумінні) мають практичний потенціал або достовірність та здійсненність. Доказ в такому випадку не обов'язково має бути повним або затвердженим в повному обсязі.

Одним з найголовніших критеріїв успіху смарт-контрактів є проведення угод без залучення третьої сторони (зазвичай, в реальному світі вони виступають гарантами виконання угоди та усіх умов домовленості). Смарт-контракти працюють за іншим принципом: лиш тільки реальні складові угоди потраплять в єдиний простір-платформу для обміну та мають відповідні електронні підписи сторін угоди – домовленість вважається закритою та відбувається автоматичний обмін цінностями.

Друга перевага – безпека та конфіденційність угод. Усі контракти зберігаються в блокчейні у зашифрованому вигляді. Про умови та предмет угоди знають тільки сторони домовленості, а зробити несанкціоновані зміни у програмний код виявляється неможливим на практиці.

Третьою перевагою смарт-контрактів є зниження витрат часу та матеріальних благ на проведення угод. Якщо усі умови домовленості виконані, то користувачі здійснюють обмін активами миттєво.

Але, не дивлячись на всі переваги, смарт-контракти мають і ряд правил, що мають бути виконаними для їх реалізації. Обов'язковими умовами в даному випадку виступають:

- Наявність децентралізованої інформаційної середи (блокчейн), в межах якої буде знаходитись достатня кількість клієнтів для отримання та виконання запитів у всесвітньому цифровому просторі;
- Наявність автоматичної бази даних (хеш-таблиці) для проведення транзакцій (укладання контрактів) в мережі;
- Наявність спеціальних інструментів та алгоритмів для виконання смарт-контрактів (хеш-функції), що б задовольняли вимогам безпечного розрахунку та однозначного визначення складових угоди, запису даних в БД;
- Використання методів асиметричного шифрування (закритого та відкритого ключів) за допомогою яких генеруються цифрові сигнатури клієнтів мережі.
- Математично доведена цілісність по Тюрінгу (можливість реалізації в системі будь-якої обчислюваної функції, що не порушує законів логіки даної системи).

Смарт-контракт – це, перш за все, програма. І, як і будь-яка програма, в ньому присутні певні недоліки:

- Складність самостійного впровадження смарт-контракту;

- Висока залежність від людського фактору (робота смарт-контракту та його безпечність ціликом залежать від правильності написаного програмного коду);
- Недостатня адаптивність (дані, що вже занесено в блокчейн, неможливо змінити);
- Погане масштабування. При одночасному запуску декількох контрактів пропускна здатність системи знижується пропорційно.

Також, знову слід зазначити, що використання смарт-контракту, як і здійснення будь-якої операції у блокчейні мають бути перевірені учасниками мережі.

Ether (ETH) це ‘паливо’ для мережі Ethereum. Коли існує необхідність надіслати токени, виконати операцію з смарт-контрактом, надіслати ETH або зробити будь що ще – ініціатор має заплатити за розрахунок такої операції. Ця плата розраховується в умовній одиниці Gas з конвертуванням у ETH.

Користувач завжди сплачує за розрахунки (перевірку хешу) незалежно від того, чи була успішною ваша операція, чи ні. Навіть якщо операція нездійсненна, майнери мають перевірити та виконати таку транзакцію (розрахувати) і саме за це має сплачувати ініціатор. Механізм дуже схожий до того, що відбувається з будь-якими банківськими операціями.

Користувач може переглянути прикріплені ‘чайові’ (gas limit × gas price) до утвореної транзакції з конвертацією по курсу. Саме ця сума буде винагородженням для майнерів, що роблять роботу з розрахунків та розміщують такі операції в блоках та захищають постійність блокчейну. [10]

Ознайомившись зі всім вищесказаним, час перейти до визначення самого блокчейну та принципів організації мережі на базі існуючого “всесвітнього павутиння”.

2.3.6 Блокчейн структура та її особливості

Уявіть таблицю значень яка дублюється тисячі разів у мережі комп'ютерів. Тепер, уявіть що ця мережа спроектована таким чином, щоб регулярно оновлювати цю таблицю даних. Ці два поняття дають базове уявлення про те, що саме складає основу технології Блокчейн.

Інформація що зберігається в блокчейні існує за принципом загальнодоступної та постійно обновлюваної бази даних. Фактично, це новий принцип використання мережі, що має свої переваги та недоліки. База даних блокчейну не зберігається на єдиному носії – це означає що всі його записи є дійсно загальнодоступними та легко верифікованими. Не існує жодної централізованої та основної копії яку б могли викрасти та зламати. Інформація поширюється через тисячі комп'ютерів одночасно, уся інформація блокчейну доступна одразу всім учасникам інтернету.

Технологія Блокчейн, так само як й інтернет, має свої вбудовані механізми захисту. Шляхом зберігання однакових блоків інформації по всій мережі, блокчейн:

- не може бути контрольованим єдиною організацією;
- не має єдиного “вразливого” центру для атак хакерів;
- постійно оновлює інформацію так, що не існує застарілих або більш нових копій даних.

Bitcoin було винайдено у 2008 році. З того часу блокчейн Bitcoin-у працює без жодних суттєвих проблем (на сьогоднішній день будь-які проблеми, пов'язані з транзакціями в блокчейні, були пов'язані з викраденням даних користувачів чи неправильним управлінням системою. Інакше кажучи, ці проблеми випливають з людських помилок, а не з недоліків основних принципів роботи системи). Але існує декілька видів блокчейнів з іншими концепціями.

Структура Ethereum блокчейну має дуже схожі принципи до вищезазначених: вона зберігає записи про все що тільки відбувається в мережі. І

при цьому кожен учасник має локальну копію цих даних разом з історією усіх подій.

Велика різниця двох блокчейнів полягає в тому, що ноди зберігають останній стан кожного смарт-контракту додатково до значень і транзакцій в мережі. Для роботи будь якого ПЗ в мережі Ethereum треба вислідити стан або нинішні показники інформації для всіх подій, включаючи баланс кожного з користувачів, код смарт-контракту та як саме до нього можна звернутися.

В блокчейні біткоіну використовується трохи інший принцип: кожного разу як відбувається якась подія, мережа ‘забирає’ необхідні дані та розраховує їх за принципом схожим до реальних валют визначаючи, яка кількість будь-чого існує та кому саме належить. Таким чином відбувається маркування на витрачені та здобуті дані.

Ethereum мережа усе виконує за принципом створення облікових записів (акаунтів) користувачів та точно зберігає, які саме події відбуваються з даним ‘віртуальним гаманцем’. Усі дані завжди перебувають десь у мережі, але не мають точного зв’язку з будь-ким для продовження лінії операцій з ними (рис. 2.23).

Блокчейн система постійно погоджена – кожні десять хвилин вона перевіряє сама себе. У вигляді само-контрольованої вбудованої системи в складі блокчейну працює механізм, що перевіряє та затверджує всі транзакції в даному інтервалі. Кожна група таких транзакцій оцінюється як один “блок”. З цих двох властивостей системи випливає:

- дані про прозорість вбудовуються в мережу в цілому, за визначенням вона є загальнодоступною;
- мережа не може бути зламаною одним з учасників мережі - для внесення змін у мережу знадобиться участь чималої кількості розрахункових потужностей в складі блокчейну.

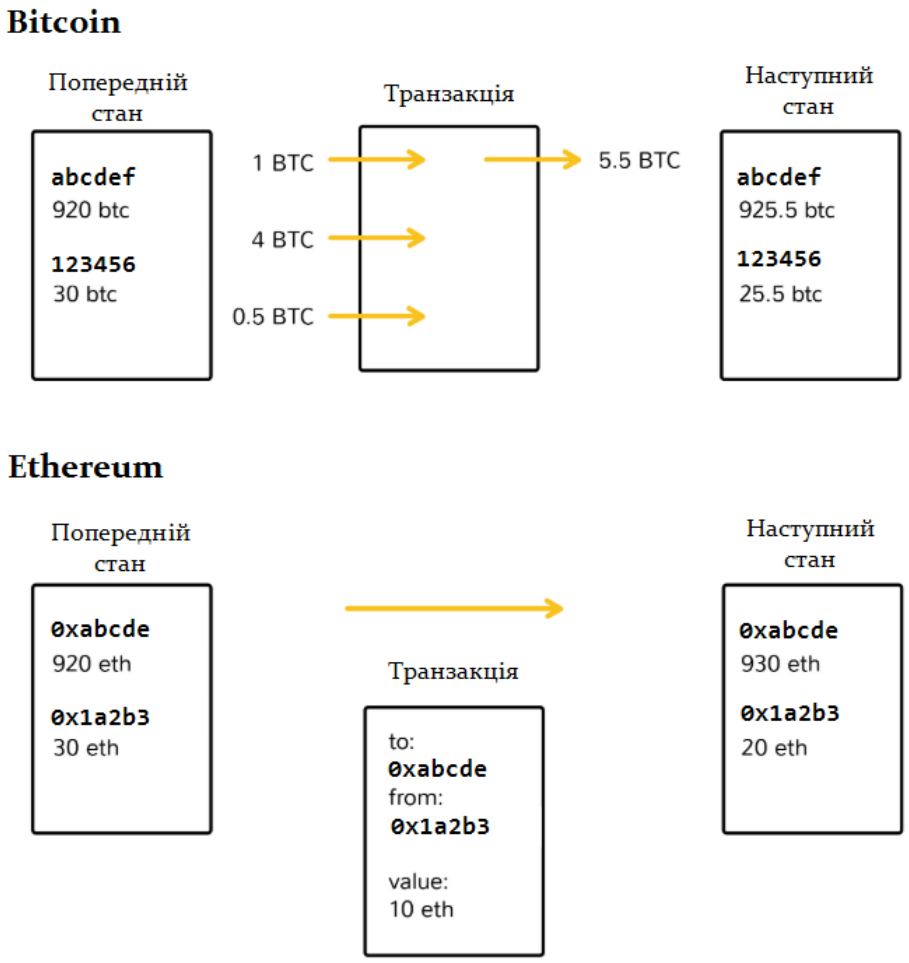


Рисунок 2.23. Різниця в принципах транзакцій в Bitcoin та Ethereum блокчейнах

Приватний ключ, так само як і пароль, дає доступ користувачеві до зашифрованих персональних даних. І безпека цих даних цілком і повністю буде належати особі, яка володіє цим ключем, без нього доступ до даних буде втрачено.

Мережа з комп'ютерів-учасників, так званих "нод" або "вузлів" і складає усю блокчейн систему. Вузол – комп'ютер під'єднаний до Blockchain мережі, що використовує клієнт (програму) за допомогою якого виконує задачі перевірки та передачі інформації про транзакції. Він отримує копію блокчейну, що завантажується автоматично під час підключення до загальнодоступної

мережі. Разом усі вузли формують потужну мережу другого рівня – зовсім інакше представлення того, як функціонує інтернет на сьогоднішній день. [11]

Кожен вузол виступає адміністратором мережі і під'єднується до неї добровільно (в цьому сенсі уся мережа залишається децентралізованою, рис. 2.24). Крім того, кожен вузол в блокчейні бере участь в тому, що може отримати винагороду у вигляді зарахування частини криптовалюти на власний рахунок.

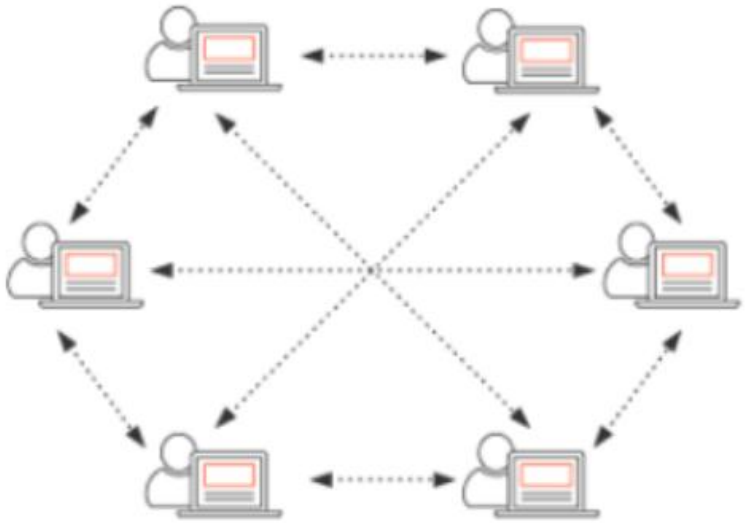


Рисунок 2.24. Схема зв'язків клієнтів в блокчейні

Вузли часто представляють як центри для "майнінгу" криптовалюти (укр. – добування; такий термін застосовують у зв'язку з необхідністю прикладати певні технічні потужності для отримання винагород в системі блокчейн), але це поняття не зовсім вірне. Фактично, ці ноди постійно конкурують між собою за отримання винагороди шляхом вирішення математичних виразів (знайдення необхідного хешу). Криптовалюта є основним сенсом існування таких нод в системі, ідейним елементом розвитку blockchain. Зараз, цифрові валюти існують лише як перший, але не єдиний спосіб застосування цієї технології.

За приблизними оцінками, зараз існує близько 1600 Bitcoin-подібних криптовалют (і їх кількість постійно зростає). Також, на даний момент в розробці існує цілий ряд інших потенційних адаптацій оригінальної концепції мережі блокчейн.

Так само як і з веб-структурами, користувач не мусить знати все про блокчейн для його використання. На даний момент, фінансова галузь пропонує найбільший перелік сценаріїв з використання технології. Міжнародні грошові перекази, наприклад. За оцінками світового банку в 2015 році було надіслано близько 430 мільярдів доларів США таким способом. І на даний момент ця галузь шукає нових розробників проектів, blockchain розробників.

Потенційно, блокчейн усуває будь-яких посередників при грошовому переказі, користувачеві доступний графічний інтерфейс користувача (GUI) через власний персональний комп’ютер або інший пристрій. Аналогічним функціоналом зараз оперують так звані “гаманці” для різноманітних криптовалют на ринку.

Висновки до розділу

1. Визначення видив побудови топологій VPN та їх основних особливостей вказує на прийнятність даних рішень для використання в складі структури блокчейн. Ці рішення можуть бути успішно застосовані при розробці основи комунікації клієнтів, як рівень транспортного зв’язку та мають допомогти у створенні основи для успішної розбудови мережі.

2. Складові структури блокчейн та особливості її роботи можуть бути використані для створення сучасного рішення у сфері приватних та надійних у роботі клієнтів VPN з повноцінними механізмами забезпечення конфіденційності користувачів, вбудованими засобами збереження оригінальності даних та можуть створити нові принципи роботи з інформаційними потоками в мережі інтернет.

3. Виконавши аналіз структури мереж блокчейн та усіх її структурних особливостей, основою для наших цілей будемо використовувати блокчейн Ethereum як найбільш привабливий варіант з вбудованими механізмами та можливостями для імплементації власного програмного забезпечення, що працюватиме на його основі.

3. МЕТОД ПОБУДОВИ ТА ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ СМАРТ-КОНТРАКТУ В МЕРЕЖІ БЛОКЧЕЙН. СТВОРЕННЯ ВЛАСНОГО МЕХАНІЗМУ ПЕРЕДАЧІ ІНФОРМАЦІЇ В СКЛАДІ СМАРТ-КОНТРАКТІВ НА БАЗІ ПЛАТФОРМИ ETHEREUM

Метою даного розділу є створення та формування принципів роботи VPN сервісу на базі Blockchain-у Ethereum з виконанням аналізу доцільності використання такого рішення, описом процесів, що мають відбуватися при передачі даних та їх реалізації, встановленні діалогів та сервіс-сесій в рамках даної концепції. В рамках даного розділу визначимо спосіб тестування запропонованого механізму та створимо детальну інструкцію, що описуватиме необхідні етапи для проектування та подальшого розвитку ідеї. В результаті виконання вищезазначеного маємо отримати цілісний опис реалізації, визначити її переваги та недоліки.

3.1 Структурний опис і призначення основних механізмів в мережі

Кожен механізм чи модель роботи сервісу має бути чітко визначеним на процесі створення концепції. У зв’язку з тим, що з технічної точки зору реалізація механізму VPN мережі в рамках Blockchain досить складна (в плані популяризації та відладці усіх процесів) – будемо керуватися метою створення власної концепції та формування принципів роботи такої мережі.

Завданням поставимо опис механізму з технічної точки зору, імплементацію смарт-контракту (використовується об’єктно-орієнтована мова програмування Solidity) з підтвердженням факту передачі даних та пропозицією реалізації локального клієнту (програми) з обробки запитів та відповідей.

Реалізацію здійснюємо шляхом налаштування тестового оточення на персональному комп’ютері (ПК) з імітацією реального блокчейну без обмежень в доступних ресурсах для проведення бажаної операції в дійсності.

3.1.1 VPN over Ethereum Blockchain (VEB)

VPN мережа на базі Ethereum Blockchain – це швидкий та масштабований рівень транспорту, технологія, що має забезпечувати безпеку даних користувачів. Даний механізм, потенційно, може слугувати для побудови децентралізованої мережі точка-точка (англ. Point-to-point, P2P) з використанням багаторазових переходів, релейного принципу та інтегрованої мережі на базі токен-моделі. Фактично, це децентралізована VPN мережа, що працює на базі блокчейну Ethereum. Це мережа з відкритою кодовою платформою, що надає можливість “надавати в оренду” свій невикористаний трафік забезпечуючи при цьому захищене з’єднання для усіх охочих учасників даної мережі.

Для того, щоб оцінити переваги та недоліки такого механізму, спробуємо виконати порівняльну характеристику стандартних VPN мереж від провайдерів сервісів, системи TOR та пропонованого нами варіанту (табл. 3.1).

Таблиця 3.1 – Порівняльна характеристика звичайних VPN мереж провайдерів, системи TOR та мережі VPN over Ethereum Blockchain

	Централізована VPN мережа	**Система TOR**	**Мережа VEB**
Анонімність	Ні	Так	Так
Децентралізована передача трафіку	Ні	Так	Так
Можливість end-to-end шифрування	Ні	Так	Так
‘Honeypot’ вразливості	Висока	Низька	Низька
Стимулювання учасників мережі	Ні	Ні	Так
Відкрита платформа	Ні	Так	Так
Потенційна швидкість	Висока	Низька	Змінна
‘Платформа як сервіс’	Ні	Ні	Так

Виходячи з даних наведених у таблиці, робимо висновок, що пропонований механізм на базі Blockchain-у може створити справжню конкуренцію для існуючих на сьогоднішній день варіантів реалізації анонімного та безпечного доступу до всесвітньої мережі та, теоретично, має започаткувати клас абсолютно нових мереж з власними принципами комутації та обміну даними, побудови діалогів та встановлення сесій між клієнтами.

Така реалізація є симбіозом технологій, що використовуються в звичайних IP мережах і розташована за межами моделі OSI та має власну класифікацію.

3.1.2 Обмін даними між нодами в мережі

Канали повідомлень дозволяють нодам проводити обмін повідомленнями. Одночасно може існувати декілька типів каналів обміну повідомленнями. Кожен тип каналу може використовувати інший протокол і схему зв'язку (вузол-вузол, передача через накладення на централізований сервіс, передача через накладення p2p). Діалог між вузлами може бути запущений через будь-який канал повідомлень, доступний для обох вузлів, що беруть участь у діалозі. Один тип каналу та відповідний протокол зв'язку має бути визначено за замовчуванням та має підтримуватися всіма мережевими клієнтами. Реалізація каналів повідомлень забезпечує ненадійний детаграмний інтерфейс передачі повідомлень. Деякі типи каналів можуть здійснювати транзитивну маршрутизацію повідомлень, і тому необхідні заходи для захисту змісту повідомлень. Кожне повідомлення, що надіслане по каналу, підписується відправником, а потім зашифровується за допомогою інтегрованої схеми шифрування за допомогою криптографічного алгоритму RSA-128 (128 – кількість бітів ключа).

Такий спосіб захисту обміну повідомленнями перешкоджатиме несанкціонованому доступу до вмісту повідомлень, але все ще є вразливим для

визначення ідентичності сторін, що спілкуються. Але, зважаючи на відсутність чіткого зв'язку між клієнтом мережі (обліковим записом, створеним на запит) та самою людиною, визначення клієнтів – унікальних комбінацій номерів облікових записів, що "спілкуються", не є критичним в сенсі викриття особи та ідентифікації запитів, що вона здійснює до віддалених інформаційних ресурсів. Навіть, знаючи про зміст діалогу, спів ставити певну інформацію з певним клієнтом мережі, яка потенційно зростатиме в обсягах та загальної кількості користувачів – майже неможливо.

3.1.3 Діалоги в мережі

Діалоги забезпечують структурований спосіб обміну інформацією двох клієнтів (англ. peers) і вказують на структуру, необхідну для організації платежів за надані послуги та створення сервіс-сесій. Згідно з визначеною структурою, лише один діалог може існувати між двома клієнтами в будь-який момент часу. Якщо один з них спробує почати новий діалог – попередній діалог буде припинено.

Діалоги забезпечують надійну передачу повідомлень через ненадійні канали Інтернет шляхом реалізації простої схеми позитивного підтвердження з'єднання та дають можливість подальшої реалізації для повторної передачі втрачених повідомлень і даних, що містяться у них (рис. 3.1).

Встановлення діалогу відбувається тоді, коли один з учасників мережі надсилає запит на встановлення діалогу іншому клієнту, а той, в свою чергу, підтверджує спосіб встановлення зв'язку між ними. Два клієнти обов'язково мають підтвердити ідентичність один-одного та оновити локальні таблиці "довірених" вузлів з метою спрощення та пришвидшення ймовірних у подальшому з'єднань.

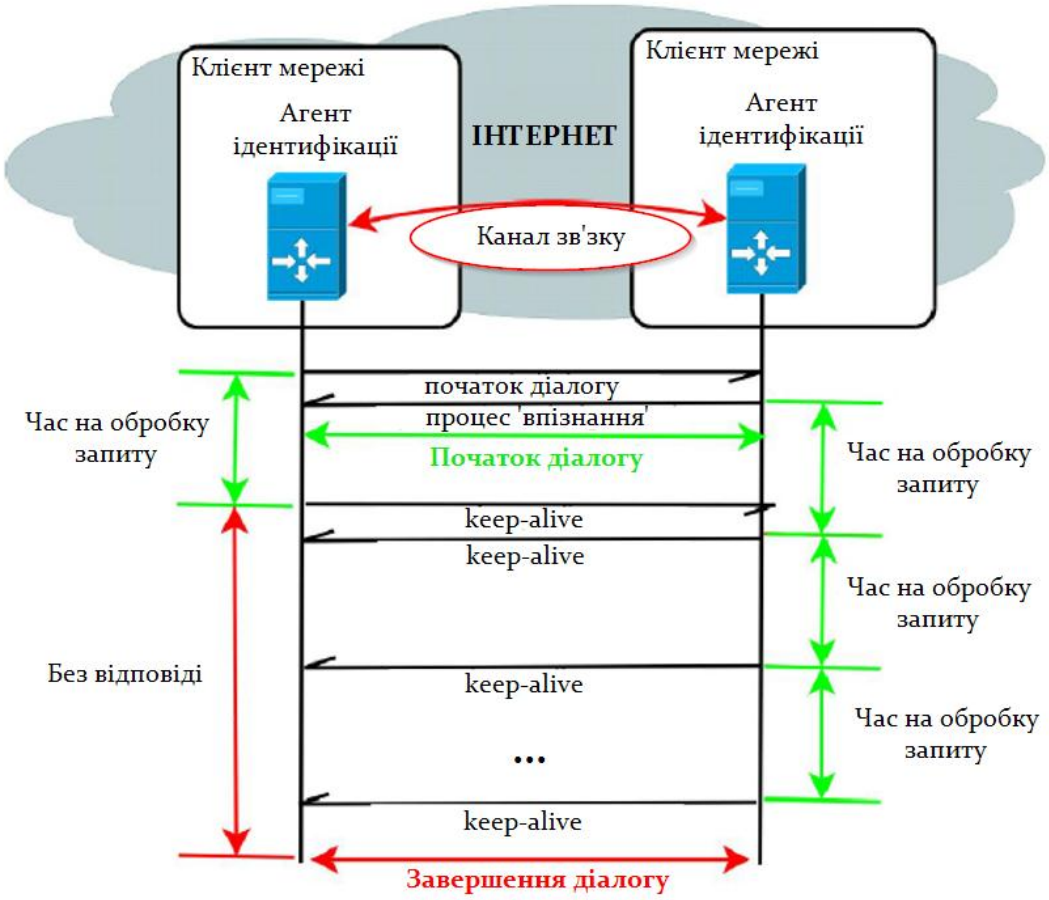


Рисунок 3.1. Життєвий цикл діалогу

Для того, щоб відкрите діалогове вікно продовжувало існувати між двома вузлами мережі, передбачена необхідність надсилання регулярних повідомлень про те, що вузол досі на зв’язку. Якщо вузол не має нічого для відправки, він все одно повинен відправляти повідомлення про своє існування (англ. keep-alive). Агенти ідентифікації (вони ж – клієнти зі спеціально встановленим ПЗ) контролюють, коли востаннє було отримано таке повідомлення від ноди на іншому боці, і за встановлених правил (не більше 3-ох пропусків відповіді на запити типу “keep-alive”) можуть вирішити закрити встановлене діалогове вікно. Ідентифікація може припинити існуючий діалог і просто ігнорувати подальші, будь-які повідомлення, що надходять з боку віддаленого клієнта. При цьому, для відновлення діалогу, двом клієнтам необхідно знову пройти всі

етапи встановлення з’єднання та ідентифікації з самого початку. З боку принципів роботи блокчейну, клієнти мережі працюють за схемою ‘Доказу роботи’ (англ. Proof Of Work, PoW), коли для підтвердження будь-якої операції дана операція вже мусить бути частиною блоку.

3.1.4 Встановлення сервіс-сесій між учасниками мережі

Щоб бути повноцінним конкурентом на ринку, служба VPN повинна забезпечувати інтерфейс для роботи та комунікації програм користувачів і мати різні канали передачі даних, створені для вузла або декількох вузлів. Цей інтерфейс є типовим для даного типу сервісів в мережі інтернет – може проявлятись як інтерфейс https протоколу, коли використовується IP-тунельний сервіс, або це може бути певний сокет, що прослуховує локальну IP-адресу, коли використовується служба “secured socks”. Канал передачі даних, який використовується для передачі трафіку створений програмою користувача, також може бути специфічним для програми. Запропоноване рішення може підтримувати багато служб у складі VPN-послуг, і таким чином, повідомлення, відправлені для встановлення сервісних сеансів, мають бути достатньо гнучкими, щоб передати специфічну інформацію до конкретних програм користувачів.

Для використання певної служби вузол має підтримувати конкретний тип служби VPN. Ця підтримка може надійти у вигляді вбудованої функціональності вузла або як додатку до вузла програмного забезпечення. Один або кілька одночасних сеансів можна відкрити в рамках одного діалогу. Усі сеанси будуть закриті провайдером у випадку зупинки діалогу.

Щоб запустити сеанс сервісу, клієнт повинен надіслати запит на створення сеансу. У запиті має міститися пропозиція методу, який використовується для виявлення служби, ідентифікатор з’єднання, який використовується для співставлення відповіді та даними програми, що

приймаються. Дані про конкретну інформацію повинні перевірятися клієнтським програмним забезпеченням служби VPN. Після отримання цього запиту співбесідник може відповісти на прийняття чи відмову. У тому випадку, якщо відкриття сесії було відхилено, також подано причину відмови. У випадку, якщо відкриття каналу прийнято, ідентифікатор сесії постачається та супроводжується специфічними даними каналу, що встановлюється (рис. 3.2).

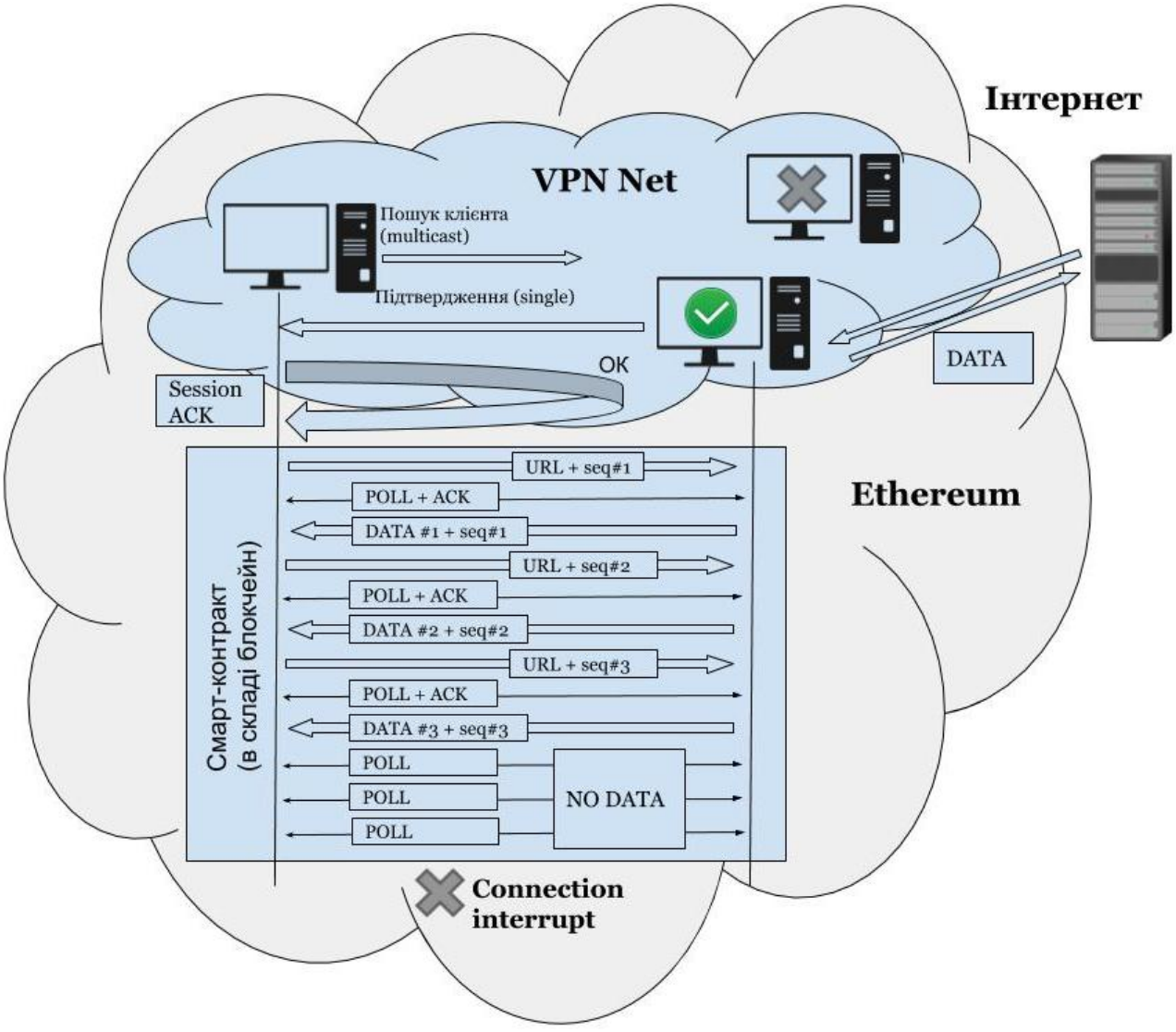


Рисунок 3.2. Встановлення сесії в межах мережі

3.2 Аналіз процесу обслуговування абонентів в Blockchain мережі

Запропонований метод побудови віртуальної приватної мережі полягає у створенні розподіленої, децентралізованої мережі пристроїв, що з'єднані за допомогою каналів Інтернет та спеціальним програмним забезпеченням встановленим локально. Встановлене ПЗ має:

- Створювати приватний/публічний ключі для шифрування трафіку та присвоєння його ноді (клієнту) мережі;
- Створювати, зберігати та оновлювати локальну копію бази даних, що представляє собою хеш-таблицю;
- Містити базу даних довірених вузлів мережі, що дозволяє використати пропускну здатність певної ноди (наприклад, користувач ПК оплатив послуги провайдера, але чимала кількість трафіку не використовується за один період оплати послуг провайдера Інтернет);
- Проводить аналіз мережі та пошук здатних до передачі трафіку хостів.

Система працює використовуючи власний алгоритм обміну даними: за надану можливість передачі трафіку користувач отримує певну кількість цифрових одиниць цифрової валюти (токенів), що в подальшому можливі для використання з ціллю придбання можливості ретранслювати власний трафік.

Мережа є захищеною, оскільки кожен блок інформації, що передається, зашифровано за асиметричним алгоритмом. Кожен блок інформації, що має бути переданим по мережі, зашифровано, підписано власником та зберігається в єдиній, децентралізованій системі.

3.3 Підготовка тестового оточення

Для створення та перевірки роботи власного алгоритму було створено тестове оточення, що включає у себе дві тестові машину на базі операційної системи Windows 10 та Linux x64, встановлюване ПЗ для запуску локальної ноди та текстовий редактор з метою написання та імплементації власного

смарт-контракту і обробника запитів, що запускається паралельно з запуском клієнта блокчейн.

3.3.1 Налаштування та запуск локальної ноди

Фактично, Ethereum нода – це будь-який пристрій, що в змозі підтримувати Ethereum протокол (блокчейн). Зазвичай, цими девайсами виступають персональні комп'ютери або ноутбуки, тоді як розробка для мобільних пристроїв все ще триває. Для того щоб стати учасником блокчейну необхідно мати спеціальні інструменти та програмне забезпечення для запуску локального клієнту та отримання даних з мережі. Завдяки цьому ми зможемо підключитись до інших нод в мережі та мати прямий доступ до інформації, що зберігає Blockchain, приймати участь в знаходженні нових блоків, **створювати нові транзакції та впроваджувати смарт-контракти.**

ПЗ клієнтів написано на мові програмування Go (Go Ethereum / Geth), C++ та Python. Найбільшою популярністю користується Geth, а, отже, це наш вибір також. Встановлення Geth, його завантаження на локальну машину, значить і завантаження усього блокчейну Ethereum, який має чималий розмір (більше 100 Gb). Тому, скористаємось варіантом створення власної локальної мережі з власними правилами і можливістю впроваджувати смарт-контракти та проводити транзакції без необхідності вкладення реальних розрахункових потужностей або матеріальних цінностей. Завантаження клієнту можливе з веб-сайту розробників (Go Ethereum): https://geth.ethereum.org/downloads/ (рис. 3.3).

За замовчуванням, для операційної системи Windows 10, дана програма буде встановлена у директорію C:\Program Files\Geth (рис.3.4).

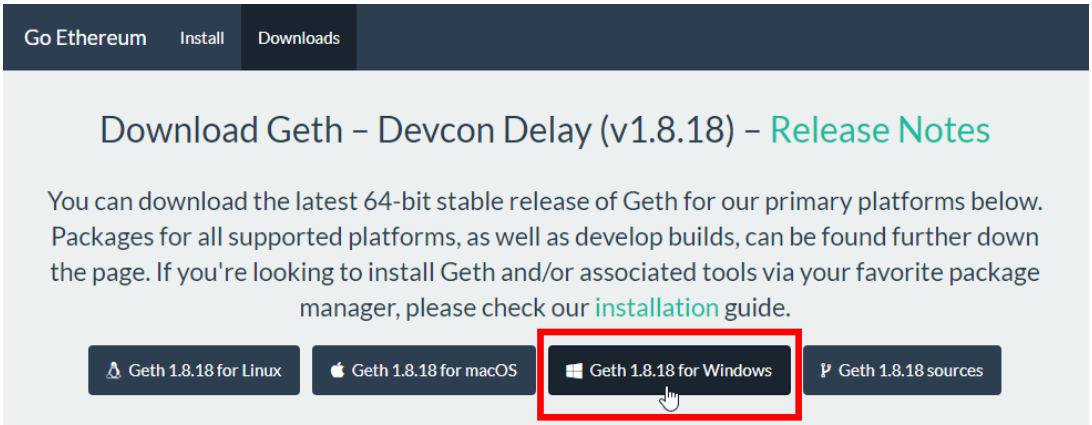


Рисунок 3.3. Завантаження Geth з сайту розробників для ОС Windows

Отже, маючи встановленим інструмент на машині можемо створити локальне оточення для запуску Blockchain механізму локально. Для цього ми маємо запустити власний приватний сервер. Таким чином, ми зможемо вільно впровадити та перевірити власне рішення без необхідності під'єднання до публічно доступної хеш-таблиці. А, отже, ми почнемо з самого початку і знаходження самого першого блоку у нашому блокчейні (англ. Genesis block).

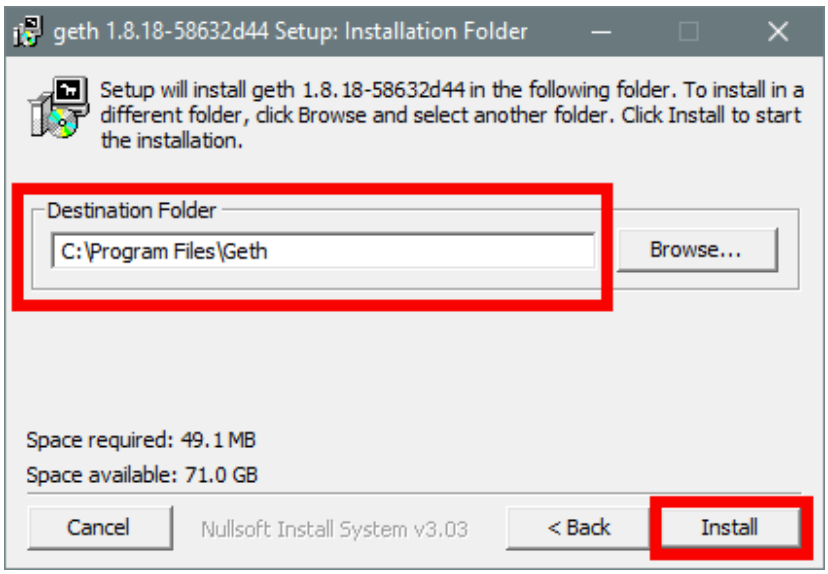


Рисунок 3.4. Встановлення Geth на локальну машину

Для змоги керувати роботою програми необхідно запустити інтерфейс командного рядка (англ. command-line interface, CLI), вбудований в ОС Windows або ж скористатися додатковим ПЗ з відкритим серцевим кодом (в нашому випадку – GitBash: https://git-scm.com/downloads). За допомогою команди cd (change directory, укр. “Змінити директорію”) робимо перехід до теки з встановленим ПЗ Geth (рис. 3.5).

Далі, нам необхідно створити теку, яка буде слугувати для збереження даних локального блокчейну і вказати ПЗ Geth її місцезнаходження разом з правилами, яким буде слідувати ця мережа. Для цього створимо в корені головного системного диску пристрою нову директорію і дамо їй назву ‘LocalChain’, в ній створимо теку ‘chaindata’, де будемо зберігати дані блокчейну, та файл genesis.json , в який занесемо такі параметри:

```
{
  "coinbase"   : "0x0000000000000000000000000000000000000001",
  "difficulty" : "0x40000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00",
  "alloc": {   }
},
  "config": {
        "chainId": 1996,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0     }
}
```

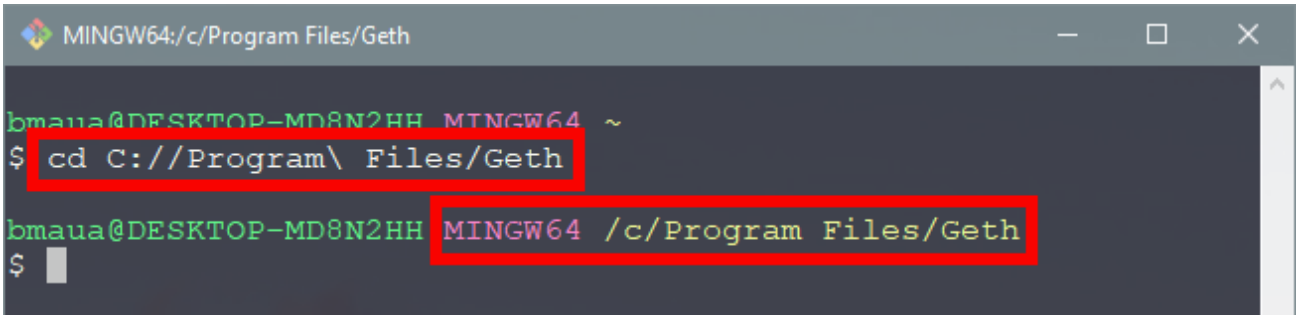


Рисунок 3.5. Зміна робочої директорії за допомогою команди cd у CLI

config

Структура `config` в файлі genesis.json керує змінними блокчейну, і не має впливу на сам блок. Тим не менш, ці значення важливі через те, що вони мають співпадати з конфігурації усіх нод в одній мережі.

1. **config: chainID**

Дана зміна декларується з метою повідомити світу, який саме ланцюг (англ. chain) використовується вами. chainID для головної мережі Ethereum – 1, і це найшвидший спосіб повідомити всім клієнтам в мережі “Я хочу працювати поряд з іншими в головній мережі” або “Я хочу виконувати операції в своїй власній підмережі яка буде створена саме для мене”.

chainID було представлено в рамках EIP155. EIP (Ethereum Improvement Proposal; укр. ‘Пропозиція з покращення Ethereum’) – підвид реалізацій роботи механізмів в мережі Ethereum, що були запропоновані відкритим співтовариством в формі дискусії та кодової реалізації. Причина додавання цього параметру – зробити операції в відмінних мережах відмінними від загальної. Усі операції мають свій власний підпис в залежності від використовуваного chainID.

2. **config: HomesteadBlock**

HomesteadBlock (укр. (досл.) – ‘Блок у основи’), зі встановленим значенням в нуль означає те, що використовуватиметься основна і друга версія

релізу платформи Ethereum. Це очікуване значення в даній конфігурації, і саме тому також встановлене у нульове значення (0 – працює за замовчуванням).

3. config: EIP155Block

EIP155, крім того, було прийнято з метою зупинити можливість атаки через втручання в процес відповідей в мережі. 0 – працює за замовчуванням.

4. config: EIP158Block

EIP158 було прийнято з метою змінити те, як клієнти Ethereum поводять себе з пустими обліковими записами. Цей новий протокол призвів до того, що такі записи почали відмічати як неіснуючі, зберігаючи об'єм інформації, що зберігається в блокчейні. 0 – працює за замовчуванням.

alloc – визначає початкові адреси учасників блокчейну та їх рахунки (якщо така потреба існує).

difficulty – значення, що визначає складність формування блоку в мережі. Наше значення вибрано в 0x4000 в 16-чній системі, що дорівнює 1024 в деїстичній системі числення. Це означає, що одна з 1024 знайдених хеш-функцій буде вдалою (такою, що вважається вірною для даного блокчейну). Реальні машини на базі графічного ядра можуть розраховувати до 50 – 60 Мега хешів у секунду.

mixhash, nonce – використовуються разом для визначення того, чи правильно було визначено та записано блок у ланцюжок. Nonce, фактично, є унікальним ідентифікатором блоку який було б достатньо тяжко розрахувати та підробити, якби не існував mixhash – змішане (об'єднане) значення усіх операцій в блоці, який також визначає унікальність значення ідентифікатора.

parentHash

256-бітне значення, що відповідає значенню mixhash попереднього блоку. В випадку першого блоку в блокчейні – присутнє лиш ради того, щоб його структура була ідентична структурі усіх подальших блоків мережі.

gasLimit

Максимальна кількість розрахунків, що підтримується одним блоком.

coinbase

160-бітне число (адреса) яка вказує на те, хто буде отримувати винагороди в процесі знаходження хеш-функцій, що відповідають критеріям створення нового блоку в ланцюжку.

timestamp

Змінна часу UTC, за яким мають бути синхронізовані усі ноди в мережі. Також, це значення додається до кожного блоку, що перевіряється.

Тож, саме цей файл буде керувати нашим блокчейном та за допомогою нього ми зможемо ініціалізувати нашу локальну мережу (даний приклад та інформація про його складові була отримана з відкритого інтернет ресурсу Stack Exchange: ethereum.stackexchange.com).

Після проведення усіх підготовчих операцій вказуватимемо ПЗ ноди де саме зберігатимемо дані та правила для нашого блокчейну та формування першого блоку (рис. 3.6), і запустимо сам клієнт – створимо з'єднання з нашою базою даних (БД) на локальній машині або, фактично, запустимо ноду.

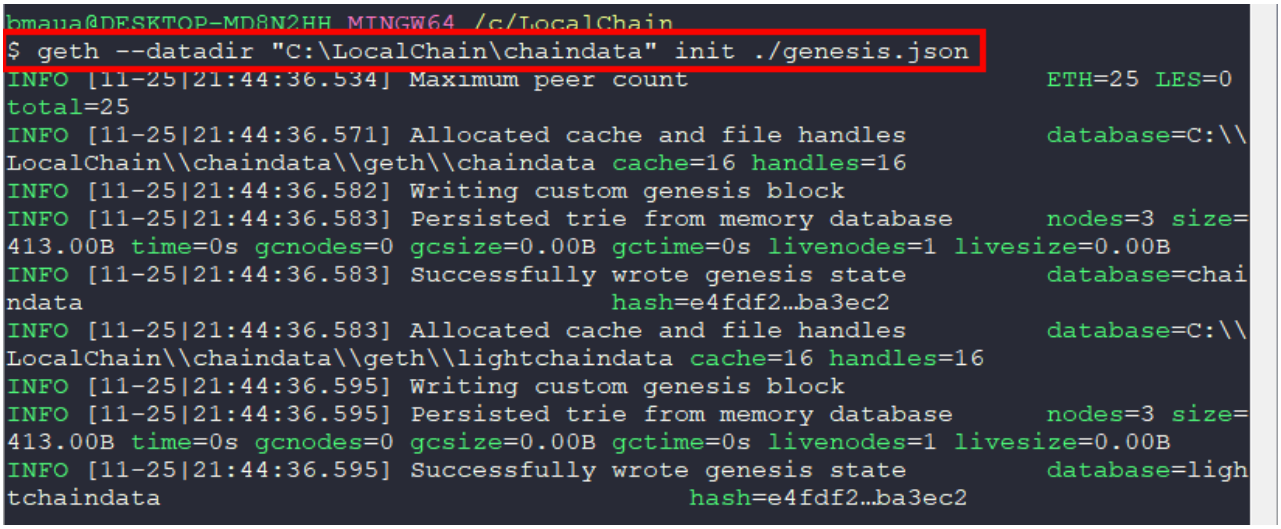


Рисунок 3.6. Ініціалізація geth з вказанням на теку та правила для локальної мережі

Вказуємо ID мережі (може використовуватися для з'єднання нових клієнтів), робочу директорію, TCP порт знаходження і перенаправляємо лог виводу консолі у файл myEth.log для зручності (рис. 3.7 та рис. 3.8):

```
bmaua@DESKTOP-MD8N2HH MINGW64 /c/LocalChain
$ geth --networkid 1996 --port 30303 --datadir "C:\LocalChain\chaindata" console
 2>>myEth.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.18-stable-58632d44/windows-amd64/go1.11.2
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

>
```

Рисунок 3.7. Запуск локальної ноди

```
bmaua@DESKTOP-MD8N2HH MINGW64 /c/LocalChain
$ tail -f myEth.log
INFO [11-25|21:49:06.007] Loaded most recent local header          number=0 hash
=e4fdf2…ba3ec2 td=262144 age=49y7mo1w
INFO [11-25|21:49:06.007] Loaded most recent local full block      number=0 hash
=e4fdf2…ba3ec2 td=262144 age=49y7mo1w
INFO [11-25|21:49:06.007] Loaded most recent local fast block      number=0 hash
=e4fdf2…ba3ec2 td=262144 age=49y7mo1w
INFO [11-25|21:49:06.009] Regenerated local transaction journal    transactions=
0 accounts=0
INFO [11-25|21:49:06.031] New local node record                    seq=1 id=91d9
e2755157aa2d ip=127.0.0.1 udp=30303 tcp=30303
INFO [11-25|21:49:06.031] Started P2P networking                   self=enode://
9895c150e21bafb2957465a3bc8f3158264fe550fc02f7bfe2004bfd8316ea7996e51ffc72b0c686
777ebde5ba57b7c370fff058350bada38249a1303d0b41a1@127.0.0.1:30303
```

Рисунок 3.8. Виведення даних в консоль через лог-файл myEth.log

На цьому налаштування локальної ноди можна вважати завершеним. В результаті попередньої операції ми отримаємо структуру зображену на рис.3.9.

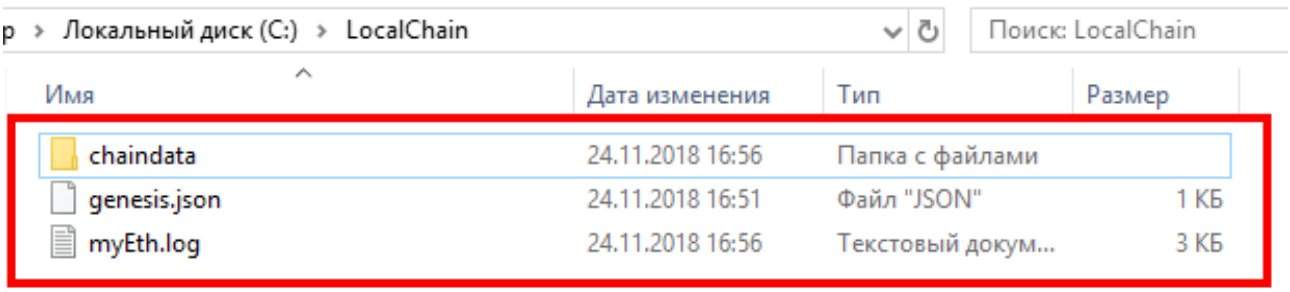


Рисунок 3.9. Структура даних локального блокчейну

Відкрита консоль Geth JavaScript буде слугувати нам інтерфейсом для роботи з створеною структурою шляхом виконання команд та отримання відповідей у відкритий інтерфейс CLI. ПЗ Geth дозволило нам створити локальну мережу (127.0.0.1) в складі якої знаходиться одна нода (TCP порт 30303) з можливістю підключення декількох клієнтів до основної БД, розташованої на локальній машині.

3.3.2 Створення та налаштування аккаунтів користувачів мережі

Тепер, коли ми створили мережу, нам необхідно створити облікові записи для однорангових клієнтів (англ. Peer – "одного рівня"), що будуть працювати в цій мережі. Згодом, використання цих клієнтів дасть нам змогу передавати інформацію з використанням локального блокчейну.

Отже, для створення нового клієнту скористаємося командою `personal.newAccount("PASSPHRASE")`, де PASSPHRASE (укр. Секретне слово) – пароль (підказка до приватного ключа, що буде згенеровано в автоматичному режимі) до нашого аккаунту. Усі приватні ключі клієнтів зберігаються в теці ./chaindata/keystore у вигляді файлу UTC з зашифрованим приватним ключем.

В результаті ми отримаємо адресу новоствореного аккаунту, що будемо використовувати для проведення транзакцій і запуску смарт-контракту в мережі (рис. 3.10). Адреса: `0x4c10bc7b44bb2b7f33fabee845ae86ffac62d9b8 (passphrase: HankaChain).` Умовно, дамо йому назву "перший користувач".

```
> personal.newAccount("HankaChain")

"0x4c10bc7b44bb2b7f33fabee845ae86ffac62d9b8"
> eth.coinbase

"0x4c10bc7b44bb2b7f33fabee845ae86ffac62d9b8"
>
```

Рисунок 3.10. Створення облікового запису першого користувача і генерація адреси в мережі

Також, на рис. 3.10 було запущено команду `eth.coinbase` для автоматичної конфігурації аккаунту за замовчуванням (використовується попередньо отримана нами адреса, єдина на даний момент в мережі). Саме ця адреса буде отримувати всі винагороди в результаті майнінгу на даній ноді.

Звідси, ми можемо зробити перевірку балансу на рахунку, закріпленого за даним адресом: `eth.getBalance(eth.coinbase).` В результаті виконання даної команди ми отримуємо ‘0’ (так як за нашим адресом немає жодної операції по знаходженню хеш-функції за правилами визначеним в genesis.json).

Для того, щоб додати ще одну ноду у мережу будемо керуватись тими самими командами, що були зазначені у розділі 3.3.1: створимо на іншому пристрої теку де будемо зберігати дані мережі і дамо їй назву ./CopyChain (копія даних з початкової БД). Скористаємося тією ж самою структурою файлів як і перед цим і будемо використовувати один і той самий перший блок для ініціалізації клієнта. Вказуємо один і той самий ідентифікатор мережі, але вже з використанням іншого TCP порту (рис. 3.11). Також, створимо додатковий аккаунт (умовного “другого користувача”), з яким в подальшому будемо працювати (рис.3.12). Адреса: `0x2d2aa4148106ac98a9d1275a8d0dc6f6f79a5566 (passphrase: HankaClient2).`

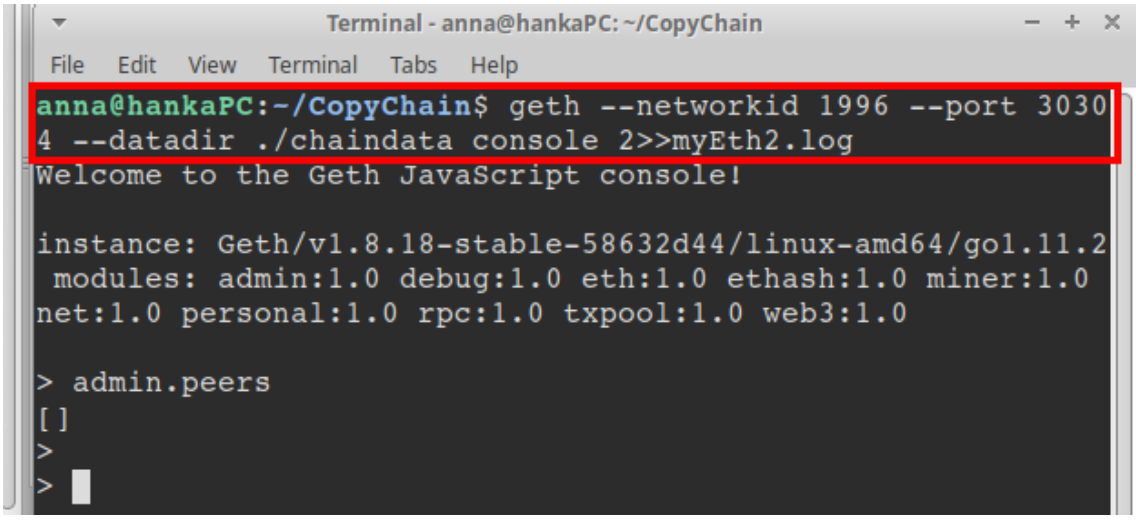


Рисунок 3.11. Ініціалізація другої ноди на 2-ому пристрої

```
> personal.newAccount("HankaClient2")
"0x2d2aa4148106ac98a9d1275a8d0dc6f6f79a5566"
>
```

Рисунок 3.12. Створення облікового запису другого користувача і генерація адреси в мережі

Нода створена, але ми не маємо жодної інформації про перший запущений клієнт. Для цього з CLI для першого користувача визначимо параметри першої створеної ноди (рис. 3.13):

```
> admin.nodeInfo.enode

"enode://d5efdbcb9ee3aeb2fc31fa875687ec144072de2e5fa9d151cbba297ccfb33831693e7c9640043fecddb1
2a6facc7bcc31d3d94ce631dd08770152df127698789@109.251.101.176:30303"
>
```

Рисунок 3.13. Параметри 1ої ноди та мережеві налаштування клієнта

Як бачимо з рисунку 3.13, для нашої ноди зовнішня адреса IP це одна з адрес провайдера, що надає сервіс. Зазвичай такі адреси не є доступними поза мережею провайдера і можуть змінюватися. Та нас цікавить саме локальна адреса ноди отримана від домашнього NAT сервера (фактично, домашнього роутера (шлюзу між приватною та відкритою мережею в нашому прикладі), яку можна дізнатися виконавши команду `ipconfig` (Windows 10) або `ifconfig` (Linux). Результат виконання команди для машини з першою нодою наведено на рис. 3.14. Інформація, що необхідна нам – IPv4 адреса пристрою (192.168.31.142).

```
DNS-суффикс подключения . . . . . :
Локальный IPv6-адрес канала . . . : fe80::6c37:9bc1:dfde:835a%8
IPv4-адрес. . . . . . . . . . . . : 192.168.31.142
Маска подсети . . . . . . . . . . : 255.255.255.0
Основной шлюз. . . . . . . . . : 192.168.31.1
```

Рисунок 3.14. Мережеві параметри доступу для першої ноди

Тому, для підключення до вже запущеної першої ноди з другого пристрою нам необхідно вказати такі її параметри:

`"enode://xxxxxx@192.168.31.142:30303"`

, де xxxxxx – адреса ноди (enode), яку було отримано раніше (рис. 3.13) та 192.168.31.142:30303 – IP адреса та TCP порт клієнта відповідно.

Перезапуск другої ноди на додатковому пристрої буде виглядати так:

```
    geth --networkid 1996 --port 30304 --datadir ./chaindata --
bootnodes
"enode://d5efdbcb9ee3aeb2fc31fa875687ec144072de2e5fa9d151cbba297ccf
b33831693e7c9640043fecddb12a6facc7bcc31d3d94ce631dd08770152df127698
789@192.168.31.142:30303" console 2>>myEth2.log
```

Команда передбачає додаткову вказівку на отриману адресу першої локальної ноди. Логування виводу знов перенаправимо у текстовий файл ./myEth2.log

Залишається перевірити, чи є зв'язок між двома нодами на різних IP адресах, що об'єднано в єдиний локальний блокчейн. Для цього зробимо два запити у консоль першої ноди: `admin.peers` (інформація про усі клієнти в мережі) та `net.peerCount` (кількість нод в мережі). Результат зображено на рис. 3.15.

В списку нод бачимо другого клієнта в мережі, що працює на базі linux з ip адресом 192.168.31.133 (знаходиться в межах нашої локальної мережі) з TCP портом 60208 (слід зазначити, що порт в даному випадку не співпадає з заданим значенням; це відбувається через те, що друга машина була створена віртуально на одному і тому ж пристрої з використанням віртуальної маршрутизації за допомогою мережевого моста; порт змінюватиметься і надалі).

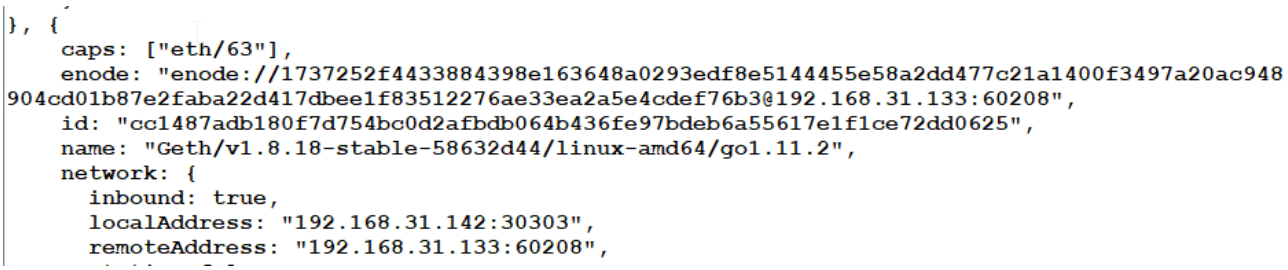


Рисунок 3.15. Виконання перевірки видимості нод та визначення їх кількості

Кількість нод в мережі дорівнює двом. Для проведення експерименту створимо ще одну віртуальну машину та створимо на ній ноду і клієнта за тим самим механізмом, як це було зроблено перед цим. Для кожної ноди має бути створено один аккаунт користувача. Звідси, ми можемо розпочати наступний етап – створення першої транзакції в мережі.

3.4 Транзакції в мережі, запуск смарт-контракту в блокчейні

Після налаштування трьох нод та створення аккаунтів для користувачів нам необхідно перевірити можливість передачі чого-небудь в рамках встановленої мережі – виконати транзакцію з передачі цифрової валюти (передачу токену ETH) з одного облікового запису на інший. Процес виконання та реалізації транзакції зображено на рис. 3.16.

Для виконання будь-якої транзакції з аккаунту перш за все необхідно розблокувати доступ до його приватного ключа. Зробимо це для першого клієнта за допомогою відповідної команди (`personal.unlockAccount (eth.accounts[0])`) та вводу ключа до аккаунта користувача. Система має дати позитивну відповідь на дану операцію. Надалі, виконуємо команду eth.sendTransaction(), де аргументами виступають адреса відправника, отримувач та кількісне значення того, що передається.

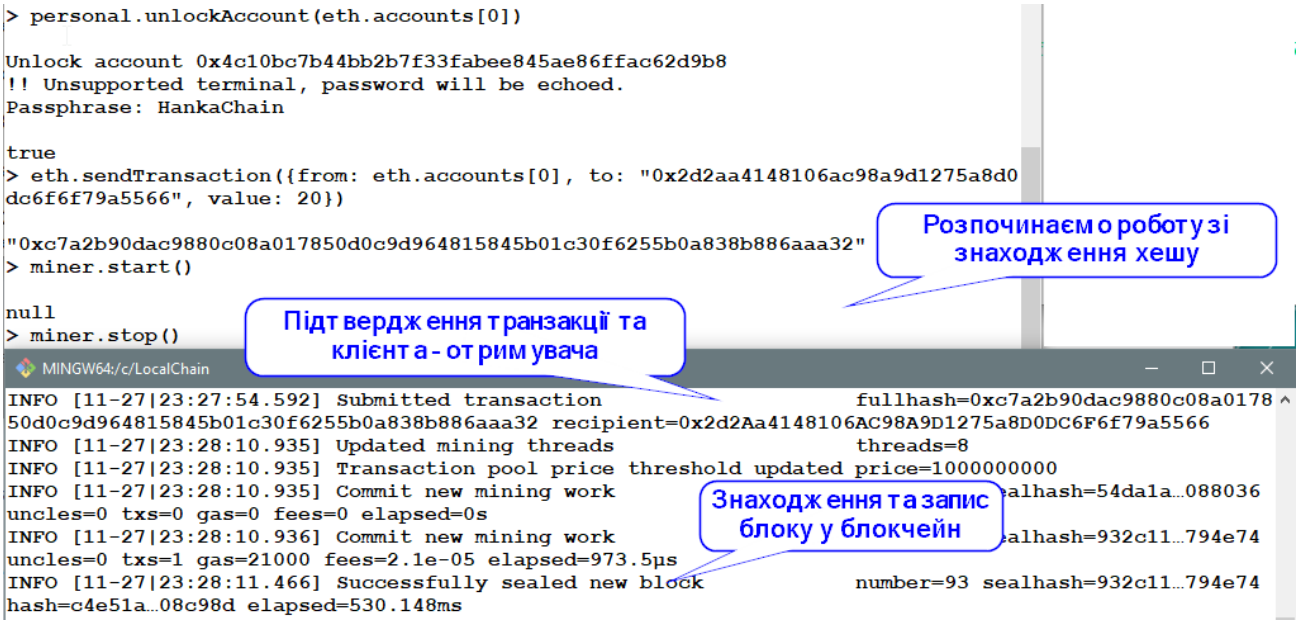


Рисунок 3.16. Затвердження транзакції в мережі та її запис у блок блокчейну

Таким чином, ми можемо виконувати передачу віртуальної валюти з одного аккаунту на інший. Але, з відправкою смарт-контрактів процедура потребує іншої підготовки.

Створення смарт-контракту це створення повноцінного ПЗ на специфічній мові програмування. Solidity поєднує у собі основи декількох мов і спеціально створювався з ціллю задовольняти потреби комунікації та роботи в блокчейні.

Розглянемо приклад реалізації контракту зі збереженням даних і результуючий код:

```
    pragma solidity ^0.5.0;

    contract mortal {
        /* Define variable owner of the type address */
        address owner;

        /* This function is executed at initialization and sets
the owner of the contract */
        constructor() public { owner = msg.sender; }

        /* Function to recover the funds on the contract */
        function kill() private { if (msg.sender == owner)
selfdestruct(msg.sender); }
```

```
}

contract sendURL is mortal {
    /* Define variable URL of the type string */
    string url;

    /* This runs when the contract is executed */
    constructor (string memory _url) public {
        url = _url;
    }

    /* Main function */
    function showURL() view public returns (string memory) {
        return url;
    }
}
```

З даного прикладу виділимо три основні елементи що нас цікавлять: public functions (публічні функції), private functions (приватні функції) та properties (властивості). Як випливає з самих назв, приватні функції мають на меті зберегти інформацію від втручання ззовні, тоді як публічні навпаки, дають доступ до інформації усім охочим. На початку коду завжди має бути вказівник на те, з якою версією компілятора коду сумісні його функції. [12]

Смарт-контракти для передачі в мережі Ethereum мають бути перетворені з серцевого коду у файлову пару .abi та .bin.

.bin – компактна бінарна репрезентація скомпільованого байт коду. Тут відсутні зручні для перегляду операційні коди, усе, що міститься в даному файлі – відтиск повної бінарної операції яка виглядає як набір випадкових символів. Саме цей набір символів буде записано у пам’яті блокчейну і присвоєно власну адресу в ньому. Але, тут постає проблема розшифрування даних. З даних байт коду неможливо визначити наявність функцій в смарт-контракті (таких як `showURL()` та `kill()`). І точно неможливо дізнатися, чи є вони відкритими чи ні. Контракт на даному етапі не має контексту.

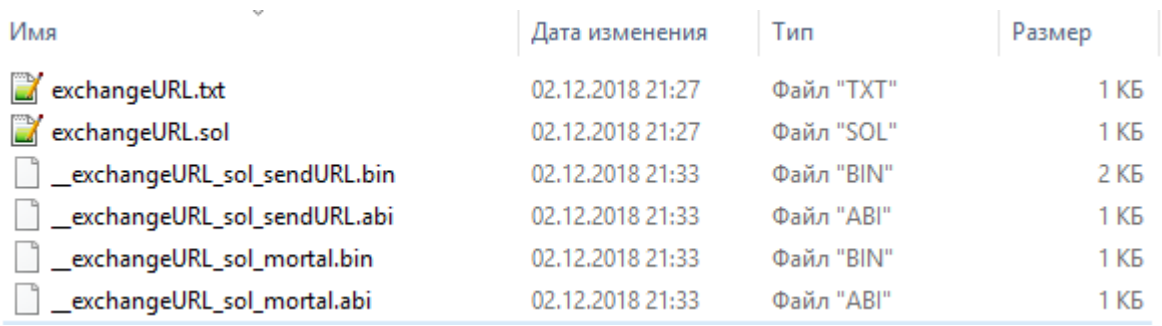
Двійковий інтерфейс програми (англ. ABI – Application Binary Interface) - .json файл який описує розгорнутий (англ. deployed) в блокчейні бінарний код і

його функції. Він дозволяє нам надати контексту (розшифрувати), що саме несе за собою дана послідовність символів та куди саме необхідно звернутися клієнту для отримання даних. Але, ABI не є кодом і не може бути запущений самостійно. Бінарний код – код, що читає та виконує EVM (Ethereum Virtual Machine), не знаючи про контекст.

Створимо нову теку ./LocalChain/inputs на нашій локальній машині та файл exchange.sol який перетворимо за допомогою встановлюваного модуля JavaScript у вихідні файли необхідних форматів:

```
solcjs ./inputs/exchangeURL.sol --bin --abi
```

В результаті, у корені теки, де збережено серцевий код отримаємо 2 документи з необхідними форматами файлів (рис. 3.17):



Имя	Дата изменения	Тип	Размер
exchangeURL.txt	02.12.2018 21:27	Файл "TXT"	1 КБ
exchangeURL.sol	02.12.2018 21:27	Файл "SOL"	1 КБ
_exchangeURL_sol_sendURL.bin	02.12.2018 21:33	Файл "BIN"	2 КБ
_exchangeURL_sol_sendURL.abi	02.12.2018 21:33	Файл "ABI"	1 КБ
_exchangeURL_sol_mortal.bin	02.12.2018 21:33	Файл "BIN"	1 КБ
_exchangeURL_sol_mortal.abi	02.12.2018 21:33	Файл "ABI"	1 КБ

Рисунок 3.17. Результат виконання команди з перетворення сирцевого коду .sol

Переглянемо, що саме записано в _exchangeURL_sol_sendURL.abi файлі:

```
[
   {
      "constant":true,
      "inputs":[

      ],
      "name":"showURL",
      "outputs":[
         {
            "name":"",
            "type":"string"
         }
      ],
```

```
            "payable":false,
            "stateMutability":"view",
            "type":"function"
         },
         {
            "inputs":[
               {
                  "name":"_url",
                  "type":"string"
               }
            ],
            "payable":false,
            "stateMutability":"nonpayable",
            "type":"constructor"
         }
      ]
```

Як бачимо, для кожної функції, визначеної в оді ми в явному вигляді отримуємо інформацію про те, що саме вона очікує на вході та виведе на виході.

В інтерактивній консолі JavaScript для першого клієнта збережемо зміст результуючих .abi та .bin у змінних оточення:

```
exchangeUrlHex = "0x<.bin file contents>"
exchangeUrlAbi = <abi file contents>
```

Для роботи з функціями, що містить контракт, створимо інтерфейс доступу, через який і будемо вести роботу з ним (декларуємо нову консольну функцію): `exchangeUrlInterface = eth.contract(exchangeUrlAbi)`

Тепер ми можемо розмістити бажану операцію в блокчейні (фактично, опублікувати отриманий байт код смарт-контракту) та виконати першу транзакцію з його використанням. Для проведення транзакції в мережі обов’язково має бути клієнт-майнер (запуск шляхом виконання `miner.start()`)!

Для роботи з обліковим записом клієнта необхідно перш за все розблокувати доступ (декриптувати приватний ключ користувача в директорії ./LocalChain/chaindata/keystore):

```
personal.unlockAccount(eth.accounts[0])
> HankaChain
```

, де `eth.accounts[0]` – аккаунт за замовчуванням (єдиний акант на нашій ноді), `HankaChain` – пароль від зашифрованого ключа.

Нарешті, публікуємо наш контракт в мережі:

```
var urlTX = exchangeUrlInterface.new(
     "google.com",
     {
          from: eth.accounts[0],
          data: exchangeUrlHex,
          gas: 4700000
     }
)
```

Після виконання команди в логах ноди отримаємо:

```
INFO [12-02|23:18:52.422] Submitted contract creation
fullhash=0xb7a51cd4af2fb1b88cc34c9884f3d31b2ea7b8e655737c90b76298a6
3b005d7b contract=0xBD904eA20D6aDE4086d01bD367df1d227f09E5C3
```

Вже через декілька секунд дана транзакція повинна бути перевірена майнером в мережі. В реальних мережах дана операція може тривати до 1 хвилини, дивлячись від завантаженості мережі та оціночної вартості транзакції (gasValue).

Перевірити статус транзакції можна за допомогою команди `txpool.status`. Pending значитиме, що транзакція в очікуванні.

Збережемо дані операції в окремі змінні urlTXHash – хеш значення транзакції; publishedUrlAddr – адреса транзакції в блокчейні (однакова для всіх учасників мережі).

```
urlTXHash = urlTX.transactionHash
publishedUrlAddr =
eth.getTransactionReceipt(urlTXHash).contractAddress
```

Тож, тепер наш смарт-контракт має бути доступним за вищезазначеним адресом для будь-якого учасника блокчейну.

На другій ноді виконаємо такі операції:

```
    exchangeUrlAbi = [<abi file contents>]
    exchangeUrlInterface = eth.contract(exchangeUrlAbi)
    publishedUrlAddr = <publishedUrlAddr from the 1st Eth.Node>

    //Creating interface to work with the contract functions
    exchangeUrlFunctions =
exchangeUrlInterface.at(publishedUrlAddr)

    //Calling function showURL from tokenExample exchangeURL
    exchangeUrlFunctions.showURL()
```

В результаті виконаних команд отримаємо бажаний url – тепер він збережений в пам’яті блокчейну (рис. 3.18):

```
> publishedUrlAddr = "0xbd904ea20d6ade4086d01bd367df1d227f09e5c3"
"0xbd904ea20d6ade4086d01bd367df1d227f09e5c3"
> exchangeUrlFunctions = exchangeUrlInterface.at(publishedUrlAddr)
{
  abi: [{
      constant: true,
      inputs: [],
      name: "showURL",
      outputs: [{...}],
      payable: false,
      stateMutability: "view",
      type: "function"
  }, {
      inputs: [{...}],
      payable: false,
      stateMutability: "nonpayable",
      type: "constructor"
  }],
  address: "0xbd904ea20d6ade4086d01bd367df1d227f09e5c3",
  transactionHash: null,
  allEvents: function(),
  showURL: function()
}
> exchangeUrlFunctions.showURL()
"google.com"
>
```

Рисунок 3.18. Вивід консолі на запит функції за визначеною адресою

Для того щоб отримати сторінку за заданим URL скористаємося інструментом wget на стороні-отримувачі повідомлення:

```
wget "https://www.google.com" -O ./goog.txt
```

Результуючий файл стиснемо у gz формат. В результаті отримаємо файл розміром у 7.2 кілобайти. Виконаємо тепер відправку даних у відповідь з другого хоста. Створимо додатковий контракт з ціллю зберігати URL та дані за посиланням. Сирцевий код контракту:

```
    pragma solidity ^0.5.0;

    contract mortal {
        /* Define variable owner of the type address */
        address owner;

        constructor() public { owner = msg.sender; }

        /* Function to recover the funds on the contract */
        function kill() private { if (msg.sender == owner)
selfdestruct(msg.sender); }
    }

    contract sendURL is mortal {
        /* Define variable URL of the type string */
        string publishedUrlAddr;
    string htmlCompressed

        /* This runs when the contract is executed */
        constructor (string memory _url, string memory _data)
public {
            url = _url;
         htmlCompressed = _data
        }

        /* Main functions */
        function showURL() view public returns (string memory) {
            return url;
        }

    function showData() view public returns (string memory) {
            return htmlCompressed;
        }
    }
```

За схожим механізмом визначимо усі змінні простору та передаэмо в складі htmlCompressed ряд символів що відповідає .gz формату файла. Звичайно, через те, що такий файл містить близько 20 тисяч символів на проведення даної операції варто закласти неабияку оцінку операції для перевірки, та, в рамках нашої мережі, дана операція не несе за собою надвеликих витрат. Для реальної мережі Ethereum, де приблизна вартість передачі одного кілобайту даних коштує приблизно 7 центів в USD, а передача нашої сторінки коштувала би близько 60 центів, що вже досить значна вартість сама по собі. Про дану особливість поговоримо трохи пізніше, оцінюючи переваги та недоліки пропонованого способу. [13]

Отже, після публікації контракту в мережі та його перевірки, після звернення до контракту через функцію `showData()` отримаємо стиснуті дані, що вимагають декриптування на локальній машині. Перетворивши запит отримаємо сирцевий код .html сторінки, відкривши у браузері отримаємо (рис. 3.19):

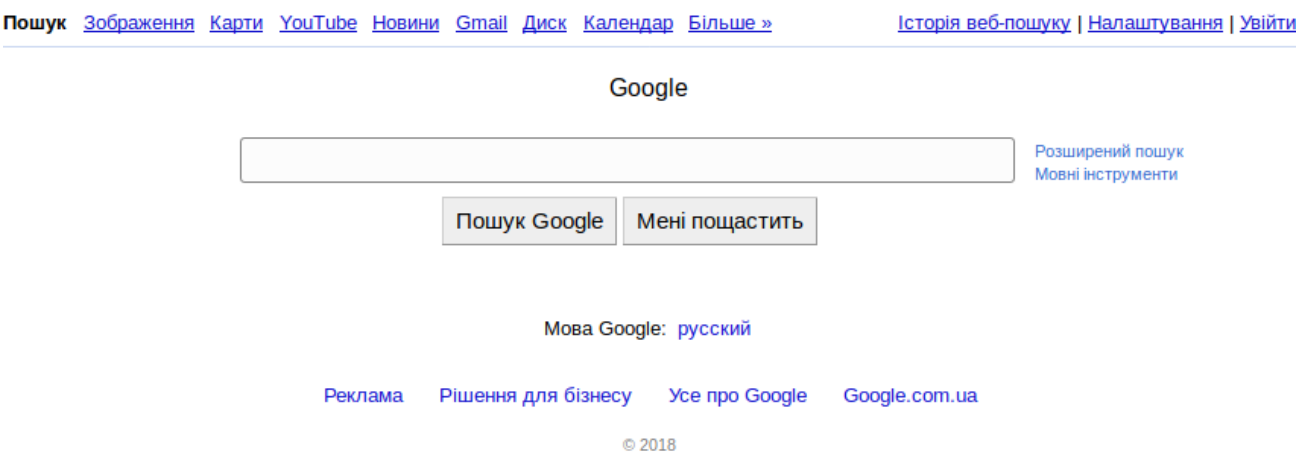


Рисунок 3.19. Відкрита .html сторінка

Фактично, прямого звернення до доменного імені не відбулося і дані адреси було ретрансльовано в стиснутому вигляді через мережу блокчейн. Для

хосту увесь процес виглядає як звернення з IP, що не має зв’язку з цільовим клієнтом.

Таким чином, ми змогли транслювати дані через створену локальну мережу та створити тунелювання з виключенням необхідності витрачати значні обсяги трафіку на завантаження сирцевої URL сторінки та маскуючи свою присутність для віддаленого цільового сервера, який отримав запит з однієї з нод в складі блокчейн структури.

3.5 Моделювання локального клієнта блокчейн VPN

Операції, що були проведені нами вище, можливо виконувати в автоматичному режимі за допомогою використання вже існуючого ПЗ (Ethereum Wallet, Mist, інше), але вони не розкривають того, з чим саме доводиться працювати, для створення з’єднання в блокчейні.

Використання такого механізму для нормального використовування середньостатистичним користувачем має бути максимально простим та автоматизованим. Планом нашої роботи не є створення повноцінного ПЗ для роботи в мережах даного типу, але ми можемо створити найпростіший клієнт (скрипт), що буде обробляти та надсилати запити в javascript консоль створеної нами ноди.

Для створення ‘ідентифікації’ використовуватиметься структура “одна нода – один клієнт”. Це обмежує функціональність усієї структури, але дозволяє реалізувати концепцію розподіленої мережі на практиці, коли кожен учасник одночасно і клієнт і джерело. Дана структура містить мультихеш ноди (enode) та пару Private & PeerAccount, для підписання транзакцій та декодування отриманого хешу відповідно:

Наступним кроком – нам необхідно вміти працювати з байт-кодом та мати змогу виконувати автоматичне криптування / декриптування даних з якими працюватиме клієнт. Для цього можливо використовувати інструмент solc js і

створювати .abi й .bin файли в автоматичному режимі шляхом звернення до встановленого модуля й обробкою вихідних файлів працюючим локальним клієнтом.

Для компіляції та роботою з сирцевим кодом використовуватимемо компілятор solc. Для встановлення на Debian-based Linux ОС:

```
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install solc
```

Створимо скрипт, який працюватиме з відправкою даних на відповідь до отриманого запиту URL. Перш за все скомпілюємо наш код таким чином:

```
solc exchangeHTML.sol --combined-json abi,asm,ast,bin,bin-runtime,clone-bin,devdoc,interface,opcodes,srcmap,srcmap-runtime,userdoc > exchangeHTML.json
```

Далі, для роботи з вже запущеною нодою будемо використовувати протокол RPC (Remote procedure call; укр. Виклик віддалених процедур) який за замовчуванням активовано для ноди на порту 8545. Тобто, транслювати команди будемо за адресою http://localhost:8545.

Також, обов’язковою умовою для виконання скрипту є встановлення менеджера пакетів npm, що дозволяє працювати з модулями node.js (JavaScript додатками). Нода, на котрій запускається клієнт має отримати дані про url адресу через описану вище процедуру (також можливо автоматизувати): `exchangeUrlFunctions.showURL()`.

Лістинг отриманого скрипту наведено нижче:

```
//Copyright: Anna Kuklina
//Contact: kuklinaas30@gmail.com

let fs = require("fs");
let Web3 = require('web3');

let web3 = new Web3();
```

```
    web3.setProvider(new
web3.providers.HttpProvider('http://localhost:8545'));

    // Зчитування URL адреси та даних стиснутої html сторінки
    let url = exchangeUrlFunctions.showURL();
    let htmlCompressed = fs.readFileSync("html.gz");

    //solc exchangeHTML.sol --combined-json abi,asm,ast,bin,bin-
runtime,clone-bin,devdoc,interface,opcodes,srcmap,srcmap-
runtime,userdoc > exchangeHTML.json

    let src = fs.readFileSync("exchangeHTML.json");
    let contracts = JSON.parse(src)["contracts"];

    // Створення змінних для роботи з .abi та .bin
    let abi = JSON.parse(contracts.exchangeHTML.abi);
    let code = contracts.exchangeHTML.bin;

    // Створення proxy класу для нашого смарт-контракту
    let exchangeHTML = web3.eth.contract(abi);

    // Розблокування акаунту на ноді (coinbase)
    console.log("Спроба розблокувати обліковий запис
користувача:");
    var password = "";
    try {
      web3.personal.unlockAccount(web3.eth.coinbase, password);
    } catch(e) {
      console.log(e);
      return;
    }

    console.log("Надсилаю дані у блокчейн мережу...");
    let rawdataTXcontract = exchangeHTML.new(url,
htmlCompressed,{from: web3.eth.coinbase, gas: 1000000, data:
code});

    // Потрапляння транзакції у чергу транзакцій
    console.log("Операцію було провелено в рамках транзакції " +
rawdataTXcontract.transactionHash);

    function sleep(ms) {
      return new Promise(resolve => setTimeout(resolve, ms));
    }

    // Очікуємо до момента поки наша транзакція не буде перевірена
    async function waitBlock() {
      while (true) {
```

```
        let receipt =
web3.eth.getTransactionReceipt(rawdataTXcontract.transactionHash);
        if (receipt && receipt.contractAddress) {
          console.log("Успішне завершення транзакції! Адреса: " +
receipt.contractAddress);
          break;
        }
        console.log("Очікування на запис транзакції в блок...
Зараз у: " + web3.eth.blockNumber);
        await sleep(4000);
      }
    }

    waitBlock();
```

В результаті запуску даного скрипту на локальній машині, що вже знає URL за вказаним механізмом та має стиснену html сторінку, буде створено новий смарт-контракт в мережі, а хеш та адреса транзакції буде виведено у консоль.

3.6 Визначення перспектив розвитку та аналіз існуючих аналогів

Кожен механізм або нова концепція намагається вирішити вже існуючу проблему своїм шляхом. При цьому, кожен з варіантів реалізації, що можливі для існування паралельно, не позбавлені як своїх переваг, так і своїх недоліків.

На сьогоднішній день, кількість інвестицій на дослідження та впровадження інструментів приватності, що здатні встановити конфіденційність користувачів в інтернеті, все ще мала. Ці обставини надають сенсу для створення додаткових захисних заходів, спрямованих на збереження відкритого та безперешкодного доступу до ресурсів Інтернет.

З винаходом та відносною зрілістю потужних однорангових комп’ютерних технологій блокчейну, можливість використовувати технологічні здобутки цих мереж у розробці механізмів ухилення від недовіри та цензури стає все більш реальним. Блокчейн, як відносно нові технологія, все ще перебуває на етапі формування. Компанії, що використовують її, намагаються

створити власне бачення та реалізацію механізмів, а не стандартизувати інструменти для роботи з вже існуючими. І це цілком нормальний процес. [14]

Та, в будь-якому випадку, нові рішення в сфері комунікацій – абсолютно необхідні речі. Методу IP адресації людство зобов'язано своїм нинішнім станом речей – доступному доступу до інформації. Але, даному протоколу скоро виповниться сорок років, що в рамках технологічного розвитку аж занадто велика цифра, його застосування сьогодні – звичка та встановлені за десятиріччя правила, які не обов'язково мають залишатися такими назавжди. У випадку з Ethereum слід зазначити, що даний механізм сам по-собі декларує новий спосіб роботи з даними і мережевими додатками.

Переваги використання технології блокчейну (Ethereum):

1. Відкритість платформи та ідентичність даних. Blockchain працює як спільна система записів і точно визначає ідентичність даних завдяки використанню хешування, усуваючи необхідність узгодження різноманітних джерел. Будь хто може стати учасником проекту та внести свій вклад у розвиток.

2. Підвищення продуктивності й уникнення маркетингового втручання. Кожен учасник мережі має власні права доступу та може керувати інформацією, яка буде розподілена у мережі. Усі додатки на базі блокчейну мають чітко визначені рамки дозволеної до обробки інформації та межу відповідальності за їх збереження. Такі дані не можуть бути передані третім особам, які могли би скористатися даними у власних цілях. Така б операція у той же час стала би загальновідомою. Такий принцип роботи мережі одночасно дозволяє застосовувати механізми доступу для розподіленої роботи групи або організації над єдиним інформаційним простором, створювати та ділитися здобутками з усіма учасниками водночас, без ризику втручання небажаних гостей у цей процес.

3. Розподілені зв’язки в системі з кращим доступом до даних. Використання блокчейну веде до уникнення точного зв’язку між особою та клієнтом. Адреса в блокчейні – лише набір даних, які не мають звязку до мережевої адреси або ідентичності. Будь-які дані стають загальнодоступними та існуючими одночасно для всіх. В рамках VPN сервісів така особливість дозволяє уникнути необхідності звертатися до самого ресурсу, достатньо запитати в мережі, чи міг би хтось надати свій вільний інформаційний канал для доступу до віддаленого ресурсу та поділитися ним в межах платформи.

Проблеми реалізації передачі даних в Ethereum:

1. Масштабованість системи. З виникненням блокчейну світ також познайомився і з криптовалютною складовою таких мереж. Для багатьох людей, що чули про блокчейн цей термін означає лише єдине поняття – за його допомогою можливо отримати гроші. Блокчейн, як структура сама по собі, не зміг би існувати в нинішній реалізації без існування майнерів в системі. Саме цими майнерами і стали перші користувачі такої структури. За бажання створити будь-що в мережі, розробник обов’язково стане перед проблемою цінових спекуляцій та проблемою збільшення пустих адрес в мережі, які несуть за собою нецільові потоки даних у самому блокчейні. Ця ситуація призводить до неможливості створити дійсно цінні проекти і не потрапити у ситуацію коли його фінансування та цілі мають різні цінові межі і можуть абсолютно відрізнятися.

2. Коливання продуктивності та стабільності мережі. Блокчейн, на думку багатьох дослідників, все ще не розкрив свого потенціалу і перебуває на етапі аналогічному до винаходу першої електронної машини – має таку ж громіздкість та нестабільність роботи. Коли інтерес до неї то зростає, то зменшується, мережа перебуває під значними навантаженнями і черга транзакцій, що вимагають перевірки, може зростати швидше, ніж зменшуватися

у розмірах. Саме тому, для повноцінної реалізації свого задуму, розробники мусять використовувати механізми, які б дозволили бути незалежними від основної мережі і функціонувати за межами інтересів публічності.

3. Складність імплементації. Дана проблема супроводжує усі проекти, що мають на меті використання нових технологій. Доти доки немає єдиного та визначеного способу роботи з мережами блокчейн і усі інструменти для них перебувають на етапі створення та дослідження – неможливо гарантувати однозначність виконання, роботи системи або програмного коду. Наприклад, на момент написання даної роботи актуальна версія мови Solidity змінила порядковий номер версії з 0.4.24 до 0.5.0 з доданням нових функцій, а geth консоль отримала чергове оновлення, що мало на меті виключити вразливості попередньої версії.

В цілому, застосування смарт-контрактів з ціллю передачі трафіку – ідея не нова, але реалізацію даного механізму в завершеному стані досі не представлено в повному обсязі, їх розробка все ще в процесі (такими проектами можна вважати мережу Privatix та Mysterium).

Застосування смарт-контрактів не єдиний варіант реалізації обміну даними в блокчейні. На даний момент вже існує декілька проектів, що не беруть за основу використання смарт-контрактів, але маркують самі дані адресами в розподіленій мережі. Кожен, хто бажає отримати копію даних повинен лише звернутися до мережі з запитом. В такому разі, у відповідь, буде надано посилання у вигляді хеш-функції на сам об’єкт, а доступ до нього буде надаватися одним з власників даного блоку інформації (механізм схожий до роботи системи Bittorrent). Таку або подібну реалізацію створено проектами IPFS, Swarm та Sia.

Використання подібного механізму може знищити потребу в публікації ресурсу на якомусь конкретному сервері. Він буде збережений одночасно на декількох машинах в мережі і доступним лише за своїм відтиском (хеш-

значенням, або індексом в мережі), але не за конкретною мережевою адресою. Даний механізм також усуває необхідність грошових витрат для утримання інформації на власному фізичному носії, дає ієрархічне представлення усіх копій файлу, в яких хоча б хто не будь зацікавлений і можливість працювати з мережевими ресурсами навіть без прямого з'єднання (ресурс – саме існування підключення до мережі).

Фактично, таке представлення мережевих принципів комунікації створює нове покоління інтернету, так званий web3.0 де всі можуть стати повноцінними членами єдиної інформаційної структури.

Висновки до розділу

1. Було сформовано основні принципи та переваги використання технології блокчейн для створення нового кластеру VPN мереж. За результатами дослідження можливо зробити висновок, що існуючі технології забезпечення приватності в мережі поступаються запропонованому методу і в випадку успішного моделювання останнього будуть неконкурентоспроможними в нинішніх реалізаціях.

2. Створення поетапного плану з виконання тестової передачі даних в мережі Ethereum доводить можливість створення описаного механізму комунікації та потребує лише впровадження додаткових механізмів ізоляції даних на всіх етапах з застосуванням можливостей мережі. Написання повноцінного (графічного) клієнту в рамках даної дисертації не передбачалось і потребує спеціальних навичок програмування та створення інтерфейсів користувача. Але, за умов вірного налаштування робочого простору, скрипт, що наведено в розділі моделювання клієнту, може виконувати задачу автоматизації команд та відправки даних у мережу.

3. Аналіз перспектив розвитку технології блокчейн та особливостей її розвитку вказує на такі основні її переваги: доступність та ідентичність даних у

мережі, уникнення небажаного контролю за даними третьою стороною, безпека та підвищена продуктивність груп користувачів, використання сучасних методів збереження конфіденційності. Недоліками подібного рішення можуть стати складність реалізації стабільної роботи на етапах розвитку засобу, відносна проблема з масштабованістю мережі та прийняття нового принципу комунікації усіма структурами й учасниками світової спільноти глобальної мережі Інтернет.

ВИСНОВКИ

1. Було наведено класифікацію VPN мереж та проведено аналіз існуючих рішень у сфері захисту інформації, їх переваги та недоліки. Надано характеристику топологіям побудови VPN мереж в результаті чого визначено такі основні їх проблеми: закритість платформи і, як результат, її вразливість до внутрішнього втручання; вразливість до атак перенавантаження мережі; відсутність розподіленої структури клієнтів та обмежена кількість резервних каналів передачі даних; відсутність наскрізного шифрування даних на всіх етапах їх передачі через мережеві канали. В якості обов'язкових компонентів, що притаманні усім мережам VPN та є невід'ємною частиною цих мереж, можна виділити: сервіс-орієнтовна структура, приватність інформації з контролем за її розповсюдженням, швидкість та вартість розгортки рішення, швидкість роботи мережі.

2. Визначено принципи роботи мережі Blockchain, розглянуто компоненти в її складі: хеш функція, хеш таблиця, асиметричні алгоритми шифрування, смарт-контракт. Основними перевагами використання цих механізмів визначено: неможливість підробки даних, прозорий механізм передачі даних, можливість перевірки інформації, що передається, надійність роботи мережі та розподілені зв'язки в системі. Ці особливості як найкраще можуть слугувати для функціонування нового покоління VPN та передачі трафіку в цілому.

3. Було створено структурний опис та схему роботи і комунікації клієнтів в мережах блокчейн, виділено основні етапи обміну даними, встановлення діалогів та сервіс-сесій. Даний матеріал повинен допомогти в розгортанні механізмів комунікації та роботи усіх клієнтів в рамках зазначеної структури.

4. Виконано пошук та підбір програмного забезпечення для розгортання тестового оточення мережі Blockchain та передачі даних по захищеному каналу. Розроблено методичні матеріали по розгортанню мережі Blockchain. За її результатами було виконано тестовий обмін інформацією в межах блокчейн

структури. Дана інформація може стати у нагоді за мети розробки власних рішень у даній сфері, створити загальне уявлення про структуру Ethereum блокчейну та подібних.

5. Проведено аналіз умов розвитку мереж VPN на базі технології Blockchain, її основних перспективних напрямків використання. За результатами даного аналізу можемо зазначити, що використання Blockchain для вирішення сучасних проблем комунікації та доступу до даних може вирішити усі насущні проблеми Інтернет зв’язку і створити паралельну та альтернативну гілку розвитку з власними принципами роботи мережі. Основною перешкодою на шляху до цього стає відносне неповноліття даної технології та відсутність чітких стандартів роботи подібних мереж. Це завдання є основним на шляху до винайдення нових застосувань даного рішення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Насколько важна защита информации в интернете для компаний // Справочные данные - микросхемы и электронные компоненты. URL: http://www.gaw.ru/html.cgi/txt/gl/internet5/naskolko-vazhna-zashhita-informacii-v-intjernjetje.htm (дата звернення: 05.10.2018).

2. Нестеренко М.М., Саєнко Б.В.,Кукліна А.С. / Аналіз побудови корпоративних мереж на основі VPN-технологій // Проблеми телекомунікацій: XI міжнар.наук. –тех. конф., квіт.2017

3. Транспортный уровень // Stud files. URL: https://studfiles.net/preview/5337444/page:6/ (дата звернення: 17.10.2018).

4. Typical VPN Topologies // Etutorials. URL: http://etutorials.org/Networking/MPLS+VPN+Architectures/Part+2+MPLS-based+Virtual+Private+Networks/Chapter+7.+Virtual+Private+Network+VPN+Implementation+Options/Typical+VPN+Network+Topologies (дата звернення: 1.11.2018).

5. Roger Dingledine, Nick Mathewson and Paul Syverson. Tor: The Second - Generation Onion Router // Onion router. URL: http://www.onionrouter.net/Publications/tor-design.pdf (дата звернення: 01.11.2018).

6. Блокчейн // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Блокчейн (дата звернення: 03.11.2018).

7. Асиметричні алгоритми шифрування // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Асиметричні_алгоритми_шифрування (дата звернення: 17.11.2018).

8. Tree Hash EXchange format (THEX) // Wayback Machine. URL: https://web.archive.org/web/20080316033726/http://www.open-content.net/specs/draft-jchapweske-thex-02.html (дата звернення: 19.11.2018).

9. How Do Ethereum Smart Contracts Work? // CoinDesk, Inc. URL: https://www.coindesk.com/information/ethereum-smart-contracts-work (дата звернення: 07.11.2018)

10. What is Gas? // MyEtherWallet – open-source interface for generating Ethereum wallets & more. URL: https://kb.myetherwallet.com/gas/what-is-gas-ethereum.html (дата звернення: 07.11.2018)

11. Стандарты токенов. Какие бывают и зачем нужны? // Medium Corporation. URL: https://medium.com/@glavcrypt/tokens-standarts-bd150592c9ae (дата звернення: 11.11.2018)

12. Solidity 0.5.1 documentation // ReadTheDocs – Solidity. URL: https://solidity.readthedocs.io (дата звернення: 12.11.2018)

13. All Questions on Ethereum // StackExchange - Ethereum. URL: https://ethereum.stackexchange.com/questions/ (дата звернення: 19.11.2018)

14. Blockchain Unleashed // IBM Blockchain Blog. URL: https://www.ibm.com/blogs/blockchain/ (дата звернення: 25.11.2018)