**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ**

**ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# Кваліфікаційна магістерська робота

на тему «***Алгорітичне та програмне забезпечення для аналізу продуктивності блокчейн мережі***»

Виконав: студент групи МгІТ-22
спеціальності
122 Комп'ютерні науки

Максим НІКІТЮК
Керівник доц. Оксана КОЛИСКО

Рецензент

Київ 2023

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ**

**ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

Спеціальність **122 Комп'ютерні науки**
Освітня програма **Комп'ютерні науки**

**ЗАТВЕРДЖУЮ**
Завідувач кафедри КН
проф.Щербань В.Ю.
"____" _____2023 року

# З А В Д А Н Н Я

**НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ**

*Нікітюку Максиму Вікторовичу*

**1. Тема роботи** Алгорітичне та програмне забезпечення для аналізу продуктивності блокчейн мережі

науковий керівник роботи Колиско Оксана Зенонівна, затверджені наказом вищого навчального закладу від "_12_"вересня_2023 року №_210-уч_

**2. Строк подання студентом роботи** _1 листопада 2023р__.

**3. Вихідні дані до роботи**: *Розробки кафедри комп'ютерних наук; рекомендована література, додатки*.

**4. Зміст дипломної роботи** : *ВСТУП; РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ; РОЗДІЛ 2 ПРОЕКТУВАННЯ; РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ; ВИСНОВКИ; СПИСОК ЛІТЕРАТУРИ; ДОДАТОК А ОКРЕМІ ФРАГМЕНТИ ПРОГРАМНОГО КОДУ; ДОДАТОК Б ПРЕЗЕНТАЦІЯ.*

**5. Перелік графічного матеріалу***: презентація на _____ слайдах*

**6. Консультанти розділів дипломної магістерської роботи**

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Розділ1 | Колиско О.З., доцент, к.т.н. | | |
| Розділ 2 | Колиско О.З., доцент, к.т.н. | | |
| Розділ 3 | Колиско О.З., доцент, к.т.н. | | |

**7. Дата видачі завдання** серпень 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної магістерської роботи | Терміни виконання етапів | Примітка про виконання | |
|---|---|---|---|---|
| | | | *Студ.* | *Керівн.* |
| 1 | Вступ | *15.09.2023* | | |
| 2 | Розділ 1. Постановка задачі | *20.09.2023* | | |
| 3 | Розділ 2 Проектування | *30.09.2023* | | |
| 4 | Розділ 3. Програмна реалізація | *10.10.2023* | | |
| 5 | Висновки | *25.10.2023* | | |
| 6 | Оформлення дипломної магістерської роботи (чистовий варіант) | *1.11.2023* | | |
| 7 | Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту) | *4.11.2023* | | |
| 8 | Перевірка дипломної магістерської роботи на наявність текстових співпадінь та помилок (за 10 днів до захисту) | *6.11.2023* | | |
| 9 | Подання дипломної магістерської роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану (за 10 днів до захисту) | *8.11.2023* | | |
| 10 | Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту) | *10.11.2023* | | |

**Студент** _____ Максим НІКІТЮК

**Науковий керівник роботи** _____ Оксана КОЛИСКО

**Директор НМЦУПФ** _____ Олена ГРИГОРЕВСЬКА

# АНОТАЦІЯ

Ніктюк М.В. Алгоритмічне та програмне забезпечення для аналізу продуктивності блокчейн мережі. – Рукопис.

Магістерська робота за спеціальністю 122 Комп'ютерні науки. – Київський національний університет технологій та дизайну, Київ, 2023.

У роботі досліджено продуктивність існуючих блокчейн-рішень та запропоновано альтернативний протокол ETAMP для підвищення швидкості та масштабованості.

Проведено аналіз архітектури, протоколів та алгоритмів ETAMP. Розглянуто криптографічні механізми, структури даних, мережеву топологію та можливості шардингу.

Здійснено тестування і порівняльний аналіз продуктивності ETAMP та традиційних блокчейнів. Підтверджено переваги ETAMP у швидкості транзакцій, масштабованості, безпеці та енергоефективності.

Визначено напрями оптимізації ETAMP. Запропоновано рекомендації щодо підвищення ефективності та інтеграції протоколу в різні системи.

*Ключові слова: блокчейн, протокол, криптографія, продуктивність, ETAMP.*

## ABSTRACT

**Nikitiuk M.V. Algorithmic and software for analyzing the performance of a blockchain network.**

Master's degree work on specialty 122 Computer Science - Kyiv National University of Technology and Design, Kyiv, 2023.

The paper investigates the performance of existing blockchain solutions and proposes an alternative ETAMP protocol to increase speed and scalability.

The architecture, protocols, and algorithms of ETAMP are analyzed. Cryptographic mechanisms, data structures, network topology and sharding capabilities are considered.

The article tests and compares the performance of ETAMP and traditional blockchains. The advantages of ETAMP in terms of transaction speed, scalability, security, and energy efficiency are confirmed.

The directions of ETAMP optimization are identified. Recommendations for improving the efficiency and integration of the protocol into various systems are proposed.



*Keywords: blockchain, protocol, cryptography, performance*

ЗМІСТ

ДОДАТОК Б

ДОДАТОК В

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ETAMP (Encrypted Token and Message Protocol) – альтернативний високопродуктивний протокол передачі даних

PEM ( Privacy-Enhanced Mail) - широко використовуваний формат для зберігання і передачі криптографічних ключів і сертифікатів

JWT (JSON Web Token) - компактний, самодостатній способ передачі інформації між сторонами у форматі JSON

ECDH (Elliptic-curve Diffie–Hellman) -криптографічний метод, що дає змогу двом сторонам, кожна з яких має пару відкритих і закритих ключів, створити спільний секретний ключ

ECDSA (Elliptic Digital Signature Algorithm) - алгоритм відіграє центральну роль у генерації та перевірці цифрових підписів, один з ключових компонентів сучасної криптографії на еліптичних кривих.

# ВСТУП

Blockchain – це розподілена база даних, в простих термінах це мережа з кількома вузлами. Такі вузли зберігають мережеві записи: коли в мережу надходить нова інформація, вона додається до всіх вузлів. Особливість мережі полягає в тому, що вона отримує лише достовірну інформацію.

Одним із топ напрямків використання блокчейн є забезпечення надійного обміну даними. Доступ до даних та обмін ними між сторонами зазвичай обмежений через розрізнення технологій та міркувань конфіденційності. Приватні та довірені DLT-платформи дозволяють організаціям безпечно взаємодіяти з даними та обмінюватись ними, гарантуючи, що перевірені, довірені треті сторони мають лише необхідні рівні доступу до даних. Не жертвуючи цілісністю даних чи конфіденційністю, організації можуть обмінюватися даними за межами компанії, а також зміцнювати співпрацю та довіру між партнерами. Наприклад, безпечний обмін даними між постачальниками медичних послуг може покращити обмін медичною інформацією про пацієнтів; у розвідувальному співтоваристві це могло б полегшити обмін розвідувальними даними про загрози та іншу важливу інформацію між відомствами та міжнародними кордонами.

Метою роботи є розробка програмного забезпечення для оцінки продуктивності блокчейнмережі при використанні різних алгоритмів хешування. Об’єктом дослідження є методи та способи забезпечення інформаційної безпеки на основі технології блокчейн. Предметом дослідження є способи захисту інформації від несанкціонованого доступу, а також уже реалізовані системи забезпечення інформаційної безпеки на основі технології блокчейн. Існує багато метрик, що стосуються логіки та якості роботи блокчейну. Вони допомагають визначити вузькі місця в коді та знайти логічні та оптимізаційні проблеми в алгоритмах консенсусу та фінальності у блокчейнах. Будь-яка розробка розподілених систем, у тому числі блокчейнів, вимагає

аналізу роботи одразу множини вузлів. Вони дозволяють команді проекту стежити за станом усієї блокчейн-мережі, бачити проблеми з окремими нодами, детектувати появу DoS атак на мережу та багато іншого. Основні з них.

«Transactions-per-second». У випадку розподілених систем, TPS — це дуже неоднозначний показник, який не завжди відображає реальну якість сервісу, що надається користувачам. Вимірювання TPS прийшли до нас із розподілених баз даних. TPS у базах даних — це деякі стандартизовані для тесту транзакції або їх набори (скільки INSERT, скільки UPDATE, стільки DELETE на тлі постійних SELECT) для жорстко заданої конфігурації кластера або взагалі на одній машині. Ці метрики зазвичай дають лише наближені оцінки продуктивності розподілених БД або блокчейнів, так як час процесингу транзакції може сильно змінюватись залежно від множини факторів.

При підрахунку TPS час обробки транзакцій збирається з вузлів мережі [1], що зовсім коректно. Більш вірним є підрахунок часу з початку формування транзакції до моменту отримання достовірної інформації про включення даної транзакції в блокчейн. Але при цьому підході на різних клієнтських програмах результат може відрізнятися в кілька разів, навіть при відправленні ідентичної транзакції.

У блокчейн мереж транзакція формується і підписується клієнтським додатком, тобто продуктивність клієнтського пристрою прямо впливає на швидкість обробки транзакцій.

Перш ніж приступити до відправки транзакції, клієнтська програма має запросити стан блокчейна. Коли в мережі мало вузлів, це несуттєво, але коли мережа розростається, таких запитів стає багато і вони можуть вплинути на завантаженість мережі. Через це такі запити будуть оброблятися повільніше.

Далі клієнтський додаток має надіслати транзакцію в один із вузлів блокчейну. Цей вузол починає розповсюджувати інформацію про транзакцію через peer-topeer (p2p) мережу до тих пір, поки транзакцію не побачить вузол,

що формує блоки, і не додасть її в один із блоків. Так як клієнтська програма знає про хеш транзакції - їй потрібно дочекатися зміни стану блокчейна і перевірити, чи є хеш в нових блоках ланцюжка. У більшості сучасних p2p мереж використовують різні модифікації протоколу Kademila [3]. Загалом, цей вид мереж складніший, ніж звичні мережі централізованих сервісів. Аналіз утруднюється впливом таких факторів, як кількість активних нод, розташованих поруч, розмір блоків та транзакції.

Додавання транзакцій в блоки та включення їх у блокчейн має відбуватися максимально швидко, але можуть виникати випадки, коли два довгі конкуруючі ланцюжки, перемикаючись між собою, роблять зміни метаданих тисяч транзакцій [4]. Головним чинником на даному етапі буде обраний алгоритм консенсусу. До того ж, смарт-конракти є програмами, які потребують ресурсів для виконання [5]. А якщо смарт-контракт генерує великий масив даних, то клієнту знадобиться більше часу на прийом інформації при повільному з'єднанні.

У результаті можна виділити наступні категорії операцій, що проводяться в блокчейн мережах, які можуть вплинути на продуктивність: криптографічні перетворення; передача даних у p2p мережі; виконання смарт-контрактів; фіксація змін у блокчейні; отримання клієнтом оновлення стану

В основі технології блокчейн лежить криптографія - наука про методи забезпечення конфіденційності інформації та збереження її справжності. Головним елементом цієї системи є хеш-функції. Говорячи двома словами, хешування забезпечує безпеку і незмінність блокчейна.

Щоб блокчейн працював, має постійно відбуватися його оновлення - додавання записів про нові транзакції в мережі. Саме під час додавання до системи нової інформації вона стає найбільш уразливою для атак. Але завдяки суворій ієрархії у блокчейні гарантується справжність всіх записів та їх захист від несанкціонованої зміни. Використання хеш-функції гарантує незмінність

вже існуючого ланцюжка транзакцій. Новий блок посилається на хеш попереднього.

При функціонування мережі блокчейн у ній створюються нові і нові блоки. Для додавання нового блоку мережа має дійти консенсусу. Для цього майнери пропонують свої версії блоків, вони проходять верифікацію і мережа вибирає блок, який стає наступним елементом реєстру. Але багато майнерів пропонують однакові блоки. Тоді яким чином вибирається той блок, що стане наступною ланкою ланцюга? Все дуже просто: комп'ютери в мережі змагаються у своєрідній хеш-грі, перебираючи множини варіантів значення "нонс" (nonce). Це число, що дає певний хеш в комбінації з попередніми даними при введенні в хеш-функцію.

При знаходженні правильного значення, його додаванні в блокчейн і введенні в хеш-функцію формується рандомний хеш - так зване рішення задачі. Комп'ютер, який його обчислив, отримує «приз» — криптовалюту біткоїн. Таким чином підтримується консенсус у мережі блокчейн та запобігають хакерським атакам. Якщо дані спотворені, то вирішити завдання неможливо. На цьому заснований принцип блокчейну та механізм безпеки зберігання даних у ньому. Зараз практично неможливо зламати систему, роботу якої побудовано на використанні криптографічної хеш-функції. І тому знадобляться пограничні обчислювальні потужності і дуже багато часу вирішення завдання. З додаванням кожного нового елемента до системи, її стійкість до атак зростає. Технологія блокчейн за рівнем надійності багаторазово перевершує всі інші системи захисту.

Розроблене програмне рабезпечення дозволить порівнювати між собою алгоритми хешування які можуть бути застосованими в мережах. Розглянуті алгоритми дають змогу забезпечити надійну передачу інформації, на основі технології блокчейн. Роботу можна розширити та використовувати отримані результати для інформаційної безпеки.

# РОЗДІЛ 1 Принципи роботи технології блокчейн

Технологія блокчейн набула широкого поширення в останні роки завдяки своїм ключовим особливостям, таким як децентралізація, прозорість і незмінність даних. Однак існуючі загальнодоступні блокчейн-мережі, такі як Біткоїн та Ефіріум, стикаються з низкою проблем, таких як відносно низька швидкість транзакцій та обмежена масштабованість.

Метою роботи є аналіз архітектури та функціональності існуючих протоколів блокчейну, розробка альтернативного високопродуктивного протоколу передачі даних ETAMP (Encrypted Token and Message Protocol) та демонстрація його переваг над існуючими рішеннями.

У дослідженні розглядаються основні концепції блокчейну, існуючі рішення та їхні технічні характеристики. Крім того, запропоновано архітектуру протоколу ETAMP та детально описано його функціонування, криптографічні механізми та структури даних.На основі ETAMP буде розроблено програмне забезпечення та проведено тестування продуктивності в порівнянні з альтернативними блокчейн-рішеннями. Очікується, що запропонований протокол ETAMP продемонструє вищу швидкість транзакцій та масштабованість за рахунок оптимізації алгоритмів консенсусу, структур даних та криптографічних примітивів.

Дослідження надає як теоретичне, так і практичне розуміння архітектури ефективних блокчейн-протоколів та можливих шляхів їх вдосконалення

## 1.1 Швидкість популярних блокчейн-систем

Однією з ключових характеристик, яка визначає продуктивність блокчейн-мережі, є швидкість транзакцій, тобто кількість транзакцій, які система може обробити за певний проміжок часу. В цьому розділі ми розглянемо показники швидкості та пропускної здатності деяких популярних блокчейн-платформ, щоб зрозуміти їх стан і тенденції розвитку. Це допоможе

визначити напрям для поліпшення існуючих рішень. Ось деякі репрезентативні дані про швидкість транзакцій і безпеку для популярних блокчейн-мереж:

*Bitcoin:*• Швидкість: 3-7 операцій/секунду

• Безпека: дуже висока, має алгоритм угоди про підтвердження роботи

*Ethereum:* • Швидкість: 10-15 операцій в секунду

• Безпека: висока, використовує доказ роботи

*Ripple:* • Швидкість: близько 1 500 операцій в секунду

• Безпека: висока, використовує консенсус Ripple

*Stellar* • Швидкість: до 1 000 операцій в секунду

• Безпека: висока, використовує консенсус Stellar

*EOS* •Швидкість: До 4 000 операцій в секунду

• Безпека: висока, використовує делегований доказ частки

*Cardano* • Швидкість: 250 операцій/с.

• Безпека: Висока, використовує Ouroboros Proof of Stake

Таким чином, мережі відрізняються за швидкістю та безпекою залежно від використовуваного алгоритму консенсусу. Біткоїн та Ефіріум відносно повільні, але дуже безпечні. Швидші мережі, такі як Ripple і EOS, досягають швидкості за рахунок певної децентралізації

Швидкість транзакцій у блокчейн-системах залежить від багатьох факторів, включаючи механізми консенсусу, розмір блоку та час створення блоку. Визначимо чому ж популярні блокчейн-системи працюють повільно:

1. Механізми консенсусу: доказ роботи (PoW): Біткоїн та Ефіріум в основному використовують алгоритм підтвердження роботи (Proof-of-Work) для перевірки транзакцій, що вимагає значних обчислювальних ресурсів і часу. Цей механізм допомагає захистити мережу, зменшуючи швидкість транзакцій.

2. Розмір блоку і час створення блоку: Блокчейн Bitcoin створює новий блок кожні 10 хвилин, в той час як Ethereum створює новий блок приблизно кожні 15

секунд. Обмеження розміру блоку і час створення блоку впливають на кількість транзакцій, які можуть бути оброблені за одиницю часу.

3. Швидкість поширення мережі: час, за який новий блок поширюється мережею, також впливає на швидкість транзакцій.

4. Вимоги до безпеки і децентралізації: високий ступінь децентралізації може уповільнити швидкість транзакцій, оскільки для досягнення консенсусу між учасниками мережі потрібен час.

5. Оптимізація та масштабування: такі блокчейни, як Ripple та EOS, використовують різні механізми консенсусу та оптимізації, які дозволяють їм обробляти більше транзакцій в секунду, але за рахунок децентралізації та зниження рівня безпеки.

### 1.2. Як працює блокчейн?

По-перше, необхідно зрозуміти, що таке блокчейн. Блокчейн - це децентралізована система записів, де інформація зберігається у вигляді серії блоків (рис 1).

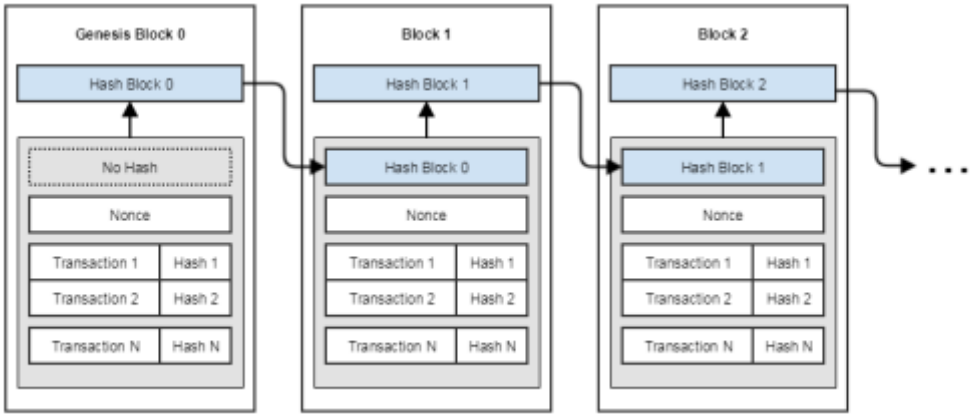


Рисунок 1. Схема блокчейн.

Основні особливості та принципи роботи блокчейну:

- Децентралізація: на відміну від традиційних баз даних, якими керує одна організація, блокчейн децентралізований, і дані розподіляються між багатьма учасниками або "вузлами" мережі..

- Блоки: кожен блок містить серію транзакцій, позначку часу і криптографічне посилання (хеш-посилання) на попередній блок. Це формує пов'язаний список або "ланцюжок" блоків
- Консенсус: перш ніж новий блок може бути доданий до блокчейну, вузли мережі повинні дійти згоди або "консенсусу". Існують різні механізми консенсусу, такі як proof-of-work (PoW) і proof-of-stake (PoS).
- Безпека. Через природу криптографії та хеш-функцій зміна інформації в одному блоці вимагає перерахунку всіх наступних блоків, що практично неможливо без контролю над більшою частиною обчислювальних потужностей мережі.
- Прозорість: всі транзакції в блокчейні є публічними і можуть бути перевірені будь-яким учасником мережі. При цьому особи учасників можуть бути анонімними або псевдоанонімними.
- Незмінність:після того, як транзакцію перевірено і додано до блокчейну, її неможливо змінити або видалити.
- Смарт-контракти: деякі блокчейни, такі як Ethereum, дозволяють створювати автоматизовані контракти, які виконуються при виконанні певних умов.
- Токени: блокчейн може використовувати токени або криптовалюти як засіб передачі вартості або представлення активів.

Послідовність дій можна описати наступним чином:

1) Коли користувач ініціює транзакцію, наприклад, надсилає криптовалюту іншому користувачеві, транзакція шифрується за допомогою спеціалізованих криптографічних алгоритмів. Такий процес не тільки захищає конфіденційність даних, але й забезпечує цілісність інформації. Після шифрування транзакція надсилається на один з вузлів мережі для подальшої обробки та верифікації.

2) Обраний вузол мережі бере на себе відповідальність за перевірку транзакції. Він оцінює, чи відповідає транзакція встановленим умовам і правилам мережі, наприклад, чи достатньо коштів на рахунку відправника, чи дійсний підпис, чи не перевищено ліміт на кількість транзакцій та інші параметри. Цей процес є важливим етапом, оскільки він запобігає можливим шахрайським діям та забезпечує дотримання правил мережі.

3) Після успішної перевірки транзакція потрапляє в блок, який потім розсилається іншим вузлам мережі для додаткової верифікації. Цей етап забезпечує децентралізовану перевірку транзакції, де кожен вузол має можливість участі в процесі підтвердження.

4) На наступному етапі вузли мережі, залежно від прийнятого механізму консенсусу, змагаються або домовляються щодо додавання блоку до ланцюга блоків. Механізм консенсусу відіграє ключову роль у забезпеченні безпеки та довіри всередині мережі. Він запобігає можливості маніпулювання та несанкціонованого втручання.

5) Коли блок успішно додається до ланцюга, транзакція вважається підтвердженою і завершеною. Користувачі можуть бути впевнені в тому, що їх транзакція була оброблена належним чином та кошти були передані відповідно до їх інструкцій. Цей етап є остаточним в циклі обробки транзакції в рамках блокчейн мережі.

### 1.3 Складові блоків

Блок в блокчейні - це важливий об'єкт в ланцюжку. (рис 2)

Блок складається з декількох важливих компонентів.

1. Заголовок блоку:
   - Хеш попереднього блоку.
   - Хеш поточного блоку
   - Мітка часу.
   - Номер блоку або висота блоку

- Доказ роботи (для систем, що використовують доказ роботи)
- Інші метадані.

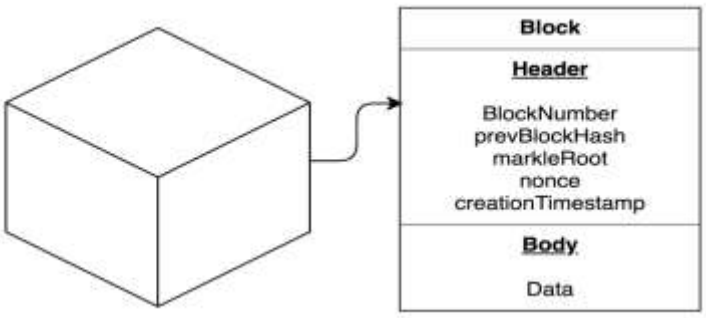


Рисунок 2 Складова блока.

2. Список транзакцій. Кожна з транзакцій в блоці містить інформацію про передачу цінності (наприклад, криптовалюти) від одного учасника до іншого. Транзакції містять наступні елементи:
    - Входи транзакції
    - Виходи транзакції
    - Криптографічний підпис
3. Інші компоненти блокчейну

Залежно від застосування та призначення блокчейну, блок може також містити додаткові дані, такі як смарт-контракти, дані голосування, інформаційні повідомлення тощо.

Опишемо детальніше терміни що застосовуються:

Хеш попереднього блоку: криптографічне представлення всієї інформації в попередньому блоці. Це забезпечує зв'язок між блоками і формує ланцюжок блоків.

Хеш поточного блоку: зашифрована версія всієї інформації в поточному блоці. Якщо хоча б один біт інформації в блоці зміниться, хеш блоку також зміниться.

Мітка часу: мітка часу, коли блок було створено або додано до ланцюжка.

Номер блоку або висота блоку: порядковий номер блоку в ланцюжку.

Доказ роботи (для систем, що використовують доказ роботи): значення, яке підтверджує, що вузол витратив певну кількість обчислювальних ресурсів на створення блоку.

Інші метадані: в залежності від реалізації блокчейну, можуть бути й інші елементи заголовка.

Вхідні дані транзакції: посилання на вихідні дані попередньої транзакції, які використовуються як вхідні дані для поточної транзакції. Результат транзакції: містить інформацію про отримувача та суму переказу.

Криптографічний підпис: підтверджує, що особа, яка ініціює транзакцію, уповноважена здійснювати її.

Кожна мережа блокчейн може мати свою власну структуру, яка відрізняється від інших, і це загальний приклад того, як це може виглядати

### 1.4 Яким чином працюють методи консенсусу?

Алгоритм консенсусу - це механізм, за допомогою якого користувачі та програми координують свою поведінку в розподіленій мережі (Рис 3). Він

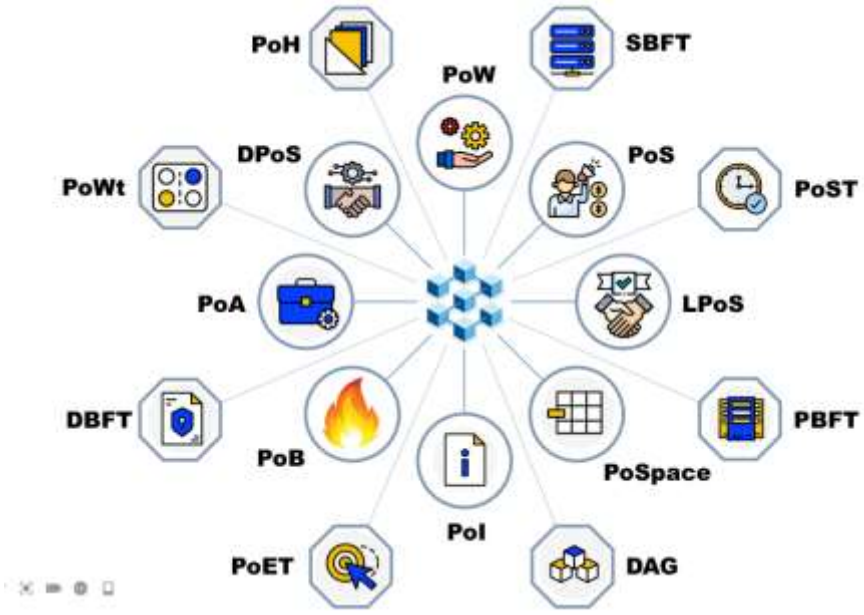


Рисунок 3. Схема мережі

гарантує що всі учасники мережі погоджуються з поточним станом даних, навіть якщо деякі вузли виходять з ладу. Іншими словами, консенсус допомагає підтримувати стійкість системи.

В централізованих структурах один суб'єкт має владу над усією системою. В більшості випадків ця особа має право вносити зміни на власний розсуд. Не існує складної системи управління для досягнення консенсусу між кількома адміністраторами. У розподіленому середовищі механізми роботи зовсім інші. Наприклад, якщо взяти розподілену базу даних, то як досягти консенсусу між усіма учасниками мережі щодо того, які дані можуть бути додані?

Досягнення консенсусу середовищі, де сторони не завжди можуть довіряти одна одній, стало однією з найважливіших подій в індустрії, яка проклала шлях до появи блокчейну. Тому акцентується важливість алгоритмів консенсусу для належного функціонування криптовалют і розподілених реєстрів.

Тут коротко переглянемо деякі поширені механізми консенсусу:

1. Proof of Work (PoW, Доказ роботи): учасники мережі (майнери) змагаються у вирішенні складних математичних задач для створення нових блоків. Перший майнер, який вирішив задачу, додає новий блок до блокчейну і отримує винагороду.
2. Proof of Stake (PoS, Доказ частки): учасники мережі з певною кількістю монет (share) можуть підтверджувати нові блоки. Чим більше у учасника частки, тим більша ймовірність того, що він буде обраний для створення нового блоку; PoS вважається більш енергоефективним, ніж PoW.
3. Delegated Proof of Stake (DPoS, Делегований доказ частки): учасники мережі обирають невелику кількість делегатів для перевірки транзакцій і створення нових блоків; DPoS збільшує швидкість транзакцій і масштабованість, але може зменшити децентралізацію.

4. Practical Byzantine Fault Tolerance (PBFT, Практична візантійська стійкість до помилок): досягнення консенсусу через серію голосувань, навіть коли учасники мережі діють недобросовісно. Це дуже складний і ефективний механізм для забезпечення безпеки в довірчому середовищі.

5. Ripple Protocol Consensus Algorithm (RPCA, Алгоритм консенсусу протоколу Ripple): учасники мережі (валідатори) голосують за правильність транзакції кожні кілька секунд і досягають консенсусу після декількох раундів голосування. Це дозволяє Ripple швидко обробляти транзакції.

6. Stellar Consensus Protocol (SCP, Протокол консенсусу Stellar): SCP використовує концепцію основних зрізів для досягнення консенсусу між учасниками мережі. Це дозволяє Stellar швидко і надійно затверджувати транзакції.

Наведені механізми консенсусу мають свої переваги і недоліки в залежності від конкретного застосування і вимог до безпеки, швидкості і децентралізації

### 1.5 Безпека блокчейна

Безпека в блокчейн-системах - це складний і багаторівневий процес. Розглянемо докладніше кожен аспект безпеки:

1. Криптографічні функції

Хеш: Блокчейн використовує криптографічні хеш-функції (наприклад, SHA-256 у біткоїні) для створення унікального ідентифікатора (хешу) для кожної транзакції та блоку. Хеш діє як цифровий відбиток пальців, і навіть невеликі зміни в даних можуть створити зовсім інший хеш.

Цифрові підписи: учасники блокчейну підписують транзакції за допомогою своїх приватних ключів. Це дозволяє перевірити, що транзакцію створив конкретний учасник, не розкриваючи його особу.

2. Розподілений реєстр (розподілена база даних). Блокчейн - це розподілений реєстр, де кожен вузол зберігає точну копію всієї історії

транзакцій. Це зменшує можливість централізованих атак і створює відмовостійку систему.

3. Цілісність блоків. Кожен блок у блокчейні містить хеш попереднього блоку і формує ланцюжок блоків. Тому зміна попереднього блоку вимагає зміни всіх наступних блоків, що є нездійсненним з обчислювальної точки зору.

4. Механізми консенсусу.

Proof of Work (PoW): учасники мережі (майнери) змагаються у вирішенні складних математичних задач для додавання нових блоків. Це вимагає великих обчислювальних ресурсів.

Proof-of-Stake (PoS): учасники мережі доводять свою "частку" в мережі для створення нових блоків.

5. Децентралізація. Децентралізована природа блокчейну знижує ризики, пов'язані з єдиною точкою відмови та централізованим контролем.

6. Прозорість та можливість аудиту. Блокчейн забезпечує повну прозорість усіх транзакцій і полегшує аудит та перевірку історії транзакцій.

7. Дозволи та доступ. У приватному блокчейні система авторизації контролює, хто може переглядати дані та здійснювати транзакції.

8. Смарт-контракти автоматизують виконання умов контрактів без посередників. Смарт-контракти тестуються та перевіряються на наявність вразливостей, щоб забезпечити безпечне виконання.

9. Системи управління ідентифікацією. Блокчейн забезпечує безпечну систему управління ідентифікацією, що дозволяє аутентифікацію та авторизацію учасників.

10. Регулярні оновлення та виправлення. Розробники мереж блокчейн регулярно випускають оновлення та патчі для усунення вразливостей і підвищення безпеки мережі.

Кожен з цих елементів відіграє важливу роль у безпеці блокчейну, а їхня ефективна взаємодія створює безпечне та надійне середовище для транзакцій та управління даними.

### 1.6. Токени


Токени є важливим елементом багатьох блокчейн-систем і являють собою тип цифрового активу або одиницю вартості, що випускається на блокчейн-платформах. Токени мають різні функції і можуть представляти різні типи активів або прав. Переглянемо характеристики токенів:

1. Визначення та функції. Токени можуть представляти активи (наприклад, акції, облігації, грошові кошти), права на доступ до певних послуг або функцій у системі блокчейн або внутрішні валюти.
2. Створення та управління. Токени створюються за допомогою смарт-контрактів на блокчейн-платформах, що підтримують смарт-контракти, таких як Ethereum, Binance Smartchain і Cardano. Смарт-контракти визначають правила і логіку роботи токенів, включаючи їх випуск, передачу і знищення.
3. Стандарт токенів. Існують різні стандарти токенів, які визначають функціональність і взаємодію токенів. Наприклад, в Ethereum є ERC-20 і ERC-721.
4. Типи токенів.

Ф'ючерсні токени (утилітарні): надають доступ до певних послуг або функцій в системі блокчейн.

Токени цінності (токени безпеки): представляють собою цифрові версії традиційних цінних паперів, таких як акції та облігації.

Колекційні токени (NFT, Non-Fungible Tokens): унікальні токени, кожен з яких має відмінні від інших характеристики.

5. Торгівля і передача. Токени можна передавати між учасниками мережі або торгувати ними на криптовалютних біржах.

6. Регулювання. Регулювання токенів залежить від юрисдикції та типу токену. Наприклад, токени вартості зазвичай підпадають під регулювання цінних паперів.

7. Оцінка і ліквідність. Вартість токенів може коливатися в залежності від попиту, пропозиції та загальних ринкових умов. Деякі токени мають високу ліквідність, в той час як інші мають обмежену ліквідність.

8. Безпека. Безпека токена залежить від безпеки блокчейн-системи, з якої він був випущений, і безпеки смарт-контракту, який управляє токеном.

Токени відіграють важливу роль в екосистемі блокчейну, забезпечуючи нові форми обміну, фінансування, торгівлі та створення цифрових активів.

## Висновки по першому розділу

У першому розділі було розглянуто основні концепції та принципи роботи технології блокчейн.

Блокчейн - це децентралізований реєстр, де дані зберігаються у вигляді послідовності блоків, пов'язаних криптографічними зв'язками. Основними характеристиками блокчейну є децентралізація, прозорість, безпека та незмінність даних.

Розглянуто роботу популярних блокчейн-систем, таких як Bitcoin, Ethereum, Ripple тощо. Показано, що вони мають обмежену швидкість транзакцій через використання алгоритмів консенсусу на основі доказу роботи, а також через обмеження розміру блоків.

Детально проаналізовано складові блоків в блокчейні та роль криптографічного хешування для забезпечення безпеки.

Розглянуто роботу різних алгоритмів досягнення консенсусу, зокрема доказу роботи, доказу частки, практичної візантійської стійкості до помилок тощо.

Отже, в першому розділі надано загальне уявлення про технологію блокчейну, її можливості та обмеження. Це дозволяє перейти до наступного розділу, де буде запропоновано альтернативний високопродуктивний протокол передачі даних на основі оптимізованих алгоритмів консенсусу та криптографії.

# РОЗДІЛ 2 Архітектура, алгоритми та протоколи ETAMP

Для подолання таких недоліків існуючих блокчейн-систем як обмеження швидкості та масштабованості пропонується альтернативний протокол - протокол зашифрованих токенів і повідомлень (ETAMP), заснований на мережевій архітектурі mesh-зірка, що складається з серверів і підключених до них клієнтів. Кожен сервер в мережі ETAMP є незалежним і може спілкуватися з іншими серверами, використовуючи протокол ETAMP. Сервери містять різні модулі, включаючи модулі аутентифікації, модулі обміну повідомленнями та модулі криптовалют. Така архітектура дозволяє ETAMP забезпечувати швидку і масштабовану передачу даних. Крім того, протокол ETAMP включає криптографічні механізми для захисту конфіденційності та цілісності даних.

ETAMP - це абревіатура від Encrypted Token and Message Protocol (протокол зашифрованих маркерів і повідомлень). Це криптографічний протокол, призначений для безпечного та конфіденційного обміну повідомленнями та транзакціями в розподілених мережах. Основні особливості протоколу ETAMP наступні:

Криптографічні механізми, такі як цифрові підписи та шифрування, використовуються для гарантування автентичності, цілісності та конфіденційності даних, що передаються.

Повідомлення і транзакції інкапсулюються в спеціальну структуру, яка включає ідентифікатор, токен у форматі JWT і цифровий підпис повідомлення з токеном.

Він підтримує різні типи повідомлень і транзакцій, такі як передача значень, обмін ключами, реєстрація користувачів і сповіщення.

Дозволяє ідентифікувати відправників і одержувачів за допомогою цифрових ідентифікаторів.

Забезпечує відстежуваність і можливість перевірки повідомлень і транзакцій.

Масштабований для використання в розподілених мережах з великою кількістю учасників.

Таким чином, ETAMP забезпечує надійний та ефективний спосіб передачі конфіденційних даних у розподіленому середовищі.

Протокол ETAMP було розроблено з особливим акцентом на безпеку, що є його основоположною характеристикою. Механізм забезпечення безпеки в цьому протоколі полягає у використанні цифрових підписів, які слугують для верифікації авторства та цілісності переданих даних. Цифрові підписи та шифрування повідомлень у протоколі ETAMP здійснюються на основі технології еліптичних кривих, яка являє собою сучасний і надійний метод криптографічного захисту. Еліптичні криві мають низку переваг, включно з високим рівнем безпеки за відносно меншої довжини ключа, порівняно з іншими криптографічними методами, що забезпечує вищу ефективність і продуктивність. Для забезпечення структурованого та безпечного передавання даних, усі передані повідомлення інкапсулюються у форматі JWT (Json Web Token). Формат JWT дає змогу подати інформацію в компактному і самодостатньому вигляді, що полегшує її передачу та обробку.

Протокол ETAMP застосовує асиметричне шифрування для створення повідомлень, транзакцій або будь-яких інших структур даних, що забезпечує високий рівень безпеки переданої інформації. Всі дані, які передаються через протокол, мають бути надійно підписані без можливості їхньої компрометації. Для цього використовується алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm), який забезпечує створення цифрових підписів, та алгоритм ECDH (Elliptic Curve Diffie-Hellman) для шифрування повідомлень.

Також у протоколі ETAMP реалізовано стандарт ключів у форматі PEM. Формат PEM (Privacy Enhanced Mail) дає змогу зберігати й передавати

криптографічні ключі в текстовому форматі, що спрощує їхнє опрацювання та управління. Це робить протокол ETAMP зручним і гнучким варіантом для забезпечення безпеки в сучасних системах обміну даними.

## 2.1. Складові протоколу.

### *2.1.1.Еліптична криптографія*

Еліптичні криві є фундаментальним об'єктом сучасної криптографії, що забезпечує високий ступінь безпеки та ефективності. Еліптичні криві описуються математичним рівнянням $y^2 = x^3 + ax + b$, де a і b - константи. Це, здавалося б, просте рівняння приховує глибокі математичні властивості і можливості, які широко застосовуються в області криптографії.

Однією з найважливіших властивостей еліптичних кривих є додавання точок. Якщо ви візьмете дві точки на кривій і проведете через них пряму, то ця пряма переріже криву в третій точці. Цю третю точку можна використовувати як результат додавання двох початкових точок. Ця властивість лежить в основі визначення арифметики точок на еліптичних кривих. Іншою важливою властивістю є властивість подвоєння точки. Якщо дотична проведена до кривої в одній точці, то дотична перерізає криву в іншій точці, яка вважається результатом подвоєння початкової точки. Ці дві операції - додавання точок і подвоєння точок - дозволяють визначити алгебраїчну структуру для множини точок на еліптичній кривій.

Проблема дискретних логарифмів на еліптичній кривій є досить виснажливою:

Знайти число k таке, що kP = Q, коли P і Q - точки на кривій.

Ця обчислювальна складність робить еліптичні криві привабливими для побудови криптосистем. У криптосистемі на еліптичній кривій секретним ключем є число k, а відкритим ключем - точка kP на кривій. Щоб створити цифровий підпис, підписувач обчислює число k і відповідну точку kP і надсилає точку kP одержувачу. Одержувач використовує точки kP і підпис для перевірки

повідомлення. Така структура робить криптосистеми еліптичних кривих ефективними та надійними, забезпечуючи при цьому автентифікацію та шифрування даних

### *2.1. 2 Задача дискретного логарифмування (DLP)*

Задача дискретного логарифмування (DLP) є фундаментальною для криптографії і лежить в основі багатьох криптосистем. Задача формулюється наступним чином: Маючи елемент g деякої групи G і ще один елемент h тієї самої групи, завдання полягає в тому, щоб знайти таке ціле число x, що $g^x = h$ у групі G.

У класичному DLP група $G$ зазвичай є групою мультиплікативних елементів скінченного поля $F_p$, де $p$ - просте число. У цьому контексті операції множення та впорядкування виконуються за модулем $p$. Отже, задача дискретного логарифмування у скінченних полях полягає у наступному: Потрібно знайти $x$ таке, що $g^x = h \bmod p$. В даний час DLP для великих значень $p$ вважається складною задачею. Це пов'язано з тим, що не існує ефективного алгоритму для знаходження $x$ в загальному випадку. Найшвидший відомий метод розв'язання DLP має експоненціальну складність по відношенню до розміру області, що робить його нерозв'язною проблемою для достатньо великих значень $p$.

Складність розв'язання DLP лежить в основі криптосистем з відкритим ключем, таких як система Діффі-Хеллмана для обміну ключами та алгоритм Ель-Гамаля для шифрування. У цих системах складність DLP гарантує, що зловмисник не зможе легко обчислити секретний ключ, використовуючи лише відкритий ключ. Проблему DLP можна узагальнити на інші алгебраїчні структури, що включають еліптичні криві, що призводить до формулювання проблеми дискретного логарифмування еліптичних кривих (ECDLP). Ця версія задачі формулюється аналогічно, але в контексті груп точок на еліптичних кривих, і розширює застосування дискретних логарифмів у криптографії.

Таким чином, проблема дискретних логарифмів продовжує представляти інтерес в криптографії, а розуміння її складності і застосувань є ключем до розуміння багатьох сучасних криптосистем

### *2.1.3. Сингулярність*

Сингулярність у контексті алгебраїчних кривих позначає точку, в якій крива не має добре визначеної дотичної, або ж має більше однієї дотичної. Це може статися, наприклад, якщо крива має "гостру" вершину або сама перетинається в якійсь точці. Сингулярності роблять криву складнішою для аналізу і можуть призвести до проблем у застосуваннях, таких як криптографія, де арифметика точок на кривій відіграє ключову роль.

Умова $4a^3 + 27b^2 \neq 0$ пов'язана з уникненням сингулярності в еліптичних кривих. Еліптичні криві в загальному вигляді задаються рівнянням y^2=x^3+ax+b. Щоб крива була справді еліптичною і не мала сингулярностей, необхідно, щоб рівняння $4a^3 + 27b^2 \neq 0$.було виконано. Ця умова гарантує, що крива не має точок самоперетину і не має "гострих" вершин, що робить можливою коректну арифметику точок на кривій. Дискримінант еліптичної кривої, який визначено як $\Delta = -16(4a^3 + 27b^2)$ , відіграє ключову роль у визначенні сингулярності. Якщо $\Delta \neq 0$, крива є невиродженою, тобто без сингулярностей, що важливо для криптографічного використання еліптичних кривих. На практиці, у криптографії обирають такі коефіцієнти $a$ і $b$, щоб умова $4a^3 + 27b^2 \neq 0$ була виконана, забезпечуючи тим самим гладкість кривої і можливість ефективного виконання арифметичних операцій на ній.

Таким чином, ця умова є критично важливою для забезпечення коректності та безпеки криптосистем на еліптичних кривих, і відіграє центральну роль у виборі підходящої еліптичної кривої для криптографічних додатків

### *2.1.4 ECDSA*

Elliptic Digital Signature Algorithm (ECDSA) є одним із ключових компонентів сучасної криптографії на еліптичних кривих. Цей алгоритм відіграє центральну роль у генерації та перевірці цифрових підписів повідомлень, використовуючи механізми криптографії еліптичних кривих.

Основні алгебраїчні структури, що використовуються в ECDSA, включають кінцеве поле та еліптичну криву, визначену над цим полем. Для роботи з кривою обирають рівняння, що описує її, і порядок базової точки, що забезпечує основу для всіх наступних операцій.

Процес генерації ключів в ECDSA починається з вибору закритого ключа, який являє собою випадкове число. Потім обчислюється відкритий ключ шляхом множення закритого ключа на базову точку кривої. Ця операція створює точку на еліптичній кривій, яка і буде використовуватися як відкритий ключ.

Підписання повідомлення включає в себе кілька кроків. Спочатку обчислюється хеш повідомлення, що дає змогу отримати унікальне числове представлення повідомлення фіксованого розміру. Потім генерується випадкове число k, яке відіграє роль тимчасового ключа для цієї операції. Використовуючи k, хеш повідомлення і закритий ключ, обчислюються компоненти цифрового підпису. Ці компоненти являють собою математичне підтвердження того, що підпис було створено власником закритого ключа.

Для перевірки підпису використовуються компоненти підпису, відкритий ключ і хеш повідомлення. Виконується низка математичних операцій, щоб перевірити рівність, яка підтверджує або відкидає коректність підпису. Ця рівність виконується тільки в разі, якщо підпис дійсно було створено власником відповідного закритого ключа.

Безпека ECDSA значною мірою залежить від складності розв'язання задачі дискретного логарифмування на еліптичній кривій. Ця складність

гарантує, що, не маючи доступу до закритого ключа, зловмисник не зможе ефективно підробити підпис.

ECDSA знайшов широке застосування у світі криптовалют, зокрема, у таких системах, як Bitcoin та Ethereum для підписання транзакцій. Ефективність і криптостійкість ECDSA роблять його чудовим вибором для забезпечення безпеки та цілісності даних у блокчейн-мережах. Тому його було обрано як основний механізм підпису в протоколі ETAMP

### *2.1.5 ECDH*

Elliptic-curve Diffie–Hellman (ECDH) є криптографічним методом, що дає змогу двом сторонам, кожна з яких має пару відкритих і закритих ключів, створити спільний секретний ключ, використовуючи еліптичні криві, не передаючи закритих ключів одна одній. Цей спільний секретний ключ потім може бути використаний для шифрування наступних повідомлень між сторонами.

На початку кожна сторона вибирає свій власний закритий ключ і обчислює відповідний відкритий ключ, помноживши закритий ключ на базову точку еліптичної кривої. Ці відкриті ключі потім обмінюються між сторонами. Після отримання відкритого ключа іншої сторони, кожна сторона множить отриманий відкритий ключ на свій власний закритий ключ, щоб отримати загальний секретний ключ. Завдяки властивостям еліптичних кривих, обидва учасники отримують один і той самий секретний ключ, який потім можна використовувати для шифрування даних.

Математична безпека ECDH заснована на складності завдання обчислення дискретного логарифма на еліптичних кривих. Це означає, що навіть якщо зловмисник перехопить обмін відкритими ключами, без знання закритих ключів йому буде вкрай складно обчислити загальний секретний ключ.

Застосування ECDH особливо корисно в сценаріях, де сторони можуть обмінюватися відкритими ключами відкрито, але потрібно захистити загальний

секретний ключ від доступу третіми сторонами. Це робить ECDH чудовим вибором для створення безпечних каналів зв'язку в сучасних криптографічних системах і мережевих протоколах.

ECDH не тільки забезпечує високий рівень безпеки, а й є відносно ефективним з погляду обчислень, що робить його привабливим вибором в ресурсно обмежених середовищах, таких як мобільні пристрої та вбудовані системи. Ефективність і безпека ECDH, роблять його важливим інструментом для забезпечення конфіденційності даних у цифровому світі

### *2.1.6 JWT*

JSON Web Token (JWT) є компактним, самодостатнім способом передачі інформації між сторонами у форматі JSON. Цю інформацію можна підтвердити та довірити, оскільки кожен токен підписується секретним ключем або парою ключів (відкритим і закритим). JWT часто використовують для автентифікації та авторизації в сучасних веб-додатках і мікросервісах.

Токен JWT складається з трьох частин: заголовка, корисного навантаження і підпису.

- Заголовок (**Header**). Заголовок зазвичай складається з двох частин: типу токена, який є JWT, і алгоритму підпису, наприклад HMAC SHA256, SHA512, ECDSA або RSA. Ця частина кодується в Base64Url і потім конкатенується з крапкою.
- Корисне навантаження (**Payload**). Корисне навантаження містить твердження або заяви про сутність (часто користувача) і додаткові дані. Існує три типи тверджень: зареєстровані, загальнодоступні та приватні твердження. Ця частина також кодується в **Base64Url** і конкатенується з крапкою.
- Підпис (**Signature**). Щоб створити підпис, потрібно взяти закодовані рядки заголовка і корисного навантаження, додати секретний ключ, а потім застосувати алгоритм підпису, зазначений у заголовку. Підпис використовується для верифікації, що повідомлення не було змінено на шляху.

Збираючи все разом, виходить рядок у форматі **header.payload.signature**, який являє собою токен JWT.

JWT має багато застосувань. Одним із найпоширеніших є аутентифікація користувачів. Після входу в систему користувачеві видається токен JWT, який потім він може використовувати для автентифікації себе під час наступних запитів. Це зменшує необхідність повторної автентифікації користувача та спрощує процес автентифікації в мікросервісній архітектурі. Також JWT може бути використаний для передачі безпечної та надійної інформації між сторонами. Оскільки кожен токен може бути підписаний, одержувач може бути впевнений у тому, що інформація не була змінена після створення токена.

Проте існують і певні ризики при використанні JWT. Наприклад, якщо зловмисник отримує доступ до токена, він може використовувати його для імперсонації користувача. Крім того, якщо використовується слабкий ключ підпису або вразливий алгоритм підпису, зловмисник може створювати підроблені токени. Загалом, JWT являє собою потужний і гнучкий спосіб управління ідентифікацією та передавання даних, що робить його важливим інструментом у сучасній веб-розробці та мікросервісах.

### *2.1.7. PEM*

Формат Privacy-Enhanced Mail (**PEM**) є широко використовуваним форматом для зберігання і передачі криптографічних ключів і сертифікатів у текстовому вигляді. Цей формат спочатку розробили для забезпечення безпеки електронної пошти, але відтоді він став стандартом для зберігання та обміну криптографічною інформацією в багатьох додатках і системах.

Файли у форматі PEM мають розширення .pem і складаються із заголовка, тіла і підвалу. Вміст являє собою базове кодування 64 (Base64) двійкових даних, укладених між рядками початку і кінця.

Заголовок і фінал: Заголовок починається рядком "-----BEGIN", за яким слідує тип об'єкта, і завершується рядком "-----". Аналогічно, фінал починається

з рядка "-----END ", за яким слідує тип об'єкта, і завершується рядком "-----". Тип об'єкта вказує на те, які дані містяться у файлі PEM, наприклад, "CERTIFICATE", "PRIVATE KEY" або "PUBLIC KEY". Тіло: Тіло файлу містить дані, закодовані в Base64. Це може містити, наприклад, відкриті або закриті ключі, сертифікати або інші криптографічні об'єкти.

Приклад файлу PEM із сертифікатом може мати такий вигляд:

```
-----BEGIN CERTIFICATE-----
MIIDXTCCAkWgAwIBAgIJALm157+vOUdEMAOGCSqGSIb3DQEBCwUAMEUxCzAJBgNV
...
-----END CERTIFICATE-----
```

Перевага формату PEM полягає в тому, що він дає змогу зберігати криптографічну інформацію в текстовому форматі. Це спрощує її зчитування та обмін між системами. До того ж, текстовий формат уможливлює включення криптографічних об'єктів до конфігураційних файлів або електронних листів без необхідності прикріплення окремих двійкових файлів.

Однак варто пам'ятати, що формат PEM не пропонує вбудованого захисту від несанкціонованого доступу. Закриті ключі та інші чутливі криптографічні об'єкти мають бути належним чином захищені під час зберігання та передавання, незважаючи на те, що вони зберігаються у форматі PEM

### *2.1.8 КривaNistP-521*

У протоколі ETAMP використовується алгоритм ECDSA за кривою nistP-521. **NIST P-521** - одна з кривих, визначених Національним інститутом стандартів і технологій (NIST) для використання в криптографії еліптичних кривих. Ця крива має довжину поля 521 біт і є частиною сімейства кривих, обраних для забезпечення високого рівня безпеки та обчислювальної ефективності. В контексті алгоритму цифрового підпису з еліптичної кривої (ECDSA), NIST P-521 використовується для створення цифрових підписів. Процес створення та перевірки підписів з використанням цієї кривої подібний

до процесу, що використовується для інших еліптичних кривих. Однак, головна відмінність полягає в розмірі ключа і, відповідно, в рівні безпеки, що забезпечується.

1. Генерація ключів: У ECDSA з NIST P-521 закритий ключ є випадковим числом, обраним з інтервалу від 1 до n-1, де n є порядком базової точки на кривій. Відкритий ключ обчислюється шляхом множення закритого ключа на базову точку кривої.

2. Підписання. Для підписання повідомлення генерується випадкове число k, обчислюється значення $R = k \cdot G$ (де G - базова точка), і потім обчислюється значення підпису з використанням формул ECDSA.

3. Перевірка підпису. Перевірка підпису містить кілька арифметичних операцій на еліптичній кривій, включно з додаванням точок і множенням скалярним, щоб упевнитися, що підпис було створено власником відповідного закритого ключа.

Використання NIST P-521 в ECDSA пропонує високий рівень безпеки завдяки великому розміру ключа. Це робить алгоритм стійким до атак на обчислення дискретного логарифма, які є основним вектором атаки проти криптосистем на еліптичних кривих. Великий розмір ключа також означає, що системи, які використовують NIST P-521, матимуть довгострокову безпеку навіть у міру поліпшення обчислювальних можливостей і методів атаки.

### 2.2. Як працює ETAMP

Протокол ETAMP, орієнтований на забезпечення високошвидкісного та масштабованого передавання даних, функціонує на основі гібридної архітектури, яка називається "зірка-сітка". Ця архітектура є сукупністю серверів і клієнтів, де кожен сервер є незалежним вузлом, здатним взаємодіяти з іншими серверами в мережі з використанням протоколу ETAMP. Сервери в архітектурі ETAMP володіють різними модулями, які включають, але не обмежуються модулями автентифікації, обміну повідомленнями та криптовалют. Ці модулі

забезпечують гнучкість і безпеку під час обробки та передачі даних між клієнтами та серверами, а також між самими серверами.

Під час надсилання повідомлення клієнтом у мережі ETAMP, повідомлення інкапсулюється в спеціальну структуру, яка забезпечує як ідентифікацію повідомлення, так і його безпеку. Структура повідомлення виглядає наступним чином і містить такі компоненти:

*{*
*"Id": "550e8400-e29b-41d4-a716-446655440000",*
*"Token": "eyJhbGciOiAiRVMyNTYiLCAidHlwIjogIkpXVCJ9...",*
*"SignatureToken": "MIHcAgEBBEIAM9C0sFzRRtxQCXYc9kujr....",*
*"SignatureMessage": "MIGbMBAGByqGSM49AgEGBSuBBAAj...."*
*}*

Id: Унікальний ідентифікатор повідомлення, який гарантує його відстежуваність у мережі.

Token: JWT (JSON Web Token) токен, що містить інформацію про повідомлення та його відправника.

SignatureToken: Цифровий підпис токена, створений з використанням алгоритму ECDSA (Elliptic Curve Digital Signature Algorithm), що забезпечує справжність токена.

SignatureMessage: Цифровий підпис усього повідомлення, включно з Id, Token і SignatureToken, що забезпечує цілісність і авторство повідомлення.

Застосування структурованого формату повідомлень і криптографічних методів у протоколі ETAMP забезпечує високий рівень безпеки та цілісності даних під час їх передавання. Ці механізми дають змогу впевнитися в тому, що повідомлення не було змінено або підроблено в процесі передання, а також гарантувати, що їх було надіслано уповноваженими суб'єктами

Протокол ETAMP втілює передові підходи для забезпечення безпеки і структурованості даних у блокчейн-мережах, використовуючи JWT (JSON Web

Token) для представлення та передачі інформації. Кожен JWT токен складається з трьох частин: заголовка, корисного навантаження та підпису.

- **Заголовок** містить інформацію про тип токена і алгоритм шифрування, гарантуючи, що токен можна правильно обробити.
- **Корисне** навантаження включає в себе клейми, які представляють собою деталізовані дані про користувача, повідомлення, транзакції та інші важливі аспекти взаємодії в мережі.
- **Підпис** забезпечує автентичність токена, переконуючись, що він не був змінений під час передачі.Типи клейм і їх роль в ETAMP

*Типи клейм і їх роль в ETAMP*

Клейми в ETAMP можна класифікувати на зареєстровані, публічні та приватні:

- **Зареєстровані** клейми включають стандартизовані поля, такі як ідентифікатор видачі токена, час життя токена та інші.
- **Публічні** клейми створені спеціально для потреб ETAMP і можуть включати ідентифікатори повідомлень, відправника, отримувача та інші необхідні дані.
- **Приватні** клейми використовуються для специфічних випадків і додатків, що працюють на базі ETAMP.

Поля *Issuer* (iss) і *Audience (aud)* надають додатковий рівень безпеки і ідентифікації учасників, допомагаючи встановити, що токен був виданий авторизованим відправником і призначений для конкретного одержувача.

Наприклад:

*"iss": "B2F7714F-891F-4A4F-93F0-D472309DE102.WebServer1.Alice.524B496E-BDE9-473B-A143-55A7F590C373"*

*"aud": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108.WebServer2.Bob.2480756A-DCBB-498B-A92D-AF1093BA26D6"*

*Процес підписання та верифікації повідомлень*

Всі повідомлення в ETAMP підписуються за допомогою ECDSA, гарантуючи цілісність даних та аутентичність відправника. Процес підписання

включає створення цифрового підпису, його включення в повідомлення та передачу одержувачу. Одержувач може перевірити підпис, використовуючи відкритий ключ відправника.

*Переваги використання клейм в ETAMP*

Клейми дозволяють передавати перевірену інформацію в компактному форматі, забезпечуючи гнучкість і розширюваність системи. Вони полегшують процес верифікації та аутентифікації, роблячи протокол ETAMP більш надійним та ефективним. Клейми в JWT токенах відіграють ключову роль в протоколі ETAMP, забезпечуючи безпечний, гнучкий і ефективний механізм обміну даними. Вони сприяють реалізації потенціалу блокчейн-технологій, підвищуючи при цьому рівень безпеки та прозорості в системі. Їх використання дозволяє ETAMP стати більш адаптивним до різних сценаріїв використання та потреб користувачів

У протоколі ETAMP визначено різні типи оновлень або повідомлень, які можуть передаватися між учасниками мережі. Зазначення типу оновлення допомагає системі правильно інтерпретувати й обробляти вхідні повідомлення. Ось деякі приклади типів оновлень у цьому протоколі:

- Message (Повідомлення). Полегшує текстове спілкування між користувачами. Деталі, як-от зміст, час відправлення, відправник і одержувач.
- Transaction (Транзакція). Передача значень або активів між учасниками. Деталі транзакції, такі як сума, валюта, відправник, одержувач і час транзакції.
- TransactionConfirmed (Підтвердження Транзакції). Підтвердження успішної транзакції. Деталі підтвердження, включно з часом підтвердження та ID підтвердження.
- TransactionPending (Транзакція, що очікує). Повідомлення про транзакцію, яка перебуває в очікуванні. Деталі, включно з часом початку транзакції та очікуваним часом підтвердження.

- TransactionCanceled (Скасована Транзакція). Повідомлення про скасовану транзакцію. Деталі, включно з часом скасування і причиною скасування.
- TransactionStatusRequest (Запит Статусу Транзакції). Запит на статус конкретної транзакції. Деталі, включно з ID транзакції та ідентифікацією запитувача.
- TransactionStatusResponse (Відповідь Статусу Транзакції). Надання статусу конкретної транзакції за запитом. Деталі статусу, включно з "Підтверджено", "Очікується", "Скасовано" тощо.
- KeyRotation (Оновлення Ключів). Оновлення ключів для учасників мережі. Деталі, включно з новими відкритими ключами і часом оновлення.
- ServerError (Помилка Сервера). Інформування про помилку на сервері. Деталі помилки, включно з кодом помилки та повідомленням про помилку.
- UserUpdate (Оновлення Користувача). Оновлення інформації користувача. Нові або оновлені дані користувача, такі як профіль або налаштування безпеки.
- SystemNotification (Системне Повідомлення). Системні повідомлення для користувачів або серверів. Інформація про важливі події або зміни в системі.
- KeyExchange (Обмін Ключами). Полегшує безпечний обмін ключами для ECDH між учасниками мережі. Деталі, такі як відкриті ключі, ідентифікація серверів і час обміну.
- KeyExchangeStatus (Статус Обміну Ключами). Повідомлення про статус спроби обміну ключами. Деталі, включно зі статусом обміну, успішно або невдало, і будь-які пов'язані тимчасові мітки або ідентифікатори.
- ClientRegistration (Реєстрація Клієнта). Реєстрація нового клієнта в мережі. Деталі, такі як ідентифікатор клієнта, відкритий ключ і дані профілю.
- ServerRegistration (Реєстрація Сервера). Реєстрація нового сервера в мережі.Деталі, такі як ідентифікатор сервера, відкритий ключ і дані профілю.

- UserLogin (Вхід Користувача). Вхід користувача в мережу. Деталі, такі як ідентифікатор користувача, хешований пароль та ідентифікатор сервера.
- ServerLogin (Вхід Сервера). Вхід сервера в мережу. Деталі, такі як ідентифікатор сервера і хеш відкритого ключа.
- Fee (Збір). Обробка комісій за транзакції в мережі. Деталі, такі як сума збору, одержувач збору і пов'язаний ID транзакції.

Докладніше структуру протоколу наведено в додатку А

Кожен тип оновлення може мати свій унікальний набір полів і вимог до обробки, роблячи систему гнучкою та адаптованою до різних сценаріїв використання

*Обмін повідомленнями*

Протокол ETAMP (Encrypted Token And Message Protocol) керує безпечною взаємодією між користувачами на різних серверах. Представлений вами документ забезпечує деталізоване пояснення робочого процесу обміну повідомленнями і транзакціями між двома користувачами, Алісою і Бобом, на різних серверах. Ось більш детальне роз'яснення цього робочого процесу:

*Початкове налаштування:*

Ідентифікація сервера:

- Аліса перебуває на сервері WebServer1 з індексом *B2F7714F-891F-4A4F-93F0-D472309DE102*.
- Боб перебуває на сервері WebServer2 з індексом *0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108*.

*Процес обміну ключами:*

Ініціація обміну ключами. Перш ніж Аліса і Боб зможуть безпечно спілкуватися, їм потрібно обмінятися відкритими ключами для встановлення безпечного каналу. Аліса ініціює оновлення KeyExchange, щоб поділитися своїм відкритим ключем із Бобом.

*{ "alg": "ES512",*

*"typ": "JWT" }*

*{*

*"jti": "550e8400-e29b-41d4-a716-446655440000",*

*"exp": 1679992314,*

*"nbf": 1679988714,*

*"messageId": "8D85BEAF-9468-49A8-A165-75AB42B01BEB",*

*"senderUserName": "Alice",*

*"senderId": "524B496E-BDE9-473B-A143-55A7F590C373",*

*"recipient": "Bob",*

*"recipientId": "2480756A-DCBB-498B-A92D-AF1093BA26D6",*

*"senderServerName": "WebServer1",*

*"senderServerId": "B2F7714F-891F-4A4F-93F0-D472309DE102",*

*"recipientServerName": "WebServer2",*

*"recipientServerId": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108",*

*"iss": "B2F7714F-891F-4A4F-93F0-D472309DE102.WebServer1.Alice.524B496E-BDE9-473B-A143-55A7F590C373",*

*"sub": "KeyExchange",*

*"audience": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108.WebServer2.Bob.2480756A-DCBB-498B-A92D-AF1093BA26D6",*

*"publicKey": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEyBvijI9MD6ZU1DZZ8...",*

*"timestamp": "2023-10-10T10:10:10Z"*

*}*


*Підтвердження обміну ключами:*

Після отримання оновлення KeyExchange, Боб підтверджує отримання і ділиться своїм відкритим ключем з Алісою через інше оновлення KeyExchange. Алісі надсилається повідомлення з підтвердженням успішного обміну ключами.


*{ "alg": "ES512",*

*"typ": "JWT" }*

*{*

*"jti": "550e8400-e29b-41d4-a716-446655440001",*

*"exp": 1679992315,*

*"nbf": 1679988715,*

*"messageId": "8D85BEAF-9468-49A8-A165-75AB42B01BEC",*
*"senderUserName": "Bob",*
*"senderId": "2480756A-DCBB-498B-A92D-AF1093BA26D6",*
*"recipient": "Alice",*
*"recipientId": "524B496E-BDE9-473B-A143-55A7F590C373",*
*"senderServerName": "WebServer2",*
*"senderServerId": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108",*
*"recipientServerName": "WebServer1",*
*"recipientServerId": "B2F7714F-891F-4A4F-93F0-D472309DE102",*
*"iss": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108.WebServer2.Bob.2480756A-DCBB-498B-A92D-AF1093BA26D6",*
*"sub": "KeyExchange",*
*"audience": "B2F7714F-891F-4A4F-93F0-D472309DE102.WebServer1.Alice.524B496E-BDE9-473B-A143-55A7F590C373",*
*"publicKey": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEyBvijI9MD6ZU2DZZ8...",*
*"timestamp": "2023-10-10T10:10:11Z"*
*}*

*Статус обміну:*

Після успішного обміну відкритими ключами обидві сторони відправляють оновлення KeyExchangeStatus для підтвердження успішного завершення процесу обміну ключами.

*{ "alg": "ES512",*
*"typ": "JWT" }*
*{*
*"jti": "550e8400-e29b-41d4-a716-446655440000",*
*"exp": 1679992314,*
*"nbf": 1679988714,*
*"messageId": "8D85BEAF-9468-49A8-A165-75AB42B01BEB",*
*"senderUserName": "Alice",*
*"senderId": "524B496E-BDE9-473B-A143-55A7F590C373",*
*"recipient": "Bob",*
*"recipientId": "2480756A-DCBB-498B-A92D-AF1093BA26D6",*

*"senderServerName": "WebServer1",*
*"senderServerId": "B2F7714F-891F-4A4F-93F0-D472309DE102",*
*"recipientServerName": "WebServer2",*
*"recipientServerId": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108",*
*"iss": "B2F7714F-891F-4A4F-93F0-D472309DE102.WebServer1.Alice.524B496E-BDE9-473B-A143-55A7F590C373",*
*"sub": "KeyExchangeStatus",*
*"audience": "0b0f5dcb-790c-4bdb-ac1a-6e77a1c02108.WebServer2.Bob.2480756A-DCBB-498B-A92D-AF1093BA26D6",*
*"status": "Successful",*
*"timestamp": "2023-10-10T10:10:10Z"*
*}*

*Безпечне спілкування:*

- Передача повідомлень.

Зі встановленим безпечним каналом Аліса і Боб тепер можуть безпечно обмінюватися повідомленнями або транзакціями. Повідомлення шифруються з використанням загальних секретів, отриманих з обміняних відкритих ключів, що забезпечує безпечне спілкування.

- Підтвердження повідомлень:

Після отримання повідомлення надсилається підтвердження для підтвердження успішного отримання повідомлення. Такий деталізований робочий процес гарантує врахування кожного кроку, що бере участь у безпечному спілкуванні між користувачами на різних серверах, роблячи протокол ETAMP надійним і надійним для різних сценаріїв використання

### 2.3 Безпека протоколу

Протокол ETAMP було розроблено з метою забезпечення високого рівня безпеки в мережевих взаємодіях між серверами та клієнтами. Ґрунтуючись на міцних криптографічних алгоритмах, як-от ECDSA і ECDH, він пропонує надійну платформу для обміну повідомленнями та транзакціями, забезпечуючи водночас збереження, достовірність і цілісність даних під час передання.

ETAMP знаходить своє застосування в різних сферах, включно з безпечним спілкуванням і фінансовими операціями, де безпека даних є критично важливою. Поміж інших протоколів, ETAMP виділяється акцентом на децентралізоване управління, обмін даними в реальному часі та забезпечення криптографічної безпеки.

*Роль Токенів*

Токени в протоколі ETAMP виконують найважливішу роль, виступаючи носіями інформації для обміну повідомленнями і транзакціями між учасниками мережі. Їх використання підкреслює прагнення до забезпечення безпеки та ефективності в мережевих взаємодіях:

- **Аутентифікація та Авторизація.** Токени містять у собі інформацію, яка допомагає ідентифікувати та автентифікувати користувача, а також визначити його права доступу. Це гарантує, що лише уповноважені особи мають доступ до певних ресурсів і можуть виконувати певні дії.
- **Безпека.** Обмеження часу дії токенів є ключовим моментом у стратегії забезпечення безпеки. Токени, що діють протягом обмеженого періоду часу (наприклад, однієї години), допомагають запобігти атакам повторного використання. У разі компрометації токена зловмисник не зможе використовувати його після закінчення встановленого часу.
- **Дотримання Принципів Безпеки.** Обмеження часу дії токенів також слідує принципу найменших привілеїв, обмежуючи можливості потенційного порушника і мінімізуючи ризик можливого збитку.
- **Зручність та Ефективність.** З використанням токенів забезпечується зручність і ефективність у процесах автентифікації та авторизації, оскільки не потрібно постійно надавати облікові дані; достатньо надати токен, який швидко й ефективно обробляється системою.

*Ротація Ключів*

Ротація ключів є важливою частиною стратегії безпеки в протоколі ETAMP. Кожен сервер у мережі несе відповідальність за проведення ротації своїх ключів та інформування інших серверів про нові публічні ключі. Ось як це працює:

Подія Ротації:

- Ініціація: Сервер створює нову пару ключів.
- Повідомлення: Генерується подія ротації ключів для оповіщення інших серверів про нові публічні ключі.

Обробка Повідомлень:

- Оновлення: Сервери оновлюють свої записи новими публічними ключами для забезпечення безпеки в майбутніх взаємодіях.

Безпечна Дистрибуція Ключів

- Канали: Гарантія безпечного каналу для поширення повідомлень про ротацію ключів і нові публічні ключі.

Верифікація

- Тестування: Після оновлення ключів сервери проводять перевірку, генеруючи, відправляючи і верифікуючи підпис тестового повідомлення, щоб упевнитися в коректності роботи нових ключів.

Журналювання та Моніторинг

- Аудит: Усі дії, пов'язані з ротацією ключів і спроби верифікації, фіксуються для подальшого аналізу.

*Верифікація після ротації ключів*

Процес верифікації після ротації ключів є критично важливим етапом, що забезпечує цілісність і безпеку мережевих взаємодій:

Підтвердження Інтеграції

- Після того як ключі було оновлено, необхідно упевнитися, що нові ключі інтегровані коректно і готові до використання. Це досягається шляхом

генерації, надсилання та підтвердження підпису тестового повідомлення між серверами.

Забезпечення Безперервності Безпеки

- Верифікація після ротації ключів гарантує, що оновлення ключів не призведе до втрати безпеки і що нові ключі забезпечують такий самий або вищий рівень захисту, ніж попередні.

Усунення можливих проблем

- Процес верифікації також слугує інструментом для виявлення та усунення можливих проблем або недоліків у процесі ротації ключів, гарантуючи, що всі етапи були виконані коректно.

Підтримка Довіри

- Успішна верифікація після ротації ключів сприяє підтримці та зміцненню довіри між учасниками мережі, оскільки вони можуть бути впевненими в надійності та безпеці криптографічних ключів, що використовуються.

Логування та Моніторинг

- Усі дії та результати верифікації фіксуються, що дає змогу проводити аудит і аналіз безпеки, забезпечуючи прозорість і звітність у процесах оновлення ключів.

Таким чином, токени та процес верифікації після ротації ключів відіграють центральну роль у забезпеченні безпеки, зручності використання та довіри в рамках протоколу ETAMP

### 2.5 Мережева топологія

Протокол ETAMP розроблено з урахуванням гнучкості в застосуванні до різних топологій мережі, проте його оптимальне функціонування досягається в умовах змішаної зірчастої  та комірчастої топології.

Змішана топологія поєднує в собі елементи зоряної та комірчастої структури, надаючи баланс між централізованим і децентралізованим управлінням. Сервери в такій мережі можуть спілкуватися один з одним

безпосередньо (комірчаста топологія), забезпечуючи відмовостійкість і розподіленість, водночас кожен сервер обслуговує певну кількість клієнтів (зоряна топологія), гарантуючи ефективність і простоту управління.

У межах цієї архітектури реалізується принцип розділеної децентралізації. Сервери можуть взаємодіяти між собою в децентралізованій манері, що збільшує стійкість мережі до відмов і атак.

Взаємодія Клієнтів і Серверів наступна: кожен клієнт у мережі прив'язаний до певного сервера (1 акаунт - 1 сервер), і вся комунікація клієнта проходить через цей сервер.

Сервер виступає в ролі посередника і захисника інтересів клієнта, забезпечуючи безпеку і конфіденційність обміну даними. Клієнти можуть спілкуватися тільки зі своїм сервером, що спрощує процес аутентифікації та авторизації.

Коли клієнту необхідно надіслати повідомлення або транзакцію іншому клієнту, який обслуговується іншим сервером, процес виглядає наступним чином:

- Створення Запиту - клієнт формує запит згідно з протоколом ETAMP і відправляє його на свій сервер.
- Перевірка та Підпис сервером - сервер перевіряє повідомлення, токен і, якщо все гаразд, підписує токен і все повідомлення, реалізуючи тим самим механізм мультипідпису. Це забезпечує додатковий рівень захисту і дає змогу серверу одержувача перевірити повідомлення і токен на предмет їхньої цілісності та незмінності.
- Отримання та перевірка повідомлення Одержувачем - після отримання повідомлення клієнт через свій сервер може запросити публічний ключ відправника для перевірки підпису токена і упевнитися в його цілісності та автентичності.

Така архітектура дає змогу поєднувати переваги децентралізованих і централізованих мереж, забезпечуючи гнучкість, масштабованість і безпеку в обміні даними. Мережа отримує переваги децентралізованого управління завдяки можливості прямого спілкування серверів, у той час як клієнти користуються простотою і безпекою взаємодії в рамках моделі "один клієнт - один сервер".

Використання протоколу ETAMP і механізму мультипідпису гарантує, що всі повідомлення і транзакції залишаються конфіденційними, цілісними і захищеними від несанкціонованих змін упродовж усього процесу передачі. Таким чином, протокол ETAMP у поєднанні зі змішаною зірковою та комірчастою топологією мережі забезпечує потужну та гнучку платформу для безпечного й ефективного обміну даними в сучасних мережевих середовищах

**2.6 Технологія шардінга**

Шард (від англ. "shard", що буквально перекладається як "осколок" або "частина") у контексті баз даних і блокчейн-технологій означає фрагментацію даних на більш дрібні частини для поліпшення продуктивності та масштабованості системи.

У традиційних базах даних шардинг використовується для поділу великої бази даних на більш дрібні, більш керовані частини, кожна з яких може бути розміщена на окремому сервері. Це дає змогу розподілити навантаження, збільшити швидкість доступу до даних і забезпечити ефективніше використання ресурсів.

- Горизонтальний шардинг: Дані поділяються за рядками. Наприклад, записи з 1 по 1000 можуть перебувати на одному шарді, а записи з 1001 по 2000 - на іншому.
- Вертикальний шардинг: Дані поділяються за стовпчиками. Наприклад, один шард може містити тільки дані користувачів, а інший - тільки транзакції.

У блокчейні шардинг використовується для поділу всієї мережі на більш дрібні сегменти (шарди), кожен з яких може обробляти транзакції і зберігати дані незалежно від інших. Це спрямовано на збільшення масштабованості та продуктивності мережі.

- Паралельне Оброблення: Кожен шард може обробляти транзакції паралельно з іншими, що значно збільшує загальну пропускну здатність мережі.

Шарди можуть функціонувати незалежно, що зменшує навантаження на кожен окремий вузол і забезпечує більш ефективне використання ресурсів

Архітектура мережі ETAMP, виконана за принципом mesh-зірка, являє собою інноваційне рішення, де кожен сервер у мережі функціонує як окремий шард для обслуговування своїх клієнтів. Ця структура дає змогу досягти видатної масштабованості та ефективності, одночасно забезпечуючи надійність і безпеку обміну даними.

Горизонтальне розширення системи досягається за рахунок додавання нових серверів у мережу, що дає змогу збільшувати пропускну здатність і можливості всієї системи без необхідності посилення окремих серверів. Розподіл клієнтів між різними серверами сприяє балансуванню навантаження, що, своєю чергою, позитивно позначається на продуктивності мережі загалом.

Ізоляція збоїв на рівні окремих серверів підвищує надійність всієї системи, оскільки проблеми на одному сервері не впливають на працездатність решти мережі. Можливості резервування і відновлення даних на різних серверах також сприяють підвищенню відмовостійкості системи.

Важливою особливістю цієї архітектури є забезпечення високого рівня безпеки. Локалізація даних клієнтів на їхніх "домашніх" серверах знижує ризик несанкціонованого доступу та витоків інформації, а можливість самостійного управління доступом до ресурсів на кожному сервері додає додатковий шар захисту.

Інтеграція шардингу на рівні серверів у мережу ETAMP забезпечує прозорість взаємодії для кінцевих користувачів, зберігаючи водночас високу продуктивність і ефективність обробки транзакцій. Це рішення дозволяє мережі ETAMP масштабуватися, задовольняючи зростаючі потреби користувачів у швидкому та безпечному обміні даними.

Таким чином, впровадження шардингу на рівні серверів в архітектурі ETAMP є ключовим елементом для забезпечення масштабованості, надійності та безпеки мережі, що робить її привабливою для широкого кола користувачів і додатків

### 2.7 Захист від атак

Протокол ETAMP, що базується на використанні криптографії еліптичних кривих, гарантує високий рівень стійкості до різних видів атак.

1. Атаки на Алгоритми Шифрування/Підпису. ETAMP використовує алгоритми ECDSA та ECDH, які є відносно надійними та стійкими до атак завдяки тому, що вони засновані на складності задачі дискретного логарифмування на еліптичній кривій. Це завдання вважається обчислювально важким, що робить злом ключів або підробку підписів вкрай складними.
2. Атаки типу "Відмова в обслуговуванні". Розподілена мережева топологія ETAMP, в якій кожен вузол є незалежним і має свої резервні канали зв'язку, забезпечує високу доступність сервісу і стійкість до атак типу "відмова в обслуговуванні".
3. Атаки на консенсус. ETAMP не використовує традиційні механізми блокчейн-консенсусу, що робить його несприйнятливим до атак на консенсус.
4. Злам закритих ключів Використання надійних криптографічних алгоритмів і довгих ключів робить злом закритих ключів вкрай складним, що забезпечує захист конфіденційності та цілісності даних.

5. Підміна вузлів мережі. Застосування цифрових сертифікатів і підписів даних дає змогу ідентифікувати та верифікувати вузли мережі, запобігаючи їхній підміні зловмисниками.
6. Аналіз і маніпуляція трафіком. Шифрування трафіку між вузлами запобігає можливості аналізу та маніпуляції переданими даними.
7. Компрометація даних. Технології захисту даних під час зберігання і передавання забезпечують збереження і конфіденційність інформації, запобігаючи її компрометації.
8. Атаки типу "Людина Посередині". Криптографічні механізми ETAMP ефективно захищають від атак типу "людина посередині", забезпечуючи цілісність і конфіденційність переданих даних.

Загалом, ETAMP демонструє високий рівень стійкості до різних видів атак, забезпечуючи надійне та безпечне середовище для взаємодії вузлів і передавання даних.

**2.8 Сфери використання.**

Протокол ETAMP є доволі потужним інструментом для реалізації безпечної та ефективної взаємодії в різних сферах, від фінансових операцій до управління даними в інтернеті речей.

- Фінансові транзакції та платежі. ETAMP забезпечує швидкий і безпечний переказ коштів між користувачами та організаціями. Завдяки використанню криптографії еліптичних кривих, транзакції захищені від шахрайства і злому. Це робить протокол ідеальним для виконання мікроплатежів, міжнародних переказів та інших фінансових операцій, що вимагають високого рівня безпеки.
- Безпечний обмін конфіденційними даними. Протокол може бути використаний для передачі важливої інформації між компаніями та урядовими установами. Шифрування даних на рівні протоколу забезпечує конфіденційність інформації, а цифрові підписи гарантують її справжність.

- Посвідчення автентичності та Авторизація. ETAMP дає змогу випускати та перевіряти цифрові сертифікати на базі технології блокчейн, що забезпечує надійну автентифікацію та авторизацію користувачів і пристроїв у мережі.
- Управління ланцюжками поставок. Протокол може використовуватися для відстеження походження і переміщення товарів між учасниками ланцюжка поставок. Це підвищує прозорість і дає змогу виявляти підробки та порушення в ланцюжку поставок.
- Безпечний Інтернет речей. ETAMP забезпечує безпечний обмін даними між "розумними" пристроями і машинами, що критично важливо для реалізації надійних і безпечних рішень у сфері інтернету речей.
- Децентралізовані додатки. Протокол можна використовувати для побудови бекенда для децентралізованих додатків, забезпечуючи надійність, безпеку та відсутність єдиної точки відмови.
- Зберігання медичних даних. ETAMP дає змогу безпечно зберігати й обмінюватися медичною інформацією, забезпечуючи конфіденційність пацієнтських даних і відповідність нормативним вимогам.
- Ідентифікація користувачів. Система може використовуватися для випуску унікальних ідентифікаторів на базі блокчейна, що забезпечує надійну ідентифікацію користувачів у мережі.
- Цифрові голосування. ETAMP може бути застосований для проведення безпечних виборів і опитувань серед зареєстрованих користувачів, гарантуючи чесність і прозорість голосування.

Усі ці сценарії демонструють широкий спектр можливостей застосування протоколу ETAMP, підкреслюючи його універсальність і високий рівень безпеки

## Висновки до другого розділу

Таким чином, у другому розділі проведено всебічний аналіз ETAMP, його компонентів, механізмів і можливостей. Були детально описані основні компоненти і механізми протоколу ETAMP, включаючи структуру даних, алгоритми шифрування і підписання повідомлень, а також спосіб взаємодії сервера і клієнта один з одним.

Протокол ETAMP використовує криптографію на еліптичних кривих для забезпечення високого рівня безпеки, при цьому забезпечуючи швидку обробку даних і знижуючи навантаження на систему. Цей протокол може бути ефективно використаний у великих масштабах, відповідаючи сучасним вимогам щодо швидкості обробки даних і безпеки.

Аналізуючи технічну основу ETAMP, було розглянуто алгоритми ECDSA та ECDH, їх роль в процесі шифрування і підпису повідомлень. Ми розглянули структуру даних на основі JWT і формат ключів PEM, забезпечуючи чітке розуміння того, як працюють ключові компоненти ETAMP.

В розділі детально описані компоненти і механізми ETAMP, включаючи формат повідомлень, структуру токенів і алгоритм підпису повідомлень. Це дозволяє розробникам та інженерам більш чітко розуміти, як вони можуть інтегрувати і використовувати ETAMP у своїх системах. Як висновок можна сказати що цей протокол не тільки відповідає сучасним вимогам безпеки та ефективності, але й надає широкий спектр можливостей для розвитку і інтеграції в різні технологічні екосистеми.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Протокол ETAMP являє собою гнучке рішення, призначене для забезпечення безпечного обміну даними в різних мережевих середовищах. Його можна адаптувати і реалізувати на різних мовах програмування і операційних системах, що робить його універсальним інструментом для захисту мережевих взаємодій.

## 3.1. Можливість реалізації протоколу різними мовами програмування

**C#**: Мова програмування C# від Microsoft має потужні інструменти для розробки як десктопних, так і веб-додатків. Завдяки вбудованій підтримці криптографічних бібліотек і мережевих протоколів, C# є чудовим вибором для створення надійних і безпечних рішень на базі ETAMP. Розробники можуть використовувати платформу .NET для створення крос-платформних додатків, що працюють на Windows, Linux і Mac OS.

**JavaScript (JS)**: JavaScript широко використовується для розробки веб-додатків і може бути застосований для реалізації клієнтської сторони ETAMP. З появою Node.js, JavaScript також став популярним вибором для створення серверних додатків. Це дає змогу використовувати одну й ту саму мову програмування для розробки як клієнтської, так і серверної частин застосунку, спрощуючи процес розробки.

**Python**: Python відомий своєю простотою і читабельністю коду, що прискорює процес розроблення і полегшує підтримку коду. Багатий набір бібліотек для роботи з криптографією і мережевими протоколами робить Python хорошим варіантом для реалізації ETAMP. До того ж, Python крос-платформний, що забезпечує його працездатність на різних операційних системах.

*Підтримка різних операційних систем*

1.Windows: ETAMP може бути реалізовано як у вигляді застосунків для кінцевих користувачів, так і у вигляді серверних рішень на платформі Windows,

забезпечуючи широке охоплення аудиторії та інтеграцію з наявною інфраструктурою.

2.Linux: Ця операційна система широко використовується в серверних середовищах завдяки своїй стабільності та безпеці. Реалізація ETAMP на Linux дасть змогу скористатися всіма перевагами цієї платформи, включно з можливістю роботи в кластерах і хмарах.

3. Mac OS: Підтримка Mac OS забезпечує доступність ETAMP для користувачів продукції Apple, включно з персональними комп'ютерами та серверними рішеннями

Для реалізації протоколу ETAMP на C# потрібно встановити такі бібліотеки:

**System.IdentityModel.Tokens.Jwt**: Для роботи з токенами JWT.

**System.Security.Cryptography**: Для роботи з генерацією та перевіркою підписів ECDSA.

```
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;
using Microsoft.IdentityModel.Tokens;
public class Program
{
    public static void Main()
    {
        // Key generation
        using ECDsa ecdsa = ECDsa.Create(ECCurve.NamedCurves.nistP521);
        var privateKey = new ECDsaSecurityKey(ecdsa);
        // Generate JWT token for a message
       var id = Guid.NewGuid().ToString();
        var token = CreateToken(id, privateKey);
        // ETAMP structure for a message
```

```
        var message_etamp = new
        {
            Id = id,
            Token = token,
            SignatureToken = SignMessage(token, ecdsa),
            SignatureMessage = SignMessage(id + token + SignMessage(token, ecdsa),
ecdsa)
        };
        Console.WriteLine(message_etamp);
    }
    public static string CreateToken(string id, SecurityKey key)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        // Generate timestamps for the JWT token
        var currentDateTime = DateTime.UtcNow;
        var expiryDateTime = currentDateTime.AddHours(1);
        var notBeforeDateTime = currentDateTime;
        var senderId = Guid.NewGuid();
        var recipientId = Guid.NewGuid();
        var serverId = Guid.NewGuid();
        var tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(new[]
            {
                new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
                new Claim("messageId", id),
                new Claim("senderUserName", "user1"),
                new Claim("senderId", senderId.ToString()),
                new Claim("recipient", "user2"),
                new Claim("recipientId", recipientId.ToString()),
                new Claim("senderServerName", "WebServer1"),
                new Claim("senderServerId", serverId.ToString()),
                new Claim("recipientServerName", "WebServer1"),
```

```
            new Claim("recipientServerId", serverId.ToString()),
            new Claim("iss", $"{serverId}.WebServer1.user1.{senderId}"),
            new Claim("sub", "Message"),
            new Claim("audience", $"{serverId}.WebServer1.user2.{recipientId}"),
            new Claim("message", "ereb5454bwehqwy-3hgerh34ebd="),
            new Claim("timestamp", DateTime.Now.ToUniversalTime().ToString())
        }),
        Expires = expiryDateTime,
        NotBefore = notBeforeDateTime,
        SigningCredentials = new SigningCredentials(key,
SecurityAlgorithms.EcdsaSha512Signature)
    };
      var token = tokenHandler.CreateToken(tokenDescriptor);
      return tokenHandler.WriteToken(token);
  }
  public static string SignMessage(string message, ECDsa ecdsa)
  {
      var data = Encoding.UTF8.GetBytes(message);
      var signature = ecdsa.SignData(data, HashAlgorithmName.SHA512);
      return Convert.ToBase64String(signature);
  }
}
```

Для реалізації протоколу ETAMP на JavaScript (Node.js) потрібно встановити такі бібліотеки:

- jsonwebtoken: Для опрацювання токенів JWT.
- elliptic: Для роботи з генеруванням і перевіркою підписів ECDSA.
- uuid: Для генерування унікальних ідентифікаторів.

Це можна зробити, виконавши таку команду в командному рядку:

```
npm install jsonwebtoken elliptic uuid
const jwt = require('jsonwebtoken');
onst ECDSA = require('elliptic').ec;
```

```
const ec = new ECDSA('p521');
const uuid = require('uuid');
const { generateKeyPairSync } = require('crypto');
// Key generation
//const keyPair = ec.genKeyPair();
const { publicKey, privateKey } = generateKeyPairSync('ec', {
    namedCurve: 'secp521r1', // This is equivalent to the p521 curve you were using
    publicKeyEncoding: {
        type: 'spki',
        format: 'pem'
    },
    privateKeyEncoding: {
        type: 'pkcs8',
        format: 'pem'
    }
});
function signMessage(message, privateKey) {
    const signature = ec.sign(message, privateKey, 'hex');
    return Buffer.from(signature.toDER()).toString('base64');
}
// Generate JWT token for a message
const id = uuid.v4();
const jti = uuid.v4();
const messageId = uuid.v4();
const senderId = uuid.v4();
const recipientId = uuid.v4();
const senderServerId = uuid.v4();
const payload = {
    jti: jti,
    exp: 1679992314,
    nbf: 1679988714,
    messageId: messageId,
    senderUserName: "user1",
```

```
    senderId: senderId,
    recipient: "user2",
    recipientId: recipientId,
    senderServerName: "WebServer1",
    senderServerId: senderServerId,
    recipientServerName: "WebServer1",
    recipientServerId: senderServerId,
// Також використаєм senderServerId для recipientServerId
    iss: `${senderServerId}.WebServer1.user1.${senderId}`,
    sub: "Message",
    audience: `${senderServerId}.WebServer1.user2.${recipientId}`,
    message: "ereb5454bwehqwy-3hgerh34ebd=",
    timestamp: new Date().toISOString()
};
const token = jwt.sign(payload, privateKey, {algorithm: 'ES512'});
// ETAMP structure for a message
const message_etamp = {
    Id: id,
    Token: token,
    SignatureToken: signMessage(token, privateKey),
    SignatureMessage: signMessage(id + token + signMessage(token, privateKey),
rivateKey)
};
console.log(JSON.stringify(message_etamp, null, 4));
```

Для мови Python необхідні бібліотеки:

- PyJWT: бібліотека Python для кодування та декодування токенів JWT.
- ecdsa: Бібліотека Python для підписання та перевірки повідомлень за допомогою ECDSA.
- uuid: Бібліотека Python для генерації унікальних ідентифікаторів.
- cryptography: бібліотека Python, що надає криптографічні рецепти та примітиви.

*pip install PyJWT ecdsa cryptography*

```
import jwt
import ecdsa
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
import uuid
import datetime
import base64
# Key generation for JWT
private_key_jwt = ecdsa.SigningKey.generate(curve=ecdsa.NIST521p)
public_key_jwt = private_key_jwt.verifying_key
# Key generation for ECDH
private_key_ecdh = ec.generate_private_key(ec.SECP256R1(), default_backend())
peer_public_key = ec.generate_private_key(ec.SECP256R1(),
default_backend()).public_key()
def private_key_to_pem(sk):
    # Convert the SigningKey to a PEM formatted string
    return sk.to_pem().decode()

def sign_message(message, private_key):
    signature = private_key.sign(message.encode())
    return base64.b64encode(signature).decode()
def encrypt_message_ecdh(message):
    shared_key = private_key_ecdh.exchange(ec.ECDH(), peer_public_key)
    return shared_key[:len(message)].hex()  # Return as hex for simplicity
def generate_etamp():
    header = {
        "alg": "ES512",
        "typ": "JWT"
    }
```

```
current_time = datetime.datetime.now(datetime.timezone.utc)
expiration_time = current_time + datetime.timedelta(seconds=3600)  # 1 hour later
not_before_time = current_time
senderServerId = str(uuid.uuid4())
senderId = str(uuid.uuid4())
recipientId = str(uuid.uuid4())
message_id = str(uuid.uuid4())
payload = {
    "jti": str(uuid.uuid4()),
    "exp": int(expiration_time.timestamp()),
    "nbf": int(not_before_time.timestamp()),
    "messageId": message_id,
    "senderUserName": "user1",
    "senderId": senderId,
    "recipient": "user2",
    "recipientId": recipientId,
    "senderServerName": "WebServer1",
    "senderServerId": senderServerId,
    "recipientServerName": "WebServer1",
    "recipientServerId": senderServerId,
    "iss": f"{senderServerId}.WebServer1.user1.{senderId}",
    "sub": "Message",
    "audience": f"{senderServerId}.WebServer1.user2.{recipientId}",
    "message": encrypt_message_ecdh("This is a secret message"),
    "timestamp": current_time.strftime('%Y-%m-%dT%H:%M:%SZ')
}
pem_private_key = private_key_to_pem(private_key_jwt)
token = jwt.encode(payload, pem_private_key, algorithm="ES512", headers=header)
# ETAMP structure for a message
message_etamp = {
    "Id": message_id,
    "Token": token,
    "SignatureToken": sign_message(token, private_key_jwt),
```

```
        "SignatureMessage": sign_message(message_id + token + sign_message(token,
private_key_jwt), private_key_jwt)
    }
    return message_etamp
if __name__ == "__main__":
    print(generate_etamp())
```

Гнучкість і універсальність протоколу ETAMP дають змогу його реалізовувати різними мовами програмування на різних операційних системах, що забезпечує широку доступність і застосовність у різноманітних сценаріях використання. Це робить ETAMP вельми привабливим рішенням для організацій і розробників, які прагнуть забезпечити високий рівень безпеки та ефективності у своїх мережевих взаємодіях.

Для детального вивчення роботи протоколу ETAMP використовується широкий спектр програмних засобів і тестових конфігурацій, вимірюються швидкості передавання даних, кількість операцій на секунду та інші ключові показники продуктивності. Також розглядаються фактори, які можуть вплинути на продуктивність протоколу, такі як характеристики обладнання, стан мережі та конфігурація самого протоколу. Це дає змогу зрозуміти, як взаємодіють різні чинники і як вони впливають на загальну продуктивність системи. Докладніше окремі частини програмного коду наведено в ***Додатку А***.

Для об'єктивної оцінки проведемо порівняльний аналіз з іншими наявними блокчейн-рішеннями з акцентом на сильні та слабкі сторони. Також буде розглянуто різні методи і стратегії, які можна застосувати для підвищення ефективності ETAMP, і визначено найперспективніші з них для подальшого впровадження.

### 3.2. Оцінка продуктивності протоколу ETAMP

Важливою частиною оцінки ефективності криптографічного протоколу є аналіз його продуктивності на різних апаратних засобах. Протокол ETAMP,

розроблений з використанням алгоритму еліптичної кривої, добре працює на різних платформах.

*Продуктивність на процесорах Intel Core*

Дані про швидкість обробки повідомлень на різних моделях процесорів Intel Core наочно демонструють масштабованість ETAMP:

- **Intel Core i3**: швидкість опрацювання становить від 50 до 80 повідомлень на секунду залежно від моделі, що цілком достатньо для обладнання початкового рівня.
- **Intel Core i5**: продуктивність значно зростає і досягає 140 повідомлень на секунду в моделі i5-14900hk.
- **Intel Core i7**: ETAMP використовує приріст продуктивності цих процесорів, досягаючи 210 повідомлень на секунду.
- **Intel Core i9**: найвища продуктивність серед усіх процесорів Intel Core, максимальна швидкість опрацювання 250 повідомлень на секунду на моделі i9-14900hk.

Ці результати показують, що ETAMP може масштабуватися зі зростанням обчислювальної потужності процесора і досягати високої продуктивності навіть на відносно старому або менш продуктивному обладнанні.

*Продуктивність на відеокартах Nvidia*

Дослідження продуктивності ETAMP на відеокартах Nvidia GTX і RTX показали ще більш вражаючі результати:

- **Nvidia GTX 1060 - RTX 2080**: швидкість обробки збільшилася з 300 до 1 000 повідомлень на секунду, що підкреслює переваги використання графічних процесорів для прискорення обробки.
- **Nvidia RTX 3060 - RTX 3080**: швидкість обробки від 1 000 до 2 000 повідомлень за секунду, що в десятки разів перевищує продуктивність на базі CPU.

- **Nvidia RTX 4060 - RTX 4080**: на цих картах ETAMP досягає швидкості від 1 500 до 3 000 повідомлень на секунду, встановлюючи нові стандарти продуктивності для блокчейн-протоколів.

*Порівняння з традиційними блокчейн-системами*

Порівняння результатів із традиційними блокчейн-системами, такими як Bitcoin і Ethereum, показує, що продуктивність ETAMP значно вища. У той час як Bitcoin обробляє близько семи транзакцій за секунду, а Ethereum - близько 30 транзакцій за секунду, ETAMP здатний обробляти тисячі повідомлень за секунду на сучасному обладнанні.

Протокол ETAMP демонструє відмінну продуктивність на різних платформах, що робить його гарним вибором для високонавантажених систем і додатків. Масштабованість зі зростанням пропускної спроможності обладнання та чудова продуктивність на відеокартах підкреслюють його потенціал для використання в широкому спектрі застосунків, які потребують швидкого та безпечного опрацювання транзакцій.

*Порівняння швидкості транзакцій ETAMP з традиційними блокчейн-мережами*

*Р*озглянемо і порівняємо швидкість транзакцій протоколу ETAMP з традиційними блокчейн-мережами, такими як Bitcoin, Ethereum, Ripple, Stellar, EOS, і Cardano

Мережа Bitcoin, що є представником першого покоління блокчейн, має обмежену пропускну спроможність, обробляючи від 3 до 7 транзакцій на секунду. Це пов'язано з обмеженнями розміру блоку та інтервалами між блоками. ETAMP, використовуючи передові криптографічні алгоритми та ефективну мережеву структуру, демонструє в десятки разів вищу швидкість обробки транзакцій навіть на базовому обладнанні

Ethereum, що надає можливості розумних контрактів, також стикається з обмеженнями у швидкості транзакцій, досягаючи 10-15 транзакцій на секунду.

ETAMP перевершує Ethereum у 5-8 разів на базових конфігураціях і показує ще більш вражаючі результати на потужному обладнанні

Ripple і Stellar були розроблені з урахуванням масштабованості та забезпечують швидкість транзакцій до 1,500 і 1,000 на секунду відповідно. Проте ETAMP все одно залишається на крок попереду, пропонуючи ще вищу продуктивність.

EOS і Cardano являють собою блокчейни нового покоління, що забезпечують високу швидкість і масштабованість. Вони демонструють швидкість до 4,000 і 250 транзакцій на секунду відповідно. ETAMP пропонує порівнянну швидкість на базовому рівні і значно перевершує на високопродуктивному обладнанні.

ETAMP демонструє видатну масштабованість завдяки своїй інноваційній архітектурі та використанню шардингу. Він здатний обробляти тисячі транзакцій на секунду, зберігаючи водночас високий ступінь безпеки та надійності. Це робить ETAMP ідеальним вибором для великомасштабних і високонавантажених систем, де потрібно забезпечити швидку і стабільну обробку транзакцій. У наступних розділах ми продовжимо аналіз, розглянувши інші важливі аспекти роботи протоколу ETAMP

*Порівняння безпекових показників з традиційними Блокчейн-мережами*

Хоча Bitcoin і Ethereum вважаються відносно безпечними, вони стикаються з обмеженнями у своїй здатності масштабуватися, що може впливати на безпеку при високих навантаженнях. Крім того, їхні консенсусні механізми (доказ виконаної роботи в Bitcoin і Ethereum) вимагають значних обчислювальних ресурсів, що може призвести до централізації майнінгової потужності та потенційних загроз безпеці.

Мережі Ripple, Stellar, EOS і Cardano пропонують поліпшені механізми консенсусу і масштабованості, що також сприяє підвищенню безпеки. Однак навіть вони не можуть зрівнятися з ETAMP в плані ефективності та швидкості

обробки транзакцій за умови забезпечення високого рівня безпеки. ETAMP пропонує видатний рівень безпеки завдяки використанню передових криптографічних методів, стійкості до різних типів атак і ефективній архітектурі. Це робить його ідеальним вибором для додатків, що вимагають високої безпеки та продуктивності.

Протокол ETAMP пропонує унікальний баланс між децентралізацією, продуктивністю і безпекою, роблячи його ідеальним вибором для застосунків, що вимагають високої доступності та стійкості до атак. У наступних розділах ми продовжимо аналіз інших ключових аспектів роботи ETAMP, детально розглядаючи його застосування і потенціал для майбутнього розвитку.

Біткойн, що використовує алгоритм консенсусу Proof-of-Work (PoW), відомий своїм високим енергоспоживанням. Майнінг у мережі Біткойн вимагає великої кількості електроенергії, що ставить під загрозу його стійкість та екологічну безпеку. ETAMP не використовує модель (PoW), яка вимагає значних обчислювальних ресурсів та енергії. Натомість, протокол спирається на більш ефективні криптографічні методи, що дає змогу суттєво знизити енергоспоживання.

### 3.3 Недоліки та потенційні проблеми протоколу ETAMP

При тому що протокол ETAMP пропонує ряд важливих переваг, але також має свої недоліки та потенційні ризики. Перерахуємо їх:

1. Відносно нова технологія. ETAMP є відносно новим протоколом, і його довгострокова стабільність і надійність ще належить перевірити. На відміну від старіших і усталених блокчейнів, таких як Біткойн і Ефіріум, ETAMP не пройшов тривалий період часу на ринку, що могло б підтвердити його стійкість до різних загроз і нападів.
2. Хоча ETAMP демонструє високу швидкість транзакцій на поточному етапі, питання масштабованості залишаються актуальними. Зі зростанням

кількості користувачів і транзакцій може виникнути необхідність у додаткових механізмах масштабування.

3. ETAMP, як будь-яка нова технологія, стикається з викликами щодо інтеграції з наявними системами та сервісами.

4. Незважаючи на високий рівень безпеки, що надається протоколом ETAMP, жодна технологія не може гарантувати 100% захист від усіх видів атак. Існує потенційний ризик вразливостей, який може бути виявлений та експлуатований зловмисниками.

5. ETAMP сильно залежить від алгоритмів криптографії на еліптичних кривих. Хоча ці алгоритми вважаються надійними і безпечними на поточний момент, розвиток квантових комп'ютерів у майбутньому може становити загрозу для цього типу криптографії.

6. ETAMP являє собою унікальний підхід до реалізації блокчейн-подібних технологій, істотно відрізняючись від традиційних блокчейнів за низкою ключових аспектів. Це призводить до того, що протокол ETAMP не сумісний з більшістю наявних блокчейн-систем

## Висновки до третього розділу

У третьому розділі магістерської роботи було наведено приклади того, як ETAMP може бути реалізований на різних мовах програмування, таких як C#, JavaScript та Python, і на різних платформах, включаючи Windows, Linux і Mac OS. Це підкреслює універсальність ETAMP і його здатність адаптуватися до різних технологічних середовищ. Також проаналізувано переваги використання мережевої топології "зірка-сітка" і роль шардингу в масштабуванні системи, показуючи, як це дозволяє ETAMP ефективно рости і адаптуватися до збільшення кількості користувачів і транзакцій.

ETAMP демонструє видатну продуктивність і ефективність, випереджаючи багато існуючих блокчейн-рішень у плані швидкості транзакцій і обробки даних. Це робить протокол особливо привабливим для використання у високонавантажених системах і додатках, де вимоги до швидкості та масштабованості особливо великі.

Гібридна архітектура мережі та унікальні криптографічні механізми ETAMP забезпечують не тільки високу продуктивність, а й відмінну масштабованість, що дає змогу системі ефективно справлятися зі зростаючими обсягами транзакцій і користувачами. При цьому протокол не втрачає в безпеці, надаючи надійні засоби захисту даних і транзакцій.

ETAMP також виділяється на тлі багатьох інших блокчейн-рішень своєю енергоефективністю, що є важливим фактором в умовах зростаючих вимог до екологічності та стійкості технологій. Гібридна структура мережі забезпечує оптимальне поєднання децентралізації та продуктивності, дозволяючи використовувати переваги обох підходів. Ґрунтуючись на проведених дослідженнях і порівняннях, можна зробити низку важливих висновків і визначити перспективи подальшого розвитку та оптимізації системи.

## Висновок

У рамках цієї дипломної роботи було представлено глибокий аналіз і розробку інноваційного протоколу ETAMP, мета якого - кардинальне поліпшення продуктивності та масштабованості блокчейн-систем. Це досягається за рахунок подолання обмежень, які стають дедалі очевиднішими в популярних платформах, таких як Bitcoin і Ethereum.

Розробка ETAMP почалася з критичного осмислення наявних проблем у сфері блокчейн-технологій, включно з повільними швидкостями транзакцій, проблемами масштабованості та енергоефективності. Відповіддю на ці проблеми стало створення нової архітектури, яка об'єднує в собі сучасні криптографічні методи та вдосконалені алгоритми досягнення консенсусу, спрямовані на оптимізацію роботи мережі.

У роботі було досліджено та детально описано технічні аспекти ETAMP, включно з його структурами даних, математичними принципами, що лежать в основі шифрування та безпеки, а також механізмами мережевої взаємодії, які забезпечують гладку та надійну роботу протоколу. Для демонстрації універсальності протоколу ETAMP було представлено приклади його реалізації в різних програмних середовищах, що підкреслює його адаптивність і сумісність із різноманітними технологічними стеками.

Експериментальна частина дослідження заслуговує на особливу увагу, оскільки вона являє собою центральну ланку підтвердження теоретично розроблених переваг ETAMP. Було показано, що в лабораторних умовах протокол забезпечує значно вищу швидкість транзакцій і можливості масштабування порівняно з традиційними блокчейнами. Ці результати підтверджують потенціал ETAMP як основи для створення нового покоління розподілених систем.

Важливою частиною роботи є аналіз досягнутих поліпшень в енергоефективності та безпеці, що робить ETAMP придатним для широкого спектра застосувань. Так, у фінансовій сфері, де важлива швидкість транзакцій і надійність, у логістиці, де потрібне відстеження великої кількості даних у реальному часі, у медицині, де конфіденційність і недоторканність інформації

мають критичне значення, та у державному управлінні, де потрібна прозорість і підзвітність, - у всіх цих галузях ETAMP може стати основою для більш ефективних і сучасних систем.

Насамкінець, результати цієї дипломної роботи роблять вагомий внесок у наукове співтовариство в галузі блокчейн-технологій і пропонують конкретні шляхи для практичного застосування розробленого протоколу, що відкриває двері для наступного покоління високопродуктивних і надійних розподілених систем зберігання даних.

# СПИСОК ЛІТЕРАТУРИ


1. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
2. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.
3. Schwartz, D., Youngs, N., & Britto, A. (2014). The Ripple protocol consensus algorithm. Ripple Labs Inc White Paper, 5.
4. Mazieres, D. (2015). The stellar consensus protocol: A federated model for Internet-level consensus. Stellar Development Foundation.
5. Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017). A taxonomy of blockchain-based systems for architecture design. In 2017 IEEE International Conference on Software Architecture (ICSA) (pp. 243-252). IEEE.
6. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017, June). An overview of blockchain technology: Architecture, consensus, and future trends. In 2017 IEEE international congress on big data (BigData congress) (pp. 557-564). IEEE.
7. Плескач В.Л., Затонацька Т.Г. Блокчейн: засади, технологія, перспективи. Економічний часопис-XXI. 2018. No 175. С. 61-65.
8. Марущак С.М. Блокчейн як інноваційна технологія цифрової економіки. Бізнес Інформ. 2019. No5. С. 124–130.
9. Кічор В.П. Блокчейн та криптовалюти: технології, перспективи і ризики використання. Наукові записки Національного університету «Острозька академія». Серія «Економіка» : науковий журнал. Острог, 2019. No 15(43). С. 22-25.
10. Савельєв Є.В. Технологія блокчейн: передумови появи та проблематика розвитку. Глобальні та національні проблеми економіки. 2018. No 22. С. 659-664.
11. Приймак В.М. Блокчейн: сутність та сфери застосування в цифровій економіці. Економіка України. 2018. No 4. С. 64-76.

12. Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. Applied Innovation, 2(6-10), 71.
13. Ølnes, S., Ubacht, J., & Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. Government Information Quarterly, 34(3), 355-364.
14. Casino, F., Dasaklis, T. K., & Patsakis, C. (2018). A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telematics and Informatics.
15. Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. Journal of the American Medical Informatics Association, 24(6), 1211-1220.
16. Григорків В. С., Дмитренко Н. П. Блокчейн – інформаційна технологія майбутнього. Молодий вчений. 2018. No3 (55) березень. С. 648-651.
17. Колиско О. Р. Перспективи та загрози використання технологій блокчейну в освітньому процесі. Педагогічні науки: теорія, історія, інноваційні технології. 2021. No 4 (108). С. 37-47.
18. Фесенко О. М., Колісніченко Д. В., Колісніченко А. В. Блокчейн-технології в банківській системі України: SWOT-аналіз. Вісник КНТЕУ. 2021. No 5. С. 120-135.
19. Сазонець І., Чубукова О., Лазуренко В. Проблеми та перспективи використання технології блокчейн в Україні. Бізнес Інформ. 2019. No5. С. 193–199.
20. Чубукова О. Ю., Лазуренко В. В., Резнікова Н. В. Застосування технології блокчейн в системі державного управління України. Публічне управління та митне адміністрування. 2019. No 2 (21). С. 95-103.
21. Марущак С.М., Водолазька Н.В. Використання технології блокчейн для забезпечення кібербезпеки систем електронного врядування в Україні. Збірник наукових праць Харківського національного університету Повітряних Сил. 2020. No 1(63). С. 171-175.

## ДОДАТОК А Окремі фрагменти програмного коду

## ДОДАТОК Б ПРЕЗЕНТАЦІЯ