НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК 004.056.55

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Оцінка ефективності атак на блокчейни BFT та PoS типу»**

Виконав:
студент IV курсу, групи ФІ-23
Радкевич Кирил Миколайович _____

Керівник:
професор кафедри ММЗІ, д.т.н., с.н.с.
Кудін Антон Михайлович _____

Рецензент:
посада, степінь, звання
Прізвище Ім'я По-батькові _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Радкевич Кирил Миколайович

1. Тема роботи: «*Оцінка ефективності атак на блокчейни BFT та PoS типу*», науковий керівник дипломної роботи: професор кафедри ММЗІ, д.т.н., с.н.с. Кудін Антон Михайлович,

затверджені наказом по університету №__ від «__» _____ 2026 р.

2. Термін подання студентом роботи: «__» _____ 2026 р.

3. Об'єкт дослідження: *протокол консенсусу DPoS, вплив атак на централізацію та методи протидії*

4. Предмет дослідження: *симуляція DPoS, його стійкість до різних сценаріїв атак*

5. Перелік завдань:

1)Огляд існуючих векторів атак;

2) Дослідження мjдифікацій DPoS та BFT протоколів;

3)Реалізація протоколу DPoS та атак на нього;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: *Презентація доповіді*

7. Орієнтовний перелік публікацій: *«планується доповідь на всеукраїнській конференції»*

8. Дата видачі завдання: 10 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2025 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2025 р.	Виконано
3	Вибір напряму та типу атак для дослідження	Жовтень-листопад 2025 р.	Виконано
4	Аналіз протоколів для обраних типів атак	Листопад-січень 2025-2026р.	Виконано
5	Реалізація симулятора протоколу DPoS та моделювання атак на нього	Лютий-квітень 2026р.	Виконано
6	Проходження преддипломної практики	Квітень-травень 2026р.	Виконано
7	Оформлення дипломної роботи	Травень-червень 2026р.	Виконано

Студент _____ Кирил РАДКЕВИЧ

Керівник _____ Антон Кудін

РЕФЕРАТ

Кваліфікаційна робота містить: 62 стор., 5 рисунків, 5 таблиць, 7 джерел.

У даній роботі були досліджені напрями атак на блокчейни. Зокрема розглянута атака pool hooping і механізм запобігання цій атаці. Основним об'єктом дослідження став протокол Delegated Proof of Stake, для якого був розроблений відповідний симулятор. Предметом дослідження стали різні сценарії таких атак: Gini Coefficient attack, атака картельних змов та апатія виборців. Зокрема були розглянуті модифікації протоколів Byzantine Fault Tolerance з метою детального аналізу їхнього механізму протидії зловмисникам в мережі.

Результатом цієї роботи є імітаційна модель DPoS та оцінка ефективності атак на нього. Окрім цього була запропонована модифікація DPoS на основі Raft для протидії таким вразливостям.

ПРОТОКОЛ КОНСЕНСУСУ, БЛОКЧЕЙН, DPOS, КОЛІЗІЙНА АТАКА

ABSTRACT

The qualification work contains: 62 p., 5 figures, 5 table, 7 sources.

This thesis investigates various attack vectors targeting blockchain networks. Specifically, it examines the pool hopping attack and explores mitigation mechanisms against it. The primary object of the research is the DPoS (Delegated Proof of Stake) consensus protocol, for which a dedicated simulator was developed. The subject of the study encompasses various attack scenarios, including the Gini Coefficient attack, cartel collusion attacks, and voter apathy. Additionally, modifications of BFT (Byzantine Fault Tolerance) protocols were analyzed to evaluate their defense mechanisms against malicious actors within the network.

As a result of this research, a DPoS simulation model was constructed, and the effectiveness of attacks against it was evaluated. Furthermore, a Raft-based DPoS modification was proposed to mitigate these vulnerabilities.

CONSENSUS PROTOCOL, BLOCKCHAIN, DPOS, COLLISION ATTACK

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

PoW — протокол Proof-of-Work

PoS — протокол Proof-of-Stake

BFT — протокол Byzantine Fault Tolerance

ВСТУП

Актуальність дослідження. Актуальність дослідження визначається широким розповсюдженням протоколів консенсусу типу BFT та PoS в сучасних блокчейн системах і є основою для побудови широкого класу сучасних децентралізованих інформаційних систем.

Метою дослідження є оцінка ефективності атак централізації на протколи зазначеного типу та побудова імітаційної моделі зазначених протоклів консенсусу щодо експерементального дослідження стійкості цих протоколів до сучасних атак.

1) Огляд існуючих векторів атак;

2) Дослідження мjдифікацій DPoS та BFT протоколів;

3) Реалізація протоколу DPoS та атак на нього.

Об'єктом дослідження є протоколи консенсусус PoS та BFT.

Предметом дослідження є атаки на зазначені протокли, які мають меті узурпувати право управління децентралізованою мережею при застосуванні протоколів PoS та BFT.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: теорія ймовірностей, системного аналізу, математичного моделювання та теорії інформації.

Наукова новизна отриманих результатів полягає в тому, що в перше, як атаки на PoS протоколи розглянути варінти атак характерних для PoW типу, зокрема pool hooping attack, а також розроблена формальна модель протколів DPoS для дослідження стійкості цих протоколів до атак централізації.

Практичне значення результатів полягає в тому, що отримані конкретні числові характеристики стійкості протоколів DPoS типу при різних практичних умовах їх використання.

1 ОГЛЯД НАУКОВИХ ДЖЕРЕЛ

В цьому розділі було розглянуто сучасний стан досліджень атак на блокчейн-системи, зокрема виконано класифікацію загроз за рівнями технологічного стеку та проаналізовано основні підходи до їх систематизації. Окрема увага приділялася атакам на механізми консенсусу, які мають критичний вплив на безпеку та стабільність децентралізованих систем. Окрім цього був виконаний детальноий аналіз конкретної атаки — **pool hopping**, а також дослідження механізмів протидії атаці данного типу на основі смарт-контрактів.

Метою даного розділу є аналіз існуючих підходів до класифікації атак на блокчейн-системи, їх вплив на централізацію, а також дослідження можливостей перенесення окремих атак, механізмів їх запобігання на блокчейни з консенсусними протоколами типу **PoS** та **BFT** з подальшим формуванням ідей для оцінки ефективності таких атак.

1.1 Блокчейн та протоли консенсусу

Блокчейн (англ. Blockchain) — це розподілений, децентралізований та незмінний ланцюг, який за обмежений час зберігає записи транзакцій та контрольні дані у вигляді послідовно пов'язаних блоків. Кожен блок створюється після задовільнення умов протоколу консенсусу і містить набір транзакцій, часову мітку та криптографічний геш попереднього блоку, що забезпечує незмінність.

Протокол консенсусу — це механізм, за допомогою якого вузли розподіленої мережі узгоджують єдине рішення щодо стану блокчейну, гарантуючи безпеку та детермінованість прийняття правильного рішення навіть за наявності альтернативних варіантів. Після досягнення згоди

відповідні вузли додають блок до свого локального ланцюга.

Протокол Proof of Work (PoW) — це механізм, за допомогою якого вузли розподіленої мережі виконують обчислювальну роботу для підтвердження блоків. Він передбачає пошук такого значення (nonce), геш якого починається з певної кількості нульових бітів. Складність роботи зростає експоненційно зі збільшенням кількості нулів, але перевірка правильності знайденого рішення потребує лише одного гешування.

У протоколі процес полягає у збільшенні nonce в блоці до тих пір, поки не буде знайдено значення, що задовольняє умови **PoW**. Після того як блок підтверджено, його змінити неможливо без повторення всієї виконаної роботи. Оскільки наступні блоки об'єднуються в ланцюжок після нього, будь-яка спроба змінити блок вимагатиме повторення роботи для всіх наступних блоків, що забезпечує стійкість і незмінність ланцюга.

Протокол Proof of Stake (PoS) — це механізм консенсусу, за допомогою якого валідатори (учасники протоколу) доводять свою зацікавленість у правильній роботі мережі, вносячи певну цінність (частку, stake), яку вони можуть втратити у разі шахрайства. Валідатори відповідають за перевірку транзакцій та блоків, а інколи — за створення нових блоків. Якщо валідатор діє нечесно (наприклад, пропонує суперечливі блоки або підтвердження), частина або вся його внесена частка може бути конфіскована або знищена. Тобто у **PoS** нові блоки створюються не через обчислювальну роботу, як у **PoW,** а через вибір валідатора на основі його частки (stake) та/або інших факторів, наприклад: час участі в мережі, випадковий відбір, репутація.

Протокол Practical Byzantine Fault Tolerance (PBFT) —- це алгоритм консенсусу для розподілених систем. Протокол розрахований на мережу з $n \geq 3k + 1$ вузлів, де до k вузлів можуть діяти неправильно або зловмисно. Інформація між вузлами шифрується за допомогою алгоритму SHA256 (одностороннє хешування), а алгоритм RSA використовується для генерації пар публічного та приватного ключів

кожного вузла, які потім використовуються для створення цифрових підписів. Процес консенсусу PBFT складається з трьох етапів: попередня підготовка (Pre-prepare), підготовка (Prepare) та фіксація (Commit).

Протокол Raft — це алгоритм консенсусу, який функціонує за принципом виділеного лідера, що є відповідальним за координацію та синхронізацію сигналів системного журналу на підпорядкованих вузлах. У межах цієї архітектури вузли можуть перебувати в одному з трьох базових станів: leader (лідер), candidate (кандидат) або follower (послідовник). Стабільність системи підтримується шляхом регулярного розсилання лідером періодичних сигналів перевірки активності (heartbeats). У разі фіксації таймауту відсутності таких сигналів, вузол-послідовник ініціює процедуру голосування, змінюючи свій статус на кандидата. Статус нового лідера присвоюється вузлу, який отримав быльше половини голосів. Такий підхід гарантує цілісність даних у мережі.

1.2 Класифікація та аспекти безпеки блокчейн-технології

Зростання економічної цінності та складності блокчейн-систем призвело до появи численних практичних атак із суттєвими фінансовими наслідками. Водночас наявні дослідження здебільшого зосереджені на окремих класах загроз і не пропонують уніфікованого підходу до оцінки їх ефективності, зокрема з урахуванням специфіки консенсусних протоколів типу BFT та Proof-of-Stake.

У цьому підрозділі будуть розглянуто і структуровано за допомогою нотації дерев 87 атак, які підсумували в [4]. На основі результатів, які отримали автори, сформулюємо наслідки для проєктування безпечних блокчейн-систем і окреслимо напрями подальших досліджень.

Згідно з поширеною класифікацією, розрізняють публічні блокчейни, де транзакції є загальнодоступними, та приватні блокчейни, у яких доступ до транзакцій мають лише авторизовані учасники. Крім

того, часні (permissioned) блокчейни обмежують участь у валідації транзакцій колом визначених вузлів, тоді як відкриті (permissionless) системи допускають участь будь-яких учасників. У межах цих архітектурних підходів з'явилися альтернативні механізми консенсусу, які замінюють обчислювально затратний та відносно неефективний Proof-of-Work, що використовується у Bitcoin. Хоча такі підходи часто забезпечують кращу масштабованість та ефективність, вони потребують вищого рівня довіри до вузлів, що беруть участь у мережі. Тому на практиці вони здебільшого застосовуються у дозволених блокчейнах, де учасники відомі та частково довірені.

Подальший розвиток базової технології дозволив реалізовувати довільну логіку у блокчейн-системах за допомогою смартконтрактів. *Смартконтракти* є комп'ютерними програмами, які зберігаються на вузлах P2P-мережі. Під час виклику функцій таких програм через транзакції всі вузли мережі виконують їхню логіку з надлишковістю. Смартконтракти є прозорими та доступними для аудиту всіма учасниками мережі.

Інформаційна безпека традиційно ґрунтується на трикутнику CIA, який охоплює цілі конфіденційності, цілісності та доступності. Останні дослідження розширюють ці базові цілі, додаючи такі властивості, як автентичність, підзвітність, аудитованість, надійність, невідмовність та приватність.

Кібератаки являють собою навмисний та несанкціонований доступ до систем і, відповідно, становлять загрозу цілям безпеки інформаційних систем. Атакувальники прагнуть досягти заздалегідь визначеної несанкціонованої мети, виконуючи послідовність спланованих дій. У процесі атаки використовуються різні інструменти для експлуатації вразливостей системи.

Блокчейн-технологія поєднує низку раніше відомих технологій, формуючи нову семантику безпеки інформаційних систем. Ключовими характеристиками безпеки блокчейнів є цілісність, незмінність та

децентралізація. Водночас ці властивості породжують і специфічні виклики для інформаційної безпеки. Розподілена природа блокчейнів, широке використання криптографії та прозорість інформації ускладнюють процеси безпечної розробки програмного забезпечення. Такі властивості, як зворотна незмінність, створюють нові проблеми безпеки. Низка відомих прикладів демонструє, що атаки, які експлуатують специфічні для блокчейнів вразливості, є практично здійсненними та становлять дедалі більшу загрозу для застосувань на їх основі.

Аналіз наявних оглядів атак на блокчейн-системи показує, що більшість досліджень зосереджені на конкретних реалізаціях, зокрема на блокчейнах із консенсусом Proof-of-Work. Водночас атаки на протоколи Byzantine Fault Tolerance та Proof-of-Stake залишаються менш систематизованими, а їх ефективність з точки зору порушення властивостей безпеки та економічної доцільності недостатньо досліджена.

1.3 Консолідація атак та узагальнення

У цьому розділі описуються всі типи атак на всі типи блокчейн-систем.

У процесі аналізу було ідентифіковано 87 атак, які були згруповані по таким критеріям: ціль атакувальника, наслідки атак, спільні характеристики реалізації тощо. У результаті було встановлено, що найкращу однозначну класифікацію забезпечує розподіл атак відповідно до технологічного стеку блокчейну, де кожна атака відноситься до окремого вектора атаки.

Виділяють п'ять рівнів узагальненого технологічного стеку блокчейну:

– P2P-мережа;

– механізм консенсусу;

– віртуальна машина та мова програмування;

– логіка застосунків (on-chain та off-chain);

– клієнтські застосунки та гаманці.

Атаки на P2P-мережу

P2P-мережа є базовим рівнем будь-якої блокчейн-системи та відповідає за комунікацію між вузлами. Атаки на цьому рівні часто є загальними мережевими атаками, зокрема DDoS або DNS-атаками. Наприклад, під час DDoS-атаки система перевантажується великою кількістю дрібних транзакцій, що дозволяє здійснювати інші атаки, зокрема подвійне витрачання.

Іншим прикладом є **routing**-атака, за якої мережа ділиться на ізольовані групи вузлів, а передача повідомлень між ними блокується, що може призвести до форків блокчейну. Загалом до цього вектора віднесено 15 атак.

Атаки на механізм консенсусу

Другий вектор атак стосується механізму консенсусу. Існують різні механізми консенсусу з різними вразливостями, однак найбільш дослідженими залишаються PoW-протоколи. Загалом розглядають атаки на консенсус як єдине ціле, оскільки більшість атак одночасно впливають і на створення, і на валідацію блоків.

Типовим прикладом є атака більшості (51%), за якої зловмисник контролює більшу частину ресурсів мережі та може впливати на порядок транзакцій. Іншим прикладом є атака Goldfinger, що особливо актуальна для PoS і базується на економічних стимулах та людському факторі. Також розглядається race-атака, яка експлуатує затримки підтвердження транзакцій.

Загалом до цього вектора було віднесено 27 атак, що підкреслює критичну роль консенсусу у безпеці блокчейн-систем.

Атаки на VM та мову програмування

Рівень віртуальної машини відповідає за детерміноване виконання логіки блокчейну. Атаки на цьому рівні зазвичай пов'язані з помилками

реалізації VM або мови програмування. Якщо ж функція реалізована коректно, але використовується небезпечно, така атака відноситься до рівня логіки застосунків.

Прикладом є short address attack в Ethereum, який експлуатує помилки обробки адрес. Загалом у цій категорії було ідентифіковано 11 атак.

Атаки на логіку застосунків (on-chain та off-chain)

Атаки на логіку застосунків поділяються на on-chain (смартконтракти) та off-chain (зовнішні сервіси та комунікації). On-chain атаки пов'язані з уразливостями смартконтрактів, код яких після розгортання є незмінним. Класичним прикладом є reentrancy-атака, що призвела до зламу DAO.

Off-chain атаки експлуатують зовнішні канали взаємодії, наприклад refund-атаки через man-in-the-middle. Загалом було ідентифіковано 16 on-chain атак та 11 off-chain атак.

Атаки на клієнтські застосунки та гаманці

Останній вектор атак стосується клієнтських застосунків і криптогаманців. Вони часто стають ціллю фішингу, соціальної інженерії або зловживання RPC-інтерфейсами. У разі незахищеного RPC-доступу можливе виконання довільних команд і компрометація приватних ключів. До цього вектора віднесено 8 атак.

Вектор атаки	**К-сть атак**	**Типи атак**
Атаки на P2P-мережу	15	Balance attack, BGP hijacking, DDoS attacks, Spam attack, Dust attack, Liveness denial, Deanonymization attack, Delay attack, DNS attack, Packet sniffing, Ransomware network attack, Routing attack, Sybil attacks, Eclipse attack, Timejacking attack
Атаки на механізм консенсусу	27	Block discarding attack, Block withholding attacks, Fork after withholding attack, Selfholding attack, Bribery attack, Changing system parameters, Double spend attacks, Alternative history attack, Finney attack, Race attack, Vector 76 attack, Front-running attack, Gas limit block stuffing, Goldfinger attack, Grinding attack, Incorrect transaction attack, Long range attack, Majority attack, Nothing-at-stake attack, Pool hopping attack, Punitive feather forking, Quantum attack, Resource exhaustion attack, Selfish mining, Splitting attack, Tradeoff attack, Triangle attack.
Атаки на VM та мову програмування	10	DoS with unexpected revert, Forcible balance transfer, Keeping secrets exploit, Overflow attack, Replay attack, Short address attack, Stack size exploit, Transaction malleability, Underflow attack, Opcode exploit.
Атаки на логіку on-chain застосунків	16	Call to the unknown, Delegate function exploit, EVM randomness exploit, Exception disorder vulnerability exploit, Fake deposit exploit, Gasless send exploit, Immutable bugs exploit, Multiple withdrawal attack, Permission check exploit, Reentrancy vulnerability attack, Rollback vulnerability exploit, Self-destruction attack, Timestamp dependence exploit, Tx.origin vulnerability exploit, Unbounded operations exploit, Vulnerable multisig exploit.
Атаки на логіку off-chain застосунків	11	Certificate authority attack, Coindash attack, Collusion attack, Cryptojacking, Ether delta attack, Fake receipt, Fake tokens, Flood & load attack, Overlay network DoS, Refund attack, SQL-injection attack.
Атаки на клієнтські застосунки та гаманці	8	Dictionary attack, Ether stealing attack, Flawed key generation exploit, Private key retrieval attack, RPC API exploit, Wallet malware attacks, Wallet availability attack, Wallet phishing.

Таблиця 1.1 – Таблиця атак на блокчейн-системи

Оскільки деякі атаки застосовні лише до конкретних протоколів, механізмів консенсусу або реалізацій, безпека блокчейн-системи значною мірою залежить від її конкретної інстанціації. На узагальненому рівні зазначається, що вразливості клієнтських застосунків і гаманців зазвичай мають менший системний вплив, ніж атаки на P2P-мережу, механізм консенсусу або віртуальну машину, які є глобально доступними компонентами.

Це дозволяє виділити ще одну важливу характеристику атак —

локальність вектора атаки. Атаки на ядро блокчейн-стеку є глобальними, тоді як атаки на гаманці часто потребують попереднього доступу до інфраструктури користувача.

Також встановлено, що частина атак є притаманною самій концепції блокчейну (наприклад, атаки на механізм консенсусу), тоді як інші виникають через помилки реалізації, зокрема у смартконтрактах. Останні часто можуть бути усунуті за допомогою належних процесів розробки та аудиту коду.

Як наслідок, можна зазначити, що не всі атаки є усувними. Частина атак може бути лише пом'якшена або взагалі неможлива для запобігання в рамках певних консенсусних механізмів (наприклад, атака 51%). Тому контрзаходи мають закладатися ще на етапі проєктування системи. Людський фактор є критично важливим як під час розробки (помилки у смартконтрактах), так і під час експлуатації (фішинг, крадіжка приватних ключів). Технічні заходи безпеки мають доповнюватися навчанням і підвищенням обізнаності користувачів.

Атаки можуть мати локальний або глобальний ефект. Наприклад, DDoS-атака впливає на всю мережу, тоді як компрометація одного гаманця має обмежений вплив. Тому пріоритет слід надавати атакам із потенційно системним ефектом.

1.4 Pool Hopping Attack

У попередньому розділі було розглянуто роботу, присвячену систематизації відомих атак на блокчейн-системи, зокрема атак на рівні мережі, консенсусу та прикладної логіки. Водночас такий узагальнений огляд не дозволяє детально проаналізувати механізми реалізації окремих атак та оцінити ефективність конкретних контрзаходів. Тому далі доцільно перейти до аналізу спеціалізованих досліджень, присвячених окремим класам атак. Однією з таких атак є pool hopping — яка є наслідком поведінки майнерів, які залишають майнінговий пул у періоди

зниження фінансової вигоди та знову приєднуються до нього тоді, коли винагорода за майнінг стає більш привабливою в блокчейн-мережі. Така стратегія приєднання до пулу лише у «вигідні» моменти призводить до того, що майнер отримує більшу винагороду, ніж відповідає його внесеній обчислювальній потужності.

Вихід майнерів із пулу зменшує його сукупну геш-потужність, унаслідок чого пул втрачає здатність успішно добути блок. У результаті конкуренти встигають згенерувати блок раніше. Існуючі дослідження описують методи виявлення та частково стійкі до атаки механізми, однак вони не пропонують надійного превентивного рішення, яке б ефективно стримувало майнерів від виходу з пулу.

Для запобігання атакам типу pool hopping у статті [5] запропоновано модель протидії, засновану на використанні смартконтрактів. Основною метою дослідження є підтримання симетричних відносин між майнерами шляхом зобов'язання всіх учасників безперервно надавати свою обчислювальну потужність до моменту успішного добування блоку.

Запропонована модель використовує реєстр майнерів у вигляді сертифікатів, які фіксують історію попередньої поведінки кожного майнера. Такий сертифікат дозволяє менеджеру пулу більш обґрунтовано формувати умови смартконтракту, захищаючи інтереси наявних учасників пулу. Модель запобігає частому перемиканню між пулами, оскільки майнери вносять кошти у вигляді ескроу та ризикують їх втратити у разі виходу з пулу до завершення майнінгу блоку.

Початковий дизайн блокчейну передбачав індивідуальний майнінг. Однак індивідуальний майнінг призводив до того, що винагороду отримували лише окремі великі майнери, а прибутковість майнінгу була непередбачуваною.

Окремий майнер зазвичай не здатен самостійно розв'язати задачу формування блоку, тоді як група майнерів може спільно успішно здійснити майнінг. У зв'язку з цим майнери почали об'єднуватися в

майнінгові пули, що дозволяє отримувати стабільніші винагороди та зменшувати індивідуальні обчислювальні витрати. Симетричний внесок майнерів у вигляді геш-потужності дозволяє пулу завершувати майнінг блоку швидше, ніж конкуруючі пули, а також забезпечує періодичну виплату винагород.

У разі успішного майнінгу винагорода між учасниками пулу розподіляється відповідно до визначених схем, таких як Pay-Per-Share (PPS), Pay-Per-Last-N-Shares (PPLNS) або Predictable Solo Mining (PSM). Сукупно майнінгові пули забезпечують близько 65,8% загальної обчислювальної потужності мережі Bitcoin та 76,9% мережі Ethereum (на 2019 рік), що робить їх критично важливими для підтримки блокчейн-інфраструктури.

Існує низка атак, спрямованих на майнінгові пули, зокрема pool hopping, block withholding (BWH) та fork after withholding (FAW). Серед них атака pool hopping вважається однією з найбільш небезпечних. Вона полягає в тому, що майнер переходить між різними пулами та бере участь у майнінгу лише тоді, коли очікуваний прибуток є високим, залишаючи пул у періоди низької прибутковості. Такі майнери отримують винагороду, вносячи менший внесок, ніж чесні учасники, що постійно залишаються в пулі.

Якщо атака pool hopping не буде обмежена, у чесних майнерів зникає мотивація залишатися в пулі. У результаті це може призвести до розпаду пулів і повернення до індивідуального майнінгу, який вимагає значних обчислювальних ресурсів і стає недоступним для більшості учасників. Це, своєю чергою, загрожує цілісності та стабільності блокчейн-мережі.

Далі ми розглянемо запропоновану в [5] систему, яка вводить поняття Hop_{Count} та використання коштів у вигляді escrow-депозиту. Hop_{Count} зростає кожного разу, коли майнер залишає пул, і визначає суму коштів, які майнер має внести як заставу. У разі дострокового виходу з пулу ці кошти втрачаються. Таким чином, escrow-депозит стимулює

майнера залишатися в одному пулі та підтримувати симетричну участь у майнінгу.

У роботі розглянуто проблему відсутності ефективних механізмів запобігання атакам pool hopping. Вихід майнерів із пулу зменшує його обчислювальну потужність, що призводить до втрати винагороди на користь конкурентів.

Основні результати роботи полягають у детальному аналізі атак pool hopping та існуючих підходів до їх пом'якшення, для чого автори запропонуванні моделі запобігання атаці на основі смартконтрактів. У межах цієї моделі було формалізовано оцінку ризиків майнера та розрахунок escrow-депозиту. Окрім цього був проведений аналіз впливаючих факторів, таких як: обчислювальна потужность, ризик та підзвітность. Для демонстрації своєї моделі був взятий приклад IoT-мережі.

1.5 Пов'язані роботи

У цьому підрозділі розглядаються дослідження, присвячені протидії атакам типу pool hopping. Ці роботи в основному зосереджені на стратегіях виявлення атак та механізмах стійкості до pool hopping. Крім того, розглядаються ключові фактори, які необхідно враховувати при розробці будь-якого механізму запобігання таким атакам. Дотримання цих факторів забезпечує практичну реалізовність та ефективність запропонованої моделі захисту.

Численні дослідники аналізували протоколи та методи, пов'язані з атаками pool hopping у майнінгових пулах. Пропонувалися методи аналізу таких атак, зосередившись переважно на їх виявленні. Вони використали модель часових епох для визначення того, чи отримував майнер винагороди з різних пулів у різні проміжки часу, аналізуючи транзакції в Coinbase-гаманцях.

Метод Slush запропонував систему балів, яка визначає винагороду

майнерів наприкінці кожного раунду. Також використовували теорію перспектив для прогнозування прибутку майнера з урахуванням його обчислювальної потужності та вартості електроенергії. У [2] запропонували децентралізований майнінговий пул, реалізований за допомогою смартконтрактів.

Однак наявні дослідження не здатні запобігти формуванню асиметричних відносин між майнерами, оскільки вони або лише виявляють порушників, або оптимізують розподіл винагород. Вони не перешкоджають майнерам залишати пул, що призводить до втрати сукупної обчислювальної потужності. В наступному абзаці будуть описані три ключові, які необхідно врахувати для ефективного запобігання атакам pool hopping.

Першим є обчислювальна потужність. Майнери потребують значної обчислювальної потужності для розв'язання криптографічних задач. Об'єднання в пул підвищує ймовірність успішного майнінгу. Вихід майнерів із пулу істотно знижує сукупний гешрейт, що ставить під загрозу головну мету — першими здобути блок. Наступним є те що нові майнери можуть становити ризик для пулу, особливо якщо вони мають історію частого переходу між пулами. Менеджер пулу має оцінювати ризик допуску такого майнера або встановлювати умови, що обмежують його можливість виходу. Третім є механізм відповідальності для майнерів, які залишають пул задля особистої фінансової вигоди. Такий механізм повинен робити вихід із пулу економічно невигідним і водночас захищати інтереси добросовісних учасників.

На основі аналізу наявних досліджень автори доходять висновку, що на сьогодні не існує ефективних механізмів запобігання pool hopping, а більшість рішень обмежуються лише виявленням або частковою компенсацією наслідків.

1.6 Запобігання атакам pool hopping на основі смартконтрактів

У цьому розділі пропонується модель запобігання атакам із використанням смартконтрактів, яка передбачає суттєве економічне покарання для майнера та надає фінансові вигоди майнінг-пулу у випадку, якщо майнер залишає пул до завершення процесу знаходження блоку.

Запропонована модель спрямована на запобігання серійним атакам типу pool hopping у блокчейн-мережах криптовалют. Зловмисні майнери використовують особливості механізмів розподілу винагород за успішне майнінг-блоків, що призводить до зменшення доходів чесних учасників. Для протидії цьому пропонується модель запобігання на основі смартконтрактів.

Переваги запропонованої моделі полягають у впровадженні комплексного підходу до безпеки, де менеджер пулу здійснює попередній облік поведінки кожного майнера перед наданням дозволу на приєднання до мережі. Крім того, модель унеможливлює часті переходи між пулами завдяки обов'язковому внесенню майнерами коштів у вигляді ескроу-депозиту, який конфіскується в разі виявлення атаки. Точність реалізації цього механізму забезпечується числовою моделлю, яка дозволяє розрахувати індивідуальний та точний розмір ескроу для кожного окремого учасника.

Модель складається з таких етапів:

– майнер подає запит на приєднання до пулу;

– менеджер пулу отримує адресу майнера та знаходить його сертифікат у блокчейні;

– якщо майнер має історію порушень (високий показник переходів між пулами або порушень смартконтрактів), він повинен погодитися на умови смартконтракту з депозитом;

– після завершення майнінгу сертифікат оновлюється, а депозит або повертається, або конфіскується.

Сертифікат майнера містить:

– адресу майнера;

– кількість переходів між пулами (Hop_{Count});

– кількість виконаних смартконтрактів (SC_{upheld});

– кількість порушених смартконтрактів ($SC_{violated}$).

Модель складається з трьох модулів:

1. Оцінювання (Evaluation)

Менеджер пулу оцінює ризик майнера на основі:

кількості переходів між пулами ($\alpha = (Hop_{Count}$);

кількості порушених контрактів ($\beta = SC_{violated}$).

Ризик майнера обчислюється як:

$$M_{rsk} = \alpha + \beta$$

Якщо ризик дорівнює нулю — майнер вважається безпечним, інакше — ризикованим.

2. Умови смартконтракту (Terms)

Для ризикових майнерів визначається депозит:

$$Es_{dep} = \alpha + \beta \ - \mu$$

де μ — кількість виконаних контрактів (SC_{upheld}).

Депозит виконує роль економічного стримувального механізму: у разі виходу з пулу до завершення майнінгу майнер втрачає кошти.

3. Оновлення сертифіката (Update)

– якщо майнер завершує майнінг — депозит повертається, ризик зменшується;

– якщо майнер покидає пул — депозит конфіскується, ризик зростає.

Таким чином, поведінка майнера фіксується і впливає на майбутні умови участі.

Запропонована модель поєднує смартконтракти та репутаційні сертифікати для стримування pool hopping атак і забезпечення справедливого розподілу ресурсів у майнінг-пулах.

1.7 Оцінка запропонованої моделі

У цьому розділі запропоновану модель оцінюють за допомогою практичного прикладу, що базується на двох різних сценаріях: коли майнер завершує майнінг блоку та коли майнер залишає майнінг-пул. Обидва сценарії аналізуються відповідно до запропонованої методології. Втрата обчислювальної потужності моделюється шляхом порівняння наявних досліджень із запропонованою моделлю. Також подано порівняльний аналіз запропонованої моделі з пов'язаними роботами. Обговорюються обмеження запропонованого методу та його наслідки.

Практичний приклад базується на двох сценаріях. В першому майнер здійснює атаку типу pool hopping. В другому майнер залишається учасником майнінг-пулу та завершує майнінг блоку.

У блокчейн-системі на основі Інтернету речей IoT менеджер пулу керує майнінг-пулом для однієї IoT-орієнтованої блокчейн-мережі «розумної будівлі». Майнінг-пул працює у публічній блокчейн-мережі Ethereum. Для налаштування вузлів Ethereum та підключення до мережі Mainnet використовується інструмент командного рядка Geth. Застосунок Mist використовується як користувацький інтерфейс для взаємодії з Geth. Веббраузерне інтегроване середовище розробки Remix застосовується разом із мовою Solidity для створення смартконтрактів. Інтерфейс прикладного програмування Web3 Ethereum JavaScript надає необхідний набір бібліотек для взаємодії з вузлом Ethereum. Для оцінювання запропонованої моделі використовується майнінг-пул Ethpool. Розглядаються два сценарії, описані нижче.

Опис першого сценарію атаки:

У розглянутому сценарії майнер $M1$, який працює в

Ethereum-орієнтованому майнінговому пулі Ethpool, що обслуговує IoT-блокчейн для «розумного будинку», виявляє, що альтернативний пул, пов'язаний із «розумною будівлею», пропонує вищу очікувану фінансову винагороду. Внаслідок цього майнер переводить усю свою обчислювальну потужність до іншого пулу, зменшуючи ефективність початкового пулу.

Етап оцінювання ризику майнера

Після запиту майнера $M1$ на приєднання до пулу Ethpool менеджер пулу виконує оцінку ризику. Для цього використовується сертифікат майнера, який зберігається у блокчейн-мережі та містить такі показники:

– кількість переходів між пулами (Hop_{Count}),

– кількість дотриманих смартконтрактів (SC_{upheld}),

– кількість порушених смартконтрактів ($SC_{violated}$).

У даному сценарії сертифікат майнера M1 має значення: $Hop_{Count} = 4, SC_{upheld} = 2, SC_{violated} = 3$, що свідчить про часту зміну пулів у минулому.

Ризик майнера обчислюється за формулою:

$$M_{rsk} = \alpha + \beta$$

де α – Hop_{Count}, а β – $SC_{violated}$. Отримане значення вказує на високий ризик для майнінгового пулу.

Формування умов смартконтракту

Оскільки майнер оцінюється як ризиковий, менеджер пулу формує смартконтракт, який передбачає внесення грошового депозиту (escrow). Розмір депозиту визначається з урахуванням попередньої поведінки майнера за формулою:

$$Es_{dep} = \alpha + \beta \ - \mu$$

де μ — кількість дотриманих смартконтрактів(SC_{upheld}). У цьому випадку майнеру необхідно внести **5 ETH** як гарантійний депозит.

Метою такого механізму є створення економічного стимулу для

майнера залишатися в пулі до завершення майнінгу блоку.

Реалізація атаки та оновлення сертифіката

Після приєднання до пулу майнер $M1$ починає участь у майнінгу. Однак зі зменшенням геш-потужності пулу та зростанням часу очікування видобутку блоку майнер ухвалює рішення залишити Ethpool і перейти до більш продуктивного пулу. Таким чином, майнер здійснює атаку pool hopping, порушуючи умови смартконтракту.

Відповідно до умов контракту:

– депозит у розмірі **5 ETH** конфіскується;

– кошти рівномірно розподіляються між іншими учасниками пулу;

– сертифікат майнера оновлюється: збільшується Hop_{Count} і $SC_{violated}$, зменшується SC_{upheld}.

Опис другого сценарію атаки:

Майнер приймає рішення залишити пул у разі зменшення його загальної хеш-потужності та збільшення очікуваного часу майнінгу блоку, переходячи до іншого пулу з вищою прибутковістю. Така поведінка зменшує колективну обчислювальну потужність пулу та знижує ймовірність успішного видобутку блоку.

Запропонована модель використовує смартконтракти з механізмом ескроу для економічного стримування подібних атак. Майнер із зафіксованою історією переходів між пулами зобов'язаний внести заставу у вигляді криптовалюти. У разі дострокового виходу з пулу кошти конфіскуються та розподіляються між чесними учасниками, що компенсує втрату обчислювальних ресурсів.

На основі числового прикладу показано, що економічні втрати майнера від конфіскації ескроу значно перевищують потенційну вигоду від зменшення витрат на електроенергію при зміні пулу. Таким чином, модель створює фінансовий стимул залишатися в пулі до завершення майнінгу блоку.

У разі дотримання умов смартконтракту ескроу повертається майнеру разом із винагородою, а його репутаційні показники

покращуються. Це демонструє ефективність запропонованого підходу як превентивного механізму, а не лише засобу виявлення атак.

Проведений порівняльний аналіз демонструє, що існуючі підходи, такі як epoch-based detection, метод Slush, зосереджені переважно на постфактум-виявленні атаки pool hopping або забезпеченні справедливого розподілу винагород, проте вони не здатні ефективно запобігати самим діям зловмисного виходу майнера з пулу. На відміну від них, запропонована модель комплексно нівелює ці ризики: вона гарантує збереження обчислювальної потужності пулу шляхом впровадження механізму фінансових санкцій, забезпечує превентивну оцінку надійності майнера ще на етапі його приєднання до пулу та підтримує незмінну підзвітність завдяки збереженню історії попередньої поведінки учасників.

Хоча запропонована модель орієнтована на PoW-блокчейни, вона демонструє загальний підхід до оцінки ефективності атак через економічні стимули та санкції. Подібні принципи можуть бути адаптовані для аналізу атак у системах з консенсусами BFT та Proof-of-Stake, де економічні механізми також відіграють ключову роль.

Новизна підходу полягає у попередній оцінці ризику майнера перед його допуском до майнінгового пулу, застосуванні смартконтрактів з механізмом застави (escrow) та використанні сертифікатів майнерів для зберігання історії їхньої поведінки. Запропоновані математичні моделі дозволяють формалізувати рівень ризику та визначити розмір застави, що створює економічно невигідні умови для дострокового залишення пулу. Таким чином, досягається збереження симетрії між учасниками пулу та захист колективної обчислювальної потужності.

Висновки до розділу 1

В главах 1.1-1.4 були описані класи атак на блокчейни. Результати підсумків дають нам розуміння на які фрагменти мережі направлена найбільша теоретична розробка атак. Проведені огляди атак на

блокчейн-системи забезпечують систематизацію загроз, однак переважно не містять кількісної оцінки їх ефективності, вартості чи впливу на властивості консенсусних протоколів. З огляду на обмеженість аналізу альтернативних механізмів консенсусу, таких як BFT та PoS, актуальним є подальше дослідження атак саме з позицій їх ефективності та практичної реалізовності.

В другому підрозділі була розглянута pool hopping-атака та модель запобігання орієнтована на блокчейни з консенсусом Proof-of-Work. Хоча атака pool hopping є характерною для блокчейн-систем з консенсусом Proof-of-Work, її базова ідея — опортуністична поведінка валідаторів з метою максимізації власного прибутку за рахунок колективної безпеки системи — має аналоги і в блокчейнах з консенсусами Proof-of-Stake та Byzantine Fault Tolerance. Це дозволяє узагальнити запропоновані підходи та розглядати їх як основу для аналізу ефективності атак і механізмів стримування у ширшому класі блокчейн-систем.

Отримані результати створюють основу для подальшого аналізу можливості перенесення подібних атак та механізмів протидії на блокчейн-системи з консенсусними протоколами типу Proof-of-Stake та Byzantine Fault Tolerance, а також для розробки підходів до оцінки ефективності таких атак у відповідних середовищах. Хоча розглянута атака не маэ прямих аналогів у PoS та BFT, вона розглядається в розрізі її можливого застосування як атаки централізації.

2 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ В АЛГОРИТМАХ КОНСЕНСУСУ

На початку даного підрозділу були дані короткі відомості з теорії графів. Далі був описаний PBFT в MAS, де на його основі запропонували покращену версію протоколу PBFT-Raft. В кінці цієї глави наведено покращений DPoS-Raft, який має захист від атак централізації.

2.1 Теоритичні відомості

Згідно з роботою [3], мультиагентні системи можна визначити як підкатегорію складного програмного забезпечення, що розгортається як високорівневий застосунок над мережевою архітектурою OSI. Специфіка таких систем, особливо відкритого типу, полягає в автономності й самостійності агентів, що вимагає розробки спеціалізованих підходів до безпеки, оскільки традиційних засобів нижніх рівнів (шифрування, автентифікація) часто виявляється недостатньо для захисту від специфічних поведінкових загроз.

Для подальшого опису моделі протидії атакам на централізації введемо наступні поняття. Система зв'язку в MAS складається з мережі, а зв'язки та комунікація між кожним агентом описується аппаратом теорії графів. Структура комунікаційної мережі описується неорієнтованим графом $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, \mathcal{A}_\mathcal{G})$, де $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ — множина, що представляє собою скінченний набір вузлів, кожному з яких присвоєно унікальний порядковий номер. $\mathcal{E}_\mathcal{G} \subseteq \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$ — це множина ребер, що визначають наявність каналів зв'язку для обміну даними між парами агентів (v_j, v_i). Окрім того для кожного окремого i-го вузла визначається множина його сусідів $\mathcal{N}_i = \{j \in \mathcal{V}_\mathcal{G} : (v_j, v_i) \in \mathcal{E}_\mathcal{G}\}$. У задачах аналізу стійкості консенсусу також враховується включна множина

сусідів $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{v_i\}$, яка додатково містить власний стан поточного вузла. $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ матриця суміжності, де елемент a_{ij} представляє зв'язок між вузлами j та i. $a_{ij} = 1$, якщо між вузлами j та i існує ребро зв'язку $(v_j, v_i) \in \mathcal{E}_\mathcal{G}$, та $a_{ij} = 0$ у протилежному випадку.

Для опису динаміки поширення інформації в мережі її вводиться матриця Лапласа L. Вона обчислюється як різниця між матрицею степенів вузлів D та матрицею суміжності $\mathcal{A}_\mathcal{G}$:

$$L = D - \mathcal{A}_G$$

Елементи матриці Лапласа відображають кількість зв'язків (степінь) кожного конкретного вузла, тоді як позадіагональні елементи $l_{ij} = -a_{ij}$ вказують на наявність або відсутність безпосереднього каналу зв'язку між i-м та j-м агентами. Коли L має лише одне власне значення, яке дорівнює нулю, неорієнтований граф є зв'язним, що означає, що між будь-якими двома вузлами графа існує канал зв'язку.

У дискретній MAS першого порядку, що складається з n агентів, динамічний стан відображається наступним чином:

$$x_i(k+1) = x_i(k) + u_i(k), \quad i \in (1, 2, ..., n)$$

де x_i та u_i представляють інформацію про стани та вхідні дані керування відповідно вузла v_i у момент часу k. Вхідні дані керування кожного агента можна описати наступним чином:

$$u_i(k) = -\sum_{v_j \in \mathcal{N}_i} a_{ij}[x_i(k) - x_j(k)], i \in (1, 2, ..., n)$$

Кінцевою метою функціонування такої децентралізованої мережі є досягнення узгодженого стану між усіма вузлами. Для досягнення консенсусу MAS має відповідати наступній умові:

$$\lim_{k \to \infty} \|x_i(k) - x_j(k)\| = 0, \quad \forall i, j \in (1, 2, ..., n)$$

Відповідно до класифікації зловмисних вузлів та механізму атак, атаки на MAS поділяються на аварійні (crash) атаки, змовницькі (colluding) атаки та візантійські (Byzantine) атаки.

Означення 2.1. (Візантійський вузол): Якщо вузол $i \in \mathcal{V}_\mathcal{G}$ у MAS надсилає окрему інформацію сусідам та застосовує іншу функцію для оновлення свого стану на певному кроці часу, він називається Візантійським вузлом.

Означення 2.2. (Зловмисний вузол): Якщо вузол $i \in \mathcal{V}_\mathcal{G}$ на кожному кроці часу він надсилає однакову інформацію усім своїм сусідам, але на якомусь кроці часу застосовує якусь іншу функцію, він називається зловмисним вузлом.

Зауважимо, що як зловмисним, так і візантійським вузлам дозволено оновлювати свої стани довільно (тобто вони мають змогу вступати для цього у змову з іншими зловмисними або візантійськими вузлами). Єдина відмінність полягає в їхній здатності до дворушництва (duplicity). Якщо мережа реалізована за допомогою сенсорного або широкомовного зв'язку, природно припустити, що вихідні сусіди отримують однакову інформацію, що й мотивує визначення зловмисного вузла. Проте, якщо мережа є типу «точка-точка» (point-to-point), то можлива візантійська поведінка. Зауважимо, що всі зловмисні вузли є візантійськими [1].

Припустимо, що агент $v_q, q \in \Gamma$ є візантійським вузлом у MAS. У наступній формулі довільна функція f_q вводиться як функція для ітераційних оновлень стану візантійських агентів, і її динамічне рівняння може бути описане таким чином:

$$xq(k+1) = f_q(x_q(k)), \quad q \in \Gamma.$$

У MAS, яка містить візантійських вузлів, якщо нормальний вузол має щонайбільше f зловмисних сусів або вийшли з ладу щонайбільше f сусідів, це називається локальною моделлю атаки для MAS [1]. Зазвичай

кількість зловмисних вузлів у всій системі встановлюється меншою за f (щонайбільше f із n вузлів виходять із ладу або є скомпрометованими), що є глобальною моделлю атаки.

Для мінімізації впливу візантійських вузлів у рамках класичної теорії керування найчастіше застосовується алгоритм Mean Subsequence Reduced(MSR). Основна ідея цього підходу полягає в локальному сортуванні даних та фільтрації крайніх випадків. Алгоритм MSR на кожному кроці ітерації k виконує таку послідовність дій для кожного чесного вузла в системі: спочатку значення станів, отримані від усіх суміжних вузлів-сусідів, сортуються в порядку спадання. Після цього обчислюється адаптивний параметр фільтрації $m_i(k)$, який залежить від загальної кількості сусідів та очікуваного числа зловмисників f.

З відсортованого списку видаляється по $m_i(k)$ найбільших та найменших значень, які суттєво відрізняються від власного значення поточного агента.Оновлення стану: Керуючий вплив обчислюється виключно на основі відфільтрованої безпечної підмножини даних.

За умови успішного відсікання шкідливих даних система досягає стану безпечного консенсусу (secure consensus). Математично це означає, що значення станів усіх нормальних агентів завжди залишаються в межах безпечного діапазону, обмеженого мінімальним $s(0)$ та максимальним $S(0)$ початковими значеннями валідних вузлів, а їхні фінальні координати асимптотично збігаються:

$$s(0) \leq \min_{i \in \Gamma} x_i(k) \leq \max_{i \in \Gamma} x_i(k) \leq S(0)$$

$$\lim_{k \to \infty} |x_i(k) - x_j(k)| = 0, \quad \forall i, j \in \Gamma_c$$

де Γ_c — підмножина виключно нормальних (легітимних) агентів системи. Проте, за умови високої концентрації атакуючих вузлів та наявності "китів"(whales) у мережі, геометрична фільтрація MSR виявляється неефективною, що потребує інтеграції криптографічних засобів у блокчейн мережу.

Хоча MSR досягає безпечного консенсусного контролю над MAS при наявності візантійських вузлів, він стає неефективним, коли кількість візантійських вузлів у топології мережі наближається до нескінченності. В такому випадку у чесного вузла декілька сусідів можуть бути зловмисними, що унеможливлює усунення екстремальних значень стану. Тому, щоб алгоритм міг ефективно застосовуватись стає критично важливо забезпечити надійність мережі та обмежити кількість візантійських вузлів.

2.2 Покращений PBFT на основі Raft

В роботі [7]автори запропонували вдосконалений підхід, який інтегрує PBFT та покращену версію Raft. Це нововведення полегшило виявлення та перевірку візантійських вузлів, що гарантує досягення консенсусу у MAS.

У мережі MAS із візантійськими вузлами алгоритм PBFT залучається для верифікації обміну інформації між вузлами. Ця верифікація здійснюється за допомогою криптографічних механізмів, перевірки підписів та трифазного процесу консенсусу. Згодом візантійські вузли ідентифікуються на основі відповідей, отриманих клієнтом.

Спочатку, кожен вузол за допомогою алгоритму RSA генерує пару приватного та публічного ключів для кожного вузла в MAS. При відправці повідомлень вузол використовує свій закритий ключ для створення цифрового підпису. Після отримання повідомлення інші вузли використовують відкритий ключ відправника для перевірки підпису та перевіряють, чи не було змінено вміст повідомлення, для підтвердження особистості відправника. Функції шифрування та розшифрування для RSA визначаються наступним чином:

$$M^* = M^{priv} mod b$$

$$M = (M^*)^{pub} mod b$$

В цих формулах M виступає повідомленням, а M^* — це зашифроване повідомлення. Пара ($priv$,b) утворює закритий ключ, де $priv$ є взаємно простим із добутком $(p-1)$ та $(q-1)$, (pub,b) відкритий ключ відповідно. p і q — два простих числа, а число $b = p \cdot q$.

Цифровий підпис використовується над повідомленням, яке попередньо було захешовано алгоритмом $SHA256$. Під час процесу передачі повідомлення воно гешується та шифрується, а закритий ключ відправника використовується як цифровий підпис. Повідомлення гешується наступним чином:

$$x_i^*(k) = SHA256(x_i(k))$$

де $x_i^*(k)$ представляє геш-значення, отримане шляхом шифрування інформації про стан $x_i(k)$ вузла i в момент часу k за допомогою односпрямованої хеш-функції SHA256. Після отримання повідомлення вузол-одержувач використовує заздалегідь відомий відкритий ключ вузла-відправника для автентифікації цифрового підпису отриманого повідомлення. Ситуації, коли автентифікація цифрового підпису зазнає невдачі або дані є неповними, вважаються потенційною передачею зловмисного повідомлення, тим самим підтверджують існування візантійських вузлів.

Наступним кроком є опис трифазного механізму верифікації та принципів його реалізації. Нехай кількість візантійських вузлів дорівнює f, тоді процес консенсусу можна поділити на п'ять етапів: стадії Запиту (Request), Попередньої підготовки (Pre-prepare), Підготовки (Prepare), Фіксації (Commit) та Відповіді (Reply). Хоча саме виявлення візантійських вузлів відбувається у три таких етапи верифікації: Попередньої підготовки, Підготовки та Фіксації. Далі детально опишемо кожен етап:

Запит: Клієнт надсилає повідомлення запиту до первинного вузла

(primary), який обирається вузлами відповідно до порядкового номера агентів. Рішення щодо того хто саме буде первинним вузлом pn приймається так:

$$pn = vie \ mod \ N$$

де vie — це номер епохи, а N загальна кількість вузлів. Якщо первинний вузол виходить з ладу або виявляється візантійським, тоді номер епохи збільшується. Структура повідомлень складається з чотирьох елементів. Де o представляє конкретну операцію запиту, t — часову мітку (timestamp), а c ідентифікує повідомлення, надіслане вузлом. Загалом це повідомлення представляється так: $\langle Request,o,t,c\rangle$.

Попередня підготовка: після того як первинний вузол отримує запит від клієнта він верифікує його і надсилає всім іншим вузлам в мережі повідомлення $\langle\langle Pre-prepare,vie,n,d\rangle,m\rangle$, яке містить номер епохи, номер повідомлення, дайджест(геш повідомлення) і саме повідомлення.

Підготовка: наступним кроком після отримання повідомлення попередньої підготовки вузли верифікують його. Якщо повідомлення приймається, то починається етап підготовки. На цьому етапі всі вузли транслюють повідомлення $\langle Prepare,vie,n,d,i\rangle$ до якого додається i, яке виступає номером вузла. Після того як вузол отримує $2f$ таких повідомлень, які пройшли верифікацію, він транслює повідомлення $\langle Commit,vie,n,d,i\rangle$ і переходить до наступного етапу.

Фіксація: далі всі отримують $2f + 1$ однакових повідомлень Commit(включно зі своїм). Вони всі верифікуються і після проходження верифікації процес верифікації консенсусу вважається завершиним.

Відповідь: у подальшому, кожен вузол, який завершив попередні етапи надсилає відповідь клієнту. Як тільки клієнт отримує $f + 1$ однакових відповідей вважається, що процес автентифікації завершено.

Цей процес верифікації направлений на те, що візантійські вузли не зможуть пройти перевірку цифрового підпису, надіслати відповідь клієнту і успішно завершити консенсус.

Якщо стається такий випадок, коли первинний вузол є візантійським, в мережі ініціюється протокол зміни епохи, в якому обирається новий первинний вузол. Процес обрання нового первинного вузла можна представити таким чином:

Всім вузлам в мережі надсилається повідомлення для зміни епохи $\langle View - Change, vie + 1, n, P, i\rangle$ ініційоване резервними вузлами, в якому P представляє сукупність повідомлень Pre-prepare та Prepare, які вузол наразі отримав, але ще не завершив їх обробку. Новий первинний вузол обирається як $p_{new} = (vie + 1)\ modN$. Після отримання $2f + 1$ повідомлень про зміну епохи, транслюється повідомлення $\langle New - View, vie + 1, VC, O\rangle$ ініційоване первинним вузлом, яке встановлює нову епоху. В якому VC множина повідомлень про зміну епохи, надісланих резервними вузлами, а O являє собою множину повідомлень, які не завершили процес верифікації. Після цього ці повідомлення можуть бути трансльовані, верифіковані та отримані знову і процес зміни епохи буде завершено.

Згідно попередніх кроків можна вважати, що візантійські вузли будуть ідентифіковані. Наступним кроком буде ізоляція цих вузлів, і механізм консенсусу буде безпечним.

Оскільки учасники протоколу PBFT серйозно обмежені умовами комунікації в мережі(кожен учасник повинен мати змогу напряму комунікувати зі всіма іншими вузлами в мережі), ця властивість часто є недосяжною в реальних умовах. Окрім цього така комунікація повинна бути стійкою.

Через те що вказані вище недоліки ускладнюють практичну реалізацію цього методу в статті [7] автори пропонують виявляти візантійськи вузли усередині мережі. Для цього вони використовують покращену версію алгоритму групування PBFT-Raft, який використовує переваги обох алгоритмів.

Головна ідея побудувати механізм консенсусу серед лідерів навколо яких групуються вузли. Задача лідерів проводити верифікацію і поширювати інформацію в своїй групі. Тобто лідер ідентифікує

візантійські вузли, ізолює їх, і чесні вузли виконують звичайний протокол консенсусу. Такий підхід спрощує вимоги до комунікації в мережі підвищуючи її стійкість.

Далі буде описаний процес групування на основі поєднання покращеного алгоритму PBFT-Raft у наступних умовах.

1) Групування вузлів на основі топології: На основі матриці степенів D, зв'язності та відносин сусідства всередині заданої комунікаційної топології мережі визначаються початкові лідери в системі. Нехай множина лідерів позначається як $\Gamma_l = \{l_1, l_2, \ldots, l_{ln}\}$, де ln — загальна кількість лідерів. Далі всі вузли групуються на основі цих лідерів. До всіх лідерів застосовуються строгі умови, де лідер повинен бути вузлом з найбільшим степенем і зв'язністю. Лідери здатні бути попарно зв'зні і через вимогу BFT їх щонайменше чотири.

Після визначення початкових лідерів решта вузлів призначаються підпорядкованими (followers), множина підпорядкованих вузлів позначається як $\Gamma_{fl} = \{fl_1, fl_2, \ldots, fl_{fn}\}$, де fn — загальна кількість підпорядкованих вузлів. Кожен лідер може належати тільки до однієї групи. Всі інші вузли підпорядковуються за такими правилами:

Якщо підпорядкований вузол fl_i підключений лише до одного лідера l_j, він призначається в групу GP_j, якою керує цей лідер. Якщо вузол підключений до декількох або жодного лідера то він призначається до групи до якої належить більшість його сусідів. Відносини сусідства визначаються на основі матриці суміжності A_G. Наприклад, якщо підпорядкований вузол fl_i підключений до лідерів l_j та l_m, і більшість його сусідів належать до GP_j, тоді fl_i призначається в GP_j.

2) Процес консенсусу поза групою

Оскільки лідер сповна впливає на досягення консенсусу в групі, тому для ідентифікації візантійських вузлів серед лідерів був застосований механізм верифікації у PBFT. Для цього протоколу механізм був застосований таким чином:

Первинний вузел обирається лідерами відповідно до порядкового номера,

якому під час фази Request клієнт надсилає повідомлення запиту. Фаза Pre-prepare ідентична до PBFT. Фаза Prepare відрізняється тим, що лідер повинен отримати $2f_l$ однакових повідомлень Prepare, перед тим як перейти до наступної стадії та транслювати повідомлення Commit. Всі лідери отримують однакові $2f_l + 1$ повідомленя Commit, які пройшли верифікацію і процес верифікації консенсусу з фазою Commit вважається завершеним. Після цього вони надсилають повідомлення відповідь клієнту і як тільки клієнт отримує $f_l + 1$ однакових відповідей, це означає, що процес автентифікації лідерів успішно завершений.

$$Re(x_i(Prepare)) \geq 2f_l, \quad i \in \Gamma_l$$

$$Re(x_i(Commit)) \geq 2f_l + 1, \quad i \in \Gamma_l$$

$$Re(x_i(Reply)) \geq f_l + 1, \quad i \in \Gamma_l$$

Де Γ_l — множина лідерів. Завдяки процесу верифікації візантійські вузли серед лідерів за допомогою перевірки цифрового підпису будуть ідентифіковані, оскільки вони не здатні успішно завершити процес консенсусу або надіслати повідомлення відповіді клієнту.

3) Процес консенсусу всередині групи

Далі в групах пропонується використовувати покращений алгоритм Raft через його високу ефективність консенсусу, низьку комунікаційну складність та масштабованість.

Як тільки процес виборів і групування завершується, лідер надсилає сигнал серцебиття (heartbeat) підпорядкованим вузлам (followers) у групі. Сигнал серцебиття хешується і підписується лідером. Далі кожен вузел в группі за допомогою відкритого ключа лідера верифікує повідомлення та визначає чи є вузел візантійським. Якщо лідер не є візантійським, то вузел відповідає на цей сигнал. Паралельно з цим журнал лідера реплікується на підпорядковані вузли, після завершення синхронізації клієнту надсилається підтвердження. У разі виявлення факту фальсифікації даних лідером, що свідчить про його належність до

візантійських вузлів, підпорядковані вузли ініціюють процедуру вибору нового лідера.

Може трапитися, що вузол підключений до іншого лідера, в такому випадку він синхронізує інформацію з лідером до якого він підключений. Після цього він надає відповідь клієнту та поширює інформацію логу на будь-які інші підпорядковані вузли у своєму з'єднанні. Якщо вузол не підключений до якогось лідера, він дублює інформацію логу від будь-якого іншого підключеного вузла. Після цього він здатний передати інформацію про синхронізацію логів клієнту.

Після відповіді на сигнал сердцебиття, за допомогою цифрового підпису, лідери розпізнають візантійські вузли в середині своїх груп і припиняють з'єднання, що унеможливить їх подальшу комунікацію з клієнтом.

2.3 Використання представленого механізму у DPoS

Класичний DPoS має велику перевагу у масштабованості і швидкості, але разом з цим він є надзвичайно вразливий до атак централізації і колізійних атак. PBFT не має таких вразливостей, але його масштабування дуже обмежене вимогами до комунікації. Тому для усунення недоліків цих двох протоколів, ми запропонуємо обєднати ідею голосування DPoS і запропонований вище механізм протидії зловмисним вузлам.

У DPoS відбір учасників відбувається за критерієм кількості активів(стейку) у обраного вузла. Фактично кожен учасник голосування може подати себе у роль кандидата в делегати. Через такий примітивний алгоритм відбору кандидатів протокол втрачає гнучкість в разі потрапляння зловмисних вузлів у множину потенційних творців блоків. У роботі [6] був запропонований механізм пониження, який хоча і швидко виключає зловмисників, але не запобігає їх повторному потраплянні в пул. Далі буде представлено модифікований DPoS-Raft:

Етап формування пулу кандидатів відбувається таким чином: з

множини вузлів V випадковим чином (стохастичний відбір) виокремлюється підмножина кандидатів у делегати $C(k) \subset V$, причому потужність цієї множини обмежена параметром M ($|C(k)| = M$, де $M < N$). Процес генерації пулу кандидатів можна представити оператором випадкового вибору $\mathcal{R}$:

$$C(k) = \mathcal{R}(V, M)$$

Далі відбувається голосування та первинне ранжування. Кожен вузол системи $v_i \in V$ володіє визначеною вагою голосу (стейком) $w_i(k) \in \mathbb{R}^+$. Вузли розподіляють свої голоси на користь кандидатів з множини $C(k)$. Результати голосування формалізуються у вигляді вектора сумарних голосів $S = [s_j]$, де для кожного кандидата $c_j \in C(k)$ акумулюється сумарна вага відданих за нього голосів:

$$s_j = \sum_{i=1}^{N} \alpha_{ij} w_i(k), \quad \forall c_j \in C(k)$$

де $\alpha_{ij} = 1$, якщо вузол v_i проголосував за кандидата c_j, та $\alpha_{ij} = 0$ в іншому випадку. На основі обчисленого вектора S формується впорядкована за спаданням сумарного стейку множина проміжних лідерів $\tilde{\Gamma}_l(k) = \{\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_N\}$, що відповідає базовому принципу DPoS:

$$\tilde{\Gamma}_l(k) = \mathrm{sort}_{\downarrow}\left(C(k), S\right)$$

З цієї впорядкованої множини обирається топ-N кандидатів, які складають попередній пул делегатів: $\tilde{\Gamma}_{del}(k) = \{\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_N\}$.

Наступним етапом буде фільтрація пулу лідерів за критерієм зв'язності. Для нівелювання ризиків, пов'язаних із просуванням у пул делегатів ізольованих, слабкозв'язаних або зловмисних, які отримали високий стейк s_j внаслідок випадкового вибору або змови, впроваджується жорсткий геометричний фільтр на основі матриці суміжності A_G.

Нехай f_l — очікувана (максимально допустима) кількість візантійських агентів серед лідерів на верхньому рівні консенсусу. Згідно з базовими положеннями теорії візантійської стійкості, для забезпечення працездатності підсистеми координації лідери повинні утворювати стійкий підграф із мінімальним ступенем зв'язності. Для кожного попередньо обраного делегата $\tilde{l}_j \in \tilde{\Gamma}_{del}(k)$ обчислюється його локальний степінь зв'язності (кількість прямих ліній зв'язку) з іншими вузлами всієї мережі, що математично еквівалентно підрахунку суми елементів відповідного рядка матриці A_G:

$$\deg(\tilde{l}_j) = \sum_{m=1}^{N} a_{jm}, \quad \text{де } a_{jm} \in A_G$$

Математичний критерій топологічної валідації: Обраний делегат вважається легітимним та здатним забезпечувати стійке керування групами тоді й тільки тоді, коли його степінь зв'язності задовольняє нерівність:

$$\deg(\tilde{l}_j) \geq 2f_l + 1$$

У випадку, якщо вузол є топологічно ізольованим або його зв'язність є недостатньою ($\deg(\tilde{l}_j) < 2f_l + 1$), запускається оператор дискваліфікації та заміни $\mathcal{D}$:

$$\Gamma_{del}(k) = \begin{cases} \tilde{\Gamma}_{del}(k) \setminus \{\tilde{l}_j\} \cup \{\tilde{l}_{N_{del}+1}\}, & \text{якщо } \deg(\tilde{l}_j) < 2f_l + 1 \\ \tilde{\Gamma}_{del}(k), & \text{якщо } \deg(\tilde{l}_j) \geq 2f_l + 1 \end{cases}$$

Таким чином, фінальна множина лідерів груп $\Gamma_l(k) = \{l_1, l_2, \ldots, l_{N_{del}}\}$ гарантовано складається виключно з високозв'язаних вузлів, що дозволяє знизити залежність від щільності глобального графа на етапі запуску внутрішньогрупового Raft-консенсусу.

Далі йде етап групування та виявлення зловмисних вузлів. Отже

після завершення фази топологічної фільтрації формується фінальна множина легітимних делегатів $\Gamma_l(k) = \{l_1, l_2, \ldots, l_{N_{del}}\}$. Решта вузлів системи складають множину підпорядкованих агентів (фоловерів):

$$\Gamma_{fl}(k) = V \setminus \Gamma_l(k)$$

Процес функціонування цього етапу поділяється на два підкроки: топологічний розподіл на групи та безперервний локальний моніторинг станів.

Кожен затверджений делегат $l_j \in \Gamma_l(k)$ стає керуючим центром локальної групи GP_j. Розподіл фоловерів $fl_i \in \Gamma_{fl}(k)$ по групах здійснюється на основі аналізу матриці суміжності A_G та вектора голосування. Якщо фоловер має зв'язки з декількома делегатами або взагалі не пов'язаний з ними безпосередньо, оператор групування аналізує множину його найближчих сусідин по графу N_i. Вузол приєднується до тієї групи, де зосереджена більшість його локальних сусідів.

Усередині кожної сформованої групи GP_j безпека комунікації між вузлами забезпечується модифікованим алгоритмом Raft з використанням криптографічних механізмів (цифрових підписів RSA/ECDSA). Процес комунікації в групах можно описати таким чином:

1) Генерація блоку (Heartbeat): Делегат l_j формує новий блок даних (або вектор керуючого впливу MAC) $M(k)$ та транслює його всередині своєї групи у вигляді повідомлення, підписаного власним закритим ключем $priv_j$:

$$\text{Heartbeat}_{l_j} = \langle M(k), \text{Sign}_{priv_j}(SHA256(M(k))) \rangle$$

2) Локальна верифікація: Кожен фоловер $fl_i \in GP_j$ приймає повідомлення та виконує перевірку за допомогою відкритого ключа делегата pub_j.

$$\text{Verify}_{pub_j}(\text{Heartbeat}_{l_j}) = \begin{cases} \text{True}, & \text{якщо чесний} \\ \text{False}, & \text{якщо зловмисний} \end{cases}$$

3) Якщо після верифікацію сигналу сердцебиття лідер виявився чесним, тоді фоловери синхронізують логи з делегатом. В іншому випадку фоловери ініціють процедуру ізоляції і перевиборів.

4) Після завершення циклу синхронізації фоловери надсилають повідомлення-відповідь зовнішньому клієнту. На основі цих повідомлень клієнт отримує точні ідентифікатори скомпрометованих делегатів для їх остаточної ізоляції.

Висновки до розділу 2

У даному розділі було описано механізм консенсусу PBFT з використанням додаткових криптографічних інструментів, які мають захистити мережу від зловмисних і візантійських вузлів. Окрім цього наступним кроком додано покращення PBFT його об'єднанням з Raft. Така модифікація усунула недолік в вимозі до комунікації між вузлами і покращила масштабованість даного підходу. З метою усунення головного недоліку класичного DPoS, який схильний до атак централізації та змов, ми використали цей механізм. Нові умови роботи протоколу мають запобігти захопленню мережі зловмисниками.

3 ІМІТАЦІЙНА МОДЕЛЬ ТА ЕКСПЕРИМЕНТИ

У даному розділі наведено математичну формалізацію та архітектурну логіку імітаційної моделі (симулятора), розробленої на мові програмування **GO** для оцінки стійкості протоколу консенсусу Delegated Proof of Stake до впливу розміру набору делегатів на продуктивність, як швидко голоси кількох "китів" централізують вибори, коли координована група отримує фактичний контроль, граничний рівень недоступності активних делегатів, чутливість до повільного розповсюдження блоків, наявністості критичних режимів і взаємодії кількох ризиків.

3.1 Опис формальної моделі

Імітаційна модель описується як дискретна стохастична система, що функціонує впродовж фіксованого горизонту моделювання, який вимірюється у загальній кількості раундів R та епох. Для генерації випадкових чисел був взятий генератор псевдовипадкових чисел пакета `math/rand` мови програмування Go, який використовує джерело ентропії з ядра операційної системи.

Для побудови математичної моделі запропонованого алгоритму спочатку формалізуємо структуру самої мережі. Оскільки розроблена мережа складається з N унікальних учасників, множину яких можна представити таким чином:

$$\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$$

В якій кожний учасник m_i де $i \in \overline{1,N}$ володіє якоюсь частиною токенів

$t_i \in \mathbb{R}^+$. Сумарний стейк мережі визначається як:

$$T_{total} = \sum_{i=1}^{N} t_i$$

Далі всі учасники, за розподілом капіталу та концентрація стейку (γ), поділяються на дві множини. До першої групи належать *звичайні токенхолдери*, серед яких рівномірно розподілено частку $(1 - \gamma) \cdot T_{total}$ та *"кити"*, 1% від всіх учасників, який тримає рівномірно розпоіделену частку розміром $\gamma \cdot T_{total}$

Після того як визначається множина учасників і між ними розподіляється сумарний стейк, відбувається відбір кандидатів в делегати, а після і відбір самих делегатів, на яких падає обов'язок створення нових блоків блокчейну. Весь процес генерації поділяється на епохи, після кожної епохи відбуваються перевибори. У кожній епосі фіксується множина C розміром у M кандидатів у делегати.

$$C = \{c_1, c_2, \ldots, c_M\}, \quad M \leq N$$

За результатами голосування з множини C відбирається підмножина активних валідаторів (делегатів) $D \subset C$ розміром $|D| = K$, які отримують право на створення блоків.

Наступним кроком є вибори делегатів, які відбуваються періодично, з кроком у E раундів. В даній моделі вважається, що кожен учасник $m_i \in \mathcal{M}$ може віддати один голос за одного кандидата $c_j \in C$. Вага голосу учасника V_i має стале значення і є еквівалентна його балансу $V_i = t_i$. Наступним кроком відбувається формування активного пулу валідаторів D, яке здійснюється шляхом сортування кандидатів за спаданням вагових коефіцієнтів та відбором топ-K елементів:

$$D = \text{top-}K\,(\text{sort\_desc}(C, W))$$

Параметр W, який показує сумарну вагу підтримки, обчислюється

незалежно для кожного кандидата c_j, тому W_j:

$$W_j = \sum_{i \in \text{Voters}(c_j)} V_i$$

Протягом періоду епохи E активні делегати $d_m \in D$ чергуються у фіксованих часових слотах для створення блоків. У кожному раунді запланований за розкладом делегат може перебувати в одному з трьох поведінкових станів, що моделюються як незалежні випадкові події. Перший стан відображає поведінку *чесного* (Honest/Online) учасника, який створює валідний блок. Ця подія відбувається з ймовірністю $p_{valid} = 1 - p_{off} - p_{inv}$. Другий стан відповідає за випадок, коли делегат *недоступний* (Offline), тобто він навмисно, або через збій апаратної частини чи DoS-атаку пропускає свій слот з ймовірністю p_{off}. Третій стан виділений для *зловмисних* учасників, які генерують некоректний або конфліктний блок з ймовірністю p_{inv}.

Зауваження. Для наближення моделі до умов реальних глобальних мереж вводиться параметр середньої мережевої затримки τ. В усіх експериментах ймовірність форку було взято 0.1.

Для аналізу результатів моделювання симулятор наприкінці кожної серії експериментів обчислює шість фундаментальних метрик стійкості:

Таблиця 3.1 – Розрахунок показників системи

Метрика	**Формула**	**Опис**
Availability	$\mathcal{A} = \frac{B_{\text{valid}}}{R}$	Частка успішно сформованих та узгоджених валідних блоків від загальної кількості слотів.
Latency	$\mathcal{L} = \frac{1}{B_{\text{valid}}} \sum_j \left(t^j_{\text{confirm}} - t^j_{\text{create}} \right)$	Середній час (або кількість слотів), необхідний для остаточного підтвердження (*finalization*) блоку.
Fork rate	$\mathcal{F} = \frac{B_{\text{fork}}}{B_{\text{total}}}$	Відношення кількості відкинутих мережею блоків до загальної кількості створених.
Capture condition	$\mathcal{P}_{\text{cap}} = \frac{E_{\text{cap}}}{E} > 0.5$	Ймовірність критичного режиму, за якого більше 51% епох було скомпрометовано картелем.
Concentration	$C = \frac{\sum_{j=1}^{3} V_j}{\sum_{i=1}^{K} V_i}$	Міра централізації голосів серед домінуючих делегатів.
Invalid share	$\mathcal{I} = \frac{B_{\text{invalid}}}{B_{\text{total}}}$	Реальна частка спроб формування некоректних або скомпрометованих блоків у мережі.

Зауваження. Для спрощення моделювання у розробленій моделі мережі метрика обчислюється через відношення загальної кількості слотів до кількості корисних (валідованих) блоків:

$$Latency = \frac{\text{totalRounds}}{\text{usefulBlocks}}$$

Оскільки модель містить значну кількість стохастичних параметрів — зокрема p_{off}, p_{inv}, а також випадковий розподіл початкових балансів учасників — результати окремого запуску можуть суттєво залежати від випадкових параметрів.

Кожна конфігурація експерименту, що відповідає фіксованому набору параметрів моделі, виконується серією незалежних запусків (не менше 100 ітерацій). Кожна ітерація використовує унікальне початкове значення генератора випадкових чисел (*seed*), що забезпечує незалежність кожного отриманого результату.

Фінальні значення метрик отримуються як середнє значення за результатами серії незалежних запусків моделі.

3.2 Тестування симуляційної моделі

Перед тим як перейти до моделювання специфічних експериментів, покажемо результати запуску симулятора консенсусу на різних вхідних даних, які відобразять вплив параметрів на оціночні метрики. Для цього буде зроблено таблицю з стандартними вхідними даними і таблицю з відповідними метриками (взято максимально наближені дані, без граничних сценаріїв).

Метою цього етапу є перевірка адекватності математичної моделі та її програмної реалізації, а також дослідження того, як природні фактори розподіленої мережі — такі як випадкова апатія виборців, технічні збої вузлів (перебування в офлайні) та випадкові затримки зв'язку (форки) впливають на загальну продуктивність та інші метрики стійкості.

Для отримання статистично достовірного результата та нівелювання фактору випадковості, базовий експеримент запускається 100 разів з однаковими стартовими параметрами, але різними випадковими значеннями, результатом цього експерименту буде усереднене значення всіх запусків.

Вхідні параметри базового експерименту

Усі конфігураційні параметри симуляції розділено на три логічні блоки: архітектура мережі, параметри поведінки вузлів та масштаб. Конкретні значення, передані у функцію тестування, наведено у таблиці 3.2.

Таблиця 3.2 – Конфігурація вхідних параметрів базового сценарію симуляції

Категорія	Параметр в коді	Значення	Опис
Архітектура мережі	`N`	2500	Загальна кількість вузлів-учасників системи.
	`M`	50	Кількість кандидатів, що висуваються на вибори.
	`K`	21	Кількість делегатів, які приймають участь у створенні блоків.
Часові рамки	`E`	2500	Тривалість епохи моделювання (кількість часових слотів).
Розподіл капіталу	`gamma`	0.40	Коефіцієнт концентрації стейку (відсоток активів, які належать "китам") .
Мережеві фактори	`pOff`	0.15 (15%)	Ймовірність тимчасового перебування делегата в офлайні (апатія).
	`pInv`	0.15 (15%)	Ймовірність генерації некоректного (невалідного) блоку.
	`tau`	0.1 (10%)	Частота виникнення мережевих затримок та форків.
Параметри атаки	`attackerPart`	0.20 (20%)	Частка зловмисників в мережі.
	`IsAttacker`	`true`	Активація логіки перевірки наявності атаки (моніторинг захоплення).
Масштаб тесту	`iterations`	100	Кількість повторних запусків для статистичного усереднення.

Таблиця 3.3 – Результати базового сценарію симуляції консенсусу DPoS (середні значення за 100 ітерацій)

Метрика виходу	**Значення**	**Науково-технічна інтерпретація показника**
Середня кількість створених блоків	2124.06	Загальна інтенсивність генерації блоків за епоху.
Середня кількість валідних блоків	2061.50	Кількість блоків, що успішно пройшли верифікацію та увійшли до основного ланцюга.
Середня кількість невалідних блоків	62.56	Блоки, які були відхилені комітетом через внутрішні помилки або зловмисні дії.
Доступність мережі (*Availability*)	82.46%	Відсоток слотів, у яких було успішно створено блоки .
Частка невалідних блоків (*Invalid Share*)	2.95%	Питома вага відхилених блоків відносно загальної кількості згенерованих.
Частота форків (*Fork Rate*)	10.03%	Частота виникнення тимчасових розгалужень ланцюга через мережеві затримки.
Середня затримка (*Latency*)	1.35 слотів	Середнія кількість слотів,необхідна для остаточного підтвердження блоку.
Рівень централізації (*Concentration*)	20.51%	Частка загального стейку мережі, яка реально зосереджена в руках діючого BFT-комітету.

Зробимо висновок стосовно точності та адекватності роботи симулятора. Система демонструє прогнозовану поведінку, яку потрібно порівняти(в перспективі) теоретичними положеннями розподілених систем:

1) Продуктивність та доступність (Liveness): Показник доступності мережі (Availability) на рівні 82.46% чітко відображає закладену у вхідних параметрах ймовірність перебування вузлів в офлайні ($p_{off} = 0.20$). Мережа успішно нівелює відсутність п'ятої частини валідаторів, не зупиняючи процес генерації, що підтверджує стійкість архітектури DPoS до динамічної зміни складу учасників.

2) Аналіз часових затримок (Latency): Метрика середньої затримки фіналізації становить 1.35 слотів. В ідеальній мережі цей показник дорівнював би 1.0 (один блок за один слот). Проте, через сукупний вплив факторів мережевої апатії (пропущені раунди), генерації невалідних блоків (2.95%) та виникнення форків (10.03%), час досягнення остаточного консенсусу (Time-to-Finality) збільшився на 35%. Це доводить, що розроблена математична модель використовує гарний

генератор псевдовипадкових чисел і коректно враховує деградацію пропускної здатності мережі під впливом наближених до реальності збоїв.

3) Вплив концентрації капіталу: За умови коефіцієнта нерівності капіталу $\gamma = 0.40$, метрика Concentration зафіксувала, що діючий комітет із $K = 21$ делегатів сукупно контролює 20.51% від усього стейку мережі. Хоча такий спосіб підрахунку централізації не є ідеальним, він все одно очікувано демонструє тенденції до децентралізації разом зі збільшенням делегатів. Звичайно в справжній моделі такий метод не зможе якісно оцінити справжню централізацію, бо в даній реалізації не має механізму запланованого голосування.

Одним із ключових архітектурних компромісів у системах консенсусу DPoS є вибір оптимальної кількості активних делегатів (K), які безпосередньо беруть участь у генерації блоків. Збільшення кількості активних делегатів повинно підвищитиє рівень децентралізації та стійкості мережі до захвату, проте може призвести до деградації продуктивності через зростання мережевих затримок, в цій моделі, через спрощення, кількість делегатів не впливає на продуктивність, тому розглянута буде тільки централізація.

Таблиця 3.4 – Вплив кількості делегатів на рівень централізації мережі DPoS

Кількість активних делегатів	**Рівень централізації**
7	47.47%
11	33.84%
21	20.52%
31	11.75%
51	9.23 %

Принцип підрахунку рівня централізації: для кількісного виміру рівня централізації у симуляторі використано модифікований індекс концентрації капіталу (активів), аналогічний класичному

коефіцієнту CR (Concentration Ratio). Алгоритм функцій розраховує питому вагу голосів, відданих за ТОП-3 найбагатших делегатів, відносно сумарного обсягу голосів (стейку) всього діючого пулу делегатів. Мережа буде вважатися захопленою певною групою, якщо рівень централізації буде більше 33%(умова BFT). В наступному абзаці буде виконан аналіз цих результатів.

При умові, що кількість делегатів $K = 7$, тоді рівень централізації досягає майже половину всієї мережі, що хоча і є очікуваною поведінкою, але такий результат(ну тут я просто не знаю що написати і як оцінити). При середньому випадку коли ($K = 21$), що є класичним варінтом для реальних DPoS-мереж (наприклад, EOS, Tron), вплив трійки лідерів зменьшується більше ніж удвічі — до 20.52%. В такому режимі захоплення мережі неможливе. Мінімальна концентрація досягається при кількості делегатів $K = 51$, в такому випадку плив окремих лідерів нівелюється, що суттєво підвищує стійкість архітектури до змов.

3.3 Експерименти

Перший експеримент: Вплив нерівності

У межах цього експерименту досліджено вплив нерівності розподілу капіталу (коефіцієнт Джині γ) на структуру виборчого процесу в системі DPoS. Головне питання дослідження: як накопичення монет у руках «китів» (Top 1% власників) впливає на реальну кількість унікальних користувачів, чиї голоси формують комітет делегатів.

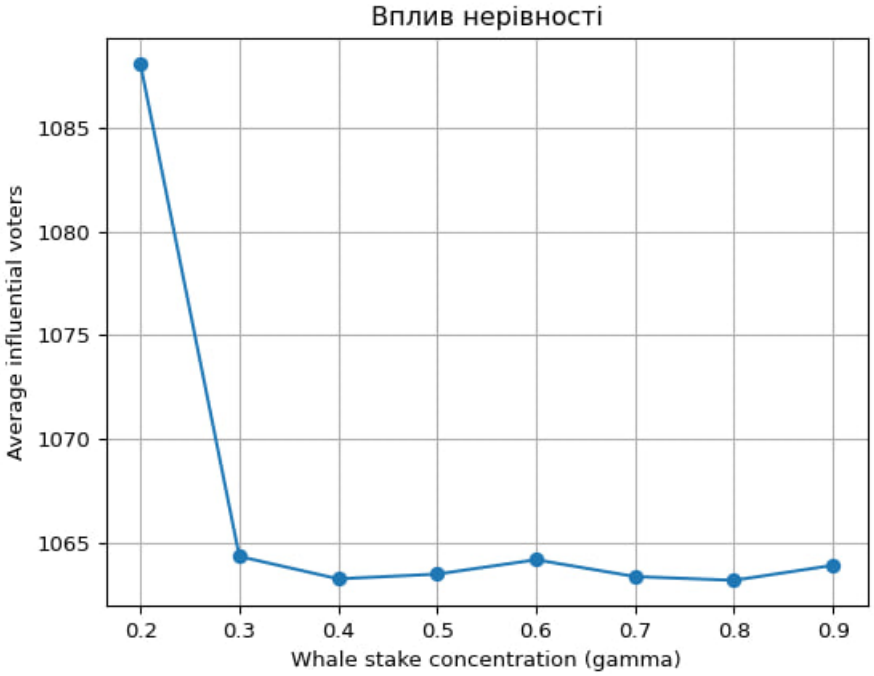

Рисунок 3.1 – Результат запуску першого експерименту

При мінімальній концентрації капіталу ($\gamma = 0.2$) кількість впливових виборців є максимальною і становить 1088 вузлів. Проте, при стрімкому зростанні всього на 10% сумарного стейку китів різко змінюється і залишається стабільним до критичного ($\gamma = 0.9$), цей показник практично не змінюється і стабілізується на рівні 1063–1064 вузлів.

З рисунку 3.1 можна зазначити, що чисельність голосуючих у DPoS є нечутливою до розподілу багатства в мережі(чого не може бути). Це доводить, що метрика кількості унікальних виборців не може слугувати надійним показником децентралізації системи без урахування фінансової ваги (стейку) цих голосів(а тут функція просто рахувала голоси віддані за кандидатів, які стали делегатами).

Другий експеримент: Стійкість до картельних змов (Sybil/Collusion Attack)

Досліджено вразливість системи DPoS до скоординованої атаки змови (Collusion/Sybil Attack). Сценарій передбачає, що зловмисник володіє певною часткою загального капіталу мережі (Attacker stake share)

і намагається використати її для максимального захоплення "крісел"делегатів. Головною метою зловмисника є досягнення двох критичних порогів. Першим є *34% делегатів* (поріг порушення безпеки / атака на Liveness), що дозволяє повністю блокувати фіналізацію блоків у BFT-системі. В другому *51% делегатів* (поріг повного контролю), що дозволяє одноосібно(або певною групою у зговорі) змінювати історію транзакцій та здійснювати подвійні витрати.

Моделювання проведено у двох конфігураціях поведінки зловмисника, результати яких представлено на рисунках 3.2 та 3.3. На першому відбувається сценарій «Випадковий розподіл», в якому в пул кандидатів, попадають випадкові учасники. На рис. 3.3 сценарій «Підкручений розподіл» (Optimized / Targeted Attack): Зловмисник застосовує оптимізовану стратегію атаки, коли в пул кандидатів одразу попадає K підконтрольних акаунтів. В обох випадках атакуючі розподіляють свій стейк між своїми для максимізації шансів проходження власного пулу кандидатів у делегати, всі інші голосують випадково.

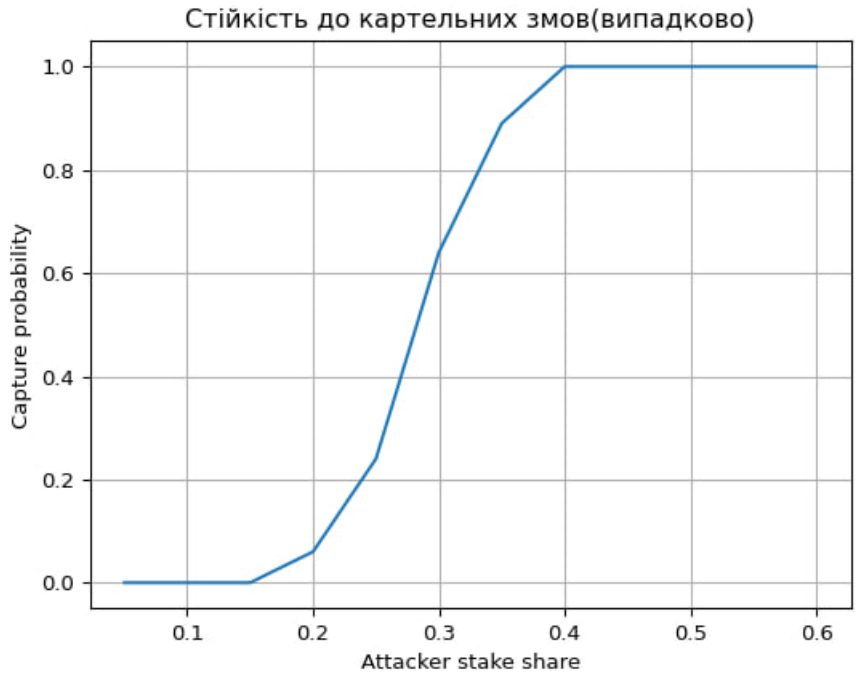

Рисунок 3.2 – Результат запуску другого експерименту

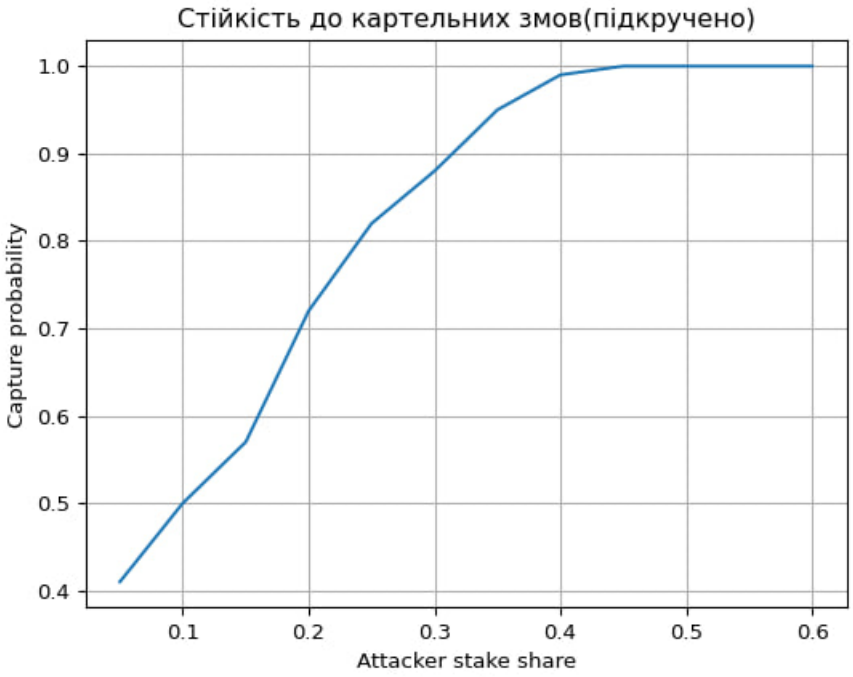

Рисунок 3.3 – Результат запуску ругого експерименту

На першому графіку, за умови випадкового голосування чесних учасників, мережа DPoS демонструє високий рівень стійкості на початкових етапах. При частці капіталу зловмисника до 15% ймовірність захоплення мережі дорівнює 0%. Ймовірність захвату починає стрімко зростати після досягнення 20% стейку, а гарантоване захоплення мережі ($P_{capture} = 1.0$) відбувається при контролі понад 40% монет системи.Такий характер кривої пояснюється тим, що за випадкового голосування капітал чесних токенхолдерів розпорошується між великою кількістю незалежних кандидатів. Водночас зловмисник діє скоординовано, фокусуючи свій стек на обмеженому пулі вузлів. Як тільки економічна частка зловмисника долає критичний поріг (20%), його централізовані голоси починають масово витісняти випадково обраних чесних делегатів.

Другий графік демонструє катастрофічне падіння стійкості консенсусу, якщо зловмисник використовує цілеспрямовану стратегію розподілу капіталу. Навіть при мінімальній частці стейку в 5%, ймовірність захвату мережі вже становить 41%. При досягненні зловмисником 15% капіталу (точка, яка в першому сценарії була

абсолютно безпечною), ймовірність захоплення контролю злітає до 57%, хоча повне захоплення системи ($P_{capture} = 1.0$) досягається вже при 45% стейку, навідміну від 40% в першому графіку.

З цього можна зробити висновок, що моделювання чітко доводить, класичний захист DPoS є ефективним лише проти неорганізованих зловмисників. У випадку скоординованої атаки із таргетованим розподілом голосів, безпековий поріг системи падає до критичних 5% стейку в руках зловмисників, що робить консенсус вкрай уразливим до формування олігархічних картелей.

Третій експеримент: Апатія виборців (Voter Apathy)

У межах цього експерименту досліджено вразливість системи до явища електоральної апатії (Voter Apathy). Сценарій моделює ситуацію, коли у голосуванні за делегатів бере участь лише фіксована частка від усіх легітимних користувачів мережі (Voter participation rate від 0.1 до 1.0).

Метою експерименту є перевірка гіпотези: чи спрощує пасивність чесних виборців завдання зловмиснику з обмеженим фіксованим капіталом (у цьому тесті частку зловмисника зафіксовано на низькому рівні — 10%) захопити контроль над мережею. На рисунку 3.4 відображен сценарій в зловмисник має 10% стейку, і в пул кандидатів зловмисники попадають випадково. На рисунку 3.5 зловмисник володіє тими ж 10% стейку, але в пул кандидатів вже попадає K підконтрольних акаунтів.

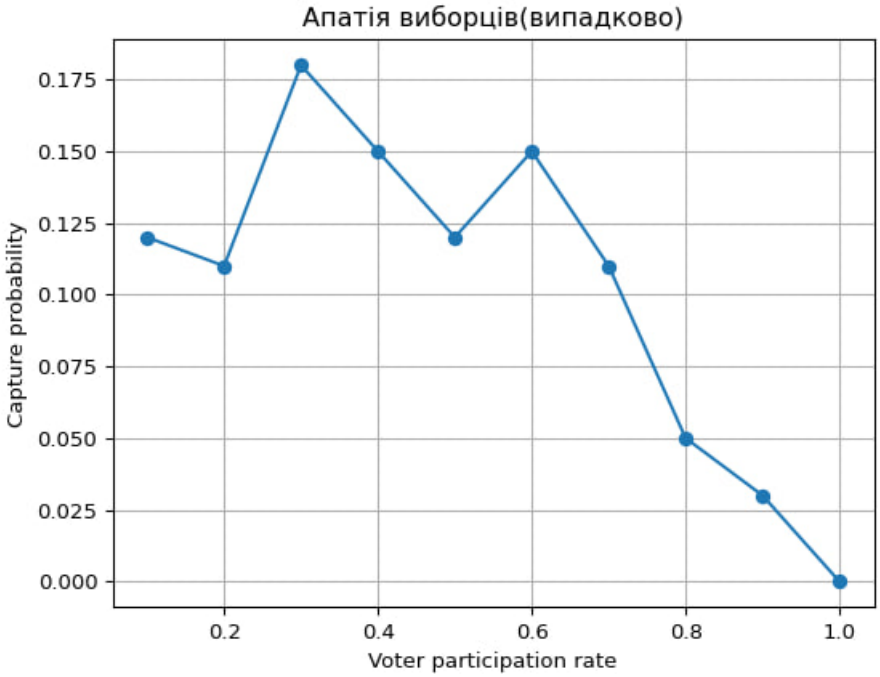

Рисунок 3.4 – Результат запуску третього експерименту

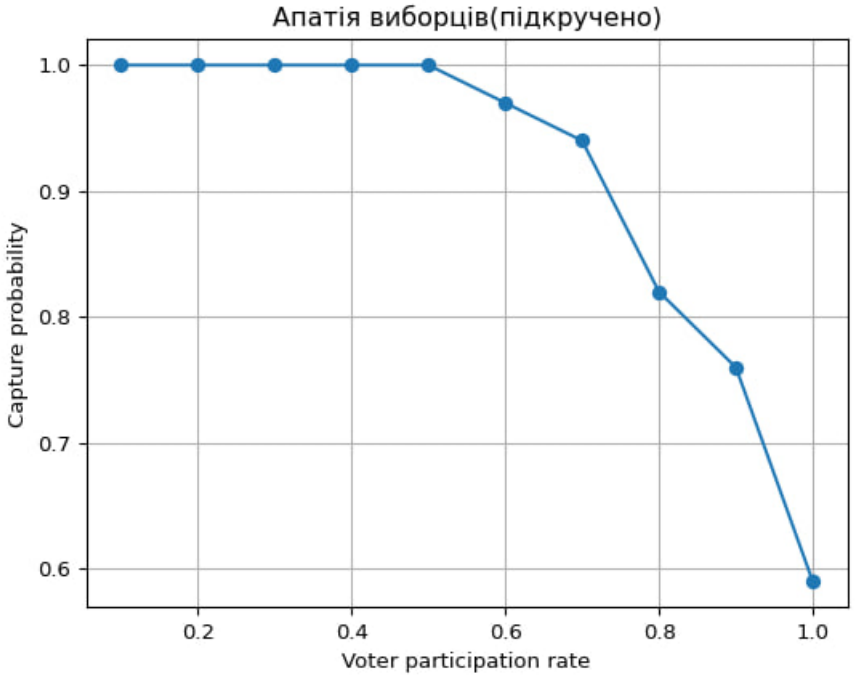

Рисунок 3.5 – Результат запуску третього експерименту

Як показує перший графік, якщо зловмисник діє неорганізовано, система DPoS загалом справляється з атакою навіть за умов низької

електоральної активності.Ймовірність захоплення мережі коливається на низькому рівні (11%–18%) і поводиться нестабільно (зигзагоподібно). За умови 100% явки чесних користувачів (Voter participation rate = 1.0), ймовірність захоплення падає до 0%. Помітно, що через випадковість голосування з обох сторін, неорганізовані 10% стейку зловмисника не здатні сформувати стійке ядро в делегатів.

Другий графік демонструє критичну вразливість консенсусу DPoS до картельних змов в при умові пасивності електорату.За умови, якщо у голосуванні бере участь 50% або менше чесних виборців, ймовірність захоплення мережі становить абсолютні 100% ($P_{capture} = 1.0$). Це означає, що за низької явки зловмисник із 10% капіталу отримує гарантований повний контроль над блокчейном.Навіть за умов ідеальної, максимальної участі чесних користувачів (Voter participation rate = 1.0), оптимізована атака все одно зберігає ймовірність захоплення на рівні 59%.

Висновки до розділу 3

Після проведення порівняння результатів алгоритмів формування пулу кандидатів (які використовувалися в випадковому і підкрученому варіанті) можна зробити важливий архітектурний висновок щодо безпеки консенсусу звичайного DPoS.

Стабільність мережі до картельних змов безпосередньо залежить від фільтрації учасників на етапі реєстрації кандидатів. Якщо система використовує повністю випадковий або пропорційний розподіл (як у першому варіанті), тоді зловмисник обмежений природною статистичною ймовірністю і не може повністю реалізувати фінансовий потенціал свого стейку, бо просто нема за кого голосувати.

Проте, якщо зловмиснику вдається обійти випадковий фільтр (наприклад, через створення масиву фейкових-акаунтів) і гарантовано виставити свій пул кандидатів на голосування (як у другому варіанті), то

безпека консенсусу повністю руйнується. У такому випадку навіть капітал у 10% стейку або електоральна апатія чесних користувачів призводять до гарантованого (100%) захоплення мережі. Це показує, що стадія відбору кандидатів у DPoS є дуже вразливим місцем і потребує додаткових механізмів захисту.

ВИСНОВКИ

У ході даної роботи був проведений аналіз опублікованих джерел за загальним группуванням атак на блокчейни і детально розглянута атака pool hooping. На основі цього аналізу як ціль дослідження були обрані протоколи PoS та BFT.

Далі в цілях визначення ефективного механізму протидії централізації був обраний PBFT, і досліджено його модифікацію, яка усунула деякі технічні недоліки класичного PBFT. Оскільки в представленому протоколі основна увага була сконцентрована саме на протидії зловмисним вузлам, завдяки чому ця проблема була вирішена. Оскількі DPoS немає таких механізмів, далі ми запропонували об'єднання його з Raft, що має протидіяти картельним атакам.

Підтверджено, що BFT протоколи справді застосовуються як частина протоколів іншого типу, тому дуже часто BFT розглядається разом з протоколами PoS і є частиною DPoS. Для дослідження стійкості протоколів DPoS розроблена імітаційна модель, реалізація якої дозволила зробити наступні висновки.

Моделювання базового сценарію показало, що всі теоретичні припущення точно моделюються в метриці симуляції. Встановлено, що за умови природних збоїв мережі середня затримка фіналізації консенсусу (Latency) прогнозовано деградує з ідеального 1.0 до 1.35–1.36 слотів.

Перевірено вплив зміни кількості делегатів на рівень централізації мережі. Перевірка показала, що збільшення кількості активних делегатів з 7 до 51 дозволяє знизити рівень централізації влади у мережі з 47.47% до 9.23%. Це демонструє, що збільшення кількості делегатів є ефективним інструментом боротьби з захопленням мережі невеликим пулом валідаторів.

Була відмічена висока аразливість до картельних змов (Sybil/Collusion Attack). Порівняльний аналіз безпеки виявив, що

консенсус базовий варіант DPoS є стійким лише до неорганізованих (випадкових) атак, де для захоплення мережі зловмиснику потрібно контролювати понад 40% загального стейку. Проте, у випадку застосування оптимізованої стратегії таргетованого розподілу голосів (Сивіллова атака з підкрученою селекцією кандидатів), безпековий поріг системи руйнується: ймовірність компрометації комітету становить 41% вже при наявності у зловмисника всього 5% капіталу.

Дослідження фактору електоральної апатії (Voter Apathy) показало, що пасивність чесних користувачів є критичним каталізатором для успіху зловмисних картелів. При умові, що у голосуванні приймають участь 50% або менше валідаторів, зловмисник із фіксованим капіталом у 10% отримує абсолютну 100% ймовірність гарантованого захоплення влади в мережі.

Репозиторій дипломного проєкту на GitHub

ПЕРЕЛІК ПОСИЛАНЬ

[1] X. Koutsoukos та S. Sundaram H. J. LeBlanc H. Zhang. *Resilient Asymptotic Consensus in Robust Networks.* Т. 31. 4. 2013, 766 – 781. URL: https://ieeexplore.ieee.org/document/6481629.

[2] Jason Teutsch та Prateek Saxena. Loi Luu Yaron Velner. *SmartPool: Practical Decentralized Pooled Mining.* 2017. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/luu.

[3] David R. Shahriar B. *A review of attacks and security approaches in open multi-agent systems.* 2012. URL: https://www.researchgate.net/publication/257513023_A_review_of_attacks_and_security_approaches_in_open_multi-agent_systems.

[4] G. Tobias та ін. *A Structured Overview of Attacks on Blockchain Systems.* 2021. URL: https://www.researchgate.net/publication/352733786_A_Structured_Overview_of_Attacks_on_Blockchain_Systems.

[5] S. Sushil Kumar та ін. *Smart Contract-Based Pool Hopping Attack Prevention for Blockchain Networks.* 2019. URL: https://www.researchgate.net/publication/334586786_Smart_Contract-Based_Pool_Hopping_Attack_Prevention_for_Blockchain_Networks.

[6] Y. Fan та ін. *Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism.* 2019.

[7] Z. Jing та ін. *Secure Consensus Control on Multi-Agent Systems Based on Improved PBFT and Raft Blockchain Consensus Algorithms.* Т. 12. 7. 2025. URL: https://ieeexplore.ieee.org/document/11062708.