НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК _____

До захисту допущено
Завідувач кафедри
_____ Дмитро ЛАНДЕ
«_____»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека та захист інформації»

на тему:

Забезпечення конфіденційності в хмарному середовищі на основі технології блокчейн

Виконав (-ла): здобувач вищої освіти **II** курсу, групи ФБ-41мп
Африканський Олег Максимович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: Доцент кафедри ІБ, доцент, к.т.н.
Гальчинський Леонід Юрійович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент: Доцент кафедри КбЗІ, к.т.н.
Пархоменко Іван Іванович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

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

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський)
Спеціальність 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Дмитро ЛАНДЕ
«____»_____2025 р.

ЗАВДАННЯ
на магістерську дисертацію здобувача вищої освіти
Африканського Олега Максимовича
(прізвище, ім’я, по батькові)

1. Тема роботи: Забезпечення конфіденційності в хмарному середовищі на основі технології блокчейн.

керівник роботи: Гальчинський Леонід Юрійович, к.т.н., доцент.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом №4797-с по університету від «04» листопада 2025 р.

2. Термін подання здобувачем вищої освіти роботи «16» грудня 2025 р.

3. Вихідні дані до роботи: Дані щодо вторгнень, що націлені на хмарні обчислювальні системи в цілому та блокчейн-додатки зокрема.

4. Зміст роботи: Досліджено архітектуру та загрози безпеці хмарних платформ в умовах масштабних обсягів даних . Спроектовано децентралізовану MAS для захищеного зберігання та аудиту логів . Розроблено алгоритм Target-based дедуплікації на основі Дерев Меркла для оптимізації витрат. Запропоновано гібридну модель «хмара–блокчейн» та реалізовано протокол публічного аудиту цілісності зі збереженням конфіденційності .

5. Перелік ілюстративного матеріалу: презентація до захисту дипломної роботи – 12 слайдів

6. Дата видачі завдання «23» вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Визначення напряму МД	05.08.2025 - 15.08.2025	Виконано
2	Визначення теми МД	15.08.2025 - 25.08.2025	Виконано
3	Збір та аналіз літературних джерел	25.08.2025 - 10.09.2025	Виконано
4	Дослідження методів дедуплікації та технології блокчейн для захисту хмарних сховищ	10.09.2025 - 23.09.2025	Виконано
5	Порівняльний аналіз архітектурних підходів до забезпечення цілісності та конфіденційності	23.09.2025 - 10.10.2025	Виконано
6	Проектування мультиагентної системи та алгоритмів децентралізованого аудиту логів	10.10.2025 - 29.10.2025	Виконано
7	Аналіз результатів, написання висновків	29.10.2025 - 10.11.2025	Виконано
8	Проходження та захист переддипломної практики	01.09.2025 - 22.10.2025	Виконано
9	Передзахист магістерської дисертації	16.12.2025	виконано

Здобувач вищої освіти _____ Олег АФРИКАНСЬКИЙ
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

Керівник роботи _____ Леонід ГАЛЬЧИНСЬКИ
(підпис) (**Власне ім'я, ПРІЗВИЩЕ**)

РЕФЕРАТ

Обсяг роботи 68 сторінок, 12 ілюстрації, 2 таблиць, 40 джерел літератури.

Об’єкт дослідження: процеси забезпечення конфіденційності, цілісності та доступності даних моніторингу в розподілених хмарних системах.

Предмет дослідження: методи та архітектурні рішення для захищеного зберігання журналів подій з використанням технології блокчейн, дедуплікації даних та мультиагентних систем.

Мета дослідження: Розробка гібридної архітектури системи аудиту, що забезпечує гарантовану незмінність та конфіденційність даних моніторингу в умовах високого навантаження та недовіреного хмарного середовища.

Методи дослідження: Системний аналіз, порівняльний аналіз архітектурних підходів, математичне моделювання, криптографічні методи захисту, метод структурного проєктування.

Отримані результати: Проведено аналіз впливу характеристик великих даних на системи безпеки та обґрунтовано неефективність традиційного логування під час DDoS-атак. Реалізовано метод дедуплікації на стороні серверу для оптимізації роботи систем безпеки. Розроблено гібридну архітектуру сховища, де об’ємні дані зберігаються у хмарі, а хеші — у блокчейні, що гарантує математичну незмінність історії подій. Спроектовано мультиагентну систему для підвищення відмовостійкості.

Результати роботи були представлені на Міжнародній мультидисциплінарній науковій інтернет-конференції Світ наукових досліджень. Випуск 47 у доповіді на тему «Порівняльний аналіз архітектурних підходів до забезпечення цілісності та конфіденційності даних у хмарних середовищах».

Ключові слова: хмарні обчислення, блокчейн, дедуплікація даних, мультиагентна система, дерево Меркла.

ABSTRACT

The volume of work is 68 pages, 12 illustrations, 2 tables, 40 references.

Research object: processes for ensuring the confidentiality, integrity, and availability of monitoring data in distributed cloud systems.

Research subject: methods and architectural solutions for secure storage of event logs using blockchain technology, data deduplication, and multi-agent systems.

Research objective: Development of a hybrid audit system architecture that ensures guaranteed immutability and confidentiality of monitoring data in high-load and untrusted cloud environments.

Research methods: System analysis, comparative analysis of architectural approaches, mathematical modeling, cryptographic protection methods, structural design method.

Results obtained: The impact of big data characteristics on security systems was analyzed, and the ineffectiveness of traditional logging during DDoS attacks was substantiated. A server-side deduplication method was implemented to optimize the performance of security systems. A hybrid storage architecture was developed, where volumetric data is stored in the cloud and hashes are stored in the blockchain, which guarantees the mathematical immutability of the event history. A multi-agent system was designed to improve fault tolerance.

The results of the work were presented at the International Multidisciplinary Scientific Internet Conference World of Scientific Research. Issue 47 in a report on «Comparative analysis of architectural approaches to ensuring data integrity and confidentiality in cloud environments.»

Keywords: cloud computing, blockchain, data deduplication, multi-agent system, Merkle tree.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....8

ВСТУП.....9

1 ТЕОРЕТИЧНА ЧАСТИНА.....11

1.1. Історія хмарних обчислювальних систем.....11

1.2. Концепції та архітектура хмарних обчислень.....16

1.3. Характеристики та моделі розгортання хмарних обчислень.....18

1.4. Віртуалізація та міграція в реальному часі.....19

1.5. Контейнеризація для полегшення віртуалізації.....21

1.6. Кібератаки та таксономії в хмарних обчисленнях.....22

1.7. Потоки хмарної безпеки.....24

1.8. Виявлення вторгнень у хмарні системи.....26

Висновки до розділу 1.....28

2 АНАЛІЗ ПРОБЛЕМАТИКИ ТА МЕТОДІВ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОГО ЗБЕРІГАННЯ ДАНИХ МОНІТОРИНГУ 30

2.1. Аналіз впливу характеристик «Великих даних» на системи безпеки хмарних середовищ 30

2.2. Дослідження надлишковості журналів подій та порівняльний аналіз методів дедуплікації 33

2.3. Аналіз загроз конфіденційності в системах з дедуплікацією 36

2.4. Теоретичні засади використання технології блокчейн для забезпечення цілісності даних 38

2.5. Аналіз архітектурних підходів до забезпечення цілісності: Централізовані БД проти Блокчейну 43

2.6. Обґрунтування використання мультиагентного підходу до децентралізації процесів захисту 47

Висновки до розділу 2.....49

3 ПРОЄКТУВАННЯ СИСТЕМИ БЕЗПЕЧНОГО АУДИТУ.....51

3.1. Розробка загальної архітектури мультиагентної системи 51

3.2. Математичне та алгоритмічне забезпечення дедуплікації на стороні серверу 54

3.3. Протокол взаємодії агентів при збереженні та дедуплікації даних 55

3.4. Реалізація підсистеми публічного аудиту та верифікації цілісності даних 57

3.5. Аналіз безпеки та ефективності запропонованої архітектури 60

ВИСНОВКИ.....63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....65

ПЕРЕЛІК СКОРОЧЕНЬ

IDS - Intrusion Detection Systems (Системи виявлення вторгнень)

IoT – Internet of Things (Інтернет речей)

NIDS – Network Intrusion Detection Systems (Системи виявлення мережевих вторгнень)

MAS – Multi-agent system (Мультиагентна система)

TPA – Third Party Auditor (Сторонній аудитор)

CSP – Cloud Service Provider (Хмарний провайдер)

SHA-256 – Secure Hash Algorithm 256-bit

FIPA-ACL – Foundation for Intelligent Physical Agents - Agent Communication Language

REST API – Representational State Transfer Application Programming Interface

ВСТУП

Актуальність теми: Стрімка міграція корпоративних інфраструктур у хмари та перехід до мікросервісної архітектури призвели до фундаментальних змін у природі даних безпеки. У сучасних центрах обробки даних обсяги логів, що генеруються системами виявлення вторгнень, досягають терабайтів щодня, підпадаючи під характеристики «Великих даних».

Критичною проблемою стає вразливість самих систем моніторингу під час кібератак. Емпіричні дані свідчать, що під час DDoS-атак або масованого сканування швидкість генерації записів зростає експоненційно. Традиційні централізовані системи аудиту мають фіксовану пропускну здатність, що призводить до черг на запис і втрати критично важливих доказів саме в момент атаки. Крім того, зберігання повної історії «сирих» логів є економічно неефективним через високу надлишковість даних.

Окремим викликом є проблема довіри у хмарній моделі спільної відповідальності. Клієнт не контролює фізичні носії інформації, тому існує ризик, що адміністратор хмарного провайдера або зловмисник із високими правами може непомітно модифікувати або видалити журнали подій, щоб приховати сліди злочину. Це робить логи ненадійним доказом для криміналістичного аналізу.

Таким чином, актуальність дослідження зумовлена необхідністю розв'язання протиріччя між вимогою зберігати повну історію подій безпеки та технічними обмеженнями пропускної здатності каналів і недовірою до середовища зберігання. Вирішення цієї проблеми лежить у площині інтеграції методів дедуплікації даних, технології блокчейн та мультиагентних систем.

Мета і завдання дослідження: Метою роботи є розробка гібридної архітектури системи аудиту, що забезпечує гарантовану незмінність, конфіденційність та доступність даних моніторингу в умовах високого навантаження та недовіреного хмарного середовища.

Для досягнення мети поставлено такі завдання:

1. Завдання: Провести аналіз сучасних підходів до організації хмарних сховищ, дослідити проблеми надлишковості даних, загрози збереження та конфіденційності даних.

2. Завдання: Дослідити теоретичні засади технології блокчейн та мультиагентних систем (MAS) як інструментів для забезпечення довіри, автоматизації та конфіденційності даних в розподілених системах.

3. Завдання: Розробити алгоритми взаємодії компонентів системи для забезпечення дедуплікації та публічного аудиту цілісності.

Об’єкт дослідження: процеси забезпечення конфіденційності, цілісності та доступності даних моніторингу в розподілених хмарних обчислювальних системах.

Предмет дослідження: методи та архітектурні рішення для захищеного зберігання журналів подій з використанням технології блокчейн, дедуплікації даних та мультиагентних систем.

Методи дослідження: У роботі використано: системний аналіз, порівняльний аналіз, криптографічні методи, математичне моделювання, метод структурного проєктування.

Практичне значення одержаних результатів: Розроблена архітектура дозволяє створити відмовостійку систему зберігання логів, що знижує витрати на хмарне сховище завдяки дедуплікації та забезпечує юридичну значущість доказів інцидентів завдяки блокчейну. Запропоноване рішення може бути імплементоване на базі платформи Microsoft Azure.

Апробація результатів дипломної роботи: Основні положення роботи доповідались на наукових семінарах та конференціях молодих вчених, а також були апробовані під час проходження переддипломної практики.

1 ТЕОРЕТИЧНА ЧАСТИНА

У цьому розділі розглянемо теоретичні засади хмарних обчислювальних систем та блокчейн-додатки в таких системах.

1.1. Історія хмарних обчислювальних систем

Історія хмарних обчислень (Рисунок 1.1) починається з концепції надання обчислювальних ресурсів як комунальної послуги, вперше висловленої Джоном Маккарті в 1960-х роках [1]. Розвиток хмарних технологій тісно пов'язаний з розвитком мережі ARPANET та подальшим зростанням інтернету. Важливими етапами стали розробка віртуалізації, що дозволила ефективно використовувати обчислювальні ресурси, та поява компаній, що почали пропонувати хмарні послуги.

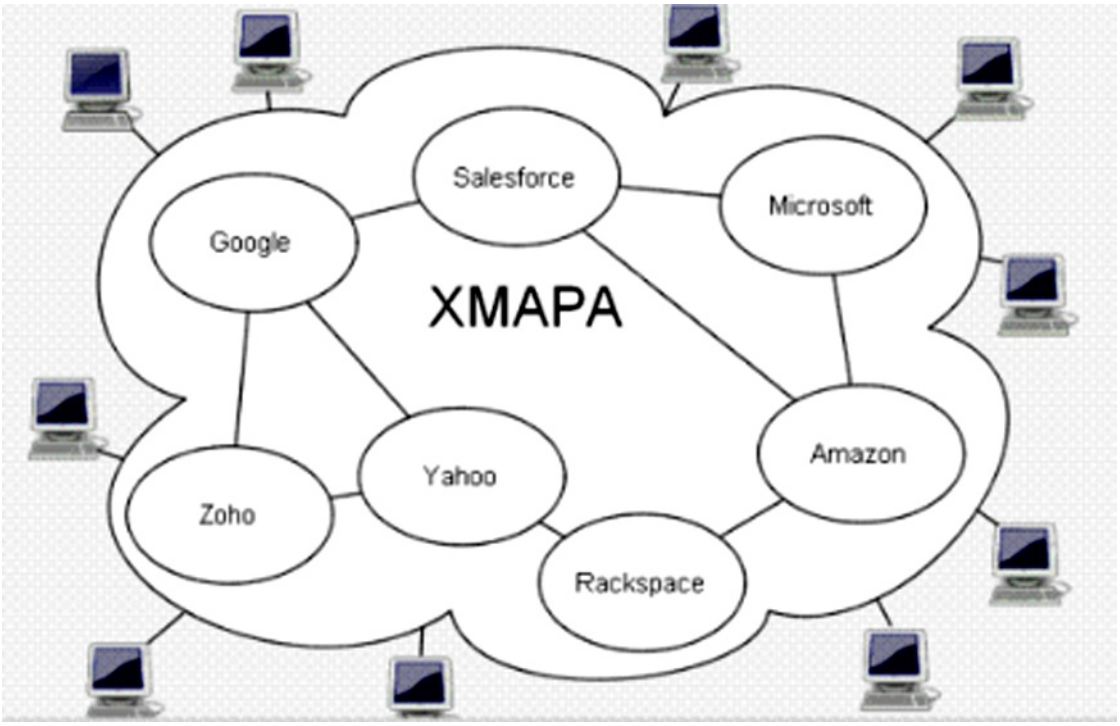

Рисунок 1.1 - Діаграма, яка ілюструє організацію хмарних обчислень

У 1970-х роках минулого століття компанія IBM розробляє операційну систему VM, яка дозволяє запускати декілька віртуальних машин на одному фізичному сервері.

У 1996 році в офісі Compaq Computer невелика група технічних керівників прогнозувала майбутнє інтернет-бізнесу та вперше запропонувала назву «хмарні обчислення». Вони «пророкували», все програмне забезпечення для бізнесу переміститься в Інтернет, а також те, що споживчі файлові сховища теж переїдуть у хмару.

Шон О'Салліван два роки створював свій стартап NetCentric, метою якого було запхати софт в інтернет (software inside the internet). Compaq проінвестували 5 млн. доларів і виділили свого директора з маркетингу Джорджа Фавалоро, щоб написати нормальний бізнес-план.

Компанія NetCentric зіткнулася з труднощами та була продана у 2001 році. Для Compaq це стало початком 2-мільярдного бізнесу із продажу серверів інтернет-провайдерам.

Ще до появи терміну «хмарні обчислення» О'Салліван заснував у 1986 році MapInfo (один з перших location-based services) і за 7 років набрав більше мільйона користувачів та капіталізацію 200 млн доларів. А в 2011 році О'Салліван намагався зробити свій Uber. А тепер знов повернемось до історії хмарних обчислень.

У 1999 році з'являється CRM-система Salesforce.com, яка надавалася за підпискою у вигляді веб-сайту [8].

Чотири товариші Marc Benioff (колишній топ-менеджер з Oracle), Parker Harris, Frank Dominguez, Dave Moellenhoff зняли апартаменти з однією спальнею і запустили cloud-based компанію.

Усі четверо розробляють прототип, носять сорочки з гавайськими принтами, призначають собаку Коа на посаду Chief Love Officer, і постійно запитують фідбек від користувачів, які працюють лише над тим, що вважають важливим і необхідним, щоб зробити це швидко, просто і правильно з першого разу.

До кінця 1999 року компанія зросла до 40 співробітників і до офісу 750 кв метрів.

При розробці Merchant.com, платформи для електронної комерції як послуги для сторонніх роздрібних продавців, програмісти Amazon зіткнулися з купою проблем і вирішили покращити життя і собі та решту - зробити зручний сервіс для автономних інженерних груп.

У липні 2002 року Amazon.com Web Services (Рисунок 1.2) під керуванням Коліна Брайара запустила свої перші веб-сервіси, відкривши платформу Amazon.com для всіх розробників. До 2004 року на його основі було створено понад сотню додатків. Цей несподіваний інтерес розробників застав Amazon зненацька і переконав їх у тому, що розробники «прагнуть більшого».

Рисунок 1.2 - Структура AWS

Amazon Elastic Compute Cloud (EC2) дозволяє користувачам орендувати віртуальні комп'ютери для запуску власних комп'ютерних програм. EC2 заохочує розгортання програм, що масштабується, надаючи веб-сервіс, через який користувач може завантажити образ машини Amazon (AMI) для налаштування віртуальної машини, яку Amazon називає «екземпляром»(instance), що містить будь-яке бажане програмне забезпечення. Користувач може створювати, запускати та завершувати екземпляри серверів у міру потреби, платячи за активні сервери посекундно — звідси й термін «еластичний». EC2 надає користувачам контроль над географічним розташуванням екземплярів, що дозволяє оптимізувати затримки та забезпечити високий рівень надмірності.

9 серпня 2006 року Ерік Шмід, CEO Google, виступив на конференції та оголосив на весь світ термін «cloud computing»

Компанія Timeweb розпочала роботу всього з одним сервером, двома співробітниками та єдиною послугою – звичайним віртуальним хостингом.

У 2008 році OpenNebula NASA в рамках проекту, що фінансується Європейською комісією RESERVOIR, стала першим програмним забезпеченням із відкритим вихідним кодом для розгортання приватних та гібридних хмар.

У квітні 2008 вийшла бета-версія Google App Engine.

Це платформа хмарних обчислень як сервіс для розробки та розміщення веб-застосунків у центрах обробки даних, керованих Google Програми ізольовані і працюють на декількох серверах.

До середини 2008 р. компанія Gartner побачила в хмарних обчисленнях можливість «формувати відносини між споживачами IT-послуг, тими, хто використовує IT-послуги, і тими, хто їх продає», і зауважила, що «організації переходять від апаратних та програмних активів, що належать компаніям, до моделі на основі послуг на основі послуг призведе до різкого зростання IT-продуктів у деяких областях та значного скорочення в інших областях» [2,4].

У 2008 році Національний науковий фонд США запустив програму Cluster Exploratory для фінансування академічних досліджень із використанням кластерної технології Google-IBM для аналізу величезних обсягів даних.

У 2009 році виходять програми Google Apps [9].

У 2009 році уряд Франції оголосив про проект «Андромеда» щодо створення «суверенної хмари» або національних хмарних обчислень, на який уряд витратить 285 мільйонів євро. Ініціатива зазнала невдачі, і 1 лютого 2020 року Cloudwatt було закрито.

У травні 2012 року було випущено попередню версію Google Compute Engine, а в грудні 2013 року вона стала загальнодоступною.

Це компонент інфраструктури як послуги (IaaS) платформи Google Cloud, побудований на глобальній інфраструктурі, де працює пошукова система Google, Gmail, YouTube та інші служби. Google Compute Engine дозволяє користувачам запускати віртуальні машини (VM) на вимогу. VM можна запускати зі стандартних образів або образів користувача.

У 2019 році Linux була найпоширенішою ОС, яка використовується в Microsoft Azure.

У грудні 2019 року Amazon анонсувала AWS Outposts, повністю керований сервіс, що розширює інфраструктуру AWS, сервіси AWS, API та інструменти практично для будь-якого клієнтського центру обробки даних, простору для спільного розміщення або локального об'єкта для дійсно узгодженого гібридного досвіду.

В наші часи хмарні обчислення активно використовуються в Києві як для індивідуальних користувачів, так і для бізнесу. Багато компаній переносять свої IT-інфраструктури в хмару, щоб оптимізувати витрати та підвищити ефективність роботи [40]. Крім того, хмарні сервіси активно використовуються для розробки та тестування програмного забезпечення.

1.2. Концепції та архітектура хмарних обчислень

В цьому розділі розглянемо концепцію та архітектуру хмарних обчислень.

Концепція хмарних обчислень

В процесі хмарних обчислень застосовуються наступні концепції [6,12].

Доступ на вимогу:

Користувачі можуть отримувати доступ до необхідного обсягу ресурсів через Інтернет без необхідності придбання та обслуговування власного обладнання.

Масштабованість:

Хмарні ресурси можна легко масштабувати, збільшуючи чи зменшуючи в залежності від потреб.

Економія:

Модель «оплата в міру використання» дозволяє оптимізувати витрати на ІТ.

Гнучкість та швидкість:

Хмарні обчислення забезпечують швидку розробку та розгортання додатків, а також гнучкість у виборі необхідних ресурсів [11].

Управління ресурсами:

Постачальники хмарних послуг забезпечують управління, обслуговування та безпеку інфраструктури.

Архітектура хмарних обчислень

Архітектура хмарних обчислень має наступний вигляд (Рисунок 1.3) [7].

Рисунок 1.2 - Архітектура хмарних обчислень

Шари:

Хмарна архітектура включає різні шари, такі як інфраструктура як послуга (IaaS), платформа як послуга (PaaS) і програмне забезпечення як послуга (SaaS).

Віртуалізація:

Віртуалізація відіграє ключову роль, дозволяючи створювати віртуальні машини на фізичних серверах, підвищуючи ефективність використання ресурсів.

Мережа:

Хмарна мережа забезпечує зв'язок між різними компонентами хмарної інфраструктури та доступом користувачів до ресурсів.

Зберігання даних:

Хмарні послуги надають різні варіанти зберігання даних, включаючи зберігання об'єктів, блокове зберігання та зберігання баз даних.

Безпека:

Хмарна архітектура включає різні механізми безпеки для захисту даних і ресурсів.

Управління:

Існують різні інструменти та платформи для управління хмарними ресурсами, включаючи оркестрацію контейнерів, автоматизацію розгортання та моніторинг.

Загалом, хмарні обчислення та архітектура хмарних обчислень є взаємопов'язаними концепціями, які дозволяють організаціям використовувати переваги гнучкої, масштабованої та економічної інфраструктури для вирішення своїх обчислювальних завдань.

1.3. Характеристики та моделі розгортання хмарних обчислень

Хмарні обчислення характеризуються кількома основними моделями розгортання та обслуговування. Моделі розгортання включають приватну, публічну, громадську та гібридну хмару, тоді як сервісні моделі - IaaS, PaaS та SaaS. Ці моделі визначають, яким чином надаються та використовуються ресурси хмари [5].

Моделі розгортання хмарних обчислень (Рисунок 1.4):

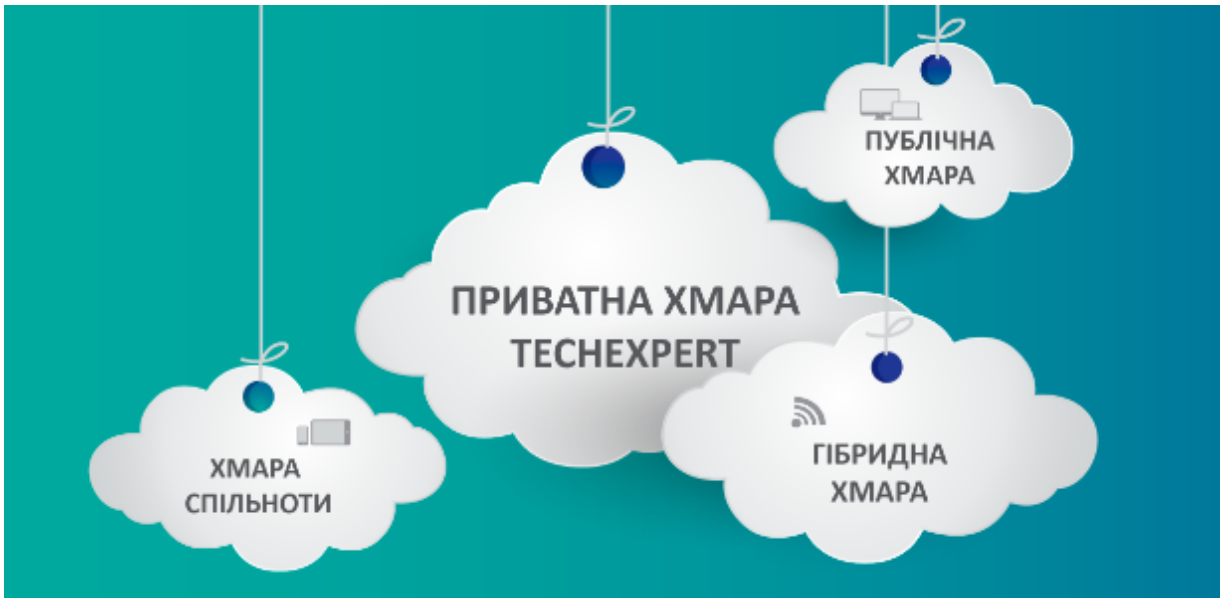

Рисунок 1.3 - Моделі розгортання хмарних обчислень

1. Публічна хмара (Public Cloud):

Інфраструктура надається провайдером і доступна для використання широкому загалу через інтернет. Ця модель зазвичай характеризується високою масштабованістю та доступністю, але може пропонувати менший рівень контролю. Приклади публічних хмарних послуг: Amazon Web Services, Google Cloud Platform, Microsoft Azure.

2. Приватна хмара (Private Cloud):

Інфраструктура призначена для використання однією організацією, часто розгорнута у власному дата-центрі або у провайдера, але з виділеними ресурсами. Приватна хмара забезпечує більш високий рівень безпеки та контролю, але може бути менш масштабованою та дорожчою [10].

3. Гібридна хмара (Hybrid Cloud):

Поєднує в собі елементи публічної та приватної хмари, дозволяючи організаціям використовувати ресурси обох. Гібридні хмари надають можливість зберігати конфіденційні дані у приватній хмарі, а менш критичні – у публічній, що дозволяє оптимізувати витрати та підвищити гнучкість.

Крім моделей розгортання, існують різні моделі надання хмарних послуг, такі як IaaS (інфраструктура як послуга), PaaS (платформа як послуга) і SaaS (програмне забезпечення як послуга). Ці моделі визначають рівень послуг і рівень контролю, що залишається у користувача.

1.4. Віртуалізація та міграція в реальному часі

Віртуалізація та міграція в режимі реального часу - це технології, що дозволяють переміщати працюючі віртуальні машини між фізичними серверами без зупинки роботи та переривання обслуговування. Віртуалізація дозволяє створювати та використовувати кілька віртуальних серверів на одному фізичному, а міграція в реальному часі забезпечує безперервність роботи при переміщенні цих віртуальних машин [10].

Віртуалізація (Рисунок 1.5) - це технологія, яка дозволяє створювати віртуальні версії обчислювальних ресурсів, таких як сервери, сховища даних та мережі на базі фізичної інфраструктури. Це досягається шляхом використання програмного забезпечення, званого гіпервізором, яке дозволяє декільком операційним системам та програмам працювати на одному фізичному сервері одночасно, розділяючи ресурси цього сервера.

Рисунок 1.4 - Віртуалізація та традиційна архітектура

Міграція в режимі реального часу (Live Migration) – це процес переміщення працюючої віртуальної машини з одного фізичного сервера на інший без зупинки та переривання обслуговування цієї віртуальної машини. Це досягається шляхом переміщення стану віртуальної машини (включаючи пам'ять, процесорні цикли та мережеві з'єднання) на цільовий сервер, забезпечуючи безперервну доступність сервісів, що працюють на цій віртуальній машині.

Основні переваги віртуалізації та міграції в режимі реального часу:

Підвищена доступність:

Міграція в реальному часі забезпечує безперервність роботи програм та сервісів, навіть при плановому обслуговуванні чи аварійних ситуаціях на фізичних серверах.

Оптимізація ресурсів:

Віртуалізація дозволяє більш ефективно використовувати обчислювальні ресурси, зменшуючи кількість фізичних серверів та знижуючи витрати на електроенергію та охолодження.

Підвищена гнучкість:

Віртуалізація та міграція в реальному часі дозволяють швидко та легко масштабувати ресурси відповідно до потреб бізнесу.

Спрощення управління:

Віртуалізація полегшує управління IT-інфраструктурою, дозволяючи централізовано управляти віртуальними машинами та ресурсами.

1.5. Контейнеризація для полегшення віртуалізації

Контейнеризація — це розгортання програмного забезпечення разом з усіма необхідними компонентами: код, бібліотеки, фреймворки тощо, таким чином, щоб вони були ізольовані у власному контейнері, Рисунок 1.6 .

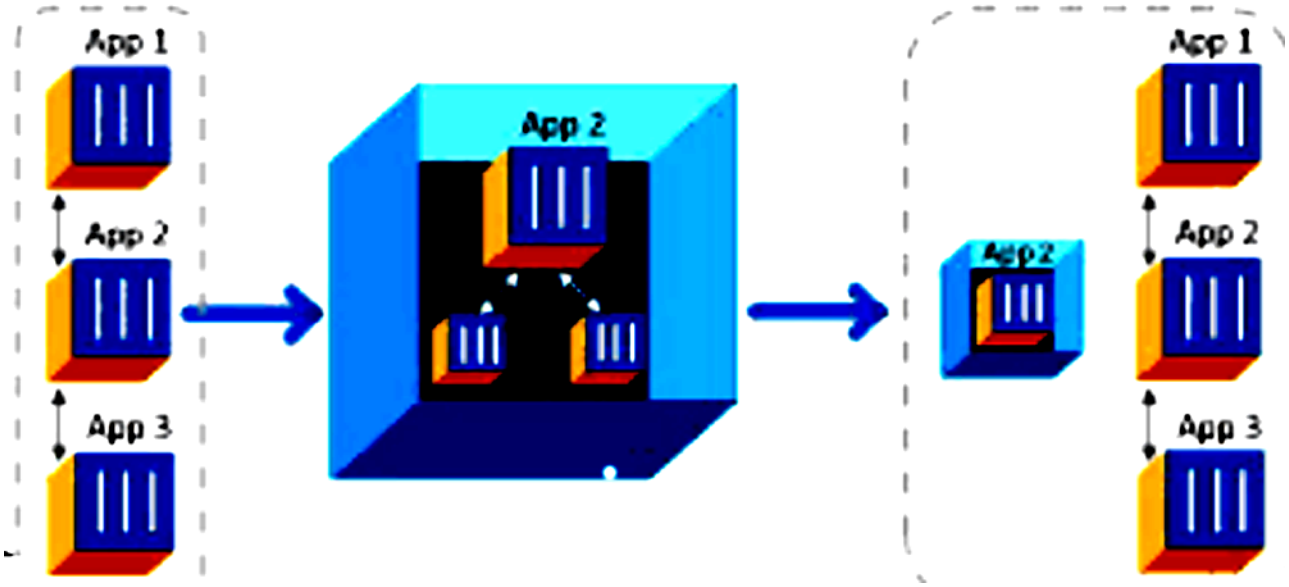

Рисунок 1.5 - Контейнеризація

Архітектура контейнеризації має декілька шарів:

Інфраструктура. Це апаратний рівень. Належить до фізичного комп’ютера чи сервера без встановленої операційної системи, на якому виконується контейнерний додаток.

Згідно з опитуванням IBM, 61% компаній, заявили, що протягом останніх років половину своїх додатків побудували за допомогою контейнеризації. Ця технологія дозволяє додаткам бути «написаними один раз і запущеними будь-де» [2]. Це пришвидшує розробку, запобігає прив’язці до одного хмарного оператора та пропонує ізоляцію, простоту управління, безпеку та багато іншого.

1.6. Кібератаки та таксономії в хмарних обчисленнях

Кібератаки в хмарних обчисленнях є загрозою для безпеки даних та інфраструктури, розміщених у хмарних середовищах. Таксономії кібератак у хмарі класифікують ці загрози за різними критеріями, такими як тип атаки, мета, використовувані інструменти тощо, що допомагає організаціям краще розуміти та протистояти кіберзагрозам .

Кібератаки у хмарних обчисленнях:

Атаки на доступ до даних:

Включають крадіжку облікових даних, фішинг, атаки з використанням шкідливого ПЗ для перехоплення даних, а також атаки на ланцюжок поставок.

Атаки на інфраструктуру:

Направлені на порушення роботи хмарної інфраструктури, включаючи DoS/DDoS-атаки, атаки на віртуальні машини, контейнери та мережі.

Атаки на додатки:

Націлені на вразливості у хмарних програмах, такі як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та інші атаки на веб-програми.

Атаки на управління хмарними ресурсами:

Включають атаки на інструменти управління хмарою, такі як API, консолі управління, і на сервіси автоматизації.

Атаки на ланцюжок поставок:

Використовують уразливості в ПЗ та сервісах, що постачаються сторонніми постачальниками, для отримання доступу до хмарних ресурсів.

Таксономії кібератак :

MITRE ATT&CK:

Популярний фреймворк, що описує різні етапи атаки, техніки, тактики та інструменти, які використовуються зловмисниками.

CIS Controls:

Набір критично важливих заходів безпеки, які допомагають організаціям захистити свої системи та дані.

NIST Cybersecurity Framework:

Рекомендації щодо управління ризиками кібербезпеки, включаючи ідентифікацію, захист, виявлення, реагування та відновлення [3].

Cloud Security Alliance (CSA) Cloud Controls Matrix (CCM):

Зведення правил безпеки для хмарних обчислень, що охоплює різні аспекти, такі як керування доступом, шифрування, безпека мережі тощо.

Вплив таксономій:

Розуміння загроз:

Таксономії допомагають організаціям зрозуміти, які типи атак найбільш актуальні для їхньої хмарної інфраструктури та додатків.

Оцінка ризиків:

Організації можуть використовувати таксономії для оцінки ризиків та визначення пріоритетів у сфері безпеки.

Розробка стратегії безпеки:

Таксономії є основою для розробки стратегій та заходів безпеки, спрямованих на захист від конкретних типів атак.

Поліпшення виявлення та реагування:

Таксономії допомагають у розробці засобів виявлення атак та реагування на них.

Приклади таксономій кібератак у хмарних обчисленнях:

MITRE ATT&CK for Cloud:

Спеціалізована версія MITRE ATT&CK, що описує техніки, тактики та інструменти, які використовуються зловмисниками у хмарних середовищах.

CSA CCM:

Фреймворк, який був розроблений Cloud Security Alliance, що надає рекомендації щодо безпеки у хмарних обчисленнях.

Cloud Security Alliance (CSA) Top Threats to Cloud Computing:

Список найпоширеніших загроз для хмарних обчислень.

Організаціям слід використовувати таксономії кібератак для покращення своєї стратегії безпеки, зниження ризиків та захисту хмарних ресурсів.

1.7. Потоки хмарної безпеки

Потік хмарної безпеки (Cloud Security Stream) – це напрям у сфері кібербезпеки, який спеціалізується на захисті хмарних інфраструктур, даних та програм від різних загроз . Воно включає технології, процеси та засоби контролю, спрямовані на забезпечення безпеки хмарних середовищ, у тому числі гібридних і мультихмарних.

Основні аспекти безпеки хмарних потоків:

Захист даних:

Забезпечення конфіденційності, цілісності та доступності даних, що зберігаються та оброблюються у хмарі.

Управління доступом:

Контроль доступу користувачів та програм до хмарних ресурсів, щоб запобігти несанкціонованому доступу та використанню.

Виявлення та реагування на загрози:

Використання засобів моніторингу та аналізу для виявлення та усунення загроз безпеці у хмарі.

Захист від захист від шкідливого ПЗ:

Використання антивірусних та антишпигунських програм для захисту від шкідливих програм, які можуть проникнути у хмарне середовище.

Захист від DDoS-атак:

Використання засобів захисту від розподілених атак типу «відмова в обслуговуванні», які можуть перевантажити хмарну інфраструктуру та зробити її недоступною.

Відповідність нормативним вимогам:

Забезпечення дотримання вимог законодавства та стандартів безпеки, що належать до хмарних обчислень.

Навчання та обізнаність:

Навчання користувачів та адміністраторів хмарних систем питанням безпеки, щоб підвищити обізнаність про ризики та способи їх запобігання.

Приклади загроз безпеки у хмарі:

Витік даних:

Несанкціонований доступ до конфіденційної інформації, що зберігається у хмарі.

Компрометування облікових записів:

Крадіжка облікових даних для отримання доступу до хмарних ресурсів.

Внутрішні загрози:

Загрози, які походять від співробітників чи підрядників.

Шкідливі програми:

Атаки із використанням шкідливого програмного забезпечення.

DDoS-атаки:

Перевантаження системи трафіком з метою зробити її недоступною.

Хмарна безпека важлива для:

Підприємств:

Для захисту конфіденційної інформації та забезпечення безперебійної роботи.

Державних установ:

Для захисту критично важливих даних та інфраструктури.

Приватні особи:

Для захисту особистої інформації та забезпечення безпеки власних даних.

1.8. Виявлення вторгнень у хмарні системи

Виявлення вторгнень у хмарних системах – це процес виявлення та реагування на несанкціонований доступ до хмарної інфраструктури або даних. Системи виявлення вторгнень (IDS) та запобігання вторгненням (IPS) є ключовими компонентами хмарної безпеки, що забезпечують захист від кібератак [14].

Основні аспекти виявлення вторгнень у хмарних системах:

IDS/IPS:

Системи виявлення вторгнень (IDS) та запобігання вторгненням (IPS) відстежують мережевий трафік та системні журнали на наявність ознак аномальної активності чи атак. IDS просто повідомляє про підозрілі активності, в той час як IPS може автоматично блокувати шкідливий трафік.

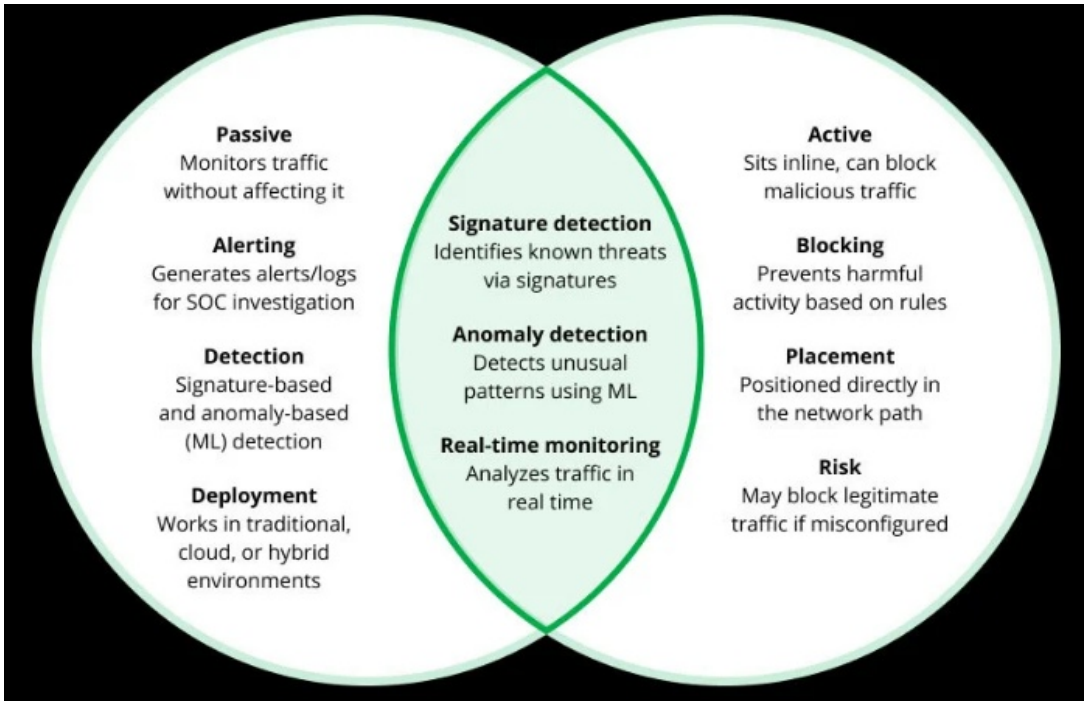

Рисунок 1.6 - Системи IDS/IPS

Хмарні особливості:

Хмарні системи мають унікальні характеристики, які впливають на процеси виявлення вторгнень. Це включає динамічну масштабованість, віртуалізацію і складну архітектуру. Спеціалізовані хмарні рішення IDS/IPS адаптовані для роботи в таких умовах.

Типи IDS:

Існують різні типи IDS, включаючи мережеві (NIDS), які аналізують мережевий трафік, та хостові (HIDS), які відстежують окремі вузли.

Методи виявлення:

Хмарні IDS використовують різні методи виявлення вторгнень, такі як аналіз сигнатур, виявлення аномалій, поведінковий аналіз та машинне навчання.

Виявлення аномалій:

Це важливий аспект, особливо в хмарних середовищах, де поведінка користувачів та програм може відрізнятися. Системи виявляють відхилення від нормальної роботи, що може вказувати на потенційні небезпеки.

Взаємодія з іншими системами безпеки:

IDS/IPS повинні бути інтегровані з іншими системами безпеки, такими як SIEM (Security Information and Event Management), для забезпечення комплексного захисту.

Хмарна ідентифікація:

Важливим елементом безпеки є управління ідентифікацією та доступом (IAM), яке включає автентифікацію та авторизацію користувачів у хмарному середовищі [13].

Та сама Вікіпедія включає захист від несанкціонованого доступу і забезпечує безпечний доступ до хмарних ресурсів.

Загалом виявлення вторгнень у хмарних системах потребує комплексного підходу, що включає використання спеціалізованих IDS/IPS, ефективні методи виявлення та інтеграцію з іншими системами безпеки.

DS розшифровується як Intrusion Detection System – система виявлення вторгнень. IPS, або Intrusion Prevention System, - система запобігання вторгненням. У порівнянні з традиційними засобами захисту – антивірусами, спам-фільтрами, брандмауерами – IDS/IPS забезпечують набагато більш високий рівень захисту мережі.

Висновки до розділу 1

У першому розділі проведено комплексне дослідження теоретичних засад функціонування хмарних обчислювальних систем та аналіз чинних підходів до їх захисту. За результатами розділу можна сформулювати такі висновки:

1. Еволюція хмарних технологій пройшла шлях від концепції надання ресурсів як комунальної послуги у 1960-х роках до створення складних еластичних інфраструктур, таких як AWS та Google Cloud, що дозволяють орендувати обчислювальні потужності з оплатою за фактичне використання.

2. Архітектура хмарних обчислень базується на шаруватій структурі сервісних моделей (IaaS, PaaS, SaaS) та різних моделях розгортання (публічна, приватна, гібридна), що визначає специфіку моделі спільної відповідальності за безпеку між провайдером та клієнтом.

3. Технології віртуалізації та контейнеризації є фундаментом сучасної хмарної інфраструктури, що забезпечує гнучкість та ізоляцію додатків, проте вони ж створюють специфічні вразливості та вектори атак через гіпервізор або спільне ядро ОС.

4. Аналіз кіберзагроз показав, що хмарні середовища вразливі до широкого спектра атак: від викрадення даних до масованих DDoS-атак, що потребує використання спеціалізованих таксономій, таких як MITRE ATT&CK for Cloud, для ідентифікації тактик зловмисників.

5. Традиційні системи виявлення вторгнень (IDS/IPS) потребують адаптації до динамічних умов хмари. Ефективний захист вимагає інтеграції різних методів (аналізу сигнатур, виявлення аномалій та машинного навчання) для забезпечення цілісності та доступності критично важливих даних моніторингу.

Отримані теоретичні результати стають підґрунтям для подальшого аналізу проблем конфіденційності та проектування децентралізованої системи аудиту в наступних розділах роботи.

2 АНАЛІЗ ПРОБЛЕМАТИКИ ТА МЕТОДІВ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОГО ЗБЕРІГАННЯ ДАНИХ МОНІТОРИНГУ

2.1. Аналіз впливу характеристик «Великих даних» на системи безпеки хмарних середовищ

Стрімка міграція корпоративних інфраструктур у хмари та перехід до мікросервісної архітектури призвели до фундаментальних змін у природі даних, що генеруються засобами моніторингу безпеки. Традиційні підходи до збору та аналізу журналів подій, які ефективно працювали в локальних мережах, стають вузьким місцем в умовах хмарних обчислень. Для формалізації цієї проблематики доцільно проаналізувати потоки даних безпеки крізь призму концепції «Великих даних», використовуючи модель п’яти «V»: обсяг, швидкість, різноманітність, достовірність та цінність [15].

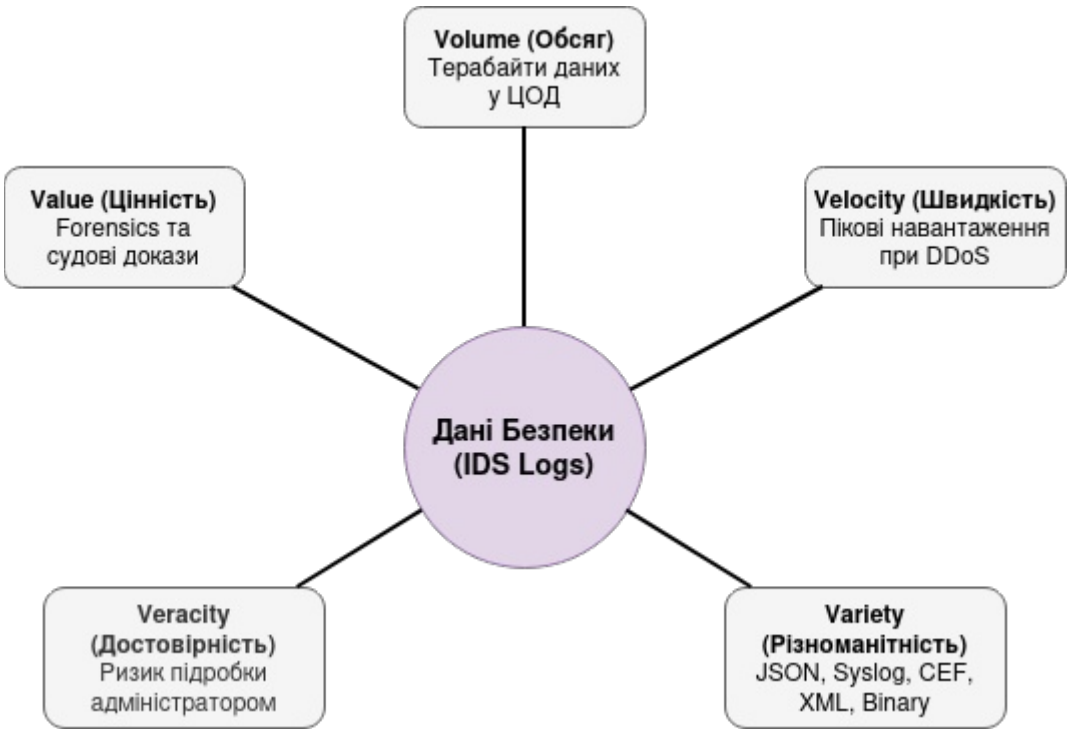

Рисунок 2.1 - Модель «5V» у контексті журналів безпеки

У сучасних центрах обробки даних обсяги логів, що генеруються системами виявлення вторгнень, брандмауерами веб-додатків та системними журналами віртуальних машин, досягають терабайтів щодня. Критичною проблемою є нелінійна залежність обсягу даних від активності зловмисників. Під час кібератак, зокрема розподілених атак на відмову в обслуговуванні або масованого сканування вразливостей, швидкість генерації записів може зростати на кілька порядків за лічені секунди.

Така динаміка створює дві архітектурні вразливості. По-перше, виникає ризик відмови у обслуговуванні підсистеми аудиту, оскільки традиційні централізовані сховища логів мають фіксовану пропускну здатність запису. Різкий сплеск активності призводить до черг на запис і, як наслідок, до втрати критично важливих доказів атаки саме в той момент, коли вони найбільше потрібні. По-друге, спостерігається економічна неефективність повного логування [16]. Зберігання повної історії сирих логів вимагає значних фінансових витрат на дисковий простір хмарного сховища. При цьому значна частина цих даних є надлишковою, що робить вартість зберігання невиправдано високою .

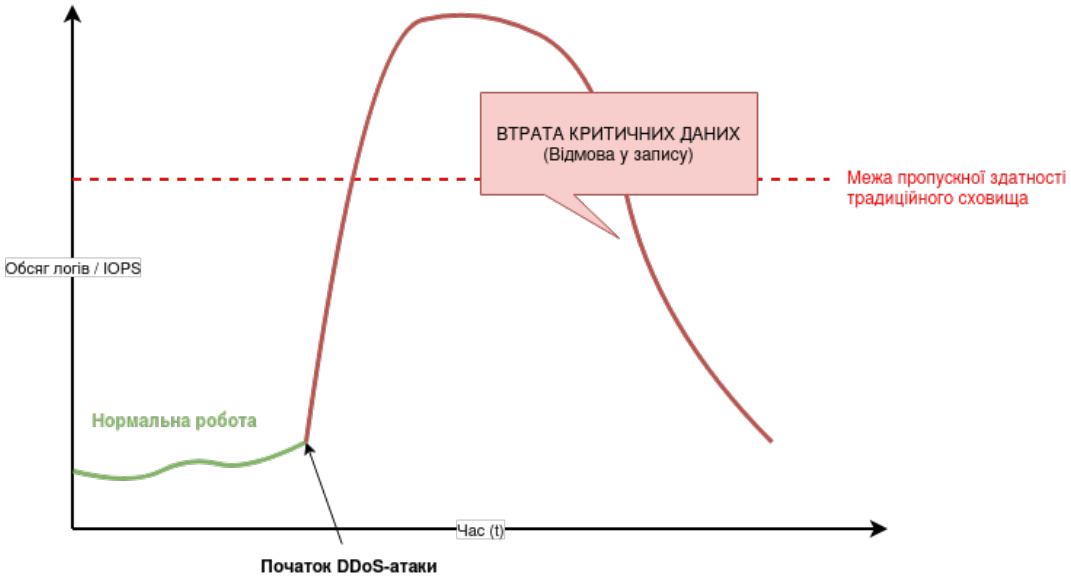

Рисунок 2.2 - Динаміка зростання обсягу логів під час інциденту безпеки

Гетерогенність хмарних середовищ призводить до високої ентропії форматів даних. Логи надходять у структурованому, напівструктурованому та неструктурованому вигляді з різнорідних джерел — від контейнерів до пристроїв інтернету речей. Це ускладнює процеси нормалізації даних та їх кореляції в реальному часі.

Найбільш критичним для задач безпеки є аспект достовірності. У хмарній моделі спільної відповідальності клієнт часто не має фізичного контролю над інфраструктурою, де зберігаються логи. Це створює фундаментальну проблему довіри. Якщо адміністратор хмарного провайдера або зловмисник, що отримав права суперкористувача, має можливість непомітно модифікувати або видалити журнали подій, то вся система безпеки втрачає сенс. Логи перестають бути надійним доказом для криміналістичного аналізу та судових розглядів.

Проведений аналіз дозволяє стверджувати, що пряме зберігання великих масивів даних безпеки у хмарі є неефективним з економічної точки зору та ризикованим з точки зору цілісності. Хмарні провайдери та споживачі послуг стикаються з дилемою: з одного боку, регуляторні вимоги вимагають забезпечити повну збереженість історії подій, а з іншого — фізичні обмеження сховищ та каналів зв'язку вимагають оптимізації трафіку. Вирішення цієї проблеми лежить у площині пошуку технологічного компромісу між необхідністю впровадження інтелектуальних механізмів скорочення обсягу даних без втрати їхньої інформативності та необхідністю гарантування незмінності даних у середовищі, якому неможливо повністю довіряти.

2.2. Дослідження надлишковості журналів подій та порівняльний аналіз методів дедуплікації

Специфіка функціонування систем виявлення вторгнень полягає у генерації значної кількості однотипних повідомлень про події безпеки. Емпіричні дані свідчать, що під час масованих кібератак або автоматизованого сканування периметра мережі сенсори можуть генерувати тисячі ідентичних записів про одну й ту ж подію за короткий проміжок часу. Частка дубльованих записів у загальному потоці логів у таких випадках є критично високою, що робить зберігання повної історії подій нераціональним використанням обчислювальних ресурсів.

Для формалізації ефективності процесу оптимізації зберігання доцільно ввести поняття коефіцієнта дедуплікації (DR), який визначається як відношення початкового обсягу даних (V_{total}) до обсягу даних після усунення дублікатів (V_{unique}) :

$$DR = \frac{V_{total}}{V_{unique}}$$

У контексті хмарної безпеки, де параметр зростає експоненційно під час атак типу «відмова в обслуговуванні», максимізація показника DR є критичним завданням для забезпечення економічної та технічної життєздатності системи аудиту. Високий коефіцієнт дедуплікації дозволяє не лише зменшити витрати на дисковий простір, а й пришвидшити процеси індексації та пошуку інцидентів .

Порівняльний аналіз архітектурних підходів до дедуплікації

Залежно від місця виконання процедури ідентифікації та усунення дублікатів, існуючі технологічні рішення класифікують на два основні типи: дедуплікація на стороні джерела (Source-based) та дедуплікація на стороні цільового сховища (Target-based) [17].

Дедуплікація на стороні джерела передбачає обчислення унікальних ідентифікаторів (хешів) блоків даних безпосередньо на пристрої користувача перед їх відправкою. Хоча цей підхід дозволяє заощадити пропускну здатність мережі, він створює значне обчислювальне навантаження на сенсори IDS, що може бути критичним в умовах обмежених ресурсів локальних вузлів або IoT-пристроїв .

Дедуплікація на стороні цільового сховища (Target-based), навпаки, передбачає передачу повного обсягу даних від сенсора до хмарного сервера, де за допомогою мультиагентної системи відбувається подальша обробка та видалення дублікатів. Сервер аналізує вхідний потік, ідентифікує ідентичні фрагменти та зберігає лише посилання на існуючі копії. Такий підхід дозволяє повністю звільнити ресурси сенсорів від необхідності виконання складних криптографічних обчислень перед відправкою, що забезпечує стабільність моніторингу навіть на малопотужних вузлах.

Результати порівняльного аналізу методів у контексті вимог до систем виявлення вторгнень наведено в Таблиці 2.1.

Таблиця 2.1 — Порівняльний аналіз методів дедуплікації для систем безпеки

Критерій порівняння	**Дедуплікація на стороні джерела (Source-based)**	**Дедуплікація на стороні цілі (Target-based)**
Ефективність використання сховища	Висока	Висока
Ефективність використання мережі	Максимальна	Низька (передаються всі дані)
Навантаження на сенсор	Середнє (обчислення хешів)	Мінімальне (лише передача)
Стійкість сенсора до перевантажень	Низька	Висока (ресурси CPU вільні)
Управління даними	Децентралізоване	Централізоване та прозоре

Висновки щодо вибору методу

На основі проведеного аналізу для проектованої системи обрано метод дедуплікації на стороні цільового сховища (Target-based). Це рішення обумовлено пріоритетністю збереження обчислювальної потужності сенсорів IDS. В умовах інтенсивного трафіку під час кібератак використання ресурсів сенсора на хешування кожного запису може призвести до його перевантаження та пропуску шкідливих пакетів. Винесення процесу дедуплікації у хмару дозволяє масштабувати обчислювальні ресурси на стороні провайдера відповідно до обсягу даних. Однак такий підхід потребує впровадження механізмів захисту від атак

побічного каналу на стороні сервера для забезпечення конфіденційності метаданих [18].

2.3. Аналіз загроз конфіденційності в системах з дедуплікацією

Впровадження механізмів дедуплікації на стороні цільового сховища у хмарних системах, які обробляють лог-файли систем виявлення вторгнень, створює фундаментальний архітектурний конфлікт із вимогами конфіденційності. При використанні моделі дедуплікації на стороні сервера виникають специфічні вразливості, пов’язані з необхідністю ідентифікації однакових фрагментів даних серед зашифрованих потоків та витоками інформації через побічні канали зв'язку.

Аналіз криптографічного конфлікту

Традиційні стандарти безпеки вимагають використання семантично стійкого шифрування, де шифротекст не розкриває жодної інформації про вихідний текст. Для досягнення цієї властивості алгоритми симетричного шифрування використовують унікальні ключі або випадкові вектори ініціалізації для кожного окремого файлу. Математично це призводить до ситуації, де для двох ідентичних повідомлень , зашифрованих різними ключами та , отримуємо різні шифротексти:

Це робить дедуплікацію на стороні провайдера неможливою без розкриття йому ключів шифрування. Для вирішення цього конфлікту в системах із нульовою довірою застосовують схеми шифрування, що залежать від вмісту повідомлення, зокрема конвергентне шифрування. Це дозволяє генерувати

ідентичні шифротексти для однакових вхідних даних, проте відкриває шлях до атак шляхом повного перебору на підмножині можливих повідомлень.

Аналіз атак побічного каналу в target-based системах

У системах із дедуплікацією на стороні сервера клієнт зазвичай передає повний обсяг даних, що нібито має маскувати факт наявності дублікатів. Проте логіка обробки запитів на стороні сховища дозволяє зловмиснику отримати інформацію про вміст бази даних через наступні вектори:

1. Атака через аналіз часу відгуку. Сервер витрачає значно менше часу на обробку дубліката, оскільки операції запису на диск замінюються оновленням посилань у базі метаданих. Вимірювання часу відгуку дозволяє зловмиснику дізнатися, чи був аналогічний інцидент безпеки вже зафіксований іншим вузлом мережі.
2. Аналіз квоти зберігання. Якщо хмарний інтерфейс відображає обсяг доступного або використаного місця, користувач може відстежити, чи змінився цей показник після завершення завантаження. Відсутність змін у лічильнику зайнятого простору є прямим підтвердженням того, що файл уже існує в системі.

Алгоритм реалізації загрози підтвердження файлу

1. Підготовка. Зловмисник формує файл логу, який відповідає сигнатурі конкретного мережевого вторгнення або критичної помилки системи.

2. Завантаження. Сформований файл надсилається до хмарного сховища під виглядом звичайної операції резервного копіювання.
3. Моніторинг побічних каналів. Зловмисник фіксує точний час виконання запиту та стан своєї дискової квоти до і після операції.
4. Висновок. Отримання аномально швидкої відповіді або відсутність фактичного зменшення доступного дискового простору підтверджує гіпотезу про те, що система захисту жертви вже виявила та зафіксувала аналогічний інцидент.

Висновки щодо загроз

Стандартні протоколи дедуплікації на стороні сервера є вразливими до витоку конфіденційної інформації через експлуатацію побічних каналів. Для нейтралізації цих загроз необхідно впроваджувати механізми рандомізації часу відгуку сервера та протоколи доведення володіння. Використання таких протоколів вимагає від клієнта довести знання всього вмісту файлу перед тим, як система надасть будь-яку реакцію на запит. Це робить атаку підтвердження файлу обчислювально складною для суб’єкта, який не володіє оригінальними даними, та забезпечує цілісність процесу обробки логів у децентралізованих середовищах .

2.4. Теоретичні засади використання технології блокчейн для забезпечення цілісності даних

Блокчейн – це особливий вид бази даних. Це децентралізований цифровий реєстр, який підтримується розподіленою мережею комп'ютерів [39]. Дані

блокчейну організовані в блоки, які хронологічно розташовані та захищені криптографією.

Ця структура гарантує прозорість, безпеку та незмінність даних. Практично неможливо змінити дані, що зберігаються в блоці, після того, як блок підтверджено та додано до ланцюжка. Децентралізована структура також усуває потребу в центральному органі. Транзакції блокчейну можуть відбуватися між користувачами без потреби в посередниках .

Існують різні типи блокчейнів з різним ступенем децентралізації. Тим не менш, термін блокчейн зазвичай стосується децентралізованого цифрового реєстру, який використовується для запису транзакцій криптовалюти.

Коротка історія блокчейну

Найдавніша модель блокчейну була створена на початку 1990-х років, коли вчений-інформатик Стюарт Габер та фізик В. Скотт Сторнетта застосували криптографічні методи в ланцюжку блоків як спосіб захисту цифрових документів від підробки даних.

Габер і Сторнетта надихнули на роботу багатьох інших вчених-інформатиків та ентузіастів криптографії, що зрештою призвело до створення Bitcoin як першої криптовалюти, що працює на технології блокчейн [20]. Відтоді використання блокчейну значно зросло, і криптовалюти зараз є глобальним явищем.

Хоча технологія блокчейн часто використовується для запису криптовалютних транзакцій [21], вона підходить для запису багатьох інших типів цифрових даних і може застосовуватися в широкому спектрі випадків використання.

Ключові характеристики та переваги блокчейну

Децентралізація: Інформація зберігається в мережі комп'ютерів (вузлів), а не на одному центральному сервері. Великі децентралізовані мережі, такі як Bitcoin, мають високу стійкість до атак.

Прозорість: Більшість блокчейнів є публічними, тобто всі учасники мають доступ до однієї бази даних. Транзакції видимі для всіх учасників.

Незмінність: Після додавання даних до блокчейну їх не можна змінити без консенсусу мережі.

Безпека даних: Криптографія та механізми консенсусу забезпечують надійний захист від підробки даних.

Ефективність: Блокчейн може забезпечити швидші та дешевші транзакції, усуваючи потребу в посередниках. Транзакції обробляються майже в режимі реального часу.

Що таке децентралізація, яка реалізована у блокчейні?

Децентралізація, що реалізована в блокчейні, стосується ідеї, що контрольні повноваження та повноваження щодо прийняття рішень у мережі розподілені між її користувачами, а не контролюються однією організацією, такою як банк, уряд чи корпорація.

У децентралізованій мережі блокчейн немає центрального органу чи посередника, який би контролював потік даних або транзакцій. Натомість транзакції перевіряються та реєструються розподіленою мережею комп'ютерів, які працюють разом для підтримки цілісності мережі.

Як працює блокчейн?

По суті, блокчейн — це цифровий реєстр, який безпечно реєструє транзакції між двома сторонами у захищений від несанкціонованого доступу спосіб. Ці дані транзакцій реєструються глобально розподіленою мережею комп'ютерів (вузлів).

Коли Аліса надсилає Бобу трохи біткойнів, транзакція транслюється в мережу. Кожен вузол автентифікує транзакцію, перевіряючи цифрові підписи та інші дані транзакцій. Після підтвердження транзакції вона додається до блоку разом з іншими транзакціями. Ми можемо уявляти кожен блок як сторінку цифрового реєстру.

Блокчейн працює наступним чином:

1. Запис транзакцій

Коли ініціюється транзакція (наприклад, переказ криптовалюти), вона транслюється до мережі вузлів. Кожен вузол перевіряє транзакцію за допомогою заздалегідь визначених правил.

2. Формування блоку

Перевірені транзакції групуються в блок. Кожен блок містить:

Дані (наприклад, деталі транзакції)

Мітку часу

Криптографічний хеш: унікальний ідентифікатор, створений шляхом проходження даних блоку через алгоритм хешування.

Хеш попереднього блоку: це те, що пов'язує блоки разом, формуючи ланцюг.

3. Механізм консенсусу

Щоб додати блок до ланцюга, учасники мережі повинні домовитися про його дійсність. Це досягається за допомогою алгоритму консенсусу, такого як Proof of Work (PoW) та Proof of Stake (PoS). Ми обговоримо обидва більш детально найближчим часом, але ось короткий виклад:

Proof of Work (PoW): Використовуваний Bitcoin, PoW вимагає від валідаторів блоків використовувати обчислювальну потужність для вирішення складних проблем.

Підтвердження частки (PoS): Використовується новішими блокчейнами, такими як Ethereum, де валідатори блоків обираються на основі їхньої частки в мережі.

4. Зв'язування ланцюгів

Після перевірки блок додається до блокчейну. Кожен наступний блок посилається на попередній, забезпечуючи захищену від несанкціонованого доступу структуру. Іншими словами, щоб новий блок був перевірений, він повинен використовувати попередній ідентифікатор блоку.

5. Прозорість

Ще однією особливістю блокчейну є його прозорість. Будь-хто може перевірити дані блокчейну, включаючи всі дані транзакцій та дані блоків, на публічних веб-сайтах, відомих як дослідники блокчейнів.

Наприклад, ви можете побачити кожну транзакцію, яка коли-небудь була записана в мережі Bitcoin, включаючи адресу гаманця відправника та одержувача, суму переказу тощо. Ви також можете відстежити всі блоки Bitcoin аж до першого блоку, відомого як блок генезису.

Принцип роботи блокчейну проілюстрований на рис. 2.3.

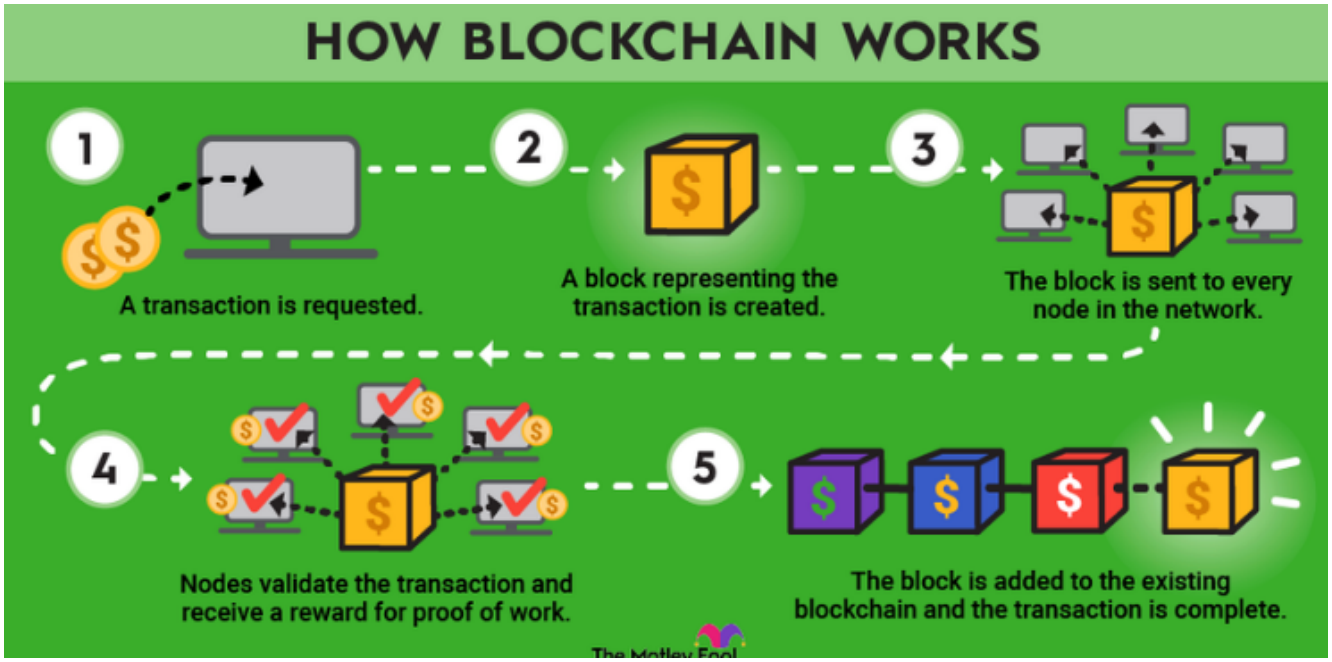

Рисунок 2.3 - Принцип роботи блокчейну

Блоки об'єднуються в ланцюжок за допомогою криптографічних методів, утворюючи блокчейн. Процес перевірки транзакцій та їх додавання до блокчейну здійснюється за допомогою механізму консенсусу – набору правил, які регулюють, як вузли в мережі досягають згоди щодо стану блокчейну та дійсності транзакцій.

2.5. Аналіз архітектурних підходів до забезпечення цілісності: Централізовані БД проти Блокчейну

В умовах розподілених хмарних обчислень, де адміністративний контроль над інфраструктурою знаходиться в руках провайдера послуг, виникає проблема довіри до збережених даних. Для журналів подій систем безпеки критично важливою вимогою є незмінність історії. Традиційні архітектурні підходи до зберігання даних, засновані на реляційних або NoSQL базах даних, мають суттєвий недолік у контексті безпеки — наявність суперкористувача (адміністратора), який має технічну можливість виконувати операції модифікації

та видалення записів. Це створює єдину точку компрометації: якщо зловмисник отримує привілейований доступ до бази даних, він може непомітно змінити історію логів, щоб приховати сліди своєї присутності.

Для вирішення цієї проблеми доцільно використовувати технологію розподіленого реєстру. Блокчейн представляє собою децентралізовану базу даних, яка забезпечує прозорість та криптографічну гарантію незмінності записів без необхідності залучення довіреної третьої сторони [22], [23], [24]. Архітектурно це реалізується через послідовне хешування блоків: кожен наступний блок містить цифровий відбиток попереднього. Спроба зміни історичного запису вимагає перерахунку хешів усіх наступних блоків у ланцюжку, що робить підробку обчислювально неможливою при достатній потужності мережі.

Для організації даних всередині блоків та забезпечення можливості їх швидкої верифікації використовується структура дерева хешів Меркла. Ця структура дозволяє перевірити належність окремого запису до великого масиву даних, маючи лише кореневий хеш дерева та шлях автентифікації, без необхідності завантаження всього масиву.

Порівняльний аналіз традиційних баз даних та технології блокчейн у контексті задачі зберігання логів наведено в Таблиці 2.2 [23].

Таблиця 2.2 — Порівняльний аналіз традиційних баз даних та технології блокчейн у контексті задачі зберігання логів

Характеристика	**Централізована БД (SQL/NoSQL)**	**Розподілений реєстр (Blockchain)**
Модель управління	Централізована (є адміністратор)	Децентралізована (консенсус)
Стійкість до підробки	Низька (залежить від чесності адміна)	Максимальна (гарантується математично)
Швидкість запису	Висока (>10 000 транзакцій/сек)	Низька (10-100 транзакцій/сек)
Вартість зберігання	Низька (за ГБ)	Дуже висока (за КБ)
Конфіденційність	Контрольована правами доступу	Публічна (вимагає шифрування даних)
Придатність для аудиту	Вимагає довіри до аудитора	Trustless (публічна верифікація)

Аналіз показує, що блокчейн суттєво поступається традиційним базам даних у швидкості запису транзакцій та вартості зберігання одиниці інформації. Прямий запис гігабайтів логів у публічний блокчейн є економічно недоцільним через високу вартість комісій та обмеження розміру блоку. Водночас традиційні бази даних не забезпечують необхідного рівня захисту від інсайдерських загроз та підробки даних.

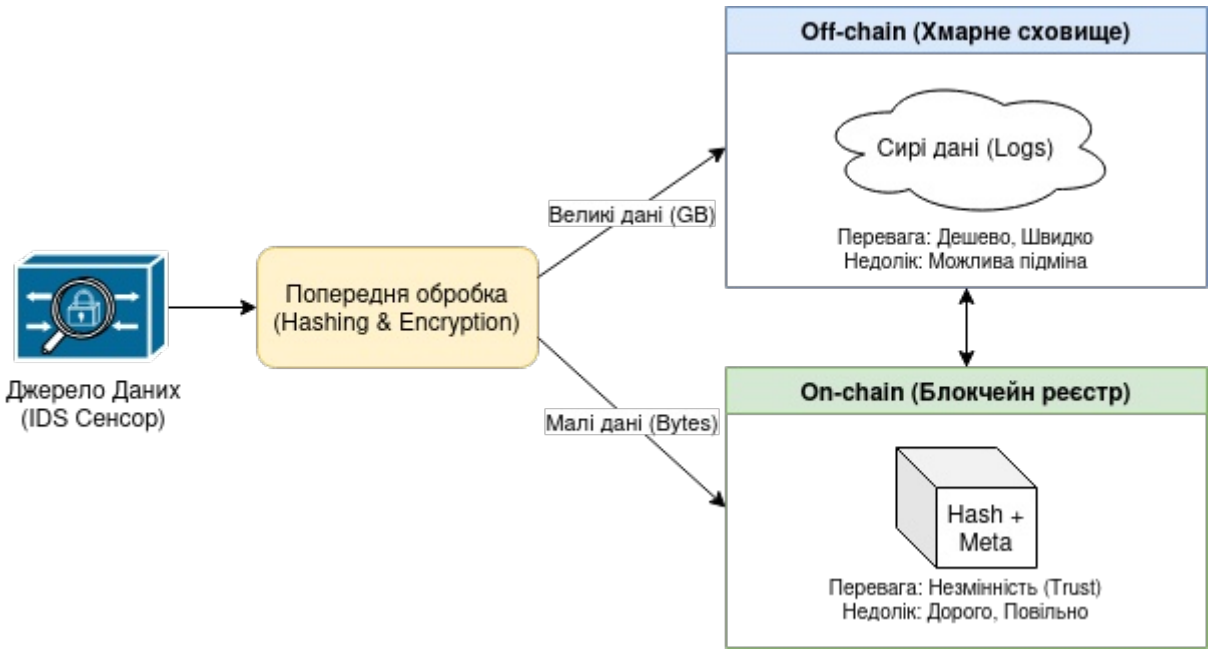

Рисунок 2.4 - Гібридна модель зберігання даних моніторингу на основі Off-chain та On-chain архітектури

На основі цього аналізу можна зробити висновок про необхідність застосування гібридного архітектурного підходу. Оптимальна модель передбачає зберігання об'ємних тіл файлів логів у високопродуктивних хмарних сховищах (Off-chain), тоді як блокчейн (On-chain) використовується виключно як реєстр якорів довіри — компактних криптографічних доказів цілісності (кореневих хешів Меркла) та метаданих. Такий підхід дозволяє поєднати високу пропускну здатність хмарних систем із гарантіями незмінності, що надаються розподіленим реєстром.

2.6. Обґрунтування використання мультиагентного підходу до децентралізації процесів захисту

Складність сучасних хмарних інфраструктур висуває нові вимоги до архітектури систем безпеки. Традиційний підхід до побудови шлюзів безпеки базується на монолітній архітектурі, де всі функціональні модулі (збір логів, шифрування, взаємодія з базою даних) об'єднані в єдиний виконуваний процес. Проведений аналіз дозволяє виділити критичні недоліки такої моделі в контексті обробки великих даних: наявність єдиної точки відмови та складність вертикального масштабування. Якщо монолітний процес аварійно завершує роботу через переповнення буфера під час DDoS-атаки, зупиняється вся система аудиту.

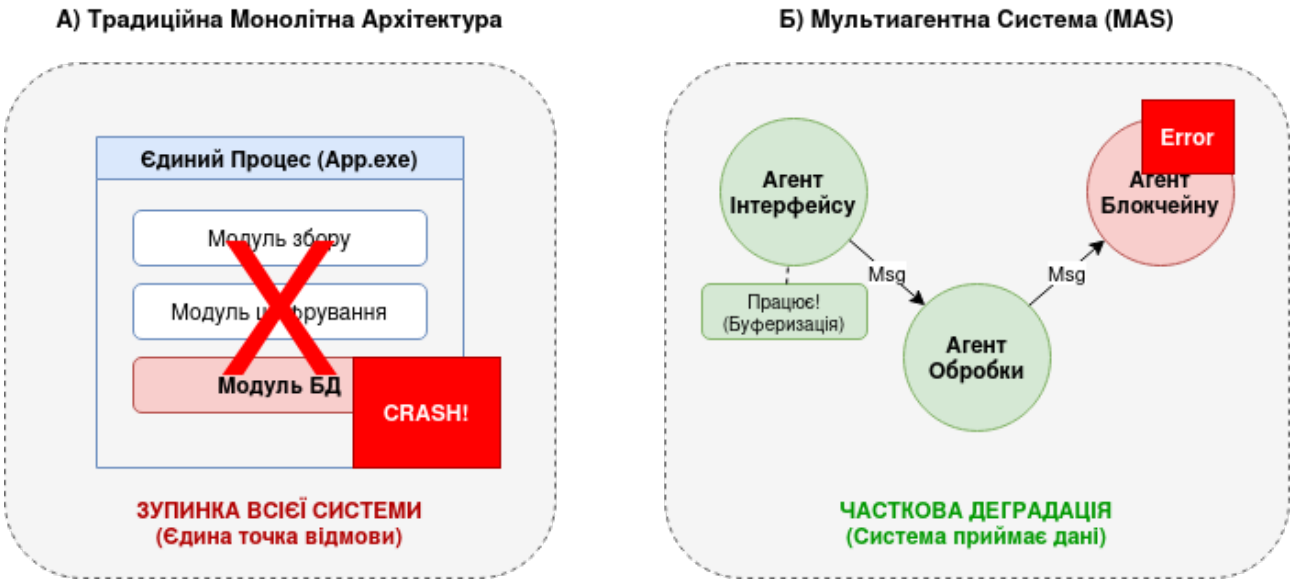

Рисунок 2.5 - Порівняння архітектур: Монолітна vs Мультиагентна (MAS)

Для вирішення цих проблем доцільно застосувати підхід, заснований на мультиагентних системах. Мультиагентна система — це сукупність автономних інтелектуальних агентів, які взаємодіють між собою для досягнення спільної

мети, використовуючи розподілені ресурси [24]. У контексті розроблюваної системи захисту декомпозиція процесів на незалежних агентів надає низку архітектурних переваг [25]:

1. Асинхронність та реактивність. Агенти працюють незалежно один від одного. Агент-інтерфейс може продовжувати приймати потік логів від сенсорів, навіть якщо Агент-даних тимчасово втратив зв'язок з блокчейном. Це дозволяє реалізувати буферизацію та уникнути втрати даних під час пікових навантажень.

2. Стійкість до відмов. У разі компрометації або падіння одного агента, інші компоненти системи продовжують функціонувати. Це відповідає принципам живучості критичних систем.

3. Розподіл навантаження. Найбільш ресурсомісткі операції, такі як обчислення хешів Меркла та конвергентне шифрування, можуть бути винесені на окремі обчислювальні вузли. Агентний підхід дозволяє гнучко масштабувати систему: при зростанні навантаження можна динамічно запустити додаткові екземпляри агентів без зупинки основної системи.

Взаємодія між агентами повинна будуватися на базі полегшених протоколів обміну повідомленнями, що дозволяє створити слабозв'язану систему, здатну адаптуватися до змін у мережевому середовищі. Такий підхід є еволюційним розвитком систем захисту, переходячи від статичних інструментів до динамічних, самоорганізованих екосистем безпеки.

Висновки до розділу 2

У другому розділі проведено комплексний аналіз проблематики забезпечення конфіденційності та цілісності даних моніторингу в хмарних середовищах. Результати дослідження дозволяють сформулювати наступні ключові положення, що визначають архітектуру проектованої системи:

Проблема масштабованості та обробки великих даних. Встановлено, що в умовах сучасних кіберзагроз традиційні методи логування не справляються з характеристиками «Великих даних» (модель 5V). Під час DDoS-атак швидкість генерації логів перевищує пропускну здатність каналів запису, що призводить до втрати критичних доказів інциденту саме в момент атаки.

Обґрунтування методу дедуплікації. Порівняльний аналіз архітектурних підходів показав, що для систем виявлення вторгнень найбільш доцільним є метод дедуплікації на стороні цільової системи (Target-based). Це дозволяє повністю звільнити обчислювальні ресурси сенсорів IDS від виконання складних криптографічних операцій під час атак, перенісши навантаження на масштабовані хмарні потужності.

Забезпечення безпеки в умовах дедуплікації. Виявлено, що використання серверної дедуплікації створює специфічні вектори атак побічного каналу, зокрема атаку підтвердження файлу. Для нейтралізації цих загроз та забезпечення конфіденційності метаданих обґрунтовано необхідність впровадження криптографічних протоколів «доведення володіння» (Proof of Ownership) та рандомізації часу відгуку сервера.

Гібридна модель зберігання та довіри. Доведено неефективність централізованих баз даних через наявність єдиної точки компрометації в особі адміністратора. Обґрунтовано оптимальність гібридної архітектури: зберігання

зашифрованих «сирих» даних у хмарному сховищі (Off-chain) з одночасною фіксацією якорів довіри у вигляді кореневих хешів Меркла в розподіленому реєстрі блокчейн (On-chain).

Переваги мультиагентного підходу. Для подолання недоліків монолітних систем захисту (низька живучість, складність масштабування) обґрунтовано перехід до мультиагентної системи (MAS). Розподіл функцій між автономними агентами дозволяє забезпечити асинхронну обробку даних, буферизацію логів та уникнути єдиної точки відмови.

Таким чином, результати аналітичного дослідження формують наукове підґрунтя для розробки у наступному розділі архітектури децентралізованої системи безпечного аудиту, яка поєднує ефективність хмарних обчислень із математичними гарантіями цілісності блокчейну.

3 ПРОЄКТУВАННЯ СИСТЕМИ БЕЗПЕЧНОГО АУДИТУ

3.1. Розробка загальної архітектури мультиагентної системи

Спираючись на результати аналізу, проведеного у другому розділі, для побудови системи захищеного аудиту було обрано архітектурний патерн мікроядерної мультиагентної системи. Така структура дозволяє нейтралізувати проблеми масштабованості та забезпечити високу відмовостійкість через розподіл функціональних завдань між автономними програмними модулями (агентами), що взаємодіють за допомогою асинхронного обміну повідомленнями.

Проєктована архітектура містить три ключові функціональні ланки, що складаються з п'яти спеціалізованих типів агентів:

1. Ланка агрегації та маршрутизації.

Агент-Інтерфейс (Agent-Interface): виконує роль вхідного шлюзу, що здійснює прийом логів від зовнішніх джерел (сенсорів IDS, серверних вузлів). Головною перевагою є впровадження механізму буферизації, що запобігає втраті даних у періоди пікового навантаження під час DDoS-атак.

Агент-Посередник (Agent-Mediator): забезпечує координацію та ізоляцію внутрішніх процесів, виконуючи функції інтелектуальної маршрутизації запитів між компонентами системи.

2. Ланка інтелектуальної обробки.

Агент-Контролер (Agent-Controller): відповідає за приведення логів до єдиного формату, сегментацію файлів на блоки та побудову криптографічної структури Дерева Меркла.

Агент-Аналізу (Agent-Analysis): реалізує алгоритми дедуплікації шляхом верифікації хеш-сум у спеціалізованих реєстрах (Merkle DB та Hash DB). Також цей модуль здійснює конвергентне шифрування даних, що гарантує конфіденційність перед їх відправкою у сховище.

3. Ланка гібридного зберігання.

Агент-Даних (Agent-Data): забезпечує фізичне розміщення інформації згідно з гібридною моделлю. Об’ємні зашифровані масиви передаються до хмарного сховища через REST API, а компактні криптографічні відбитки (якорі довіри) фіксуються в реєстрі блокчейн. Такий підхід до організації сховища відповідає сучасним вимогам безпеки в децентралізованих середовищах [33].

Взаємодія між усіма агентами організована за допомогою брокера повідомлень, що забезпечує слабку зв'язність системи [26]. Це гарантує безперервність збору логів навіть у разі тимчасової деградації сервісів блокчейну або хмарного провайдера. Загальну компонентну діаграму архітектури мультиагентної системи представлено на рисунку 3.1.

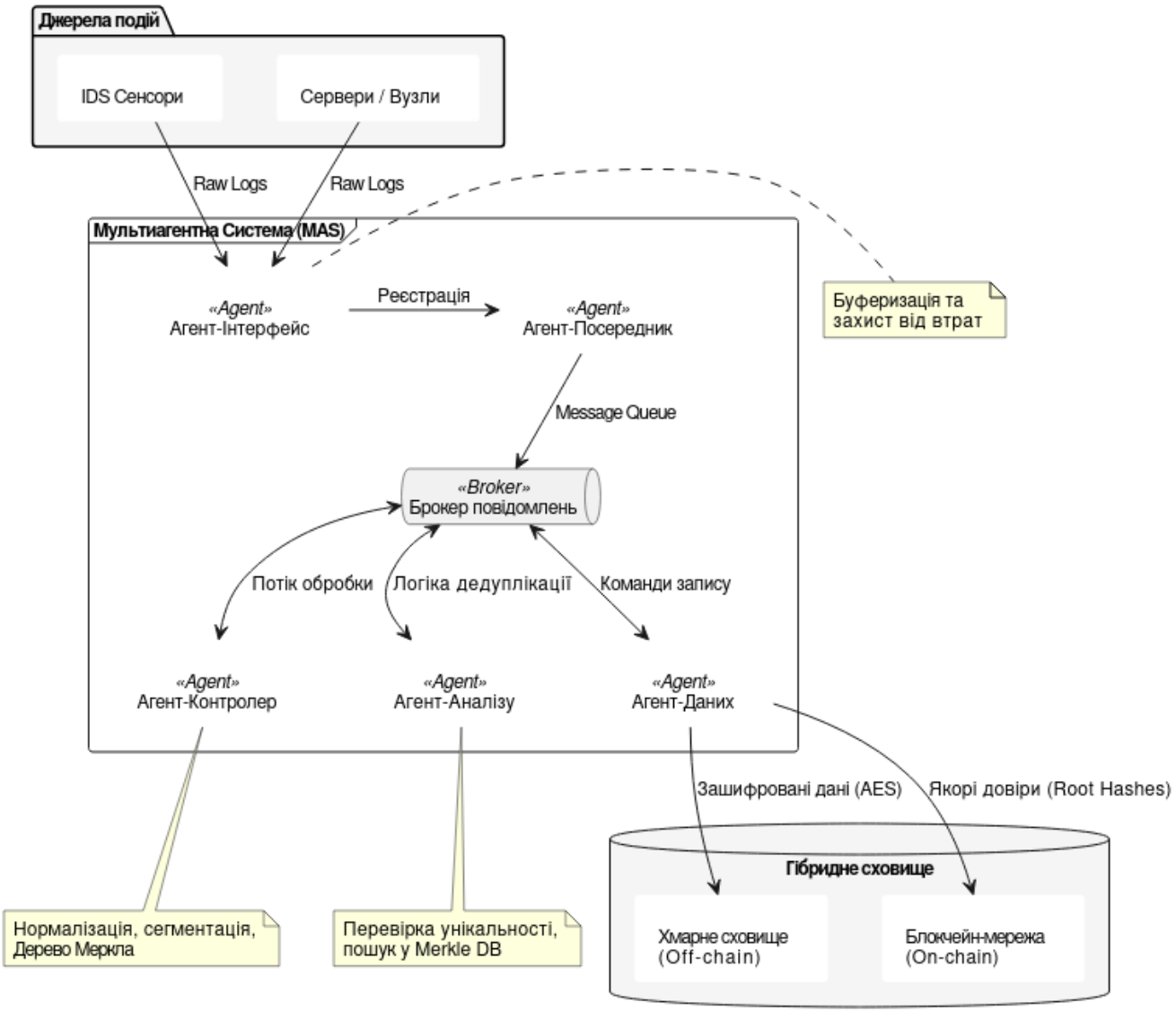

Рисунок 3.1 - Архітектура мультиагентної системи аудиту логів

Запропонована децентралізована модель дозволяє уникнути наявності єдиної точки відмови: у разі критичної помилки одного з екземплярів обробника, брокер повідомлень автоматично перенаправляє завдання на активні вузли, підтримуючи цілісність процесу аудиту.

3.2. Математичне та алгоритмічне забезпечення дедуплікації на стороні серверу

У розробленій системі реалізовано метод дедуплікації на стороні цільової системи (Target-based deduplication). Цей підхід передбачає передачу повного файлу від клієнта до хмарного середовища, де за процес виявлення та усунення надлишковості відповідає мультиагентна система. Використання серверної дедуплікації дозволяє перенести обчислювальне навантаження з інфраструктури замовника на потужності провайдера та забезпечити централізоване управління цілісністю даних.

Математична модель обробки даних у системі базується на використанні криптографічного гешування та структури Дерева Меркла. Процес формалізується наступним чином:

1. Попередня обробка та сегментація. Агент обробки (Agent-Processor) отримує вхідний потік від Агента збору (Agent-Collector). Файл розділяється на блоки фіксованого розміру. Для кожного блоку обчислюється хеш-сума за алгоритмом SHA-256.

2. Побудова Дерева Меркла. Агент обробки (Agent-Processor) рекурсивно будує дерево хешів, де листками є, а результатом — кореневий хеш.

3. Логіка дедуплікації. Агент обробки (Agent-Processor) виконує перевірку на рівні файлу та блоків. При виявленні у базі даних, файл вважається

дублікатом, і Агент обробки оновлює список власників без фізичного копіювання даних.

4. Фіксація результатів. Агент зберігання (Agent-Storage) виконує запис унікальних блоків та формує транзакцію в блокчейні.

Візуалізацію процесу обміну повідомленнями та послідовність дій компонентів системи під час реалізації цього сценарію наведено на рисунку 3.2.

Використання Дерева Меркла у поєднанні з блокчейном гарантує, що навіть при зберіганні лише унікальних фрагментів, система зберігає криптографічний доказ цілісності оригінального файлу. Будь-яка зміна навіть одного біта в блоці призведе до зміни кореневого хешу, що буде негайно виявлено під час аудиту.

3.3. Протокол взаємодії агентів при збереженні та дедуплікації даних

Ефективність функціонування розробленої системи залежить від узгодженості дій розподілених компонентів. Для забезпечення надійної комунікації між автономними модулями реалізовано протокол взаємодії, що базується на специфікаціях FIPA-ACL (Agent Communication Language) [28]. Це дозволяє структурувати обмін повідомленнями між Агентом-Інтерфейсом, Агентом-Посередником (Mediator), Агентом-Контролером та іншими вузлами системи, забезпечуючи асинхронність обробки запитів та балансування навантаження .

Процес обробки вхідних даних реалізовано за двома основними сценаріями взаємодії, які визначаються результатами перевірки унікальності файлу.

Сценарій первинного збереження даних (Унікальний файл)

Даний алгоритм ініціюється, коли система отримує файл, кореневий хеш якого відсутній у базі даних (Merkle Roots Database). Послідовність взаємодій агентів виглядає наступним чином:

1. Ініціалізація. Клієнт передає файл через захищений канал Агенту-Інтерфейсу, який перенаправляє потік даних Агенту-Посереднику. Роль посередника полягає у маршрутизації завдань та ізоляції внутрішньої логіки системи від зовнішнього інтерфейсу, що підвищує загальну безпеку архітектури.

2. Обробка. Агент-Посередник передає файл Агенту-Контролеру. Останній виконує сегментацію даних на блоки фіксованого розміру, обчислює хеш-суми та формує Дерево Меркла для отримання кореневого ідентифікатора.

3. Аналіз. Агент-Контролер надсилає запит з отриманим кореневим хешем та списком хешів блоків до Агента-Аналізу. Цей агент виконує пошук у локальних базах даних (Merkle DB та Hash DB).

4. Збереження. Отримавши підтвердження про відсутність дублікатів, Агент-Контролер надає команду Агенту-Даних. Цей агент виконує дві паралельні операції: фізичний запис унікальних блоків у хмарне сховище та генерацію транзакції з метаданими у блокчейні [27].

5. Завершення. Після успішного запису Агент-Даних повертає унікальний вказівник (pointer) через ланцюжок агентів назад до клієнта [30].

Сценарій обробки дубльованих даних

У випадку виявлення надлишковості даних активується оптимізований протокол взаємодії, спрямований на мінімізацію дискових операцій [29].

1. Виявлення дубліката. Агент-Аналізу, отримавши запит від Агента-Контролера, встановлює, що кореневий хеш файлу вже наявний у реєстрі.

2. Перевірка прав доступу. Агент виконує перевірку списку власників (*OwnerList*), асоційованого з даним хешем.

Якщо ідентифікатор поточного користувача (*UserID*) вже присутній у списку, Агент-Аналізу повертає статус «Повний дублікат». Агент-Контролер негайно видаляє тимчасові дані з оперативної пам'яті, не ініціюючи звернень до дискової підсистеми.

Якщо файл існує, але завантажений іншим користувачем, активується процедура підтвердження володіння (Proof of Ownership) [19]. Агент-Аналізу оновлює метадані, додаючи нового користувача до списку доступу. Агент-Даних фіксує цю зміну в блокчейні без фізичного дублювання тіла файлу .

Реалізація такого протоколу дозволяє досягти значної економії ресурсів: пропускна здатність мережі використовується лише для передачі унікальних блоків, а час відгуку системи зменшується за рахунок виключення зайвих операцій запису для популярних файлів.

3.4. Реалізація підсистеми публічного аудиту та верифікації цілісності даних

Забезпечення довіри до збережених даних у хмарному середовищі вимагає впровадження механізму, який дозволяє переконатися у незмінності інформації

без необхідності завантаження повного обсягу даних. У розробленій системі цю функцію виконує підсистема публічного аудиту (Public Auditing), де роль верифікатора делегується незалежній сутності — Сторонньому Аудитору (Third Party Auditor — TPA) [31]. Архітектура підсистеми побудована таким чином, щоб забезпечити конфіденційність даних, тобто аудитор може перевірити цілісність файлу, не отримуючи доступу до його змісту.

Процес аудиту реалізовано за криптографічним протоколом «Виклик-Відповідь» (Challenge-Response) з використанням Дерева Меркла (Merkle Hash Tree). Алгоритм верифікації, адаптований для використання у мультиагентній системі, складається з наступних етапів:

1. Ініціалізація аудиту (Audit Request).

Клієнт надсилає запит до TPA на перевірку конкретного файлу. TPA звертається до хмарного провайдера (CSP) для отримання метаданих: версії файлу (v) та часової мітки (t). Одночасно аудитор отримує еталонний кореневий хеш Меркла (n_0) з блокчейну, який слугує незмінним якорем довіри.

2. Генерація виклику (Challenge Generation).

Для захисту від підробок TPA генерує динамічний запит.

- Спочатку обчислюється унікальне зерно генерації (seed) r на основі еталонного кореневого хешу: $r = H(n_0)$.
- За допомогою генератора псевдовипадкових чисел (PRNG) G формується набір індексів блоків $\{I_j\}$ для перевірки:

$$I_j = G(r, j), j = 1, \ldots, c$$

де c — кількість блоків, що перевіряються. Сформований запит $\{I_j\}$ надсилається до CSP.

3. Формування доказу (Proof Response).

CSP, отримавши індекси, знаходить у локальному Дереві Меркла відповідні блоки та шляхи аутентифікації (sibling paths) — набір сусідніх хешів, необхідних для відновлення кореня. Провайдер надсилає цей набір доказів аудитору.

4. Верифікація доказу (Verification).

TPA виконує математичну перевірку отриманих даних у два кроки:

- Обчислення кореня: На основі надісланих доказів TPA рекурсивно обчислює новий кореневий хеш n'_0 і порівнює його з еталонним.
- Перевірка відповідності: TPA обчислює нове зерно $r' = H(n'_0)$ та відновлює індекси $I'_j = G(r', j)$. Якщо обчислені індекси I'_j збігаються із запитаними I_j, це гарантує, що провайдер надав докази саме для тих блоків, які були запитані, а не підставив заздалегідь заготовлені дані .

Якщо обидві перевірки проходять успішно, цілісність файлу вважається підтвердженою. У випадку розбіжності фіксується факт порушення цілісності, і система генерує сповіщення для власника даних. Такий підхід дозволяє виконувати пакетний аудит (Batch Auditing), перевіряючи дані від різних користувачів одночасно, що знижує комунікаційне навантаження на мережу [32].

3.5. Аналіз безпеки та ефективності запропонованої архітектури

Розроблена система, що поєднує мультиагентний підхід, блокчейн та криптографічні методи перевірки, забезпечує комплексний захист даних на всіх етапах їх життєвого циклу [38]. У цьому підрозділі наведено аналіз стійкості системи до основних векторів атак та оцінку ефективності механізмів дедуплікації й аудиту.

Аналіз безпеки (Security Analysis)

1. Забезпечення конфіденційності даних від аудитора (Privacy-Preserving).

Використання протоколу публічного аудиту створює ризик витоку інформації до третьої сторони (TPA). У запропонованій схемі TPA оперує виключно хеш-сумами та метаданими. Оскільки функція SHA-256 є незворотною (one-way function), відновлення вихідного тексту логів із отриманих доказів є обчислювально неможливим. Це гарантує, що конфіденційна інформація компанії залишається недоступною для аудитора, навіть якщо він скомпрометований.

2. Захист від підміни даних (Tamper-Proofing).

Інтеграція з блокчейном вирішує проблему довіри до адміністратора хмарного сховища. Кореневий хеш Дерева Меркла (n_0), збережений у розподіленому реєстрі, діє як еталон. Будь-яка спроба модифікації, видалення або вставки запису у файл призведе до лавиноподібної зміни хешів аж до кореня. Оскільки змінити запис у блокчейні неможливо через консенсусний механізм Proof-of-Work (або Proof-of-Stake), цілісність даних гарантується математично.

3. Стійкість до атак повторного відтворення (Replay Attack Resistance) [34].

У протоколі аудиту використовується динамічна генерація викликів на основі випадкового зерна ($seed$) та часових міток. Це унеможливлює атаку, при якій недобросовісний провайдер зберігає старі валідні докази (proofs) і надсилає їх аудитору замість виконання реальної перевірки актуального стану файлу [35]. Кожен запит аудиту є унікальним і вимагає нових обчислень .

Аналіз ефективності (Efficiency Analysis) [36]

1. Оптимізація простору зберігання (Storage Efficiency).

Реалізований алгоритм дедуплікації на стороні сервера (Server-side deduplication) дозволяє зберігати лише один фізичний екземпляр унікального блоку даних, незалежно від того, скільки користувачів його завантажили. Для лог-файлів, які часто містять повторювані патерни (заголовки, системні повідомлення), коефіцієнт дедуплікації може досягати значних величин, суттєво знижуючи витрати на хмарне сховище .

2. Ефективність пакетного аудиту (Batch Auditing).

Мультиагентна архітектура дозволяє TPA обробляти запити від багатьох користувачів одночасно. Замість виконання N окремих протоколів перевірки, аудитор може агрегувати запити для однакових файлів або блоків. Це дозволяє зменшити комунікаційні витрати (communication overhead) та навантаження на обчислювальні потужності провайдера порівняно з послідовним аудитом .

3. Розподіл навантаження [37].

Використання спеціалізованих агентів (Interface, Mediator, Control, Analysis, Data) дозволяє розпаралелити процеси обробки. Поки Агент-Даних виконує "важкі" операції шифрування та запису на диск, Агент-Аналізу може обробляти метадані наступного файлу. Така асинхронність забезпечує високу пропускну здатність системи навіть при пікових навантаженнях.

Таким чином, запропонована система не лише вирішує задачу гарантування цілісності даних у недовіреному середовищі, але й пропонує економічно ефективний підхід до зберігання великих обсягів інформації.

ВИСНОВКИ

У магістерській дисертації проведено комплексне дослідження та розв’язано важливу науково-практичну задачу розробки децентралізованої системи безпечного аудиту в хмарних середовищах, що дозволило забезпечити цілісність та конфіденційність журналів подій в умовах високого навантаження. На основі аналізу сучасного стану хмарних технологій було встановлено, що стрімка міграція корпоративних інфраструктур у хмари та перехід до мікросервісної архітектури призвели до формування потоків даних безпеки, які відповідають характеристикам «Великих даних» за моделлю «5V». Доведено, що традиційні централізовані системи моніторингу виявляються неефективними під час масованих кібератак або DDoS-інцидентів, оскільки експоненційне зростання обсягу логів призводить до переповнення буферів і втрати критично важливих доказів. Для вирішення цієї проблеми було обґрунтовано вибір методу дедуплікації на стороні цільової системи (Target-based), що дозволяє перенести обчислювальне навантаження з обмежених ресурсів локальних сенсорів на масштабовані хмарні потужності, забезпечуючи стабільність моніторингу навіть під час пікових навантажень.

Запропонована в роботі гібридна архітектура зберігання успішно поєднує переваги хмарних об'єктних сховищ та технології блокчейн: об’ємні зашифровані «сирі» дані розміщуються поза ланцюжком (Off-chain), тоді як їхні криптографічні відбитки у формі кореневих хешів Меркла фіксуються в розподіленому реєстрі (On-chain). Це створює математично гарантований механізм незмінності історії подій, усуваючи загрозу модифікації журналів адміністраторами провайдера або зловмисниками з високими правами доступу. Розроблена мультиагентна система (MAS), що складається з автономних агентів збору, обробки та зберігання, забезпечує високу живучість інфраструктури аудиту. Завдяки використанню брокера повідомлень та асинхронній взаємодії

компонентів, система здатна ефективно буферизувати дані, запобігаючи втраті записів у разі тимчасової недоступності окремих вузлів або мережевих збоїв.

Окрему увагу в роботі приділено реалізації підсистеми публічного аудиту за участю Стороннього Аудитора (TPA) на основі протоколу «Виклик-Відповідь». Це рішення дозволяє верифікувати цілісність великих масивів даних без їхнього завантаження та без розкриття вмісту логів самому аудитору, що повністю відповідає вимогам конфіденційності. Практичне значення одержаних результатів полягає у розробці життєздатної моделі, яка може бути імплементована на базі Microsoft Azure для створення економічно ефективних та юридично значущих систем зберігання цифрових доказів. Таким чином, розроблена система демонструє комплексний підхід до захисту хмарних даних, поєднуючи сучасні методи оптимізації зберігання з децентралізованими технологіями забезпечення довіри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Antonopoulos N., Gillam L. Cloud Computing. Principles, Systems and Applications. London: Springer-Verlag, 2010. 379 p.
2. Buyya R., Broberg J., Goscinski A. Cloud computing. Principles and Paradigms. New Jersey: John Wiley & Sons, Inc., 2011. 641 p.
3. Mell P., Grance T. The NIST Definition of Cloud Computing. NIST Special Publication 800-145, 2011.
4. Биков В. Ю. Технології хмарних обчислень, ІКТ-аутсорсінг та нові функції ІКТ-підрозділів навчальних закладів і наукових установ. Інформаційні технології в освіті. 2011. № 10. С. 8–23.
5. Глазунова О. Г. Принципи формування «академічної хмари» сучасного університету на основі відкритих програмних платформ. Інформаційні технології і засоби навчання. 2014. № 5 (43).
6. Литвинова С. Г., Спірін О. М., Анікіна Л. П. Хмарні сервіси Office 365: навчальний посібник. Київ: Компринт, 2015. 170 с.
7. Олексюк В. П. Досвід інтеграції хмарних сервісів Google Apps у інформаційно-освітній простір вищого навчального закладу. Інформаційні технології і засоби навчання. 2013. № 3.
8. Олексюк В. П. Проектування моделі хмарної інфраструктури ВНЗ на основі платформи Apache CloudStack. Інформаційні технології і засоби навчання. 2016. № 4.
9. Сейдаметова З. С. та ін. Хмарні технології та освіта. Сімферополь: ДИАЙПІ, 2012. 204 с.
10. Фамілярська Л. Організація освітнього середовища післядипломної педагогічної освіти засобами хмарних сервісів Google. Інформатика та ІТ в навчальних закладах. 2015. № 5/6.

11. Шишкіна М. П. Формування і розвиток хмаро орієнтованого освітньо-наукового середовища вищого навчального закладу: монографія. Київ: УкрІНТЕІ, 2015. 256 с.
12. Зінченко О. В. та ін. Хмарні технології. Київ: Державний університет телекомунікацій (ФОП Гуляєва В. М), 2020.
13. Купрієнко А., Гальчинський Л. Агентна Модель Майнінгу Прав Доступу В Хмарних Середовищах. Scientific Collection «InterConf». 2023. № 184. С. 482–490.
14. Chawla S., Thamilarasu G. Security as a service: real-time intrusion detection in internet of things. Proceedings of the Fifth Cybersecurity Symposium. 2018.
15. Кеті О'Ніл. BIG DATA. Зброя математичного знищення. Як великі дані збільшують нерівність і загрожують демократії. Київ: BookChef, 2019. 336 с.
16. Fern A., Gresty D., Yang M. Big Data Analytics and Cloud Security. IEEE Conference on Communications and Network Security. 2018. P. 1–9.
17. Douceur J. R. et al. Reclaiming space from duplicate files in a serverless distributed file system. Proceedings of the 22nd ICDCS. IEEE, 2002. P. 617–629.
18. Harnik D., Pinkas B., Shulman-Peleg A. Side channels in cloud services: Deduplication in cloud storage. IEEE Security & Privacy. 2010.
19. Halevi S. et al. Proofs of ownership in remote storage systems. Proceedings of the 18th ACM CCS. ACM, 2011. P. 491–500.
20. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. 9 p.
21. Топчій , М., & Гальчинський , Л. (2022). ПІДВИЩЕННЯ РІВНЯ БЕЗПЕКИ СМАРТ-КОНТРАКТІВ В МЕРЕЖІ ETHEREUM ВІД ШАХРАЙСТВА ЗА РАХУНОК ВИКОРИСТАННЯ РЕВЕРСИВНИХ ТОКЕНІВ. Collection of Scientific Papers «ΛΌГОΣ», (November 11, 2022; Paris, France), 71–77. https://doi.org/10.36074/logos-11.11.2022.20
22. Gaetani E. et al. Blockchain-based database to ensure data integrity in cloud computing environments. Proceedings of the 1st Workshop on Cryptocurrencies. 2017.

23. Wang Q. et al. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE TPDS. 2011.

24. Африканський О, М., & Гальчинський Л. Ю. МЕТОДИКА КОНФІДЕНЦІЙНОСТІ ХМАРНИХ ДАНИХ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ БЛОКЧЕЙН. Світ наукових досліджень. Випуск 47: матеріали Міжнародної мультидисциплінарної наукової інтернет-конференції (м. Тернопіль, Україна, м. Ополе, Польща, **17-18 грудня 2025** р.) / за ред. : О. Патряк та ін. ГО “Наукова спільнота”, WSZIA w Opolu. Тернопіль: ФОП Шпак В.Б. 2025. 64-68 с.

25. El Ghazouani M. et al. Blockchain & Multi-Agent System: A New Promising Approach for Cloud Data Integrity Auditing with Deduplication. IJCNIS. 2019.

26. Kreps J., Narkhede N., Rao J. Kafka: a distributed messaging system for log processing. Proceedings of the NetDB. 2011.

27. Xu X., Weber I., Staples M. Architecture for Blockchain Applications. Springer, 2019. 315 p.

28. Bellifemine F., Caire G., Greenwood D. Developing Multi-Agent Systems with JADE. Wiley, 2007. 300 p.

29. Hu Y. et al. A secure and efficient deduplication protocol for cloud storage. Proceedings of the 10th ACM ASIA CCS. 2015.

30. Zheng Q., Xu S. Secure and efficient proof of storage with deduplication. Proceedings of the 2nd ACM CODASPY. 2012.

31. Zhu Y. et al. Dynamic audit services for outsourced storages in clouds. IEEE Transactions on Services Computing. 2013.

32. Yuan J., Yu S. Secure and constant cost public cloud storage auditing with deduplication. IEEE CNS. 2013. P. 145–153.

33. Sharma P. et al. Blockchain-based cloud storage system with CPABE-based access control and revocation process. The Journal of Supercomputing. 2021.

34. Collen A. et al. Ghost-safe-guarding home IoT environments with personalised real-time risk control. Springer. 2018.

35. Kanuparthi A. et al. Hardware and embedded security in the context of internet of things. ACM. 2013.

36. Ganapathy S. et al. Intelligent feature selection and classification techniques for intrusion detection in networks. EURASIP. 2013.

37. Pandey P., Barve A. An Energy-Efficient Intrusion Detection System for MANET. Springer. 2019.
38. Pacheco J., Hariri S. IoT security framework for smart cyber infrastructures. IEEE. 2016.
39. Swan M. Blockchain: Blueprint for a New Economy. O'Reilly Media, 2015. 152 p.
40. Криворучко К. М. Застосування хмарних технологій. Кіровоградський національний технічний університет, 2013.