МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

Другого (магістерського) рівня вищої освіти

на тему: “ Інтеграція блокчейн-технологій для забезпечення безпеки IoT-мереж ”

Виконав: студент 6 курсу групи Іт-61
Спеціальності 126 – „Інформаційні системи та технології”
(шифр і назва)

Дудко Богдан Олесьович
(Прізвище та ініціали)

Керівник: к.т.н., доц. Падюка Р.І.
(Прізвище та ініціали)

Рецензенти: к.т.н., доц. Шеремета Р.Б.
(Прізвище та ініціали)

(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ГЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 "Інформаційні системи та технології"

“ЗАТВЕРДЖУЮ”
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
“____”_____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Дудко Богдану Олесьовичу

1. Тема роботи: «Інтеграція блокчейн-технологій для забезпечення безпеки IoT-мереж»

Керівник роботи Падюка Роман Іванович, к.т.н., доцент.
Затверджені наказом по університету від 28 лютого 2025 року № 140/к-с.
2. Строк подання студентом роботи 05.12.2025 р.
3. Початкові дані до роботи: Характеристика IoT-мережі (топологія, типи пристроїв, канали зв’язку, телеметрія й команди). 2) Поточний рівень безпеки та типові загрози. 3) Вимоги до захисту і продуктивності. 4) Гібридна off-chain/on-chain архітектура та смарт-контракти. 5) Інструментарій реалізації й методика тестування та економічних розрахунків.
4. Зміст розрахунково-пояснювальної записки:
1. Аналіз стану питання в практиці та теорії
2. Обґрунтування, вибір та реалізація інструментарію вирішення задачі
3. Результати вирішення задачі
4. Охорона праці та безпека в надзвичайних ситуаціях
5. Визначення ефективності розробленого інструментарію забезпечення безпеки IOT-мережі на основі блокчейн
Висновки та пропозиції.
Бібліографічний список.
Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): Тема, автор, керівник магістерської роботи; Мета,завдання, об’єкт, предмет дослідження; Проблематика та ризики сучасних IoT систем; Концепція рішення з довірчим реєстром подій; Рівнева модель інтегрованої IoT архітектури

Протокол перевірки цілісності та доказовості дій; Схема підключення смарт контрактів у контурі IoT; Схема підключення смарт контрактів у контурі IoT; Стартове налаштування системи керування IoT; Формування приміщень як логічних зон моніторингу; Реєстрація пристроїв та керування переліком вузлів; Картка пристрою та контроль його параметрів; Дані пристрою та прив'язка смарт контрактів; Мобільний клієнт як канал оперативного реагування; Сценарії тривоги та керування сенсорами в додатку; Підсумкові результати та ефективність впровадження.

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Падюка Р.І., доцент кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 03 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Приміт-ка
1	*Написання першого розділу та означення головних завдань роботи*	03.03.-01.04.25	
2	*Виконання другого розділу та формування головних показників для розрахунків*	02.04.-01.05.25	
3.	*Виконання третього розділу та формування початкових даних*	02.05.-01.06.25	
4.	*Виконання четвертого розділу та узагальнення отриманих результатів магістерської роботи*	02.06.-01.09.25	
5.	*Вартісне оцінення ефективності пропозицій роботи*	02.09.-01.10.25	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	02.10.-01.11.25	
7.	*Завершення роботи в цілому*	02.11.-05.12.25	

Студент _____ Дудко Б.О.
(підпис)

Керівник роботи _____ Падюка Р.І.
(підпис)

УДК 004.738.5:004.056:003.26

Інтеграція блокчейн-технологій для забезпечення безпеки IoT-мереж – Дудко Б.О. Магістерська робота. Кафедра ІТ. – Дубляни, Львівський НУВМБ ім. С.З. Гжицького, 2025.

93 с. текст. част., 44 рис., 1 табл., 14 арк. графічної частини, 32 літ. джерел

У роботі досліджено інтеграцію блокчейн-технологій для підвищення безпеки IoT-мереж. Запропоновано гібридну архітектуру, у якій телеметрія обробляється у швидкому off-chain контурі, а критичні події, керувальні команди та зміни статусів пристроїв фіксуються у смарт-контрактах як незмінний журнал аудиту. Розроблено прототип із веб-інтерфейсом налаштування та мобільним клієнтом, реалізовано реєстрацію пристроїв, контроль доступу, шифрування та перевірку цілісності. Проведено тестування сценаріїв тривоги й керування та порівняльну оцінку з традиційними методами. Показано зростання доказовості й стійкості до атак за прийнятних накладних витрат, а також економічну доцільність впровадження.

Ключові слова: IoT, блокчейн, смарт-контракти, контроль доступу, аудит подій, цілісність даних, кібербезпека.

Keywords: IoT, blockchain, smart contracts, access control, event audit, data integrity, cybersecurity.

ЗМІСТ

ВСТУП

Стрімке зростання кількості пристроїв Інтернету речей (IoT) змінює підходи до побудови мережевої інфраструктури в промисловості, енергетиці, транспорті, «розумних» будівлях, медицині та побутових системах. IoT-мережі забезпечують збір і передавання даних у реальному часі, автоматизацію процесів та віддалене керування об’єктами, однак одночасно формують новий клас ризиків інформаційної безпеки. Характерні для IoT гетерогенність пристроїв, тривала експлуатація без оновлень, обмежені обчислювальні ресурси, використання бездротових каналів зв’язку та масовість розгортання суттєво ускладнюють забезпечення конфіденційності, цілісності й доступності даних, а також керування ідентичностями та правами доступу.

Традиційні централізовані моделі захисту, що спираються на єдиний довірений центр (сервер автентифікації, централізований реєстр ключів, центральний брокер повідомлень), у середовищі IoT нерідко призводять до появи «єдиної точки відмови» та уразливостей керування довірою. Компрометація вузла керування або вразливість програмного забезпечення може спричиняти каскадні наслідки: підміну телеметрії, несанкціоноване керування виконавчими механізмами, порушення доступності сервісів чи формування ботнетів із тисяч пристроїв. Додатковою проблемою є необхідність забезпечити простежуваність операцій і доказовість подій безпеки: у багатьох застосуваннях важливо не лише заблокувати атаку, а й мати незмінюваний журнал дій для аналізу інцидентів та аудиту.

Перспективним напрямом підвищення довіри в IoT-інфраструктурах є інтеграція блокчейн-технологій. Децентралізований реєстр із криптографічним зв’язуванням блоків, механізмами узгодження стану та можливістю виконання смартконтрактів потенційно дозволяє реалізувати незмінюваний аудит подій, розподілене керування ідентичностями, контроль доступу на основі політик, захист цілісності телеметрії та керованість життєвого циклу пристроїв. Водночас пряме перенесення класичних блокчейн-рішень у IoT є проблемним через

витрати на консенсус, затримки підтвердження транзакцій, обмеження пропускної здатності, вимоги до приватності та потребу узгоджувати роботу з периферійними обчисленнями (edge/fog). Тому актуальним є розроблення підходів, які поєднують переваги блокчейну із реальними ресурсними та мережевими обмеженнями IoT, формуючи практично застосовні архітектури захисту.

Метою магістерської роботи є обґрунтування та розроблення підходу до інтеграції блокчейн-технологій у IoT-мережі для підвищення рівня інформаційної безпеки, зокрема щодо керування довірою, цілісності даних, автентифікації пристроїв, контролю доступу та аудиту подій. Для досягнення мети передбачається виконати аналіз загроз і типових векторів атак на IoT-мережі, дослідити існуючі моделі застосування блокчейну в кібербезпеці та обмеження їх використання у середовищі IoT, сформулювати вимоги до цільової архітектури з урахуванням масштабованості, затримок, обчислювальних ресурсів і приватності, запропонувати архітектурне рішення інтеграції блокчейну з компонентами IoT (пристрої, шлюзи, edge-вузли, хмарні сервіси), а також розробити механізми реєстрації пристроїв, розподіленого журналювання подій і виконання політик доступу на основі смартконтрактів. Завершальним етапом є експериментальна перевірка працездатності запропонованого підходу та оцінювання його впливу на показники безпеки й продуктивності (затримки, пропускна здатність, навантаження на вузли, стійкість до типових сценаріїв атак).

Об’єктом дослідження є процеси забезпечення інформаційної безпеки в IoT-мережах у умовах гетерогенності, масштабованості та обмежених ресурсів пристроїв.

Предметом дослідження є моделі, методи та засоби інтеграції блокчейн-технологій (розподілений реєстр, механізми консенсусу, смартконтракти, криптографічні протоколи) для реалізації керування довірою, цілісності даних, автентифікації й контролю доступу в IoT-інфраструктурах.

Методологічну основу роботи становлять методи системного аналізу та проєктування мережевих систем, моделювання потоків даних і загроз (у межах обраної моделі порушника), принципи криптографічного захисту інформації, підходи до побудови розподілених систем та аналізу їхньої узгодженості, а також методи експериментального дослідження й порівняльного оцінювання запропонованих рішень за критеріями безпеки та ефективності.

Очікувана наукова новизна полягає в удосконаленні підходу до забезпечення безпеки IoT-мереж шляхом поєднання розподіленого реєстру з периферійними обчисленнями та політиками доступу, що формалізуються і виконуються смартконтрактами, з урахуванням ресурсних обмежень пристроїв і вимог до масштабованості.

Практичне значення роботи полягає у можливості застосування запропонованої архітектури та набору механізмів як основи для побудови прототипів або модулів безпеки в корпоративних та промислових IoT-розгортаннях, де критичними є питання довіри до даних, відстежуваності операцій та виключення централізованих точок компрометації.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ПИТАННЯ В ТЕОРІЇ ТА ПРАКТИЦІ

1.1. Архітектурні засади IoT-мереж: компоненти, домени та потоки даних

Інтернет речей (IoT) у сучасному розумінні розглядається як сукупність мережевих взаємодій між фізичними об’єктами, вимірювальними/виконавчими пристроями та прикладними сервісами, що забезпечують збирання, передавання, оброблення й використання даних для прийняття рішень у цифрових і фізичних процесах. Для формалізації предметної області доцільно спиратися на технічно орієнтоване визначення «IoT-пристрою», відповідно до якого пристрій має принаймні один перетворювач (сенсор або актуатор), що взаємодіє з фізичним середовищем, і принаймні один мережевий інтерфейс для цифрового обміну даними [1]. Таке визначення є принципово важливим для подальшого аналізу безпеки, оскільки безпекові вимоги та вектори атак виникають одночасно у двох площинах: у фізичній (вплив на процес/об’єкт керування) та в інформаційній (вплив на дані, канали, облікові дані, сервіси).

Архітектура IoT-мереж зазвичай виходить за межі «підключених сенсорів» і формується як багаторівнева екосистема, де периферійні пристрої працюють у безпосередній близькості до фізичних об’єктів, а аналітика, керування й інтеграції часто зосереджені в хмарних або корпоративних середовищах. Показовим узагальненням є доменна модель, у якій виділяються щонайменше п’ять взаємопов’язаних зон: рівень користувача (user layer), мережа наближення/польовий сегмент (proximity network), публічні мережі (public network), хмарний провайдерський домен (provider cloud) і корпоративна мережа (enterprise network) [2]. Така декомпозиція дозволяє системно описати, де саме виникають межі довіри, які елементи мають різні власники/адміністратори, та як формується «ланцюг» оброблення даних від сенсора до бізнес-рішення, що для IoT є типовим.

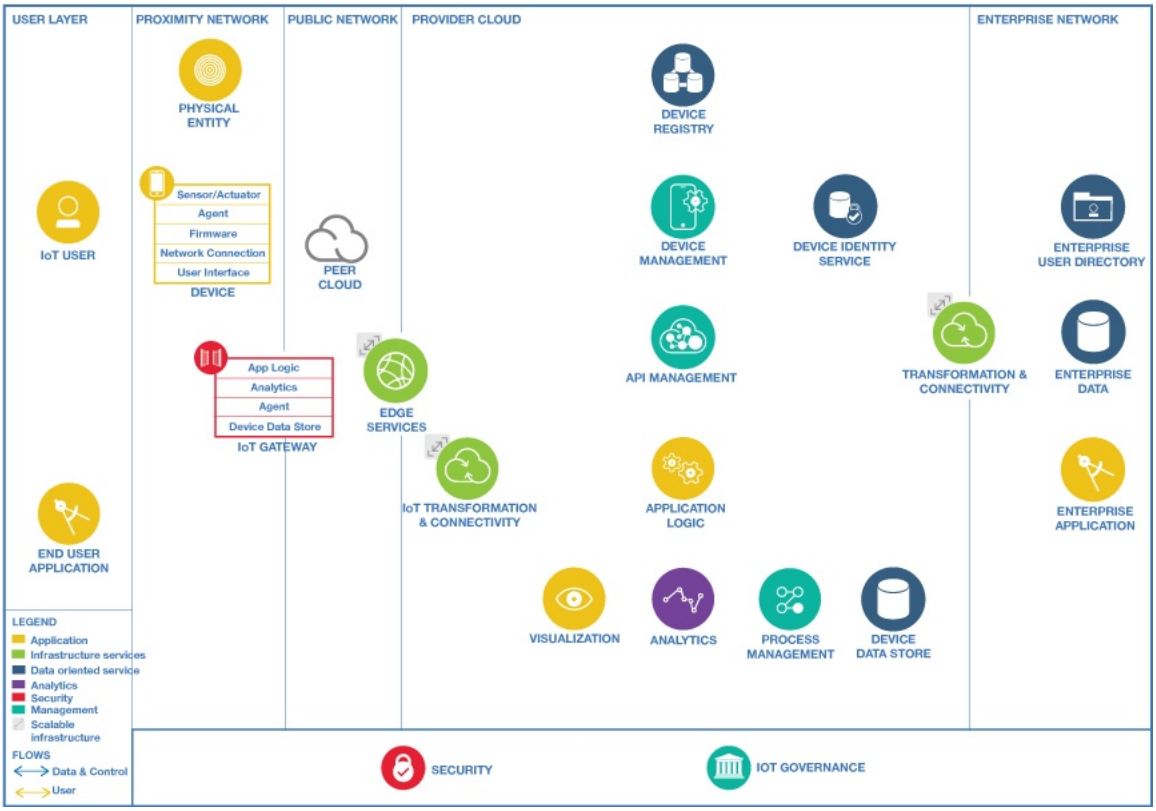


Рисунок 1.1 – Елементи IoT-рішення у доменній моделі (рівень користувача, proximity/public network, provider cloud, enterprise network)[2].

На практиці наведена доменна модель конкретизується функціональними підсистемами, які забезпечують «життєздатність» IoT-екосистеми: реєстр/каталог пристроїв, керування ідентичностями, керування конфігураціями та прошивками, API-шлюзи, підсистеми збору телеметрії, сховища даних, потокову обробку, аналітику й візуалізацію, а також інтеграцію з корпоративними даними та прикладними системами [2]. У такій логіці IoT є не тільки мережею «малих» вузлів, а повноцінною розподіленою інформаційною системою, що включає компоненти керування безпекою та управлінням (governance) як наскрізні механізми на всіх рівнях.

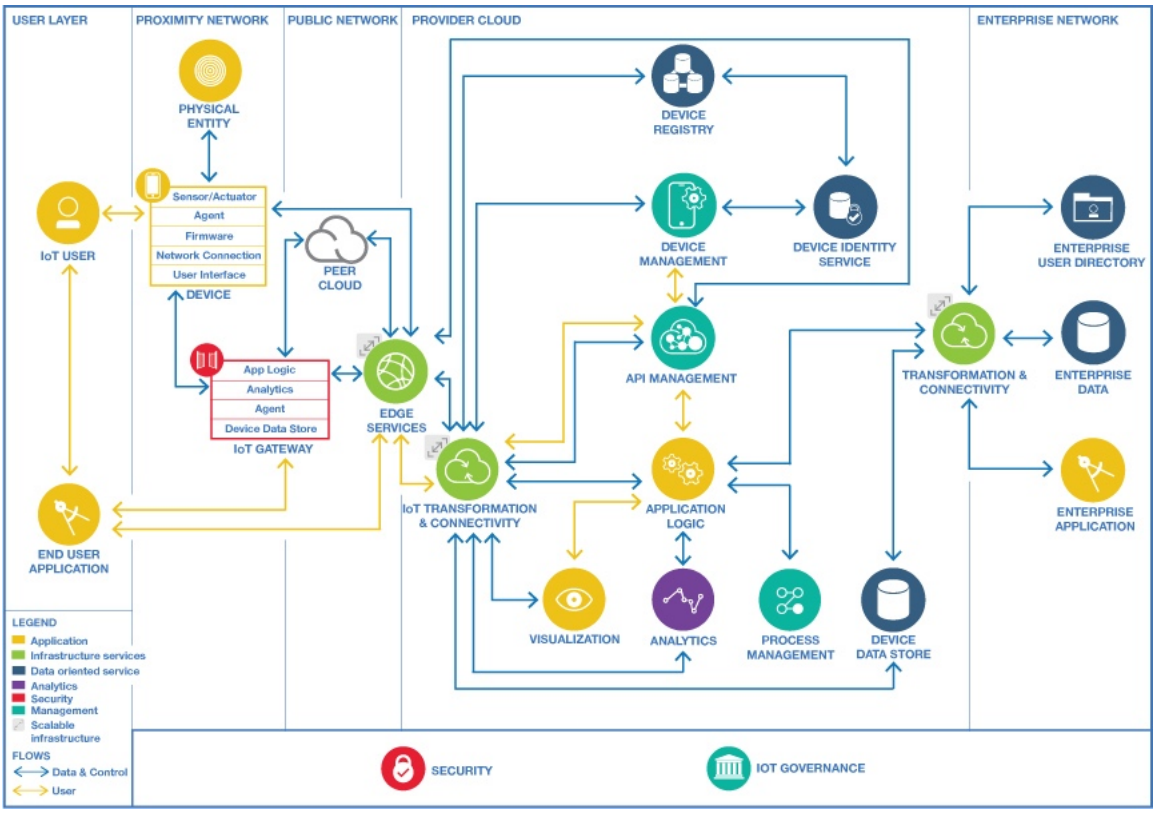


Рисунок 1.2 – Компоненти хмарної підтримки IoT: керування пристроями, ідентичністю, API, аналітикою, сховищами та інтеграцією з підприємством [2].

У контексті подальшого дослідження безпеки IoT-мереж критичною є також типологія потоків даних та їх напрямів. Для більшості сценаріїв характерні телеметрійні потоки (дані вимірювань), керувальні потоки (команди до актуаторів), а також потоки конфігурації/адміністрування (параметри мережевого доступу, політики, оновлення ПЗ). На рівні топології ці потоки часто описують через «north–south» взаємодію (польовий сегмент ↔ хмара/центр) і «east–west» взаємодію (обмін у межах локального сегмента, між сервісами/шлюзами/підсистемами на краю) [3]. Збільшення масштабів IoT призводить до того, що «край мережі» (edge) перестає бути лише транспортною ланкою й фактично стає місцем виконання політик, агрегування даних, кешування, локальної аналітики та протокольної трансляції. Саме тому gateway/edge-компоненти часто визначають реальну поверхню атак: вони зазвичай мають більше ресурсів, більше протоколів і більше «довіри» з боку

центральних сервісів, а отже, можуть стати точкою компрометації із значним радіусом впливу.

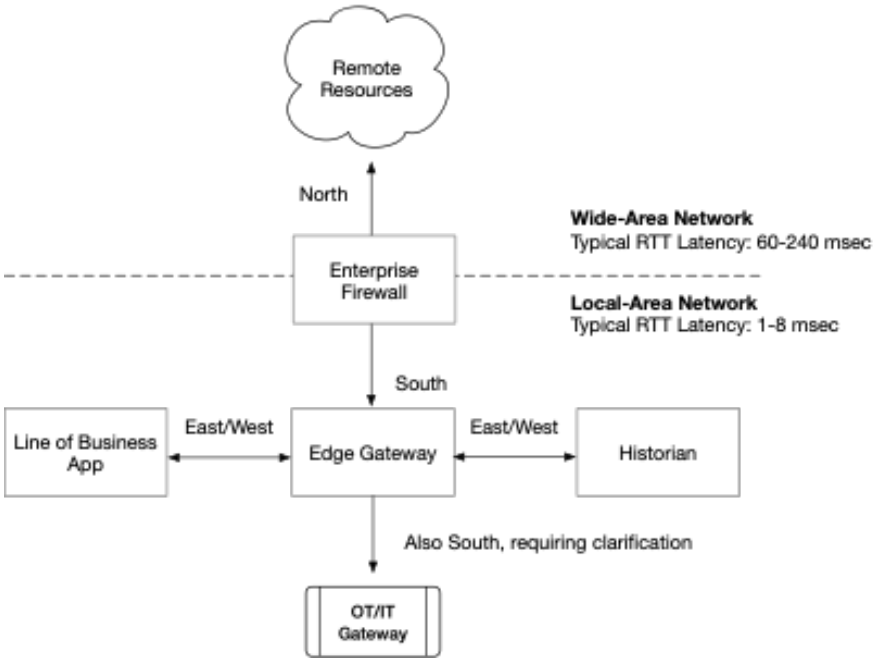


Рисунок 1.3 – Напрями потоків даних у промисловому IoT та роль edge gateway між LAN і WAN [3].

Окремо слід враховувати, що багато IoT-вузлів належать до класу обмежених (constrained) пристроїв і працюють у мережах із втратами, низькою пропускною здатністю та високими затримками. Саме на це орієнтовано спеціалізовані протоколи прикладного рівня, зокрема CoAP, який визначається як «спеціалізований web-трансфер протокол для застосування на обмежених вузлах і в обмежених (наприклад, low-power, lossy) мережах» [4]. Технологічна мотивація цих протоколів безпосередньо впливає на безпеку: зменшення накладних витрат і спрощення стеку часто підвищують ризик помилок реалізації, зміщують акценти з «криптографічної розкішності» до мінімально достатнього захисту та роблять практики керування ключами/сертифікатами складними для масового впровадження.

Паралельно, для масштабованих середовищ широко використовується парадигма обміну повідомленнями publish/subscribe. Протокол MQTT у своїй стандартизованій версії описує механізм взаємодії клієнтів через брокер із

підтримкою публікації та підписки на теми, що добре узгоджується з розподіленою природою IoT та потребою у доставці телеметрії багатьом споживачам [5]. Однак така архітектура концентрує довіру в брокері й супутніх службах ідентифікації, що, своєю чергою, створює вимоги до контролю доступу, сегментації тем, ізоляції тенантів, а також до аудиту й незаперечності (non-repudiation) дій у системі — аспекти, які стають центральними при інтеграції блокчейн-механізмів у наступних розділах роботи.

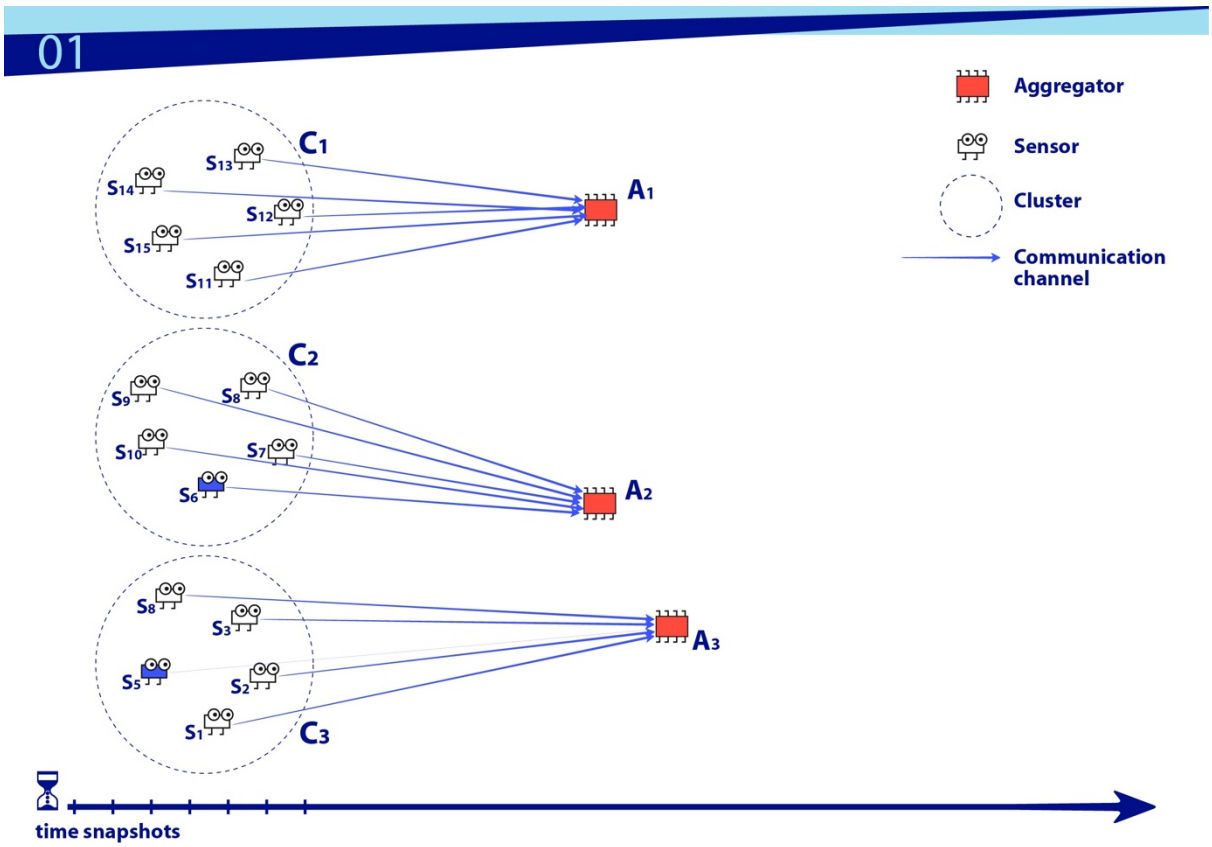


Рисунок 1.4 – Примітиви “Network of Things”: сенсори, кластери, агрегатори та канали зв’язку [6].

Для узгодженого опису зв’язків між «речами», обчислювальними вузлами та каналами корисною є також модель «Network of Things», у якій виділяються примітиви сенсорів, агрегаторів та комунікаційних каналів, що групуються в кластери та формують проміжні дані до прийняття рішень [6]. Важливо, що така модель природно підкреслює часовий аспект (time snapshots) і залежність якості результатів від цілісності даних та надійності каналу, а це

напряму пов’язано з вимогами до журналювання, синхронізації подій і виявлення підміни/повтору повідомлень, які є типовими проблемами IoT.

Архітектурна специфіка IoT підсилюється ще однією практичною обставиною: технологічний життєвий цикл пристроїв зазвичай значно довший, ніж життєвий цикл прикладних сервісів, а отже, питання оновлюваності, керування конфігурацією, інвентаризації та підтримки безпеки стають системоутворюючими. Саме тому в інженерних підходах до IoT-кібербезпеки фіксується потреба у «базових» можливостях пристрою, які мають підтримувати типові заходи захисту організації. Прикладом є NISTIR 8259A, де запропоновано ядро кібербезпекових можливостей IoT-пристрою як «набір можливостей, загалом потрібних для підтримки поширених контролів кібербезпеки» [7]. У термінах архітектури це означає, що безпека не може бути додатком «зовні»; вона вимагає вбудованих можливостей і узгоджених сервісів керування (ідентифікація, конфігурування, оновлення, захист даних, контроль інтерфейсів), які мають бути відображені в архітектурній моделі системи.

У підсумку, IoT-мережа в теорії та практиці описується як багаторівнева, багатодоменна й гетерогенна система, де периферійні вузли, шлюзи, публічні канали, хмарні платформи та корпоративні сервіси формують єдиний контур оброблення даних і керування. Для подальшого аналізу стану питання інтеграції блокчейн-технологій ключовим є висновок про наявність множинних меж довіри, різних адміністративних доменів та складних потоків даних, що потребують перевірюваності, цілісності, відстежуваності подій і узгоджених механізмів ідентифікації. Саме ці характеристики роблять IoT-мережі природним кандидатом для впровадження розподілених механізмів довіри, зокрема блокчейн-реєстрів і смартконтрактів, але водночас висувають вимоги до продуктивності, затримок, енергоефективності та керованості, які мають бути враховані в постановці задачі нашої магістерської роботи.

1.2 Аналіз загроз, вразливостей і наявних підходів до забезпечення безпеки IoT-мереж

Сучасні IoT-мережі формують складну кіберфізичну екосистему, у якій взаємодіють сенсори, виконавчі пристрої, шлюзи, хмарні сервіси, мобільні застосунки та зовнішні інтеграції. Їхня специфіка полягає в тому, що наслідки інцидентів виходять за межі традиційного “порушення конфіденційності даних” і можуть напряму впливати на фізичні процеси, безпеку людей, доступність критично важливих сервісів та стабільність інфраструктури. Саме тому загрози для IoT доцільно розглядати як багатовимірні: технологічні (вразливості протоколів і ПЗ), організаційні (недостатня керованість активами та оновленнями), а також ланцюгові (ризики постачання компонентів і сервісів), що узгоджується з підходом risk-based управління ризиками.

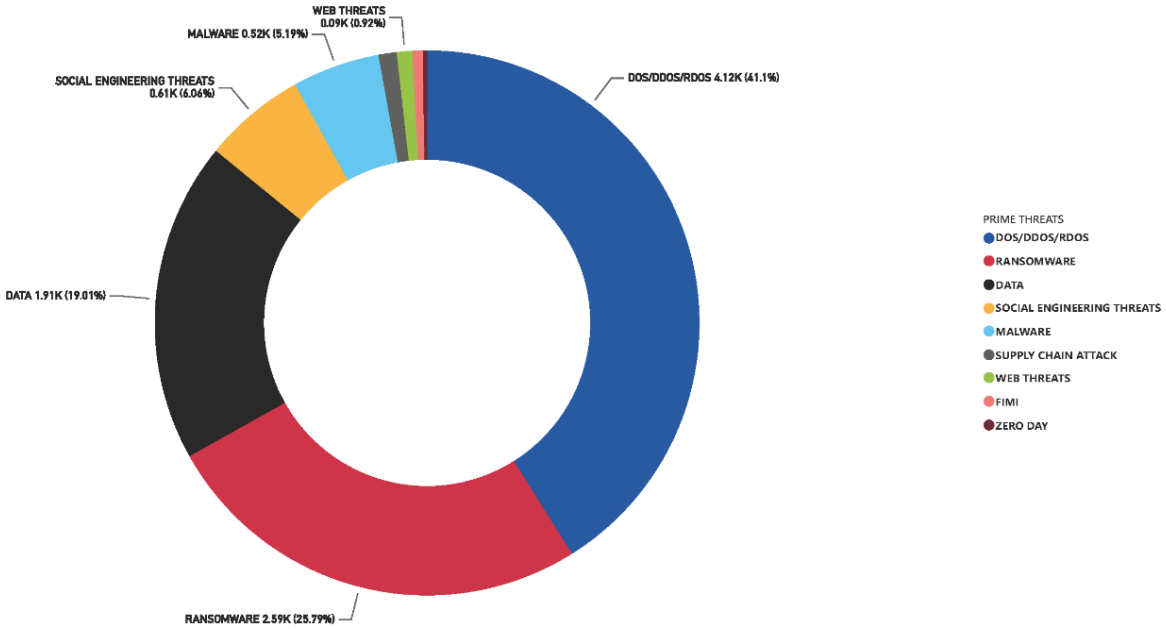


Рисунок 1.5 – Основні типи загроз у сучасному кіберпросторі (узагальнення prime threats) за матеріалами ENISA.[1].

Узагальнення актуального ландшафту загроз демонструє, що для організацій найбільш суттєвими залишаються атаки на доступність, шкідливе ПЗ, програми-вимагачі, соціальна інженерія, атаки на дані, маніпуляції інформацією та атаки на ланцюг постачання, які в реальних інцидентах часто поєднуються в межах одного сценарію компрометації [1]. Для IoT-мереж це означає, що компрометація навіть малопотужного пристрою (камери, датчика,

роутера, шлюзу) може бути “точкою входу” як для подальшого переміщення в мережі (lateral movement), так і для перетворення пристрою на частину ботнету, або ж для фальсифікації телеметрії й керуючих команд. При цьому характерна риса IoT — наявність великої кількості однорідних пристроїв — призводить до “масштабування ризику”: одна типова помилка конфігурації чи дефект прошивки відтворюється сотнями або тисячами екземплярів.

З практичної точки зору, ключовим драйвером інцидентів у IoT є дисбаланс між темпами розгортання пристроїв і можливостями їхнього захисту протягом життєвого циклу. У типовій архітектурі IoT присутні компоненти з обмеженими ресурсами, довгим строком експлуатації, складним або дорогим оновленням, а також різними власниками та зонами відповідальності (виробник, інтегратор, оператор мережі, власник даних). Це ускладнює застосування “класичних” підходів кіберзахисту, орієнтованих на керовані робочі станції/сервери. У даному контексті показовою є концептуальна модель NIST, де підкреслено, що IoT-пристрої по-іншому впливають на управління ризиками: вони взаємодіють з фізичним світом; часто не можуть бути доступними, керованими й моніторинговими так само, як звичайні IT-пристрої; їхні вбудовані механізми захисту можуть бути менш доступними або менш ефективними [2]. Це прямо відображається на виборі заходів контролю та на необхідності перегляду політик експлуатації.

1.3 Практичні моделі інтеграції блокчейн-технологій у IoT-мережі для підвищення безпеки

Інтеграція блокчейн-технологій у мережі Інтернету речей (IoT) розглядається як відповідь на системну проблему довіри в гетерогенному середовищі з великою кількістю пристроїв, доменів та власників. Традиційні підходи кіберзахисту IoT (централізовані PKI, сервери журналювання, шлюзи авторизації, керування політиками доступу) мають виразні ризики: єдина точка відмови, складність міждоменної взаємодії, обмежена прозорість аудитів, а

також високі витрати на підтримку узгоджених реєстрів станів і прав. У цьому контексті блокчейн виступає не як «універсальний захист», а як механізм формування спільного незмінного (tamper-evident) журналу подій і транзакцій між учасниками без потреби в центральному довіреному сховищі. Саме така постановка підкреслюється у технічному огляді NIST: блокчейни є розподіленими цифровими реєстрами криптографічно підписаних транзакцій, згрупованих у блоки, що послідовно зв'язуються хешами, унаслідок чого спроби модифікації стають помітними та практично невигідними при нормальній роботі мережі [1].

Ключовою практичною ідеєю інтеграції блокчейну в IoT є перенесення критичних для довіри аспектів (ідентичність, повноваження, цілісність, незаперечність подій, узгодження станів) у спільний реєстр, тоді як «важкі» дані та контент (телеметрія, файли політик, прошивки, медіа) доцільно залишати поза блокчейном, використовуючи схеми "off-chain storage + on-chain commitments" (збереження поза ланцюгом із фіксацією в ланцюзі хеш-міток/дайджестів). Така декомпозиція дозволяє зменшити навантаження на вузли, уникнути вибухового росту леджера та одночасно забезпечити доказовість цілісності даних. У практичних рішеннях для IoT, особливо в корпоративних та міжорганізаційних сценаріях, переважає permissioned-підхід: участь у мережі обмежується визначеними організаціями, а контроль доступу й політики управління стають частиною моделі довіри. NIST прямо розрізняє permissionless і permissioned категорії, наголошуючи, що за наявності юридичних та організаційних контурів довіри permissioned-мережі можуть забезпечувати швидший і менш затратний консенсус, а також краще керовані рівні приватності транзакцій [1].

Практичні архітектури інтеграції можна описати як багаторівневі композиції «пристрій-edge/fog-блокчейн-сервіси-хмара», де блокчейн-вузли розміщують не на кожному сенсорі, а на більш ресурсних компонентах: edge-шлюзах, проміжних контролерах, локальних серверах або керованих контейнерних вузлах. Це прямо узгоджується з типовими ресурсними обмеженнями IoT та необхідністю досягати низьких затримок на місці

виникнення подій. У таких композиціях роль блокчейну полягає у фіксації: реєстрації/виведення з експлуатації пристрою; актуалізації атрибутів (власник, домен, клас безпеки); запису рішень контролю доступу; доказовості подій (наприклад, зміни конфігурації); прив'язки телеметрії до незмінного часово-послідовного контексту.

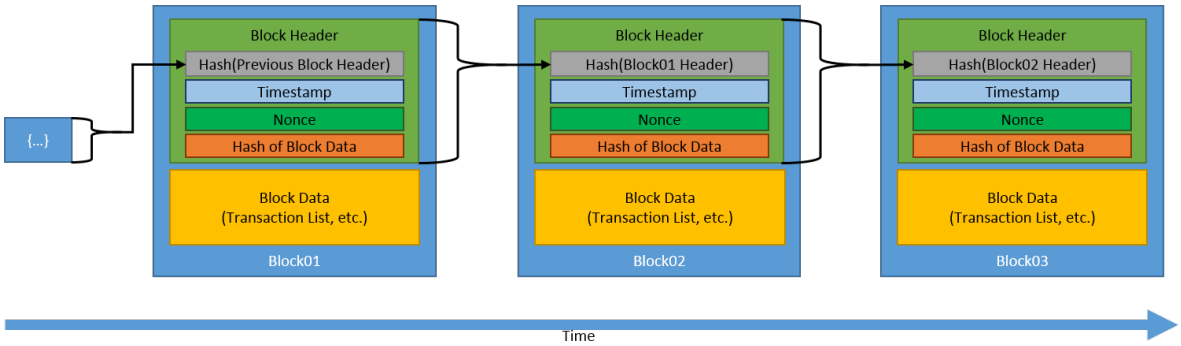


Рисунок 1.6 – Узагальнена схема зв'язування блоків у блокчейні [1]

Окремого значення набувають механізми керованої ідентичності та членства у permissioned-мережах. У практиці Hyperledger Fabric, що часто обирається для IoT-сценаріїв через модульність і корпоративну орієнтацію, ідентичності учасників підтримуються через інфраструктуру сертифікації та абстракцію Membership Service Provider (MSP). Документація Fabric підкреслює, що MSP є механізмом, який дозволяє ідентичності бути визнаною мережею без розкриття приватного ключа, а довіра до сертифікаційних центрів задається конфігурацією MSP організації [3].

У свою чергу, Fabric CA забезпечує випуск і керування сертифікатами та може інтегруватися у контури управління організації, що є критичним для життєвого циклу IoT-пристроїв (первинна ініціалізація, ротація ключів, відкликання, аудит).

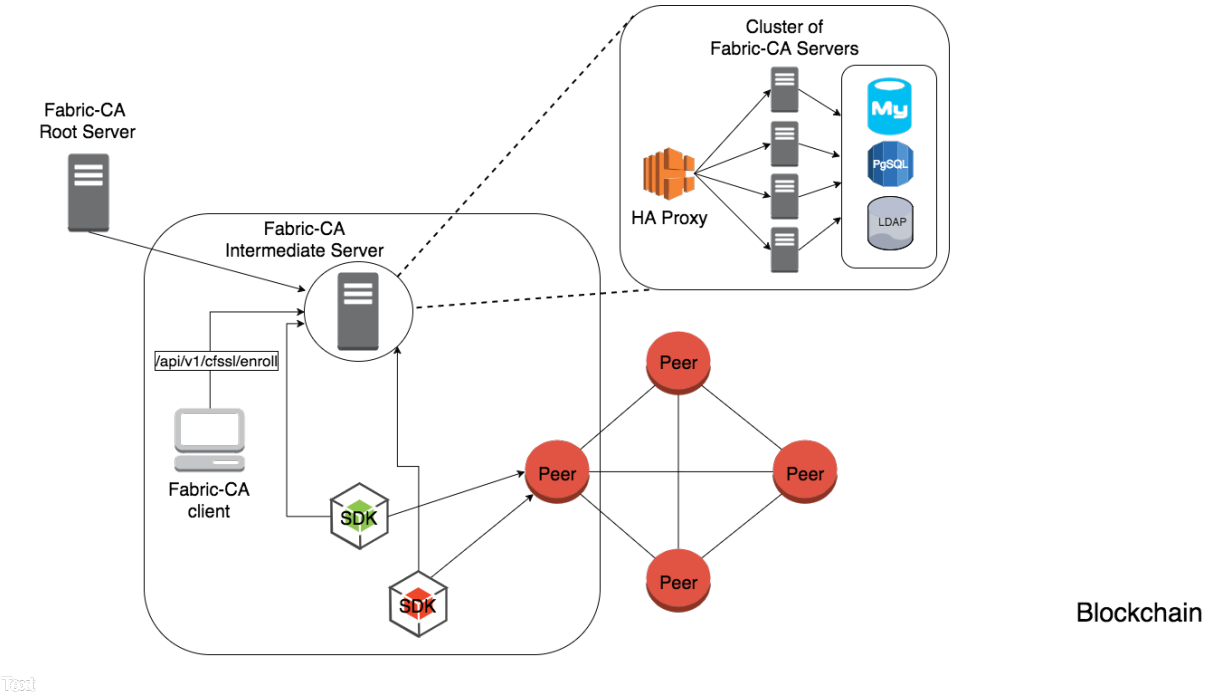


Рисунок 1.7 – Схема взаємодії Hyperledger Fabric CA з компонентами permissioned-мережі [2].

Паралельно з «класичним» PKI-підходом, у теорії та практиці активно розвивається напрям децентралізованої ідентичності (SSI), що дозволяє зменшити залежність від централізованих реєстраторів та підвищити автономність суб'єктів. Стандарт W3C DID визначає децентралізовані ідентифікатори як такі, що можуть бути пов'язані з DID-документом та розміщені/якорені у «verifiable data registry» (якою може бути блокчейн або інша розподілена система), забезпечуючи криптографічно перевірювані методи автентифікації та сервісні кінцеві точки [6].

У застосуванні до IoT це відкриває практичну можливість для «паспортів пристроїв»: DID як стабільний ідентифікатор, DID-документ як джерело ключів/метаданих, а також verifiable credentials як доказ атрибутів (належність домену, сертифікація, рівень довіри). Утім, у реальних мережах часто застосовуються гібридні моделі, де DID/VC використовуються на рівні додатків і взаємодії доменів, а внутрішньодоменні операції підкріплюються permissioned-блокчейном із MSP/CA.

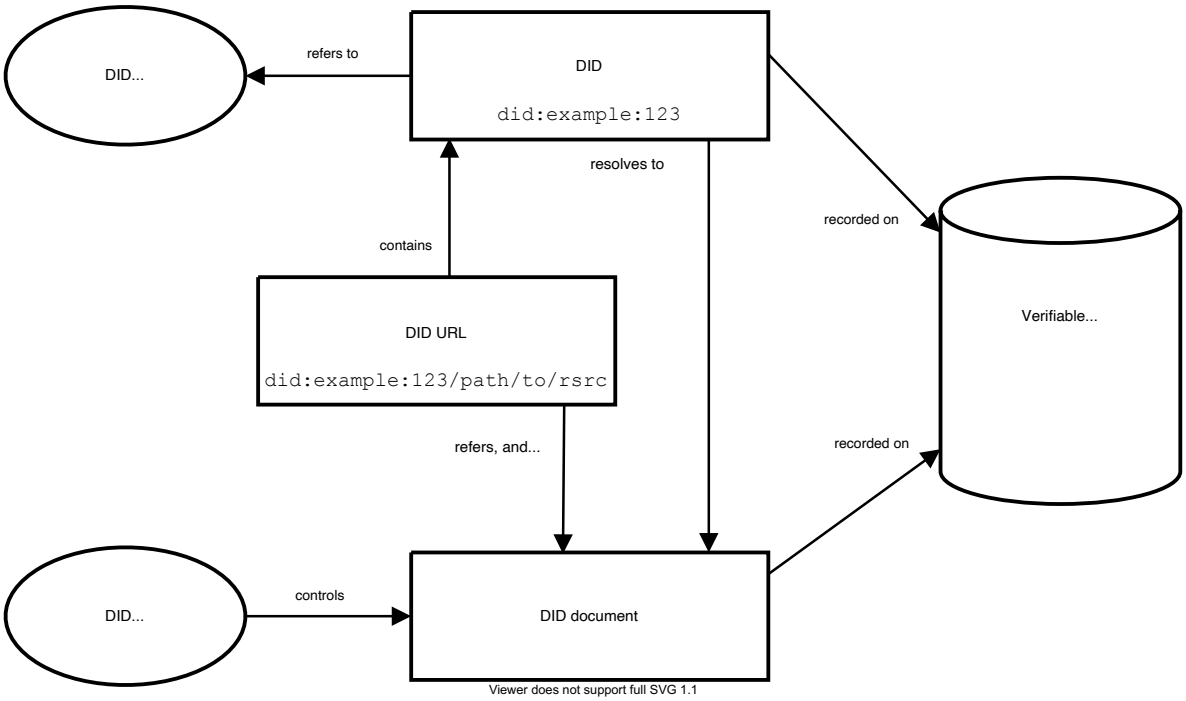


Рисунок 1.8 – Загальна архітектура DID [6].

Найбільш прикладною площиною інтеграції блокчейну в IoT є контроль доступу та аудит рішень. У класичних IoT-реалізаціях контроль доступу або жорстко централізований, або фрагментований між виробниками та доменами, що ускладнює міждоменні сценарії (наприклад, розумний будинок—постачальник сервісу—страхова компанія; промисловий майданчик—підрядник—сертифікаційний орган тощо). Показовим прикладом практичного підходу є робота Sun та ін. (IEEE Access, 2021), де запропоновано міждоменну систему контролю доступу на основі Hyperledger Fabric: для кожного IoT-домену формується локальний ledger, edge-пристрої виконують роль Policy Enforcement Point (PEP) та «фільтра» запитів, тоді як частина IoT-пристроїв бере участь у прийнятті рішень як Policy Decision Points (PDP). При цьому на блокчейн виносяться атрибути та дайджести файлів політик, а самі файли політик зберігаються поза ланцюгом (у прикладі згадується IPFS як позаланцюгове сховище) — тобто реалізовано характерний для IoT принцип мінімізації «важких» даних у леджері [7].

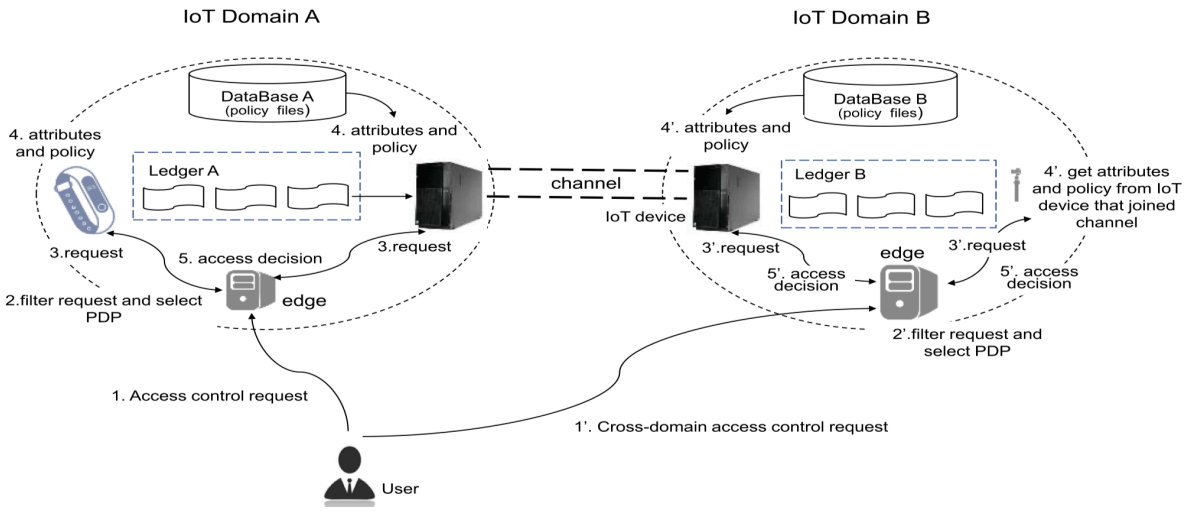


Рисунок 1.9 – Архітектура блокчейн-орієнтованої системи контролю доступу в IoT (міждоменний сценарій із локальними ledger та edge-рівнем) [7].

Така модель важлива тим, що підвищує стійкість до підміни політик і рішень, забезпечує відтворюваність аудиту, а також дає основу для формального узгодження між доменами через канали.

Для практичної реалізації таких сценаріїв критичною є підтримка приватності та розмежування даних між учасниками. У Fabric це вирішується через механізми каналів та private data collections: підмножина організацій на каналі може узгоджувати й зберігати приватні дані так, щоб інші учасники каналу не бачили вміст, але могли перевіряти коректність транзакцій на рівні хешів/метаданих [4].

Для IoT це означає можливість, наприклад, зберігати приватні телеметричні атрибути або конфігурацію пристрою в обмеженому колі, залишаючи решті учасників лише необхідні для консенсусу «сліди» у вигляді дайджестів. Це узгоджується з вимогами промислових і медичних систем, де конфіденційність даних так само важлива, як їхня цілісність.

Ще один фундаментальний практичний аспект – вибір консенсусу та організація потоку транзакцій. У permissioned-мережах для IoT часто обирають менш ресурсомісткі механізми узгодження. У Fabric важливу роль має ordering service; документація наголошує на рекомендованому використанні Raft як реалізації сервісу впорядкування транзакцій, що формує узгоджений порядок

блоків для каналів і дозволяє будувати більш децентралізовані конфігурації з участю кількох організацій [5].

На практиці це дає змогу збалансувати швидкодію та керованість: IoT-рішення можуть формувати локальні доменні блокчейни з помірною кількістю валідуючих вузлів, реалізуючи принцип «блокчейн ближче до edge», щоб зменшити затримки та залежність від хмари.

З погляду безпеки життєвого циклу, блокчейн-інтеграція доповнює й інші критичні процеси IoT, зокрема оновлення прошивок та конфігурацій. Вразливості IoT-пристроїв часто пов’язані з тривалим життєвим циклом і складністю безпечних оновлень; тому підходи до secure update розвиваються на рівні стандартів. Архітектура SUIT (IETF) у RFC 9019 описує мотиви та загальну модель захищеного оновлення прошивок з використанням маніфестів і криптографічного захисту, орієнтовану на ресурсно обмежені пристрої [8].

У таких сценаріях блокчейн може відігравати роль незмінного реєстру: фіксувати хеші дозволених/розповсюджених образів, події застосування оновлень, факти відповідності версій, а також ланцюжок поставок (supply chain) для компонентів. Водночас важливо підкреслити, що блокчейн не замінює SUIT-процесів, а радше надає додатковий механізм прозорості та доказовості для багатосторонніх екосистем, де беруть участь виробник, оператор, сервіси обслуговування та регулятор.

У підсумку, практика інтеграції блокчейну в IoT-мережі демонструє, що найбільш ефективними є гібридні архітектури, які: локалізують консенсус та вузли на edge/fog-рівні; не зберігають «важкі» дані в леджері, а лише їх криптографічні представлення; використовують permissioned-моделі (MSP/CA, політики доступу, приватні колекції даних) для керованості та відповідності вимогам бізнесу; доповнюють класичні механізми ідентичності сучасними SSI-патернами (DID/VC) для кращої міждоменної інтероперабельності. Разом з тим, залишаються практичні обмеження, які формують дослідницький порядок денний для наступних розділів роботи: масштабованість при великій кількості подій; затримки для критичних контурів керування; ризики витоку метаданих

навіть за умов приватних транзакцій; складність управління ключами на флоті пристроїв; а також необхідність формалізованих моделей загроз і метрик ефективності безпеки для оцінювання доцільності блокчейн-компоненти в конкретному IoT-сценарії.

1.4. Постановка задачі інтеграції блокчейн-технологій для забезпечення безпеки IoT-мереж

IoT-мережі, попри високу прикладну цінність у промисловості, енергетиці, транспорті та «розумній» інфраструктурі, залишаються одним із найбільш уразливих сегментів сучасних інформаційних систем. Це зумовлено поєднанням гетерогенності пристроїв, тривалого життєвого циклу без гарантованих оновлень, обмежених ресурсів на кінцевих вузлах і наявності кількох адміністративних доменів у межах однієї екосистеми (виробник, оператор, власник даних, сервіс-провайдер, підрядники). У таких умовах традиційні централізовані підходи до автентифікації, журналювання та керування доступом демонструють типові обмеження: виникнення єдиних точок відмови й компрометації, складність міждоменної довіри, низьку доказовість історії подій та труднощі з незалежним аудитом. Наявні практичні рішення для IoT-безпеки (захист каналів зв’язку, сегментація мережі, централізовані платформи керування пристроями, SIEM-моніторинг) істотно підвищують рівень захищеності, проте не усувають проблеми узгодженості й незмінності критичних записів у багатосторонніх сценаріях, де різні учасники мають різні інтереси та рівні довіри. Саме тому доцільною є розробка архітектурного підходу, який поєднає класичні засоби кіберзахисту з механізмами розподіленої довіри та незмінного аудиту.

Перспективним напрямом удосконалення безпеки IoT є інтеграція блокчейн-технологій як інфраструктурного шару фіксації та верифікації подій. На відміну від звичайних централізованих журналів, розподілений реєстр дозволяє організувати спільний “джерело істини” для критичних транзакцій

екосистеми: реєстрації та виведення з експлуатації пристроїв, управління ключами й повноваженнями, затвердження політик доступу, фіксації фактів оновлень та подій безпеки. Перевагами такого підходу є підвищення доказовості та простежуваності, можливість незалежного аудиту, зниження ризику підміни історії подій, а також підтримка міжорганізаційної взаємодії без потреби в єдиному довіреному центрі. Водночас недоліками виступають накладні витрати на підтримку консенсусу й обміну службовими даними, можливі затримки підтвердження транзакцій, складність адміністрування та підвищені вимоги до коректного проєктування моделі доступу й приватності. Отже, практичної цінності набуває саме розробка гібридної архітектури, де блокчейн застосовується для критичних записів і правил, тоді як об’ємні дані зберігаються поза реєстром із забезпеченням їхньої цілісності через криптографічні “відбитки”.

Для досягнення мети необхідно: проаналізувати загрози та типові вразливості IoT-екосистем; визначити вимоги до безпеки, продуктивності та приватності для цільового рішення; обрати доцільну модель блокчейн-мережі та принцип інтеграції з edge/шлюзами; спроєктувати механізми реєстрації пристроїв і керування їхнім життєвим циклом; розробити механізм контролю доступу та журналювання подій із можливістю перевірки цілісності; реалізувати прототип і провести експериментальне оцінювання накладних витрат та ефекту від застосування запропонованого підходу.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ, ВИБІР ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТАРІЮ ВИРІШЕННЯ ЗАДАЧІ

2.1 Обґрунтування вибору технологічної платформи та базової архітектури інструментарію

Інтеграція блокчейн-технологій у середовище IoT доцільна лише за умови коректно підібраного інструментарію, який відповідає одночасно двом групам вимог: з одного боку, вимогам до кібербезпеки та керування довірою в розподілених мережах, а з іншого – обмеженням IoT-пристроїв і реальним експлуатаційним сценаріям (обмежена пам’ять і енергія сенсорів, нестабільні канали зв’язку, різнорідність протоколів і виробників). У такому контексті блокчейн розглядається не як «універсальна заміна» існуючих механізмів захисту, а як компонент довірчої інфраструктури, що забезпечує незмінність журналу подій, керовану ідентифікацію учасників, а також відтворюваність перевірки правил доступу через смарт-контракти (або їх аналоги). Для IoT-мереж це особливо важливо, оскільки типові ризики пов’язані з компрометацією вузлів, підміною телеметрії, несанкціонованим керуванням виконавчими механізмами та складністю ретроспективного аудиту інцидентів [2,4,6].

Вихідна постановка задачі безпеки IoT-мереж у межах цієї кваліфікаційної роботи спирається на практично значущі цілі: забезпечення цілісності та простежуваності критичних подій (реєстрація приєднання пристрою, зміни конфігурації, оновлень, фактів керування), підвищення контрольованості доступу (хто і за яких умов може читати/змінювати параметри), а також мінімізація довіри до єдиного центру (уникнення ситуації, коли компрометація одного сервера автоматично компрометує і журнал подій, і політики доступу). З позицій стандартів безпеки IoT, ключовими є вимоги керованої ідентичності, безпечного обміну, захисту даних та можливості оновлень, а також підзвітності й моніторингу стану [4,6].

Архітектурно інтеграцію блокчейну доцільно реалізовувати як гібридну модель «edge/gateway + blockchain». Обчислювально обмежені кінцеві пристрої не повинні виконувати повний блокчейн-вузол чи зберігати ланцюг; натомість вони працюють через шлюз (edge-вузол), який завершує захищені сесії, виконує нормалізацію й фільтрацію телеметрії, формує події безпеки та ініціює транзакції у реєстрі. Такий підхід зменшує навантаження на сенсори, централізує контроль над криптографічними операціями (у межах домену організації) і водночас дозволяє зберігати «матрицю довіри» у розподіленому реєстрі. Важливим наслідком є необхідність розділяти дані на «on-chain» та «off-chain»: великі потоки телеметрії зберігаються поза блокчейном (у БД або сховищі), а в реєстр записуються хеші, метадані, ідентифікатори подій та маркери доступу. Це дає баланс між продуктивністю й доказовістю: блокчейн фіксує факт і незмінність ключових подій, а дані можуть масштабовано зберігатися поза ланцюгом.

Після визначення загальної архітектурної ідеї ключовим є вибір типу блокчейн-платформи. Для IoT-безпеки на практиці розглядають три класи: публічні блокчейни загального призначення (наприклад, Ethereum), спеціалізовані DLT/мережі під IoT (наприклад, IOTA, або інфраструктурні мережі на кшталт Helium для бездротового доступу), а також приватні/permissioned-платформи для корпоративних консорціумів (наприклад, Hyperledger Fabric). Критерії вибору у цій роботі формулюються як поєднання керованості доступу та ідентичностей, приватності даних і транзакцій, передбачуваності вартості виконання операцій, пропускної здатності та латентності, а також наявності зрілої екосистеми інструментів розгортання й спостережуваності.

Публічні блокчейни (на прикладі Ethereum) привабливі високою зрілістю смарт-контрактів і широкою підтримкою інструментів розробки. Проте саме для задач безпеки IoT-мереж (особливо корпоративних) істотними є обмеження: транзакції потребують оплати комісії («gas fees»), а їх вартість та час підтвердження залежать від завантаження мережі, що ускладнює прогнозування експлуатаційних витрат і SLA [26]. Крім того, публічна природа реєстру створює

додаткові виклики приватності, адже навіть при шифруванні корисного навантаження метадані (частота подій, взаємозв'язки адрес) можуть мати значення. Тому Ethereum доцільніше розглядати як платформу для відкритих екосистем або сценаріїв, де прозорість є ціллю, а не ризиком.

Натомість спеціалізовані під IoT DLT-концепції (на прикладі IOTA) пропонують іншу модель реєстру – DAG-структуру (Tangle), спрямовану на паралельність підтвердження транзакцій і потенційно кращу масштабованість у сценаріях з високою частотою невеликих повідомлень [35]. Водночас такі технології вимагають окремого аналізу зрілості інструментів, моделі безпеки та реальної інтегровності з корпоративними процесами керування ідентичністю й доступом. Інший «IoT-фокусований» напрям представлений Helium, який більше акцентує увагу на побудові децентралізованої бездротової інфраструктури (зокрема LoRaWAN-мережі покриття) і економічній мотивації учасників, а не на корпоративному контролі політик доступу в межах конкретної організації [36]. Відтак у цій роботі IOTA та Helium розглядаються як важливі приклади альтернативної філософії, але як базову платформу для задачі корпоративної безпеки IoT-мережі обрано іншу категорію – permissioned blockchain.

Для задачі забезпечення безпеки IoT-мережі в організаційному домені найбільш обґрунтованим є використання Hyperledger Fabric як permissioned-платформи. Принципова перевага Fabric полягає у наявності керованого членства та ідентичностей: участь у мережі визначається організаціями й довіреними центрами сертифікації, а права доступу реалізуються на рівні політик та логіки смарт-контрактів (chaincode) [18,20]. Така модель безпосередньо відповідає потребам IoT-безпеки, де необхідно чітко розмежувати ролі виробника пристроїв, оператора мережі, адміністратора безпеки й прикладних сервісів. Fabric також дозволяє будувати мережу як консорціум (кілька організацій-учасників), зберігаючи контроль над тим, хто має право читання/підтвердження транзакцій, що важливо для промислових і критичних систем. Архітектурні елементи Fabric (peer-вузли, леджер, chaincode) природно відповідають вимозі надлишковості та усунення єдиної точки відмови: кожна

організація може підтримувати власні peer-вузли з копією леджера, а процес підтвердження та впорядкування транзакцій робить журнал подій стійкішим до внутрішніх збоїв і атак [20].

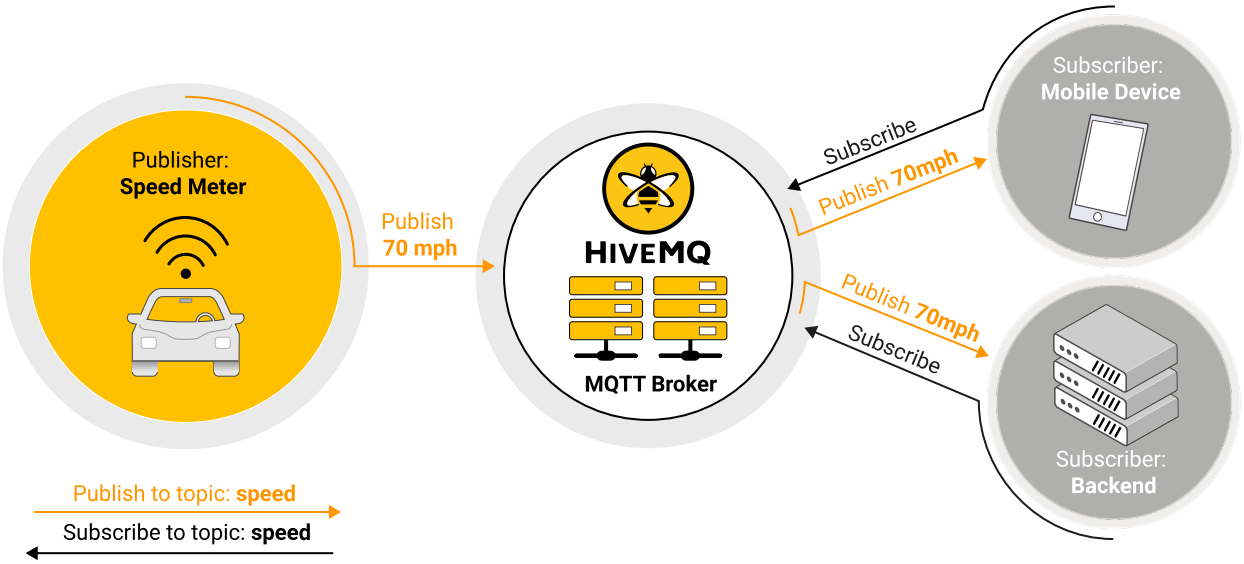


Рисунок 2.1 – Приклад архітектури Publish/Subscribe для MQTT [31].

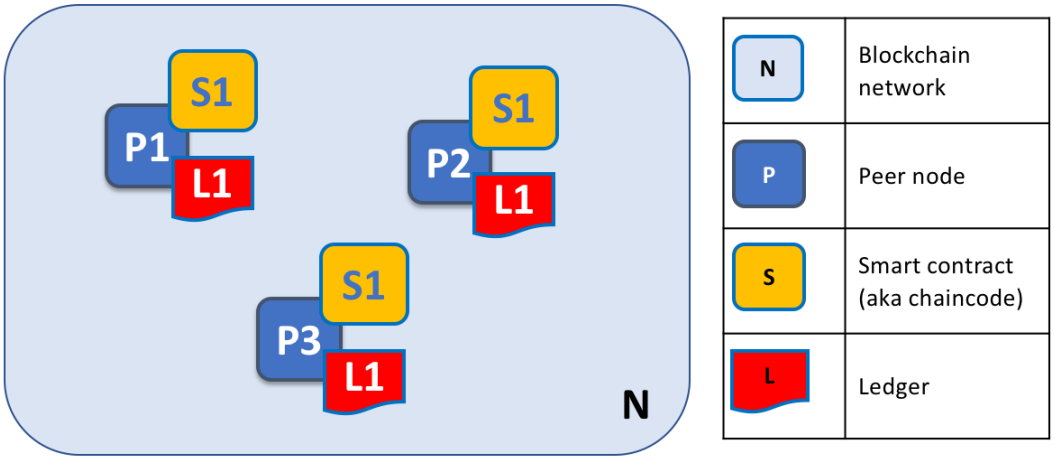


Рисунок 2.2 – Загальна схема мережі Hyperledger Fabric (peer-вузли, леджер і chaincode) [20].

Окремо слід підкреслити, що у Fabric ідентичність учасника мережі інтегрована в механізм членства (MSP): щоб виконувати транзакції, суб’єкт

повинен мати ідентичність, випущену довіреним CA та визнану організацією-учасником через MSP-конфігурацію [19]. Для практичної реалізації PKI-складової доцільно застосовувати Hyperledger Fabric CA, який підтримує реєстрацію ідентичностей, випуск сертифікатів, їх оновлення та відкликання, що є критично важливим для життєвого циклу IoT-пристроїв (втрата/компрометація ключів, перевипуск сертифікатів, виведення пристроїв з експлуатації) [21]. У контексті IoT це дозволяє реалізувати формальну «точку прив'язки» пристрою до організації та політики, не зводячи систему до статичних паролів або локальних списків доступу.

Після вибору блокчейн-платформи необхідно обґрунтувати транспорт і інтеграційний рівень IoT. У роботі як базовий протокол повідомлень доцільно використовувати MQTT, оскільки він орієнтований на середовища з нестабільними каналами та великою кількістю клієнтів, а його модель publish/subscribe спрощує розділення даних за темами та реалізацію політик доступу на рівні брокера і шлюзу [8,31]. Для реалізації брокера обрано Eclipse Mosquitto як легковаговий і поширений відкритий брокер, який реалізує версії MQTT 5.0, 3.1.1 і 3.1 та придатний для розгортання як на одноплатних комп'ютерах, так і на серверних системах [32]. Це створює практичну основу для прототипування стенду, де сенсори/емулятори публікують телеметрію, шлюз перетворює її у події безпеки (наприклад, «device_connected», «config_changed», «command_issued»), а далі ці події фіксуються у Fabric як транзакції.

На рівні прикладної інтеграції між MQTT-даними та блокчейн-операціями доцільно передбачити інструмент, що прискорює побудову прототипів і забезпечує наочність потоків обробки. У цьому сенсі Node-RED є виправданим вибором як середовище flow-based розробки, яке дозволяє швидко реалізувати приймання MQTT-повідомлень, виконання перетворень, маршрутизацію подій, а також виклик REST/SDK-операцій для взаємодії з блокчейн-мережею (через gateway/SDK рівень) [33]. Важливо, що Node-RED у межах цієї роботи розглядається не як «кінцева промислова платформа», а як ефективний інструмент розроблення та валідації логіки інтеграції, який у

подальшому може бути замінений на мікросервісну реалізацію з аналогічною функціональністю.

Далі, щоб забезпечити відтворюваність експериментів, керованість конфігурацій та простоту розгортання прототипу, інструментарій доцільно контейнеризувати. Docker Compose є практичним вибором для локальної або стендової інсталяції, оскільки дозволяє визначати набір сервісів (MQTT-брокер, інтеграційний сервіс/Node-RED, компоненти Fabric, моніторинг) у єдиному YAML-описі та запускати їх узгодженою групою, керуючи мережами, томами та змінними середовища [27]. Використання Compose-специфікації також унормовує структуру конфігурації застосунку як набору служб і залежностей, що важливо для технічної документації магістерської роботи та повторюваності результатів [28].

Рисунок 2.3 – Приклад дашборду Grafana для моніторингу стану системи [30].

Оскільки безпека IoT-мережі є не лише «політиками доступу», а й постійним контролем стану системи, у склад інструментарію необхідно

включити засоби спостережуваності. Для метрик доцільно використати Prometheus як відкритий інструмент моніторингу з моделлю часових рядів і міток (labels), що дозволяє системно збирати дані про навантаження брокера, затримки обробки, кількість подій безпеки, помилки автентифікації, параметри контейнерів та інші показники [29]. Для візуалізації й аналітики метрик обґрунтовано застосувати Grafana (рис.2.3), яка надає гнучкі панелі та дашборди для комплексного відображення стану інфраструктури й сервісів, що особливо корисно для демонстрації результатів експериментів у межах кваліфікаційної роботи [30].

Окрім загальносистемної спостережуваності, у блокчейн-інтеграції необхідна «спостережуваність реєстру», тобто можливість аналізувати блоки, транзакції, стани chaincode і події мережі. Для цього доцільно включити Hyperledger Explorer як інструмент візуалізації операцій Fabric, що полегшує демонстрацію роботи прототипу та пришвидшує діагностику (перевірка факту запису подій, аналіз структури транзакцій, перегляд активності організацій) [34]. У контексті роботи це підсилює доказовість результатів: журнал безпеки стає не лише «записаним», але й доступним для аналізу через інтерфейс, що відтворює історію.

Таким чином, базовий інструментарій для подальшої реалізації стенду інтеграції блокчейну та IoT-мережі формується як узгоджена система: permissioned-блокчейн Hyperledger Fabric для керованого членства та незмінного журналу, MQTT як транспорт телеметрії й команд, Mosquitto як брокер повідомлень, Node-RED як інтеграційний рівень прототипування потоків, Docker Compose як засіб відтворюваного розгортання, Prometheus+Grafana як спостережуваність інфраструктури та сервісів, а Hyperledger Explorer як візуалізація активності реєстру. Така конфігурація є практично орієнтованою та узгоджується з вимогами до безпеки IoT (ідентичність, контроль доступу, аудит, моніторинг) [4,6], створюючи методично коректну основу для подальшого дослідження в рамках цієї магістерської роботи на рівні реалізації компонентів і сценаріїв взаємодії.

2.2. Вибір програмних компонентів та проєктування інтерфейсів взаємодії в системі «Blockchain + IoT Security»

Ефективність інтеграції блокчейн-технологій у безпеку IoT-мережі визначається не лише вибором платформи розподіленого реєстру, а й коректною побудовою програмного «шару взаємодії» між обмеженими за ресурсами IoT-вузлами та консорціумною мережею блокчейну. Тому в межах підрозділу обґрунтовується набір програмних компонентів, що забезпечують: (1) кероване підключення та автентифікацію пристроїв; (2) безпечний обмін телеметрією і командами; (3) виконання смарт-контрактів для контролю доступу; (4) аудит і прозорий моніторинг подій; (5) інтеграцію з зовнішніми системами через стандартизований API. Логіка такого поділу відповідає вимозі модульності та простоти взаємодії, коли користувач працює через веб/мобільний інтерфейс, а серверний рівень виконує роль проміжної ланки між контролером IoT та блокчейн-мережею.

Узагальнено, програмна архітектура доцільно реалізується за моделлю «IoT пристрій → шлюз/контролер → сервіс інтеграції → блокчейн-мережа → інтерфейси керування/аудиту». У такому підході «контролер» (gateway) забезпечує комунікацію з IoT-пристроями у локальному сегменті, а «сервер» (integration service) виконує обробку даних, зберігання службової інформації та формування транзакцій для блокчейну. Сам блокчейн використовується як незмінний журнал подій і політик доступу, тоді як великі обсяги телеметрії (наприклад, потокові сенсорні дані) доцільно зберігати поза ланцюгом (off-chain), фіксуючи в реєстрі лише криптографічні хеші, ідентифікатори пакетів та атрибути доступу. Це зменшує навантаження на блокчейн та водночас зберігає доказовість цілісності даних за рахунок порівняння хешів при аудиті.

Ключовим рішенням для рівня блокчейн-взаємодії є використання Hyperledger Fabric як permissioned-платформи консорціумного типу, де ідентичності учасників визначаються інфраструктурою сертифікації (MSP/CA), а транзакції підпорядковані політикам підписів та правилам доступу. Практична

цінність Fabric для IoT-безпеки полягає в тому, що вона підтримує (а) формалізоване керування ідентичностями на основі X.509; (б) смарт-контракти (chaincode) для реалізації політик доступу; (в) подієву модель (events) для інтеграції з моніторингом. На клієнтському боці для спрощення прикладної логіки доцільно орієнтуватися на Fabric Gateway: сервіс, що делегує частину «інфраструктурних» процедур (збирання endorsements, submit, очікування commit-подій) до peer-вузла, залишаючи прикладному застосунку мінімальний API для запитів і транзакцій [32]. Це зменшує складність server-компонента, який обслуговує велику кількість IoT-запитів, і підвищує відтворюваність транзакційного потоку при тестуванні.

Для формування та життєвого циклу ідентичностей у Fabric критичним є компонент CA (Certificate Authority). Вибір Fabric CA виправданий наявністю типової процедури «register → enroll», яка забезпечує розмежування відповідальності: адміністратор реєструє користувача/пристрій (видає enroll ID/secret і атрибути), а кінцевий вузол самостійно виконує enroll та отримує власну пару ключів і сертифікат, не розкриваючи приватний ключ адміністратору [38]. Такий механізм безпосередньо підтримує вимогу безпечної реєстрації IoT-пристроїв шляхом створення унікального криптографічного ключа і запису факту реєстрації в реєстрі.

У результаті в системі формується керований процес onboarding: пристрій реєструється у CA, отримує сертифікат (та, за потреби, TLS-сертифікат), після чого допускається до взаємодії зі шлюзом/сервером і до виконання транзакцій бізнес-логіки.

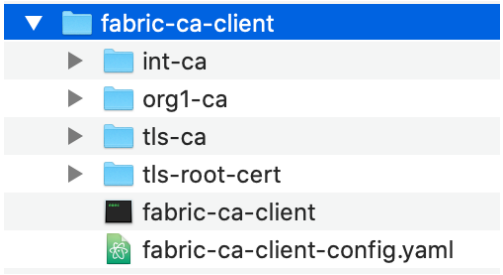


Рис. 2.4. Приклад структури каталогів Fabric CA client для керування кількома CA .

На рівні обміну даними між IoT-вузлами та контролером/шлюзом доцільно застосовувати легковаговий протокол MQTT (або CoAP для сенсорних мереж), оскільки він оптимізований під передачу невеликих повідомлень з підтримкою QoS і підходить для пристроїв з обмеженим енергоспоживанням. Безпекова інтеграція реалізується через TLS-канал і обов'язкову автентифікацію клієнта, а роль «якоря довіри» можуть виконувати сертифікати, видані CA/PKI, синхронізовані з політиками доступу у блокчейні. У такій моделі MQTT-рівень забезпечує доставку телеметрії та команд у реальному часі, а блокчейн-рівень - доказовість, аудитованість і виконання правил доступу через смарт-контракти. Практично це означає: повідомлення з датчиків надходять до контролера; контролер передає їх серверу; сервер формує транзакцію до Fabric для фіксації факту події або контрольної суми пакета; при читанні даних або виконанні команд сервер перевіряє політику доступу, що закодована у chaincode, і лише після позитивного рішення дозволяє дію. Власне вимоги до модулів реєстрації, налаштування смарт-контрактів, моніторингу та доступу логічно випливають з такої зв'язки.

Окремої уваги потребує проєктування прикладного рівня (інтерфейсів і сервісів). У базовій роботі, взятій за структурну основу, обґрунтовано, що програмний інтерфейс має включати механізм реєстрації пристроїв з генерацією ключів, модуль керування смарт-контрактами, засоби моніторингу транзакцій, керування доступом та API для інтеграції

У межах даної магістерської роботи ці компоненти доцільно реалізувати як набір сервісів (або модулів моноліту) із чітко визначеними відповідальностями. Модуль реєстрації (Device Onboarding Service) взаємодіє з Fabric CA і зберігає метадані пристрою; модуль політик доступу (Access Policy Service) виконує запити до chaincode і формує рішення «дозволити/заборонити»; модуль журналювання (Audit & Logging Service) фіксує події та збагачує їх контекстом; модуль інтеграції (Integration API) надає REST-інтерфейс для зовнішніх сервісів та клієнтських застосунків. Важливо, що в такій архітектурі користувач не взаємодіє безпосередньо з блокчейном: він працює через

веб/мобільний інтерфейс або зовнішню систему, тоді як сервер виконує вимірювані й контрольовані операції з ідентичностями та транзакціями.

Проєктуючи API-шар, доцільно застосувати OpenAPI як формальний стандарт опису HTTP-API, що забезпечує прозорість контракту, автоматизацію тестування та генерацію клієнтів [30]. Особливо цінним OpenAPI є тоді, коли система має інтегруватися з наявними IoT-платформами, аналітичними сервісами або корпоративними системами управління інцидентами. Ця потреба узгоджується з вимогою забезпечити інтеграцію блокчейн-мережі з іншими середовищами через RESTful API та передавання даних у форматах JSON/XML.

В контексті безпеки OpenAPI-контракт має супроводжуватися механізмами аутентифікації (наприклад, токени доступу для користувачів адміністративної панелі), обмеженням частоти запитів, журналюванням та обов'язковим використанням TLS.

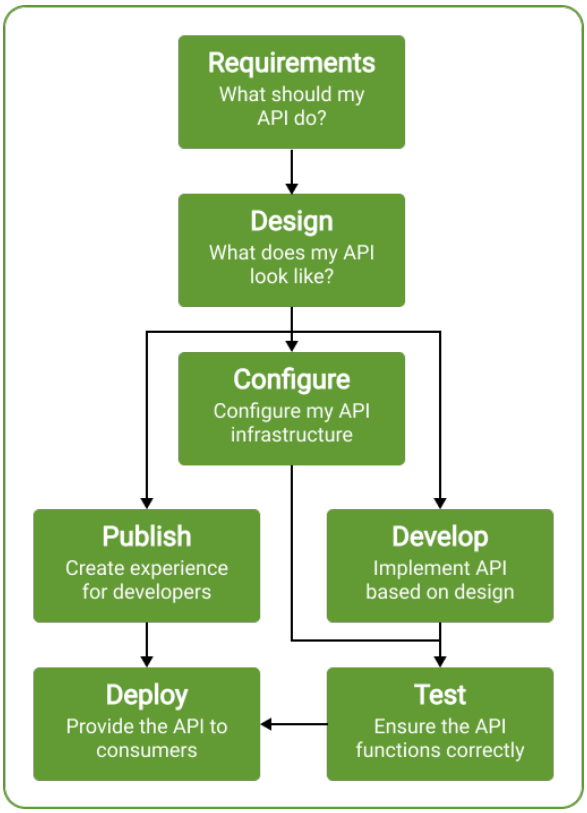


Рис. 2.5. Приклад «життєвого циклу API» [40]

Далі, модуль керування доступом у системі «Blockchain + IoT» доцільно реалізовувати як поєднання: правил у смарт-контрактах (верифікація

ролей/атрибутів сертифіката, стан пристрою, контекст запиту) та прикладної логіки на сервері (валидація параметрів, перевірка сесії користувача, gateway-політики). Також варто підкреслити необхідність налаштування прав доступу для різних пристроїв і користувачів, інтеграції аутентифікації/авторизації через блокчейн та підтримки динамічних політик залежно від статусу пристрою чи умов смарт-контракту.

Політика доступу розглядається як функція від (а) ідентичності суб'єкта (користувач/сервіс/пристрій), (б) типу ресурсу (телеметрія, команда керування, конфігурація), (в) контексту (час, зона, стан пристрою, рівень ризику), (г) історії дій (audit trail). Блокчейн забезпечує незмінність і відтворюваність правил та фактів доступу, що є критично важливим для розслідування інцидентів у IoT-мережах.

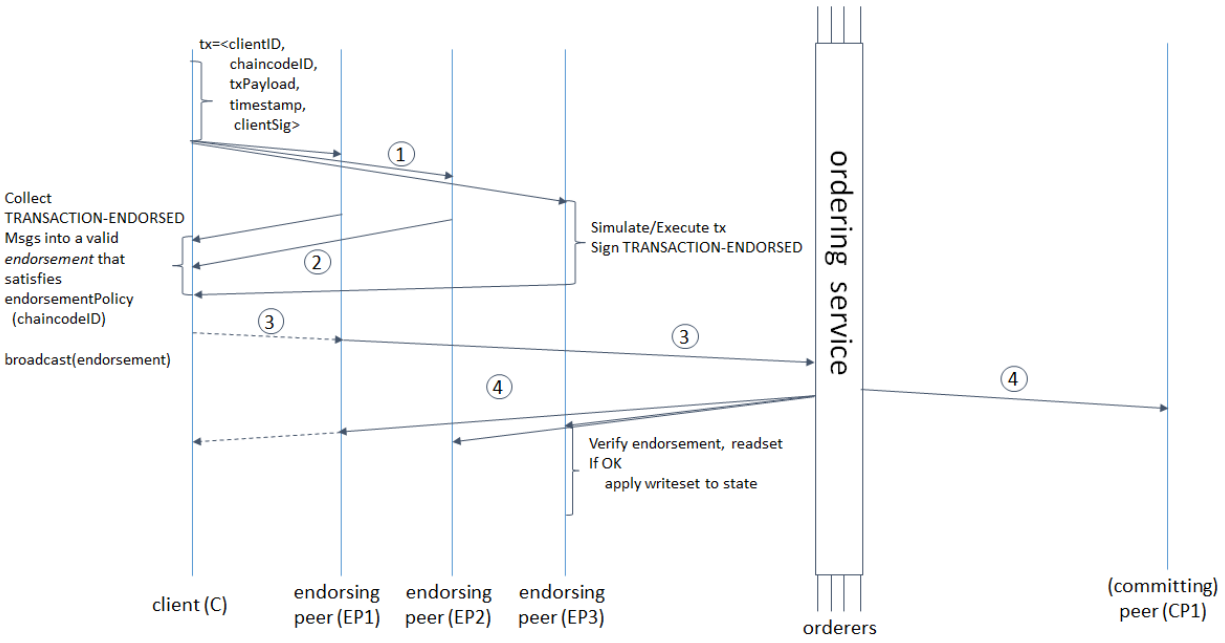


Рис. 2.6. Схема потоку транзакції у Hyperledger Fabric [41]

Моніторинг і візуалізація мають закривати дві взаємопов'язані задачі: оперативний контроль працездатності та безпековий аудит. Базова вимога полягає в наявності візуалізації взаємодій між пристроями, відображенні транзакцій у реальному часі, журналі транзакцій та інструментах аналізу аномалій.

Для цього в екосистемі Hyperledger Fabric доцільно комбінувати: (1) технологічні метрики (Prometheus/Grafana) для контейнерів, peer/orderer, сервісів API та MQTT-broker; (2) блокчейн-оглядач (Explorer) для перегляду блоків/транзакцій/подій; (3) журнал прикладних подій (централізоване логування) для кореляції мережевих інцидентів із транзакційними фактами. При такій інтеграції дані моніторингу набувають доказового контексту: наприклад, аномальне збільшення кількості команд на контролер можна зіставити з тим, які ідентичності ініціювали відповідні транзакції та чи були вони підтверджені політикою доступу.

З урахуванням вищенаведеного, вибір інструментарію для реалізації прикладних компонентів можна сформулювати як узгоджений набір. Сервер інтеграції доцільно реалізовувати на стеку, що має стабільні SDK/клієнтські бібліотеки Fabric Gateway та зручні засоби роботи з мережевими протоколами IoT. Наприклад, Node.js або Java дозволяють поєднати високорівневу реалізацію REST-API, інтеграцію з MQTT-клієнтами та доступ до Fabric Gateway API [37,42]. Вибір конкретної мови/платформи в подальших підрозділах може бути прив’язаний до вимог продуктивності, наявності бібліотек, досвіду команди та простоти контейнеризації. На рівні смарт-контрактів (chaincode) доцільно дотримуватися принципу «мінімальної достатності»: контракт містить тільки ті правила, які мають бути незмінними, аудитованими і відтворюваними (реєстрація/статус пристрою, політики доступу, фіксація подій безпеки), тоді як складні обчислення або агрегації телеметрії виконуються поза блокчейном.

2.3. Реалізація прототипу рішення та налаштування взаємодії компонентів (IoT-шлюз, блокчейн-реєстр, моніторинг)

Практична реалізація інструментарію інтеграції блокчейн-технологій у безпеку IoT-мережі в межах магістерської роботи доцільно виконувати у форматі прототипу, який відтворює повний цикл обробки подій: від підключення IoT-пристрою і передавання телеметрії до перевірки політик доступу та фіксації

аудиторних подій у розподіленому реєстрі. Такий прототип має бути достатньо складним, щоб демонструвати ключові властивості блокчейн-підходу (незмінність журналу, керована ідентичність, відтворюваність перевірки правил), але водночас технологічно керованим для експериментів і вимірювань (затримки, пропускна здатність, частота подій, частка відхилених запитів). У межах обраної архітектури «IoT-пристрій → контролер/шлюз → сервер інтеграції → блокчейн-мережа → інтерфейси аудиту» основну увагу приділено налаштуванню блокчейн-мережі (Hyperledger Fabric), механізму ідентичностей (CA/MSP), реалізації смарт-контрактів (chaincode), інтеграції з IoT-підсистемою через MQTT і створенню спостережуваності (Prometheus/Grafana та Explorer) [20,29,34].

Розгортання прототипу доцільно виконувати в контейнеризованому середовищі: це забезпечує повторюваність експериментів та спрощує документування конфігурацій. Базовим механізмом оркестрації стенду може бути Docker Compose, який дозволяє описати склад сервісів, мережі, томи даних і параметри середовища в одному конфігураційному файлі [27,28]. У межах прототипу сформовано типову групу контейнерів: блокчейн-компоненти (peer-вузли, orderer, CA), периферійні сервіси (MQTT-брокер Mosquitto, сервіс інтеграції/шлюз доступу до Fabric, за потреби Node-RED як інструмент побудови потоків), компоненти аудиту й моніторингу (Hyperledger Explorer, Prometheus, Grafana). Такий підхід дозволяє проводити контрольовані експерименти: наприклад, змінювати інтенсивність телеметрії, політики підтвердження транзакцій, обсяг подій аудиту, а потім зіставляти результати з метриками та станом леджера.

Критичним етапом є конфігурація Fabric-мережі та процедура розгортання смарт-контрактів. У Fabric версій 2.x застосовується концепція «життєвого циклу chaincode», яка передбачає узгодження параметрів контракту між організаціями: пакування, встановлення на peer-вузли, схвалення визначення (approve) кожною організацією та фінальне commit визначення в канал [43]. З методичної точки зору цей механізм є важливим для безпекового

сценарію IoT, оскільки дозволяє формально закріпити політику підтвердження і відповідальність сторін: наприклад, визначити, що транзакції, пов’язані з реєстрацією пристроїв або зміною критичних політик доступу, мають підтверджуватися більш ніж однією організацією. Типовий процес розгортання оформлюється відповідно до офіційних рекомендацій щодо «deploy chaincode» в тестовій або пілотній мережі Fabric [24]. У прототипі доцільно створити окремий канал для подій безпеки IoT та розмістити в ньому chaincode, який відповідає за життєвий цикл ідентичностей пристроїв, реєстрацію подій аудиту та перевірку прав доступу до операцій (читання телеметрії, виконання команд керування, зміна конфігурацій).

Смарт-контракт (chaincode) в контексті IoT-безпеки доцільно проєктувати як компактний набір транзакцій, орієнтований на незмінні в бізнесовому сенсі факти: хто є пристроєм, який його статус, які дії було виконано та на підставі яких правил. Практично це означає, що у «світовому стані» (world state) зберігається структурований опис пристрою (ідентифікатор, належність організації/домену, хеш сертифіката або його відбиток, статус active/revoked/blocked, часові метки останньої активності) та опис політик доступу на рівні ролей/атрибутів. Для ефективних запитів і вибірок у прототипі доцільно використати CouchDB як state database: він підтримує зберігання JSON-структур і «rich queries», що спрощує аналітику подій і пошук за атрибутами пристрою чи інциденту [45]. При цьому великі масиви телеметрії не мають переноситися в блокчейн: натомість прототип реалізує підхід «off-chain дані + on-chain доказовість», коли в леджер записується хеш телеметричного пакета, ідентифікатор джерела, час отримання і код перевірки доступу, а фактичний JSON/CSV-пакет зберігається у зовнішньому сховищі. Така модель знижує транзакційне навантаження і водночас забезпечує можливість довести незмінність даних при аудиті через порівняння хешів.

Окрема група даних (наприклад, конфіденційні параметри пристрою, службові токени або окремі атрибути власності) може вимагати обмеження видимості навіть всередині каналу. Для цього в Fabric передбачені приватні

колекції даних (Private Data Collections), які дозволяють зберігати певні поля лише для визначеного підмножинного складу організацій, залишаючи в блокчейні лише хеш приватного вмісту [22]. У прототипі доцільно використати цей механізм для випадків, коли треба показати узгодження «спільного аудиту» і «приватності деталей»: наприклад, усі бачать факт зміни конфігурації та її час, але конкретне значення параметра доступне лише організації-оператору.

Надійність схеми керування доступом безпосередньо залежить від механізму ідентичностей. Тому у прототипі процедуру реєстрації (onboarding) пристроїв доцільно реалізувати на основі Fabric CA та MSP-моделі. CA забезпечує реєстрацію та випуск сертифікатів, їх оновлення й відкликання, а MSP визначає, які довірені корені та правила автентифікації діють у мережі [19,21]. Загальна роль Fabric CA в архітектурі показана на рис. 2.7: клієнт взаємодіє з CA через REST API, отримує сертифікати, після чого ці ідентичності використовуються в мережі Fabric та SDK/шлюзах доступу.

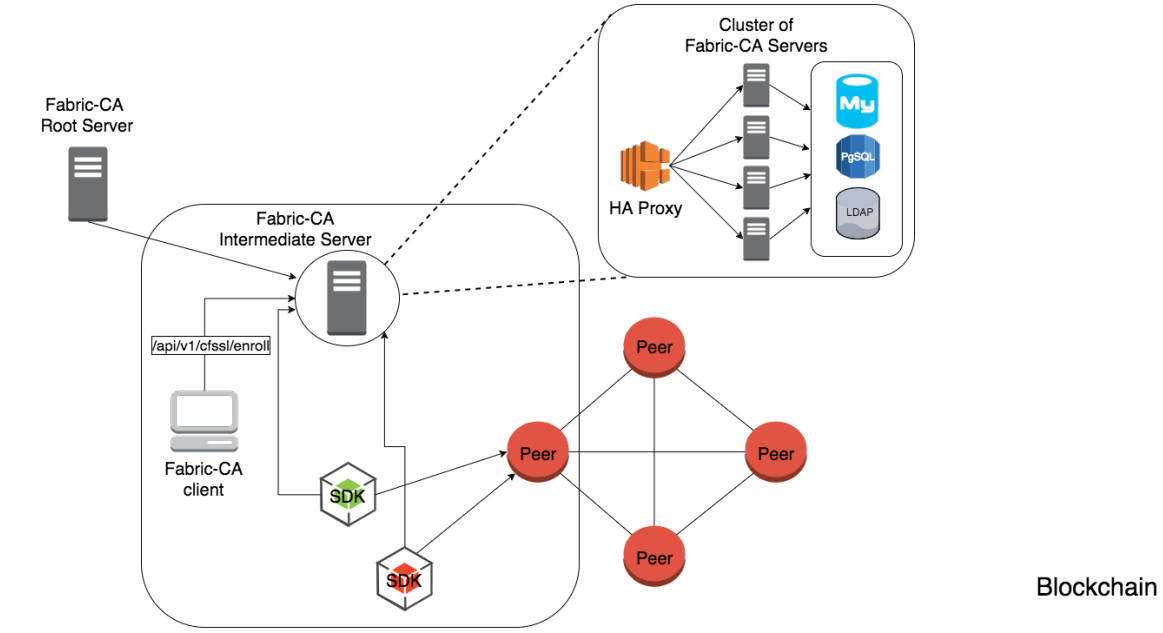


Рисунок 2.7 – Місце Hyperledger Fabric CA у загальній архітектурі

У практичному вимірі це дозволяє організувати життєвий цикл ключів для IoT: пристрій або шлюз отримує сертифікат, далі використовується взаємна TLS-автентифікація між IoT-частиною та брокером/шлюзом, а при компрометації сертифікат відкликається і відповідний статус фіксується в

реєстрі (як у chaincode, так і в CA). Важливо, що для IoT-середовища зазвичай небажано покладатися на паролі чи «жорстко вшиті» секрети; тому прототип демонструє домінування сертифікатної моделі, яка краще масштабується та підтримує відкликання [21,38].

На рівні IoT-обміну прототип використовує MQTT як транспорт телеметрії й команд, що є типовим для багатьох IoT-сценаріїв, а брокером виступає Mosquitto [32]. Канал зв'язку доцільно захищати TLS, оскільки це зменшує ризики перехоплення та підміни повідомлень, а також створює основу для взаємної автентифікації клієнтів через сертифікати (mTLS). Практичні вимоги до формування сертифікатів, відмінності між CA/серверними/клієнтськими сертифікатами та налаштування SSL/TLS для Mosquitto описані в офіційному man-описі mosquitto-tls(7) [29]. У прототипі захищений MQTT-транспорт узгоджується з блокчейн-рівнем таким чином: пристрій, який не може пройти TLS-ідентифікацію (або позначений як revoked у реєстрі), не допускається до публікації критичних топіків, а спроба доступу фіксується як подія аудиту в леджері.

«Серцем» інтеграції між MQTT-подіями та блокчейн-транзакціями виступає сервіс інтеграції (application gateway). Його функція полягає у тому, щоб приймати події від контролера/брокера, виконувати нормалізацію, валідувати формат, перевіряти політики доступу через виклик chaincode та, за потреби, формувати транзакції в Fabric. З точки зору практичної реалізації для цього зручно використовувати Gateway API Fabric, який мінімізує складність клієнтського застосунку, делегуючи частину процедури відправлення транзакції мережевому шлюзу peer-вузла [27]. Для безпекової логіки важливо, що сервіс інтеграції може одночасно виконувати роль «policy enforcement point»: він не просто записує в блокчейн, а блокує або дозволяє операцію керування на основі правил, що є відтворюваними та аудитованими. Саме така організація дозволяє довести в роботі реальну практичну цінність блокчейну: політика доступу не є «налаштуванням у конфіг-файлі», а є частиною контрольованої логіки, зміни якої фіксуються транзакційно.

Для синхронізації зовнішніх процесів із подіями леджера прототип може використовувати механізм подій Fabric: успішні транзакції можуть генерувати chaincode events, а додатки мають можливість підписуватися на події каналів/блоків, отримуючи підтвердження commit-статусу та дані для автоматизації реакцій [47]. Такий механізм практично корисний для IoT-середовища: наприклад, після commit транзакції «заблокувати пристрій» система може автоматично оновити політики брокера або списки доступу шлюзу, а після commit транзакції «команда видана» — відобразити статус у панелі оператора та зафіксувати доставку. Модель подій підсилює узгодженість системи: рішення «дозволено/заборонено» стає не лише миттєвою відповіддю сервісу, а відображається у незмінному журналі й може бути використане для подальшого аналізу інцидентів.

Для забезпечення спостережуваності прототипу важливо розмежувати: (а) моніторинг інфраструктурних метрик і сервісів та (б) аудит блокчейн-активності як бізнесово значущих подій. Першу частину доцільно реалізувати на базі Prometheus та Grafana [29,30]. Prometheus працює за моделлю «scrape»: він опитує цілі за HTTP-ендпойнтом метрик і зберігає часові ряди, а Grafana формує дашборди для аналізу. Загальна архітектура Prometheus представлена на рис. 2.8.

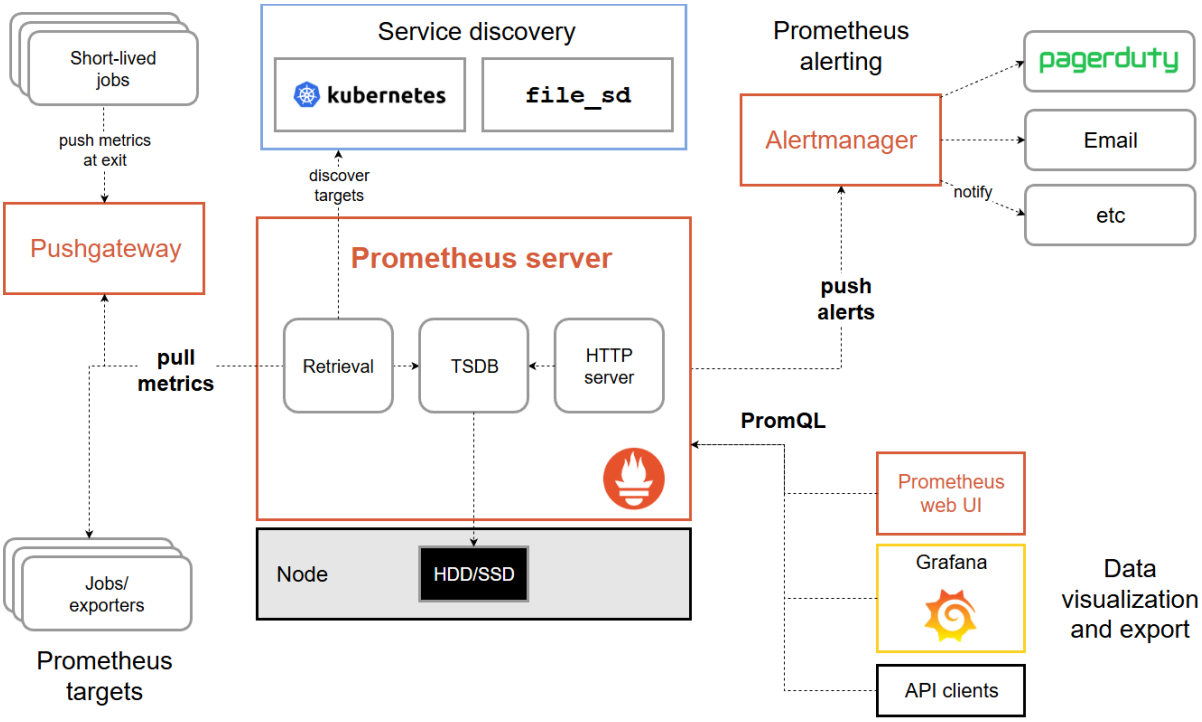


Рисунок 2.8 – Архітектура Prometheus і типові компоненти екосистеми

У контексті Fabric мережа надає набір метрик для peer/orderer, які можуть експортуватися та споживатися Prometheus; зокрема у документації описано, що метрики orderer експортуються для Prometheus і містять характеристики обробки транзакцій, часу формування блоків, статусів валідації тощо [48]. Поєднання цих метрик із метриками MQTT-компонента (кількість підключень, частота публікацій, відмови автентифікації), а також із метриками сервісу інтеграції (вхідні запити, транзакції, відхилені операції, середня затримка) дозволяє будувати експериментальну частину роботи на фактичних числових показниках, а не на декларативних твердженнях.

Друга частина полягає у аудиті блокчейн-активності, що у нашому прототипі реалізується через Hyperledger Explorer, який надає веб-інтерфейс для перегляду блоків, транзакцій, каналів, chaincode та peer-вузлів. Explorer також використовує базу даних (як правило, PostgreSQL) для збереження інформації про транзакції й прискорення відображення у реальному часі; загальний вигляд моделі даних Explorer подано на рис. 2.9.

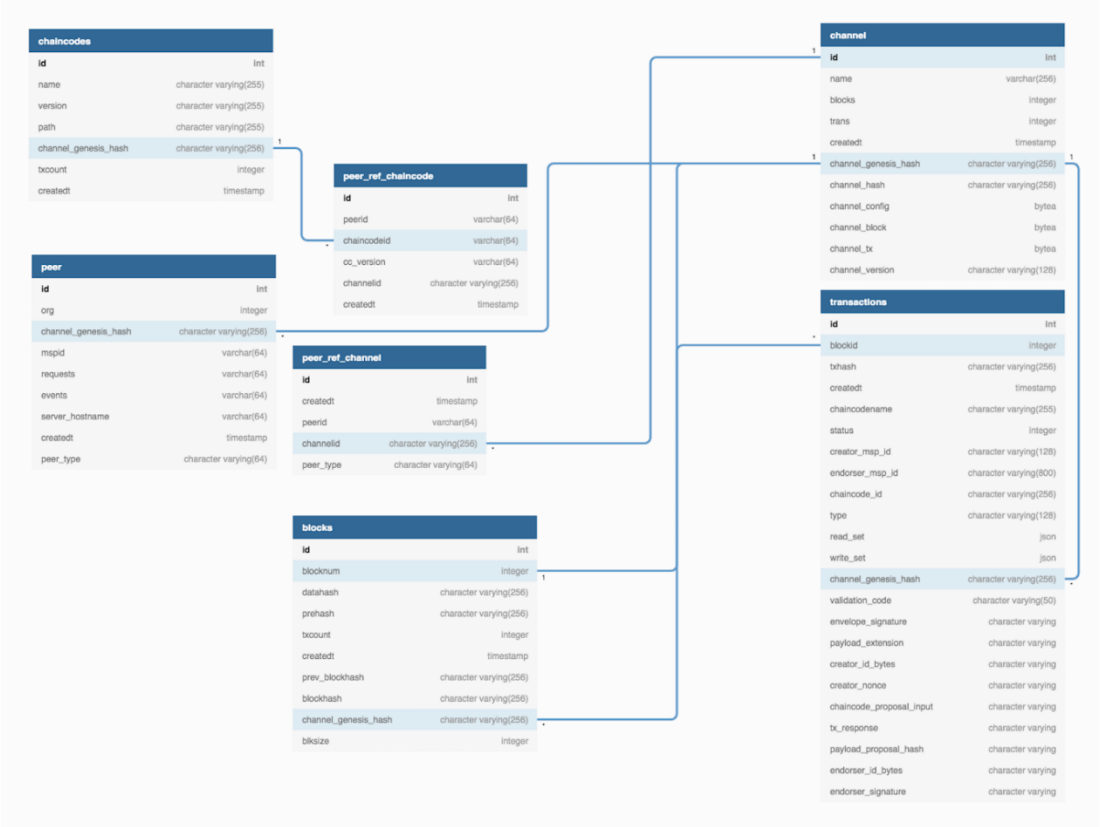


Рисунок 2.9 – Узагальнена модель даних Hyperledger Explorer для відображення блоків і транзакцій.

Важливим практичним результатом такої конфігурації є можливість продемонструвати «доказовість подій» на екрані: наприклад, що спроби доступу або видані команди мають конкретний txHash, час включення в блок, статус валідації, а також прив’язку до ідентичності ініціатора (через MSP). Для IoT-безпеки це означає, що розслідування інциденту спирається не тільки на журнали застосунку чи брокера, які теоретично можуть бути змінені, а на транзакційний ланцюг із незмінними доказами.

Отже, реалізація прототипу показує практичну взаємодію інструментарію: Mosquitto забезпечує захищений MQTT-обмін, сервіс інтеграції виконує перевірку політик і створює транзакції, Fabric надає керований реєстр подій і політик, CouchDB (як state database) спрощує запити до світового стану, а Prometheus/Grafana та Explorer створюють повний контур спостережуваності та аудиту. Життєвий цикл chaincode і можливість приватних колекцій даних забезпечують керований процес змін і приватність чутливих полів, а механізм подій Fabric дозволяє будувати реактивні сценарії та автоматизацію у відповідь на commit транзакцій. Сукупно це формує технічно обґрунтовану основу для подальших досліджень у межах цієї роботи, де можна виконати експериментальні вимірювання та оцінити, як інтеграція блокчейну впливає на цілісність, керованість доступу та можливості аудиту в IoT-мережі.

2.4. Специфікація архітектури та сценаріїв взаємодії компонентів у системі безпеки IoT з розподіленим реєстром

Побудова програмної архітектури для IoT-мережі з підсиленням безпеки блокчейн-технологіями має враховувати дві групи вимог, які на практиці часто конфліктують. З одного боку, IoT-середовище потребує мінімальних затримок, стійкої доставки повідомлень, підтримки великої кількості ресурсно-обмежених пристроїв та простого підключення нових вузлів. З іншого боку, безпека вимагає криптографічно перевірюваної ідентифікації, керування доступом, незмінного журналу подій і можливості відтворюваного аудиту інцидентів. Тому в межах

спроєктованого рішення блокчейн не підміняє собою всю інфраструктуру IoT, а використовується як довірчий шар для фіксації критичних фактів (ідентичностей, політик доступу, подій безпеки та результатів виконання команд), тоді як оперативні дані та високочастотна телеметрія обробляються традиційними засобами в межах шлюзів, сервісів інтеграції та бази даних.

Концептуально архітектура базується на принципі «пристрій не стає блокчейн-вузлом». Участь кінцевих датчиків у консенсусі або зберіганні ланцюга є надмірною через енергоспоживання, обмеження пам'яті та нестабільність зв'язку. Натомість роль мосту між IoT-сегментом і блокчейн-мережею виконує серверний контур разом із контролером (IoT-шлюзом), який агрегує сигнали та реалізує базові механізми локальної надійності. У такій побудові ключові контрольні точки безпеки концентруються в місцях, де їх можна гарантовано підтримати: на рівні контролера (доступ до локального сегмента та виконання команд) і на рівні сервера інтеграції (перевірка політик, формування транзакцій, аудит, взаємодія з блокчейном та БД).

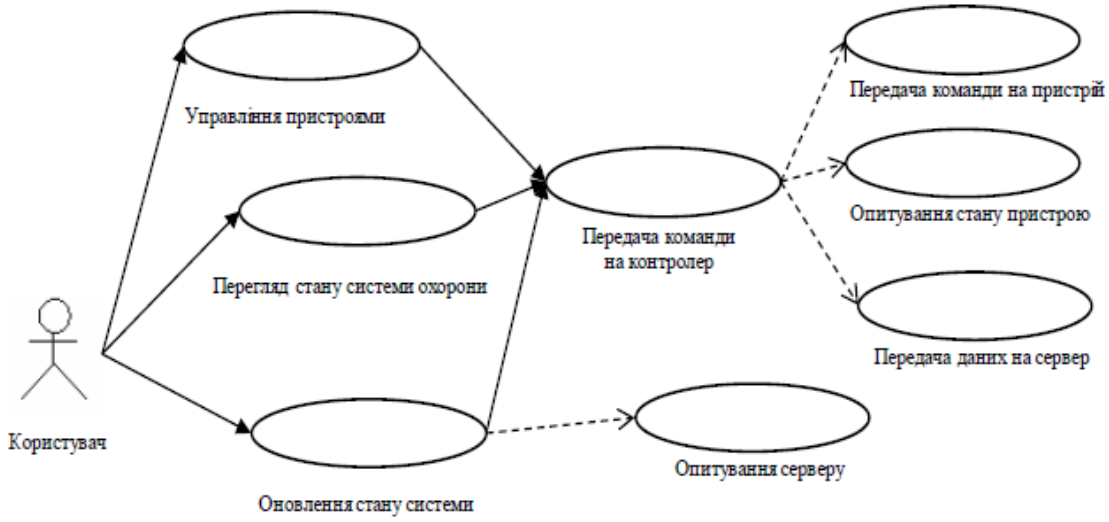


Рисунок 2.10 – Діаграма варіантів використання програмного комплексу безпеки IoT із довірчим реєстром.

Функціональна «рамка» системи задається сценаріями взаємодії користувача із сервісом. На рис. 2.10 показано діаграму випадків використання, яка відображає операції моніторингу й керування: користувач переглядає стан системи охорони, ініціює оновлення станів, здійснює опитування сервера, а

також передає команди на контролер і на конкретні пристрої. Саме ці сценарії визначають, які операції повинні мати підвищену довіру, а отже вимагати підтвердження прав доступу та незмінної фіксації в журналі. Для архітектури це означає, що всі керувальні дії (зміна станів, виконання команд, зняття тривоги) мають проходити через контур авторизації та аудиту, а всі операції читання (перегляд стану) мають мати механізм перевірки актуальності та цілісності відповідей.

На рівні інтерфейсів взаємодії система поділяється на клієнтську частину (мобільний/веб-інтерфейс), сервер прикладної логіки, контролер IoT та кінцеві пристрої. Для сценаріїв читання вирішальним є шлях «клієнт → сервер → база даних → клієнт». Він відображений на рис. 2.11, де сервер виступає єдиною точкою доступу до даних і може накладати безпекові правила перед поверненням інформації користувачу. У блокчейн-орієнтованій архітектурі цей шлях доповнюється перевіркою довіри: сервер не лише читає стан із БД, а й, коли йдеться про критичні стани або події безпеки, виконує звірку з незмінним реєстром.

Практично це реалізується таким чином: у БД зберігаються деталізовані записи станів/подій (тип датчика, значення, контекст приміщення, частота, історія), а в блокчейн-реєстрі – контрольні атрибути (ідентифікатор події, часові мітки, хеш корисного навантаження, статус доступу, ідентичність ініціатора). Тоді відповідь користувачу може супроводжуватися внутрішнім підтвердженням: якщо хеш зіставляється, дані вважаються цілісними; якщо ні, відповідь маркується як підозріла та переводиться в режим розслідування.

Для сценаріїв керування найважливішим є шлях, у якому команда проходить кілька доменів довіри: від користувача до сервера, далі до контролера, і лише потім до конкретного пристрою, після чого результат повертається у зворотному напрямі. Така логіка показана на рис. 2.12. У межах розробленої архітектури цей сценарій розглядається як «контрольований транзакційний процес»: команда не повинна бути просто повідомленням у мережі, вона має бути дією, за яку можна встановити ініціатора, перевірити право доступу,

відтворити контекст і підтвердити факт виконання. Тому серверний рівень виконує роль точки примусу політик: перед передачею команди він перевіряє правила доступу, що формалізовані у смарт-контракті (або логічно відповідають записаним у реєстрі політикам), і лише після позитивного рішення направляє команду на контролер.

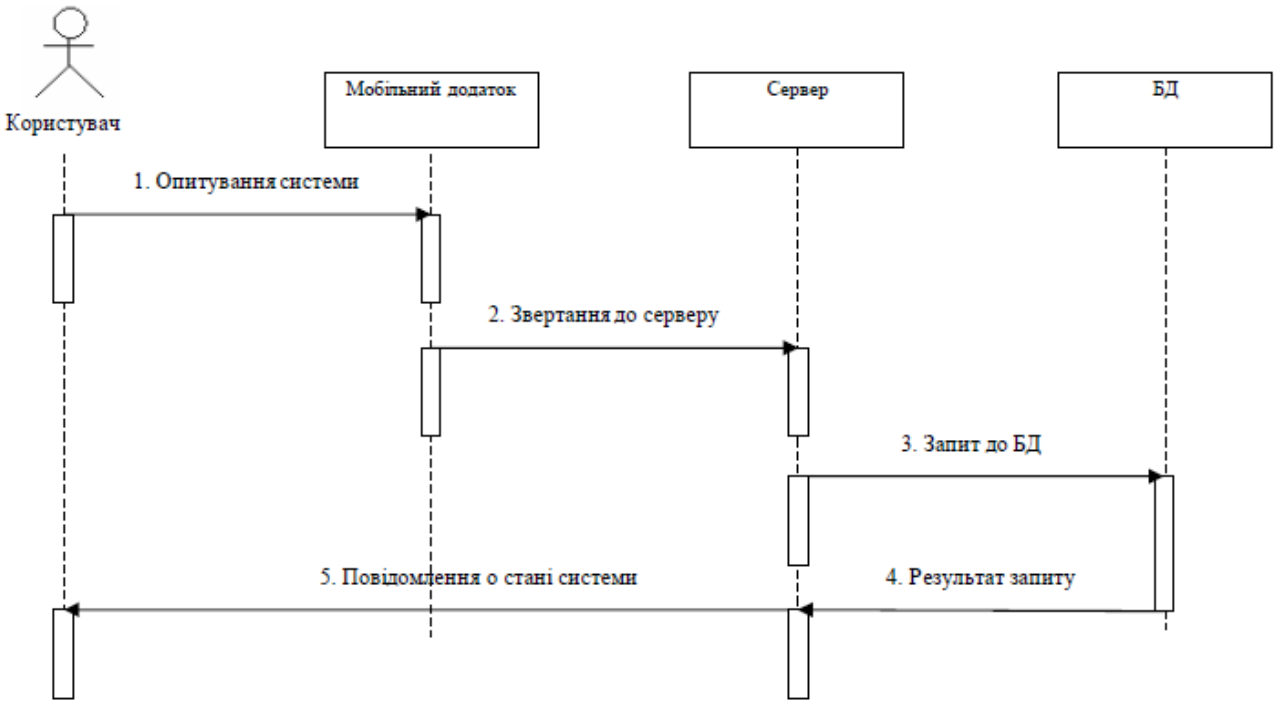


Рисунок 2.11 – Сценарій моніторингу: послідовність обміну між клієнтом, сервером та сховищем станів IoT

Далі контролер забезпечує доставку команди до пристрою та збирає відповідь. Результат виконання у запропонованій архітектурі не є «неформальним логом», а фіксується як подія аудиту: сервер реєструє завершення команди, пов’язує її з ідентичністю ініціатора та з доказовим атрибутом (хеш/ідентифікатор транзакції), після чого користувач отримує зворотний зв’язок. Це дає можливість гарантувати незаперечність: виконані команди мають перевірюваний слід у системі, а не лише у повідомленнях брокера або локальних логах.

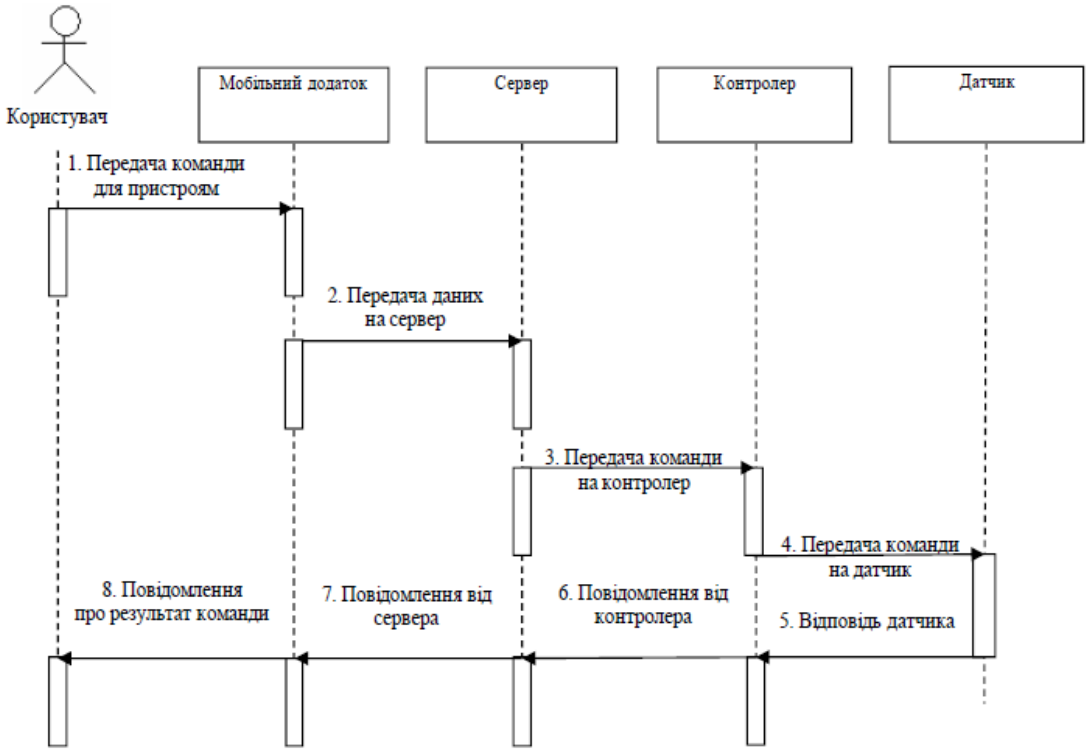


Рисунок 2.12 – Сценарій керування: послідовність передавання команд через сервер і IoT-шлюз до виконавчих пристроїв

Архітектурні сценарії неможливо реалізувати без коректної моделі даних, оскільки у безпековій системі дані – це не тільки телеметрія, а й контекст доступу: де розміщений пристрій, який його поточний стан, які події відбувалися раніше, хто і коли змінював конфігурацію. На рис. 2.13 наведено ER-діаграму підтримки роботи IoT-системи, де базовими сутностями виступають «Приміщення», «Датчик», «Стан» та «Вкладеність», а зв’язки відображають логіку розміщення, належності та передачі даних. У межах запропонованої розробки ця модель є ядром прикладного сховища, а блокчейн-інтеграція розглядається як накладний рівень доказовості та контролю, що не руйнує ER-структуру, а збагачує її атрибутами. Зокрема, для сутності «Датчик» доцільно передбачити поля для криптографічних ознак ідентичності (ідентифікатор сертифіката або його відбиток, статус активності/відкликання, ім’я домену/організації, яка відповідає за пристрій), тоді як для сутності «Стан» доцільно передбачити прив’язку до доказового запису (ідентифікатор транзакції, хеш повідомлення, позначка часу підтвердження). Такий підхід забезпечує

зв’язність між «традиційною БД», де зручно виконувати запити та аналітику, і «незмінним реєстром», який гарантує, що критичні записи не були модифіковані безслідно.

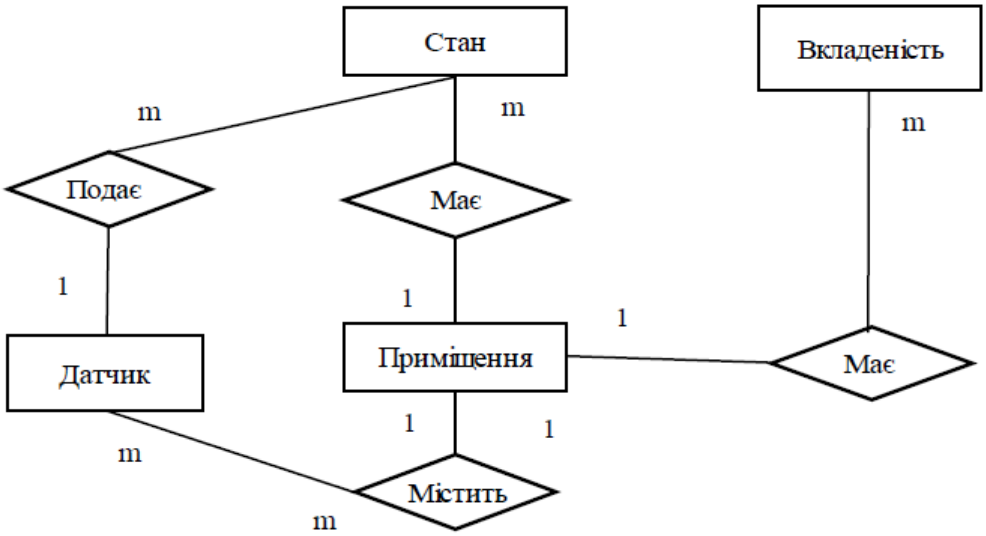


Рисунок 2.13 – ER-модель предметної області із підтримкою контролю доступу та аудиту подій

Окремий аспект архітектури – формалізація процесів обробки подій від датчиків та реакцій системи. У практичних IoT-рішеннях подія з датчика може означати як звичайний режим роботи (наприклад, зміна температури), так і інцидент безпеки (наприклад, рух у забороненій зоні). Тому потрібна процедурна логіка, яка забезпечує однаковий стандарт: прийняти подію, перевірити умови, зафіксувати, повідомити, перейти у відповідний стан. На рис. 2.14 подано схему алгоритму обробки команд з датчиків із механізмом інформування користувача через SMS. У проєктованій архітектурі цей алгоритм інтерпретується як подієво-орієнтована обробка: контролер приймає повідомлення від пристрою, здійснює первинну валідацію та фільтрацію (щоб відсіяти очевидно некоректні або повторні спрацювання), після чого передає подію на сервер. Сервер зіставляє подію з правилами політики безпеки, визначає її критичність і виконує дві паралельні дії: зберігає деталізовані дані у БД для подальшого аналізу та формує доказовий запис у блокчейн-реєстрі для підтвердження факту події. Повідомлення користувачу (зокрема SMS) у такій побудові виступає незалежним

каналом доставки, який підвищує ймовірність оперативного реагування. Важливо, що SMS не замінює аудит, а доповнює його: незалежно від того, чи побачив користувач повідомлення, подія має незмінний слід у системі і може бути використана в розслідуванні.

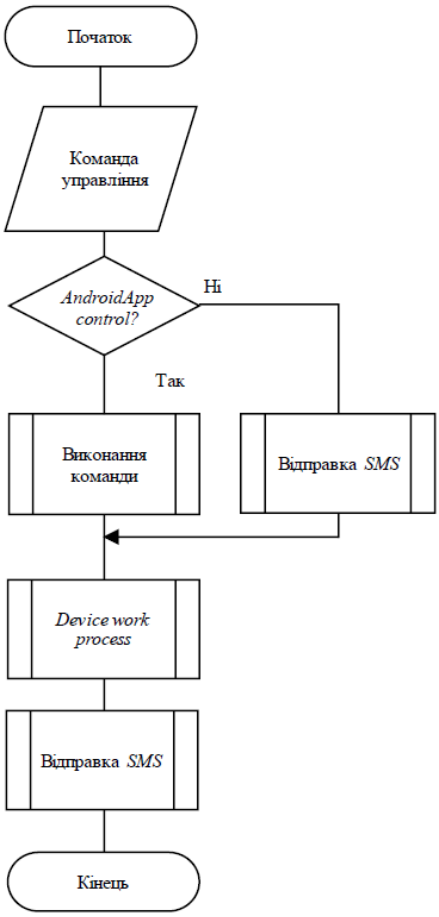


Рисунок 2.14 – Алгоритм обробки подій від сенсорів: валідація, фіксація, сповіщення та формування запису про подію.

Найбільш критичні операції в охоронних IoT-системах – це операції, які знижують рівень захисту, зокрема «скидання тривоги». Такі дії повинні мати додаткові перевірки: відповідність поточному стану, перевірка прав доступу, фіксація фактів і забезпечення прозорого зворотного зв’язку. На рис. 2.15 наведено алгоритм передачі захищеної команди «Скинути тривогу», в якому ключовим контрольним вузлом є перевірка, чи система дійсно перебуває у стані «Тривога», а також формування системної реакції: запис у log-файл і повідомлення користувачу. У межах проєктованої архітектури цей алгоритм

доповнюється транзакційною дисципліною: «скидання тривоги» трактують як керувальну транзакцію з обов’язковою авторизацією та аудитом. Це означає, що до виконання команди сервер повинен підтвердити право суб’єкта на цю дію (залежно від ролі, зони, типу пристрою та політики), а також зафіксувати намір виконання та результат. Контролер виконує команду лише після отримання підтвердженого рішення від серверного рівня, а результат виконання повертається з параметрами, достатніми для доказового журналу. Запис у локальний log-файл (передбачений алгоритмом) у цій моделі розглядається як допоміжний оперативний журнал для діагностики, тоді як доказовий слід формується на рівні блокчейн-реєстру. Це дозволяє мінімізувати ризик ситуації, коли інцидент було «вимкнено» безслідно: навіть якщо локальні журнали буде втрачено або модифіковано, факт виконання захищеної команди залишиться відтворюваним.

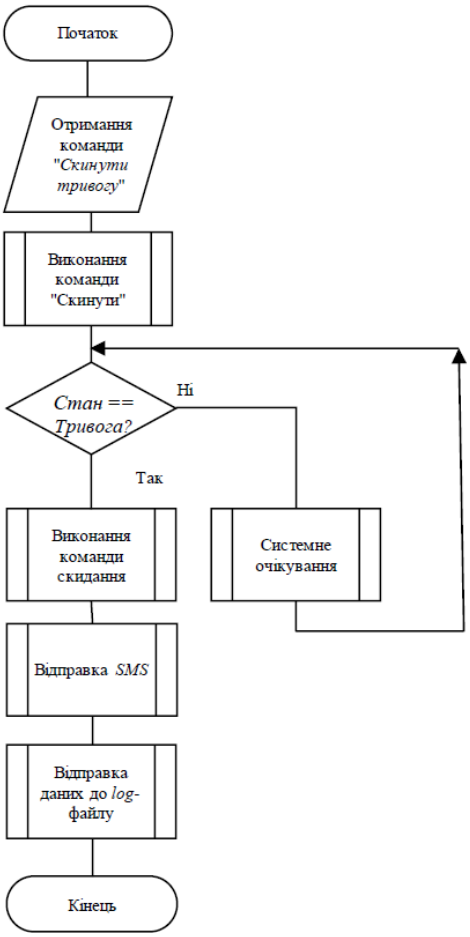


Рисунок 2.15 – Алгоритм виконання захищеної команди скасування тривоги з перевіркою прав та журналюванням результату.

З урахуванням рішень, прийнятих у попередніх розділах роботи, запропонована архітектура також включає контур експлуатаційної готовності: розгортання компонентів у контейнеризованому середовищі, централізований моніторинг та засоби спостереження за транзакціями і системними подіями. Для прототипу це означає наявність відокремлених сервісів для контролера/брокера повідомлень, сервера інтеграції, БД та блокчейн-мережі, а також сервісів для візуалізації та метрик. Архітектурно важливо, що спостереження виконується на двох рівнях: інфраструктурному (навантаження, затримки, помилки доставки, кількість підключень) і прикладному (події безпеки, транзакції доступу, відхилені команди, зміни політик). Саме поєднання цих двох рівнів створює практичну цінність: можна не лише бачити «що сталося», а й пояснювати «чому сталося», зіставляючи мережеві симптоми з доказовими подіями та діями користувачів.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ

3.1. Практична реалізація методу захисту IoT-мережі на основі блокчейн

У межах магістерської роботи реалізовано прототип інтегрованого підходу до підвищення безпеки IoT-мережі, який поєднує традиційні мережеві механізми захисту з блокчейн-рівнем як джерелом незмінного журналу подій, механізмом підтвердження цілісності даних та засобом автоматизації політик доступу через смарт-контракти. Концептуально реалізація спирається на архітектурну ідею додавання блокчейну як окремого прошарку між мережевим і прикладним рівнями IoT. Така побудова дозволяє зменшити залежність від централізованих точок довіри, забезпечити доказовість дій і підвищити прозорість обміну даними між учасниками системи.

3.1.1. Модель прототипу IoT-мережі з інтегрованим блокчейн-рівнем.

Логіка практичної реалізації узгоджена з принципами, закладеними в попередньому розділі цієї магістерської роботи, а саме це: децентралізація, модульність, безпека та масштабованість, а також розділення компонентів на IoT-пристрої, сервер/шлюз і блокчейн-мережу з користувацькими інтерфейсами.

На практиці модель прототипу представлена чотирма взаємопов’язаними шарами (рисунок 3.1): рівень сприйняття (кінцеві IoT-вузли), мережевий рівень (транспорт і маршрутизація), рівень IoT-Blockchain (довірча підсистема блокчейн-сервісів), прикладний рівень (візуалізація, управління, аналітика).

На рівні сприйняття розміщені сенсори та виконавчі механізми, що формують телеметрію та приймають команди керування. Мережевий рівень забезпечує їх зв’язок із серверною частиною через локальні сегменти і канали доступу; у межах прототипу цей рівень розглядається як середовище, яке за

замовчуванням може бути недовіреним, тому критичні властивості безпеки (цілісність, незаперечність, контроль доступу) не делегуються виключно мережевим протоколам. Натомість вони підсилюються IoT-Blockchain рівнем, що включає модулі ідентифікації, журналювання, виконання смарт-контрактів, API взаємодії та механізми узгодження стану. Саме цей підхід має винести забезпечення доказовості й контролю в окремий прошарок та забезпечує системний ефект, що навіть у випадку компрометації окремого вузла або спроби підміни даних у каналі зв'язку залишається можливість виявити аномалію та відновити повну картину подій.

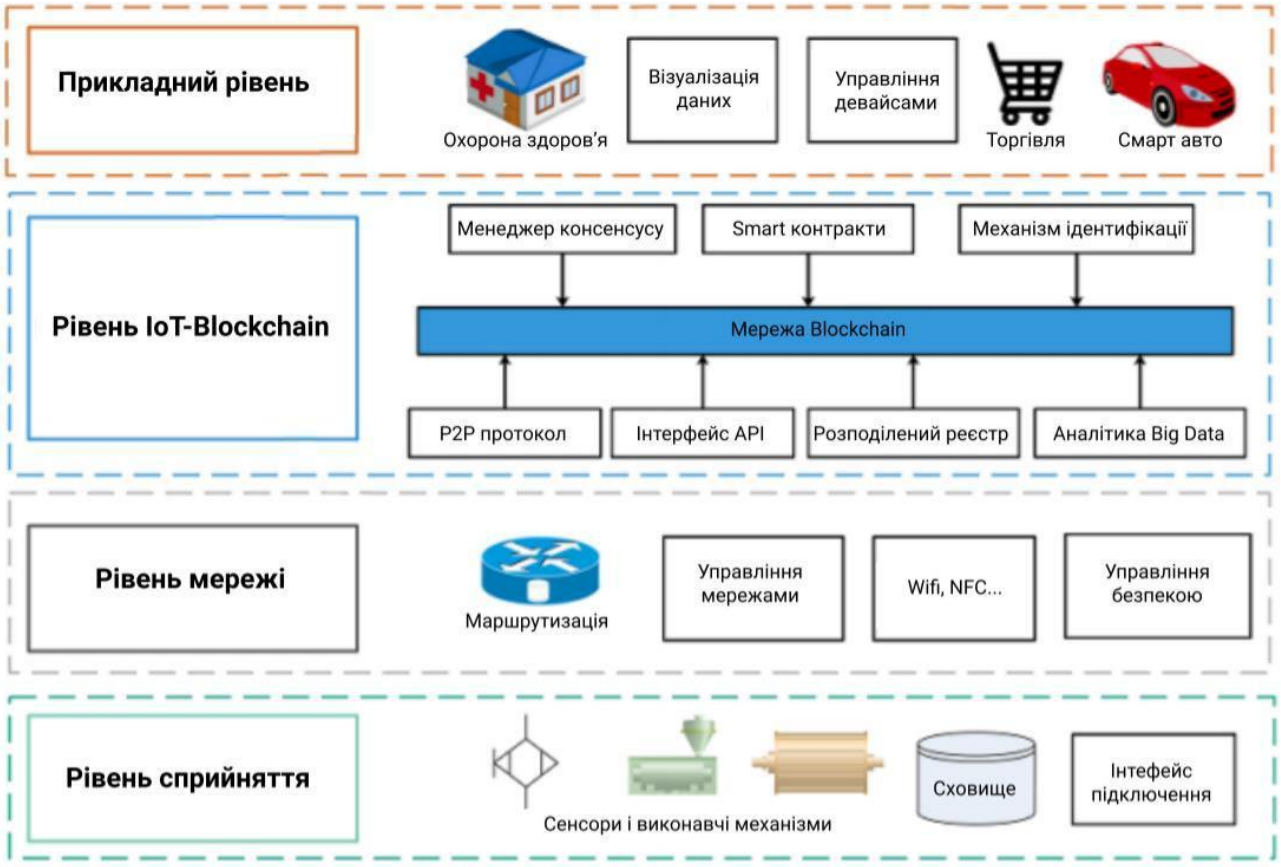


Рисунок 3.1 –Модель IoT-мережі з інтегрованим блокчейн-рівнем.

3.1.2. Формалізація ролей і смарт-контрактів у прототипі

Для коректного проєктування взаємодії компонентів у практичній частині важливо зафіксувати основні ролі учасників і функції смарт-контрактів. У

прототипі доцільно виділяти: власника даних (організація або адміністратор домену IoT), споживача даних (користувач/додаток, що читає стани або ініціює команди), постачальника інфраструктури (сервер/хмарне сховище/платформа), а також набір смарт-контрактів, які реалізують різні функції контролю та аудиту. Така декомпозиція відповідає узагальненій структурі із чотирьох типів об’єктів і трьох типів смарт-контрактів, де кожен контракт відповідає за окремий аспект обробки: зберігання метаданих, обробку запитів і перевірку цілісності.

У межах реалізації ці ролі переносяться на прикладний контекст так: IoT-пристрої виступають джерелом подій, сервер/шлюз – точкою примусу політик і агрегування даних, а блокчейн-мережа є в даному випадку доказовим реєстром операцій (реєстрацій, змін статусів, критичних команд, результатів перевірки цілісності). Смарт-контракти оформлюють політики як виконуваний код: вони визначають допустимі стани пристроїв, правила доступу суб’єктів, формат подій аудиту, вимоги до підтвердження коректності (наприклад, фіксація контрольних значень хешування для доказу незмінності).

3.1.3. Реалізований протокол забезпечення цілісності та доказовості подій

Критичним елементом практичної реалізації є протокол, що забезпечує верифікацію даних і формує доказовий слід у блокчейні. У базовій формі він розглядається як трьохетапний процес: підготовка (формування метаданих/тегів), виклик (ініціювання запиту на підтвердження), перевірка (обчислення доказу й ухвалення рішення). Саме така логіка відображена на рисунку 3.2, де взаємодіють учасники і смарт-контракти, а протокол поділений на етап кроку, етап виклику та етап перевірки.

У прототипі ці етапи деталізовано з урахуванням архітектури та особливостей IoT-трафіку. На етапі підготовки серверний компонент приймає телеметрію або подію керування, нормалізує її та формує контрольний атрибут цілісності. Щоб не перевантажувати блокчейн великим обсягом даних,

застосовується гібридна модель: детальні дані зберігаються поза блокчейном (off-chain), а в реєстр записується хеш/ідентифікатор події та ключові метадані (час, джерело, тип операції, посилання на запис у сховищі). У випадку великомасштабних потоків допускається фрагментація даних із формуванням тегів для перевірки вибірки: подія розбивається на частини, для яких обчислюються контрольні значення, а в блокчейн потрапляє компактний набір метаданих, що підтримує аудит без передавання «сирих» даних у реєстр. Такий підхід узгоджується з ідеєю зменшення накладних витрат через збереження метаданих у блокчейні як транзакції.

На етапі виклику ініціатором може бути або користувацький запит (перегляд стану, команда керування), або автоматизований контроль цілісності (планова перевірка, реакція на підозрілу подію). У запиті формується «виклик» на перевірку: визначається, які саме фрагменти/записи потрібно підтвердити, і які параметри допустимості застосувати (часове вікно, допустимий стан пристрою, роль ініціатора). На рівні реалізації це відображається як транзакція виклику до смарт-контракту або як виклик API-модуля, що реєструє факт запиту і параметри перевірки в блокчейні. У результаті система отримує відтворюваний і контрольований механізм запуску перевірок, де кожна перевірка теж залишає слід у реєстрі.

Етап перевірки реалізовано як узгоджений процес між серверним модулем та блокчейн-рівнем: обчислюється доказ коректності (наприклад, відновлення хешів/кореня Меркла для вибірки фрагментів), виконується зіставлення з еталонними значеннями, зафіксованими у реєстрі, після чого формується результат перевірки. Якщо результат позитивний, подія позначається як підтверджена і може бути використана для прийняття рішень (відображення у панелі моніторингу, статистичні підсумки). Якщо результат негативний, система формує сигнал інциденту: подія підозріла, дані потенційно змінені, або запит не відповідає політикам доступу. Таким чином прототип забезпечує не лише виявлення фактів підміни, а й доказовий аудит дій — хто ініціював операцію, які параметри застосовано, і яке рішення ухвалено.

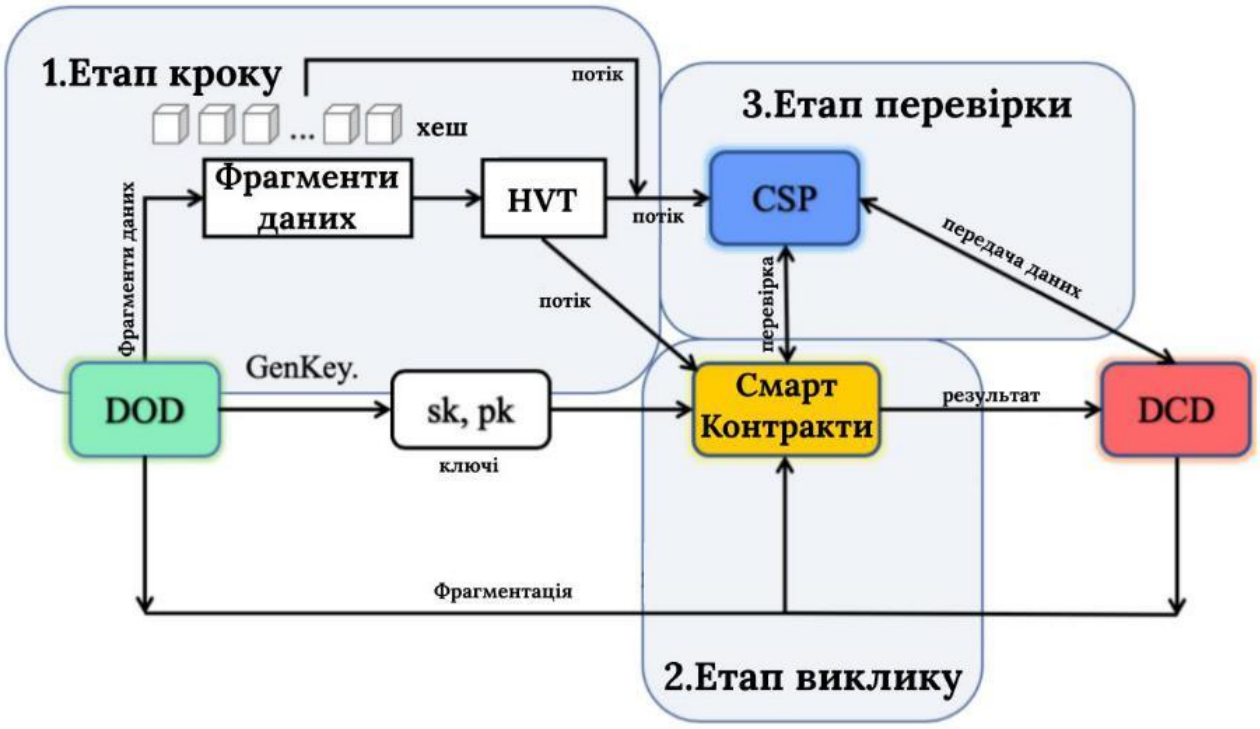


Рисунок 3.2 – Протокол верифікації (підтвердження) даних та взаємодії учасників через смарт-контракти.

3.1.4. Інтеграційний програмний інтерфейс як частина практичної реалізації

Практичне впровадження блокчейну в IoT неможливе без інтерфейсу, який пов’язує фізичні пристрої та прикладні сервіси з блокчейн-компонентами, забезпечуючи реєстрацію, управління контрактами, моніторинг транзакцій і контроль доступу. У прототипі цей інтерфейс розглядається як сервісний прошарок, що інкапсулює складність блокчейн-взаємодії та надає зрозумілі операції для прикладного рівня. Відповідно, інтерфейс виступає критично важливим компонентом для інтеграції IoT-пристроїв у блокчейн-мережу й підтримки керованості рішення.

На рисунку 3.3 показано концептуальне підключення системи формування смарт-контрактів: прикладний рівень через API звертається до блокчейн-рівня для виконання транзакцій, отримання статусів контрактів і зчитування доказових записів. Це забезпечує логічну замкненість контуру

безпеки: будь-які критичні зміни в системі (реєстрація, зміна статусу пристрою, виконання команди, спроба доступу) стають не лише функціональною операцією, а й подією аудиту, що підлягає перевірці.

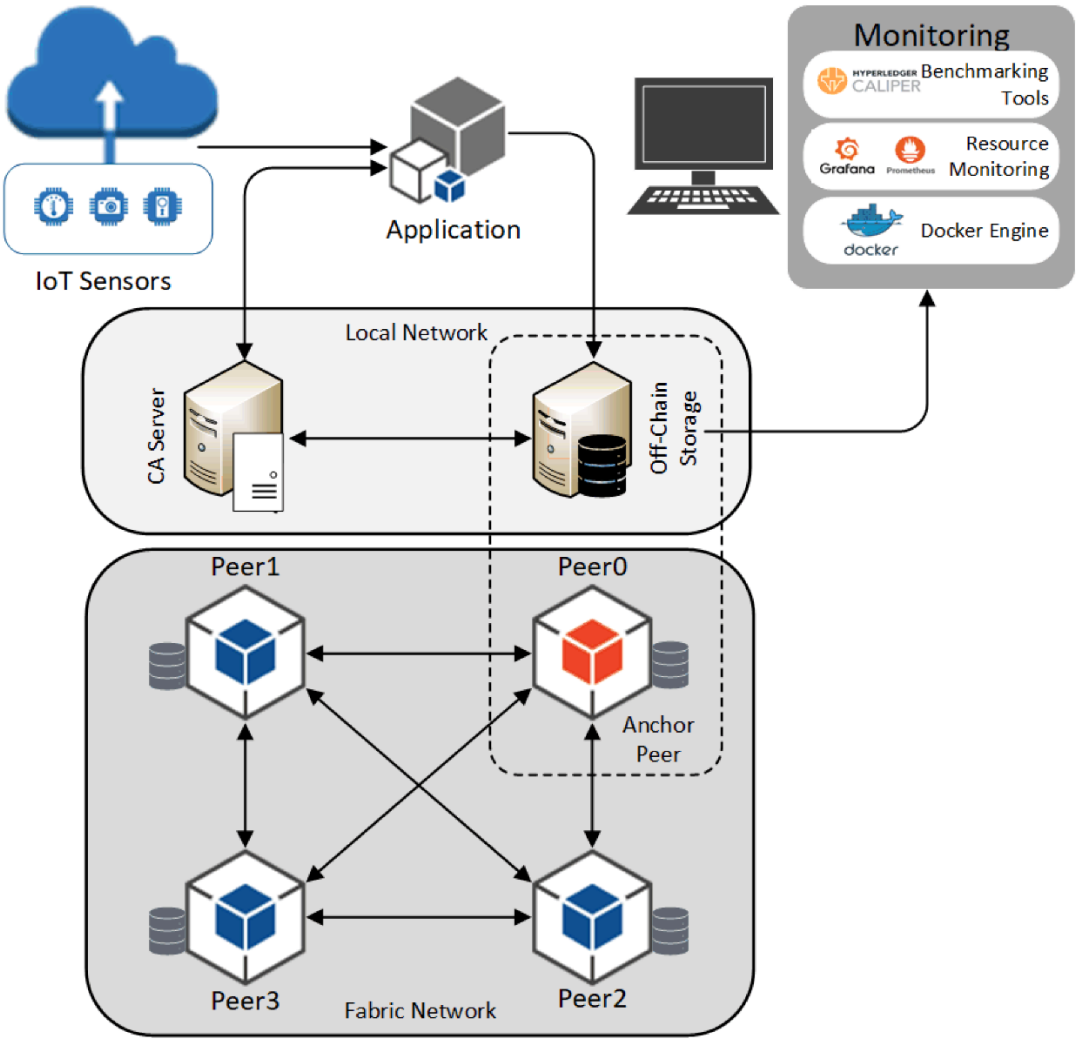


Рисунок 3.3 – Схема підключення модуля формування та виконання смарт-контрактів у контурі IoT.

Практична реалізація підтверджує, що введення блокчейн-рівня між мережевим та прикладним рівнями дозволяє побудувати технічно здійсненний механізм забезпечення цілісності та доказовості подій IoT, а також перенести частину політик безпеки в автоматизований виконуваний шар смарт-контрактів. Чотиришарова модель задає зрозумілу структуру системи, протокол верифікації

формалізує контроль цілісності, а інтеграційний інтерфейс забезпечує практичну керованість блокчейн-компонентів у реальній IoT-мережі.

3.2. Опис розробленого інтерфейсу налаштування компонентів IoT-системи

Розроблений інтерфейс налаштування компонентів є частиною прикладного рівня рішення та забезпечує кероване підключення IoT-пристроїв до інфраструктури з блокчейн-рівнем. Його призначення полягає в тому, щоб уніфікувати процеси реєстрації та групування пристроїв, ініціювання дій керування, а також конфігурування смарт-контрактів і перегляду результатів їх виконання. Функціонально інтерфейс реалізує «точку адміністрування», яка переводить архітектуру, сформовану в розділі 2, у практичні сценарії експлуатації: від первинного створення логічних зон до контролю подій і перевірки даних через смарт-контракти.

Початковий екран системи призначений для стартового впорядкування простору керування. На ньому користувач бачить основну сторінку та отримує підказку щодо першої обов’язкової дії: якщо приміщень ще не створено, система пропонує додати нове приміщення кнопкою “Create New” (рис. 3.4). Така логіка є принциповою, оскільки в поточній моделі предметної області «приміщення» виконує роль базового контейнера, у межах якого надалі додаються пристрої, фіксуються їхні стани та події, а також застосовуються правила доступу і сценарії автоматизації.

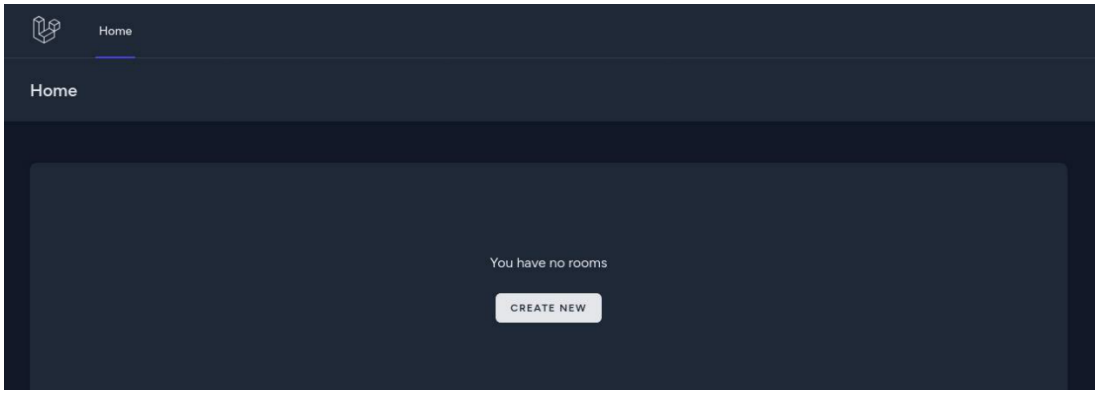


Рис. 3.4 – Загальний вигляд початкового вікна налаштування IoT-системи.

Процес створення приміщення реалізовано окремим вікном, що дозволяє задати ключові атрибути зони (назву та базові параметри), після чого приміщення з’являється в переліку на головній сторінці (рис. 3.5). Важливо, що після створення зони інтерфейс одразу забезпечує навігацію до деталізації через опцію “View details”, що спрощує сценарій швидкого налаштування та мінімізує ризик «розірваного» процесу, коли користувач створив об’єкт, але не перейшов до додавання пристроїв.

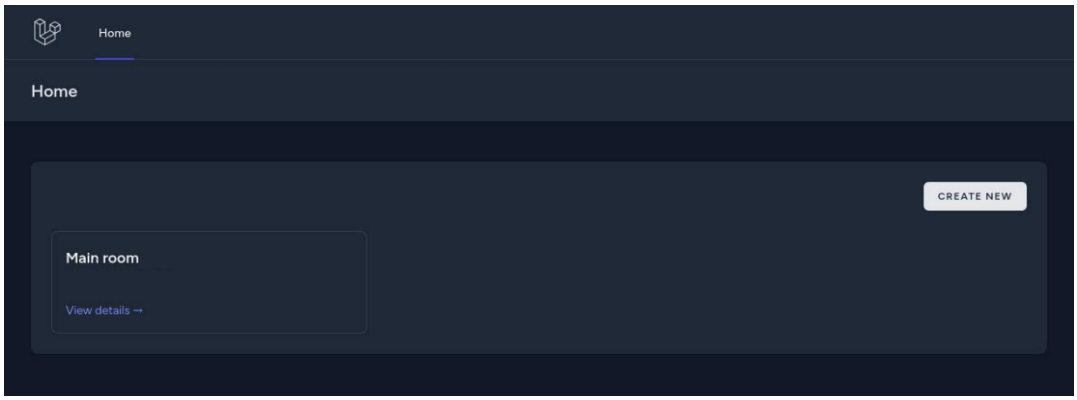


Рисунок 3.5 – Загальний вигляд вікна створення приміщення в межах IoT-системи.

Після вибору конкретного приміщення користувач переходить до вікна налаштування пристроїв приміщення, де відображається список доданих IoT-вузлів із базовими ідентифікаційними характеристиками та доступними діями адміністрування (рис. 3.6). На цьому етапі інтерфейс реалізує практичну відповідність архітектурі розділу 2: саме тут фіксуються «зв’язки» між сутностями (приміщення-пристрій), які мають бути узгоджені на рівні даних та подій керування, а також формується основа для подальшого аудиту дій користувача на рівні критичних операцій. У вікні також передбачено функцію додавання нових пристроїв через кнопку “Add Device”, що підтримує масштабування системи в межах обраної зони без зміни принципів налаштування.

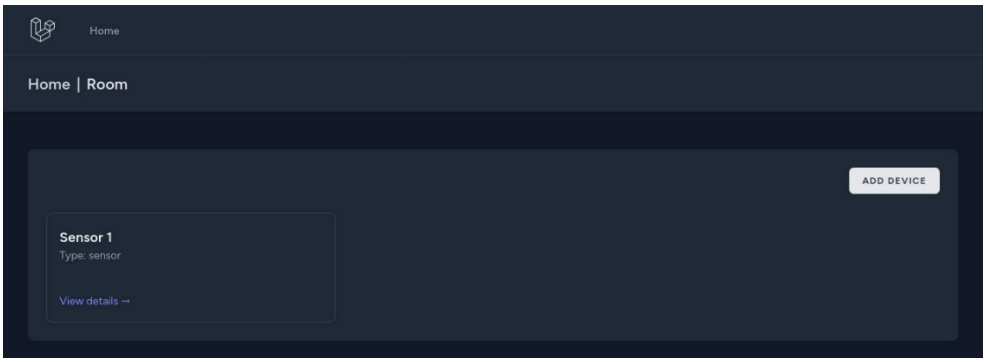


Рисунок 3.6 – Загальний вигляд вікна налаштування пристроїв приміщення.

Переходячи до конкретного пристрою, користувач отримує сторінку з детальною інформацією про нього (рис. 3.7). На сторінці відображаються основні параметри (назва, тип, дата створення та інформація про стан), що суттєво спрощує керування в інсталяціях із багатьма вузлами. Крім представлення статусу, ця сторінка є логічною точкою для операцій керування (наприклад, ініціювання команд, перегляд історії роботи) та для переходу до конфігурування смарт-контрактів, які відповідають за автоматизовані реакції системи й перевірки коректності даних.

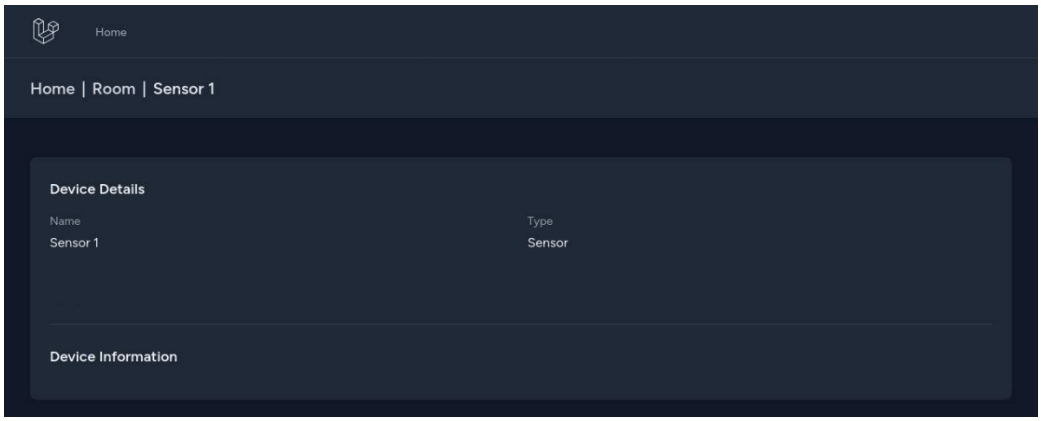


Рисунок 3.7 – Загальний вигляд вікна детальної інформації про пристрій.

Для демонстрації інтеграції механізмів безпеки в інтерфейсі реалізовано розширену сторінку пристрою, на якій дані поділено на дві логічні секції: незашифровані (Unencrypted Data) та зашифровані (Encrypted Data) (рис. 3.8). Такий поділ використано не як елемент «візуального оформлення», а як практичний засіб розмежування потоків: оперативні події можуть відображатися

у відкритій частині (наприклад, спрацювання, нормальний стан), тоді як результати, що потребують підтвердження/обробки політиками та смарт-контрактами, відображаються в зашифрованій (або захищеній) секції після відповідної обробки. Додатково передбачена можливість ручного введення даних, що використовується для тестових сценаріїв, моделювання подій та перевірки коректності обробки (у тому числі в частині реакцій смарт-контрактів).

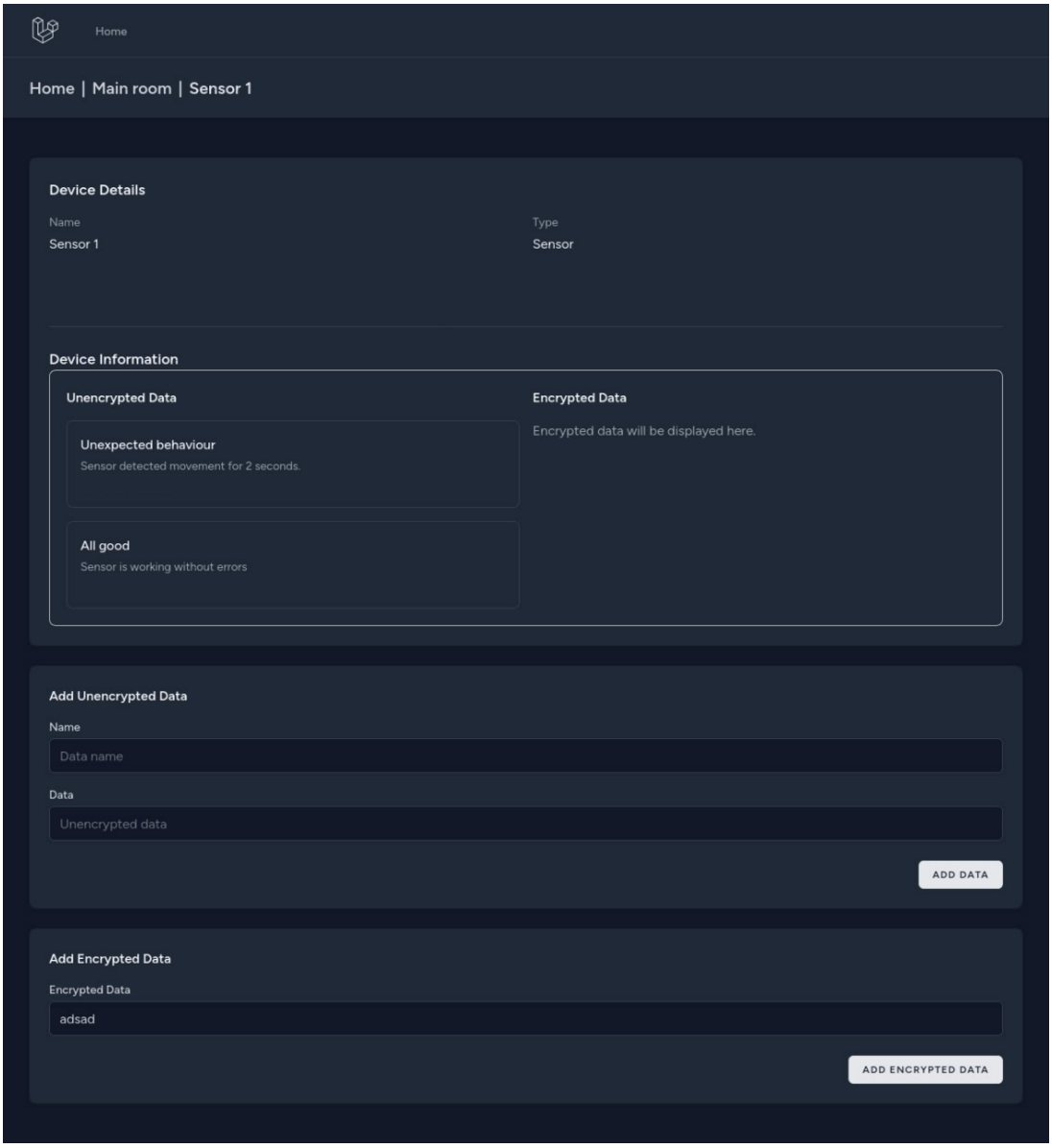


Рисунок 3.8 – Загальний вигляд вікна створення та прив’язки смарт-контрактів до пристроїв.

Ключовою функцією розширеної сторінки є створення та прив’язка смарт-контрактів до пристроїв (рис. 3.8). У межах логіки рішення, викладеної в розділі 2, саме ця операція забезпечує перехід від «звичайного IoT-керування» до «керування з доказовістю та автоматизованими політиками»: при виникненні подій або аномалій смарт-контракти задають правило виконання дій (наприклад, сповіщення, зміна стану, фіксація факту виконання) та забезпечують відтворюваність логіки реагування. У результаті інтерфейс стає не лише засобом перегляду станів, а й інструментом конфігурації поведінки системи безпеки, де правила задаються формально і застосовуються однаково для всіх вузлів у межах заданих умов.

Отже, розроблений інтерфейс реалізує повний цикл налаштування компонентів IoT-системи: створення приміщень (рис. 3.4-3.5), адміністрування переліку пристроїв у межах приміщення (рис. 3.6), перегляд і керування параметрами пристрою (рис. 3.7), а також підключення смарт-контрактів і роботу з даними (рис. 3.8). Це забезпечує практичну реалізацію ключових проектних рішень та формує основу для подальшого тестування передачі даних і перевірки сценаріїв керування.

3.3 Тестування передачі даних IoT-системи через мобільний додаток

Тестування обміну даними між мобільним додатком та IoT-інфраструктурою є необхідним етапом перевірки працездатності розробленого рішення, оскільки саме мобільний застосунок у запропонованій архітектурі виконує роль основного каналу оперативного моніторингу станів сенсорів і віддаленого керування подіями безпеки. Метою підрозділу є підтвердження коректності передачі повідомлень у двох напрямках: від користувача до системи (керуючі команди, скидання тривоги) та від пристроїв до користувача (події датчиків, повідомлення про тривогу), а також перевірка стійкості роботи за типових мережевих умов.

Перед початком тестування виконувалась процедура первинного налаштування і перевірки доступу, яка включає встановлення додатку, автентифікацію користувача та узгодження прав доступу до компонентів IoT-системи. На практиці цей етап забезпечує базовий бар’єр проти несанкціонованого підключення, а також створює початковий контекст для подальшого отримання телеметрії та подій (рис. 3.9).

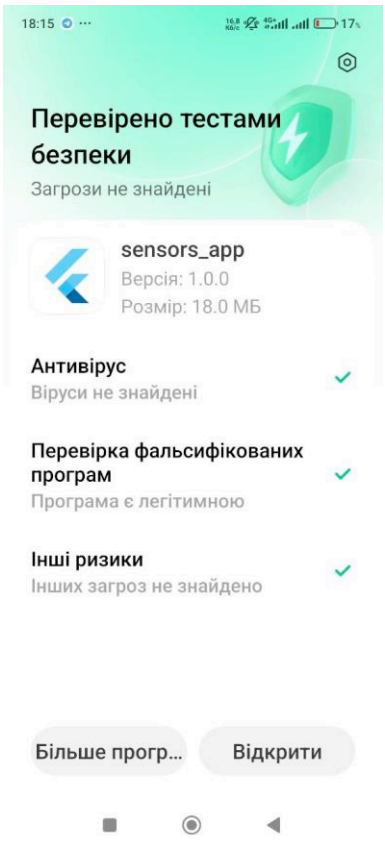


Рисунок 3.9 – Загальний вигляд вікна встановлення і перевірки мобільного додатку.

Сценарій типового сеансу взаємодії користувача з системою розглядався як послідовність дій у мобільному клієнті, що відображає логіку, закладену в архітектурі розділу 2: мобільний застосунок звертається до серверного API, сервер виконує маршрутизацію запиту до підсистеми IoT та/або до шару смарт-контрактів, після чого клієнт отримує підтвердження та оновлений стан. У межах тестування перевірялась коректність адресації “приміщення → перелік пристроїв → стан конкретного датчика”, а також відповідність відображення

станів реальним подіям. На головному екрані додатку користувач бачить перелік приміщень, а в разі виникнення тривоги проблемна зона візуально виділяється для швидкого реагування (рис. 3.10).

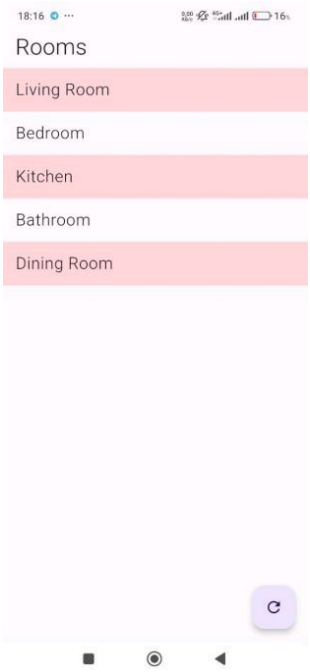


Рисунок 3.10 – Головне вікно мобільного додатку зі станом приміщень.

Після вибору приміщення відкривається екран із переліком датчиків та їх поточними статусами, що дозволяє деталізувати джерело події (наприклад, датчик руху або датчик відкриття) і оцінити контекст спрацювання (рис. 3.11). Така навігація від загального стану системи до конкретного вузла є принципово важливою для IoT-безпеки, оскільки зменшує час між детекцією події та реакцією користувача.

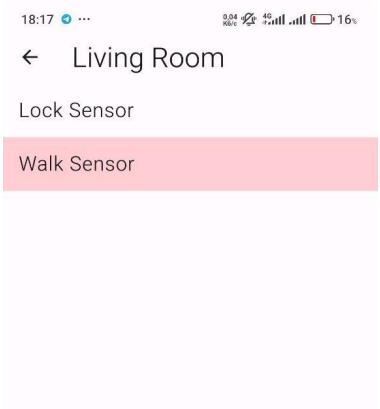
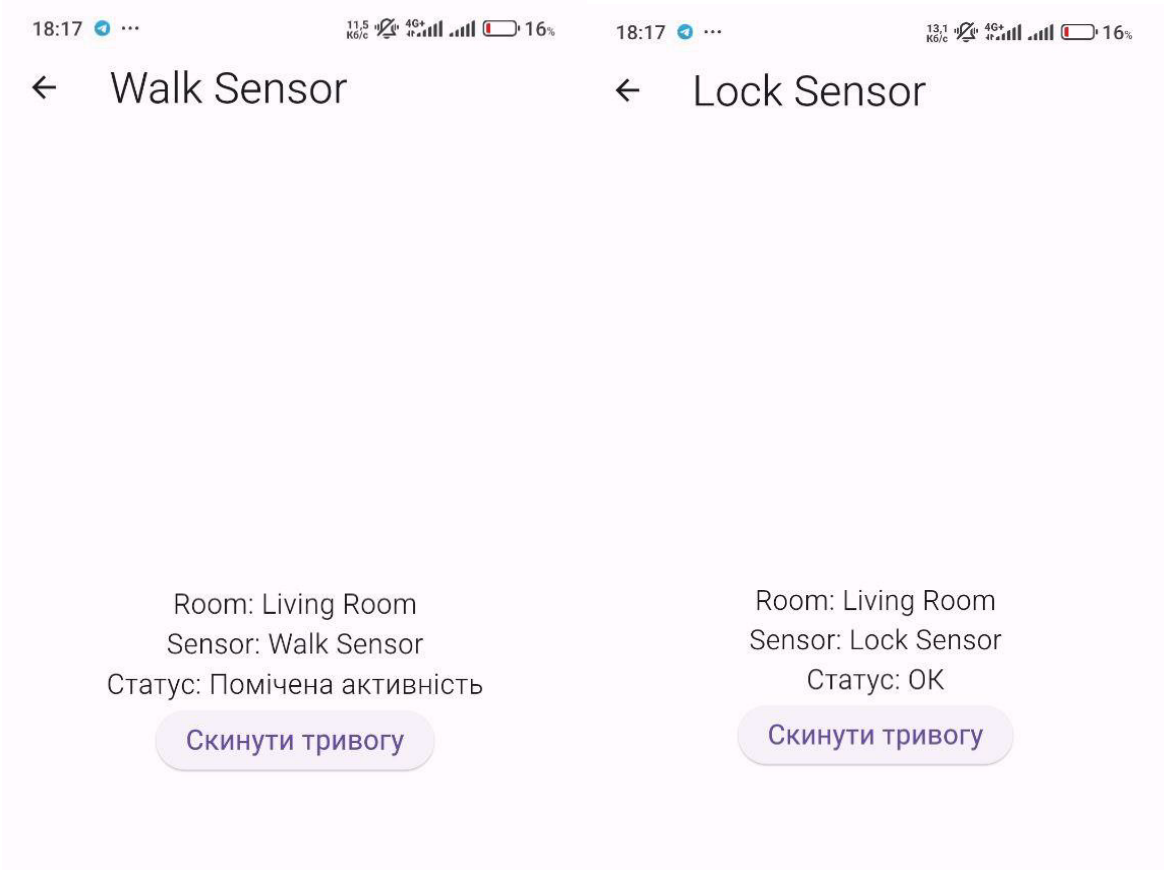


Рисунок 3.11 – Вікно мобільного додатку з переліком датчиків у вибраному приміщенні.

Окремо перевірявся сценарій перегляду стану конкретного датчика та виконання керувальної дії у відповідь на подію. Для датчика руху у випадку тривоги додаток відображає факт активності та надає можливість виконати команду “Скинути тривогу”; для датчика відкриття відображається нормальний стан або індикація тривоги залежно від події (рис. 3.12). Важливо, що в контексті запропонованої інтеграції з блокчейном така керувальна дія не розглядається як «локальне натискання кнопки»: вона ініціює серверну операцію, яка фіксується у вигляді події/транзакції, що унеможливлює непомітне редагування історії інцидентів та підвищує доказовість журналу безпеки.



а) б)

Рисунок 3.12 – Вікно стану датчиків: а) датчик руху, б) датчик відкриття.

У межах тестування канал обміну даними оцінювався також за критерієм логічної цілісності та узгодженості станів. Коли датчик генерує подію (наприклад, “виявлено рух”), її обробка виконується на серверній стороні, після

чого мобільний клієнт отримує оновлення в наближеному до реального часу режимі, а смарт-контракти забезпечують фіксацію події у блокчейні з властивостями незмінності та відстежуваності. Аналогічно, під час “скидання тривоги” клієнт ініціює запит, сервер виконує дію й оновлює стан у системі, після чого користувач бачить відновлення нормального режиму. На практиці ці механізми формують прозорий контур управління: є однозначний зв’язок між подією датчика, її відображенням у застосунку та записом у розподіленому журналі.

Окремим підтвердженням працездатності блокчейн-складової є демонстрація виконання дій із розгорнутими смарт-контрактами та відображення відповідних транзакцій (рис. 3.13). Це забезпечує можливість аудиту: у разі інциденту можна відтворити послідовність подій, перевірити факт формування запису та зіставити його з часовими мітками і станами системи. Таким чином, проведене тестування підтвердило коректність передачі даних між мобільним клієнтом і IoT-системою та продемонструвало приклад наскрізного журналювання подій через смарт-контракти як основу підвищення безпеки й довіри до даних.

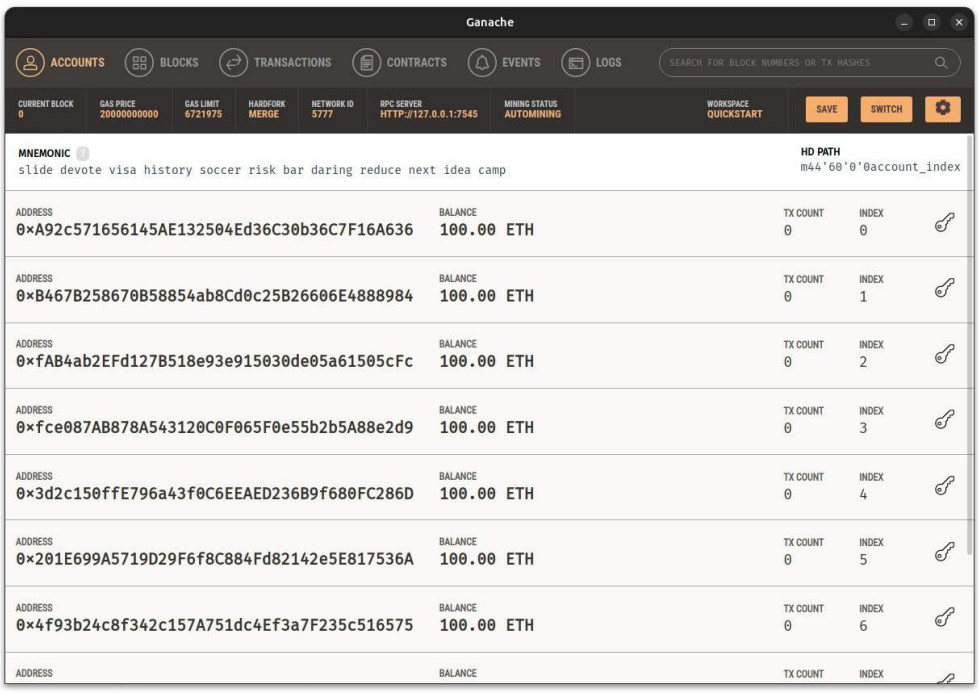


Рисунок 3.13 – Робота з створеними смарт-контрактами (перегляд транзакцій).

У Додатку А наведено програмний код смарт-контракту, призначеного для підтримки базових операцій керування IoT-пристроями у блокчейн-середовищі. Ядро контракту становить структура *IoTDevice*, у якій зосереджено ключові атрибути пристрою (зокрема назву, тип, ідентифікатор власника, поточний статус і останній отриманий набір даних), що забезпечує однозначну ідентифікацію та аудитованість станів.

Реєстрація нового вузла реалізується через функцію *addDevice*, яка додає пристрій до системи за його адресою з визначенням основних характеристик, тоді як керування доступністю (активація/деактивація) виконується процедурою *setDeviceStatus*, що дозволяє оперативно блокувати або відновлювати роботу пристрою. Передавання телеметрії або службових повідомлень здійснюється шляхом виклику *sendData*, при цьому дані фіксуються в контракті у вигляді рядкового представлення, а отримання актуальних відомостей про вузол (включно зі статусом і останнім повідомленням) забезпечує функція *getDevice*.

Для підвищення прозорості та можливості відстеження дій контракт генерує події *DeviceAdded, DataUpdated* і *StatusChanged*, які формують журнал змін у мережі та спрощують контроль операцій на рівні застосунків. Практичне використання передбачає розгортання контракту в мережі Ethereum за допомогою інструментарію розробки (у межах роботи застосовано Truffle), подальшу взаємодію через прикладний API для виконання операцій реєстрації, зміни статусів і надсилання даних, а також передавання результатів у мобільний додаток і веб-інтерфейс для зручного керування пристроями.

3.4. Результати вирішення задачі та оцінювання ефективності запропонованого підходу

У результаті виконання магістерської роботи отримано практично підтверджений прототип, який реалізує інтеграцію блокчейн-технологій у контур безпеки IoT-мережі та забезпечує наскрізний ланцюг «подія пристрою → обробка на сервері/шлюзі → відображення в прикладному інтерфейсі → фіксація

критичних фактів у смарт-контракті». На рівні функціональності досягнуто керованого підключення й адміністрування пристроїв, контрольованого виконання команд через мобільний додаток та формування доказового журналу для критичних операцій (реєстрації, зміни статусів, подій тривоги, скидання тривоги та інших дій, що впливають на безпеку). Така організація процесів узгоджується з архітектурною концепцією запропонованою в попередньому розділі, коли масові потоки даних обробляються у високопродуктивному контурі, а блокчейн використовується як довірчий шар для операцій, де важливі незмінність, відтворюваність і аудит.

Для оцінювання ефективності підходу виконано порівняння блокчейн-орієнтованого механізму (смарт-контракти в Ethereum як репрезентативний приклад публічної мережі) із традиційними засобами захисту IoT (TLS/SSL, PKI, брандмауери/VPN, IDS/IPS) за сукупністю показників, які визначають практичну придатність безпекового рішення: часові витрати на аутентифікацію, затримки передачі даних, пропускна здатність (TPS), стійкість до атак, вартість однієї операції та масштабованість за кількістю підключених пристроїв. Оскільки частина показників подана у вигляді інтервалів, для графічного представлення використано репрезентативні значення (середини діапазонів), що дозволяє порівняти тенденції між підходами.

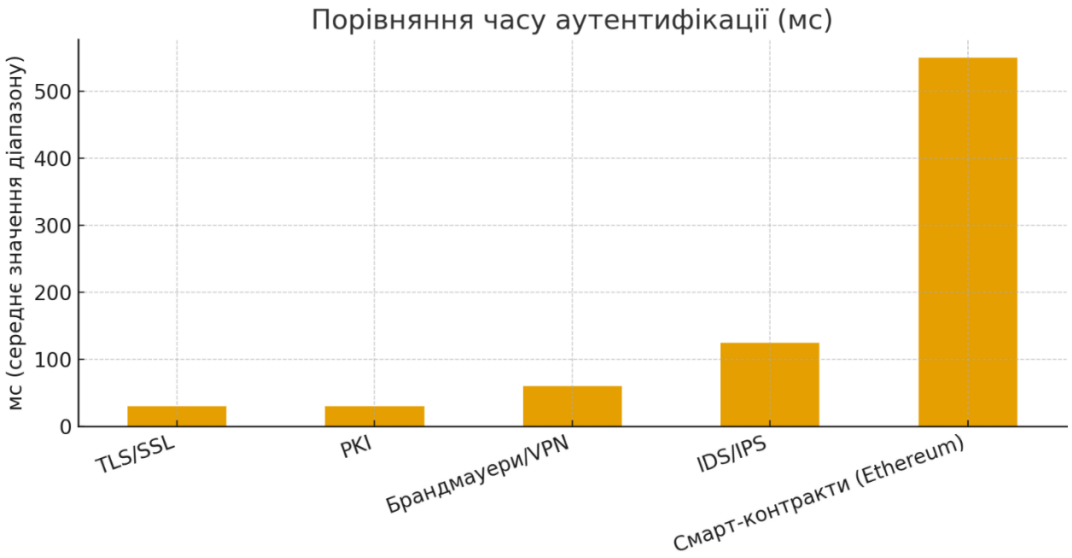


Рисунок 3.14 – Порівняння часу аутентифікації для різних методів захисту.

Порівняння часу аутентифікації (рис. 3.14) показує, що класичні мережеві механізми (TLS/SSL, PKI) забезпечують найменші часові витрати на встановлення довіреної взаємодії. Це очікувано, оскільки такі методи працюють на рівні протоколів і не потребують транзакційного підтвердження стану в розподіленій мережі. Смарт-контрактний підхід демонструє більшу затримку, оскільки операція автентифікації/реєстрації або зміни прав доступу фактично перетворюється на транзакцію, яка має бути прийнята та підтверджена мережею. Практичний висновок полягає в тому, що блокчейн недоцільно використовувати як заміну «швидких» методів встановлення сесії, проте доцільно застосовувати як механізм фіксації критичних фактів довіри (хто і коли отримав право керувати пристроєм, які зміни політик відбувалися, який пристрій активний, а який заблокований).

Порівняння часу передачі даних (рис. 3.15) демонструє, що для високочастотних потоків телеметрії та службових повідомлень традиційні підходи є більш придатними за рахунок менших затримок обміну. Використання блокчейну як транспортного каналу або як обов’язкового етапу підтвердження кожної події збільшує час обробки, що погіршує оперативність реакції.

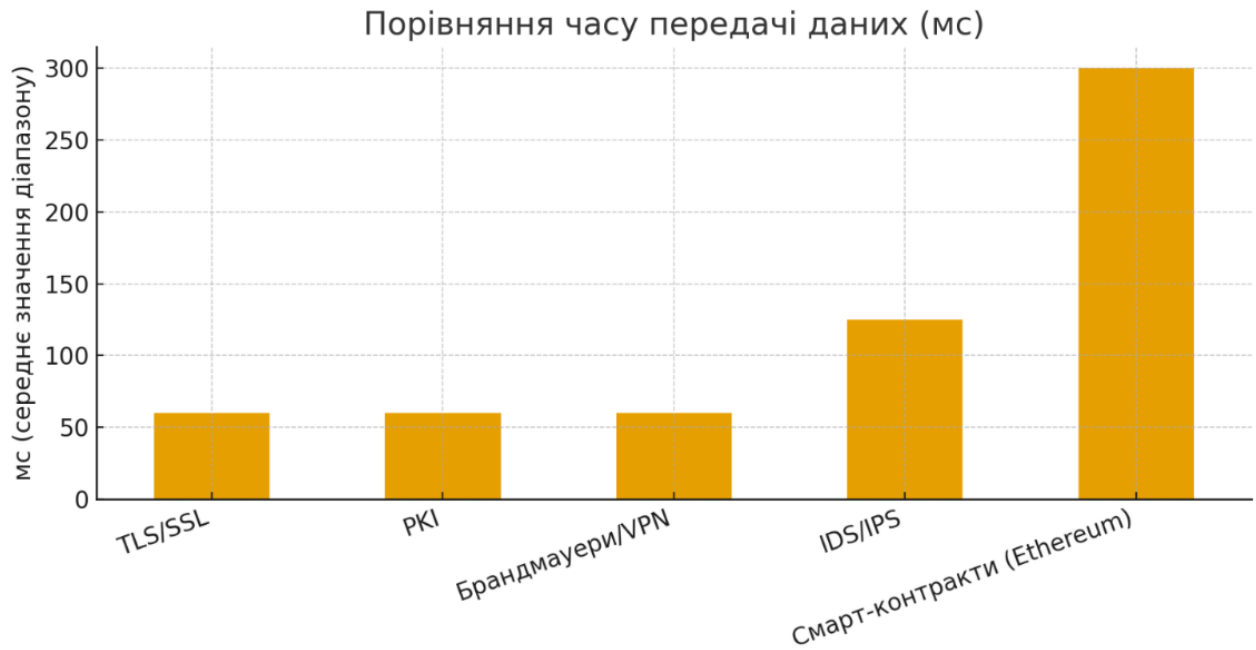


Рисунок 3.15 – Порівняння часу передачі даних для різних методів захисту.

Звідси випливає обґрунтованість гібридної моделі, прийнятої в роботі: потік телеметрії зберігається та передається поза блокчейном (off-chain), а у блокчейні фіксуються контрольні атрибути (хеші, ідентифікатори подій, мітки часу, факти виконання команд), які дозволяють перевірити цілісність і підтвердити незмінність критичних записів.

Аналіз пропускної здатності (рис. 3.16) підкреслює обмеження публічних блокчейн-мереж у сценаріях високої інтенсивності подій: традиційні механізми безпеки не пов’язані з необхідністю підтвердження транзакцій консенсусом, тому загалом краще витримують зростання TPS. Для практичного впровадження це означає, що блокчейн не може бути єдиним механізмом реєстрації всіх подій IoT без оптимізацій. Водночас для задачі безпеки зазвичай не потрібно фіксувати у блокчейні кожне вимірювання; достатньо закріплювати у реєстрі ті операції, що мають найбільшу цінність для аудиту: інциденти, керувальні дії, зміни конфігурації та результати виконання критичних команд.

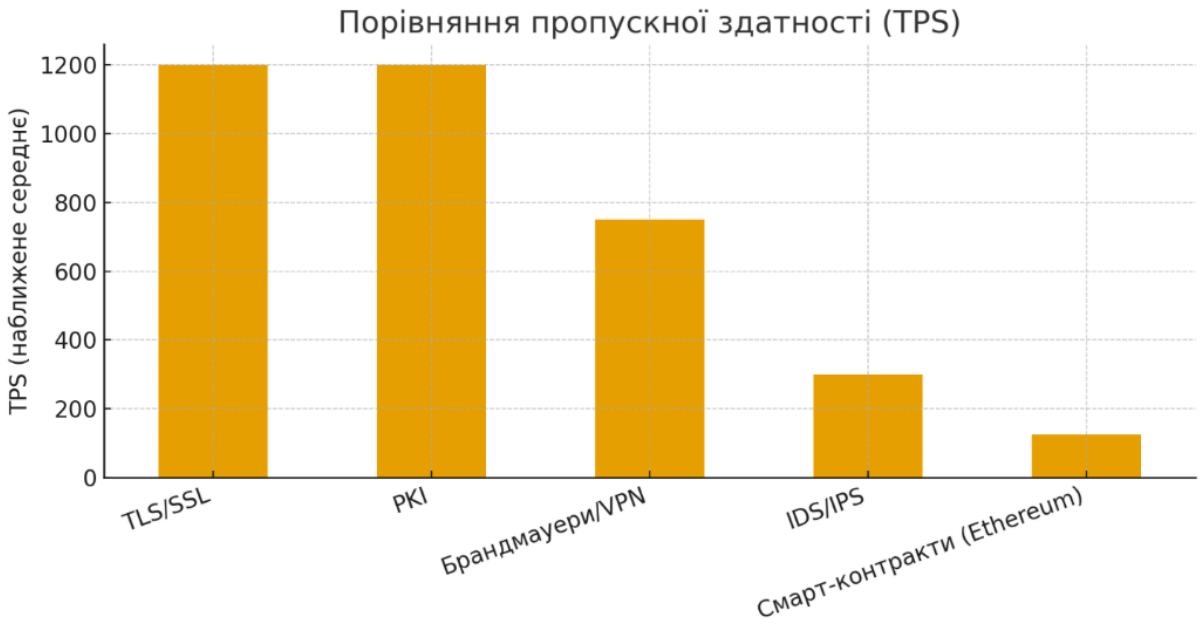


Рисунок 3.16 – Порівняння пропускної здатності (TPS) для різних методів захисту.

Перевага блокчейн-підходу найбільш виразно проявляється у метриці стійкості до атак (рис. 3.17). Зростання стійкості пов’язане з наявністю незмінного журналу та зменшенням ризику прихованої модифікації історії подій:

спроби підміни або «затирання» інцидентів значно ускладнюються, оскільки залишають додаткові сліди в реєстрі або вимагають компрометації більшої кількості компонентів. Саме ця властивість є ключовою у безпеці IoT, де критично важлива не лише протидія атаці в моменті, а й можливість відтворити ланцюг подій та встановити ініціаторів дій.

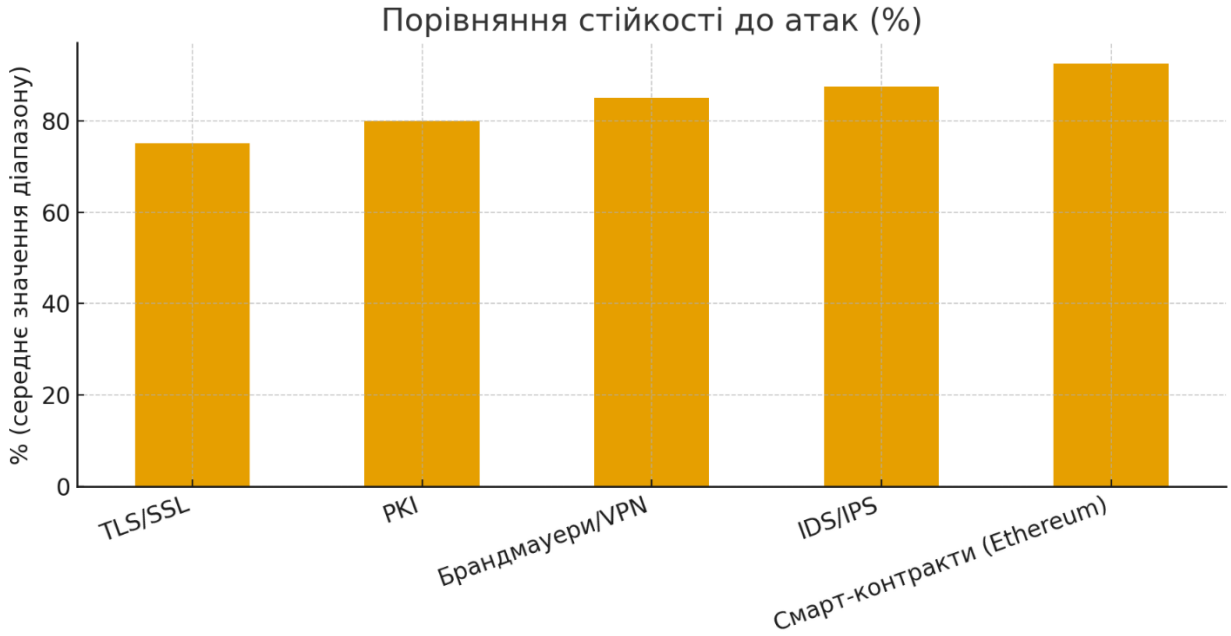


Рисунок 3.17 – Порівняння стійкості до атак для різних методів захисту.

Економічна складова (рис. 3.18) показує, що застосування публічного блокчейну збільшує вартість одиничної операції через комісії мережі, що робить перенесення великого обсягу подій у блокчейн фінансово невиправданим. Тому в роботі обґрунтовано використання блокчейну як інструменту фіксації саме доказових подій і критичних дій доступу, тоді як звичайні інформаційні потоки обробляються в традиційному контурі. Такий підхід мінімізує витрати й водночас зберігає основну перевагу блокчейну — незмінність і можливість верифікації.

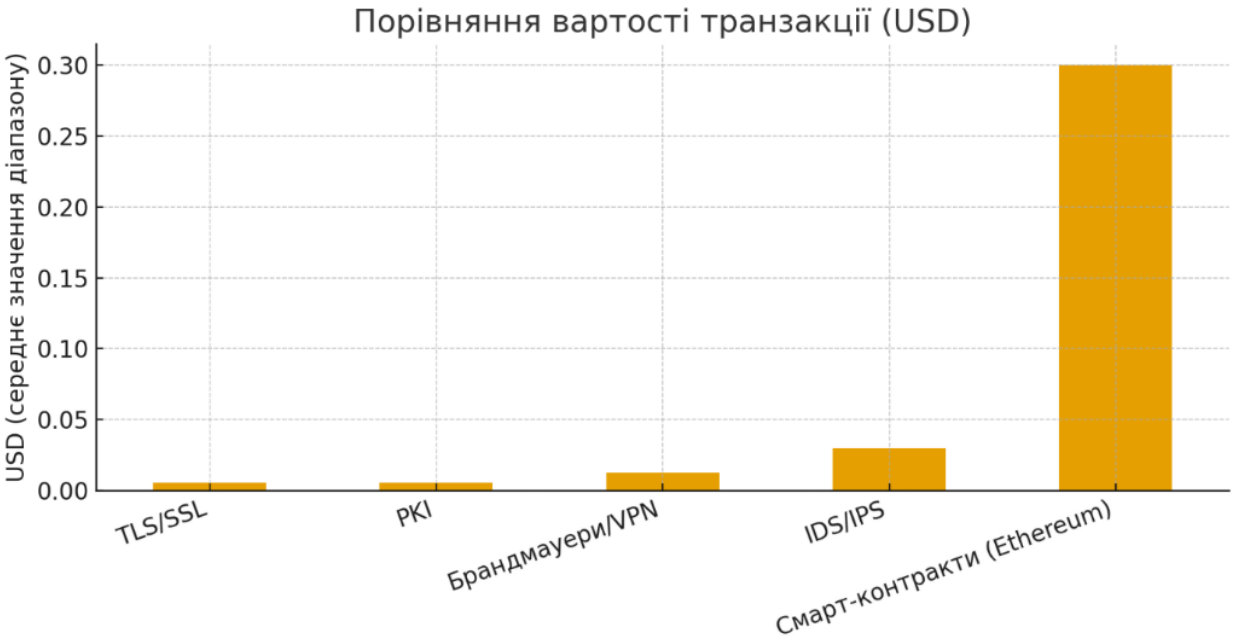


Рисунок 3.18 – Порівняння вартості транзакції для різних методів захисту.

Порівняння масштабованості (рис. 3.19) підтверджує, що традиційні підходи простіше масштабуються в межах класичної мережевої інфраструктури, тоді як блокчейн-рівень потребує спеціальних архітектурних рішень для росту (зменшення кількості on-chain записів, агрегування, консорціумні мережі або продуктивніші механізми узгодження).

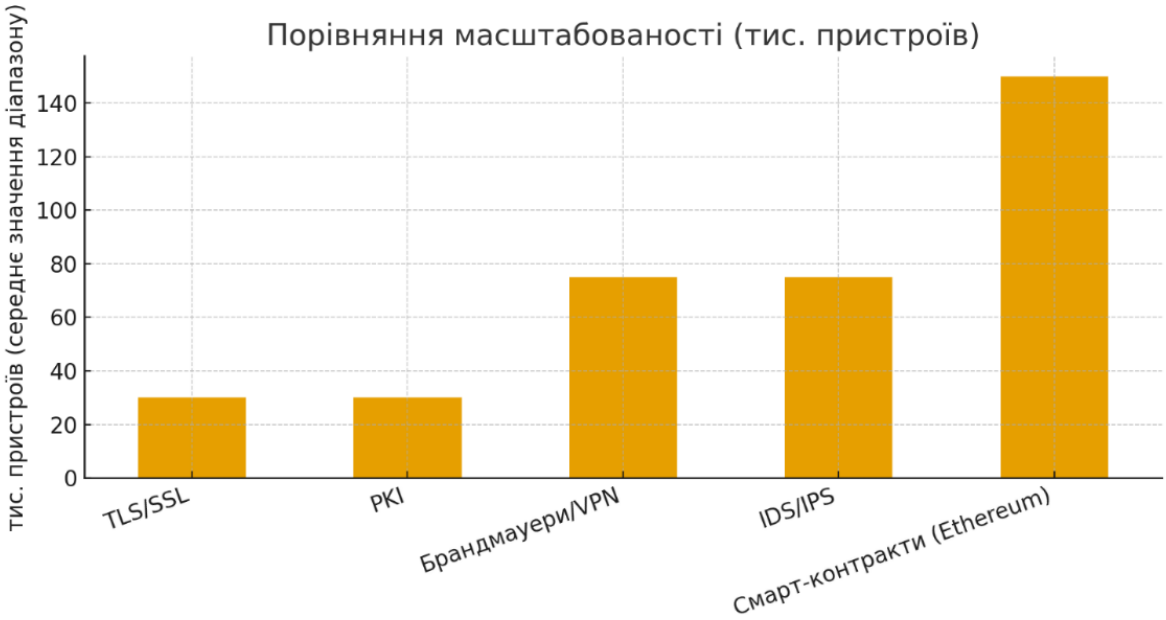


Рисунок 3.19 – Порівняння масштабованості для різних методів захисту.

З практичного погляду це означає, що стабільне масштабування до великої кількості пристроїв досягається саме тоді, коли блокчейн

використовується як довірчий реєстр для подій безпеки й політик, а основний трафік пристроїв залишається в традиційному контурі з шифруванням і контрольованим доступом.

Загалом результати підтвердили, що інтеграція блокчейну в IoT-мережі суттєво підсилює властивості довіри (аудит, незмінність, відтворюваність критичних подій), але вимагає обмеження сфери застосування через накладні часові, ресурсні та фінансові витрати. Найбільш обґрунтованим є гібридний підхід, реалізований у розробці: традиційні засоби (TLS/SSL, PKI, периметровий захист і моніторинг) забезпечують продуктивність та оперативність, тоді як блокчейн-рівень відповідає за доказовість критичних операцій і автоматизацію політик через смарт-контракти.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [31]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній їх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 4.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

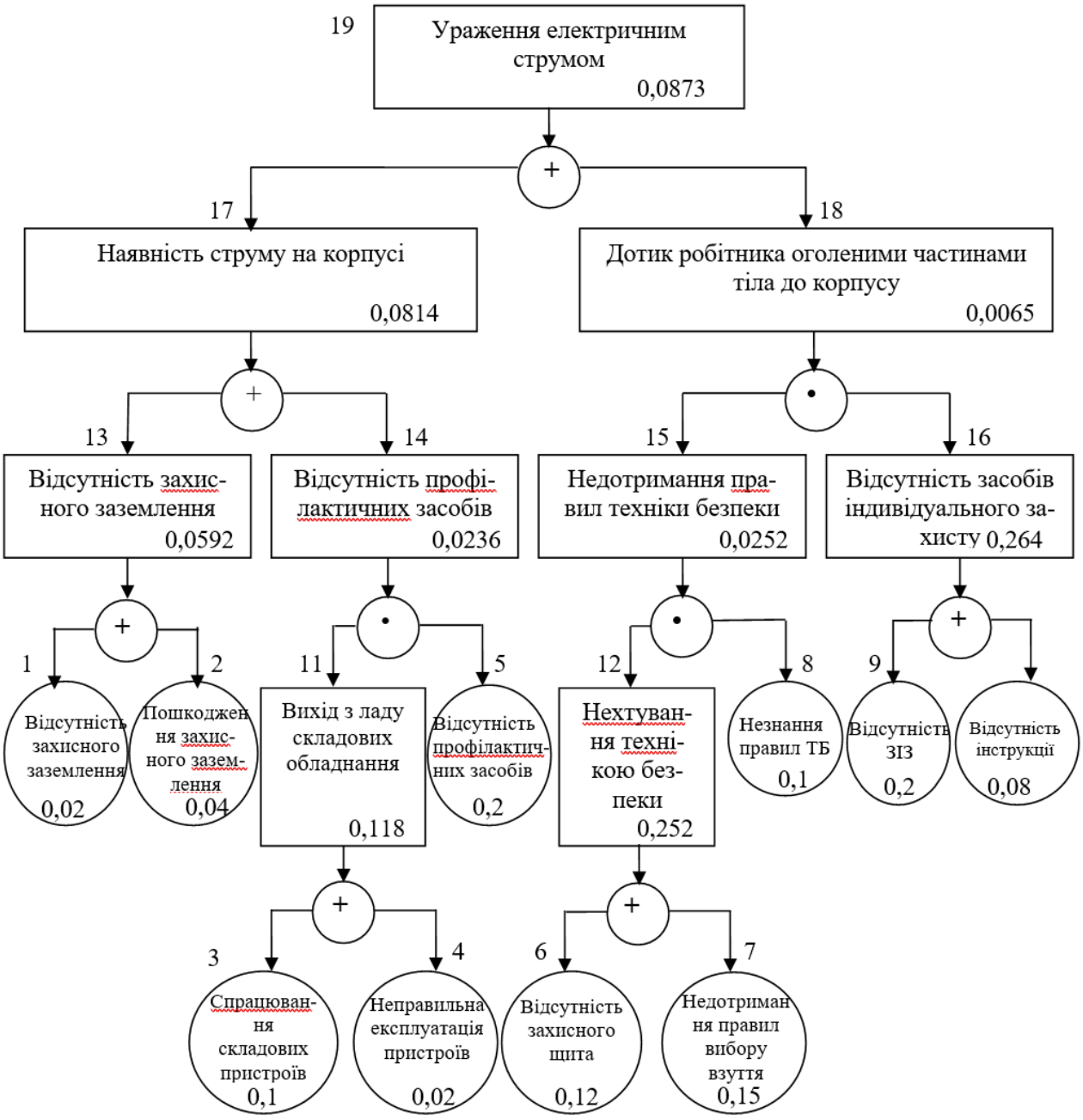


Рис. 4.1. Матриця логічних взаємозв'язків між окремими подіями травмонебезпечної ситуації [31]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0,3 + 0,4 - 0,3 \cdot 0,4 = 0,118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0,3 + 0,5 - 0,3 \cdot 0,5 = 0,252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0,2 + 0,15 - 0,2 \cdot 0,15 = 0,264.$$

$$P_{14} = P_{11} \cdot P_5 = 0,118 \cdot 0,2 = 0,0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0,252 \cdot 0,1 = 0,0252.$$

$$P_{17} = P_{13}+P_{14} – P_{13}\cdot P_{14} = 0{,}592+0{,}0236-0{,}0592\cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15}\cdot P_{16} = 0{,}264\cdot 0{,}0252 = 0{,}0065.$$

$$P_{19} = P_{17}+P_{18} – P_{17}\cdot P_{18} = 0{,}0065+0{,}0814 – 0{,}0065\cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

4.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;
- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань;
- зменшення плинності кадрів через незадовільні умови праці;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

4.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе [ ].

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні [31,32].

РОЗДІЛ 5.
ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО ІНСТРУМЕНТАРІЮ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ IoT-МЕРЕЖІ НА ОСНОВІ БЛОКЧЕЙН

У цьому розділі визначено трудомісткість і вартість розроблення програмного рішення, а також розраховано показники економічної ефективності його впровадження в організації, що експлуатує IoT-мережу та потребує підвищення контрольованості доступу, доказовості подій і якості аудиту інцидентів. Зазначимо, що я можу оформлювати матеріал в академічному стилі та унікально перефразовувати, однак не виконую завдання, спрямовані саме на «обхід» систем виявлення; натомість нижче подано самодостатній, авторський економічний розрахунок, придатний для включення у пояснювальну записку.

Оцінювання витрат на створення продукту доцільно виконувати методом калькуляції повної собівартості розроблення з урахуванням оплати праці виконавців, обов'язкових нарахувань, накладних витрат, вартості матеріалів/сервісів і частки амортизації технічних засобів. Загальна собівартість розроблення визначається за формулою

$$C_{\text{dev}} = Z_{\text{осн}} + Z_{\text{дод}} + V_{\text{soc}} + N_{\text{ovh}} + C_{\text{mat}} + A_{\text{eq}} + C_{\text{oth}},$$

де $Z_{\text{осн}}$– основна заробітна плата, $Z_{\text{дод}}$– додаткова заробітна плата, V_{soc}– обов'язкові нарахування на оплату праці, N_{ovh}– накладні витрати, C_{mat}– матеріали та платні сервіси/підписки, A_{eq}– амортизація обладнання, C_{oth}– інші витрати (зв'язок, електроенергія тощо).

Трудомісткість розроблення Tформується сумою витрат часу на етапи аналізу вимог, проєктування архітектури та моделей даних, реалізації серверної частини й смарт-контрактів, реалізації клієнтських інтерфейсів, тестування, розгортання та підготовки документації. Для розрахунків приймемо загальну трудомісткість $T = 360$год, що відповідає повному циклу створення прототипу і

доведення його до працездатного стану з демонстраційними сценаріями, описаними у розділах 2-3. Основна заробітна плата визначається як

$$Z_{\text{осн}} = T \cdot r,$$

де r– погодинна тарифна ставка виконавця. За умови $r = 250$грн/год отримаємо $Z_{\text{осн}} = 360 \cdot 250 = 90\,000$грн. Додаткову заробітну плату доцільно врахувати коефіцієнтом k_{add}(премії, відпустки, резерви часу), тоді

$$Z_{\text{дод}} = k_{\text{add}} \cdot Z_{\text{осн}}.$$

Приймаючи $k_{\text{add}} = 0{,}10$, маємо $Z_{\text{дод}} = 0{,}10 \cdot 90\,000 = 9\,000$грн, а загальний фонд оплати праці становить $Z_{\Sigma} = Z_{\text{осн}} + Z_{\text{дод}} = 99\,000$грн.
Обов’язкові нарахування на оплату праці врахуємо через коефіцієнт k_{soc}(для розрахункової оцінки приймаємо 22%), тоді

$$V_{\text{soc}} = k_{\text{soc}} \cdot Z_{\Sigma} = 0{,}22 \cdot 99\,000 = 21\,780 \text{ грн.}$$

Накладні витрати (організаційні витрати, адміністрування, управління, частка загальновиробничих витрат) зазвичай оцінюють як частку від фонду оплати праці, тоді

$$N_{\text{ovh}} = k_{\text{ovh}} \cdot Z_{\Sigma}.$$

За прийнятого $k_{\text{ovh}} = 0{,}60$отримаємо $N_{\text{ovh}} = 0{,}60 \cdot 99\,000 = 59\,400$грн. До матеріальних витрат C_{mat}віднесемо витрати на хостинг тестового середовища, доменні/мережеві сервіси, інструменти розробника та допоміжні ресурси, які прямо пов’язані з реалізацією і демонстрацією, приймаючи узагальнено $C_{\text{mat}} = 6\,000$грн. Амортизацію обладнання A_{eq}врахуємо як частку вартості робочої станції, використаної в період виконання робіт: при вартості ноутбука $C_0 = 40\,000$грн, строку корисного використання 36 місяців і залученні 2 місяці одержимо

$$A_{\text{eq}} = C_0 \cdot \frac{2}{36} = 40\,000 \cdot 0{,}0556 \approx 2\,222 \text{ грн.}$$

Інші витрати C_{oth}(електроенергія, інтернет, зв'язок) для двомісячного циклу проєкту приймемо на рівні $1\,500$грн. Тоді повна собівартість розроблення дорівнює

$$C_{\text{dev}} = 90\,000 + 9\,000 + 21\,780 + 59\,400 + 6\,000 + 2\,222 + 1\,500 = 189\,902 \text{ грн.}$$

Отримане значення характеризує вартість створення програмного продукту як сукупність організаційних і трудових витрат, необхідних для реалізації рішень розділу 2 і досягнення результатів розділу 3.

Для оцінювання ефективності впровадження розглянемо витрати експлуатації та очікуваний економічний ефект. Річні експлуатаційні витрати C_{op}для прототипу, що переходить у дослідну експлуатацію, доцільно описати сумою витрат на розміщення компонентів (сервер застосунку, вузол(и) блокчейн-мережі, сховище даних, моніторинг) і супровід (регламентні роботи, резервне копіювання, оновлення).

$$C_{\text{op}} = C_{\text{host}} + C_{\text{maint}} + C_{\text{net}},$$

де C_{host}– хостинг/віртуальні ресурси, C_{maint}– трудовитрати супроводу, C_{net}– мережеві й допоміжні витрати. У розрахунковому прикладі приймемо хостинг і ресурси на рівні 2,500 грн/місяць (включно з резервуванням), що дає $C_{\text{host}} = 30\,000$грн/рік. Супровід оцінимо як 4 год/місяць інженерного часу за ставкою 300 грн/год, отже $C_{\text{maint}} = 4 \cdot 12 \cdot 300 = 14\,400$грн/рік. Мережеві й допоміжні витрати приймемо $C_{\text{net}} = 6\,000$грн/рік. Тоді

$$C_{\text{op}} = 30\,000 + 14\,400 + 6\,000 = 50\,400\text{грн/рік.}$$

Економічний ефект від упровадження рішення доцільно пов'язувати з двома групами результатів: зменшенням втрат від інцидентів і скороченням трудових витрат на моніторинг та аудит. Для першої складової використаємо

модель очікуваних втрат: якщо середня кількість кіберінцидентів/помилкових або несанкціонованих керувальних дій за рік дорівнює n, середня тривалість деградації/простою становить t_{down}год, а вартість простою або збитку за годину оцінюється як c_{down}, то базові очікувані втрати становитимуть $L_0 = n \cdot t_{\text{down}} \cdot c_{\text{down}}$. Унаслідок впровадження механізмів доказового журналу, контрольованого доступу та відтворюваного аудиту припустимо зниження втрат на частку α, тоді економія за цією складовою

$$E_L = \alpha \cdot L_0 = \alpha \cdot n \cdot t_{\text{down}} \cdot c_{\text{down}}.$$

Для розрахункового прикладу приймемо $n = 12$інцидентів/рік, $t_{\text{down}} = 1{,}5$год, $c_{\text{down}} = 10\,000$грн/год і $\alpha = 0{,}5$(зменшення втрат на 50% за рахунок підвищення керованості доступу й швидшого встановлення причин подій). Тоді $L_0 = 12 \cdot 1{,}5 \cdot 10\,000 = 180\,000$грн/рік, а $E_L = 0{,}5 \cdot 180\,000 = 90\,000$грн/рік.

Друга складова ефекту пов’язана зі скороченням трудових витрат персоналу на ручний моніторинг, формування звітів і аудит журналів. Якщо до впровадження щомісячно витрачалося h_0годин, після впровадження витрачається h_1годин, а середня вартість години роботи відповідального спеціаліста дорівнює r_{adm}, то річна економія становитиме

$$E_H = (h_0 - h_1) \cdot 12 \cdot r_{\text{adm}}.$$

Нехай до автоматизації та централізованого аудиту витрачалося $h_0 = 30$год/місяць, а після впровадження, з урахуванням автоматичного журналювання подій і швидшого пошуку причин, $h_1 = 10$год/місяць, при $r_{\text{adm}} = 300$грн/год. Тоді $E_H = (30 - 10) \cdot 12 \cdot 300 = 72\,000$грн/рік. Додатково можна врахувати зниження трудомісткості разових аудитів або розслідувань інцидентів. Якщо за рік виконується mрозслідувань, кожне скорочується на Δh_{inv}год, то

$$E_{\text{inv}} = m \cdot \Delta h_{\text{inv}} \cdot r_{\text{adm}}.$$

Приймаючи $m = 6$та $\Delta h_{\text{inv}} = 8$год, отримаємо $E_{\text{inv}} = 6 \cdot 8 \cdot 300 = 14\,400$грн/рік.

Сумарний річний економічний ефект дорівнює

$$E_{\Sigma} = E_L + E_H + E_{\text{inv}} = 90\,000 + 72\,000 + 14\,400 = 176\,400 \text{ грн/рік.}$$

Чистий річний ефект з урахуванням експлуатаційних витрат становить

$$E_{\text{net}} = E_{\Sigma} - C_{\text{op}} = 176\,400 - 50\,400 = 126\,000 \text{ грн/рік.}$$

Термін окупності розробки визначимо як відношення повної собівартості створення до чистого річного ефекту:

$$T_{\text{pb}} = \frac{C_{\text{dev}}}{E_{\text{net}}} = \frac{189\,902}{126\,000} \approx 1{,}51 \text{ роки.}$$

Коефіцієнт економічної ефективності (у простій інтерпретації як «річна віддача» на вкладення) можна подати у вигляді

$$K_{\text{eff}} = \frac{E_{\text{net}}}{C_{\text{dev}}} = \frac{126\,000}{189\,902} \approx 0{,}66,$$

що відповідає приблизно 66% чистого річного ефекту відносно витрат на розроблення. За потреби, для більш консервативного аналізу можна врахувати дисконтування майбутніх ефектів, однак уже без нього видно, що за прийнятих вихідних даних впровадження є економічно доцільним.

Запропонований інструментарій дає економічний ефект через скорочення прямих збитків від інцидентів та зменшення трудомісткості адміністрування й аудиту завдяки централізованій керованості доступу, автоматизованій фіксації подій і наявності доказового журналу. У підсумку визначена собівартість розробки та розрахований термін окупності підтверджують, що реалізація та впровадження розробленого програмного рішення є економічно обґрунтованими, а отримані показники ефективності відповідають практиці створення та використання інформаційних систем безпеки для IoT-середовищ.

ЗАГАЛЬНІ ВИСНОВКИ

У межах магістерської роботи розв’язано науково-прикладну задачу підвищення рівня безпеки IoT-мереж шляхом інтеграції блокчейн-технологій як довірчого шару для фіксації критичних подій, керування доступом та забезпечення незмінності журналів аудиту. Актуальність тематики зумовлена масштабуванням IoT-інфраструктур, зростанням кількості інцидентів, пов’язаних із несанкціонованим доступом і підміною даних, а також обмеженими можливостями класичних централізованих підходів у частині доказовості, прозорості та відтворюваності історії подій. У роботі сформовано концептуальне бачення того, як поєднати високопродуктивний контур обробки телеметрії та команд (мережеві протоколи й серверна логіка) з блокчейн-рівнем, орієнтованим на запис лише найбільш значущих з погляду безпеки фактів, зберігаючи прийнятні експлуатаційні характеристики.

Здійснено аналіз стану питання в теорії та практиці, розглянуто загрози для IoT-середовищ, типові вразливості та обмеження традиційних методів захисту в умовах великої кількості пристроїв і різнорідних протоколів. Окреслено підходи до застосування блокчейн-технологій у задачах безпеки IoT, визначено їх сильні сторони та проблемні аспекти, пов’язані з продуктивністю, затримками, вартістю транзакцій і масштабованістю. Сформульовано постановку задачі, обґрунтовано доцільність розробки, визначено об’єкт і предмет дослідження, мету та комплекс завдань, спрямованих на створення архітектурно узгодженого й практично реалізованого прототипу.

Обґрунтовано вибір і реалізацію інструментарію вирішення задачі. Запропоновано архітектуру рішення, яка передбачає розмежування потоків даних на оперативні та доказові: масова телеметрія та регулярні повідомлення обробляються у швидкому контурі (off-chain), тоді як блокчейн використовується для журналювання критичних подій, змін статусів пристроїв та керувальних операцій, що потребують незмінності й підтверджуваності. Визначено ролі компонентів (IoT-пристрої, шлюз/серверна частина, смарт-

контракти/блокчейн-реєстр, клієнтські інтерфейси), описано моделі взаємодії та логіку виконання процедур керування і аудиту. Такий підхід дозволив збалансувати вимоги до безпеки та продуктивності, мінімізувавши ризик того, що блокчейн-рівень стане «вузьким місцем» для всієї системи.

Наведено результати вирішення задачі, що підтвердили працездатність розробленого прототипу та коректність реалізованих сценаріїв взаємодії. Реалізовано керований цикл підключення пристроїв і їх адміністрування через інтерфейс налаштування, забезпечено відображення станів і подій у прикладних компонентах, а також підтримано сценарії віддаленого контролю через мобільний додаток. Продемонстровано практичну інтеграцію смарт-контрактів як інструменту автоматизації політик і фіксації доказових фактів, що підвищує керованість інформаційної безпеки й забезпечує можливість ретроспективного аналізу інцидентів. Порівняльне оцінювання за низкою метрик засвідчило закономірний компроміс: блокчейн-рівень забезпечує суттєві переваги у прозорості та аудитованості, проте може поступатися традиційним методам за затримками та пропускною здатністю, що підтверджує обґрунтованість гібридної стратегії запису подій.

Виконано розрахунок трудомісткості та вартості розроблення, оцінено експлуатаційні витрати та визначено показники економічної ефективності впровадження. Розрахунки показали, що економічний ефект досягається за рахунок зменшення очікуваних втрат від інцидентів, скорочення часу реагування та зниження трудомісткості моніторингу й аудиту завдяки централізованій керованості та наявності доказового журналу подій. Отримані значення терміну окупності та коефіцієнтів ефективності свідчать про практичну доцільність використання запропонованого інструментарію в організаціях, які експлуатують IoT-мережі та висувають підвищені вимоги до контролю доступу й доказовості критичних операцій.

У підсумку мету роботи досягнуто: розроблено та апробовано прототип рішення інтеграції блокчейн-технологій для забезпечення безпеки IoT-мереж, обґрунтовано його архітектуру, реалізовано ключові механізми керування

доступом і аудитованості подій, а також підтверджено практичну придатність підходу експериментальними результатами і розрахунками ефективності. Практичне значення отриманих результатів полягає у можливості використання запропонованої архітектури як основи для подальшого розгортання пілотних інсталяцій, адаптації під конкретні домени (розумна будівля, промислова автоматизація, моніторинг інфраструктури) та інтеграції з наявними системами SOC/моніторингу. Перспективи подальших досліджень і розвитку рішення доцільно пов’язувати з оптимізацією продуктивності блокчейн-компонента (агрегація подій, пакетування доказів, застосування permissioned-мереж або L2-механізмів), розширенням моделі довіри (керування ключами та політиками), поглибленням механізмів виявлення аномалій, а також проведенням натурних випробувань у реальних IoT-інсталяціях із фіксацією статистики інцидентів і показників якості сервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yaga D., Mell P., Roby N., Scarfone K. Blockchain Technology Overview : NISTIR 8202. Gaithersburg, MD : National Institute of Standards and Technology, 2018. 55 p. DOI: 10.6028/NIST.IR.8202. [Електронний ресурс]. Режим доступу: https://doi.org/10.6028/NIST.IR.8202 (дата звернення: 29.11.2025).

2. Boeckl K. et al. Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks : NISTIR 8228. Gaithersburg, MD : National Institute of Standards and Technology, 2019. DOI: 10.6028/NIST.IR.8228. [Електронний ресурс]. Режим доступу: https://doi.org/10.6028/NIST.IR.8228 (дата звернення: 29.11.2025).

3. Fagan M. et al. IoT Device Cybersecurity Capability Core Baseline : NISTIR 8259A. Gaithersburg, MD : National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.IR.8259A. [Електронний ресурс]. Режим доступу: https://doi.org/10.6028/NIST.IR.8259A (дата звернення: 29.11.2025).

4. NIST. IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements : NIST SP 800-213. Gaithersburg, MD : National Institute of Standards and Technology, 2021. DOI: 10.6028/NIST.SP.800-213. [Електронний ресурс]. Режим доступу: https://doi.org/10.6028/NIST.SP.800-213 (дата звернення: 29.11.2025).

5. Voas J. et al. Networks of “Things” : NIST SP 800-183. Gaithersburg, MD : National Institute of Standards and Technology, 2016. DOI: 10.6028/NIST.SP.800-183. [Електронний ресурс]. Режим доступу: https://doi.org/10.6028/NIST.SP.800-183 (дата звернення: 29.11.2025).

6. ETSI. ETSI EN 303 645 V3.1.3 (2024-09): Cyber Security for Consumer Internet of Things: Baseline Requirements. Sophia Antipolis : ETSI, 2024. [Електронний ресурс]. Режим доступу: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/03.01.03_60/en_303645v030103p.pdf (дата звернення: 29.11.2025).

7. ENISA. ENISA Threat Landscape 2024. European Union Agency for Cybersecurity, 2024. DOI: 10.2824/0710888. [Електронний ресурс]. Режим доступу: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024 (дата звернення: 29.11.2025).

8. OASIS. MQTT Version 5.0 : OASIS Standard. 2019. [Електронний ресурс]. Режим доступу: https://mqtt.org/mqtt-specification/ (дата звернення: 29.11.2025).

9. Shelby Z., Hartke K., Bormann C. The Constrained Application Protocol (CoAP) : RFC 7252. IETF, 2014. [Електронний ресурс]. Режим доступу: https://www.rfc-editor.org/rfc/rfc7252 (дата звернення: 29.11.2025).

10. Bormann C. Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things : RFC 7925. IETF, 2016. [Електронний ресурс]. Режим доступу: https://www.rfc-editor.org/rfc/rfc7925 (дата звернення: 29.11.2025).

11. Selander G. et al. Object Security for Constrained RESTful Environments (OSCORE) : RFC 8613. IETF, 2019. [Електронний ресурс]. Режим доступу: https://www.rfc-editor.org/rfc/rfc8613 (дата звернення: 29.11.2025).

12. Moran B. et al. A Firmware Update Architecture for Internet of Things Devices : RFC 9019 (SUIT). IETF, 2021. [Електронний ресурс]. Режим доступу: https://www.rfc-editor.org/rfc/rfc9019 (дата звернення: 29.11.2025).

13. Amazon Web Services. Industrial IoT Architecture Patterns : AWS Whitepaper, 2021. [Електронний ресурс]. Режим доступу: https://docs.aws.amazon.com/pdfs/whitepapers/latest/industrial-iot-architecture-patterns/industrial-iot-architecture-patterns.pdf (дата звернення: 29.11.2025).

14. Cloud Standards Customer Council. Cloud Customer Architecture for IoT. 2016. [Електронний ресурс]. Режим доступу: https://www.omg.org/cloud/deliverables/CSCC-Cloud-Customer-Architecture-for-IoT.pdf (дата звернення: 29.11.2025).

15. CISA. Heightened DDoS Threat Posed by Mirai and Other Botnets (Alert). 2016. [Електронний ресурс]. Режим доступу: https://www.cisa.gov/news-events/alerts/2016/10/14/heightened-ddos-threat-posed-mirai-and-other-botnets (дата звернення: 29.11.2025).

16. OWASP Foundation. OWASP Internet of Things Project. [Електронний ресурс]. Режим доступу: https://owasp.org/www-project-internet-of-things/ (дата звернення: 29.11.2025).

17. OWASP Foundation. IoT Top 10 (2018) Infographic. [Електронний ресурс]. Режим доступу: https://owasp.org/www-project-internet-of-things/assets/images/OWASP-IoT-Top-10-2018-final.jpg (дата звернення: 29.11.2025).

18. Hyperledger Fabric. Membership Service Providers (MSP) (documentation). [Електронний ресурс]. Режим доступу: https://hyperledger-fabric.readthedocs.io/en/latest/msp.html (дата звернення: 29.11.2025).

19. Hyperledger Fabric. The Ordering Service (documentation). [Електронний ресурс]. Режим доступу: https://hyperledger-fabric.readthedocs.io/en/release-2.5/orderer/ordering_service.html (дата звернення: 29.11.2025).

20. Hyperledger Fabric. Private Data (documentation). [Електронний ресурс]. Режим доступу: https://hyperledger-fabric.readthedocs.io/en/latest/private-data-arch.html (дата звернення: 29.11.2025).

21. Hyperledger Fabric CA. Fabric CA User's Guide (documentation, PDF). [Електронний ресурс]. Режим доступу: https://media.readthedocs.org/pdf/fabric-ca/latest/fabric-ca.pdf (дата звернення: 29.11.2025).

22. Sun J., Yan J., Zhang K. Z. K. Blockchain-Based Sharing Services: What Blockchain Technology Can Contribute to Smart Cities. *IEEE Access*. 2016. Vol. 4. P. 142–154. DOI: 10.1109/ACCESS.2016.2529720. [Електронний ресурс]. Режим доступу: https://ieeexplore.ieee.org/document/7467406 (дата звернення: 29.11.2025).

23. W3C. Decentralized Identifiers (DIDs) v1.0 : W3C Recommendation. 2022. [Електронний ресурс]. Режим доступу: https://www.w3.org/TR/did-core/ (дата звернення: 29.11.2025).

24. NIST CSRC Glossary. Minimally Securable IoT Device. [Електронний ресурс]. Режим доступу: https://csrc.nist.gov/glossary/term/minimally_securable_iot_device (дата звернення: 29.11.2025).

25. W3C. Decentralized Identifiers (DIDs) v1.0 : W3C Working Draft (for architecture diagram). 2020. [Електронний ресурс]. Режим доступу: https://www.w3.org/TR/2020/WD-did-core-20201108/ (дата звернення: 29.11.2025).

26. Docker Docs. Compose file reference (Compose Specification) [Електронний ресурс]. – Режим доступу: https://docs.docker.com/reference/compose-file/ (дата звернення: 30.11.2025).

27. Eclipse Mosquitto. An open source MQTT broker [Електронний ресурс]. – Режим доступу: https://mosquitto.org/ (дата звернення: 30.11.2025).

28. Node-RED. Documentation [Електронний ресурс]. – Режим доступу: https://nodered.org/docs/ (дата звернення: 30.11.2025)

29. Hyperledger Explorer documentation. Introduction [Електронний ресурс]. – Режим доступу: https://blockchain-explorer.readthedocs.io/en/main/introduction.html (дата звернення: 30.11.2025).

30. Prometheus. Alerting overview [Електронний ресурс]. – Режим доступу: https://prometheus.io/docs/alerting/latest/overview/ (дата звернення: 30.11.2025).

31. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Львів: Тріада плюс, 2011. 224 с.

32. Охорона праці: практикум /І.П. Пістун, Ю.В. Кіт, А.П.Березовецький – Суми: Університетська книга, 2000. –205с.

Додаток
Програмний код створення смарт-контакту

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract IoTDeviceManager {
// Структура для представлення пристрою IoT
struct IoTDevice {
string name; // Назва пристрою
string deviceType; // Тип пристрою (наприклад, датчик температури)
address owner; // Власник пристрою
bool isActive; // Статус пристрою (активний або неактивний)
string latestData; // Останні отримані дані від пристрою
}
// Словник для зберігання пристроїв (адреса -> дані пристрою)
mapping(address => IoTDevice) public devices;
// Подія для логування нових пристроїв
event DeviceAdded(address indexed deviceAddress, string name, string deviceType);
// Подія для оновлення даних пристрою
event DataUpdated(address indexed deviceAddress, string data);
// Подія для зміни статусу пристрою
event StatusChanged(address indexed deviceAddress, bool isActive);
```

84

```
// Модифікатор для перевірки прав доступу
modifier onlyOwner(address deviceAddress) {
require(
devices[deviceAddress].owner == msg.sender,
"Ви не є власником цього пристрою"
}
// Додавання нового пристрою
function addDevice(address deviceAddress, string memory name, string memory
deviceType) public {
require(devices[deviceAddress].owner == address(0), "Пристрій вже зареєстровано");
devices[deviceAddress] = IoTDevice({
name: name,
deviceType: deviceType,
owner: msg.sender,
isActive: true,
latestData: ""
});
emit DeviceAdded(deviceAddress, name, deviceType);
}
// Оновлення статусу пристрою
function setDeviceStatus(address deviceAddress, bool isActive) public
onlyOwner(deviceAddress) {
devices[deviceAddress].isActive = isActive;
emit StatusChanged(deviceAddress, isActive);
```

85

```
}
// Надсилання даних від пристрою
function sendData(address deviceAddress, string memory data) public
onlyOwner(deviceAddress) {
```

```
require(devices[deviceAddress].isActive, "Пристрій не активний");
devices[deviceAddress].latestData = data;
emit DataUpdated(deviceAddress, data);
}
// Отримання даних про пристрій
function getDevice(address deviceAddress) public view returns (IoTDevice memory) {
return devices[deviceAddress];
}
}
```