МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КІБЕРБЕЗПЕКИ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
"___" _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ СТУПЕНЯ «МАГІСТР»

Тема: Метод захисту даних в *IoT*-системах

Виконавець: Ярослав ТИМЧУК

Керівник: к.т.н., доцент Андрій ПЕТРЕНКО

Нормоконтролер: к.т.н., доцент Андрій ПЕТРЕНКО

Київ 2024

ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет комп'ютерних наук та технологій
Кафедра кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітньо професійна програма «Безпека інформаційних і комунікаційних систем»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
"___" _____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Тимчука Ярослава Сергійовича

1.Тема кваліфікаційної роботи: Метод захисту даних в *IoT*-системах
затверджена наказом ректора від « 30 » серпня 2024 року № 1695 /ст.

2. Термін виконання роботи: з 30.08.2024 по 15.12.2024

3. Вихідні дані до роботи: захист даних в мережі *IoT* з використанням технології блокчейн.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) огляд існуючих методів захисту даних в *IoT*-системах;

2) блокчейн технології як метод захисту даних в *IoT*;

3) експериментальне дослідження ефективності розробленого методу захисту даних в *IoT*-системах.

5. Перелік обов'язкового графічного матеріалу: презентація, схеми алгоритмів роботи модулів системи, робочі вікна мобільного додатку, вікна налаштування блокчейн модуля

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Примітка
1	Провести аналіз літератури за темою кваліфікаційної роботи	30.08.24 - 03.09.24	*Виконано*
2	Провести огляд методів захисту даних в *IoT*-системах	04.09.24- 09.09.24	*Виконано*
3	Провести порівняльний аналіз використання технологій блокчейн в *IoT*	10.09.24- 11.09.24	*Виконано*
4	Визначити структуру апаратно-програмної системи обміну даними в мережі *IoT*	12.09.24- 16.09.24	*Виконано*
5	Розробити програмні компоненти системи обміну даними в мережі *IoT* з використанням технології *IoT*	17.09.24- 21.10.24	*Виконано*
6	Провести тестування розробленої системи	22.10.24- 24.10.24	*Виконано*
7	Написати пояснювальну записку	25.10.24- 18.11.24	*Виконано*
8	Пройти нормконтроль, оформити супроводжувальні документи та підготувати презентацію для виступу	19.11.24- 25.11.24	*Виконано*

7. Дата видачі завдання « 30» серпня 2024 р.

Керівник кваліфікаційної роботи_____ Андрій ПЕТРЕНКО
(підпис керівника)

Завдання прийняв до виконання_____ Ярослав ТИМЧУК
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Метод захисту даних в *IoT*-системах» складається з: 82 с., 20 рис., 2 таблиці, 29 літературних джерел, 1 додаток.

Мета роботи – розробка і реалізація методу захисту даних в *IoT*-системах на основі блокчейн-технології, що забезпечує підвищення рівня безпеки, прозорості і стійкості до атак.

Об'єкт дослідження – процес захищеної передачі даних в *IoT*-системах з використанням технології блокчейн.

Предмет дослідження – вдосконалення методу захисту даних в *IoT*-системах з використанням технології блокчейн.

Наукова новизна: вдосконалено метод, який поєднує децентралізоване управління даними, автоматизовану аутентифікацію пристроїв і смарт-контракти для управління доступом, що забезпечує високий рівень безпеки в розподілених *IoT*-системах.

Методи дослідження: аналіз загроз на *IoT*-системи та методів захисту від цих загроз, кількісна оцінка розробленого програмного рішення за ключовими метриками (стійкість до атак, продуктивність, масштабованість, витрати)

Практична цінність: розроблено і протестовано прототип *IoT*-системи, що використовує блокчейн для забезпечення захисту даних, із впровадженням смарт-контрактів для управління пристроями та обробки транзакцій.

Результати кваліфікаційної роботи рекомендується використовувати при розгортанні *IoT* систем з підвищеним ступенем захисту передачі даних.

ІНТЕРНЕТ РЕЧЕЙ, БЛОКЧЕЙН, ЗАХИСТ ДАНИХ, СМАРТ-КОНТРАКТИ, КІБЕРБЕЗПЕКА, АУТЕНТИФІКАЦІЯ ПРИСТРОЇВ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API – Application Programming Interface

DAG – Directed Acyclic Graph

DLP – Data Loss Prevention

ERC – Ethereum Request for Comments

EVM – Ethereum Virtual Machine

HYPR – Hyper-Personalized Recognition

IoT – Internet of Things

IoTA – Internet of Things Application

M2M – Machine-to-Machine

NFC – Near-Field Communication

P2P – Peer-to-Peer

PKC – Public Key Cryptography

PKI – Public Key Infrastructure

TLS – Transport Layer Security

ВСТУП

Інтернет речей (*IoT*) – це одна з найперспективніших і найшвидше зростаючих технологій, яка має потенціал суттєво змінити різні галузі, включаючи промисловість, охорону здоров'я, транспорт, сільське господарство та побут. *IoT* дозволяє підключати фізичні пристрої до Інтернету для збору, обробки та обміну даними в реальному часі, що відкриває широкі можливості для автоматизації процесів і покращення якості життя. Проте, разом із зростанням кількості підключених пристроїв, з'являються нові виклики, серед яких захист даних є одним із ключових.

Актуальність дослідження полягає в тому, що застосування блокчейн-технологій для захисту даних в *IoT*-системах є новим і перспективним напрямком. Це дослідження спрямоване на розробку методу захисту даних на основі блокчейн для *IoT*, який поєднує децентралізований підхід з криптографічними засобами для забезпечення підвищеного рівня безпеки. Тема дослідження є особливо актуальною в умовах сучасних кіберзагроз і тенденцій розвитку цифрових технологій, де питання безпеки даних набувають стратегічного значення для успішної реалізації інноваційних проектів в різних галузях.

IoT-системи зазвичай характеризуються великою кількістю пристроїв з обмеженими обчислювальними ресурсами, які функціонують в децентралізованому середовищі. Це робить їх вразливими до різних типів атак, таких як несанкціонований доступ, перехоплення даних, злом пристроїв, атаки типу «людина посередині» (*Man-in-the-Middle*), а також розповсюдження шкідливого програмного забезпечення через мережу пристроїв. З огляду на це, необхідність у розробці нових методів захисту інформації в *IoT*-системах стає все більш актуальною.

Традиційні підходи до захисту даних, такі як використання брандмауерів, систем запобігання вторгнень та централізованих моделей аутентифікації, не

завжди можуть забезпечити належний рівень безпеки в *IoT*-середовищі. Це пов'язано з наступними факторами:

1. Централізованість – багато існуючих рішень базуються на централізованих серверах для управління безпекою та аутентифікацією пристроїв, що створює єдину точку відмови, вразливу до атак.

2. Обмежені ресурси пристроїв – більшість *IoT*-пристроїв мають низьку обчислювальну потужність і обмежені можливості для реалізації складних алгоритмів шифрування або обробки даних.

3. Висока складність управління безпекою – зі збільшенням кількості пристроїв, керування безпекою в масштабованих *IoT*-системах стає складнішим завданням, що ускладнює запобігання та виявлення атак.

В контексті захисту даних в *IoT*-системах, блокчейн-технологія представляє нові можливості. Блокчейн – це децентралізована технологія розподіленого реєстру, яка забезпечує незмінність записів, прозорість транзакцій і захищену передачу даних між учасниками системи. Основні переваги використання блокчейн для захисту *IoT*-систем включають:

– децентралізація – блокчейн усуває необхідність у централізованих серверах для керування безпекою, знижуючи ризик атак на центральні вузли;

– незмінність і прозорість даних – завдяки незмінності блокчейн-записів, маніпулювати даними після їх фіксації практично неможливо, що підвищує рівень довіри до переданих даних;

– аутентифікація і криптографія – технологія блокчейн дозволяє реалізувати розподілену аутентифікацію на основі криптографічних ключів, що робить можливим безпечну і надійну ідентифікацію пристроїв в *IoT*-системі.

Мета роботи – розробка і реалізація методу захисту даних в *IoT*-системах на основі блокчейн-технології, що забезпечує підвищення рівня безпеки, прозорості і стійкості до атак.

Об'єкт дослідження – процес захищеної передачі даних в *IoT*-системах з використанням технології блокчейн.

Предмет дослідження – вдосконалення методу захисту даних в *IoT*-

системах з використанням технології блокчейн.

Завдання дослідження:

– провести аналіз загроз на *IoT*-системи та методи захисту від цих загроз;

– розробити метод використання блокчейн-технології для підвищення рівнів цілісності даних та безпечності механізмів управління доступом до даних і ресурсів у *IoT*-системах;

– використати розроблений метод при управлінні пристроями в *IoT*-системі;

– провести тестування розробленого методу на реальних системах.

Наукова новизна: вдосконалено метод, який поєднує децентралізоване управління даними, автоматизовану аутентифікацію пристроїв і смарт-контракти для управління доступом, що забезпечує високий рівень безпеки в розподілених *IoT*-системах.

Результати кваліфікаційної роботи рекомендується використовувати при розгортанні *IoT* систем з підвищеним ступенем захисту передачі даних.

Апробація отриманих результатів: результати кваліфікаційної роботи представлено на 2 науково-практичних конференціях:

1. Тимчук Я.С. Метод захисту даних в іот-системах з використанням блокчейн / Я.С. Тимчук // Інтелектуальні технології лінгвістичного аналізу: матеріали міжн. наук.-техн. конф. [Київ], 23-24 жовт. 2024 р. / М-во освіти і науки України, НАУ – Київ, 2024. – С. 22.

2. Тимчук Я.С. Принципи захисту даних в *IoT*-системах / Я.С. Тимчук // Сучасні тенденції розвитку системного програмування: матеріали наук.-практ. конф. [Київ], 21-22 лист. 2024 р. / М-во освіти і науки України, НАУ – Київ, 2024. – С. 21-22.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ЗАХИСТУ ДАНИХ В *IOT*-СИСТЕМАХ

1.1. Особливості захисту даних в *IoT*

Інтернет речей (*IoT*) – це екосистема взаємопов'язаних пристроїв, які можуть взаємодіяти між собою через мережу Інтернет. Основна ідея *IoT* полягає в автоматизації обміну інформацією між фізичними об'єктами для підвищення ефективності різних процесів – від виробничих ліній до повсякденного життя [1]. Проте зростання кількості пристроїв, підключених до мережі, створює нові виклики для забезпечення захисту даних, які циркулюють між цими пристроями (рис. 1.1).

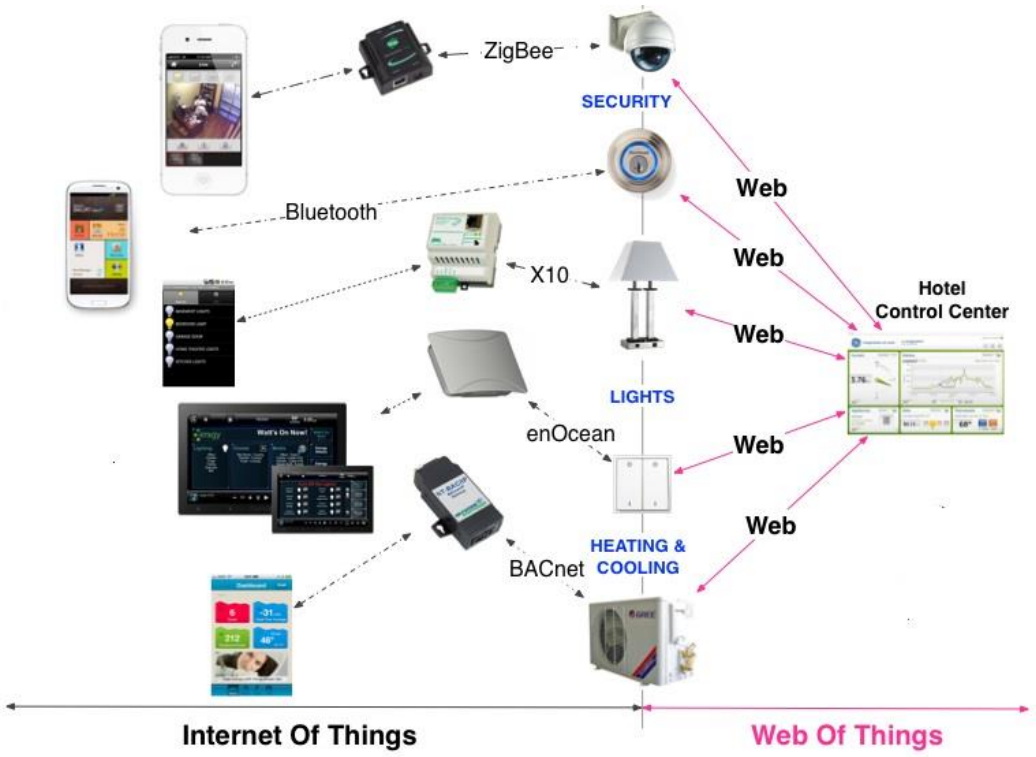


Рис. 1.1. Різнотипні компоненти *IoT*

IoT-системи мають унікальні особливості, що роблять їх захист складнішим порівняно з традиційними інформаційними системами:

1. Децентралізована архітектура. На відміну від традиційних систем, *IoT* не має єдиної централізованої структури керування, оскільки більшість рішень в системах приймаються на основі взаємодії між розподіленими пристроями. Це робить важчим контроль за захистом інформації, оскільки централізовані методи аутентифікації або шифрування не завжди можуть бути застосовані в таких умовах.

2. Обмежені ресурси пристроїв. *IoT*-пристрої, такі як датчики, камери або мікроконтролери, часто мають обмежені обчислювальні можливості, що значно обмежує здатність таких пристроїв виконувати ресурсоємні операції шифрування або використовувати складні алгоритми захисту даних.

3. Велика кількість підключених пристроїв. *IoT*-системи можуть містити тисячі або навіть мільйони пристроїв, кожен з яких взаємодіє з іншими пристроями через мережу. Така масивність збільшує ризик несанкціонованого доступу до даних через вразливості окремих пристроїв або через централізовані сервери, які керують великими обсягами даних.

4. Динамічність топології мережі. Пристрої можуть підключатися або відключатися від мережі без попередження, змінюючи топологію мережі в реальному часі. Ця властивість ускладнює застосування традиційних методів захисту, які зазвичай передбачають фіксовану топологію та набір пристроїв.

5. Гетерогенність пристроїв і протоколів. В *IoT*-системах використовуються різноманітні типи пристроїв, які можуть працювати з різними операційними системами, протоколами передачі даних і механізмами шифрування. Це створює додаткові труднощі в стандартизації і уніфікації методів захисту, оскільки різні пристрої можуть мати різні рівні безпеки.

З огляду на специфіку *IoT*, захист даних у таких системах стикається з низкою загроз, які можна поділити на кілька категорій [1]:

1. Несанкціонований доступ до пристроїв. Багато *IoT*-пристроїв мають слабкі механізми аутентифікації або не мають їх взагалі. Це відкриває можливість для зловмисників отримати контроль над пристроями і використовувати їх для збору даних або запуску атак на інші системи.

2. Перехоплення даних. В *IoT*-системах велика кількість даних передається через відкриті мережі, що робить їх вразливими до перехоплення зловмисниками. Наприклад, атаки типу «людина посередині» (*MITM*) дозволяють зловмисникам отримувати доступ до даних, що передаються між пристроями або пристроями та серверами.

3. Фізичні атаки на пристрої. *IoT*-пристрої часто розміщені у відкритих або легкодоступних місцях, що робить їх вразливими до фізичного злому або несанкціонованого втручання. Зловмисники можуть фізично втручатися в роботу пристрою, змінювати його функціонування або отримувати доступ до внутрішньої пам'яті для викрадення конфіденційних даних.

4. Атаки на конфіденційність. Оскільки *IoT*-пристрої збирають велику кількість даних, включаючи особисті або чутливі дані, ризик витоку конфіденційної інформації стає важливою загрозою. Витік таких даних може мати серйозні наслідки для конфіденційності користувачів та безпеки системи в цілому.

5. Атаки на доступність. *IoT*-системи можуть стати ціллю для атак на відмову в обслуговуванні (*DDoS*), коли зловмисники навмисно перевантажують мережу або пристрої, щоб зробити їх недоступними для нормального функціонування. Такі атаки можуть призвести до зупинки критично важливих систем, наприклад, у розумних містах або промислових автоматизованих системах.

1.1.1. Підходи до забезпечення захисту даних в *IoT*-системах

З огляду на наведені загрози та особливості *IoT*, для захисту даних у таких системах використовують різноманітні підходи, кожен з яких має свої переваги та недоліки:

1. Шифрування даних. Шифрування є одним із найпоширеніших методів захисту даних під час передачі і зберігання. Однак застосування складних криптографічних алгоритмів у *IoT*-пристроях може бути проблематичним через

обмежені обчислювальні ресурси цих пристроїв. Зазвичай використовуються легковагові алгоритми шифрування, такі як *AES*, *RSA* або *ECC* (еліптичні криві), які забезпечують прийнятний баланс між безпекою і продуктивністю.

2. Аутентифікація пристроїв. Аутентифікація дозволяє переконатися, що пристрій є тим, за кого себе видає. У *IoT*-системах це можна здійснювати за допомогою традиційних паролів, сертифікатів або асиметричних криптографічних ключів. Використання інфраструктури відкритих ключів (*PKI*) є ефективним способом забезпечення надійної аутентифікації, але знову ж таки, може вимагати значних обчислювальних ресурсів.

3. Мережеві захисні механізми. Для захисту *IoT*-систем використовуються такі традиційні мережеві механізми безпеки, як брандмауери, системи запобігання вторгненням (*IPS*) і системи виявлення вторгнень (*IDS*). Однак ці рішення, як правило, розраховані на централізовані мережі і можуть не враховувати специфіку *IoT*-систем з їх динамічною і децентралізованою природою.

4. Контроль доступу. Системи контролю доступу дозволяють визначати, які пристрої або користувачі мають право доступу до конкретних даних або ресурсів. У контексті *IoT* це може реалізовуватися за допомогою політик доступу, побудованих на основі атрибутів (*ABAC*) або ролей (*RBAC*).

5. Моделі довіри. У децентралізованих *IoT*-системах для визначення надійності пристроїв або користувачів можуть використовуватися моделі довіри. Наприклад, пристрої можуть отримувати рейтинги довіри на основі своєї попередньої поведінки або взаємодії з іншими пристроями в мережі.

1.1.2. Проблеми та виклики в забезпеченні захисту даних в *IoT*

Незважаючи на існування різних методів захисту даних в *IoT*, забезпечення надійної безпеки в таких системах залишається складним завданням через низку факторів:

1. Обмеження ресурсів пристроїв. Більшість *IoT*-пристроїв не мають достатньої обчислювальної потужності для виконання складних криптографічних операцій, що може знижувати ефективність використання сучасних методів шифрування і аутентифікації.

2. Відсутність єдиних стандартів. *IoT*-системи часто включають пристрої від різних виробників, що використовують різні протоколи передачі даних і методи захисту. Це ускладнює створення уніфікованих рішень для забезпечення безпеки.

3. Велика кількість пристроїв. Зі збільшенням кількості підключених до мережі пристроїв стає складніше забезпечити ефективний моніторинг і управління безпекою в масштабі.

4. Атаки на фізичний рівень. *IoT*-пристрої часто знаходяться в легкодоступних місцях, що робить їх вразливими до фізичного втручання або пошкодження.

5. Забезпечення безпеки в реальному часі. Оскільки *IoT*-системи працюють у режимі реального часу, забезпечення захисту даних без впливу на продуктивність системи стає важливим викликом.

Особливості *IoT*-систем і складність захисту даних у таких умовах вимагають розробки нових підходів до забезпечення безпеки. Один з перспективних напрямків – використання блокчейн-технологій, які можуть допомогти вирішити низку поточних проблем, забезпечуючи децентралізований захист даних і високий рівень безпеки при передачі інформації.

1.2. Виявлення вразливостей та розробка моделей загроз в *IoT*

IoT є потужним інструментом, який може трансформувати різні аспекти життя, включаючи промисловість, охорону здоров’я, транспорт і побут. Однак, зі зростанням кількості підключених до Інтернету пристроїв збільшуються ризики безпеки, які виникають через вразливість систем.

Захист *IoT*-систем вимагає глибокого розуміння їхніх вразливостей, а також моделювання загроз, які можуть становити небезпеку для таких систем [2]. Для цього важливо визначити, що саме потрібно захищати, і розробити систему, яка здатна ефективно забезпечувати цей захист. Розглянемо кожен етап цього процесу:

У контексті *IoT*-систем головні активи, які необхідно захищати, можуть бути класифіковані наступним чином [3]:

1. Конфіденційні дані: це можуть бути особисті дані користувачів, дані про стан здоров'я, місцезнаходження або інформація з різних пристроїв, яка може бути використана зловмисниками для незаконних дій.

2. Інфраструктура та обладнання: самі пристрої *IoT* і їх мережі є цінними активами, які можуть бути вразливі до фізичних і кібернетичних атак.

3. Операційна безпека: у деяких *IoT*-системах, наприклад, в індустріальних чи медичних системах, стабільність роботи критичних процесів є важливим аспектом захисту.

4. Доступність даних і систем: *IoT*-системи часто працюють у реальному часі, і будь-яка перерва в їх роботі може призвести до значних втрат або загроз для безпеки.

Для захисту *IoT*-систем застосовуються різні методи та інструменти [4]:

1. Шифрування даних: одним з основних інструментів захисту є шифрування даних, яке гарантує конфіденційність переданих і збережених даних.

2. Аутентифікація та авторизація: ці механізми дозволяють забезпечити, що доступ до даних і пристроїв мають лише уповноважені користувачі або інші пристрої. Це знижує ризик несанкціонованого доступу.

3. Моніторинг і виявлення аномалій: постійний моніторинг мережевого трафіку та пристроїв допомагає виявляти потенційні загрози або відхилення від нормальної роботи.

4. Системи виявлення вторгнень (*IDS*) та запобігання вторгнень (*IPS*): ці системи забезпечують активне виявлення та блокування спроб атаки на мережу або пристрої.

Виявлення вразливостей – це процес, під час якого ідентифікуються слабкі місця в системі, які можуть бути використані зловмисниками для проникнення або порушення роботи [5]. Основні вразливості в *IoT*-системах включають:

1. Відсутність або слабкі механізми аутентифікації: багато *IoT*-пристроїв мають слабкі паролі або не передбачають складної багатофакторної аутентифікації, що робить їх легкою мішенню для атак.

2. Незахищені канали зв'язку: якщо дані передаються через незахищені канали, вони можуть бути перехоплені зловмисниками. Це особливо важливо для пристроїв, що працюють через публічні мережі.

3. Обмежені обчислювальні ресурси пристроїв: *IoT*-пристрої часто мають обмежені ресурси, що не дозволяє їм використовувати складні алгоритми захисту, такі як шифрування або моніторинг у реальному часі.

4. Фізична вразливість пристроїв: *IoT*-пристрої можуть бути встановлені у відкритих або легкодоступних місцях, що робить їх вразливими до фізичного втручання або пошкодження.

Модель загроз для *IoT*-систем повинна враховувати всі можливі типи атак, що можуть бути спрямовані на систему. Основні загрози включають:

1. Атаки на конфіденційність: зловмисники можуть перехоплювати дані, що передаються між пристроями, або отримувати несанкціонований доступ до пристроїв, щоб викрасти чутливу інформацію.

2. Атаки на цілісність даних: зміна або підробка даних, що передаються через *IoT*-системи, може мати серйозні наслідки, наприклад, у медичних або промислових системах.

3. Атаки на доступність: атаки на відмову в обслуговуванні (*DoS*/*DDoS*) можуть призвести до порушення роботи системи або до повного блокування доступу до пристроїв.

4. Фізичні атаки: оскільки багато *IoT*-пристроїв розташовані у фізично вразливих місцях, вони можуть стати ціллю для фізичного пошкодження або несанкціонованого втручання.

5. Зловмисні програми: *IoT*-пристрої можуть стати частиною ботнетів або використовуватися для атак на інші системи після зараження шкідливим програмним забезпеченням.

Для того щоб ефективно захистити *IoT*-системи, необхідно розробити модель загроз [6], що враховує кожен з цих векторів атак, і передбачає відповідні методи захисту, включаючи шифрування, аутентифікацію, контроль доступу та моніторинг системи.

IoT-системи збирають величезні обсяги даних від користувачів, включаючи особисту інформацію, дані про фізичну активність, геолокацію тощо. Витік таких даних може мати серйозні наслідки для конфіденційності користувачів. Основними ризиками є:

– несанкціонований доступ до даних: якщо система не має надійних засобів контролю доступу, зловмисники можуть отримати доступ до конфіденційних даних;

– перехоплення даних: під час передачі даних через незахищені канали зв’язку можливе їх перехоплення зловмисниками, які можуть використовувати ці дані для несанкціонованих дій.

Цілісність даних є критично важливою для *IoT*-систем, особливо в таких сферах, як охорона здоров’я або транспорт, де маніпуляція даними може призвести до фатальних наслідків [7]. Основні загрози включають:

– модифікація даних: зловмисники можуть змінити передані або збережені дані, що може призвести до помилкових висновків або дій на основі підробленої інформації;

– атаки типу "людина посередині" (*Man-in-the-Middle*): під час атаки цього типу зловмисник перехоплює та модифікує дані під час їх передачі між двома легітимними сторонами.

IoT-системи використовуються для управління важливими інфраструктурами, тому їхня безперебійна робота є критичною. Загрози доступності можуть зробити систему недоступною або порушити її нормальне функціонування. Основні типи загроз доступності включають:

– атаки на відмову в обслуговуванні (*DoS*/*DDoS*): мета такої атаки – перевантажити систему, зробивши її недоступною для легітимних користувачів. *IoT*-пристрої з обмеженими ресурсами можуть стати легкою ціллю для таких атак;

– фізичні атаки на пристрої: зловмисники можуть фізично пошкодити або втрутитися у роботу пристроїв, особливо якщо вони встановлені в незахищених місцях;

– аутентифікація і авторизація є ключовими для захисту *IoT*-систем від несанкціонованого доступу. Вразливі аутентифікаційні механізми можуть призвести до того, що зловмисники отримають контроль над пристроями або отримають доступ до конфіденційних даних;

– слабкі паролі або відсутність аутентифікації: багато *IoT*-пристроїв мають слабкі або заздалегідь встановлені паролі, які легко можна зламати. Відсутність механізмів багатофакторної аутентифікації також створює ризики.

Загрози безпеки в *IoT* можна класифікувати за кількома рівнями [8], що охоплюють різні аспекти роботи систем Інтернету речей (рис. 1.2).

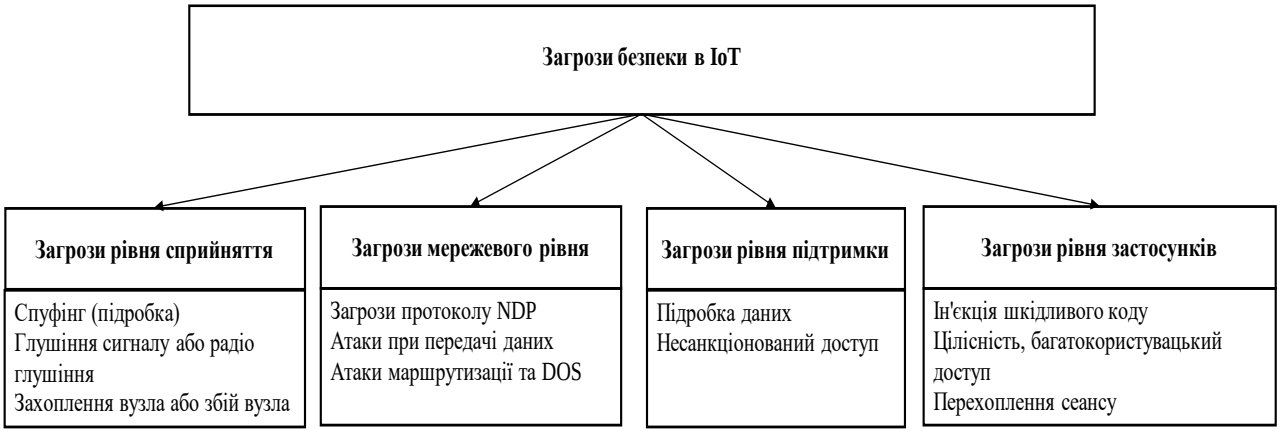


Рис. 1.2. Класифікація загроз системам *IoT*

Розглянемо детально кожен тип загроз, зображених на схемі:

1. Загрози рівня сприйняття (*Perception Layer Threats*). На цьому рівні знаходяться пристрої, такі як сенсори та датчики, які безпосередньо взаємодіють із фізичним світом і збирають дані. Основні загрози цього рівня включають:

– спуфінг (підробка): зловмисник може створити фальшивий пристрій або сигнал, що імітує справжній пристрій *IoT*, для того щоб обдурити систему. Це може призвести до збору хибної інформації або навіть до саботажу;

– глушіння сигналу або радіо глушіння: використання перешкод для блокування або ослаблення бездротових сигналів між *IoT*-пристроями. Це може вплинути на збирання даних або передачу критично важливої інформації;

– захоплення вузла або збій вузла: зловмисник може захопити контроль над пристроєм, що призведе до втрати контролю над окремим компонентом *IoT*-мережі. Це може бути небезпечно, особливо у великих системах, таких як смарт-міста або промислові системи.

2. Загрози мережевого рівня (*Network Layer Threats*). Мережевий рівень відповідає за передачу даних між пристроями *IoT* і серверами, що управляють цими системами. Основні загрози на цьому рівні включають [9]:

– загрози протоколу *NDP* (*Neighbor Discovery Protocol*): *NDP* використовується для автоматичного виявлення пристроїв у мережі *IPv*6. Атаки на цей протокол можуть дозволити зловмиснику перенаправляти трафік або підробляти інформацію про сусідні пристрої;

– атаки при передачі даних: дані, що передаються між пристроями, можуть бути перехоплені, модифіковані або видалені зловмисниками. Це може порушити конфіденційність або цілісність переданих даних;

– атаки маршрутизації та *DOS* (*Denial of Service*): атаки на маршрутизацію можуть змінити маршрут передачі даних або заблокувати передачу між вузлами. *DoS*-атаки спрямовані на те, щоб перевантажити мережу або сервери, що обробляють *IoT*-трафік, призводячи до зупинки нормальної роботи.

3. Загрози рівня підтримки (*Support Layer Threats*). Рівень підтримки включає платформи та системи, що забезпечують зберігання, обробку і управління даними з *IoT*-пристроїв. Основні загрози:

– підробка даних: зловмисники можуть модифікувати або підробляти дані, що надходять з пристроїв. Це може призвести до неправильних рішень, які будуть прийняті на основі хибної інформації;

– несанкціонований доступ: злом облікових записів або використання вразливостей в механізмах доступу може дозволити зловмисникам отримати доступ до конфіденційних даних або навіть взяти під контроль систему.

4. Загрози рівня застосунків (*Application Layer Threats*). Цей рівень включає в себе програмне забезпечення та сервіси, які обробляють дані з *IoT*-пристроїв і надають користувачам доступ до системи. Загрози цього рівня включають:

– ін'єкція шкідливого коду: зловмисники можуть вбудовувати шкідливий код у програми *IoT*, щоб отримати доступ до даних або повністю контролювати пристрої. Це може включати атаки через *SQL*-ін'єкції або шкідливі скрипти;

– цілісність, багатокористувацький доступ: багатокористувацькі *IoT*-системи можуть зіткнутися з проблемами збереження цілісності даних і управління правами доступу між користувачами. Наприклад, один користувач може отримати доступ до даних, які призначені для іншого користувача;

– перехоплення сеансу: зловмисник може захопити сеанс користувача і отримати доступ до його даних або послуг. Це може статися через атаки типу «людина посередині» (*Man-in-the-Middle*), коли трафік між користувачем і сервером перехоплюється.

Загрози безпеці в *IoT* варіюються від фізичних атак на пристрої до кіберзагроз, що можуть порушити конфіденційність, цілісність і доступність даних. Для ефективного захисту *IoT*-систем необхідно враховувати кожен з рівнів загроз і впроваджувати відповідні заходи безпеки, такі як шифрування, аутентифікація, контроль доступу та моніторинг мережі.

Моделі безпеки *IoT* охоплюють різноманітні методи і технології для забезпечення конфіденційності, цілісності та доступності даних у середовищі з великою кількістю підключених пристроїв. Оскільки *IoT* охоплює широкий спектр застосувань, від промислових систем до побутових пристроїв, його моделі безпеки повинні відповідати численним вимогам, що варіюються залежно від контексту використання. Розглянемо детальніше деякі еталонні моделі безпеки *IoT*, запропоновані основними стандартами і компаніями.

1. Модель безпеки від *ENISA* (*European Union Agency for Cybersecurity*). *ENISA* розробила *IoT Security Reference Architecture*, яка базується на ідентифікації потенційних загроз і вразливостей в *IoT*-системах. Основними компонентами цієї моделі є:

– безпека на рівні пристроїв: захист фізичних пристроїв (датчиків, контролерів) через шифрування даних, надійні методи аутентифікації та захист від фізичних атак;

– мережева безпека: захист каналів зв'язку, що передають дані між пристроями, використовуючи шифрування та захищені протоколи передачі;

– управління ключами: забезпечення надійного процесу генерації, зберігання та використання криптографічних ключів;

– політики контролю доступу: управління правами доступу до ресурсів і пристроїв на основі ролей та атрибутів користувачів.

ENISA також приділяє особливу увагу питанням оновлень програмного забезпечення та патчів для *IoT*-пристроїв, що допомагає усунути відомі вразливості та загрози.

2. Модель від *ISO*/*IEC*. Міжнародна організація стандартів (*ISO*) та Міжнародна електротехнічна комісія (*IEC*) розробили стандарти для *IoT*, зокрема *ISO*/*IEC* 27001 для управління інформаційною безпекою та *ISO*/*IEC* 29182 для забезпечення безпеки в сенсорних мережах. Основними компонентами моделі *ISO*/*IEC* для *IoT* є:

– захист конфіденційності даних: використання алгоритмів шифрування та технологій приховування інформації для забезпечення конфіденційності персональних даних;

– цілісність і контроль доступу: важливими аспектами є захист даних від несанкціонованої модифікації та забезпечення аутентифікації користувачів;

– оцінка ризиків: модель безпеки від *ISO*/*IEC* передбачає постійну оцінку ризиків та їхнє зниження через впровадження відповідних засобів захисту на всіх рівнях *IoT*-архітектури.

3. Модель *ITU-T* (*International Telecommunication Union*). *ITU-T* пропонує серію стандартів *X*.805 для оцінки ризиків і реалізації заходів безпеки в телекомунікаційних системах, включаючи *IoT*. Основні компоненти цієї моделі:

– захист на рівні управління мережею: використання політик безпеки для управління мережею і виявлення загроз;

– захист фізичної інфраструктури: модель передбачає заходи фізичної безпеки для захисту важливих елементів мережі;

– мережева безпека: включає захист від *DDoS*-атак, моніторинг трафіку і виявлення несанкціонованого втручання.

Модель *ITU-T* забезпечує повну безпеку *IoT*-систем, охоплюючи всі етапи комунікації між пристроями, від збору даних до їх передачі і зберігання на серверах.

4. Моделі від провідних технологічних компаній (*Cisco, Intel, IBM*). Компанії, такі як *Cisco*, *Intel* і *IBM*, пропонують власні еталонні моделі безпеки *IoT*. Наприклад, *Cisco* розробила модель "*Internet of Everything*" (*IoE*), яка орієнтована на інтеграцію людей, даних, процесів і пристроїв в єдину захищену мережу. Основні принципи *Cisco* включають:

– конфіденційність та шифрування даних: Захист даних на всіх етапах їх життєвого циклу через шифрування і контроль доступу;

– політики безпеки по всьому ланцюгу: Безпека впроваджується на всіх етапах: від датчиків до хмарних платформ;

– безперервний моніторинг та виявлення загроз: *Cisco* пропонує технології для моніторингу мережі в реальному часі з використанням машинного навчання для виявлення аномалій і загроз.

Intel і *IBM* також пропонують подібні моделі, що базуються на принципах надійного шифрування, управління ключами та автоматизованого моніторингу пристроїв для забезпечення їхньої безпеки в *IoT*-мережах.

Незалежно від конкретної моделі або стандарту, існує кілька основних аспектів, які включають всі моделі безпеки для *IoT*:

1. Захист конфіденційності даних: використання шифрування даних на рівні пристроїв і під час їхньої передачі через мережу.

2. Аутентифікація та авторизація: забезпечення, що доступ до пристроїв і даних мають тільки авторизовані користувачі або системи.

3. Моніторинг та управління загрозами: виявлення та аналіз підозрілої активності в мережі через засоби моніторингу та автоматизованого виявлення аномалій.

4. Управління ризиками: оцінка та управління ризиками на кожному етапі розробки і впровадження *IoT*-систем, що дозволяє вчасно реагувати на нові загрози і вразливості.

5. Оновлення та підтримка систем: автоматичні оновлення програмного забезпечення пристроїв *IoT* для виправлення вразливостей і покращення безпеки.

Моделі безпеки *IoT* представляють комплексний підхід до захисту даних, пристроїв і мереж у контексті сучасних загроз кібербезпеці. Використання стандартів *ENISA, ISO/IEC, ITU-T* і моделей від провідних компаній дозволяє забезпечити надійну безпеку для різноманітних *IoT*-систем, зокрема через шифрування, контроль доступу, моніторинг та автоматизоване виявлення загроз. Кожна модель відповідає конкретним потребам *IoT*-мереж, допомагаючи забезпечити безперебійну роботу систем і захист від зловмисників.

1.3. Модель порушника в *IoT*-системах

Згідно з НД ТЗІ 1.1-002-99 "Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу", основними поняттями щодо моделі загроз і моделі порушника є наступні [5].

Модель загроз – це концептуальна схема, яка описує потенційні загрози, з якими може зіткнутися інформаційна система. Вона включає можливі джерела загроз, методи атак, вразливі компоненти системи та можливі наслідки. Основними складовими моделі загроз є:

– джерела загроз: внутрішні та зовнішні суб’єкти, які можуть спричинити загрозу. До внутрішніх джерел належать співробітники або інші користувачі системи, а до зовнішніх – зловмисники, хакери, конкуренти;

– методи атак: засоби, які можуть бути використані для несанкціонованого доступу до даних або порушення роботи системи (наприклад, злам паролів, використання шкідливого програмного забезпечення);

– вразливості: слабкі місця в системі, які можуть бути використані для реалізації загроз;

– наслідки: потенційні втрати або пошкодження внаслідок реалізації загрози (втрата даних, компрометація конфіденційної інформації, порушення функціонування системи).

Модель порушника визначає профіль суб'єкта, який може спробувати реалізувати загрозу. Це може бути:

– зовнішній порушник: людина або організація, яка не має авторизованого доступу до системи, але намагається зламати її для отримання доступу до інформації;

– внутрішній порушник: особа, яка має авторизований доступ до системи, але використовує цей доступ для зловживань (наприклад, крадіжка даних);

– можливості порушника: модель порушника описує технічні та фінансові можливості, рівень підготовки, знання та ресурси, доступні для проведення атаки;

– мотиви порушника: також можуть бути враховані, включаючи фінансову вигоду, саботаж або інші види мотивації.

Модель порушника є важливим компонентом систем захисту інформації, який дозволяє визначити, хто і як може атакувати комп'ютерну систему. Модель порушника враховує характеристики потенційних зловмисників, їх можливості, ресурси і цілі. Це допомагає побудувати захист, що буде здатен протистояти реальним загрозам.

Основні елементи моделі порушника

1. Характеристики порушника:

– мотивація: фінансова вигода, політичні причини, саботаж, шпигунство тощо;

– знання і досвід: рівень технічної підготовки порушника, доступ до спеціалізованих знань або інструментів;

– ресурси: матеріальні та технічні можливості для здійснення атаки, включаючи фінанси, обладнання та доступ до технологій;

– доступність часу: скільки часу порушник може інвестувати в атаку, що впливає на складність методів, які він може використовувати.

2. Можливості порушника визначаються його технічними навичками і ресурсами. Це можна математично описати через формули, що відображають доступні технічні та обчислювальні ресурси для реалізації атак.

Математично модель порушника можна представити через описання його дій, можливостей та мотивів.

Так модель порушника може використовувати функцію атаки $A(t,r,e,m)$, яка залежить від наступних параметрів:

– t – час, протягом якого порушник може працювати над атакою;

– r – ресурси (обчислювальні потужності, фінанси);

– e – рівень технічних знань і досвіду порушника;

– m – мотиви порушника (шкода, фінансова вигода, тощо).

Функція атаки може бути представлена як:

$$A(t,r,e,m) = \alpha \cdot t + \beta \cdot r + \gamma \cdot e + \delta \cdot m,$$

де α, β, γ, δ – коефіцієнти, що відображають вагу кожного з факторів.

Наприклад, чим більше досвіду у порушника (велике значення e), тим більше його можливості успішно завершити атаку.

Імовірність успішної атаки можна оцінити через імовірність подолання захисту системи. Якщо Z – рівень захисту системи, то ймовірність успіху атаки $P_{\text{успіх}}$ можна виразити через співвідношення між можливостями порушника та рівнем захисту:

$$P_{\text{успіх}} = A(t,r,e,m) / Z.$$

де Z – загальна ефективність захисту системи, що може включати складність шифрування, рівень аутентифікації, контроль доступу тощо. Чим вищий рівень захисту Z, тим нижча ймовірність успіху атаки.

Також для оцінки ризику атаки можна використовувати модель витрат. Витрати на атаку можуть бути представлені як:

$$C_{\text{атака}} = c_t \cdot t + c_r \cdot r + c_e \cdot e,$$

де c_t, c_r, c_e – це вартість часу, ресурсів і технічних знань відповідно.

З іншого боку, система має захисні витрати $C_{\text{захист}}$, які включають витрати на впровадження заходів безпеки, шифрування та моніторинг. Ризик атаки можна оцінити через співвідношення витрат порушника до витрат на захист:

$$R = C_{\text{атака}} / C_{\text{захист}}.$$

Якщо $R>1$, то ризик атаки високий, оскільки витрати на атаку можуть перевищити витрати на захист. Якщо $R<1$, то ризик менший, оскільки захист системи перевищує потенційні витрати на атаку.

З урахуванням даних функцій можна визначити наступні категорії порушників:

1. Скрипт-кідді (*Script kiddie*): порушники з обмеженими технічними знаннями, які використовують наявні інструменти для атак. Вони зазвичай не

мають значних ресурсів, тому їх атаки обмежуються простими методами, такими як атаки на слабкі паролі або використання відомих вразливостей.

Формула для оцінки можливостей такого порушника може виглядати так:

$$A(t, r, e, m) = \alpha \cdot t + \beta \cdot r + 0 \cdot e + \delta \cdot m,$$

де e=0 (рівень знань), оскільки вони не мають достатніх технічних навичок.

2. Зловмисник середнього рівня: цей тип порушника має середній рівень технічних знань і може використовувати більш складні інструменти для зламу, включаючи створення шкідливого ПЗ, атаки на мережеві протоколи або злам через соціальну інженерію.

Формула для такого порушника:

$$A(t, r, e, m) = \alpha \cdot t + \beta \cdot r + \gamma \cdot e + \delta \cdot m,$$

де e – середнє значення, що відображає компетентність у різних техніках атак.

3. Державний або корпоративний хакер (*Advanced Persistent Threat – APT*): найбільш серйозні загрози походять від організацій, які мають значні ресурси та високий рівень технічних знань. Вони можуть інвестувати час і кошти в атаки з високою складністю, такі як атаки на критичну інфраструктуру або шпигунські атаки.

Формула для таких порушників:

$$A(t,r,e,m)=\alpha \cdot t + \beta \cdot r + \gamma \cdot e + \delta \cdot m,$$

де r і e є високими через наявність значних ресурсів і знань.

Модель порушника є невід'ємною частиною оцінки безпеки комп'ютерних систем. Вона допомагає зрозуміти, якими ресурсами та можливостями можуть володіти потенційні зловмисники і як це вплине на ймовірність успішної атаки. Формули, що використовуються для моделювання атаки, дозволяють оцінювати ризики і розробляти ефективніші стратегії захисту системи.

Моделі загроз і порушника допомагають організувати захист системи, дозволяють оцінити вразливості й визначити необхідні заходи для їх нейтралізації, наприклад впровадження багатофакторної аутентифікації, шифрування даних, або заходів фізичного захисту.

1.4. Висновки до розділу

Огляд методів захисту даних в *IoT*-системах показав необхідність впровадження сучасних підходів до безпеки, враховуючи специфіку цих систем. *IoT*-системи характеризуються великою кількістю різнорідних пристроїв, що працюють у розподіленому середовищі з обмеженими обчислювальними ресурсами. Це вимагає використання спеціалізованих підходів до захисту, таких як шифрування даних на рівні пристроїв, контроль доступу, багатофакторна аутентифікація та виявлення аномалій у мережевому трафіку. Проте, особливістю є те, що багато з пристроїв не мають достатніх ресурсів для застосування традиційних алгоритмів захисту, що створює додаткові виклики для реалізації ефективних систем безпеки.

Важливим елементом захисту *IoT* є побудова моделей загроз і моделей порушника. Модель загроз дозволяє ідентифікувати можливі вразливості та сценарії атак на різних рівнях системи, від фізичних пристроїв до мережевого трафіку та хмарних сервісів. Це включає загрози перехоплення даних, атаки типу «людина посередині», *DDoS*-атаки, спуфінг, а також вразливості протоколів передачі даних. Модель порушника визначає характеристики потенційного зловмисника, його технічні знання, доступ до ресурсів та мотиви для атаки. Правильне розуміння моделі порушника дозволяє більш точно оцінити ризики та розробити механізми захисту, які адекватно відповідають можливостям зловмисника. Таким чином, для забезпечення надійної безпеки в *IoT*-системах необхідно застосовувати багаторівневий підхід до захисту, що базується на ідентифікації загроз і профілюванні потенційних атакуючих.

РОЗДІЛ 2
БЛОКЧЕЙН ТЕХНОЛОГІЇ ЯК МЕТОД ЗАХИСТУ ДАНИХ В *IOT*

2.1. Принципи роботи блокчейн для захисту даних

Технологія блокчейн стала однією з найперспективніших інновацій у сфері захисту даних та безпеки інформації. Вона була спочатку розроблена як основа для криптовалюти *Bitcoin,* але її принципи швидко знайшли застосування в інших галузях, включаючи захист даних в *IoT*-системах. Головна ідея блокчейну полягає у створенні децентралізованої, розподіленої та незмінної бази даних, де кожен учасник мережі має доступ до копії реєстру, що фіксує всі транзакції чи інші дії.

Основні принципи роботи блокчейн:

1. Децентралізація блокчейн є розподіленою системою, де кожен учасник мережі (вузол) має доступ до всієї бази даних. Це означає, що система не залежить від централізованого сервера або одного адміністратора. У контексті *IoT,* це надзвичайно важливо, оскільки усувається необхідність у єдиному центральному контролері, який міг би стати ціллю для атак або зламів. Кожен пристрій в мережі може виконувати функцію автономного вузла, що дозволяє підвищити стійкість системи до збоїв і зовнішніх втручань.

2. Незмінність записів (*Immutable Ledger*) Однією з основних характеристик блокчейн є незмінність (*immutability*) даних, що зберігаються в реєстрі. Кожен блок у блокчейні містить хеш попереднього блоку, що робить будь-які зміни в історії транзакцій неможливими без зміни всього ланцюжка. Це забезпечує високий рівень цілісності даних. У *IoT*-системах це означає, що будь-які дані, передані через блокчейн, не можуть бути змінені або підроблені зловмисниками, що робить цю технологію надзвичайно корисною для збереження чутливих або критично важливих даних.

3. Прозорість та контрольованість (*Transparency and Accountability*) Всі транзакції в блокчейні є видимими для всіх учасників мережі, що забезпечує прозорість і контрольованість дій. Для *IoT*-систем це дозволяє забезпечити повний аудит всіх взаємодій між пристроями. Кожен учасник системи може бути впевненим, що дані, які передаються, є достовірними і підтвердженими іншими учасниками мережі.

4. Аутентифікація та авторизація на основі криптографії блокчейн використовує криптографічні методи для забезпечення аутентифікації учасників і перевірки транзакцій. Кожен вузол або пристрій в мережі має унікальний криптографічний ключ, який використовується для підпису транзакцій. Це дозволяє переконатися, що дані походять саме від довіреного джерела, і що вони не були змінені під час передачі. У *IoT* це дозволяє підвищити рівень безпеки за рахунок забезпечення криптографічної аутентифікації пристроїв без необхідності використання централізованих серверів.

5. Смарт-контракти (*Smart Contracts*) Смарт-контракти – це автоматизовані програми, що виконуються в блокчейні при настанні певних умов. Вони дозволяють здійснювати транзакції або інші операції без посередників, що може бути особливо корисним для *IoT*-систем. Наприклад, смарт-контракти можуть автоматизувати правила доступу до ресурсів або виконання дій між пристроями в мережі. Це підвищує ефективність і безпеку системи, оскільки всі операції виконуються лише тоді, коли виконуються заздалегідь визначені умови, а їх результат автоматично фіксується в блокчейні.

6. Розподілена консенсусна модель Для забезпечення узгодженості даних між різними учасниками мережі блокчейн використовує різні моделі консенсусу, такі як *Proof of Work* (*PoW*), *Proof of Stake* (*PoS*) або інші алгоритми. Це означає, що жоден вузол не може самостійно змінити або додати новий блок до ланцюжка без підтвердження від інших учасників мережі. Ця розподілена модель консенсусу є важливою для *IoT*, оскільки гарантує, що

будь-які зміни в системі мають бути підтверджені кількома незалежними пристроями або учасниками, що робить атаку на систему складнішою і менш ймовірною.

7. Надійність та стійкість до відмов (*Fault Tolerance*) Через те, що блокчейн є розподіленою системою, він забезпечує високу стійкість до відмов. Навіть якщо окремі вузли вийдуть з ладу або будуть скомпрометовані, система продовжить функціонувати, оскільки інші вузли підтримують роботу ланцюга. Це критично важливо для *IoT*-систем, де велика кількість пристроїв може бути розташована в різних фізичних місцях, і доступ до них може бути обмеженим.

Застосування блокчейн-технології для захисту даних в *IoT* є особливо ефективним завдяки декільком важливим аспектам:

1. Захист від несанкціонованого доступу: у традиційних централізованих системах зловмисник може атакувати центральний сервер, отримуючи доступ до всієї системи. Використання децентралізованої моделі блокчейн унеможливлює подібну атаку, оскільки немає єдиної точки відмови.

2. Цілісність даних: завдяки незмінності даних у блокчейні, будь-яка спроба змінити або підробити інформацію буде негайно помічена іншими учасниками мережі. Це важливо для *IoT*, де точність і достовірність даних мають вирішальне значення, наприклад, у системах моніторингу здоров’я або промислових процесах.

3. Аутентифікація пристроїв: кожен пристрій у *IoT*-мережі може бути автентифікований за допомогою криптографічних ключів і блокчейн-записів. Це робить систему стійкою до спроб підробити ідентичність пристрою або отримати несанкціонований доступ до мережі.

4. Можливість масштабування: традиційні методи захисту даних, такі як централізовані сервіси, можуть бути перевантажені великою кількістю підключених пристроїв. Блокчейн, завдяки своїй децентралізованій природі, легко масштабується і може підтримувати великі мережі пристроїв без втрати ефективності або безпеки.

5. Автоматизація за допомогою смарт-контрактів: смарт-контракти можуть використовуватися для автоматизації безпечного обміну даними між пристроями або виконання дій лише за наявності певних умов. Це дозволяє мінімізувати людське втручання і можливі помилки або зловживання.

6. Захист від *DDoS*-атак: у *IoT*-системах з централізованою архітектурою є ризик того, що зловмисники можуть атакувати центральний сервер або вузол, перевантажуючи його запитами і виводячи з ладу систему. У блокчейн-системах цей ризик значно зменшується, оскільки система є децентралізованою, і атакувати всі вузли мережі одночасно є надзвичайно складним завданням.

2.2. Переваги використання блокчейн для захисту даних в *IoT*

Технологія блокчейн демонструє значний потенціал у вирішенні проблем, пов'язаних із безпекою даних в Інтернеті речей (*IoT*). Враховуючи характер децентралізованих мереж *IoT* і велику кількість підключених пристроїв, блокчейн може ефективно забезпечити безпечну передачу та збереження даних. Основні переваги використання блокчейну для захисту даних в *IoT* включають:

Однією з ключових переваг блокчейн-технології є її децентралізована природа. У традиційних системах *IoT* безпека зазвичай забезпечується за допомогою централізованих серверів або хмарних рішень, що створює єдину точку відмови (*single point of failure*). Якщо центральний сервер буде атакований або виведений з ладу, вся система може стати вразливою. У блокчейні кожен учасник (вузол) мережі зберігає копію ланцюжка блоків, тому немає необхідності у централізованому управлінні. Це підвищує стійкість системи до зовнішніх атак і збоїв.

Для *IoT* це означає, що кожен пристрій в мережі може функціонувати незалежно, обмінюючись даними з іншими пристроями без необхідності в постійному звертанні до центрального вузла. Це знижує вразливість системи до

атак на один компонент і забезпечує постійну доступність даних навіть у разі відмови окремих пристроїв.

У блокчейні кожен блок містить запис транзакцій і хеш попереднього блоку, що робить зміни в ланцюжку неможливими без зміни всього ланцюжка. Ця властивість блокчейну дозволяє забезпечити високий рівень цілісності даних. Дані, записані в блокчейн, не можуть бути змінені або видалені, що гарантує їхню точність і надійність.

Для *IoT*-систем це означає, що будь-які дані, передані між пристроями або збережені в мережі, є захищеними від маніпуляцій зловмисниками. Це особливо важливо для чутливих даних, таких як медичні записи або дані про стан обладнання в промислових застосуваннях.

Блокчейн використовує криптографічні методи для забезпечення безпеки даних і аутентифікації користувачів або пристроїв. Кожен вузол або пристрій у мережі має свій унікальний криптографічний ключ, який використовується для підпису транзакцій. Це дозволяє забезпечити високий рівень довіри до даних, оскільки лише пристрій із відповідним ключем може створювати і підтверджувати транзакції.

Для *IoT* це означає, що кожен пристрій в мережі може бути надійно автентифікований, що знижує ризик несанкціонованого доступу або підробки даних. Крім того, використання криптографії підвищує рівень безпеки навіть у разі перехоплення даних під час передачі між пристроями.

Одна з основних переваг блокчейн-систем – прозорість і можливість відстеження всіх транзакцій. Кожна транзакція, яка додається до блокчейну, зберігається у незмінному вигляді та може бути переглянута будь-яким учасником мережі. Це створює надійний механізм для аудиту і відстеження дій у мережі.

В *IoT* це дозволяє відслідковувати всі взаємодії між пристроями і користувачами, забезпечуючи прозорість та можливість перевірки всіх операцій. Це є важливим аспектом для таких застосувань, як логістика або

управління ланцюгом постачання, де необхідно відстежувати кожен етап передачі або обробки даних.

Блокчейн надає можливість реалізувати гнучкі та безпечні механізми управління доступом до даних і ресурсів у *IoT*-системах. Використання смарт-контрактів дозволяє автоматизувати процеси управління доступом, що може бути корисним для великих мереж *IoT*, де важко централізовано керувати всіма пристроями.

Смарт-контракти можуть бути налаштовані таким чином, що доступ до певних ресурсів або даних надається лише за умови виконання певних умов. Це значно підвищує безпеку, оскільки автоматизовані контракти не залежать від людського втручання і працюють за заздалегідь визначеними правилами.

Традиційні централізовані системи можуть стати жертвами атак на відмову в обслуговуванні (*DDoS*), коли зловмисники перевантажують сервери великою кількістю запитів, що робить їх недоступними для користувачів. У децентралізованих системах на основі блокчейн цей ризик значно знижується, оскільки немає єдиної точки, яка б могла стати ціллю для атаки.

Кожен вузол у блокчейн-мережі незалежно обробляє транзакції, і для того щоб здійснити успішну *DDoS*-атаку, зловмиснику необхідно атакувати більшість вузлів одночасно, що є надзвичайно складним завданням. Це робить блокчейн надійним рішенням для захисту від такого типу атак у великих *IoT*-мережах.

Оскільки блокчейн є розподіленою системою, вихід з ладу окремих вузлів не впливає на роботу всієї мережі. Кожен учасник мережі має копію всього блокчейну, що дозволяє продовжувати роботу навіть у разі втрати зв’язку з окремими вузлами.

Для *IoT*-систем це є важливою перевагою, оскільки велика кількість пристроїв може бути розташована в різних фізичних місцях, і доступ до них може бути обмеженим. Блокчейн забезпечує безперебійну роботу системи, навіть якщо деякі пристрої виходять з ладу.

Блокчейн-системи легко масштабуються завдяки своїй децентралізованій природі. Кількість пристроїв, які можуть підключатися до мережі, майже не обмежена. Крім того, система може адаптуватися до нових умов без необхідності централізованого оновлення або управління.

В *IoT* це означає, що блокчейн може ефективно підтримувати розширення мережі пристроїв без втрати ефективності або безпеки. Це є ключовою перевагою в таких сферах, як розумні міста або промислові застосування, де кількість пристроїв постійно збільшується.

2.3. Огляд сучасних рішень на основі блокчейн для *IoT*

Інтеграція блокчейн-технологій з *IoT* відкриває нові горизонти для підвищення безпеки, масштабованості та ефективності роботи цих систем. Блокчейн усуває проблеми централізації, які властиві традиційним підходам, дозволяючи створювати системи без єдиної точки відмови. Сучасні рішення в цій сфері демонструють значний прогрес у різних галузях.

IoTA (*Internet of Things Application*) є одним із найбільш відомих прикладів використання блокчейн-технологій у *IoT*. Унікальність *IoTA* полягає у використанні *Directed Acyclic Graph* (*DAG*), що дає змогу передавати мікротранзакції між *IoT*-пристроями без необхідності в централізованих серверах. Це рішення ідеально підходить для розумних міст, де тисячі пристроїв можуть безпечно взаємодіяти в режимі реального часу. *DAG* також дозволяє досягти високої швидкості обробки транзакцій, уникаючи традиційних обмежень блокчейнів.

Переваги *IoTA*: висока швидкість транзакцій, низькі витрати на обробку, стійкість до атак на мережу.

Недоліки: складність масштабування при збільшенні кількості пристроїв, вразливість до квантових обчислень.

VeChain спеціалізується на управлінні ланцюжками постачання, використовуючи блокчейн для забезпечення прозорості й відстежуваності. *IoT*-

пристрої, такі як датчики і *RFID*-мітки, інтегруються з *VeChain*, що дозволяє в реальному часі відслідковувати рух товарів, запобігати підробкам і оптимізувати логістичні процеси.

Практичне використання: *VeChain* використовується в логістиці, охороні здоров'я, харчовій промисловості. Наприклад, система може контролювати умови перевезення чутливих продуктів (температура, вологість) і автоматично фіксувати дані в блокчейні. Обмеження: висока вартість впровадження для малого бізнесу.

HYPR забезпечує безпечну аутентифікацію *IoT*-пристроїв. Замість зберігання паролів у централізованих базах даних, *HYPR* використовує децентралізовані криптографічні ідентифікатори, що мінімізує ризик злому облікових даних.

Особливості: *HYPR* дозволяє інтегрувати біометричну аутентифікацію і керувати пристроями без необхідності використання паролів. Це підвищує безпеку та зручність використання.

Переваги: зменшення ризику атак на базу даних, адаптивність для великих *IoT*-мереж.

Недоліки: обмежена сумісність із деякими протоколами *IoT*.

Ethereum забезпечує платформу для впровадження смарт-контрактів, які автоматизують взаємодію між *IoT*-пристроями. Наприклад, у розумних енергомережах смарт-контракти можуть використовуватися для автоматичної оплати спожитої енергії.

Переваги: прозорість операцій, мінімізація ручного втручання, зменшення ризику помилок.

Приклад: у розумному будинку система може автоматично блокувати доступ до енергомережі, якщо користувач не виконав оплату, причому вся транзакція прозоро фіксується в блокчейні.

Обмеження: висока вартість транзакцій через перевантаженість мережі *Ethereum*.

Helium Network використовує блокчейн для мікротранзакцій між *IoT*-пристроями. Її основна мета – забезпечити доступність мережевих ресурсів для *IoT* через низькоенергетичні пристрої. *Helium* дозволяє датчикам, які працюють у розумних містах, автоматично оплачувати використання ресурсів у реальному часі.

Переваги: низька вартість роботи пристроїв, зменшення адміністративних витрат.

Обмеження: залежність від спеціалізованої інфраструктури для підтримки мережі.

Результати зведеного аналізу наведено в табл. 2.1.

Таблиця 1.2

Порівняння методів захисту даних в *IoT* з використанням блокчейн

Рішення	Основна функція	Переваги	Недоліки
IoTA	Децентралізований обмін мікротранзакціями	Швидкість, відсутність плати за транзакції	Складність інтеграції, вразливість *DAG*
VeChain	Управління ланцюжками постачання	Прозорість, боротьба з підробками	Висока вартість впровадження
HYPR	Децентралізована аутентифікація	Зменшення атак на базу даних	Сумісність з іншими протоколами
Ethereum	Смарт-контракти для автоматизації операцій	Прозорість, автоматизація	Висока вартість транзакцій
Helium Network	Мікротранзакції між *IoT*-пристроями	Доступність ресурсів, економічна ефективність	Потреба в спеціальній інфраструктурі

Блокчейн-технології для *IoT* демонструють значний потенціал у забезпеченні безпеки, прозорості та ефективності. Кожне з рішень має свої унікальні особливості та обмеження, що робить їх застосування залежним від конкретного сценарію. Вибір платформи залежить від вимог системи, таких як швидкість обробки транзакцій, рівень захисту або вартість впровадження. У перспективі очікується подальше вдосконалення цих рішень із впровадженням нових протоколів і механізмів захисту.

2.4. Проєктування архітектури програмного рішення для *IoT* на основі блокчейн

Проєктування архітектури програмного рішення для *IoT* на основі блокчейн-технологій є важливим етапом розробки, який передбачає інтеграцію *IoT*-пристроїв у розподілену систему з високим рівнем захисту даних. Основна мета цього розділу – розробити архітектуру, яка поєднує переваги *IoT* та блокчейну для забезпечення конфіденційності, цілісності, доступності та масштабованості.

Розробка архітектури базується на наступних принципах:

– децентралізація: використання блокчейн-технологій для усунення єдиної точки відмови і забезпечення прозорості даних.

– модульність: чітке розділення компонентів системи на окремі модулі (*IoT*-пристрої, сервер, блокчейн-мережа, мобільний додаток).

– безпека: впровадження криптографічних методів, таких як шифрування даних і використання смарт-контрактів.

– масштабованість: забезпечення можливості інтеграції великої кількості *IoT*-пристроїв без втрати продуктивності.

– простота взаємодії: надання користувачам доступу до системи через інтуїтивно зрозумілий мобільний додаток або веб-інтерфейс.

Програмне рішення включає такі компоненти:

– *IoT*-пристрої: сенсори та контролери, які збирають дані та виконують

команди. Вони взаємодіють із контролером через локальні мережі.

– контролер: центральний елемент, який забезпечує комунікацію між *IoT*-пристроями та сервером.

– сервер: проміжна ланка між контролером і блокчейн-мережею, яка обробляє дані, зберігає їх у базі даних та передає в блокчейн.

– блокчейн-мережа: забезпечує зберігання даних і виконання смарт-контрактів для управління доступом і аутентифікації пристроїв.

– мобільний додаток: інтерфейс для користувача, який дозволяє керувати пристроями, переглядати стан системи і налаштовувати її параметри.

На рис. 2.1 наведено діаграму випадків використання (*Use Case Diagram*), яка демонструє взаємодію користувача із функціональними можливостями *IoT* системи.

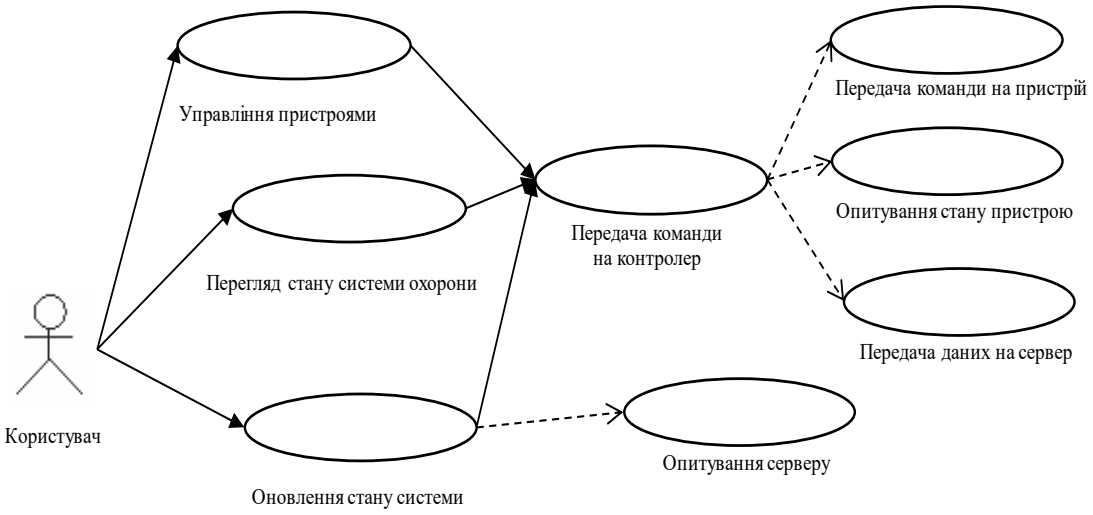


Рис. 2.1. *Use case* діаграма *IoT* системи

Основними компонентами діаграми є:

1. Актор (Користувач):

– представлений у вигляді стилізованої людини, яка символізує фізичну особу, що взаємодіє із системою;

– актор має доступ до всіх функціональних можливостей системи і може здійснювати контроль та моніторинг за її станом.

2. Випадки використання (*Use Cases*):

– керування пристроями: користувач має можливість надсилати команди на пристрої для виконання конкретних дій, наприклад, увімкнення освітлення або регулювання температури.

– перегляд стану системи: користувач може отримувати інформацію про поточний стан системи, наприклад, активність датчиків руху або інших пристроїв.

– оновлення стану системи: функція дозволяє оновити інформацію про стан всієї системи, синхронізуючи дані між пристроями.

– передача команд на контролер: цей випадок використання є ключовим для передачі команд з мобільного додатка або інтерфейсу користувача до центрального контролера системи.

– передача команди на пристрої: контролер отримує команди від користувача і надсилає їх на конкретні пристрої.

– опитування стану пристроїв: контролер запитує поточний стан пристроїв для оновлення даних.

– передача даних на сервер: дані, зібрані від пристроїв, надсилаються до сервера для обробки, збереження або подальшого аналізу.

– опитування сервера: користувач через систему може запитувати дані з сервера про стан системи, історію подій або інші параметри.

Користувач є основним ініціатором дій у системі. Через взаємодію із системою (мобільний додаток, контролер, сервер) він керує пристроями, отримує дані про стан системи або виконує налаштування.

Взаємодія між компонентами системи забезпечує автоматизацію і прозорість у роботі системи.

На рис. 2.2 показано процес взаємодії між основними компонентами системи під час опитування стану системи. Цей процес ілюструє, як користувач отримує актуальну інформацію про стан системи через мобільний додаток.

Основними елементами діаграми є:

1. Учасники процесу:

– користувач (Актор): Фізична особа, яка взаємодіє із системою через

мобільний додаток. Ініціює запит для отримання інформації про стан системи;

– мобільний додаток: Інтерфейс для взаємодії користувача з системою. Формує і надсилає запити до сервера, а також отримує відповіді для відображення користувачу;

– сервер: проміжна ланка між мобільним додатком і базою даних. Відповідає за обробку запитів, взаємодію з базою даних і передачу результатів назад до мобільного додатка;

– база даних (БД): зберігає інформацію про стан системи, пристроїв, журнал подій та інші дані. Відповідає на запити сервера.

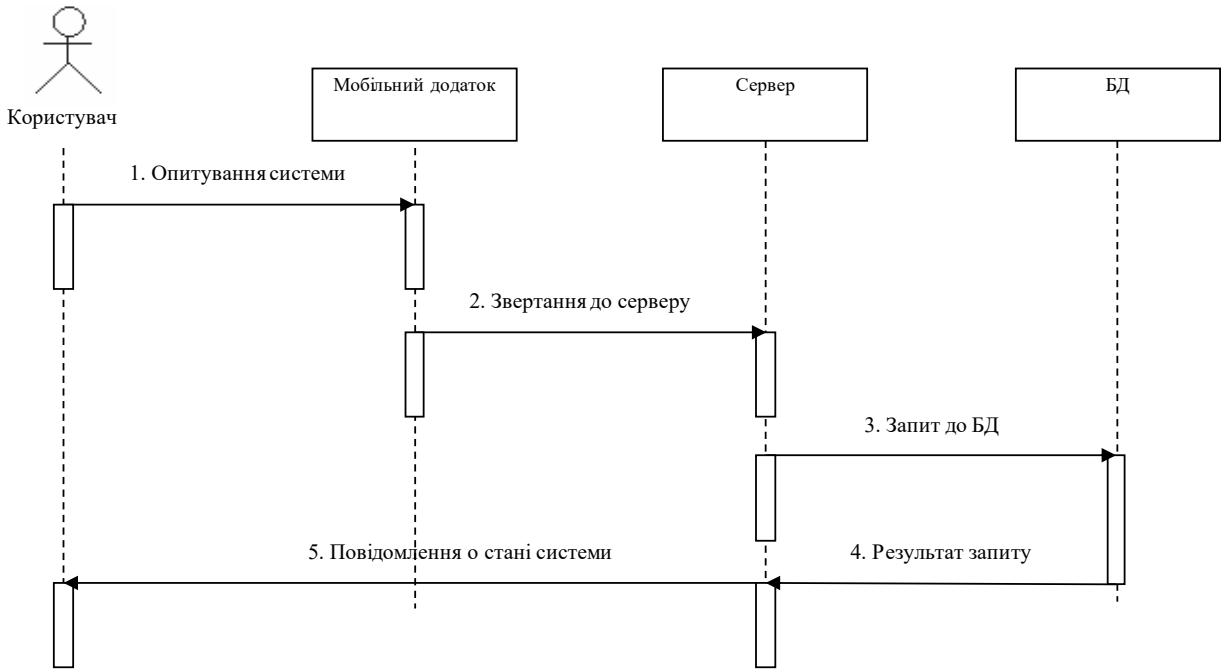


Рис. 2.2. Діаграма послідовності дій при роботі через мобільний додаток

Послідовність дій:

1. Користувач ініціює опитування системи: користувач через мобільний додаток надсилає запит для отримання інформації про стан системи, наприклад, стан датчиків або пристроїв.

2. Мобільний додаток надсилає запит до сервера: запит формується в мобільному додатку і передається серверу для подальшої обробки.

3. Сервер звертається до бази даних: сервер приймає запит від мобільного додатка і взаємодіє з базою даних, надсилаючи запит для отримання потрібної

інформації.

4. База даних відповідає серверу: база даних обробляє запит і повертає результати (наприклад, поточний стан датчиків, історію подій тощо) на сервер.

5. Сервер передає відповідь мобільному додатку: сервер формує відповідь на основі даних, отриманих від бази даних, і надсилає її мобільному додатку.

6. Мобільний додаток відображає результат користувачу: користувач отримує інформацію про стан системи, відображену у зручному форматі на екрані мобільного пристрою.

Діаграма послідовності показує взаємодію між компонентами системи в процесі опитування стану. Вона допомагає зрозуміти роль кожного учасника в обробці запиту та передачі інформації, а також служить основою для оптимізації архітектури системи.

Діаграма на рис. 2.3 демонструє послідовність взаємодій між користувачем, мобільним додатком, сервером, контролером та датчиком у системі *IoT* під час передачі команди до пристрою.

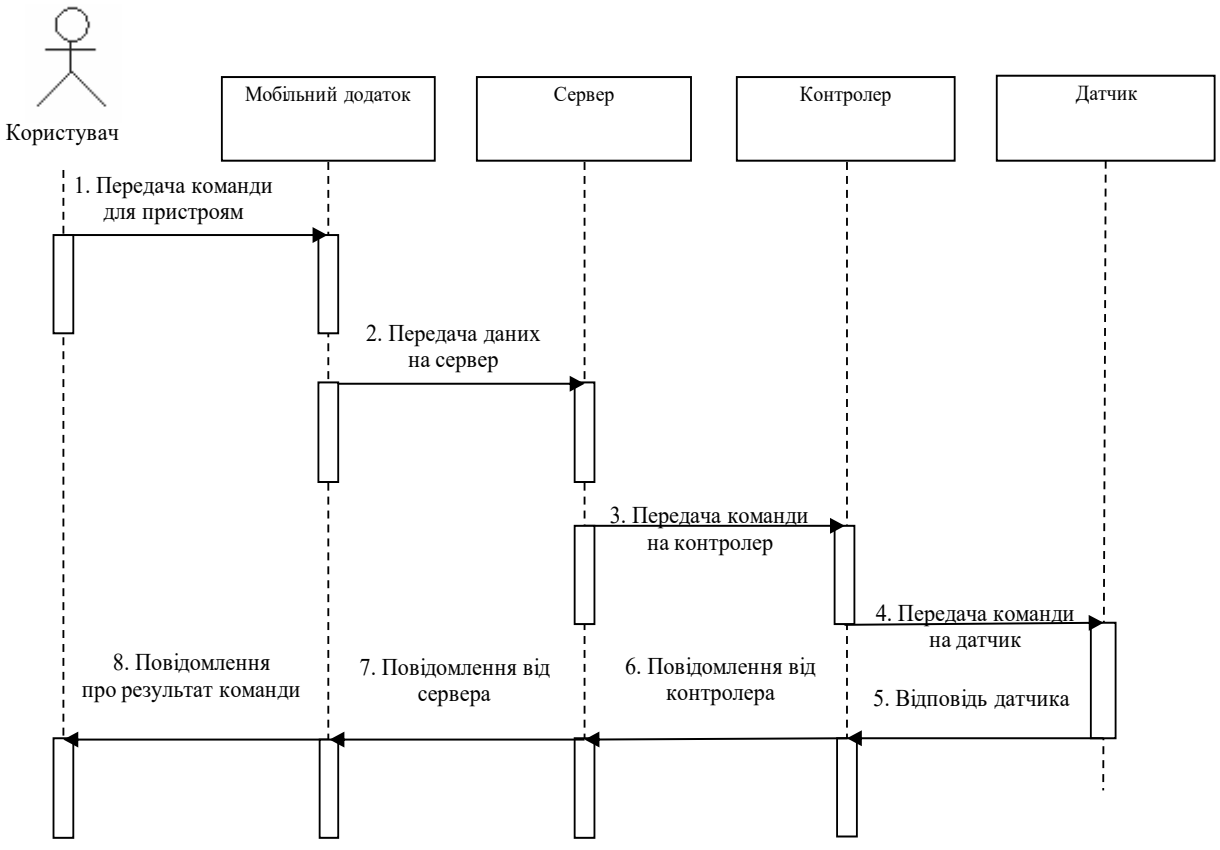


Рис. 2.3. Діаграма послідовності дій при роботі з кінцевими пристроями

Основними учасниками процесу є:

– користувач (Актор): ініціює дію через мобільний додаток, наприклад, надсилає команду для зміни стану пристрою.

– мобільний додаток: інтерфейс, через який користувач надсилає команди до системи. Передає команди на сервер і отримує зворотні повідомлення.

– сервер: відповідає за обробку отриманих команд і пересилання їх до відповідного контролера. Забезпечує передачу результатів виконання команди мобільному додатку.

– контролер: центральний компонент, що взаємодіє з пристроями. Виконує команди, отримані від сервера.

– датчик (або інший *IoT*-пристрій): пристрій, який змінює свій стан або виконує дію на основі отриманої команди.

Послідовність дій:

1. Користувач передає команду через мобільний додаток: користувач ініціює дію (наприклад, увімкнення пристрою) через інтерфейс мобільного додатка.

2. Мобільний додаток надсилає команду на сервер: додаток формує запит на основі дії користувача і передає його серверу для обробки.

3. Сервер передає команду на контролер: сервер аналізує команду, визначає відповідний пристрій і надсилає команду на контролер.

4. Контролер передає команду на датчик: контролер отримує команду від сервера і надсилає її до конкретного пристрою (датчика).

5. Датчик виконує команду: пристрій змінює свій стан або виконує дію на основі отриманої команди (наприклад, увімкнення освітлення).

6. Контролер надсилає зворотний зв'язок серверу: після виконання команди контролер передає інформацію про результат виконання назад серверу.

7. Сервер надсилає відповідь мобільному додатку: сервер отримує зворотний зв'язок від контролера, формує відповідь і надсилає її мобільному додатку.

8. Мобільний додаток інформує користувача: користувач отримує повідомлення про результат виконання команди (наприклад, підтвердження про успішне виконання дії).

Діаграма послідовності відображає, як команда від користувача проходить через усі рівні системи до пристрою і як результат цієї дії повертається користувачу. Вона підкреслює важливість взаємодії між компонентами системи для забезпечення стабільної та безпечної роботи *IoT*-рішень.

Розробка *ER*-діаграми є важливим етапом проєктування бази даних для системи *IoT*. Вона дозволяє зрозуміти, як організовані та взаємопов’язані дані, що забезпечує ефективну підтримку роботи системи, її моніторинг і управління.

ER-діаграма (рис. 2.4) відображає логічну структуру даних, які використовуються в системі *IoT*, зокрема в підсистемі охорони. Вона призначена для моделювання основних сутностей, їхніх властивостей і зв'язків між ними. Ця діаграма допомагає зрозуміти, як інформація організована і як взаємодіють різні компоненти системи.

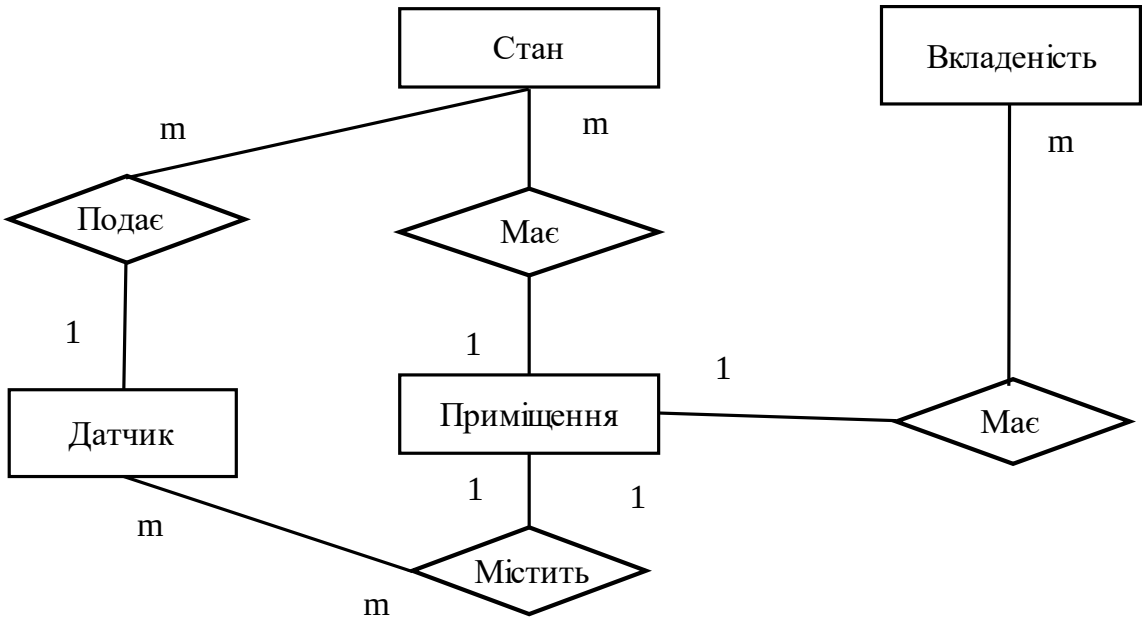


Рис. 2.4. *ER*-діаграма підтримки роботи *IoT* системи

Основні сутності діаграми:

1. Датчик: основна сутність системи, яка відповідає за збирання даних про навколишнє середовище (наприклад, температура, рух, вологість). Пов'язана із станом, приміщенням і вкладеністю.

2. Стан: описує поточний стан датчика (активний, неактивний, аварійний тощо). Має зв’язок «один до одного» з датчиком.

3. Приміщення: представляє приміщення, в якому встановлений датчик (наприклад, кухня, вітальня, склад). Має зв’язок «один до багатьох» з датчиками.

4. Вкладеність: додаткові дані, пов’язані з датчиком (наприклад, документація, фотографії чи налаштування). Пов’язана з датчиком зв’язком «один до багатьох».

Основні зв'язки між сутностями:

1. Датчик-Стан: зв’язок «один до одного». Кожен датчик має один поточний стан, який описує його роботу.

2. Датчик-Приміщення: зв’язок «багато до одного». Один датчик може належати лише одному приміщенню, але одне приміщення може містити кілька датчиків.

3. Датчик-Вкладеність: зв’язок «один до багатьох». Кожен датчик може мати кілька вкладених елементів (наприклад, файли налаштувань, логи тощо).

4. Стан-Вкладеність: зв’язок «один до багатьох». Стан датчика може мати додаткові вкладені дані, які описують деталі про цей стан (наприклад, час зміни стану або пов'язані події).

На рис. 2.5 представлено алгоритм обробки команд, отриманих від датчиків у системі *IoT*. Алгоритм починається з прийому команди управління, яка може бути ініційована користувачем або автоматичною системою. Після цього система перевіряє джерело команди. Якщо команда надходить через мобільний додаток, вона переходить до виконання. Якщо джерело інше, обробка може йти за альтернативним сценарієм.

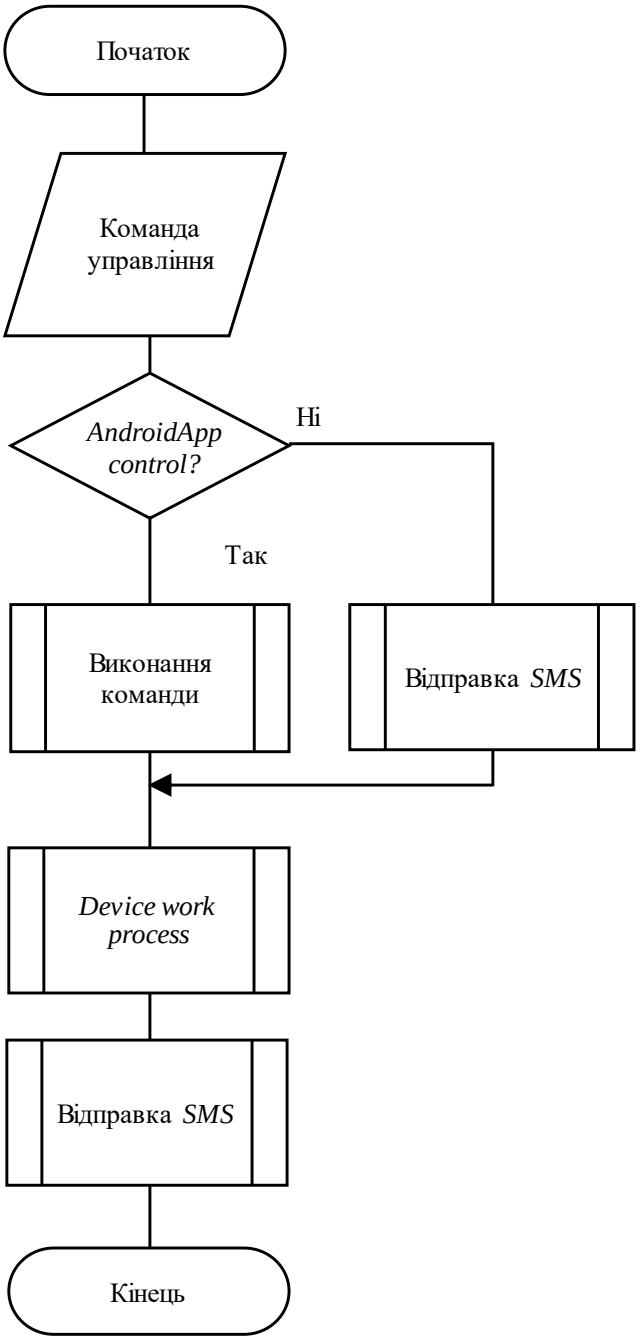


Рис. 2.5. Схема алгоритму обробки команд з датчиків

Виконання команди передбачає її реалізацію на рівні пристрою, що включає виконання відповідної дії, наприклад, ввімкнення, вимкнення пристрою або зміну його стану. У процесі виконання пристрій передає дані про стан і результати роботи, що фіксується в системі. Для забезпечення прозорості та інформування користувача відправляється *SMS*-повідомлення. Це повідомлення може містити інформацію про успішне виконання команди, поточний стан пристрою або повідомлення про помилки.

Після завершення обробки команда вважається виконаною, а система переходить до стану очікування нових команд. Така структура алгоритму дозволяє забезпечити гнучке управління пристроями, чіткий зворотний зв’язок з користувачем і високу ефективність роботи *IoT*-системи.

На рис. 2.6 представлено алгоритм передачі та обробки команди «Скинути тривогу» у системі *IoT*.

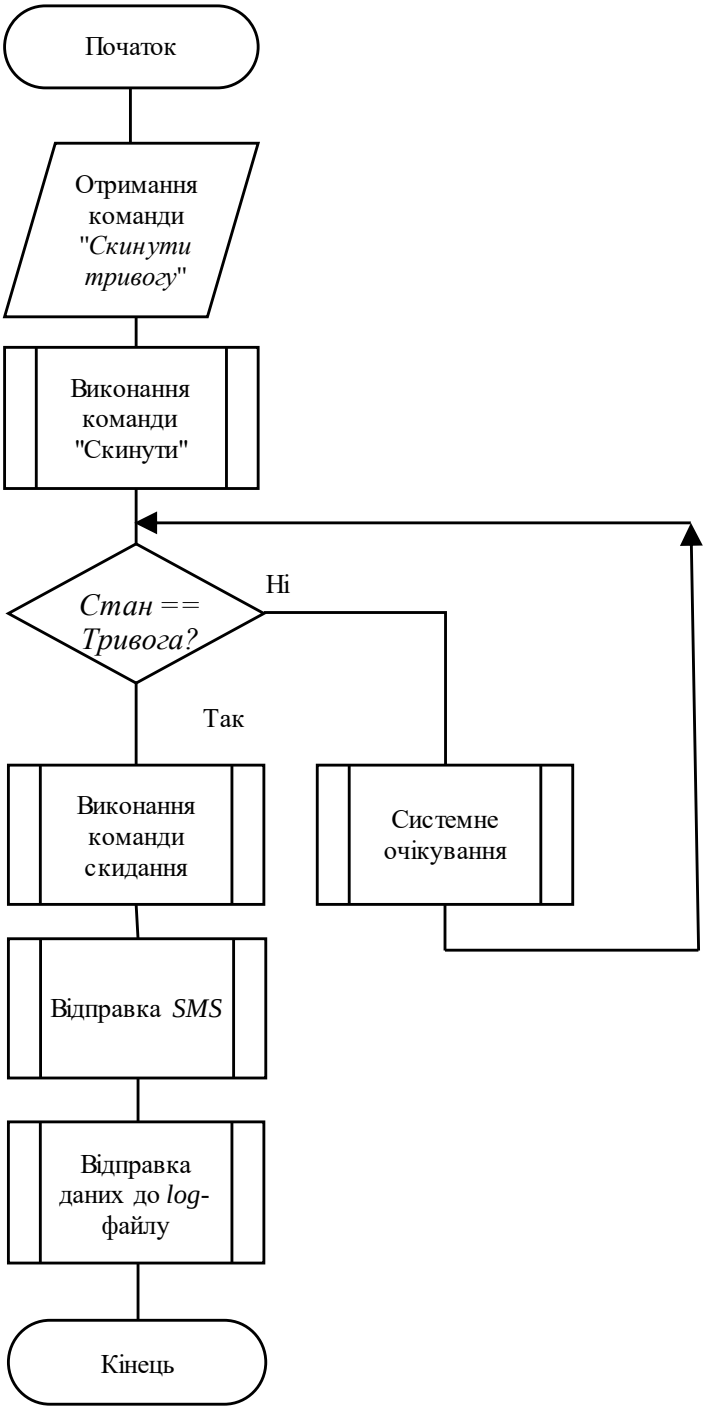


Рис. 2.6. Схема алгоритму передачі захищеної команди «Скинути тривогу»

Процес починається з отримання відповідної команди від користувача або автоматизованої системи. Після цього система виконує команду «Скинути тривогу» і переходить до перевірки поточного стану. У випадку, якщо стан системи все ще визначається як «Тривога», система переходить у режим очікування для повторної перевірки або подальшої обробки сигналу. Якщо стан змінюється на нормальний, виконується команда скидання, яка може включати відключення сповіщення або переведення датчиків у пасивний режим. Далі система відправляє *SMS*-повідомлення для інформування користувача про виконання команди, що забезпечує зворотний зв'язок і підтвердження дії. Крім цього, дані про виконану команду записуються до *log*-файлу для створення історії подій та можливості подальшого аналізу. Завершенням алгоритму є повернення системи до стандартного режиму роботи та очікування нових команд. Алгоритм забезпечує чітку послідовність дій, контроль стану системи та ефективне інформування користувача.

2.5. Висновки до розділу

У розділі було детально розглянуто можливості використання блокчейн-технологій для захисту даних у *IoT*-системах. Проведений аналіз показав, що блокчейн забезпечує високий рівень захисту інформації завдяки децентралізованій природі, криптографічній безпеці, прозорості та незмінності даних. Принципи роботи блокчейну, включаючи використання розподілених реєстрів, алгоритмів консенсусу та смарт-контрактів, надають інструменти для забезпечення конфіденційності, цілісності та доступності інформації, яка передається між *IoT*-пристроями.

Описані переваги блокчейн для *IoT*-систем, зокрема підвищення стійкості до атак, прозорість дій, автоматизація процесів через смарт-контракти та можливість обробки мікротранзакцій, підтверджують доцільність його використання у розподілених системах. Блокчейн дозволяє усунути центральну

точку відмови, що є критично важливим для забезпечення безперебійної роботи *IoT*-систем навіть у складних умовах.

Огляд сучасних рішень на основі блокчейн для *IoT* виявив, що такі платформи, як *Ethereum, Hyperledger* і *IoTA*, активно впроваджують інновації у сфері захисту даних. Рішення на основі смарт-контрактів дозволяють автоматизувати аутентифікацію пристроїв, управління доступом і запис подій у розподілений реєстр, що суттєво підвищує безпеку та зменшує операційні витрати.

У розділі також було розроблено архітектуру програмного рішення для *IoT*-системи, яка використовує блокчейн як основний механізм захисту. Запропонована архітектура забезпечує інтеграцію *IoT*-пристроїв, серверів і блокчейн-мережі, що дозволяє досягти високого рівня безпеки та масштабованості. Вона враховує взаємодію між компонентами системи та ефективну організацію зберігання і передачі даних.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО МЕТОДУ ЗАХИСТУ ДАНИХ В *IOT*-СИСТЕМАХ

3.1. Практична реалізація методу захисту на основі блокчейн

В попередньому розділі окремо будо розглянуто рішення з використанням блокчейну для подолання низки пролем *IoT*. В цьому розділі буде запропоновано модель *IoT* з впровадженням блокчейну і наведено доведенняїї ефективоності.

Проста архітектура *IoT*-блокчейн складається з чотирьох шарів. Блокчейн додається як окремий шар між мережевим та прикладним рівнями.

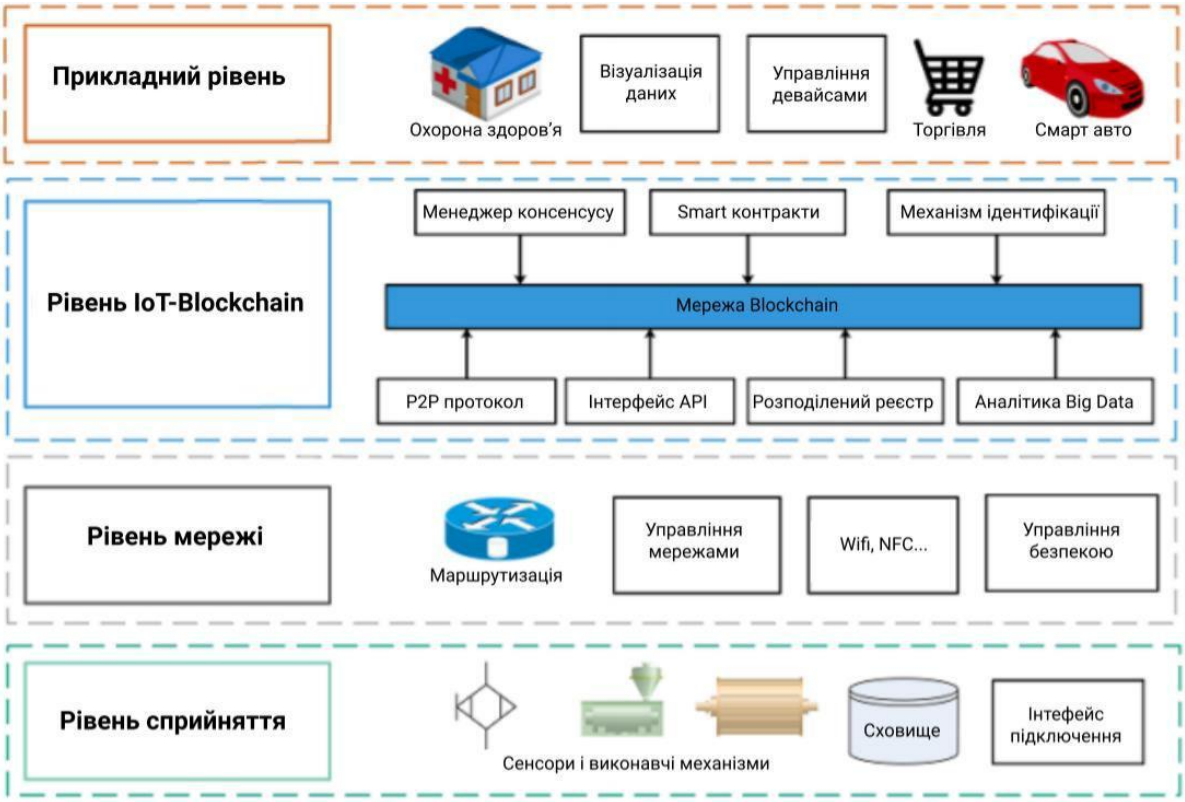


Рис.3.1. Рівнева модель *IoT* з використанням блокчейн

Перший рівень – це рівень сприйняття, який містить речі і предмети *IoT*, такі як датчики та виконавчі механізми, які використовуються для сприйняття навколишнього середовища та збору відповідних даних, які можуть допомогти

зрозуміти оточення. Потім мережевий рівень, який здійснює управління мережею та маршрутизацією, він дозволяє зв’язувати всі об’єкти Інтернету речей та взаємодіяти через Інтернет [15]. Цей рівень включає в себе мережеві та захисні пристрої, які забезпечують зв'язок та управління безпекою.

Доданий рівень включає всі модулі, що дозволяють реалізувати різні функції технології *blockchain* в системі *IoT*. Ці особливості включають зв'язок *P2P*, розподілену книгу, смарт-контракти, *API*, аналіз великих даних, управління консенсусом та управління ідентифікацією. Протоколи *P2P* необхідні для забезпечення децентралізованого зв'язку між різними об'єктами *IoT*. Крім того, розподілений журнал є однією з важливих функцій, на котрі кожен пристрій *IoT* може мати копію, щоб його можна було оновлювати за кілька хвилин або навіть секунд у мережі *IoT*. Книга може бути створена як з дозволу, так і без нього. Тип книги буде сильно залежати від контексту та кількості вузлів у мережі *IoT*.

Модуль аналізу великих даних дозволяє блокчейну забезпечити ефективне онлайн-зберігання та обробку даних, оскільки система *IoT* створює величезні обсяги даних, які неможливо обробити за допомогою традиційних методів. Крім того, багато операцій депонуються в структурованих формах книг, що вимагатиме подальшого аналізу даних. Розумні контракти також є однією з важливих частин технології блокчейну, яка застосовується для автоматизованих рішень на основі заздалегідь визначених умов.

Як правило, смарт-контракт – це програмний код, який запускається за допомогою блокчейну для виконання набору дій, коли заздалегідь визначені умови виконуються або перевіряються.

Управління консенсусом також є однією з основних функцій, необхідних для інтеграції блокчейну з *IoT*. Він діє як центральний сервер, який підтримує довіру між комунікаційними вузлами в мережі.

Управління ідентифікацією використовується для контролю та ідентифікації різних вузлів у мережі *IoT*. Крім того, інтерфейс *API* дозволяє додаткам *IoT* отримувати доступ до послуг блокчейну. Рівень додатків – це

верхній рівень, який включає різні програми *IoT* та забезпечує завдання візуалізації даних, які створюють численні оцифровані служби та допомагають особам, що приймають рішення, приймати точні та точні рішення на основі зібраних даних з фізичних пристроїв *IoT* [16].

Традиційні методи перевірки цілісності даних зазвичай використовують методи шифрування для захисту даних у хмарі, покладаючись на довірених сторонніх аудиторів (*TPA*). Схеми цілісності даних, засновані на блокчейні, можуть успішно уникнути проблеми довіри *TPA*, однак їм доводиться стикатися з проблемами великих обчислювальних та комунікаційних накладних витрат. Для вирішення вищезазначених питань пропонується схема цілісності даних (*BB-DIS*) на основі блокчейну та білінейного картографування для великомасштабних даних *IoT*. У *BB-DIS* дані *IoT* нарізані на фрагменти та генеруються гомоморфні теги (*HVT*) для перевірки вибірки. Цілісність даних може бути досягнута відповідно до характеристик білінійного зіставлення у вигляді транзакцій блокчейну [18].

3.1.1. Представлення математичної моделі

Запропонована модель є децентралізованою для вирішення проблеми єдиної точки довіри в традиційній моделі служби аудиту даних колективної довіри. Протокол дозволяє користувачам відстежувати історію своїх даних. Таким чином, більшість існуючих методів перевірки цілісності даних, заснованих на технології блокчейну, зосереджуються на проблемі довіри, а не на розмірі даних [19]. Більш помітним питанням є те, що дані *IoT*, що зберігаються хмарі, потрібно оновлювати в режимі реального часу, щоб задовольнити найновіші вимоги різних додатків. Тому необхідно запропонувати динамічне рішення на основі блокчейну, спрямоване на оновлення даних для перевірки цілісності даних [20].

Технологія блокчейн реалізує децентралізовані однорангові транзакції, координацію та співпрацю без потреби довіри за допомогою шифрування

даних, відмітки часу та розподіленого консенсусу. Це може вирішити проблему високої вартості, неефективності та небезпечного зберігання даних централізованих систем [17].

Таблиця 3.1

Структурні елементи моделі

Позначення	Пояснення
DOD	Прилад – власник даних
DCD	Прилад – користувач даних
CSP	Провайдер хмарних послуг
HSSC	Смарт-контракт на зберігання *HVT*
CRSC	Виклик отримання смарт- контракту
IVSC	Смарт-контракт для перевірки доброчесності

Структура моделі в основному включає чотири типи об'єктів, тобто смарт-контракти, пристрої, що володіють даними (*DOD*), споживачі даних (*DCD*) та постачальники хмарних послуг (*CSP*). Для досягнення різних функцій існує три види смарт-контрактів, тобто смарт-контракт на зберігання *HVT* (*HSSC*), інтелектуальний контракт на отримання виклику (*CRSC*) та інтелектуальний контракт для перевірки цілісності (*IVSC*). Усі ці сутності можуть діяти як вузли блокчейну в мережі блокчейнів. Насправді перевірка цілісності даних залучає декількох власників даних та споживачів даних. Перевірка цілісності виконується за допомогою смарт-контрактів у системі блокчейнів. Користувачі з вимогами цілісності можуть запускати клієнтів блокчейну на своїх вузлових пристроях або виходити з мережі блокчейн. *CSP* також служить вузлом у мережі блокчейнів, що робить вузли повністю розпорошеними, а перевірку цілісності більш ефективною [21].

DOD і *DCD* слід додавати до мережі блокчейнів, коли система блокчейнів ініціалізується для створення пари ключів. Власник даних повинен заплатити за взаємодію зі смарт-контрактом та хмарною службою зберігання. *CSP* може

виступати як майнер-вузол у мережі блокчейнів, який кваліфікований для надання послуг через майнінг та отримання відповідної винагороди. Споживач даних просить використовувати дані, що зберігаються на хмарному сервері, і оплачує відповідні витрати за це. У цій структурі *CSP* забезпечує загальну послугу зберігання даних для власників даних, тоді як нехмарні дані можуть передаватися через міжвузольну мережу *P2P*.

3.1.2. Протокол верифікації

Процес верифікації цієї схеми показаний на рис. 3.2, а транзакції між різними смарт-контрактами та всіма учасниками. Протокол розділений на три етапи: етап кроку, етап виклику та етап перевірки. Розумний контракт та *CSP* виконують роль перевіряючого та постачальника доказів відповідно [22].

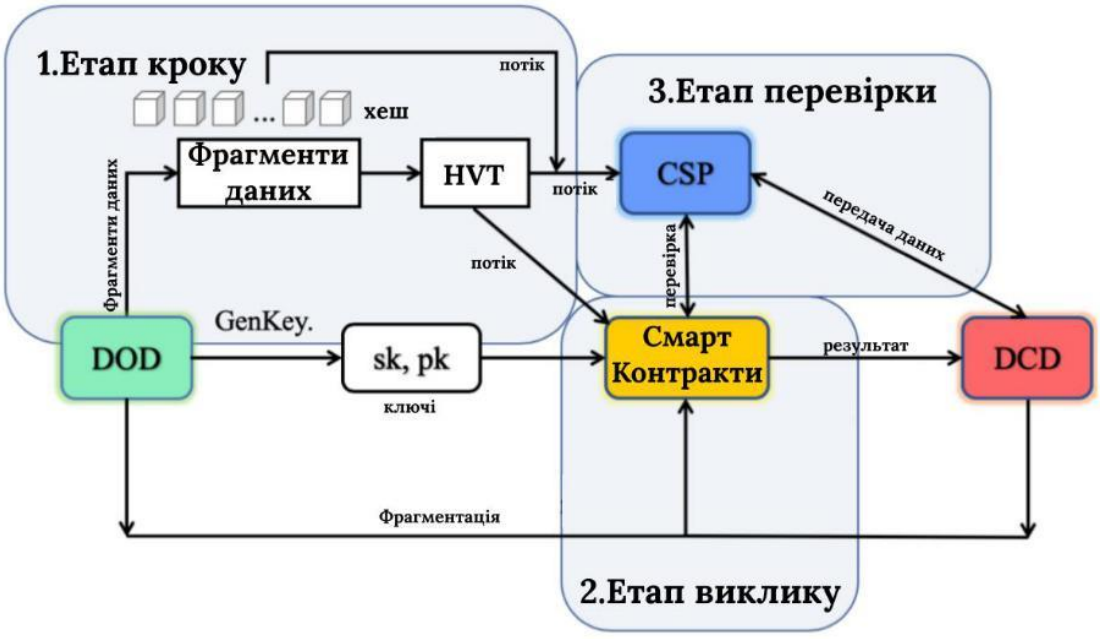


Рис.3.2. Протокол перевірки отриманих даних

Етап кроку: *DOD* встановлює двобічне відображення, вибирає хеш-функцію короткого підпису, випадково вибирає приватний ключ і обчислює відповідний відкритий ключ із закритого ключа. Потім *DOD* нарізає файл даних на набір з декількох фрагментів даних однакової довжини [23]. Після

хешування *DOD* обчислює *HVT* кожного фрагменту даних, щоб сформувати набір метаданих аутентифікації, що зменшує накладні витрати на зв'язок, підтримуючи публічний аудит. *DOD* завантажує набір фрагментів даних і метадані на сервер хмарного сховища. Набір метаданих надсилається *HSSC* через мережу блокчейнів у формі транзакції. *DOD* видаляє файл даних локально [24].

Етап виклику: *DOD* витягує *c*-елементи з *HSSC* для побудови випадково встановленого індексу фрагментів даних і надсилає ряд випадкових значень разом із індексом фрагменту даних, заданому *CSP* та *CRSC* у вигляді *chal*, запиту на виклик.

Етап перевірки: *IVSC* отримує *HVT* та *chal* від *HSSC* та *CRSC* відповідно. Отримавши запит на виклик, *CSP* обчислює доказ {*R*, μ, η} і відправляє його в *IVSC*. *IVSC* перевіряє, чи є підтвердження правильним. Якщо це правильно, дані, що зберігаються в хмарі, інтегруються, і *IVSC* повертає результат до *DOD*.

3.1.3. Програмний інтерфейс для налаштування блокчейн в мережі *IoT*

Для інтеграції блокчейн-технологій в *IoT*-системи необхідно розробити зручний програмний інтерфейс, який дозволить легко налаштовувати і керувати блокчейн-компонентами в мережі пристроїв. Програмний інтерфейс для налаштування блокчейну в *IoT* повинен мати можливість гнучко адаптуватися до різних типів пристроїв і застосувань, а також підтримувати автоматизовані процеси конфігурації та управління.

Основні компоненти програмного інтерфейсу:

1. Інтерфейс для реєстрації пристроїв: кожен *IoT*-пристрій, що підключається до блокчейн-мережі, повинен бути автентифікований і зареєстрований [25]. Програмний інтерфейс забезпечує механізм реєстрації пристроїв шляхом створення унікального криптографічного ключа для кожного пристрою, що дозволяє ідентифікувати його у мережі;

Функціональність:

– генерація криптографічних ключів (асиметричні або симетричні алгоритми);

– реєстрація пристрою в блокчейні з автоматичним створенням запису у реєстрі;

– підтвердження автентичності пристрою на основі унікального ключа.

2. Інтерфейс для налаштування смарт-контрактів

Смарт-контракти в блокчейні відіграють важливу роль в автоматизації процесів обміну даними і управління доступом до ресурсів в *IoT*-мережі. Програмний інтерфейс повинен забезпечити можливість налаштування та управління смарт-контрактами без необхідності глибоких технічних знань;

Функціональність:

– створення та налаштування смарт-контрактів для автоматизації дій між *IoT*-пристроями;

– визначення умов виконання контрактів (наприклад, надання доступу до даних тільки за умови виконання певних дій);

– можливість перевірки та редагування смарт-контрактів через простий інтерфейс користувача.

3. Моніторинг і візуалізація блокчейн-активності: для забезпечення прозорості і контролю за процесами в *IoT*-мережі важливо мати доступ до інструментів моніторингу транзакцій та інших дій, що виконуються в блокчейні. Програмний інтерфейс повинен мати зручну систему візуалізації, яка дозволить відслідковувати взаємодію пристроїв і перевіряти статус смарт-контрактів;

Функціональність:

– відображення у реальному часі транзакцій між пристроями в мережі;

– журнал транзакцій для аудиту всіх дій в блокчейні;

– інструменти аналізу даних для виявлення можливих проблем або аномалій у роботі мережі.

4. Інтерфейс для управління доступом: важливою частиною захисту даних в *IoT*-системах є правильне управління доступом до ресурсів та даних.

Програмний інтерфейс повинен дозволяти адміністраторам або користувачам налаштовувати політики доступу до різних пристроїв або даних, використовуючи можливості смарт-контрактів і блокчейн-записів;

Функціональність:

– Налаштування прав доступу для різних пристроїв і користувачів;

– Інтеграція механізмів аутентифікації та авторизації через блокчейн;

– Можливість встановлення динамічних політик доступу залежно від статусу пристроїв або умов смарт-контрактів.

5. *API* для інтеграції з іншими системами: програмний інтерфейс повинен підтримувати можливість інтеграції блокчейн-мережі з іншими системами через стандартні *API* (*Application Programming Interface*). Це дозволить підключати *IoT*-пристрої, які вже працюють в інших середовищах, до блокчейн-мережі для забезпечення їх безпеки і захисту даних;

Функціональність:

– *RESTful API* для інтеграції з зовнішніми системами та сервісами;

– Підтримка *JSON* або *XML* для передачі даних між системами;

– Можливість отримання та передачі даних із зовнішніх додатків у блокчейн.

Переваги програмного інтерфейсу:

– простота використання: інтерфейс повинен бути інтуїтивно зрозумілим, щоб навіть користувачі без глибоких технічних знань могли налаштувати блокчейн-систему для своїх *IoT*-пристроїв;

– безпека: усі взаємодії з блокчейн-мережею повинні бути захищені криптографічними механізмами, що гарантує високий рівень безпеки під час налаштування та управління системою;

– гнучкість: програмний інтерфейс має підтримувати налаштування для різних типів *IoT*-пристроїв, а також дозволяти інтеграцію з іншими системами і платформами;

– масштабованість: система повинна бути здатною обробляти велику кількість пристроїв і транзакцій без втрати ефективності, що важливо для великих мереж *IoT*.

Програмний інтерфейс для налаштування блокчейн в мережі *IoT* є критично важливим компонентом для впровадження технології в *IoT*-системи. Він забезпечує зручний і безпечний спосіб реєстрації пристроїв, налаштування смарт-контрактів, моніторингу транзакцій і управління доступом. Завдяки інтуїтивному інтерфейсу та високому рівню безпеки, користувачі можуть легко інтегрувати свої *IoT*-пристрої в блокчейн-мережу, забезпечуючи надійний захист даних і безперебійну роботу системи (рис. 3.3).

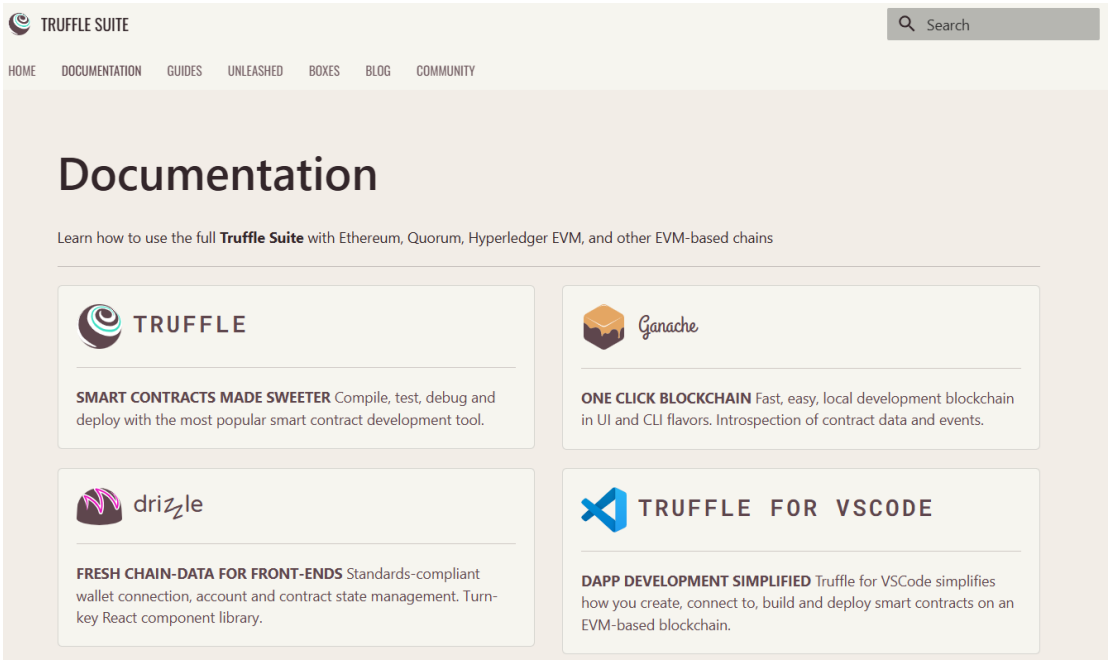


Рис. 3.3. Підключення системи формування смарт-контактів

3.2. Описання розробленого інтерфейсу налаштування компонентів *IoT*-системи

На початковому екрані користувач бачить основну сторінку системи (рис. 3.4). Якщо в системі ще не створено жодного приміщення, користувачу пропонується створити нове приміщення через кнопку "*Create New*".

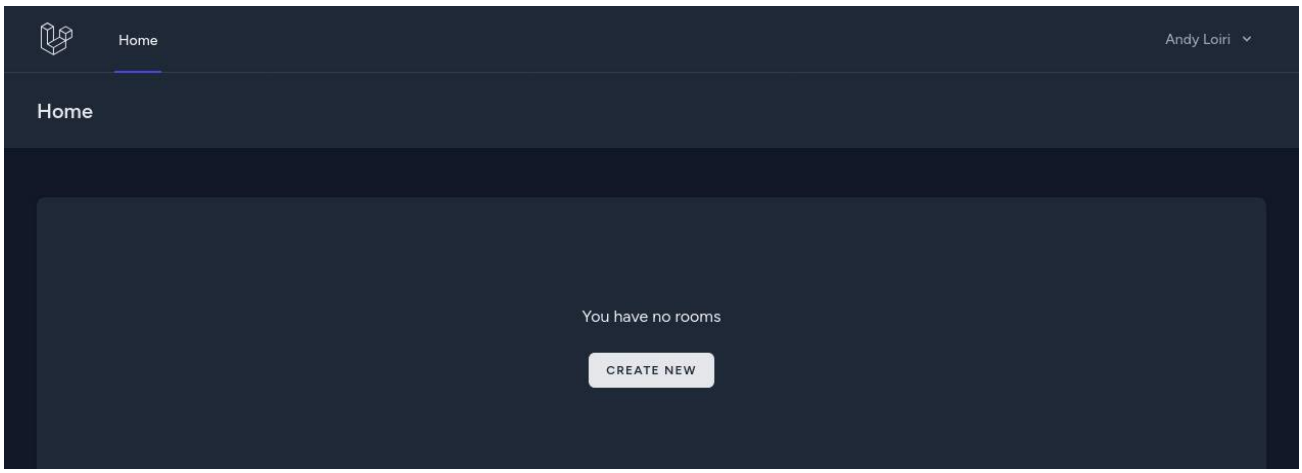


Рис. 3.4. Початкове вікно налаштування *IoT*-системи

Це дає змогу організувати простір для подальшого додавання пристроїв та їх налаштувань.

Після створення приміщення (рис. 3.5) на головній сторінці відображається список наявних приміщень. Для кожного приміщення вказується його назва, дата створення, а також є опція "*View details*" для перегляду додаткової інформації. Користувач може додати нові приміщення за допомогою кнопки "*Create New*".

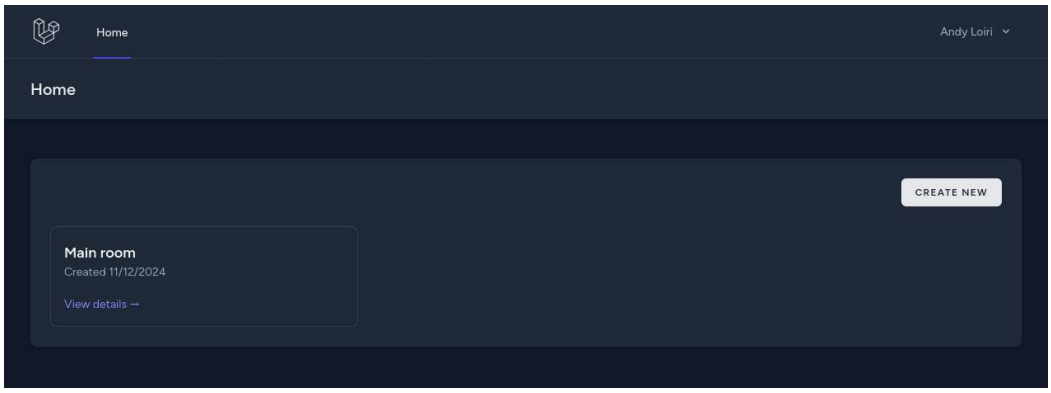


Рис. 3.5. Вікно створення приміщення в межах *IoT*-системи

Вибравши конкретне приміщення (рис. 3.6), користувач переходить на сторінку, де відображається список доданих до цього приміщення пристроїв. Наприклад, можна побачити пристрій "*Sensor* 1", для якого вказано тип (сенсор) та дату додавання. Для роботи з пристроєм передбачена кнопка "*View details*", а також опція додавання нових пристроїв через кнопку "*Add Device*".

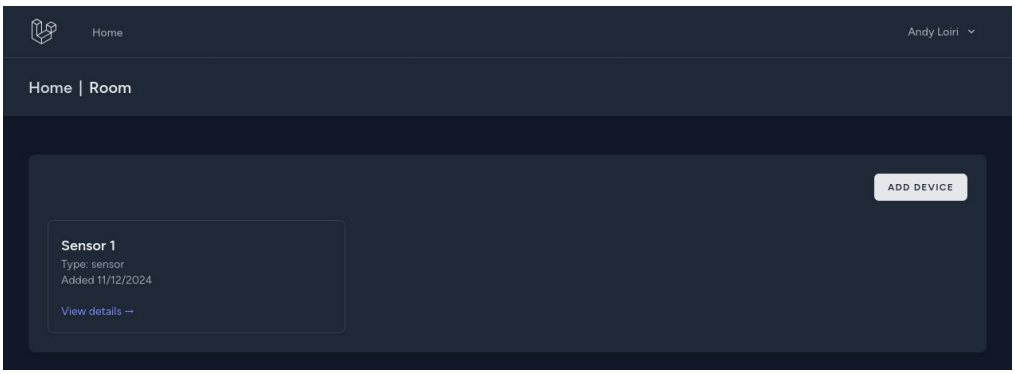


Рис. 3.6. Вікно налаштування пристроїв приміщення

При виборі пристрою (рис. 3.7) користувач отримує доступ до сторінки з детальною інформацією. Відображаються основні параметри пристрою, такі як назва, тип, дата створення та додаткова інформація про його стан. Ця сторінка дає можливість керувати пристроєм, переглядати історію роботи або конфігурувати смарт-контракти.

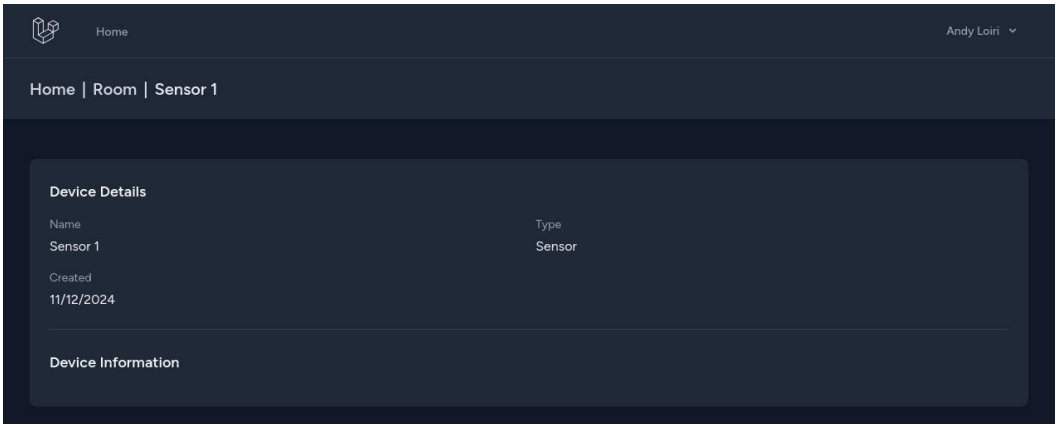


Рис. 3.7. Вікно детальної інформації про пристроій

На розширеній сторінці пристрою (рис. 3.8) відображаються дві секції: незашифровані дані (*Unencrypted Data*) та зашифровані дані (*Encrypted Data*). У секції незашифрованих даних можна побачити інформацію про події, наприклад, виявлення руху або підтвердження нормального стану пристрою. Для зашифрованих даних передбачено поле, де відображатиметься відповідна інформація після її обробки смарт-контрактами.

Додатково користувач має змогу вручну додавати дані в систему. Для цього доступні поля введення, що дозволяють додати як незашифровані, так і

зашифровані дані. Ці функції допомагають налаштувати специфічні параметри пристроїв та забезпечити безпеку обробки даних.

Система забезпечує створення та прив'язку смарт-контрактів до пристроїв, що дозволяє автоматизувати обробку даних і виконання дій. Наприклад, у разі виявлення аномалій пристрій може автоматично виконувати задані дії через смарт-контракти, зокрема надсилання сповіщень або зміни стану.

Інтерфейс системи є зручним та інтуїтивно зрозумілим, що дозволяє користувачам ефективно управляти пристроями, створювати смарт-контракти та відстежувати стан *IoT*-системи в режимі реального часу.

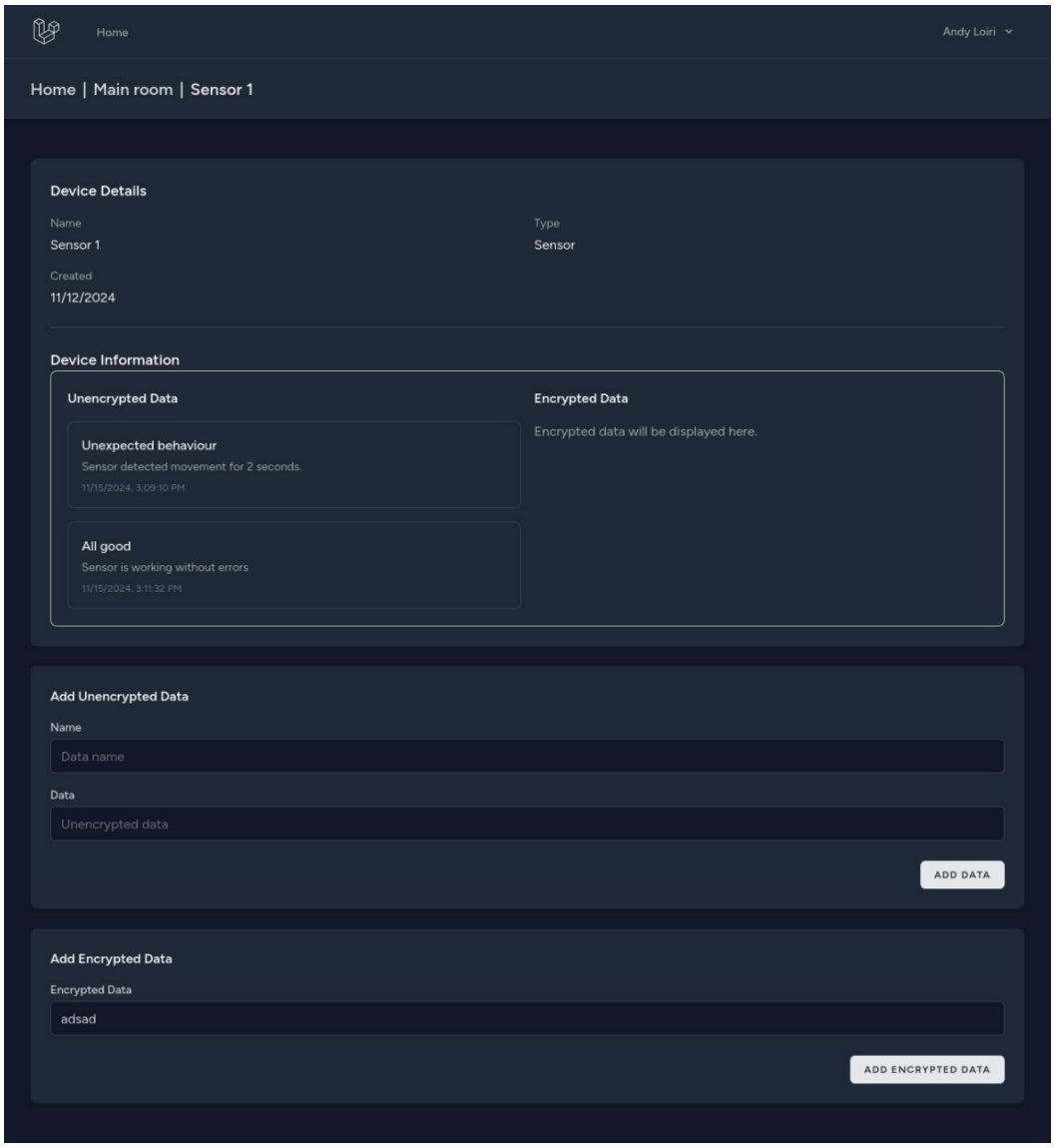


Рис. 3.8. Вікно створення та прив'язки смарт-контрактів до пристроїв

3.3. Тестування передачі даних *IoT*-системи через мобільний додаток

Тестування передачі даних *IoT*-системи через мобільний додаток є важливим етапом для перевірки стабільності, швидкості та коректності обміну інформацією між користувачем і пристроями.

Мобільний додаток виступає основним інтерфейсом для управління *IoT*-системою, дозволяючи надсилати команди, отримувати дані з датчиків та стежити за станом системи в реальному часі. Перед початком роботи з додатком необхідно пройти процедуру перевірки системи, що забезпечує авторизацію користувача та аутентифікацію пристроїв (рис. 3.9).

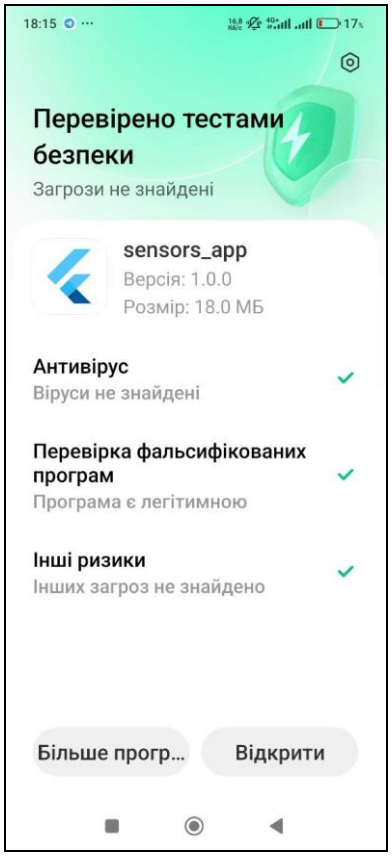


Рис. 3.9. Вікно встановлення і перевірки додатку

Цей процес включає завантаження мобільного додатка, реєстрацію облікового запису, введення унікальних ідентифікаційних даних та підтвердження прав доступу. Така перевірка дозволяє уникнути несанкціонованого доступу до *IoT*-системи.

Під час тестування передача даних здійснюється шляхом взаємодії між мобільним додатком, сервером та *IoT*-пристроями. Основні етапи включають:

1. Надсилання запиту від мобільного додатка до серверу для виконання певної дії, наприклад, увімкнення пристрою чи отримання інформації від датчика.

2. Обробку запиту сервером, який передає команду на відповідний контролер або пристрій.

3. Виконання дії пристроєм та повернення результатів у вигляді даних або підтвердження виконання.

4. Передачу результатів на сервер і подальшу їх відправку мобільному додатку, де інформація відображається користувачу.

Для тестування було створено кілька сценаріїв, включаючи надсилання команд, отримання даних із датчиків, обробку помилок і тестування швидкості обміну інформацією. Наприклад, тестування включало відправку команди «увімкнути світло», перевірку отримання зворотного повідомлення про успішне виконання та аналіз часу, необхідного для виконання цієї операції. Також було перевірено роботу в умовах нестабільного підключення до мережі.

Результати тестування показали, що мобільний додаток забезпечує коректну передачу даних у більшості випадків, а час обробки команд знаходиться в межах прийнятних значень. Однак було виявлено кілька моментів, які потребують оптимізації, наприклад, покращення роботи в умовах низької швидкості мережі. Завдяки тестуванню вдалося підтвердити ефективність роботи додатка та його здатність забезпечувати стабільний зв’язок із *IoT*-системою.

У мобільному додатку, який інтегровано з сервером через смарт-контракти, реалізовано передачу даних для моніторингу та управління станом *IoT*-системи. Основна мета взаємодії – забезпечення користувачу доступу до інформації про стан приміщень та датчиків, а також можливості швидко реагувати на сигнали тривоги.

На рис. 3.10 відображено головне вікно додатку, де користувач бачить список приміщень системи.

Рис. 3.10. Вікно стану системи в мобільному додатку

У разі виявлення тривоги, відповідне приміщення виділяється, що дозволяє користувачу швидко ідентифікувати зону з проблемою. Користувач може обрати будь-яке приміщення зі списку для перегляду детальної інформації про стан датчиків [26].

На Рис. 3.11 наведено перелік датчиків у вибраному приміщенні. Для кожного датчика зазначається його тип і поточний стан. Наприклад, у приміщенні можуть бути присутні датчики руху (*Walk Sensor*) та датчики відкриття (*Lock Sensor*). Якщо тривога включена, це додатково позначається в інтерфейсі.

Рис. 3.12 демонструє вікна стану окремих датчиків. У випадку датчика руху (*Walk Sensor*) користувач бачить інформацію про те, що виявлено активність у приміщенні, і має змогу натиснути кнопку «Скинути тривогу» для відновлення нормального режиму роботи. Аналогічно, вікно стану датчика відкриття (*Lock Sensor*) показує, що стан пристрою нормальний (*OK*), або, в разі тривоги, інформує про її наявність.

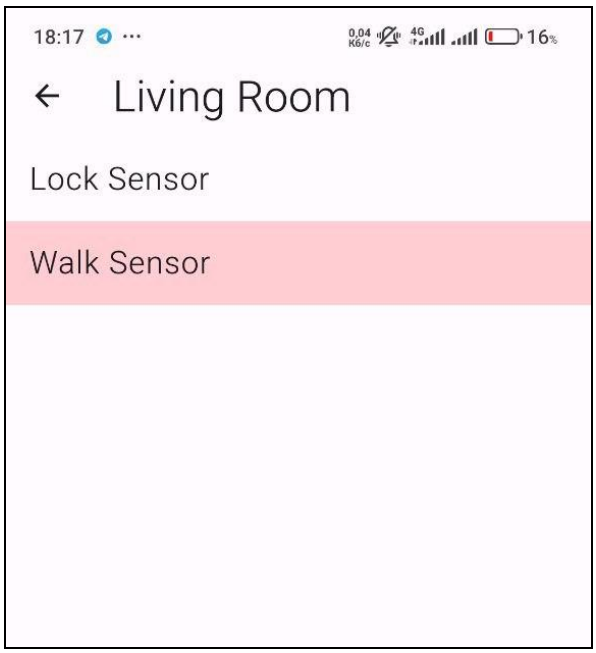
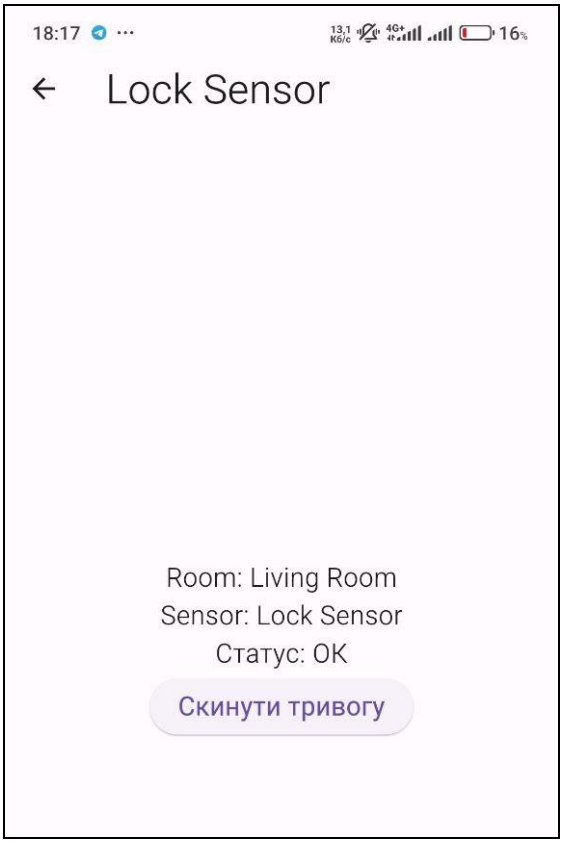


Рис. 3.11. Вікно мобільного додатку з переліком датчиків

а



б

Рис. 3.12. Вікно стану датчиків:

а) датчик руху, б) датчик відкриття

Передача даних між мобільним додатком і сервером виконується через смарт-контракти, які забезпечують автоматичне опрацювання подій. Наприклад, коли датчик виявляє рух, сервер передає цю інформацію в мобільний додаток у реальному часі. Смарт-контракт забезпечує, що дані про тривогу зашифровані і записані в блокчейн, що гарантує їхню незмінність і безпеку. Водночас мобільний додаток використовує *API* сервера для отримання оновлень і виконання дій, таких як скидання тривоги.

Таким чином, інтеграція мобільного додатку з сервером через смарт-контракти забезпечує прозорість роботи *IoT*-системи, високий рівень безпеки та зручність управління для користувача [26].

В Додатку А наведено код створення смарт-контракту основні компоненти якого наступні:

1. Структура *IoTDevice*: визначає атрибути пристрою, такі як назва, тип, власник, статус і останні отримані дані.

2. Функція *addDevice*: дозволяє користувачу реєструвати новий пристрій, вказуючи його адресу, назву та тип.

3. Функція *setDeviceStatus*: використовується для активації або деактивації пристрою.

4. Функція *sendData*: дозволяє пристрою надсилати дані, які зберігаються у вигляді рядка.

5. Функція *getDevice*: дозволяє отримувати інформацію про пристрій, зокрема його статус та останні дані.

6. Події: *DeviceAdded*, *DataUpdated*, *StatusChanged* – використовуються для логування дій у блокчейн-мережі.

Для використання необхідно:

1. Розгорнути контракт у мережі *Ethereum* за допомогою інструментів, таких як *Remix* або *Truffle* (в роботі використовувався *Truffle*);

2. Скоритстатись *API* для взаємодії з контрактом, щоб реєструвати пристрої, змінювати їхній статус або передавати дані;

3. Передати дані в мобільний додаток і веб-інтерфейс для зручного управління пристроями (рис. 3.13).

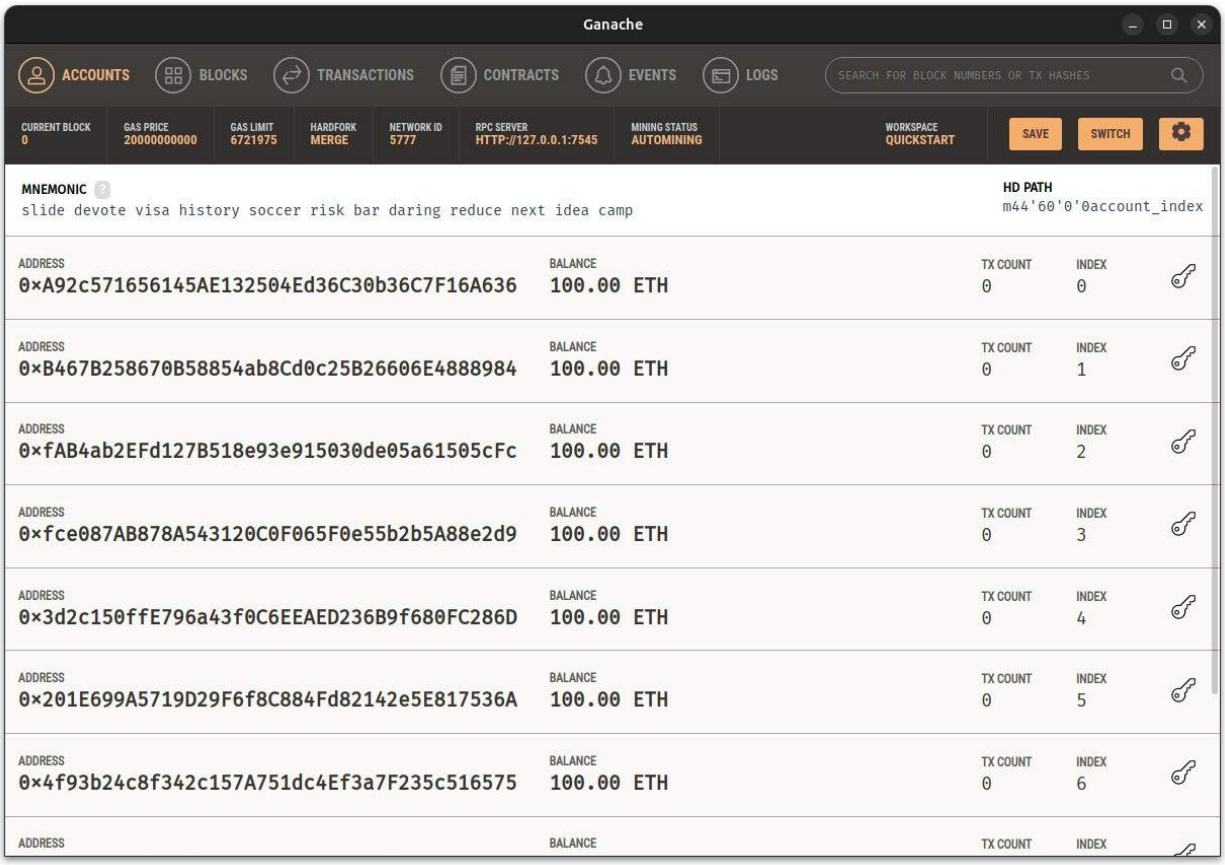


Рис. 3.13. Приклад роботи з створеними смарт-контрактами

3.4. Порівняння ефективності блокчейн технології із традиційними методами захисту даних в *IoT*

3.4.1. Проведення експерименту для порівняння блокчейн технології із традиційними методами захисту даних в *IoT*

Для проведення аналізу ефективності захисту *IoT*-систем розглянемо два підходи: традиційні методи захисту та використання смарт-контрактів на *Ethereum*. Також будуть запропоновані метрики для оцінки цих підходів.

Традиційні методи захисту *IoT*:

1. Шифрування даних (*TLS*/*SSL*): використання протоколів шифрування для забезпечення конфіденційності даних при передачі. Наприклад, *TLS* захищає дані між *IoT*-пристроями і сервером.

2. Аутентифікація через *PKI*: інфраструктура відкритих ключів (*PKI*) дозволяє забезпечити безпечну ідентифікацію *IoT*-пристроїв через сертифікати.

3. Брандмауери і *VPN*: забезпечують захист периметру мережі *IoT*, запобігаючи несанкціонованому доступу.

4. *IDS*/*IPS*: системи виявлення (*IDS*) та запобігання вторгненням (*IPS*) аналізують мережевий трафік і блокують підозрілі активності.

Протоколи *TLS* (*Transport Layer Security*) і *SSL* (*Secure Sockets Layer*) є одними з найпоширеніших стандартів для захисту передачі даних між *IoT*-пристроями та сервером. Ці протоколи використовують шифрування для забезпечення конфіденційності та цілісності переданої інформації.

TLS/*SSL* працює за рахунок встановлення захищеного каналу зв'язку між клієнтом (*IoT*-пристроєм) і сервером. Процес включає кілька етапів:

1. Аутентифікація: сервер передає клієнту сертифікат, виданий довіреним центром сертифікації (*CA*), для підтвердження своєї особи.

2. Установлення ключів: клієнт і сервер обмінюються криптографічними ключами, використовуючи асиметричне шифрування, щоб створити сеансовий ключ.

3. Шифрування: усі дані, передані через канал, шифруються за допомогою симетричного шифрування, використовуючи сеансовий ключ.

Цей метод захищає дані від несанкціонованого доступу під час передачі, запобігаючи таким атакам, як «людина посередині» (*MITM*). Однак *TLS*/*SSL* залежить від централізованого управління ключами, що створює потенційні ризики компрометації у разі зламу центрального сервера або витоку сертифікатів.

Основні переваги:

– захист даних у реальному часі;

– висока швидкість шифрування;

– широке використання та стандартизація.

Недоліки:

– вразливість до атак у разі компрометації ключів або серверу;

– необхідність централізованого управління сертифікатами.

Аутентифікація через *PKI* забезпечує механізм для створення, управління і перевірки цифрових сертифікатів, які підтверджують автентичність пристроїв *IoT*. Ці сертифікати використовуються для перевірки особи пристрою перед наданням доступу до ресурсів або даних.

PKI базується на парі ключів: відкритому і закритому. Відкритий ключ публікується і використовується для шифрування даних або перевірки підпису, тоді як закритий ключ зберігається конфіденційно і використовується для розшифрування або підпису.

Процес аутентифікації через *PKI* включає:

1. Видачу сертифікатів: Центр сертифікації (*CA*) видає кожному *IoT*-пристрою сертифікат, який містить відкритий ключ і інформацію про пристрій.

2. Перевірку сертифікатів: Перед взаємодією пристрій перевіряє сертифікат іншого пристрою або сервера, щоб переконатися, що він дійсний і виданий довіреним *CA*.

PKI забезпечує високу надійність і безпеку, але має обмеження у масштабованості, оскільки управління великою кількістю сертифікатів може бути складним завданням у великих *IoT*-мережах.

Основні переваги:

– надійна аутентифікація пристроїв;

– високий рівень захищеності за умови належного управління сертифікатами.

Недоліки:

– складність управління сертифікатами в масштабованих системах;

– залежність від довірених центрів сертифікації.

Брандмауери і віртуальні приватні мережі (*VPN*) забезпечують захист *IoT*-систем, ізолюючи їх від зовнішніх мереж і запобігаючи несанкціонованому доступу.

Брандмауери аналізують вхідний і вихідний мережевий трафік відповідно до визначених правил. Вони можуть:

– блокувати небажані з'єднання;

– обмежувати доступ до певних *IP*-адрес;

– забезпечувати фільтрацію на основі протоколів і портів.

VPN створює захищений тунель між *IoT*-пристроями і сервером, використовуючи шифрування для забезпечення конфіденційності даних. Це особливо корисно для захисту від атак у публічних мережах.

Однак ці методи можуть бути недостатньо ефективними проти атак, які здійснюються компрометованими внутрішніми пристроями, або якщо зловмисники отримують доступ до облікових даних *VPN*.

Основні переваги:

– захист периметру *IoT*-мережі;

– простота впровадження та налаштування.

Недоліки:

– не захищають від внутрішніх атак;

– обмежена масштабованість у великих *IoT*-мережах.

Системи виявлення вторгнень (*IDS*) і запобігання вторгненням (*IPS*) є інструментами для аналізу мережевого трафіку та виявлення аномалій, які можуть свідчити про атаку.

IDS пасивно моніторить трафік і надсилає оповіщення адміністратору про підозрілу активність. *IPS*, на відміну від *IDS*, активно втручається в трафік, блокуючи небажані дії.

IDS/*IPS* аналізують такі аспекти:

1. Поведінковий аналіз: Виявлення відхилень від нормальної роботи пристроїв.

2. Сигнатурний аналіз: Пошук відомих шаблонів атак.

Хоча ці системи є ефективними для захисту від зовнішніх атак, вони можуть бути менш ефективними проти нових, складних або таргетованих атак, які не мають явних сигнатур [27].

Основні переваги:

– Забезпечують виявлення і запобігання більшості атак;

– Дозволяють швидко реагувати на підозрілу активність.

Недоліки:

– Можуть мати високий рівень помилкових спрацьовувань;

– Ефективність залежить від налаштування і регулярного оновлення баз даних сигнатур.

Смарт-контракти дозволяють автоматизувати захист *IoT*-систем завдяки використанню блокчейн-технологій. Їхні основні функції:

1. Аутентифікація пристроїв: Кожен *IoT*-пристрій може бути зареєстрований у смарт-контракті з унікальним ідентифікатором. Це забезпечує безпечний механізм перевірки пристроїв без необхідності централізованого сервера.

2. Автоматизація управління доступом: Смарт-контракти дозволяють задавати правила доступу для пристроїв і автоматично виконувати їх. Наприклад, пристрої можуть отримувати доступ лише в певних часових інтервалах або при виконанні заданих умов.

3. Прозоре збереження журналів дій: Усі події в *IoT*-системі можуть бути записані в блокчейн, що забезпечує незмінність і можливість аудиту. Це знижує ризик фальсифікації даних або приховування дій.

Для проведення порівняння традиційних методів захисту та смарт-контрактів на *Ethereum* у *IoT*-системах, використовуємо наступні метрики:

1. Час аутентифікації (*Authentication Time,* в мілісекундах);

2. Час передачі даних (*Data Transmission Time,* в мілісекундах);

3. Пропускна здатність (*Throughput,* кількість транзакцій/секунда);

4. Стійкість до атак (*Resilience*) – відсоток успішно відбитих атак;

5. Витрати на обробку транзакцій (*Cost per Transaction,* в *USD*);

6. Масштабованість – кількість пристроїв, що підтримуються системою (в тисячах).

Результати аналізу кожного з методів наведено в табл. 3.2.

Таблиця 3.2

Порівняння ефективності блокчейн технології із традиційними методами захисту в *IoT*

Метрика	*TLS/ SSL*	*PKI*	Бранд–мауери/*VPN*	*IDS*/*IPS*	Смарт-контракти (*Ethereum*)
Час аутентифікації (мс)	10-50	10-50	20-100	50-200	500-600
Час передачі даних (мс)	20-100	20-100	20-100	50-200	100-500
Пропускна здатність (*TPS*)	1000+	1000+	500-1000	100-500	50-200
Стійкість до атак (успіх відбиття)	70-80%	75-85%	80-90%	85-90%	90-95%
Витрати на транзакцію (*USD*)	0.001-0.01	0.001-0.01	0.005-0.02	0.01-0.05	0.1-0.5
Масштабованість (тисяч пристроїв)	10-50	10-50	50-100	50-100	100-200

Традиційні методи, такі як *TLS*/*SSL* і *PKI*, забезпечують високу швидкість і низькі витрати, але страждають від централізованих ризиків і складності масштабування. Смарт-контракти на *Ethereum* надають переваги у стійкості до атак, прозорості та автоматизації, але мають обмеження у швидкості, пропускній здатності та витратах. Оптимальне рішення може включати комбінування цих підходів для досягнення балансу між безпекою та ефективністю.

3.4.2. Рекомендації щодо подальшого вдосконалення рішення

У процесі розробки та аналізу впровадження блокчейн-рішення для захисту даних в *IoT*-системах було виявлено низку аспектів, які можуть бути вдосконалені. Нижче наведено основні рекомендації щодо подальшого розвитку цього рішення:

1. Оптимізація швидкості обробки транзакцій: Однією з ключових проблем використання смарт-контрактів є низька пропускна здатність і значний час підтвердження транзакцій. Для вирішення цього питання рекомендовано впровадження масштабованих блокчейн-протоколів, таких як *Layer* 2 (наприклад, *Polygon* для *Ethereum*), що дозволяють обробляти транзакції поза основним блокчейном. Це зменшить затримки і підвищить продуктивність системи.

2. Інтеграція алгоритмів оптимізації енергоємності: Використання блокчейн-рішень, заснованих на алгоритмах консенсусу типу *Proof of Stake* (*PoS*), замість *Proof of Work* (*PoW*), дозволить зменшити енергоспоживання і зробить систему більш екологічно стійкою. Також доцільно врахувати можливість використання приватних блокчейнів, які менш ресурсозатратні.

3. Розширення функціоналу смарт-контрактів: Поточна реалізація смарт-контрактів охоплює базові функції захисту та автоматизації. Для підвищення гнучкості системи рекомендовано впровадити більш складні сценарії управління, такі як адаптивна аутентифікація, автоматичне виявлення аномалій та інтеграція із системами машинного навчання для прогнозування потенційних загроз.

4. Забезпечення більшої конфіденційності даних: Оскільки публічні блокчейни можуть зберігати транзакції у відкритому доступі, важливо впровадити додаткові шифрувальні механізми для захисту конфіденційних даних. Використання технологій, таких як *Zero-Knowledge Proofs* (*ZKP*) або шифрування даних на стороні пристроїв перед передачею в блокчейн, значно підвищить рівень конфіденційності.

5. Інтеграція механізмів адаптивної безпеки: Рекомендовано додати функціонал динамічного управління рівнем безпеки залежно від ризиків. Наприклад, при виявленні аномальної активності система може автоматично змінювати політики доступу або блокувати підозрілі дії.

6. Зниження вартості транзакцій: Зважаючи на високі витрати на транзакції в мережах, таких як *Ethereum*, доцільно розглянути альтернативи, наприклад, використання більш економічних блокчейн-платформ (*Polygon*, *Binance Smart Chain*) або гібридних рішень, які комбінують централізовану й децентралізовану архітектури.

7. Покращення інтеграції з *IoT*-пристроями: Рекомендовано розробити стандартизовані *API* для підключення різних *IoT*-пристроїв до блокчейн-мережі, що забезпечить легшу інтеграцію і кращу підтримку взаємодії між пристроями.

Реалізація цих рекомендацій дозволить значно підвищити ефективність, безпеку і продуктивність *IoT*-системи, забезпечуючи її стійкість до майбутніх викликів і конкурентоспроможність на ринку.

3.5. Висновки до розділу

У розділі 3 було проведено експериментальне дослідження ефективності розробленого методу захисту даних в *IoT*-системах, заснованого на блокчейн-технології. Основна увага приділялася практичній реалізації, розробці інтерфейсу налаштування компонентів системи, тестуванню передачі даних через мобільний додаток, а також порівнянню ефективності розробленого методу з традиційними підходами до захисту.

Практична реалізація методу продемонструвала ефективність інтеграції смарт-контрактів у *IoT*-системи. Смарт-контракти забезпечили прозорість, автоматизацію процесів і підвищену стійкість до атак, завдяки децентралізованій природі блокчейн. Було створено мобільний додаток із зручним інтерфейсом для налаштування і управління компонентами *IoT*-

системи. Це забезпечило користувачам можливість оперативно взаємодіяти з пристроями, отримувати актуальні дані про їхній стан і виконувати захисні дії.

Тестування передачі даних через мобільний додаток виявило низку переваг і викликів використання блокчейн у реальних умовах. Було підтверджено безпеку та цілісність даних, однак час підтвердження транзакцій і обмеження пропускної здатності блокчейн-мережі вказують на необхідність оптимізації для більш ефективної роботи в масштабованих *IoT*-системах.

Порівняння з традиційними методами захисту, такими як *TLS*/*SSL, PKI*, брандмауери та *IDS*/*IPS*, продемонструвало, що блокчейн-технологія має значні переваги у стійкості до атак, прозорості та можливості аудиту. Водночас, традиційні методи забезпечують вищу швидкість аутентифікації та передачі даних, а також нижчі витрати на обробку транзакцій.

На основі отриманих результатів було запропоновано рекомендації для подальшого вдосконалення рішення, зокрема використання масштабованих блокчейн-протоколів (*Layer* 2), інтеграція механізмів адаптивної безпеки та впровадження приватних блокчейн-мереж для підвищення конфіденційності та продуктивності.

Таким чином, результати дослідження підтвердили доцільність використання блокчейн-технологій у *IoT*-системах, а також виявили напрямки для подальшого розвитку, спрямованого на підвищення ефективності та зниження витрат.

ВИСНОВКИ

Проведене в кваліфікаційній роботі дослідження, яке включало розробку системи передачі даних для *IoT*, підтвердило актуальність і важливість розробки сучасних методів захисту даних в *IoT*-системах, враховуючи специфічні виклики, пов'язані з їх розподіленою архітектурою, обмеженими обчислювальними ресурсами пристроїв і високими вимогами до безпеки. Запропонований метод захисту даних, заснований на використанні блокчейн-технології, демонструє перспективи забезпечення надійного захисту інформації завдяки децентралізованій природі, прозорості дій, незмінності даних і можливості автоматизації управління доступом через смарт-контракти.

Огляд існуючих методів захисту *IoT*-систем виявив, що традиційні підходи, такі як *TLS*/*SSL*, *PKI*, брандмауери та *IDS*/*IPS*, забезпечують базовий рівень захисту. Водночас, вони мають певні обмеження, включаючи залежність від централізованих систем, складність масштабування та недостатню стійкість до складних атак. Розробка моделей загроз і моделей порушника є ключовим інструментом для оцінки ризиків і розробки адекватних механізмів захисту, що враховують можливості зловмисника та вразливості системи.

В межах роботи:

– проведено аналіз загроз на *IoT*-системи та методи захисту від цих загроз;

– розроблено метод використання блокчейн-технології для підвищення рівнів цілісності даних та безпечності механізмів управління доступом до даних і ресурсів у *IoT*-системах;

– використано розроблений метод при управлінні пристроями в *IoT*-системі;

– проведено тестування розробленого методу на реальних системах.

Експериментальна частина роботи підтвердила ефективність розробленого методу. Практична реалізація з використанням смарт-контрактів

продемонструвала переваги в стійкості до атак, прозорості та автоматизації. Було виявлено виклики, пов’язані з часом підтвердження транзакцій і обмеженнями пропускної здатності блокчейн-мережі, що вказує на необхідність подальшої оптимізації.

Запропонований у роботі метод захисту даних, що базується на блокчейн-технології, продемонстрував значні переваги порівняно з традиційними підходами. Децентралізація, прозорість і автоматизація управління доступом через смарт-контракти дозволяють ефективно усувати ризики, пов’язані з централізованими точками відмови та атакою на інфраструктуру *IoT*. Експериментальне тестування підтвердило стійкість розробленої системи до атак і збереження цілісності даних, хоча використання блокчейн також виявило певні виклики, зокрема в контексті часу обробки транзакцій і витрат.

Порівняння ефективності блокчейн-технології з традиційними методами захисту виявило значний потенціал її застосування в *IoT*-системах, особливо для підвищення стійкості до атак і забезпечення можливості аудиту дій усіх учасників мережі. Проте для забезпечення максимальної ефективності та продуктивності пропонується подальше вдосконалення розробленого методу. Це включає інтеграцію сучасних масштабованих блокчейн-протоколів, таких як *Layer* 2 рішення, для прискорення обробки транзакцій і зменшення витрат, а також впровадження адаптивних механізмів виявлення і реагування на нові загрози.

Аналіз проблематики захисту *IoT*-систем показав, що *IoT*-системи потребують багаторівневого підходу до захисту, який враховує унікальні характеристики таких систем: розподіленість, обмежені ресурси пристроїв, велика кількість точок входу для атак. Моделі загроз і порушників дозволяють детально оцінити ризики та адаптувати стратегії захисту до конкретних умов роботи *IoT*.

Блокчейн як технологія для *IoT* виявився потужним інструментом для підвищення рівня безпеки. Його децентралізована архітектура усуває проблему єдиної точки відмови, забезпечує прозорість і надійність передачі даних.

Завдяки смарт-контрактам вдалося автоматизувати управління доступом і зберіганням журналів подій, що значно знижує операційні витрати та підвищує загальну ефективність системи.

Експериментальна перевірка методу продемонструвала, що блокчейн підвищує стійкість до атак (90-95% успіху відбиття атак) порівняно з традиційними методами (70-85%). Утім, блокчейн поступається у швидкості передачі даних (200-500 мс проти 20-100 мс у традиційних системах). Високі витрати на транзакції (0.1-0.5 *USD*) вказують на необхідність оптимізації.

Визначено наступні рекомендації щодо вдосконалення:

– використання сучасних масштабованих блокчейн-протоколів, таких як *Proof of Stake,* для зменшення витрат і часу підтвердження транзакцій.

– інтеграція адаптивних механізмів безпеки, що дозволяють автоматично виявляти нові загрози та реагувати на них у реальному часі.

– розробка приватних блокчейн-мереж для підвищення продуктивності та конфіденційності.

Тестування передачі даних через мобільний додаток виявило низку переваг і викликів використання блокчейн у реальних умовах. Було підтверджено безпеку та цілісність даних, однак час підтвердження транзакцій і обмеження пропускної здатності блокчейн-мережі вказують на необхідність оптимізації для більш ефективної роботи в масштабованих *IoT*-системах.

Порівняння з традиційними методами захисту, такими як *TLS*/*SSL, PKI*, брандмауери та *IDS*/*IPS*, продемонструвало, що блокчейн-технологія має значні переваги у стійкості до атак, прозорості та можливості аудиту. Водночас, традиційні методи забезпечують вищу швидкість аутентифікації та передачі даних, а також нижчі витрати на обробку транзакцій.

На основі отриманих результатів було запропоновано рекомендації для подальшого вдосконалення рішення, зокрема використання масштабованих блокчейн-протоколів (*Layer* 2), інтеграція механізмів адаптивної безпеки та впровадження приватних блокчейн-мереж для підвищення конфіденційності та продуктивності.

Таким чином, результати дослідження підтвердили доцільність використання блокчейн-технологій у *IoT*-системах, а також виявили напрямки для подальшого розвитку, спрямованого на підвищення ефективності та зниження витрат.

Розроблена система, інтегрована з мобільним додатком, дозволяє не лише ефективно захищати дані, але й спрощує взаємодію користувачів із *IoT*-пристроями завдяки інтуїтивному інтерфейсу та автоматизованим процесам. Смарт-контракти забезпечують надійність і прозорість усіх операцій, що підвищує довіру до системи та створює передумови для впровадження *IoT*-рішень у різних сферах, таких як розумні будинки, промислові процеси та транспорт.

Подальші перспективи розвитку включають:

– масштабування системи для роботи з великими *IoT*-мережами за рахунок впровадження приватних блокчейнів або комбінованих архітектур (гібридних рішень);

– інтеграцію із системами штучного інтелекту для автоматичного аналізу загроз та адаптації політик безпеки в реальному часі;

– використання енергоефективних алгоритмів консенсусу, таких як *Proof of Stake*, для зменшення витрат і покращення стійкості.

Впровадження розробленого методу у практичні *IoT*-рішення може значно покращити загальний рівень безпеки, знизити ризики витоку даних і забезпечити стабільну роботу систем у розподілених середовищах. Отримані результати можуть бути корисними для підприємств, які шукають інноваційні підходи до захисту своїх *IoT*-інфраструктур, а також для дослідників, зацікавлених у подальшому розвитку блокчейн-технологій у цій сфері.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Damor D. IoT Security Threats and Solutions* [Електронний ресурс]. Режим доступу: *https://www.einfochips.com/blog/iot-securitythreats-and-solutions/* (дата звернення: 10.10.2024).

2. *Du L., Li K., Liu Q., Wu Z., Zhang S. Dynamic multi-client searchable symmetric encryption with support for Boolean queries / Du L., Li K., Liu Q., Wu Z., Zhang S.//Inf. Sci.,* 2020. *vol*. 506. *pp*. 234–257.

3. *Sezer S. T1C: IoT Security: Threats, Security Challenges and IoT Security Research and Technology Trends / S. Sezer //* 31*st IEEE International System-on-Chip Conference* (*SOCC*). *Arlington, VA,* 2018. *P*. 1-2.

4. *Garg H. Securing IoT Devices and Securely Connecting the Dots Using REST API and Middleware / H. Garg, M. Dave //* 4*th International Conference on Internet of Things: Smart Innovation and Usages. Ghaziabad, India,* 2019. *P*. 1-6.

5. Мнушка О.В. Архітектура веб-орієнтованої *SCADA*-системи / О.В. Мнушка // Вісник Національного технічного університету "Харківський політехнічний інститут". Збірник наукових праць. Серія: Інформатика та моделювання. Харків: НТУ "ХПІ", 2018. № 24 (1300). С. 117-128.

6. *Attrapadung N., Imai H. Attribute-based encryption supporting direct/indirect revocation modes / Attrapadung N., Imai H.//IMACC. Cirencester, U.K.,* 2009. *pp*. 278–330.

7. *Digital Economy Report: UNCTAD/DER/*2024*, United Nations Conference on Trade and Development*. [Електронний ресурс]. Режим доступу: *https://unctad.org/publication/digital-economy-report-*2024/ (дата звернення: 11.10.2024). Назва з екрана.

8. *Dong C., Yang K., Qiu J., Chen Y. Outsourced revocable identity-based encryption from lattices / Dong C., Yang K., Qiu J., Chen Y.// Trans. Emerg. Telecommun. Technol. vol*. 30*, no*. 11. 2019. *pp*. 3529-3536.

9. *Akavia A., Goldwasser S., Vaikuntanathan V. Simultaneous hardcore bits and cryptography against memory attacks/ Akavia A., Goldwasser S., Vaikuntanathan V.// TCC. San Francisco, USA,* 2019. *pp*. 474–495.

10. *Bhatia T., Verma A. K. Data security in mobile cloud computing paradigm*: *A survey, taxonomy and open research issues / Bhatia T., Verma A. K.//J. Supercomput*. *vol*. 73, *no*. 6. 2017. *pp*. 2558–2631.

11. *Boldyreva A., Goyal V. Identity-based encryption with efficient revocation / Boldyreva A., Goyal V. //ACM CCS. Alexandria, USA,* 2008. *pp*. 417–426.

12. *Ateniese G., Burns R., Curtmola R., Herring J. Provable data possession at untrusted stores / Ateniese G., Burns R., Curtmola R., Herring J.// ACM-CSS. New York, USA,* 2017. *pp*. 598–609.

13. *Bethencourt J., Sahai A. Ciphertext-policy attribute-based encryption/ Bethencourt J., Sahai A.// IEEE Symp*. *Secur*. *–Berkeley, USA,* 2007. *pp*. 321–334.

14. *Chillotti I., Gama N., Georgieva M., Izabachène M. TFHE*: *Fast fully homomorphic encryption over the torus / Chillotti I., Gama N., Georgieva M., Izabachène M.//J. Cryptol., vol*. 33, *no*. 1. 2020. *pp*. 34–91.

15. *Cui B., Liu Z., Wang L. Key-aggregate searchable encryption* (*KASE*) *for group data sharing via cloud storage / Cui B., Liu Z., Wang L.//IEEE Trans. Comput., vol*. 65, *no*. 8. 2016. *pp*. 2374–2385.

16. *Deng H., Qin Z., Wu Q., Guan Z. Flexible attribute-based proxy re-encryption for efficient data sharing /Deng H., Qin Z., Wu Q., Guan Z. //Inf. Sci, vol*. 511. 2020. *pp*. 94–113.

17. *Baek J., Safavi-Naini R. Public key encryption with keyword search revisited / Baek J., Safavi-Naini R. // ICCSA. Perugia, Italy,* 2018. *pp*. 249–259.

18. *Bellare M., Boldyreva A., O'Neill A. Deterministic and efficiently searchable encryption / Bellare M., Boldyreva A., O'Neill A.//CRYPTO, vol*. 3025. *Santa Barbara, USA,* 2017. *pp*. 535–552.

19. *Berti F., Pereira O., Peters T. On leakage-resilient authenticated encryption with decryption leakages / Berti F., Pereira O., Peters T.//IACR Trans. Symmetric Cryptol*. *vol*. 2217, *no*. 3. 2017. *pp*. 271–293,

20. *Dodis Y., Kalai Y. T. On cryptography with auxiliary input / Dodis Y., Kalai Y. T. //* 41*st Annu. ACM Symp. Symp. Theory Comput.,* 2009. *pp*. 621–630.

21. *Boneh D., Boyen X. Secure identity-based encryption without random oracles / Boneh D., Boyen X. // CRYPTO, vol*. 3152. *Berlin, Germany*: *Springer,* 2023. *pp*. 443–459.

22. *Boneh D., Franklin M. Identity-based encryption from the Weil pairing / Boneh D., Franklin M. //CRYPTO, vol*. 3139. *Berlin, Germany*: *Springer,* 2021, *pp*. 213–229.

23. *Cheung L., Newport C. Provably secure ciphertext policy ABE / Cheung L., Newport C.//*14*th Proc. ACM CCS, Alexandria, VA, USA,* 2007. *pp*. 456–465.

24. *Dziembowski S. Intrusion-resilience via the bounded-storage model, in Proc. TCC, vol*. 3876. *Berlin, Germany*: *Springer,* 2006, *pp*. 207–224.

25. *Makkaoui K., Ezzati A., Beni-Hssane A., Ouhmad S. Fast Cloud–Paillier homomorphic schemes for protecting confidentiality of sensitive data in cloud computing / Makkaoui K., Ezzati A., Beni-Hssane A., Ouhmad S. // Humanized Comput. Vol*. 11, *no*. 6. 2020. *pp*. 2205–2214.

26. Тимчук Я.С. Метод захисту даних в іот-системах з використанням блокчейн / Я.С. Тимчук // Інтелектуальні технології лінгвістичного аналізу: матеріали міжн. наук.-техн. конф. [Київ], 23-24 жовт. 2024 р. / М-во освіти і науки України, НАУ Київ, 2024. С. 22.

27. Тимчук Я.С. Принципи захисту даних в *IoT*-системах / Я.С. Тимчук // Сучасні тенденції розвитку системного програмування: матеріали наук.-практ. конф. [Київ], 21-22 лист. 2024 р. / М-во освіти і науки України, НАУ Київ, 2024. С. 21-22.

28. Слободян О. Положення про кваліфікаційні роботи (проєкти) здобувачів вищої освіти Національного авіаційного університету. К.: НАУ, 2024. 62 с.

29. ДСТУ 3008-2015 "Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення". К.: ДП "УКРНДНЦ", 2015. 26 с.

Додаток А

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);
```

```
    // 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);
```

```
    }

    // 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];
    }
}
```