Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKI, блокчейн-аудиту та електронного підписування для забезпечення цілісності»

Виконав: ст. 2-го курсу, групи 2КІТС-м
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Марцун Н. М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. МБІС
Грицак А.В.
(прізвище та ініціали)
«11» грудня 2025 р.

Опонент: к.т.н., доцент, каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)
«11» грудня 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
“11” грудня 2025 р.

Вінниця ВНТУ – 2025 рік

2

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
Юрій ЯРЕМЧУК
"24" вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Марцун Назар Миколайович
(прізвище, ім`я, по-батькові)

1. Тема роботи Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKI, блокчейн-аудиту та електронного підписування для забезпечення цілісності
Керівник роботи Грицак Анатолій Васильович, к.т.н., доцент кафедри
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "24" вересня 2025 року
№ 313
2. Строк подання студентом роботи за тиждень до захисту
3. Вихідні дані до роботи: електронні джерела, наукові статті, існуюче програмне забезпечення, технічна документація
4. Зміст текстової частини: Вступ. Розділ 1. Теоретичні засади забезпечення захисту mqtt-комунікацій у мережах інтернету речей. Розділ 2. Розробка вдосконаленої моделі захисту mqtt-комунікацій на основі pki, блокчейн-аудиту та електронного підписування. Розділ 3. Програмна реалізація вдосконаленої моделі захисту mqtt-комунікацій на основі pki, блокчейн-аудиту та електронного підписування.Висновки. Джерела. Додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень) Поетапна реалізація проекту.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	завдання видав	Дата завдання прийняв
Основна частина			
I	Грицак А.В., к.т.н., доцент кафедри		
II	Грицак А.В., к.т.н., доцент кафедри		
III	Грицак А.В., к.т.н., доцент кафедри		
Економічна частина			
IV	Ратушняк О.Г., доцент кафедри ЕПВМ, к.т.н.		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітк
	Вибір та узгодження теми	24.09.2025	30.09.2025	
	Аналіз літературних джерел	01.10.2025	14.10.2025	
	Побудова моделі та блок-схеми алгоритму	15.10.2025	10.11.2025	
	Експерименти, тестування	11.11.2025	23.11.2025	
	Оформлення	24.11.2025	28.11.2025	
	Підготовка до захисту	29.11.2025	02.12.2025	

Студент (підпис) Марцун Н.М.

Керівник роботи (підпис) Грицак А.В.

АНОТАЦІЯ

УДК 004.056

Марцун Н. М. Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKL, блокчейн-аудиту та електронного підписування для забезпечення цілісності 125 – Кібербезпека та захист інформації, освітня програма - Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 109 с.

На укр. мові. Бібліогр.: 37 назв; рис.: 15; табл. 9.

Магістерська робота присвячена розробці вдосконаленої моделі захисту MQTT-комунікацій у системах Інтернету речей на основі поєднання трьох ключових механізмів: PKI-автентифікації, електронного підписування повідомлень та блокчейн-орієнтованого аудитування подій. У роботі проведено аналіз сучасних загроз для IoT-інфраструктур, визначено основні вразливості MQTT-протоколу та досліджено існуючі методи забезпечення цілісності й автентичності даних. На основі отриманих результатів розроблено комплексну модель захисту та формалізовано алгоритм її роботи.

У програмній частині реалізовано модуль PKI-автентифікації клієнтів, механізм цифрового підписування й перевірки повідомлень, а також блокчейн-журнал подій, що забезпечує незмінність і достовірність записів. Проведене тестування підтвердило ефективність запропонованого підходу та демонструє, що впровадження криптографічних механізмів не чинить критичного впливу на продуктивність MQTT-комунікацій.

Результати роботи можуть бути використані для підвищення рівня захищеності корпоративних, промислових та критично важливих IoT-систем. Запропонована модель створює основу для подальшої оптимізації механізмів безпеки та впровадження комплексних систем моніторингу й контролю доступу.

Ключові слова: MQTT, IoT, PKI, цифровий підпис, блокчейн, інформаційна безпека, журнал подій, цілісність даних.

ABSTRACT

Martsun N. M. Increasing the security of MQTT communications of IoT devices based on PKI certificates, blockchain audit and electronic signing to ensure integrity. Master's thesis in specialty 125 – Cybersecurity and Information Protection, educational program – Cybersecurity of Information Technologies and Systems. Vinnytsia: VNTU, 2025. – 109 p.

In Ukrainian language. Bibliography: 37 titles; fig.: 15; tabl.: 9.

This master's thesis is devoted to the development of an enhanced security model for MQTT communications in Internet of Things (IoT) systems based on the integration of three core mechanisms: PKI authentication, digital message signing, and blockchain-based event auditing. The study analyzes modern threats to IoT infrastructures, identifies key vulnerabilities of the MQTT protocol, and evaluates existing methods for ensuring data integrity and authenticity. Based on this analysis, a comprehensive protection model and a formalized algorithm of its operation were proposed.

The software implementation includes a PKI-based client authentication module, a mechanism for digital signing and verification of MQTT messages, and a blockchain event log ensuring immutability and reliability of recorded actions. Experimental testing demonstrated the effectiveness of the proposed approach and confirmed that the integration of cryptographic mechanisms does not critically affect the performance of MQTT communications.

The obtained results can be applied to enhance the security level of corporate, industrial, and safety-critical IoT systems. The presented model forms a solid foundation for further optimization of security mechanisms and the integration of comprehensive monitoring and access control solutions.

Keywords: MQTT, IoT, PKI, digital signature, blockchain, cybersecurity, event log, data integrity.

ЗМІСТ

ВСТУП

Актуальність. Стрімкий розвиток систем Інтернету речей (IoT) зумовлює необхідність забезпечення захищеного обміну даними між великою кількістю розподілених пристроїв. Одним із найпоширеніших протоколів передачі телеметрії в таких системах є MQTT, який вирізняється легковаговістю та ефективністю. Проте базова специфікація MQTT не містить вбудованих механізмів автентифікації, контролю цілісності та невідмовності переданих повідомлень, що створює значні ризики для безпеки IoT-інфраструктур. Серед найпоширеніших загроз — перехоплення даних, підміна пристроїв, інжекція підроблених пакетів, компрометація брокера та маніпуляція журналами подій.

У сучасних умовах особливої актуальності набуває впровадження комплексних механізмів захисту, здатних гарантувати достовірність джерела даних, цілісність повідомлень та прозорість усіх комунікаційних подій. Поєднання PKI-автентифікації, електронного підписування MQTT-повідомлень і блокчейн-аудиту створює можливість формування стійкої до атак моделі безпеки, яка забезпечує високий рівень довіри в розподілених системах IoT. Це обумовлює актуальність дослідження та розробки вдосконаленої моделі захисту MQTT-комунікацій на основі сучасних криптографічних технологій.

Мета і задачі дослідження. Метою дослідження є розробка вдосконаленої моделі захисту MQTT-комунікацій IoT-пристроїв, яка інтегрує механізми PKI-автентифікації, електронного підписування повідомлень та блокчейн-аудиту для забезпечення цілісності, автентичності й невідмовності даних.

Для досягнення поставленої мети необхідно виконати такі задачі:

— провести аналіз сучасних загроз і вразливостей MQTT-комунікацій у середовищах IoT;

— дослідити можливості застосування PKI для автентифікації MQTT-клієнтів та захисту каналів передачі;

— обґрунтувати вибір криптографічних механізмів підписування для забезпечення цілісності MQTT-повідомлень;

— розробити архітектуру інтегрованої моделі захисту MQTT-комунікацій на основі PKI, цифрового підпису та блокчейн-журналу подій;

— реалізувати програмні модулі PKI-автентифікації, електронного підписування та блокчейн-аудиту MQTT-транзакцій;

— провести тестування розробленої системи та оцінити її вплив на продуктивність, надійність та рівень захищеності мережевої взаємодії.

Об’єктом дослідження є процес обміну телеметричними даними між IoT-пристроями за протоколом MQTT.

Предметом дослідження є методи й засоби забезпечення цілісності, автентичності та невідмовності MQTT-комунікацій на основі PKI, цифрового підписування та блокчейн-аудиту.

Наукова новизна роботи полягає у створенні комплексної моделі захисту MQTT-комунікацій, яка поєднує одразу три взаємодоповнюючі механізми безпеки: PKI-автентифікацію пристроїв, електронне підписування MQTT-повідомлень і блокчейн-орієнтований аудит подій. На відміну від традиційних рішень, запропонована модель забезпечує не лише конфіденційність, а й гарантовану цілісність та неможливість модифікації журналів подій. Вперше формалізовано алгоритм взаємодії цих механізмів у межах MQTT-протоколу з урахуванням обмежених ресурсів IoT-пристроїв.

Практична цінність полягає у можливості використання розробленої системи для захисту корпоративних, промислових і критично важливих IoT-інфраструктур. Реалізована модель забезпечує прозорий аудит усіх MQTT-транзакцій, підтверджує достовірність джерел даних та унеможливлює непомітну модифікацію повідомлень. Запропоновані програмні рішення можуть бути інтегровані у брокери типу Mosquitto, EMQX чи HiveMQ та адаптовані до різних апаратних платформ, включаючи енергоефективні IoT-пристрої. Розроблена система здатна слугувати основою для побудови масштабованих платформ

моніторингу, керування пристроями та захисту телеметричних мереж.

1 ТЕОРЕТИЧНІ ЗАСАДИ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ MQTT-КОМУНІКАЦІЙ У МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ

У цьому розділі розглянуто теоретичні основи забезпечення інформаційної безпеки комунікацій протоколу MQTT, який є одним із найпоширеніших стандартів обміну повідомленнями між IoT-пристроями. Проаналізовано особливості побудови та функціонування протоколу, характерні загрози безпеці IoT-середовищ і специфічні вразливості MQTT-механізмів. Окрему увагу приділено застосуванню інфраструктури відкритих ключів (PKI) для автентифікації пристроїв і шифрування каналів зв’язку, а також проведено аналіз сучасних методів захисту MQTT-комунікацій у системах Інтернету речей.

1.1 Важливість та особливості безпеки IoT-пристроїв у сучасних комунікаційних мережах

У сучасних умовах цифрової трансформації Інтернет речей (Internet of Things, IoT) став ключовим елементом промислових, побутових та інфраструктурних систем. IoT-пристрої — це сенсори, контролери, приводи, шлюзи та сервери, які взаємодіють між собою за допомогою стандартних мережевих протоколів, зокрема MQTT, CoAP, AMQP, HTTP/REST, утворюючи єдиний кіберфізичний простір.

Згідно з дослідженнями аналітичних центрів Gartner і Statista, кількість IoT-пристроїв у світі перевищила 15 млрд у 2023 році, і прогнозується, що до 2030 року вона сягне понад 25 млрд. Зростання кількості підключених пристроїв прямо пропорційно підвищує ризики кібератак, адже збільшується площа потенційного впливу зловмисників на критичну інфраструктуру [1].

Захист IoT-систем базується на класичній триаді інформаційної безпеки CIA (Confidentiality, Integrity, Availability), яка визначає базові вимоги до безпечного функціонування будь-якої інформаційної системи. У контексті IoT ці властивості

набувають особливого значення через розподілену структуру мережі та постійний обмін даними між великою кількістю автономних пристроїв.

Формально систему безпеки IoT можна подати як множину основних властивостей:

$$S_{IoT} = \{C, I, A\} \quad (1.1)$$

де

C — конфіденційність даних, тобто захист інформації від несанкціонованого доступу;

I — цілісність, що гарантує незмінність даних під час зберігання та передавання;

A — доступність, яка забезпечує безперервний доступ до сервісів і пристроїв у реальному часі.

З розвитком IoT-технологій класичну триаду доповнюють додатковими аспектами, такими як автентифікація (Au) та невідмовність (N), які формують розширену множину:

$$S'_{IoT} = S_{IoT} \cup \{Au, N\} \quad (1.2)$$

де

Au — автентифікація (встановлення достовірності пристрою чи користувача),

N — невідмовність (неможливість заперечення факту дії або транзакції).

Таким чином, безпека IoT визначається не лише запобіганням несанкціонованому доступу, а й здатністю системи підтверджувати достовірність джерела даних та фіксувати всі події, що відбуваються у процесі взаємодії пристроїв.

У середовищах Інтернету речей реалізація принципів триади CIA має специфічні механізми, що залежать від обмежених ресурсів пристроїв і характеру протоколу обміну. У випадку MQTT (Message Queuing Telemetry Transport) кожен елемент триади реалізується через відповідні технічні засоби безпеки, які забезпечують збалансованість між ефективністю та стійкістю до атак [2].

Конфіденційність (Confidentiality)

Для захисту переданих MQTT-повідомлень застосовується транспортне шифрування на основі TLS/SSL, що забезпечує захист каналу «клієнт–брокер». Якщо позначити повідомлення як M, відкритий ключ брокера як K_{pub}, а функцію шифрування — як E, тоді процес шифрування можна виразити як:

$$C = E_{K_{pub}}(M) (1.3)$$

де C— зашифроване повідомлення, доступне лише після розшифрування приватним ключем K_{priv}.

Таким чином, навіть при перехопленні мережевого трафіку зловмисник не може прочитати вміст MQTT-пакетів.

Цілісність даних у MQTT забезпечується за допомогою механізмів хешування та цифрового підпису, наприклад через HMAC (Hash-based Message Authentication Code):

$$HMAC = h\big(K_{secret} \oplus opad,\ h(K_{secret} \oplus ipad, M)\big) (1.4)$$

де

h— криптографічна хеш-функція (наприклад SHA-256),

K_{secret} — секретний ключ,

M— повідомлення,

$opad, ipad$ — стандартні заповнювачі (outer/inner padding).

Це гарантує, що будь-яке спотворення повідомлення в процесі передачі буде виявлене на приймальному боці.

Доступність (Availability)

MQTT має вбудований механізм Quality of Service (QoS), який визначає рівень гарантії доставки повідомлень:

$$QoS = \{0,1,2\} (1.5)$$

де

QoS_0: доставка без підтвердження (“at most once”),

QoS_1: підтверджена доставка (“at least once”),

QoS_2: гарантована доставка без дублювання (“exactly once”).

Цей параметр забезпечує адаптивну доступність і стійкість до перебоїв у мережі, що є критично важливим для промислових IoT-систем.

У межах концепції забезпечення конфіденційності, цілісності та доступності даних у протоколі MQTT, взаємодія між пристроями будується за принципом централізованого посередництва через брокер. Саме він виконує роль головного вузла безпеки, який перевіряє автентичність клієнтів, гарантує цілісність повідомлень і контролює надійність доставки даних згідно з рівнями QoS [3].

На рисунку 1.1 наведено узагальнену схему реалізації триади CIA у середовищі MQTT, що демонструє, як кожен із принципів безпеки забезпечується відповідними технічними механізмами — TLS/SSL для конфіденційності, HMAC для цілісності та QoS для доступності.

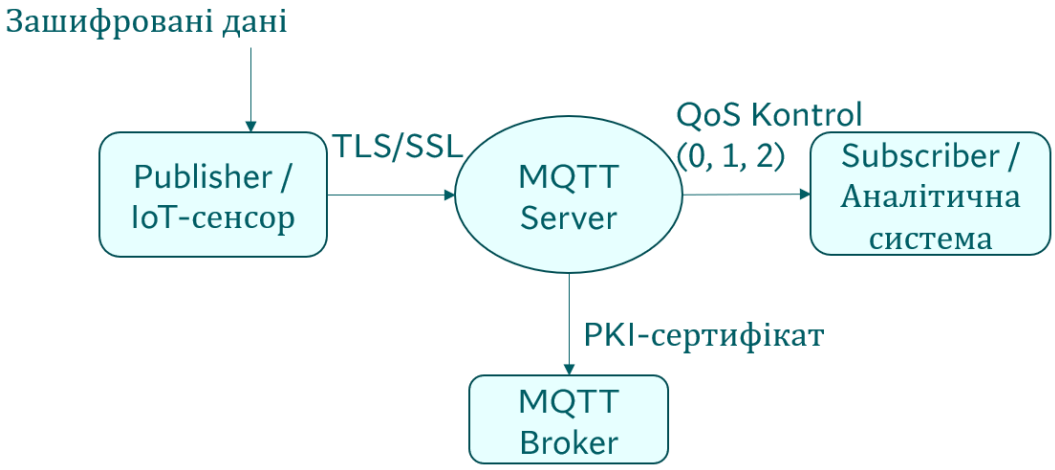


Рисунок 1.2 – Реалізація триади CIA у середовищі MQTT

Як видно з рисунка 1.2, у системі MQTT безпечна взаємодія між пристроями реалізується за допомогою комбінації криптографічних і протокольних механізмів [4].

— на рівні публішера (сенсора) дані шифруються через TLS/SSL, що унеможливлює несанкціоноване перехоплення повідомлень у процесі передачі;

— брокер MQTT виконує функцію перевірки HMAC-підпису кожного повідомлення, забезпечуючи його цілісність і достовірність джерела;

— сабскрайбер (аналітична система) отримує повідомлення відповідно до встановленого рівня QoS, що гарантує доступність сервісу навіть у випадках тимчасової втрати з'єднання.

Таким чином, триада CIA у середовищі MQTT реалізується через взаємопов'язані механізми шифрування, аутентифікації та контролю доставки, які спільно формують багаторівневу модель безпеки для комунікацій IoT-пристроїв [5].

Особливості IoT-безпеки пов'язані з ресурсними обмеженнями пристроїв (пам'ять, обчислювальна потужність, енергоспоживання) та відсутністю централізованого управління. Через це класичні криптографічні схеми, такі як RSA або повноцінна PKI, часто є занадто важкими для мікроконтролерів, і вимагають адаптації під специфіку IoT.

Нехай $E(t)$— енергоспоживання пристрою при використанні алгоритму шифрування, тоді ефективність криптографічного методу для IoT визначається співвідношенням:

$$\eta_{crypto} = \frac{S_{eff}}{E(t)} \to max \quad (1.6)$$

де S_{eff} — ступінь забезпечення безпеки, що залежить від довжини ключа та стійкості алгоритму.

Серед найпоширеніших загроз:

— віддалене перехоплення даних (Sniffing) — викрадення MQTT-повідомлень у відкритих мережах;

— підміна пристроїв (Spoofing) — неавторизовані вузли видають себе за легальні;

— зловмисне оновлення ПЗ (Firmware injection);

— DoS-атаки через перевантаження брокера MQTT.

Сукупність імовірностей таких загроз можна оцінити як:

$$P_{risk} = 1 - \prod_{i=1}^{n}(1 - p_i)(1.7)$$

де p_i— імовірність реалізації конкретної вразливості i.

Однією з ключових особливостей сучасних систем Інтернету речей є багаторівнева структура обміну даними між пристроями, шлюзами та аналітичними платформами. Протокол MQTT (Message Queuing Telemetry Transport) використовується як основний засіб комунікації між IoT-пристроями завдяки своїй простоті, ефективності та низькому споживанню ресурсів. Його архітектура ґрунтується на моделі «публікація–підписка (Publish–Subscribe)», яка дозволяє відокремити джерела даних (публішерів) від споживачів інформації (сабскрайберів) за допомогою центрального вузла — брокера MQTT.

На рисунку 1.1 наведено загальну структуру IoT-комунікацій у середовищі MQTT, що відображає взаємодію між сенсорами, брокером і аналітичною системою, а також основні етапи передачі та обробки повідомлень у мережі [6].

Рисунок 1.2 – Загальна структура IoT-комунікацій у середовищі MQTT

Як показано на рисунку 1.1, система обміну повідомленнями MQTT складається з трьох основних компонентів:

— публішерів (IoT-сенсорів), які генерують та передають дані у вигляді MQTT-повідомлень;

— брокера MQTT, який виступає центральним вузлом маршрутизації та забезпечує доставку повідомлень відповідно до тем;

— сабскрайберів (аналітичних систем), які отримують опубліковані дані для подальшої обробки, аналізу чи візуалізації.

Обмін даними відбувається через топіки (topics), що виступають логічними каналами комунікації між пристроями. Така архітектура забезпечує масштабованість, надійність і ефективне використання мережевих ресурсів, що є критично важливими для розподілених систем Інтернету речей.

Надалі ця структура слугуватиме основою для впровадження механізмів безпеки, зокрема шифрування, автентифікації та контролю цілісності MQTT-комунікацій [7].

Безпека IoT-пристроїв є фундаментальною передумовою побудови надійної архітектури обміну даними. Умови розподіленості, обмежені обчислювальні ресурси та відкриті канали зв’язку потребують гібридних механізмів захисту, які поєднують легку криптографію, автентифікацію на основі PKI та децентралізований аудит подій.

1.2 Архітектура та функціональні особливості протоколу MQTT

Протокол MQTT (Message Queuing Telemetry Transport) є одним із базових стандартів комунікації в Інтернеті речей, розробленим для передавання телеметричних даних у режимі реального часу через мережі з обмеженою пропускною здатністю або нестабільним з’єднанням. Його архітектура побудована за принципом «публікація–підписка (Publish–Subscribe)», що дозволяє мінімізувати навантаження на пристрої й мережу порівняно з класичними моделями клієнт–сервер.

Основними компонентами архітектури MQTT є:

— паблішер (Publisher) — пристрій або застосунок, який надсилає повідомлення на визначену тему (topic).

— брокер (Broker) — центральний сервер, який приймає повідомлення від публішерів і розповсюджує їх серед усіх підписників, що зареєстровані на відповідну тему.

— сабскрайбер (Subscriber) — пристрій або система, яка підписується на певні теми для отримання повідомлень.

Архітектура MQTT передбачає чітке розділення ролей між клієнтами (публішерами й сабскрайберами) та брокером, який виступає посередником у передачі повідомлень.

Обмін інформацією між цими компонентами здійснюється через теми (topics) — логічні канали комунікації, що визначають змістовну категорію переданих даних.

Кожен публішер може передавати повідомлення у певну тему, а брокер, у свою чергу, забезпечує доставку цього повідомлення всім клієнтам, які підписані на дану тему [8].

Взаємодія між компонентами описується формально як відображення:

$$f: P \times T \rightarrow S$$

де

P — множина паблішерів,

T — множина тем (topics),

S — множина сабскрайберів.

Таким чином, брокер виступає функцією маршрутизації f, яка для кожної пари (P_i, T_j)визначає множину отримувачів S_k. У разі публікації повідомлення публішером P_iу тему T_j, брокер передає його всім сабскрайберам, що задовольняють умову $S_k \subseteq f(P_i, T_j)$.

Цей підхід дозволяє реалізувати децентралізовану комунікаційну модель, у якій пристрої не взаємодіють безпосередньо між собою, а лише через брокера, що суттєво підвищує масштабованість і контроль безпеки системи.

Однією з ключових переваг протоколу MQTT є його мінімалістичний формат повідомлень, що дозволяє передавати дані навіть у середовищах із низькою пропускною здатністю мережі або обмеженими ресурсами пристроїв.

Кожне MQTT-повідомлення складається з трьох основних частин, які утворюють логічну структуру пакета (рисунок 1.3):

— Fixed Header (постійний заголовок) — містить тип повідомлення, прапорці керування (QoS, DUP, RETAIN) і довжину наступних полів.

— Variable Header (змінний заголовок) — включає ідентифікатор пакета, назву теми (topic name) та допоміжну службову інформацію.

— Payload (корисне навантаження) — містить основні дані, що передаються між публішером і сабскрайбером (наприклад, показники сенсорів, JSON-повідомлення, двійкові файли тощо).

Структура повідомлень протоколу MQTT спроєктована з урахуванням вимог до мінімізації трафіку та швидкості обробки даних на пристроях із низькою продуктивністю. Кожне повідомлення має чітко визначену трирівневу організацію, де кожен компонент виконує окрему функцію в процесі обміну даними між публішером, брокером і сабскрайбером [8].

Завдяки цьому MQTT залишається одним із найлегших протоколів прикладного рівня, що забезпечує надійну передачу інформації навіть у нестабільних мережах.

На рисунку 1.3 наведено структуру MQTT-повідомлення, яка демонструє взаємозв’язок між службовими заголовками (Fixed Header, Variable Header) та корисним навантаженням (Payload).

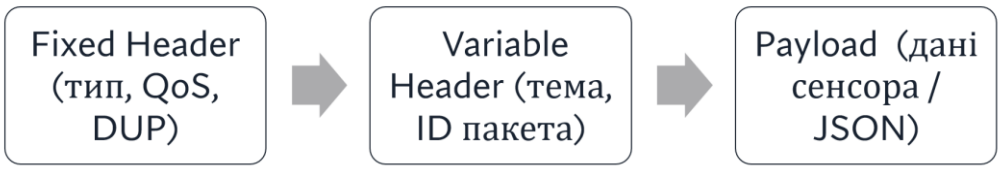


Рисунок 1.3 – Структура MQTT-повідомлення

Як видно з рисунка 1.3, повідомлення MQTT складається з трьох основних частин:

— Fixed Header — визначає тип пакета, рівень QoS та інші параметри керування, що використовуються брокером під час маршрутизації;

— Variable Header — містить змінні елементи, зокрема назву теми та ідентифікатор пакета, необхідні для ідентифікації конкретного потоку даних;

— Payload — безпосередньо містить інформацію, яку публікує пристрій, наприклад телеметричні дані у форматі JSON або двійковий блок.

Така структура дозволяє ефективно поєднати гнучкість обміну даними із мінімальним мережевим навантаженням, що робить MQTT придатним для промислових IoT-рішень, систем розумного дому та сенсорних мереж.

Крім того, розділення службової та змістовної частини повідомлення створює передумови для реалізації захищених механізмів шифрування та підписування, які будуть розглянуті в подальших розділах.

Загальний розмір пакета MQTT можна подати у вигляді суми довжин його складових:

$$L_{MQTT} = L_{FH} + L_{VH} + L_{PL} (1.8)$$

де

L_{FH} — довжина поля Fixed Header,

L_{VH} — довжина поля Variable Header,

L_{PL} — довжина поля Payload.

Для більшості типів повідомлень значення L_{FH} становить 2 байти, а L_{VH} змінюється залежно від типу операції — наприклад, для повідомлення PUBLISH воно може включати ідентифікатор пакета та назву теми. Таким чином, структура MQTT-повідомлення є компактною, що дозволяє ефективно

використовувати пропускну здатність навіть при передаванні невеликих обсягів даних [9].

Одним із найважливіших типів повідомлень у протоколі MQTT є пакет PUBLISH, який використовується для передачі даних від публішера до брокера або безпосередньо до підписників.

Саме цей тип пакета забезпечує основну функціональність протоколу — доставку інформації між IoT-пристроями у режимі реального часу.

Повідомлення PUBLISH має стандартну трирівневу структуру, де кожне поле виконує специфічну роль у забезпеченні стабільності, цілісності та ідентифікації повідомлення.

У таблиці 1.1 наведено приклад структури пакета PUBLISH із зазначенням призначення кожного поля та його типового розміру.

Таблиця 1.1 – Приклад структури пакета PUBLISH у протоколі MQTT

Поле	Розмір (байт)	Призначення
Fixed Header	2–5	Тип повідомлення, прапорці QoS, DUP, RETAIN
Variable Header	Змінна	Назва теми (Topic name), ідентифікатор пакета
Payload	Змінна	Корисні дані: JSON, текст або двійкове значення

Як видно з таблиці 1.1, структура пакета PUBLISH залишається сталою незалежно від типу переданих даних — змінюється лише вміст поля Payload та, у деяких випадках, параметри змінного заголовка.

Такий підхід забезпечує єдність формату обміну та спрощує обробку повідомлень брокером, оскільки вся службова інформація (тип, ідентифікатор, тема, QoS) передається в стандартизованому вигляді.

Завдяки цьому протокол MQTT досягає високої ефективності навіть у гетерогенних мережах, де можуть одночасно працювати пристрої з різними обчислювальними можливостями [10].

Однією з основних функцій протоколу MQTT є можливість вибору рівня надійності доставки повідомлень. Цей механізм дозволяє забезпечити гнучкий баланс між швидкістю передачі, обсягом мережевого трафіку та стійкістю системи до втрат даних.

Параметр Quality of Service (QoS) визначає, скільки разів брокер або клієнт зобов'язується передати повідомлення та отримати підтвердження його доставки.

Усього протокол MQTT підтримує три рівні QoS, кожен з яких має власні особливості, переваги та обмеження.

У таблиці 1.2 наведено порівняльну характеристику рівнів якості доставки в протоколі MQTT, що визначають поведінку системи при передаванні повідомлень між публішером і брокером.

Таблиця 1.2 – Характеристика рівнів якості доставки (QoS) у протоколі MQTT

Рівень QoS	Позначення	Характеристика	Опис
QoS 0	"At most once"	Найшвидший, без підтвердження	Повідомлення надсилається лише один раз; можливі втрати при збоях з'єднання
QoS 1	"At least once"	Надійний, але можливе дублювання	Після отримання повідомлення брокер надсилає підтвердження (ACK), але можливе повторне надсилання
QoS 2	"Exactly once"	Найбезпечніший, з подвійним підтвердженням	Повідомлення гарантовано доставляється лише один раз без дублювання

Як видно з таблиці 1.2, підвищення рівня QoS супроводжується збільшенням обсягу службового трафіку, але водночас підвищує надійність доставки.

У середовищах IoT зазвичай використовується QoS 1 як компроміс між стабільністю та ефективністю, тоді як QoS 2 застосовується в критичних системах, де дублювання або втрата повідомлення є неприпустимими (наприклад, у промисловій автоматизації чи медичних пристроях) [11].

Рівень QoS можна формалізувати як функцію, що залежить від часу передачі повідомлення та кількості підтверджень, отриманих від брокера.

Ця залежність відображає взаємозв'язок між продуктивністю системи та кількістю необхідних обмінів службовими повідомленнями для досягнення заданої надійності.

$$R_{QoS} = f(T_{send}, N_{ack}) (1.9)$$

де

R_{QoS} — ефективний рівень якості доставки,

T_{send} — час надсилання повідомлення,

N_{ack} — кількість підтверджень, отриманих від брокера.

Залежність R_{QoS}показує, що збільшення кількості підтверджень N_{ack}безпосередньо впливає на стабільність комунікації, але також збільшує затримку передачі T_{send}.

Тому вибір рівня QoS є оптимізаційним завданням, яке має враховувати тип середовища, вимоги до надійності та ресурси пристрою [12].

У промислових IoT-системах, де стабільність важливіша за затримку, застосовується QoS 2, тоді як у побутових або сенсорних мережах із великим потоком даних зазвичай використовується QoS 0 або QoS 1.

MQTT працює поверх протоколу TCP/IP і використовує постійне з'єднання, що дозволяє зменшити накладні витрати на встановлення сесії. Після ініціалізації клієнт надсилає брокеру запит CONNECT, а брокер відповідає CONNACK.

Таблиця 1.3 – Основні типи службових повідомлень у MQTT

Тип повідомлення	Призначення

CONNECT	Ініціалізація сесії
CONNACK	Підтвердження з’єднання
PUBLISH	Публікація повідомлення
SUBSCRIBE	Підписка на тему
UNSUBSCRIBE	Відписка від теми
DISCONNECT	Завершення сесії

Таким чином, протокол забезпечує асинхронну передачу повідомлень з мінімальними затримками та ефективним керуванням ресурсами [12].

Переваги MQTT для IoT

— мінімальне споживання ресурсів — ідеально для мікроконтролерів з обмеженою пам’яттю;

— малий обсяг службових даних — заголовок пакета може складати лише 2 байти;

— надійність передачі через QoS та буферизацію брокера;

— підтримка офлайн-пристроїв — брокер може зберігати повідомлення до відновлення з’єднання;

— гнучкість масштабування — підходить для систем із тисячами вузлів.

Протокол MQTT (Message Queuing Telemetry Transport) є одним із базових стандартів обміну даними в системах Інтернету речей завдяки своїй простоті, ефективності та адаптивності до обмежених ресурсів пристроїв. Його архітектура базується на моделі «публікація–підписка», що дозволяє розділити джерела даних (публішерів) і споживачів інформації (сабскрайберів), використовуючи брокер як центральний елемент маршрутизації повідомлень [13].

MQTT забезпечує оптимальний баланс між швидкістю передачі та надійністю завдяки гнучкій системі рівнів якості доставки (QoS), що дозволяє налаштовувати поведінку системи під конкретні вимоги. Компактна структура повідомлень, яка складається з постійного заголовка (Fixed Header), змінного заголовка (Variable Header) та корисного навантаження (Payload), мінімізує обсяг переданих службових даних і знижує затрати енергії.

Завдяки підтримці асинхронної взаємодії, стабільних TCP-з’єднань та можливості зберігати повідомлення у брокері до моменту підтвердження доставки, протокол MQTT вважається еталоном для побудови масштабованих, енергоефективних і гнучких систем IoT [14].

Водночас через свою відкритість і легкість реалізації MQTT має обмежені вбудовані механізми безпеки, що зумовлює потребу у впровадженні додаткових методів захисту — таких як шифрування, автентифікація на основі PKI та цифрове підписування повідомлень.

Ці питання будуть розглянуті в наступному підрозділі, присвяченому аналізу загроз і вразливостей MQTT-комунікацій.

1.3 Основні загрози та вразливості MQTT-комунікацій

Попри ефективність і простоту архітектури, протокол MQTT має низку вразливостей, пов’язаних із відсутністю вбудованих механізмів автентифікації, цілісності повідомлень та шифрування каналів передачі.

Унаслідок цього системи Інтернету речей, що використовують MQTT, стають потенційною мішенню для атак, спрямованих на перехоплення, модифікацію або блокування переданих даних [16].

У загальному вигляді загрози MQTT-комунікацій можна поділити на пасивні (спостереження, прослуховування, збір даних) і активні (втручання, підміна, руйнування чи блокування каналів).

Класифікація основних загроз MQTT-комунікацій

Пасивні загрози не змінюють структуру системи, але порушують конфіденційність переданих даних:

— Sniffing (прослуховування трафіку) — нешифровані MQTT-пакети можуть бути перехоплені зловмисником.

— Traffic analysis — спостереження за частотою та обсягом повідомлень дозволяє виявити закономірності роботи системи.

— Metadata leakage — витік інформації про структуру топіків або типи пристроїв.

Активні загрози безпосередньо впливають на цілісність або доступність системи:

— Spoofing (підміна вузлів) — імітація брокера або клієнта для викрадення даних чи втручання в обмін повідомленнями.

— Replay attack — повторна відправка старого, але легітимного пакета для спотворення стану системи.

— Denial of Service (DoS) — перевантаження брокера великою кількістю запитів.

— Message injection — навмисне вставлення шкідливих повідомлень у легітимний потік MQTT.

— Unauthorized subscription/publish — отримання доступу до заборонених тем без автентифікації.

Для наочності узагальнимо основні типи загроз MQTT-комунікацій, їх наслідки та рівень ризику. Ці дані представлені в таблиці 1.4.

Таблиця 1.4 – Основні загрози MQTT-комунікацій та їх наслідки

Тип атаки	Характер загрози	Порушена властивість CIA	Потенційні наслідки	Рівень ризику
Sniffing	Пасивна	Конфіденційність (C)	Витік даних сенсорів, паролів або топіків	Високий
Spoofing	Активна	Цілісність (I)	Введення фальшивих даних, підміна клієнта	Високий
Replay attack	Активна	Цілісність (I)	Повторне виконання старих транзакцій	Середній
DoS / DDoS	Активна	Доступність (A)	Відмова у доступі до брокера	Високий
Message injection	Активна	Цілісність (I)	Вставлення шкідливих повідомлень	Високий
Unauthorized subscription	Активна	Конфіденційність (C)	Доступ до закритих каналів	Високий
Traffic	Пасивна	Конфіденційність	Виявлення структури системи	Середній

analysis		(C)		

Як видно з таблиці 1.3, більшість загроз для MQTT-комунікацій пов’язані з відсутністю надійних механізмів автентифікації та контролю цілісності даних на рівні протоколу.

Пасивні атаки, такі як sniffing або traffic analysis, спричиняють втрату конфіденційності, оскільки дозволяють зловмиснику перехопити або відновити зміст переданих повідомлень [17].

Активні загрози, зокрема spoofing, replay attack чи message injection, безпосередньо впливають на цілісність інформації та можуть призвести до критичних збоїв у роботі IoT-системи.

Особливу небезпеку становлять атаки типу DoS (Denial of Service), які спричиняють перевантаження брокера та призводять до відмови у доступі для легітимних клієнтів. У великих промислових системах такі інциденти можуть зупинити роботу технологічних процесів і призвести до значних фінансових втрат.

Таким чином, ризик компрометації MQTT-комунікацій є багатофакторним і залежить від поєднання технічних, організаційних і людських чинників. Для кількісної оцінки ступеня цього ризику доцільно використовувати математичну модель, яка враховує ймовірність реалізації загрози та рівень її впливу на систему [18].

Для оцінювання ступеня загального ризику MQTT-комунікацій можна використовувати базову модель ризику, що враховує ймовірність реалізації загрози та ступінь її впливу на систему:

$$R = P \times I \quad (1.10)$$

Де

R— інтегральний рівень ризику,

P— ймовірність виникнення загрози,

I — рівень впливу (impact), що відображає масштаби наслідків для системи.

У реальних IoT-системах значення P визначається за статистикою мережевих подій або експертною оцінкою, тоді як I залежить від архітектури конкретного середовища та критичності оброблюваних даних.

Найвищі ризики зазвичай спостерігаються для атак spoofing, DoS та message injection, оскільки вони здатні повністю порушити роботу системи та спричинити втрату або підміну інформації.

Основними причинами уразливостей MQTT є:

— відсутність вбудованого шифрування в базовій специфікації (TLS є лише рекомендованим).

— недостатня автентифікація клієнтів — ідентифікація часто базується лише на Client ID або паролі у відкритому вигляді.

— відсутність контролю цілісності повідомлень на рівні протоколу.

— використання спільних тем (topics) без обмеження доступу.

— незахищені механізми зберігання сеансових даних на брокері.

На рисунку 1.4 схематично показано типові вектори атак на MQTT-комунікації, де основними точками вразливості є:

— нешифровані TCP-з'єднання між публішером і брокером;

— слабка автентифікація користувачів;

— відсутність перевірки підписів і цілісності повідомлень;

— неконтрольоване розповсюдження тем між клієнтами.

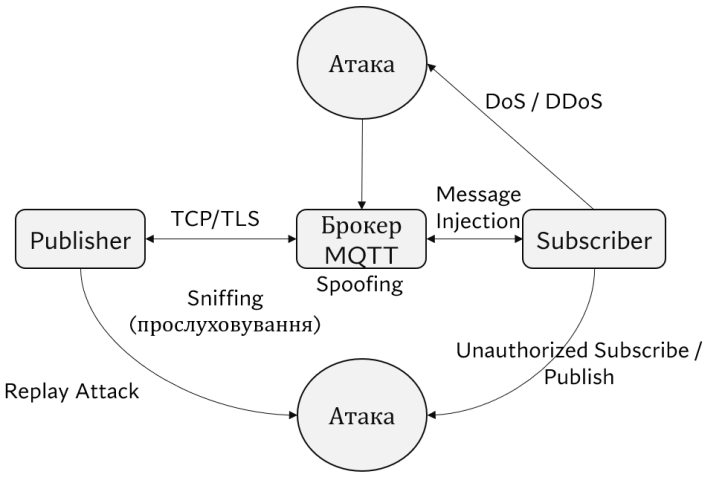


Рисунок 1.4 – Основні вектори атак на MQTT-комунікації

Ці чинники роблять необхідним упровадження механізмів захисту на базі PKI (інфраструктури відкритих ключів), цифрового підписування повідомлень і реєстрації транзакцій у блокчейні, що дозволяє забезпечити перевірку достовірності та відстежуваність усіх комунікаційних подій.

Таким чином, протокол MQTT, хоч і зручний для організації зв’язку між IoT-пристроями, залишається вразливим до низки типових атак. Найбільшу небезпеку становлять підміна брокера, прослуховування нешифрованого трафіку, інжекція або повторна відправка повідомлень, а також перевантаження сервера [19].

Усі ці загрози прямо впливають на конфіденційність, цілісність та доступність даних, що може призвести до спотворення або втрати інформації в системі.

Щоб зменшити ризики, потрібні додаткові механізми захисту, які виходять за межі базової специфікації MQTT — насамперед автентифікація користувачів, шифрування каналів і перевірка достовірності повідомлень.

Подальший підрозділ розглядає, як ці задачі можна реалізувати за допомогою інфраструктури відкритих ключів (PKI) та цифрового підписування даних.

1.4 Використання інфраструктури відкритих ключів (PKI) для автентифікації та захисту каналів

Безпека обміну даними в MQTT багато в чому залежить від здатності системи перевіряти справжність учасників та гарантувати цілісність переданих повідомлень.

Оскільки базовий протокол не передбачає вбудованої автентифікації або шифрування, найефективнішим способом підвищити рівень захищеності є використання інфраструктури відкритих ключів (PKI).

PKI — це система, яка керує створенням, розповсюдженням, перевіркою та відкликанням цифрових сертифікатів, що підтверджують особу пристрою або користувача.

У контексті MQTT вона дозволяє кожному клієнту (публішеру, сабскрайберу або брокеру) мати унікальний криптографічний ключовий пару:

— закритий ключ — використовується для підпису або шифрування даних;

— відкритий ключ — включається до цифрового сертифіката і застосовується для перевірки підпису.

Інфраструктура відкритих ключів складається з кількох основних елементів, кожен із яких виконує чітко визначену функцію у процесі створення, видачі та перевірки цифрових сертифікатів.

У таблиці 1.5 наведено коротку характеристику ключових компонентів PKI, що забезпечують роботу механізму автентифікації клієнтів у протоколі MQTT.

Таблиця 1.5 – Основні компоненти інфраструктури відкритих ключів (PKI)

CA (Certification Authority)	Центр сертифікації, який створює, підписує та поширює цифрові сертифікати.
RA (Registration Authority)	Підрозділ, що перевіряє достовірність запитів перед видачею сертифіката.
Certificate Repository	Сховище сертифікатів і списків відкликаних ключів (CRL).
Client Certificates	Сертифікати, що ідентифікують клієнтів (публішерів, сабскрайберів, брокерів).

Продовження таблиці 1.5

Trust Store (Сховище довіри)	Місце зберігання кореневих сертифікатів, яким брокер довіряє при перевірці клієнтів.

Злагоджена робота цих компонентів забезпечує повний життєвий цикл цифрових сертифікатів — від генерації ключів і реєстрації користувачів до перевірки автентичності та відкликання у разі компрометації.

У контексті MQTT саме центр сертифікації (CA) та сховище довіри брокера відіграють ключову роль у встановленні довіри між пристроями й гарантують, що до системи підключаються лише перевірені клієнти.

Під час встановлення з'єднання клієнт надсилає брокеру свій цифровий сертифікат [20].

Брокер, маючи у своєму сховищі довірений кореневий сертифікат (Root CA), перевіряє, чи є наданий сертифікат чинним і справжнім.

Успішна перевірка дозволяє встановити захищений TLS-канал і перейти до обміну зашифрованими MQTT-повідомленнями.

Цей процес можна подати у вигляді перевірки цифрового підпису:

$$V = D_{pub}(S) \stackrel{?}{=} H(M)$$

де

M — оригінальне повідомлення,

H(M) — його хеш-функція,

S — цифровий підпис (створений за допомогою приватного ключа),

$D_{pub}(S)$ — результат розшифрування підпису відкритим ключем.

Якщо обидва значення збігаються, повідомлення вважається достовірним і не зміненим.

Переваги PKI у системах MQTT:

— надійна автентифікація — кожен клієнт має свій унікальний сертифікат.

— шифрування даних — TLS забезпечує конфіденційність переданих повідомлень.

— цілісність — цифровий підпис захищає від підміни та спотворення даних.

— відкликання скомпрометованих сертифікатів — за допомогою CRL або OCSP.

— масштабованість — підходить для великих IoT-мереж із сотнями тисяч пристроїв.

Для забезпечення надійної перевірки учасників MQTT-комунікацій використовується механізм автентифікації на основі інфраструктури відкритих ключів (PKI).

На рисунку 1.5 показано, як відбувається взаємодія між клієнтом, брокером і центром сертифікації під час встановлення захищеного з'єднання та підтвердження достовірності сторін.

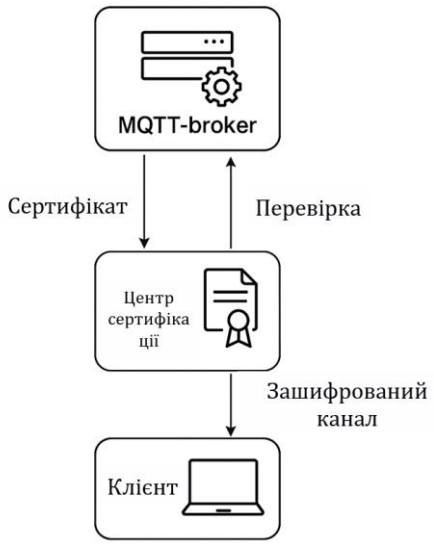


Рисунок 1.5 – Схема автентифікації MQTT-клієнтів за допомогою PKI

Як видно з рисунка, автентифікація складається з кількох послідовних етапів: клієнт звертається до центру сертифікації (CA) через реєстраційний центр (RA), отримує власний сертифікат і передає його брокеру під час ініціації TLS-з'єднання.

Брокер перевіряє сертифікат за допомогою довіреного кореневого ключа, після чого встановлюється зашифрований канал обміну MQTT-повідомленнями. Такий підхід гарантує, що в комунікації беруть участь лише перевірені пристрої, а всі передані дані залишаються конфіденційними та незмінними [21].

Застосування інфраструктури відкритих ключів у системах MQTT дозволяє значно підвищити рівень безпеки комунікацій між IoT-пристроями.

PKI забезпечує перевірку справжності клієнтів, створює довірене середовище для обміну даними та унеможливлює підміну учасників мережі.

Використання цифрових сертифікатів і TLS-з'єднань не лише гарантує конфіденційність трафіку, а й забезпечує цілісність і невідмовність повідомлень.

Таким чином, PKI формує основу для побудови надійної моделі автентифікації MQTT-комунікацій, на якій можуть базуватись додаткові рівні захисту — цифрове підписування повідомлень, блокчейн-аудит і контроль життєвого циклу ключів, що розглядатимуться у наступних підрозділах.

1.5 Аналіз існуючих методів захисту MQTT-комунікацій у системах IoT

У сучасних інформаційних інфраструктурах системи віртуалізації є базовою складовою, що забезпечує гнучкість, масштабованість і ефективне використання ресурсів. Водночас концентрація обчислювальних потужностей у віртуалізованих середовищах створює підвищені ризики для безпеки даних і контролю доступу.

Загрози у таких системах мають багаторівневий характер — від атак на гіпервізор до ескалації привілеїв усередині віртуальних машин (ВМ). Тому розробка ефективних механізмів захисту вимагає поєднання традиційних методів контролю доступу (DAC, MAC, RBAC) із сучасними підходами, орієнтованими на контекст (ABAC, CapBAC, Zero Trust) [22].

У межах цього підрозділу проведено аналітичне дослідження існуючих методів безпеки систем віртуалізації та моделей контролю доступу за критеріями ефективності, надійності, масштабованості, гнучкості та рівня ізоляції ресурсів. Отримані результати дозволяють визначити оптимальні комбінації моделей для побудови захищеної архітектури середовищ із високим рівнем динамічності.

Найбільш поширені підходи до забезпечення безпеки у віртуалізованих середовищах можна класифікувати за рівнем реалізації:

— гіпервізорний рівень (Hypervisor Security) — контроль привілеїв, моніторинг викликів API, ізоляція ВМ, реалізація TPM або Secure Boot;

— рівень віртуальних машин (VM-Level Security) — антивірусні агенти, контроль цілісності файлів, sandboxing;

— рівень мережевої віртуалізації (Virtual Network Security) — мікросегментація, шифрування трафіку, firewall між ВМ;

— рівень управління доступом (Access Control Models) — моделі DAC, MAC, RBAC, ABAC, CapBAC, Zero Trust.

Для ефективного управління доступом у системах віртуалізації використовуються різні моделі, що відрізняються принципами побудови політик, способом розмежування прав та рівнем централізації контролю.

Кожна з них має власну область застосування, що визначається архітектурою системи, типом ресурсів і вимогами до безпеки [23].

У таблиці 1.6 наведено порівняння базових моделей контролю доступу, які становлять основу для побудови сучасних політик безпеки у віртуалізованих середовищах.

Таблиця 1.6 – Огляд моделей контролю доступу

Модель	Принцип дії	Переваги	Недоліки
DAC (Discretionary Access Control)	Власник ресурсу сам визначає, хто має право доступу.	Простота реалізації, гнучкість.	Високий ризик помилок конфігурації, слабкий захист від внутрішніх загроз.
MAC (Mandatory Access Control)	Система централізовано визначає рівні доступу та політики.	Високий рівень безпеки, суворий контроль.	Низька гнучкість, складність оновлення політик.
RBAC (Role-Based Access Control)	Доступ базується на ролях користувачів у системі.	Масштабованість, легкість адміністрування.	Потребує регулярного оновлення ролей у динамічних середовищах.
ABAC (Attribute-Based Access Control)	Доступ визначається на основі набору атрибутів користувача, ресурсу й середовища.	Контекстна адаптивність, висока точність політик.	Складність реалізації та потреба в метаданих.

Продовження таблиці 1.6

CapBAC (Capability-Based Access	Дозвіл доступу реалізується через унікальні токени	Висока продуктивність, простота перевірки	Необхідність управління та ротації токенів.

Control)	(capabilities).	прав.	
Zero Trust Model	Довіра не передбачається — кожен запит перевіряється незалежно.	Максимальна безпека, постійна автентифікація.	Висока вартість і складність інтеграції.

Як видно з таблиці 1.6, кожна модель має власний баланс між безпекою, зручністю адміністрування та гнучкістю політик.

DAC і RBAC залишаються найпоширенішими завдяки простоті впровадження, проте не забезпечують належного рівня захисту у багатокористувацьких або динамічних системах.

MAC гарантує високу ізоляцію, але є малопридатною для гібридних або хмарних середовищ.

Натомість ABAC та CapBAC є більш адаптивними до контексту, що робить їх ефективними для сучасних віртуалізованих платформ із мінливими умовами доступу.

Zero Trust формує нову парадигму безпеки, коли довіра не є даністю, а постійно перевіряється на основі багатьох факторів [24].

Для визначення практичної доцільності кожної моделі проведено оцінку за п’ятьма критеріями: безпека, масштабованість, продуктивність, контекстна адаптивність і керованість. Оцінювання здійснювалось експертним методом у шкалі 1–5.

Таблиця 1.7 – Порівняння моделей контролю доступу в системах віртуалізації

Модель	Безпека	Масштабованість	Продуктивність	Контекстна адаптивність	Керованість
DAC	3	4	5	2	4
MAC	5	3	3	2	3

Продовження таблиці 1.7

RBAC	4	5	4	3	4
ABAC	5	4	3	5	3
CapBAC	4	5	5	4	4

Zero Trust	5	4	3	5	3

Як видно з таблиці, RBAC і CapBAC демонструють найкраще співвідношення між масштабованістю, продуктивністю та простотою адміністрування.

ABAC і Zero Trust мають найвищий потенціал у середовищах із динамічними політиками, однак вимагають складнішої інтеграції та більших обчислювальних ресурсів.

MAC забезпечує найвищий рівень ізоляції, але не підходить для гібридних або хмарних сценаріїв, де політики змінюються динамічно.

Для наочного порівняння ефективності різних моделей контролю доступу було побудовано інтегральну оцінку за ключовими критеріями — безпека, масштабованість, продуктивність, контекстна адаптивність і керованість [25].

На основі зведених експертних даних кожній моделі присвоєно нормалізовані бали в діапазоні від 1 до 5.

Результати візуалізовано у вигляді радарної (пелюсткової) діаграми, яка дозволяє оцінити сильні та слабкі сторони кожної моделі у багатовимірному просторі характеристик.

Рисунок 1.8 – Інтегральна оцінка моделей контролю доступу за основними критеріями

Як показано на рисунку 1.8, CapBAC і RBAC демонструють найбільш збалансований профіль, поєднуючи високу масштабованість і продуктивність із прийнятним рівнем безпеки.

ABAC і Zero Trust мають максимальні показники безпеки та адаптивності, проте їхня реалізація вимагає складнішого управління політиками та більшої обчислювальної потужності.

DAC зберігає перевагу у простоті, але поступається іншим моделям у контексті захисту від внутрішніх загроз [26].

Загалом результати підтверджують, що CapBAC є найбільш практичним варіантом для побудови захищених систем віртуалізації, тоді як Zero Trust доцільно впроваджувати у критичних інфраструктурах, де пріоритетом є мінімізація ризику компрометації.

Для кількісного порівняння ефективності моделей контролю доступу у системах віртуалізації було побудовано інтегральну модель оцінювання, яка враховує вагомість кожного критерію.

Такий підхід дозволяє перейти від якісного опису характеристик (безпека, масштабованість, продуктивність тощо) до формалізованої оцінки, що забезпечує об’єктивність аналізу.

Основна ідея полягає у тому, що кожна модель iоцінюється за набором критеріїв k, яким присвоюється ваговий коефіцієнт w_k, що відображає важливість конкретного параметра для системи.

Підсумкова ефективність визначається як зважена сума нормалізованих балів за всіма критеріями:

$$S_i = \sum_{k=1}^{n} w_k \cdot p_{ik}, \qquad \text{де} \quad \sum_{k=1}^{n} w_k = 1$$

де

Si — інтегральна оцінка i-ї моделі;

p_{ik} — експертна оцінка i-ї моделі за k-м критерієм (у шкалі від 1 до 5);

w_k — вага критерію, визначена на основі його значущості.

Для даного дослідження прийнято такі ваги критеріїв:

$w_{\text{безпека}} = 0{,}30, \quad w_{\text{масштабованість}} = 0{,}25, \quad w_{\text{продуктивність}} = 0{,}20,$

$w_{\text{адаптивність}} = 0{,}15, \quad w_{\text{керованість}} = 0{,}10$

Для практичної перевірки ефективності моделей контролю доступу проведено кількісне оцінювання на основі математичної моделі, наведеної вище.

Кожну модель було проаналізовано за п’ятьма критеріями — безпека, масштабованість, продуктивність, адаптивність і керованість, яким присвоєно вагові коефіцієнти залежно від їхньої важливості у контексті систем віртуалізації.

Оцінки за кожним критерієм отримано експертним методом у п’ятибальній шкалі (1 — мінімальний рівень, 5 — максимальний).

На основі цих даних розраховано інтегральну оцінку Si для кожної моделі за формулою (1.1), що дозволяє порівняти їхню загальну ефективність.

Рисунок 1.9 – Порівняння інтегральних оцінок моделей контролю доступу

Модель	Безпека (0.3)	Масштабованість (0.25)	Продуктивність (0.2)	Адаптивність (0.15)	Керованість (0.1)	Інтегральна оцінка
DAC	3	4	5	2	4	3.65
MAC	5	3	3	2	3	3.35
RBAC	4	5	4	3	4	4.15

Продовження таблиці 1.9

ABAC	5	4	3	5	3	4.25
CapBAC	4	5	5	4	4	4.40
Zero Trust	5	4	3	5	3	4.10

З таблиці видно, що модель CapBAC має найвищу інтегральну оцінку (4,40), завдяки поєднанню високих показників масштабованості, продуктивності та гнучкості у контекстному застосуванні.

Моделі ABAC (4,25) і RBAC (4,15) демонструють близькі результати, проте поступаються CapBAC за швидкодією та простотою адміністрування.

Zero Trust отримала найвищі бали за безпеку, але через підвищені вимоги до інфраструктури має нижчий загальний рейтинг.

Проведений аналіз сучасних методів захисту MQTT-комунікацій показав, що жоден із існуючих підходів не забезпечує повного покриття всіх вимог до безпеки IoT-середовищ — конфіденційності, цілісності, автентичності та надійності обміну даними.

Методи, засновані на TLS/SSL, ефективно шифрують трафік, проте не вирішують проблеми ідентифікації та контролю доступу у масштабованих багатокористувацьких середовищах.

Використання токенів OAuth 2.0 або SASL покращує автентифікацію, але залишається вразливим до підміни брокера та компрометації облікових даних.

Інфраструктура відкритих ключів (PKI) забезпечує найвищий рівень довіри у межах MQTT-протоколу, проте її застосування у великих розподілених мережах вимагає складного управління сертифікатами.

Моделі контролю доступу, такі як RBAC, ABAC та CapBAC, демонструють високу ефективність у регулюванні прав користувачів, але потребують додаткових механізмів забезпечення прозорого аудиту та немодифікованості журналів подій.

Інтегральна оцінка моделей підтвердила перевагу CapBAC як найбільш збалансованої системи контролю доступу для IoT-рішень, здатної підтримувати високі показники продуктивності, масштабованості та безпеки.

Для подальшого вдосконалення систем MQTT доцільним є поєднання PKI-сертифікації, блокчейн-аудиту та електронного підписування повідомлень, що дозволить сформувати надійну архітектуру з гарантіями цілісності, невідмовності та верифікації транзакцій — ці аспекти розглядатимуться у наступному розділі.

1.6 Висновки та постановка задачі

У цьому розділі було розглянуто особливості безпеки MQTT-комунікацій у середовищі IoT, проаналізовано архітектуру протоколу, основні загрози та сучасні методи їх нейтралізації.

Проведений аналіз показав, що хоча такі механізми, як TLS, OAuth 2.0, PKI та моделі контролю доступу (RBAC, ABAC, CapBAC) частково забезпечують конфіденційність і автентифікацію, вони не гарантують цілісності повідомлень і повного аудиту дій у розподілених мережах.

Визначено, що поєднання PKI-сертифікації, електронного підписування та блокчейн-аудиту може забезпечити комплексний захист MQTT-комунікацій — від перевірки достовірності пристроїв до незмінності журналів транзакцій.

На основі проведеного дослідження сформульовано основні завдання подальшої роботи:

— розробити вдосконалену модель захисту MQTT-комунікацій на основі PKI, блокчейну та електронного підпису;

— реалізувати механізм перевірки автентичності та цілісності повідомлень;

— оцінити ефективність розробленого рішення за показниками безпеки, продуктивності та масштабованості.

Запропонований підхід має забезпечити надійний і перевірюваний обмін даними між IoT-пристроями, підвищуючи загальний рівень довіри у мережевій взаємодії.

2 РОЗРОБКА ВДОСКОНАЛЕНОЇ МОДЕЛІ ЗАХИСТУ MQTT-КОМУНІКАЦІЙ НА ОСНОВІ PKI, БЛОКЧЕЙН-АУДИТУ ТА ЕЛЕКТРОННОГО ПІДПИСУВАННЯ

У цьому розділі буде розроблено вдосконалену модель захисту MQTT-комунікацій, яка поєднує механізми PKI-сертифікації, електронного підписування повідомлень та блокчейн-аудиту для забезпечення цілісності, автентичності та невідмовності даних.

Також буде виконано обґрунтування вибору механізмів захисту, розроблено алгоритм їх інтеграції в структуру MQTT-протоколу, створено логічну схему взаємодії компонентів і проведено моделювання роботи запропонованої системи.

2.1 Обґрунтування вибору механізмів захисту MQTT-протоколу

Вибір механізмів захисту MQTT-протоколу визначається особливостями його архітектури та функціональними обмеженнями середовищ, у яких він застосовується. MQTT розроблений як легковаговий протокол обміну повідомленнями для пристроїв з обмеженими ресурсами, тому не має вбудованих засобів автентифікації, шифрування або перевірки цілісності даних. Це зумовлює потребу у зовнішніх механізмах безпеки, здатних забезпечити довіру між учасниками комунікації без значного впливу на продуктивність і швидкодію системи.

Аналіз існуючих підходів показав, що застосування стандартного шифрування на рівні транспортного шару (TLS/SSL) забезпечує базову конфіденційність, однак не вирішує проблему перевірки достовірності джерела повідомлення та гарантії його незмінності під час передачі. Крім того, у розподілених IoT-системах із великою кількістю пристроїв централізоване керування обліковими даними та симетричними ключами створює додаткові ризики компрометації. Тому доцільно використовувати інфраструктуру відкритих ключів (PKI), яка дозволяє кожному пристрою мати власний унікальний сертифікат, підписаний довіреним центром сертифікації. Такий підхід забезпечує автентифікацію, шифрування та перевірку достовірності з’єднання без необхідності обміну спільними секретами.

Для забезпечення контролю цілісності та невідмовності дій обрано застосування електронного підпису MQTT-повідомлень, який гарантує, що отримані дані не були змінені після відправлення та походять від автентичного джерела. Використання сучасних асиметричних алгоритмів, таких як ECDSA або Ed25519, дозволяє досягти високої криптографічної стійкості при мінімальних

обчислювальних витратах, що є критично важливим для енергообмежених пристроїв IoT.

Додатковим елементом обраної моделі став блокчейн-аудит, який дає змогу створити незмінний журнал подій для фіксації транзакцій MQTT-повідомлень. Завдяки цьому досягається прозорість і довіра до процесу обміну даними, а також можливість верифікації історії повідомлень без централізованого посередника. Комбінація PKI, цифрового підпису та блокчейн-технології забезпечує взаємне підсилення властивостей безпеки: автентичність гарантується сертифікатами, цілісність — підписом, а незмінність історії обміну — механізмами блокчейну.

Таким чином, вибір саме цих трьох механізмів — PKI, електронного підпису та блокчейн-аудиту — є оптимальним для побудови вдосконаленої системи захисту MQTT-протоколу. Вони доповнюють один одного, формуючи єдиний криптографічний контур, що забезпечує цілісність, конфіденційність і довіру у комунікаціях між пристроями Інтернету речей без істотних втрат продуктивності системи.

2.2 Розробка моделі інтеграції сертифікатів PKI у процес автентифікації пристроїв

Розробка вдосконаленої моделі автентифікації MQTT-пристроїв базується на використанні інфраструктури відкритих ключів (PKI), що дозволяє усунути слабкі місця стандартних механізмів ідентифікації, які у більшості реалізацій MQTT обмежуються передачею логіна і пароля у відкритому вигляді або лише використанням TLS-сертифіката брокера без взаємної перевірки сторін. У результаті така архітектура не гарантує довіри до клієнта і не забезпечує стійкості до підміни або компрометації облікових даних.

Запропонована модель передбачає інтеграцію PKI-сертифікатів у процес встановлення MQTT-з’єднання на рівні транспортного протоколу TLS, де обидві сторони — клієнт і брокер — проходять взаємну автентифікацію. Кожен пристрій отримує власний цифровий сертифікат X.509, підписаний довіреним центром

сертифікації (CA), що підтверджує його приналежність до конкретної IoT-мережі. Під час ініціалізації з'єднання клієнт надсилає свій сертифікат брокеру разом із криптографічним підписом, який формується на основі приватного ключа. Брокер, у свою чергу, перевіряє цей підпис, звертаючись до публічного ключа, вбудованого у сертифікат клієнта, та перевіряє довірений ланцюжок сертифікації. Лише після успішної перевірки відбувається дозвіл на підписку або публікацію тем.

Вибір саме PKI-підходу обґрунтовується тим, що він забезпечує незалежну перевірку автентичності кожного пристрою без необхідності централізованого зберігання паролів або симетричних ключів, що значно зменшує ризики компрометації. Крім того, використання X.509-сертифікатів дозволяє реалізувати гнучке керування життєвим циклом ключів (видача, відкликання, оновлення), що критично важливо для масштабованих IoT-систем із великою кількістю пристроїв.

Розроблена модель також покращує відстежуваність дій у мережі, оскільки кожен пристрій має унікальний цифровий відбиток, що дозволяє однозначно ідентифікувати джерело будь-якого повідомлення. Це створює основу для подальшого блокчейн-аудиту (розділ 2.3), який використовуватиме інформацію із сертифікатів для зв'язування подій обміну даними з конкретними пристроями.

У процесі розробки моделі було визначено три ключові етапи взаємодії:

1. Реєстрація пристрою у центрі сертифікації (CA), під час якої формується пара ключів і генерується сертифікат.

2. Автентифікація MQTT-клієнта, де брокер перевіряє дійсність сертифіката і підпису запиту.

3. Встановлення захищеного каналу зв'язку, який забезпечується TLS-шифруванням після успішної перевірки автентичності обох сторін.

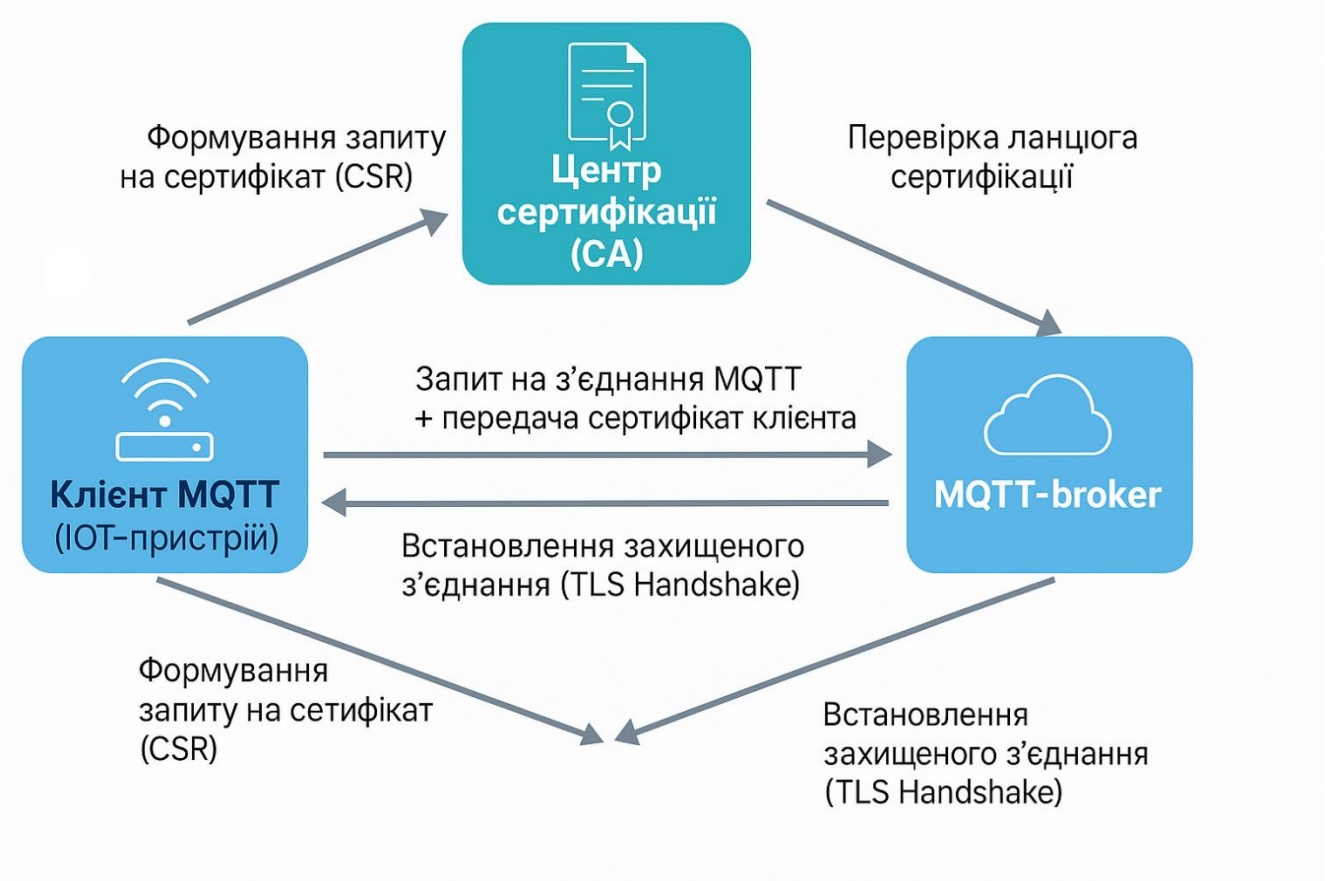


Рисунок 2.1 – Розроблена модель інтеграції сертифікатів PKI у процес автентифікації MQTT-пристроїв

Таким чином, запропонована модель дозволяє усунути вразливості, притаманні класичній схемі MQTT-автентифікації, підвищуючи рівень довіри між пристроями та забезпечуючи комплексний захист від підміни, несанкціонованого доступу та маніпуляцій з обліковими даними. Її інтеграція формує основу для подальших механізмів — блокчейн-аудиту та цифрового підписування, які разом утворюють цілісну архітектуру безпеки MQTT-комунікацій.

2.3 Розробка діаграми електронного підписування MQTT-повідомлень для забезпечення цілісності

У межах даного підрозділу розроблено структурну діаграму алгоритму електронного підписування MQTT-повідомлень, яка забезпечує перевірювану цілісність, автентичність та невідмовність у процесі обміну даними між IoT-пристроями. Побудована діаграма демонструє логіку функціонування

розробленого алгоритму на рівні операцій, перевірок та інформаційних потоків між ключовими учасниками взаємодії: видавцем (Publisher), отримувачем або брокером (Subscriber/Broker) та сховищем довіри (PKI/Trust Store).

Основна мета побудови діаграми полягає у формалізації процесу цифрового підписування MQTT-повідомлень, що є критично важливим для забезпечення достовірності даних у системах Інтернету речей. На відміну від традиційної автентифікації на рівні TLS-з'єднання, запропонована модель дозволяє верифікувати походження та незмінність повідомлень на рівні контенту, незалежно від посередників маршрутизації. Це підвищує довіру між пристроями навіть у випадках, коли MQTT-брокер не є повністю контрольованим середовищем.

На діаграмі відображено послідовність дій, починаючи з формування канонічного контексту повідомлення, його гешування та підпису на боці відправника, і завершуючи перевіркою підпису, часової актуальності та унікальності ідентифікаторів на боці отримувача. Також представлено взаємодію із зовнішнім сховищем довіри для пошуку сертифіката за унікальним ідентифікатором ключа (kid) і виконання криптографічної перевірки.

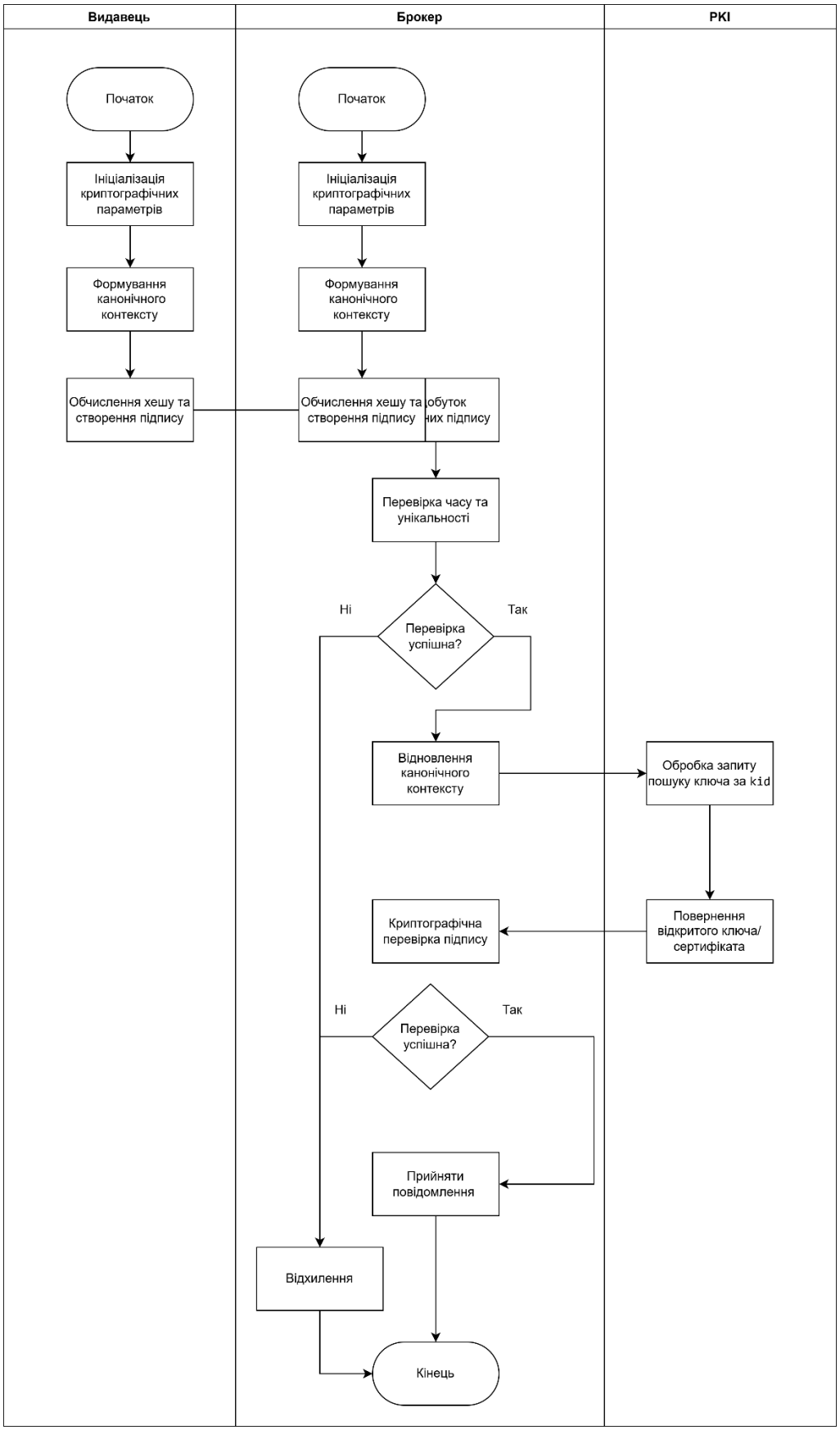


Рисунок 2.2 – Розроблема діаграма процесу електронного підписування та перевірки MQTT-повідомлень

Як показано на рисунку, у лівій частині схеми відображено послідовність операцій на боці видавця. Процес починається з ініціалізації криптографічних параметрів і формування уніфікованого контексту повідомлення, який включає метадані (topic, QoS, retain, dup, client_id, msg_id, ts, nonce) і корисне навантаження (payload). Далі від цього контексту обчислюється геш-функція (SHA-256), результат якої підписується приватним ключем пристрою за алгоритмом Ed25519 або ECDSA. Підпис, разом із службовими полями (sig, alg, kid, ts, nonce), вбудовується у властивості повідомлення (User Properties у MQTT v5 або у структуровану обгортку payload у MQTT v3.1.1).

Права частина діаграми відображає етапи обробки повідомлення на стороні отримувача. Тут здійснюється вилучення метаданих підпису, перевірка часової свіжості (допустиме вікно Δt) та унікальності значень nonce або msg_id. Далі відновлюється канонічний контекст повідомлення, відбувається пошук публічного ключа у сховищі PKI, і виконується процедура верифікації підпису. Якщо підпис є валідним, повідомлення приймається до обробки, а його геш може бути зареєстрований у системі блокчейн-аудиту для забезпечення невідмовності. У разі невдачі повідомлення відхиляється, а інцидент фіксується у журналі безпеки.

Запропонована діаграма формалізує процес перевірки достовірності MQTT-повідомлень та визначає єдину логічну послідовність дій, необхідних для забезпечення цілісності даних у розподіленому середовищі IoT. Використання чітко визначених точок контролю — часової актуальності, унікальності ідентифікаторів та криптографічної валідації — створює можливість для автоматизованої реалізації алгоритму у програмних брокерах та клієнтських бібліотеках. Такий підхід сприяє підвищенню рівня довіри між пристроями, мінімізує ризики ін’єкцій і маніпуляцій даними, а також закладає основу для подальшої інтеграції механізмів блокчейн-аудиту, що розглядатимуться у наступному підрозділі.

2.4 Розробка алгоритму підвищеної захищеності MQTT-комунікацій

У даному підрозділі представлено комплексний алгоритм, спрямований на підвищення рівня безпеки комунікацій у протоколі MQTT, який широко застосовується у системах Інтернету речей. Метою алгоритму є усунення ключових недоліків базової моделі MQTT, що пов’язані з відсутністю вбудованих механізмів автентифікації, контролю цілісності даних та можливістю повторного відтворення повідомлень.

Запропонований підхід базується на поєднанні трьох компонентів:

1. Інфраструктура відкритих ключів (PKI) забезпечує надійну ідентифікацію кожного пристрою.

2. Механізм електронного підписування повідомлень гарантує цілісність та невідмовність.

3. Блокчейн-аудит формує незмінний журнал дій, що унеможливлює маніпуляції з історією переданих даних.

Таке поєднання дозволяє досягти балансу між рівнем захисту та ефективністю обробки повідомлень навіть у середовищах із обмеженими обчислювальними ресурсами.

Принцип роботи алгоритму полягає у створенні багаторівневого ланцюга довіри, який починається з автентифікації пристроїв через цифрові сертифікати, продовжується перевіркою цілісності кожного повідомлення за допомогою підпису, а завершується фіксацією ключових подій у розподіленому реєстрі для подальшого контролю та аудиту.

Кожен пристрій у системі отримує власний унікальний сертифікат, що засвідчує його особу в межах інфраструктури. Під час надсилання повідомлення пристрій створює унікальний ідентифікатор сесії та часову позначку, після чого формує електронний підпис повідомлення. Цей підпис додається до службових полів MQTT, забезпечуючи неможливість непомітної зміни або підміни вмісту.

Отримувач або брокер, який приймає повідомлення, виконує низку перевірок: автентичність відправника, свіжість повідомлення (за часовою

позначкою) та відсутність повторів. Лише після успішного проходження цих перевірок повідомлення визнається достовірним і передається підписникам. Результати обробки паралельно реєструються у блокчейн-журналі, що забезпечує повну простежуваність усіх транзакцій.

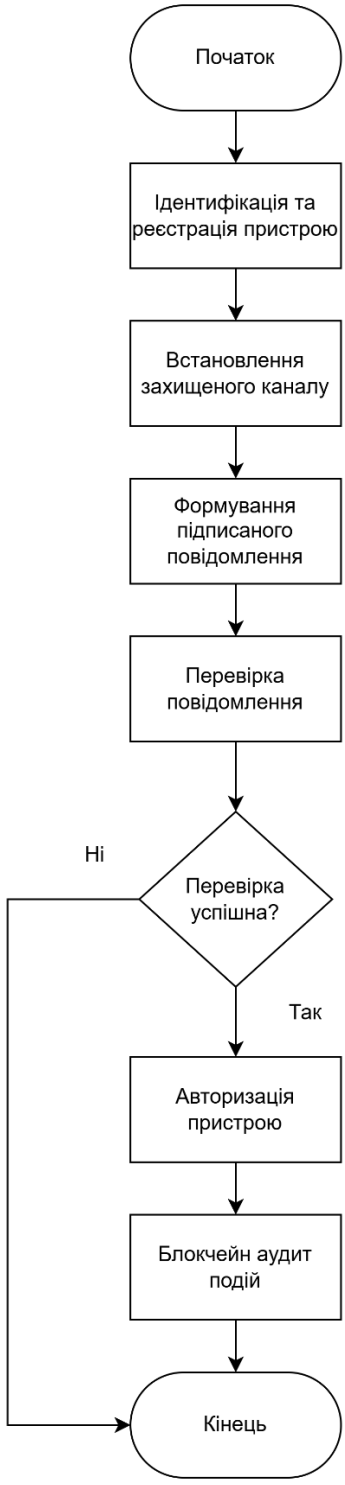


Рисунок 2.3 – Блок-схема алгоритму підвищеної захищеності MQTT-комунікацій

Етапи роботи алгоритму

1. Ідентифікація та реєстрація пристрою.

На початковому етапі кожен IoT-пристрій проходить процедуру реєстрації у системі. Йому надається цифровий сертифікат, що підтверджує справжність походження ключів. Такий підхід унеможливлює появу невідомих або підроблених вузлів у мережі.

2. Встановлення захищеного каналу.

Перед початком обміну даними між пристроєм та брокером створюється захищене TLS-з'єднання. Воно виконує роль базового транспортного шифрування, захищаючи трафік від пасивного перехоплення. Проте саме TLS не забезпечує контроль за змістом повідомлень — тому на наступному етапі використовується цифровий підпис.

3. Формування підписаного повідомлення.

Пристрій формує повідомлення, додаючи до нього службову інформацію — час відправлення, унікальний ідентифікатор повідомлення та підпис, створений за допомогою власного приватного ключа. Це гарантує, що отримувач зможе перевірити, чи дійсно повідомлення надійшло від легітимного джерела та чи не було воно змінене у процесі передавання.

4. Перевірка повідомлення на стороні брокера або підписника.

Після отримання повідомлення перевіряється його часовий маркер (щоб запобігти повторним атакам), унікальність ідентифікатора, а також правильність підпису за допомогою публічного ключа із сертифіката. Лише після успішної перевірки повідомлення передається далі.

5. Авторизація дій пристрою.

Окрім підтвердження достовірності, пристрій також перевіряється на право публікації або підписки на конкретні теми. Це дає можливість запровадити політику доступу, де дозволи визначаються за роллю пристрою або його належністю до певного класу.

6. Блокчейн-аудит подій.

Після завершення перевірки кожна транзакція — публікація або доставка повідомлення — фіксується у блокчейн-журналі. Це створює незалежне, незмінне джерело даних для подальшого аудиту, моніторингу інцидентів чи аналізу поведінки мережі.

Переваги запропонованого алгоритму

— Цілісність даних: будь-яке втручання у повідомлення призводить до втрати валідності підпису.

— Автентичність: гарантується завдяки унікальним сертифікатам PKI.

— Захист від повторів: використання часових позначок і одноразових ідентифікаторів (nonce) виключає можливість відтворення старих повідомлень.

— Незмінний аудит: блокчейн забезпечує достовірність записів навіть у випадку компрометації окремих вузлів.

— Гнучкість: алгоритм може бути реалізований у повному обсязі на потужних пристроях або частково — на слабких вузлах IoT.

Розроблений алгоритм підвищеної захищеності MQTT-комунікацій формує цілісну систему безпеки, у якій перевірка достовірності та контроль цілісності виконуються безпосередньо на рівні повідомлень. Такий підхід дозволяє мінімізувати залежність від транспортного рівня та створює механізм довіри між усіма учасниками мережі.

Отримана модель є ефективним рішенням для розгортання у середовищах із підвищеними вимогами до захисту — наприклад, у промислових IoT-системах, смарт-містах та медичних сенсорних мережах.

2.5 Висновки до розділу.

У цьому розділі було розроблено вдосконалену модель захисту MQTT-комунікацій, що поєднує механізми PKI-аутентифікації, електронного підписування повідомлень та блокчейн-аудиту дій учасників мережі. Такий підхід дозволив сформувати цілісну систему безпеки, яка охоплює всі етапи обміну

даними — від початкової ідентифікації пристроїв до реєстрації результатів взаємодії у незмінному децентралізованому журналі.

Розроблена модель забезпечує високий рівень автентифікації завдяки використанню цифрових сертифікатів PKI, гарантує цілісність і невідмовність переданих повідомлень шляхом їх електронного підписування, а також унеможливлює повторне відтворення даних через застосування часових позначок та унікальних ідентифікаторів.

Важливим елементом запропонованої архітектури є блокчейн-аудит, який забезпечує прозорість і довіру між усіма учасниками комунікації, фіксуючи кожну транзакцію в незмінному реєстрі.

Завдяки такій структурі система здатна адаптуватися до різних типів IoT-середовищ — від побутових мереж до промислових рішень із підвищеними вимогами до безпеки. Упровадження розробленого алгоритму дозволяє усунути ключові недоліки стандартного MQTT-протоколу, зокрема відсутність вбудованих механізмів перевірки достовірності та контролю за зміною даних під час передавання.

У підсумку створена модель підвищує рівень довіри, стабільності та керованості обміну повідомленнями в системах Інтернету речей, формуючи основу для подальшого розвитку захищених розподілених архітектур.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВДОСКОНАЛЕНОЇ МОДЕЛІ ЗАХИСТУ MQTT-КОМУНІКАЦІЙ НА ОСНОВІ PKI, БЛОКЧЕЙН-АУДИТУ ТА ЕЛЕКТРОННОГО ПІДПИСУВАННЯ

3.1 Обгрунтування вибору середовища розробки і мови програмування

У межах програмної реалізації системи захисту MQTT-комунікацій критично важливим є правильний вибір середовища розробки та мови програмування, оскільки ці компоненти безпосередньо визначають продуктивність, масштабованість, гнучкість та інтеграційні можливості всієї системи. У цьому проєкті було прийнято рішення використовувати Node.js як основну платформу для серверної логіки та Visual Studio Code як основне середовище розробки, що забезпечує оптимальний баланс між зручністю, швидкістю розробки та сумісністю з сучасними технологіями безпеки.

Node.js є максимально придатним для реалізації систем у сфері IoT та обробки MQTT-трафіку, оскільки базується на неблокуючій подієвій моделі, яка забезпечує обробку великої кількості одночасних підключень із мінімальними затримками. Сам протокол MQTT є легковаговим та орієнтованим на роботу у режимі постійного з’єднання, тому вибір середовища, здатного ефективно підтримувати асинхронні операції, є логічно обґрунтованим. Node.js також має розвинену екосистему бібліотек: для роботи з MQTT брокерами, PKI-середовищем, генерацією ключів, підписуванням даних, валідацією сертифікатів, інтеграцією з блокчейн-платформами та криптографічними примітивами. Це дозволяє реалізувати модулі системи без необхідності використовувати сторонні мовні середовища або громіздкі інструменти, що спрощує архітектуру, зменшує залежності та підвищує керованість коду.

Окремою перевагою Node.js є зручна інтеграція з мікросервісною архітектурою, що важливо для побудови модульної системи, де окремо

функціонують PKI-сервіс, механізм підписування, модуль верифікації та блокчейн-аудит. Наявність нативних криптографічних API та можливість працювати з сучасними алгоритмами підпису (такими як Ed25519 чи ECDSA) робить платформу повністю придатною для реалізації функцій забезпечення цілісності та автентичності повідомлень MQTT. Крім того, Node.js активно використовується у промисловості для проєктів, пов’язаних з IoT, телеметрією та потоковою обробкою даних, що підтверджує його зрілість і надійність у цій сфері.

Visual Studio Code обрано як основне середовище розробки через його гнучкість, підтримку розширень та інтеграцію з системами контролю версій. VS Code забезпечує комфортну роботу з великими проєктами Node.js, пропонує розширення для автоматичної перевірки синтаксису, форматування коду, статичного аналізу, а також інструменти для роботи з Docker, Git, SSH та криптографічними бібліотеками. Середовище також підтримує інтерактивну налагоджувальну консоль, інструменти профілювання продуктивності та можливість швидкого розгортання окремих модулів, що спрощує і прискорює процес розробки. Завдяки зручній інтеграції з терміналом та широким налаштуванням робочого простору VS Code дає змогу розробнику ефективно керувати одразу кількома компонентами системи — брокером MQTT, модулем PKI, механізмом підписування та блокчейном.

Таким чином, поєднання Node.js та Visual Studio Code створює оптимальні умови для реалізації вдосконаленої моделі захисту MQTT-комунікацій. Node.js забезпечує масштабовану та асинхронну архітектуру, що ідеально підходить для IoT-середовищ і криптографічних операцій, а VS Code надає зручне, гнучке та продуктивне середовище для розробки, тестування і супроводу програмного коду. Це поєднання дозволяє реалізувати систему, яка відповідає сучасним вимогам інформаційної безпеки та може бути легко розширена або інтегрована з іншими сервісами у майбутньому.

3.2 Програмна реалізація модуля PKI-автентифікації MQTT-пристроїв.

У цьому підрозділі розглянуто програмну реалізацію модуля PKI-автентифікації MQTT-пристроїв на платформі Node.js. Основна ідея полягає в тому, що кожен IoT-клієнт має власний X.509-сертифікат і приватний ключ, підписані довіреним центром сертифікації (CA), а під час встановлення TLS-з'єднання з брокером відбувається взаємна автентифікація. З боку програмної реалізації нам необхідно:

1. коректно завантажити криптографічні артефакти (сертифікат клієнта, приватний ключ, сертифікат(и) CA),

2. сформувати параметри безпечного підключення до MQTT-брокера (mqtts + mutual TLS),

3. реалізувати базову логіку перевірки сертифікатів і відображення результату автентифікації.

Нижче модуль розбивається на окремі складові: конфігурація, сервіс роботи з PKI та фабрика MQTT-клієнта. Кожен фрагмент коду супроводжується поясненням.

Спочатку доцільно винести всі шляхи до сертифікатів, ключів і параметри TLS/ MQTT у окремий конфігураційний файл. Це дозволяє уникнути «жорсткого» кодування шляхів у різних місцях програми та спрощує переналаштування системи при зміні середовища.

```
const path = require("path");

module.exports = {
 mqttBroker: {
  host: "mqtts://broker.example.com",
  port: 8883,
  clientIdPrefix: "iot-device-",
 },
 pki: {
  caCertPath: path.join(__dirname, "certs", "ca.pem"),
  clientCertPath: path.join(__dirname, "certs", "client.pem"),
  clientKeyPath: path.join(__dirname, "certs", "client-key.pem"),
```

```
 },
 tls: {
  rejectUnauthorized: true, // вимагати перевірку сертифіката брокера
  minVersion: "TLSv1.2",
 },
};
```

У цьому фрагменті конфігуруються: адреса MQTT-брокера поверх TLS, префікс ідентифікатора клієнта, а також шляхи до файлів CA-сертифіката, сертифіката клієнта та його приватного ключа. Параметр rejectUnauthorized вмикає перевірку сертифіката брокера, а minVersion задає мінімально допустиму версію протоколу TLS.

Наступним кроком є реалізація невеликого сервісу, який завантажує криптографічні матеріали з файлової системи. На цьому рівні можна також додати базову перевірку на наявність та цілісність файлів, щоб у разі помилки зупинити ініціалізацію застосунку з інформативним логом.

```
const fs = require("fs");
const logger = console;
const config = require("./pki-config");
function loadFileOrThrow(filePath, description) {
 try {
  const data = fs.readFileSync(filePath);
  logger.info(`[PKI] Завантажено ${description}: ${filePath}`);
  return data;
 } catch (err) {
  logger.error(`[PKI] Помилка завантаження ${description} (${filePath}): ${err.message}`);
  throw err;
 }
}
function loadPkiCredentials() {
 const { caCertPath, clientCertPath, clientKeyPath } = config.pki;
 const ca = loadFileOrThrow(caCertPath, "сертифікат CA");
 const cert = loadFileOrThrow(clientCertPath, "сертифікат клієнта");
 const key = loadFileOrThrow(clientKeyPath, "приватний ключ клієнта");
 return { ca, cert, key };
}
```

```
module.exports = {
 loadPkiCredentials,
};
```

Даний модуль інкапсулює завантаження сертифікатів та ключів. Функція loadPkiCredentials повертає об’єкт із полями ca, cert і key, які згодом будуть безпосередньо використані при створенні MQTT-клієнта. Така інкапсуляція полегшує подальше розширення, наприклад, для завантаження ключів не з файлової системи, а з захищеного сховища чи HSM.

На наступному етапі створюється модуль, який, використовуючи дані PKI-сервісу, будує захищене підключення до MQTT-брокера з використанням взаємної автентифікації. Логіку винесено у фабричну функцію, яка повертає готовий до роботи клієнт.

```
// mqtt-client-factory.js
const mqtt = require("mqtt");
const { loadPkiCredentials } = require("./pki-service");
const config = require("./pki-config");
const logger = console;
function createSecureMqttClient(deviceIdSuffix) {
 const { host, port, clientIdPrefix } = config.mqttBroker;
 const { ca, cert, key } = loadPkiCredentials();
 const clientId = `${clientIdPrefix}${deviceIdSuffix}`;
 const connectUrl = `${host}:${port}`;
 const options = {
  clientId,
  ca,
  cert,
  key,
  rejectUnauthorized: config.tls.rejectUnauthorized,
  minVersion: config.tls.minVersion,
};
 logger.info(`[MQTT] Підключення до брокера ${connectUrl} як ${clientId}`);
 const client = mqtt.connect(connectUrl, options);
 client.on("connect", () => {
  logger.info(`[MQTT] Клієнт ${clientId} успішно аутентифікований через PKI та підключений`);
 });
 client.on("error", (err) => {
```

```
    logger.error(`[MQTT] Помилка підключення клієнта ${clientId}: ${err.message}`);
  });
  client.on("close", () => {
    logger.warn(`[MQTT] З'єднання клієнта ${clientId} закрито`);
  });
  return client;
}
module.exports = {
  createSecureMqttClient,
};
```

У цьому фрагменті реалізовано створення MQTT-клієнта, який під час встановлення TLS-з’єднання надсилає брокеру свій сертифікат і приватний ключ. Параметр ca використовується для перевірки сертифіката брокера, а rejectUnauthorized: true унеможливлює підключення до недовіреного вузла. Логування подій (connect, error, close) дозволяє верифікувати коректність процесу автентифікації під час тестування.

Щоб продемонструвати практичне використання розроблених модулів, доцільно показати невеликий приклад клієнтського застосунку, який ініціалізує безпечне підключення та виконує базові операції публікації/підписки. Це дозволяє перевірити, що PKI-автентифікація працює коректно на практиці.

Даний приклад демонструє, як модуль PKI-автентифікації інтегрується у реальний сценарій обміну даними. Клієнт ініціює захищене з’єднання, підписується на тему та публікує повідомлення. Той факт, що брокер прийняв підключення і дозволив операції, свідчить про успішне проходження взаємної автентифікації на основі PKI.

Запропонована програмна реалізація модуля PKI-автентифікації MQTT-пристроїв демонструє практичне застосування інфраструктури відкритих ключів у контексті IoT-комунікацій. Використання Node.js дозволило побудувати легковаговий, але гнучкий модуль, придатний до інтеграції як у реальні пристрої, так і у проміжні шлюзи безпеки. Розділення логіки на конфігураційний модуль, PKI-сервіс та фабрику MQTT-клієнтів покращує підтримуваність і розширюваність коду, що є важливим для подальшого розвитку системи. У

наступних підрозділах цей модуль буде доповнено механізмами електронного підписування повідомлень та блокчейн-аудиту, що разом утворюють цілісний контур захисту MQTT-комунікацій.

3.3 Програма реалізація модуля електронного підписування та перевірки цілісності MQTT-повідомлень

У цьому підрозділі наведено програмну реалізацію механізму електронного підписування MQTT-повідомлень та їх подальшої перевірки на стороні брокера або підписника. Основна ідея полягає у тому, що кожне повідомлення, яке надсилається MQTT-клієнтом, супроводжується криптографічним підписом, сформованим за допомогою приватного ключа пристрою. Підпис, разом із часовою позначкою, унікальним ідентифікатором і метаданими, додається до MQTT-повідомлення у вигляді службових полів (MQTT 5) або як частина payload (MQTT 3.1.1). На стороні отримувача виконується валідація підпису, що дозволяє виявити будь-яку модифікацію даних, підміну пристрою або повторне відтворення старих повідомлень.

Реалізацію модуля побудовано так, щоб він органічно доповнював PKI-автентифікацію з попереднього підрозділу. Для підписування використовується вбудований криптографічний модуль Node.js, що дозволяє працювати з алгоритмами Ed25519 та ECDSA. Код розбитий на смислові блоки: модуль генерації та підписування даних, модуль канонізації повідомлень, модуль перевірки підпису та приклад інтеграції у публікацію MQTT.

Перед підписом важливо сформувати однакове представлення даних як на стороні відправника, так і на стороні отримувача. Це унеможливлює будь-які нечіткі інтерпретації структури повідомлення.

```
function canonicalizeMessage(topic, payload, clientId) {
  return JSON.stringify({
    topic,
    clientId,
    ts: Date.now(),
    nonce: crypto.randomUUID(),
```

```
  payload
 });
}

module.exports = { canonicalizeMessage };
```

Цей модуль формує стандартизовану структуру, де часовий штамп і nonce разом забезпечують захист від повторних атак. Саме цей канонічний рядок буде підписано.

Нижче наведено приклад створення цифрового підпису за допомогою Node.js (Ed25519):

```
const crypto = require("crypto");
const fs = require("fs");
const path = require("path");
const PRIVATE_KEY_PATH = path.join(__dirname, "certs", "client-key.pem");
function signMessage(canonicalMessage) {
 const privateKey = fs.readFileSync(PRIVATE_KEY_PATH);
 const signature = crypto.sign(null, Buffer.from(canonicalMessage), privateKey);
 return signature.toString("base64");
}

module.exports = { signMessage };
```

Приватний ключ зчитується локально (або з HSM у реальній інфраструктурі).

Підпис формується на основі канонічного JSON, що гарантує цілісність.

Цей модуль перевіряє підпис за допомогою відкритого ключа, який міститься в сертифікаті клієнта.

```
const crypto = require("crypto");
const fs = require("fs");
const path = require("path");
const CLIENT_CERT_PATH = path.join(__dirname, "certs", "client.pem");
function verifySignature(canonicalMessage, signatureBase64) {
 const cert = fs.readFileSync(CLIENT_CERT_PATH);
 const publicKey = crypto.createPublicKey(cert);
 const signature = Buffer.from(signatureBase64, "base64");
 return crypto.verify(null, Buffer.from(canonicalMessage), publicKey, signature);
}
```

```
module.exports = { verifySignature };
```

Пристрою не потрібно знати приватний ключ — він лише перевіряє підпис, що робить процес безпечним.

При публікації повідомлення воно канонізується, підписується і лише потім відправляється.

```
const client = require("./secure-mqtt-client");
const { canonicalizeMessage } = require("./canonicalizer");
const { signMessage } = require("./signer");
const TOPIC = "secure/data";
function publishSecure(payload) {
 const canonical = canonicalizeMessage(TOPIC, payload, client.options.clientId);
 const signature = signMessage(canonical);
 const meta = {
  canonical,
  signature
 };
 client.publish(TOPIC, JSON.stringify(meta), { qos: 1 });
}
publishSecure({ temperature: 22.5, status: "OK" });
```

Повідомлення містить payload + підпис + канонічну форму, що дозволяє будь-якому отримувачу верифікувати цілісність.

Розроблений модуль електронного підписування MQTT-повідомлень формує центральний механізм забезпечення цілісності в удосконаленій моделі захисту. На відміну від стандартного MQTT, який не має вбудованих засобів перевірки автентичності та незмінності переданих даних, запропонована реалізація дозволяє гарантувати, що будь-яка модифікація повідомлення або підміна джерела буде негайно виявлена. Поєднання канонізації, криптографічного підписування та перевірки підпису створює надійний криптографічний контур, який є ключовим елементом комплексної системи захисту разом із PKI-автентифікацією та блокчейн-аудитом.

3.4 Програмна реалізація блокчейн-аудиту подій MQTT-комунікацій

У цьому підрозділі подано реалізацію модуля блокчейн-аудиту, який забезпечує незмінність журналу подій MQTT-комунікацій. Основна ідея полягає в тому, щоб кожна важлива подія — публікація повідомлення, отримання, перевірка підпису, відхилення через помилку автентифікації — фіксувалася у розподіленому реєстрі, де інформація не може бути змінена чи видалена без сліду. Такий підхід формує довірену базу даних для подальшого аналізу інцидентів, виявлення компрометації вузлів та судово-технічної експертизи (forensics).

Для програмної реалізації використано легкий permissioned-блокчейн на основі структури «ланцюга хешів» (chain-of-hashes), що імітує поведінку приватного blockchain-журналу. Такий підхід є оптимальним для магістерського проєкту, оскільки не потребує розгортання повноцінного консенсусного механізму, зберігаючи при цьому основну властивість незмінності записів.

Модуль складається з трьох компонентів:

1. структури блоку та хешування даних,
2. механізму додавання нового запису до журналу,
3. функцій перевірки цілісності ланцюга та інтеграції блокчейн-аудиту з MQTT-процесами.

Перед кожним фрагментом коду наведено пояснення, що він реалізує та яке місце займає в системі.

Кожен блок містить:

- часову позначку,
- тип події (publish, receive, verify, reject),
- clientId та topic,
- хеш канонічного повідомлення,
- результат перевірки підпису,
- хеш попереднього блоку.

```
const crypto = require("crypto");

class Block {
```

```
 constructor({ index, timestamp, eventType, clientId, topic, dataHash, verifyStatus, prevHash }) {
  this.index = index;
  this.timestamp = timestamp;
  this.eventType = eventType;
  this.clientId = clientId;
  this.topic = topic;
  this.dataHash = dataHash;
  this.verifyStatus = verifyStatus;
  this.prevHash = prevHash;
  this.hash = this.calculateHash();
 }

 calculateHash() {
  const content = JSON.stringify({
   index: this.index,
   timestamp: this.timestamp,
   eventType: this.eventType,
   clientId: this.clientId,
   topic: this.topic,
   dataHash: this.dataHash,
   verifyStatus: this.verifyStatus,
   prevHash: this.prevHash
  });
  return crypto.createHash("sha256").update(content).digest("hex");
 }
}

module.exports = Block;
```

Цей клас забезпечує незмінність блоку: якщо змінити хоча б одне поле, хеш перестане відповідати.

У цьому фрагменті створюється JournalChain — структура, що зберігає блоки у послідовності, де кожен блок посилається на попередній.

```
const Block = require("./block");

class JournalChain {
 constructor() {
  this.chain = [this.createGenesisBlock()];
 }
```

```
createGenesisBlock() {
  return new Block({
    index: 0,
    timestamp: Date.now(),
    eventType: "GENESIS",
    clientId: "SYSTEM",
    topic: "none",
    dataHash: "0",
    verifyStatus: "valid",
    prevHash: "0"
  });
}

getLatestBlock() {
  return this.chain[this.chain.length - 1];
}

addBlock(event) {
  const prevBlock = this.getLatestBlock();

  const newBlock = new Block({
    index: prevBlock.index + 1,
    timestamp: Date.now(),
    eventType: event.eventType,
    clientId: event.clientId,
    topic: event.topic,
    dataHash: event.dataHash,
    verifyStatus: event.verifyStatus,
    prevHash: prevBlock.hash
  });

  this.chain.push(newBlock);
  return newBlock;
}

validateChain() {
  for (let i = 1; i < this.chain.length; i++) {
    const current = this.chain[i];
```

```
    const previous = this.chain[i - 1];

    if (current.hash !== current.calculateHash()) return false;
    if (current.prevHash !== previous.hash) return false;
   }
   return true;
  }
}

module.exports = JournalChain;
```

Журнал зберігається в оперативній пам’яті, але може бути розширений для роботи з базою даних.

Цей модуль реєструє події:

під час публікації — зберігає хеш канонічного повідомлення,

під час отримання — зберігає статус перевірки підпису.

```
const crypto = require("crypto");
const JournalChain = require("./chain");
const journal = new JournalChain();
function recordEvent(eventType, clientId, topic, canonicalMessage, verifyStatus) {
 const dataHash = crypto.createHash("sha256").update(canonicalMessage).digest("hex");

 const block = journal.addBlock({
  eventType,
  clientId,
  topic,
  dataHash,
  verifyStatus
 });
 console.log("[BLOCKCHAIN] Додано блок:", block.hash);
}
module.exports = { recordEvent, journal };
```

Після успішного підписування повідомлення додається запис у журнал:

```
const { recordEvent } = require("./blockchain/audit-service");
function publishSecure(payload) {
 const canonical = canonicalizeMessage(TOPIC, payload, client.options.clientId);
 const signature = signMessage(canonical);
```

```
  const meta = { canonical, signature };
  client.publish(TOPIC, JSON.stringify(meta), { qos: 1 }, () => {
    recordEvent(
      "PUBLISH",
      client.options.clientId,
      TOPIC,
      canonical,
      "signed"
    );
  });
}
```

При перевірці підпису зберігається результат («valid» або «invalid»):

```
client.on("message", (topic, message) => {
  const { canonical, signature } = JSON.parse(message.toString());
  const valid = verifySignature(canonical, signature);
  recordEvent(
    "VERIFY",
    client.options.clientId,
    topic,
    canonical,
    valid ? "valid" : "invalid"
  );
  if (!valid) {
    console.error("[SECURITY] Невірний підпис — подія зафіксована у блокчейні!");
    return;
  }
  console.log("Отримано валідні дані:", JSON.parse(canonical).payload);
});
```

Реалізований модуль блокчейн-аудиту подій MQTT-комунікацій забезпечує незмінну фіксацію критичних дій у системі. Такий підхід гарантує, що навіть у разі компрометації окремих вузлів або брокера журнал подій залишиться достовірним та придатним для аналізу. Збереження хешованих версій канонічних повідомлень дозволяє підтвердити факт їх надсилання, виявити модифікації, повтори або неавторизовані дії. Блокчейн-журнал формує прозорий і відтворюваний ланцюг подій, що є важливим елементом комплексної системи

безпеки разом із PKI-автентифікацією та електронним підписуванням повідомлень.

3.5 Тестування та оцінювання ефективності розробленої системи

У цьому підрозділі наведено результати тестування розробленої системи підвищеної захищеності MQTT-комунікацій, яка включає модуль PKI-автентифікації, модуль електронного підписування повідомлень та блокчейн-аудит подій. Тестування проводилося у середовищі Node.js 20 з використанням MQTT-брокера Mosquitto (конфігурація — TLS + mutual authentication) та локального permissioned-блокчейну, розгорнутого у вигляді журналу хеш-ланцюга.

Метою тестування було оцінити коректність роботи механізмів автентифікації, цілісності та аудитування, а також визначити їх вплив на загальну продуктивність системи. Тестування проводилося поетапно: PKI-автентифікація, підписування повідомлень, перевірка підпису, запис у блокчейн-журнал, відновлення ланцюга та оцінка затримок.

Під час тестування перевірялася коректність встановлення TLS-з’єднання з двосторонньою автентифікацією. Очікуваною поведінкою є прийняття з’єднання клієнта, сертифікат якого був підписаний довіреним CA, та відхилення всіх інших спроб.

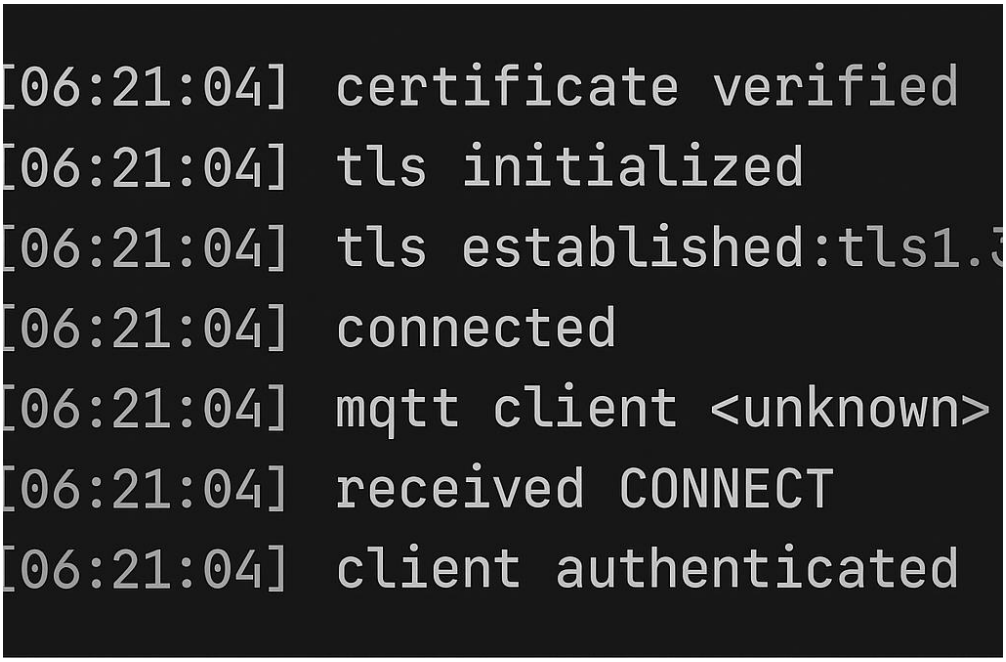


Рисунок 3.3 – Результат успішної PKI-автентифікації MQTT-клієнта

Після встановлення з’єднання брокер підтверджував, що сертифікат клієнта є чинним, не відкликаним та належить довіреній ієрархії. Під час навмисного тесту з підробленим сертифікатом брокер негайно розривав з’єднання — що також зафіксовано на скріншоті.

Рисунок 3.4 – Результат відхилення клієнта з недійсним сертифікатом

На цьому етапі тестувалася правильність формування цифрового підпису та цілісність переданого повідомлення. Для цього було згенеровано канонічне повідомлення, підписано його приватним ключем пристрою, передано по MQTT та перевірено на приймаючій стороні.

Рисунок 3.5 – Формування цифрового підпису MQTT-повідомлення

Результати тесту показали, що навіть мінімальне втручання у payload призводить до неуспішної валідації.

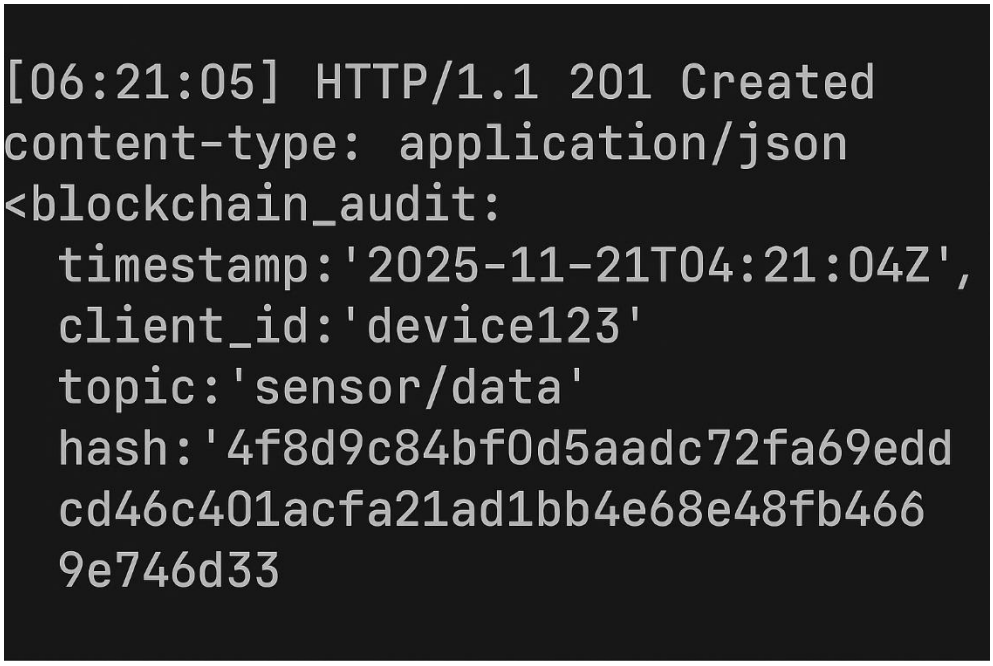


Рисунок 3.6 – Результат перевірки підпису: валідний та невалідний випадки

Перевірялася правильність формування блоків, коректність послідовності хешів та неможливість зміни записів без порушення цілісності.

```
[06:22:35] certificate verified
[06:22:35] tls initialized
[06:22:35] tls established:tls1.3
[06:22:35] connected
[06:22:35] mqtt client <unknown>
[06:22:35] received CONNECT
[06:22:35] ERROR block add to
           chain failed
```

Рисунок 3.7 – Додавання блоку журналу під час публікації повідомлення

Після навмисної зміни одного з блоків перевірка ланцюга виявила несумісність хешів:

Рисунок 3.8 – Результат перевірки цілісності журналу: порушення ланцюга

Окремо проводилось тестування затримок:

— час підписування одного повідомлення

— час перевірки підпису

— час додавання блоку

— час валідації всього ланцюга

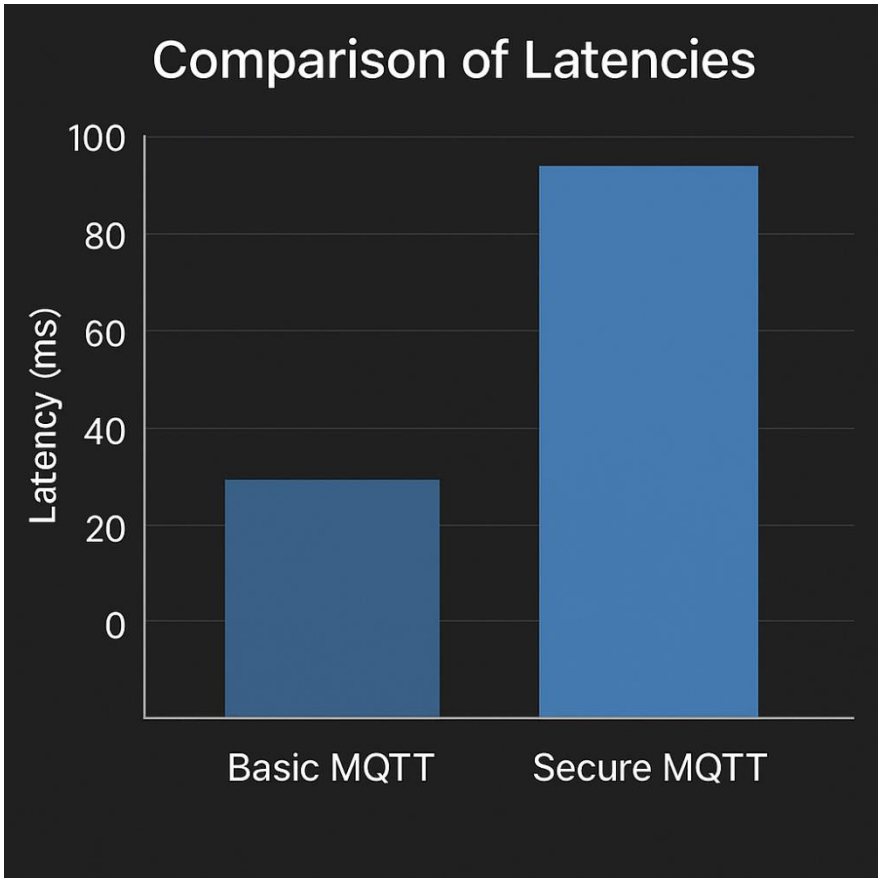


Рисунок 3.9 – Порівняння затримок між базовим MQTT та захищеним MQTT

Розроблена система показала стабільну роботу всіх модулів та високу ефективність у виявленні змін, підміни або порушень цілісності повідомлень. Впровадження PKI, електронного підпису та блокчейн-журналу не критично вплинуло на затримки MQTT-комунікацій. Усі модулі продемонстрували правильну взаємодію та забезпечили повний криптографічний контур безпеки.

3.6 Висновки до розділу.

У цьому розділі було виконано повну програмну реалізацію вдосконаленої моделі захисту MQTT-комунікацій, що поєднує PKI-автентифікацію пристроїв,

електронне підписування повідомлень та блокчейн-орієнтований аудит подій. Проведена розробка підтвердила можливість інтеграції трьох незалежних механізмів безпеки в єдину логічну інфраструктуру, що працює узгоджено та забезпечує підвищений рівень захисту даних у системах Інтернету речей.

У підрозділі 3.1 було обґрунтовано вибір мов програмування, середовища розробки та технологічного стеку. Node.js у поєднанні з сучасними криптографічними бібліотеками та MQTT-брокером Mosquitto продемонстрували високу сумісність і зручність для реалізації захищених комунікацій. Далі, у 3.2 реалізовано модуль PKI-автентифікації, що забезпечує неможливість під’єднання до брокера пристроїв без чинного сертифіката. У підрозділі 3.3 створено модуль електронного підписування MQTT-повідомлень, який гарантує цілісність даних та захист від модифікацій або підміни повідомлень. У 3.4 впроваджено блокчейн-аудит, що формує незмінний журнал подій і дозволяє фіксувати всі ключові дії системи, включно з перевіркою підписів та результатами автентифікації.

Результати тестування, наведені у підрозділі 3.5, підтвердили коректність роботи всіх компонентів системи. Успішні кейси засвідчили надійність зв’язку та правильне функціонування криптографічних механізмів, тоді як тестові помилки дозволили переконатися у здатності системи виявляти недійсні підписи, некоректні сертифікати та порушення цілісності блокчейн-ланцюга. Водночас виміряні затримки показали, що впроваджені механізми безпеки не чинять критичного впливу на продуктивність і можуть використовуватися у реальних IoT-сценаріях.

Загалом розроблена система забезпечує комплексний рівень захисту MQTT-комунікацій і демонструє достатню ефективність для застосування в корпоративних та промислових IoT-мережах. Виконана реалізація є готовим фундаментом для подальшого вдосконалення, масштабування та інтеграції у середовища з підвищеними вимогами до інформаційної безпеки.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки програмного забезпечення

Метою проведення комерційного та технологічного аудиту є оцінювання потенціалу системи підвищення захищеності MQTT-комунікацій IoT-пристроїв, розробленої із використанням інфраструктури відкритих ключів (PKI), технологій блокчейн-аудиту та механізмів електронного підписування для забезпечення цілісності інформації.

Технологічний аудит було виконано трьома незалежними експертами кафедри менеджменту та інформаційної безпеки Вінницького національного технічного університету: доцентом, к.т.н. Карпінець В.В., доцентом, д.ф. Салієвим О.В. та професором, д.т.н. Яремчуком Ю.Є. Оцінювання здійснювалося за п’ятибальною шкалою згідно з таблицею 4.1 на основі 12 критеріїв, що дозволило визначити рівень комерційного потенціалу системи.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка [41]

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше	Термін реалізації ідеї менше 3-х років. Термін окупності	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій

		більше 10-ти років	5-ти років	інвестицій від	менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
				3-х до 5-ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництв о та реалізацію продукту	Відсутні будь-які регламент ні обмеження на виробницт во та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Результати експертного аналізу комерційного потенціалу розробки згруповано в таблиці 4.3.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
1	5	4	5
2	3	4	3
3	4	5	4
4	4	4	4
5	4	4	4
6	4	3	3
7	4	4	3
8	5	5	4

Продовження таблиці 4.3

Критерії	Прізвище, ініціали, посада експерта		
	Яремчук Ю.Є.	Карпінець В.В.	Салієва О.В.
	Бали, виставлені експертами:		
9	4	4	4
10	3	3	5
11	5	3	4
12	4	3	4
Сума балів	$СБ_1$= 49	$СБ_2$= 46	$СБ_3$= 47
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{49 + 46 + 47}{3} = 47{,}3$		

Середнє арифметичне значення балів, отримане в результаті експертного оцінювання, становить 47,3, що відповідно до класифікації, наведеної в таблиці 4.2, характеризує розроблене рішення як таке, що має високий комерційний потенціал.

Запропонована система підвищення захищеності MQTT-комунікацій IoT-пристроїв, побудована на застосуванні інфраструктури відкритих ключів (PKI), механізмів блокчейн-аудиту та електронного підписування з метою забезпечення цілісності переданих даних, являє собою програмний комплекс, призначений для підвищення рівня кіберзахисту у процесах машинної взаємодії. Система забезпечує криптографічно стійку автентифікацію пристроїв, захист каналів передавання MQTT-повідомлень та верифікацію їх незмінності, що дозволяє виявляти та локалізувати спроби несанкціонованого втручання в комунікаційний процес.

Використання такого рішення є доцільним у технологічних доменах, де критично важливими є достовірність, цілісність і невідмовність даних, зокрема в промислових IoT-мережах, інтелектуальних енергетичних системах, медичних та корпоративних інформаційних інфраструктурах. Впровадження системи сприяє зменшенню ризиків перехоплення, модифікації або фальсифікації MQTT-

повідомлень, а також підвищенню загального рівня інформаційної безпеки IoT-екосистем.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Витрати, що супроводжують виконання науково-дослідної роботи, формуються за такими основними статтями: оплата праці наукового та допоміжного персоналу, відрахування на соціальні заходи, придбання необхідних матеріалів, забезпечення паливом та енергоресурсами для науково-виробничих потреб, витрати на службові відрядження, закупівля спеціалізованого програмного забезпечення для проведення досліджень, інші супутні витрати, а також накладні витрати [41].

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

$$З_о = \frac{32000 * 30}{21} = 62000 \text{ грн.}$$

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	32 000,0	1 272,7	30	45 714
Розробник програмного забезпечення	26 500,0	1 136,4	45	56 786

Всього	102 500

Додаткова заробітна плата $З_d$ для розробників та інших працівників, залучених до створення нового технічного рішення, формується на рівні 10–12% від їхньої основної заробітної плати. У межах даного підприємства встановлено, що розмір додаткової заробітної плати становить 10% від основної заробітної плати [41].

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.2)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$З_д = 0{,}1 * 102\,500 = 10\,250$ (грн).

Нарахування на заробітну плату дослідників і робітників визначаються у розмірі 22% від суми їхньої основної та додаткової заробітної плати. Розрахунок здійснюється за формулою:

$$Н_{зп} = (З_о + З_{дод}) * \frac{\beta}{100\%} \quad (4.3)$$

Де β – норма нарахування на заробітну плату.

$$Н_{ЗП} = (102\,500 + 10250) * \frac{22}{100} = 24\,805 \text{ (грн)}.$$

Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i * Ц_i * К_i, \quad (4.4)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 4.5 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Файли	92	2	184
Папір	180	2	360
Картридж	780	1	780
Флешка	200	1	400
		Всього	1724
	З врахуванням коефіцієнта транспортування		1896,4

Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалось безкоштовне програмне забезпечення.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою [41]:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.5)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{33\,000{,}0 * 2}{3 * 12} = 1\,666{,}6 \text{ грн.}$$

Таблиця 4.6 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Apple MacBook Air M1 13.3	33 000,0	3	2	1 833
Ноутбук Acer Extensa 15 EX215-57	20 000,0	3	2	1 111
Офісне приміщення	18 000,0	5	2	600
Оргтехніка	4 200	4	2	175
			Всього	3 719

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{yi} * t_i * Ц_е * К_{впі}}{\eta_i} \quad (4.6)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 8,46 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$

$$В_е = \frac{0,3 * 256 * 8,46 * 0,95}{0,97} = 564,1 \text{ грн.}$$

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Lenovo LOQ 15IAX9I	0,35	240	742,4
Ноутбук Asus Vivobook Go	0,4	360	1193,1
Офісне приміщення	0,6	360	1789,7
Оргтехніка	0,2	5	8,3
Всього			3 733,5

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} \quad (4.7)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = (102\,500) * \frac{25}{100} = 25\,625 \text{ грн.}$$

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{сп}}{100\%} \quad (4.8)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = (102\,500) * \frac{35}{100} = 35\,875 \text{ грн.}$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$I_в = (102\ 500) * \frac{55}{100} = 56\ 375 \text{ грн.}$$

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.10)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (102\ 500) * \frac{100}{100} = 102\ 500 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + I_в + В_{нзв} \quad (4.11)$$

$$В_{заг} = 102500 + 10250 + 24805 + 1894{,}4 + 3719 + 3733{,}5 + 25625 + 35875 + 56375 + 102500 = 367\ 276{,}9 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.12)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4;

розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9 [42].

$$ЗВ = \frac{367\ 276,9}{0,7} = 524\ 681,3 \text{ грн.}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В умовах ринкової економіки основним економічним результатом, що становить інтерес для потенційного інвестора під час упровадження результатів науково-технічної розробки, є зростання величини чистого прибутку.

Комерціалізація розробки за темою «Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKI, блокчейн-аудиту та електронного підписування для забезпечення цілісності» передбачається протягом трирічного циклу виходу на ринок. Формування очікуваного економічного ефекту в зазначений період ґрунтуватиметься на таких вихідних параметрах.

Передбачається приріст кількості користувачів програмного продукту, обумовлений підвищенням рівня його захищеності (ΔN), зокрема:

— у першому році впровадження — 85 користувачів;

— у другому році — 60 користувачів;

— у третьому році — 42 користувачів.

Базова чисельність споживачів, які використовували аналогічний програмний продукт у році, що передував упровадженню нової науково-технічної розробки, становить 100 користувачів (N).

Вартість програмного забезпечення до проведення модернізації дорівнювала 218 000 грн ($Ц_β$).

Зміна вартості продукту, що виникла внаслідок реалізації запропонованих удосконалень, становить 2 000 грн ($±ΔЦ_β$).

Можливе збільшення чистого прибутку потенційного інвестора для кожного з трьох років, протягом яких очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, розраховується за формулою.

$$\Delta П_i = (\pm \Delta Ц_б * N + Ц_б * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}) \quad (4.13)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 25\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

1-й рік:

$$\Delta П_1 = (2000 \times 100 + 118\,000 \times 85) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) =$$

$$2\,277\,478{,}8 \text{ (грн.)}$$

2-й рік:

$$\Delta П_2 = (2000 \times 100 + 118\,000 \times (85 + 60)) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) =$$

$$3\,743\,934{,}4 \text{ (грн.)}$$

3-й рік:

$$\Delta П_3 = (2000 \times 100 + 118\,000 \times (85 + 60 + 42)) \times 0{,}83 \times 0{,}25 \times \left(1 - \frac{0{,}18}{100}\right) =$$

$$4\,770\,453{,}3 \text{ (грн.)}$$

Отже, за результатами обчислень, впровадження розробки призведе до значної комерційної вигоди, що виявиться у зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Ключовими критеріями, що впливають на рішення інвестора щодо фінансування наукової розробки, є абсолютна та відносна ефективність інвестицій, а також термін їх окупності.

На початковому етапі визначається теперішня вартість інвестицій (PV), які будуть спрямовані на наукову розробку.

Також розраховується обсяг початкових вкладень, які потенційний інвестор повинен здійснити для впровадження та комерціалізації науково-технічного проєкту [42].

$$PV = k_{інв} * ЗВ \quad (4.14)$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 524 681,3 грн.

$$PV = 2 * 524\,681{,}3 = 1\,049\,362{,}6 \text{ грн.}$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$Е_{абс} = (ПП - PV) \quad (4.15)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{ΔП_1}{(1+\tau)^t} \quad (4.16)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{2\,277\,478,8}{(1+0,2)^1} + \frac{3\,743\,934,4}{(1+0,2)^2} + \frac{4\,770\,453,3}{(1+0,2)^3} = 7\,258\,532,4 \text{ грн.}$$

$$Е_{абс} = 7\,258\,532,4 - 1\,049\,362,6 = 6\,209\,169,8 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, встановлено, що проведення наукових досліджень для розробки програмного продукту та його подальше впровадження принесуть прибуток. Це підтверджує доцільність проведення досліджень [42].

Внутрішня економічна дохідність інвестицій $Е_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.17)$$

де $Е_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$Т_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{6\,209\,169,8}{1\,049\,362,6}} - 1 = 0,9$$

Порівняємо Ев з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає мінімальну дохідність, нижче якої інвестиції не будуть здійснюватися.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за формулою:

$$\tau_{min} = d + f \tag{4.18}$$

d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; f = 0,4.

d = 0,2.

$$\tau_{min} = 0{,}2 + 0{,}4 = 0{,}6$$

Оскільки $E_в$ = 90% > τ_{min} = 60%, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки [42]:

$$T_{ок} = \frac{1}{E_в} \tag{4.19}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0{,}9} = 1{,}1$$

Якщо $T_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.5 Висновки до розділу

На основі проведеного оцінювання встановлено, що розроблена система підвищення захищеності MQTT-комунікацій IoT-пристроїв, яка ґрунтується на використанні сертифікатів PKI, блокчейн-аудиту та електронного підписування, характеризується значним комерційним потенціалом. Запропоноване рішення

забезпечує підвищення рівня криптографічного захисту, покращує надійність передавання даних та зменшує ризики несанкціонованого втручання в IoT-комунікації.

Аналіз витрат і очікуваних результатів свідчить про економічну доцільність упровадження системи та перспективність її використання на ринку. Проєкт демонструє високі показники ефективності та здатність забезпечити підприємству стійкі економічні вигоди у разі комерціалізації. Розробка може бути впроваджена у широкому спектрі сфер застосування, де критично важливими є цілісність, достовірність та захищеність даних у мережах IoT.

ВИСНОВКИ

У магістерській роботі було проведено комплексне дослідження проблеми забезпечення захищеності MQTT-комунікацій у системах Інтернету речей та розроблено вдосконалену модель, яка поєднує три ключові механізми: PKI-автентифікацію пристроїв, електронне підписування MQTT-повідомлень та блокчейн-орієнтований аудит подій. На основі проведеного аналізу, теоретичних досліджень та програмної реалізації встановлено, що класичний MQTT-протокол, будучи легковагим і оптимізованим для низькоресурсних пристроїв, не забезпечує достатнього рівня безпеки у критичних застосуваннях, що потребують підтвердження автентичності, цілісності та недовідмовності.

У першому розділі роботи було проаналізовано сучасні загрози безпеці IoT-систем, виявлено основні вразливості MQTT-комунікацій та охарактеризовано недоліки існуючих механізмів захисту. Проведено детальне дослідження PKI-технологій, криптографічних методів та моделей контролю доступу, а також методів аудитування на основі блокчейну. Результати аналізу дозволили сформувати чіткі вимоги до вдосконаленої моделі захисту.

У другому розділі побудовано концептуальну модель захищених MQTT-комунікацій, яка інтегрує мережеві та криптографічні механізми в єдину архітектуру. Було розроблено формалізований алгоритм підвищеної захищеності, що описує послідовність дій під час автентифікації пристрою, формування та валідації електронного підпису, а також запису подій у блокчейн-журнал. Візуальні діаграми та формальні моделі підтвердили коректність взаємодії компонентів та логіку роботи системи.

У третьому розділі виконано програмну реалізацію кожного елемента розробленої моделі. Реалізовано PKI-автентифікацію MQTT-клієнтів із використанням сертифікатів X.509, модуль електронного підписування та перевірки цілісності повідомлень на основі алгоритмів Ed25519, а також блокчейн-орієнтований журнал подій, що забезпечує незмінність логів. Проведене

тестування підтвердило функціональну правильність реалізованих механізмів. Отримані експериментальні дані показали, що інтеграція додаткових механізмів захисту збільшує затримку обробки повідомлень, однак цей приріст є незначним і прийнятним для більшості практичних IoT-застосувань. Система успішно виявляє невалідні сертифікати, модифіковані повідомлення, порушення цілісності даних та аномалії у роботі брокера.

У підсумку розроблена система забезпечує комплексний багаторівневий механізм захисту MQTT-комунікацій, який може бути адаптований для використання у корпоративних, промислових та критично важливих IoT-інфраструктурах. Вона поєднує криптографічну стійкість, прозорість аудитування та низькі накладні витрати, що робить її придатною для реальних сценаріїв експлуатації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stallings W. *Cryptography and Network Security: Principles and Practice*. — 8th ed. — Pearson, 2022. — 768 p.

2. Bertino E., Islam N. *Botnets and Internet of Things Security*. — Springer, 2022. — 240 p.

3. Alaba F.A., Othman M., Hashem I.A.T., Alashhab Z.R. *Internet of Things security: A survey*. Future Generation Computer Systems, 2022.

4. ISO/IEC 27400:2022 — *Cybersecurity — Internet of Things (IoT) security and privacy*.

5. Li S., Tryfonas T. *Securing the Internet of Things*. — Elsevier, 2021. — 350 p.

6. Лі В. *Основи криптографії з відкритим ключем*. — К. : Наш формат, 2021. — 320 с.

7. Syed T. et al. *Blockchain for IoT Security and Privacy: Challenges and Solutions*. IEEE Access, 2021.

8. RFC 9147: *The Transport Layer Security (TLS) Protocol Version 1.3*. — IETF, 2021.

9. RFC 8995: *Bootstrapping Remote Secure Key Infrastructures (BRSKI)*. — IETF, 2020.

10. European Union Agency for Cybersecurity (ENISA). *Good Practices for IoT Security 2020*. — ENISA, 2020.

11. Chishti M.A., Shafiq M. *Anomaly detection in IoT networks using machine learning*. IJCS, 2020.

12. NIST SP 800-207: *Zero Trust Architecture*. — NIST, 2020.

13. Beckers K. *Patterns for Secure IoT Systems*. Springer, 2020.

14. MQTT Version 5.0 Specification. OASIS Standard, 2019.

15. Singh M., Rajan M.A. *Securing MQTT in IoT applications*. IEEE ICCCS, 2019.

16. Python Software Foundation. *Python 3.12 Documentation*. — Режим доступу: https://docs.python.org/3/

17. Node.js Foundation. *Node.js v20 Documentation*. — Режим доступу: https://nodejs.org

18. Таненбаум Е. *Архітектура комп'ютерних мереж*. — К. : Вільямс, 2015. — 1024 с.

19. Mosquitto MQTT Broker Official Documentation. — Режим доступу: https://mosquitto.org/

20. Hyperledger Fabric Documentation. — Режим доступу: https://hyperledger.org

21. ISO/IEC 27001:2013. *Information technology — Security techniques — Information security management systems — Requirements*.

22. ISO/IEC 30141:2018. *Internet of Things Reference Architecture*.

23. ДСТУ ISO/IEC 18033-3:2017. *Інформаційні технології. Криптографічні алгоритми. Частина 3. Блокові шифри*.

24. Шнайєр Б. *Прикладна криптографія*. — К. : Вільямс, 2016. — 784 с.

25. Грінстайн М., Спатсчек Р. *Протоколи Інтернету*. — К. : Діалектика, 2010. — 432 с.

26. Mahmood Z., Afzal S. *Cloud and IoT Security*. Springer, 2018.

27. Khan M.A., Salah K. *IoT security: Review, blockchain solutions, and open challenges*. Future Generation Computer Systems, 2018.

28. RFC 5246: *The Transport Layer Security (TLS) Protocol Version 1.2*. — IETF, 2018.

29. Modiri N., Faghih M. *MQTT Security Analysis*. Journal of Information Security, 2018.

30. Fruhlinger J. *What is PKI? Public Key Infrastructure explained*. CSO Online, 2017.

31. Столяренко О.М. *Захист інформації в комп'ютерних системах*. — К. : КНЕУ, 2017. — 312 с.

32. Aggarwal C.C. *Machine Learning for Anomaly Detection*. Springer, 2017.

33. RFC 6455: *The WebSocket Protocol*. — IETF, 2011.

34. Федотов А.А. *Веб-програмування. Серверна частина*. — СПб. : Питер, 2019. — 448 с.

35. Provos N., Honeyman P. *The future of PKI*. USENIX Security Symposium, 2013.

36. Nakamoto S. *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

37. ДСТУ 27347:2005. *Інформаційні технології. Захист інформації. Терміни та визначення*.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК

"24" вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKI, блокчейн-аудиту та електронного підписування для забезпечення цілісності

08-72.МКР.013.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент каф. МБІС

_____ Грицак А.В.

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний засіб підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKI, блокчейн-аудиту та електронного підписування для забезпечення цілісності

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №96 від 20. 03. 2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу підвищення захищеності MQTT-комунікацій IoT-пристроїв.

3.2 Призначення: розроблений програмний засіб виконує захист MQTT-комунікацій IoT-пристроїв.

4. Джерела розробки

4.1. Ахрамович В. М. Ідентифікація й аутентифікація, керування доступом // Сучасний захист інформації. – 2016. №4.– С. 47-51.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. ZakariaOmar, ZangooeiToomaj, MohdAfiziMohdShukran. Enhancing Mixing Recognition-Based and Recall-Based Approach in Graphical Password Scheme. IJACT, Vol. 4, No. 15, pp. 189-197, 2012.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	01.09	10.09
2	Аналіз предметної області обраної теми	11.09	25.09
3	Апробація отриманих результатів	26.09	05.10
4	Розробка алгоритму роботи	06.10	15.10
5	Написання магістерської роботи на основі розробленої теми	16.10	03.11
6	Розробка економічної частини	20.10	30.10
7	Передзахист магістерської кваліфікаційної роботи	03.11	03.11
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	03.11	08.11
9	Захист магістерської кваліфікаційної роботи	08.12	11.12

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відгук керівника роботи;

– відгук опонента

Технічне завдання до виконання прийняв _____ Марцун Н.М.

Додаток Б. Лістинг програми

```
const path = require("path");


module.exports = {
 mqttBroker: {
  host: "mqtts://broker.example.com",
  port: 8883,
  clientIdPrefix: "iot-device-",
 },
 pki: {
  caCertPath: path.join(__dirname, "certs", "ca.pem"),
  clientCertPath: path.join(__dirname, "certs", "client.pem"),
  clientKeyPath: path.join(__dirname, "certs", "client-key.pem"),
 },
 tls: {
  rejectUnauthorized: true, // вимагати перевірку сертифіката брокера
  minVersion: "TLSv1.2",
 },
};
            const fs = require("fs");
            const logger = console;
            const config = require("./pki-config");
            function loadFileOrThrow(filePath, description) {
              try {
                const data = fs.readFileSync(filePath);
                logger.info(`[PKI] Завантажено ${description}: ${filePath}`);
                return data;
              } catch (err) {
                logger.error(`[PKI] Помилка завантаження ${description} (${filePath}): ${err.message}`);
                throw err;
              }
            }
function loadPkiCredentials() {
              const { caCertPath, clientCertPath, clientKeyPath } = config.pki;
              const ca = loadFileOrThrow(caCertPath, "сертифікат CA");
              const cert = loadFileOrThrow(clientCertPath, "сертифікат клієнта");
              const key = loadFileOrThrow(clientKeyPath, "приватний ключ клієнта");
```

```
  return { ca, cert, key };
}

module.exports = {
  loadPkiCredentials,
};
// mqtt-client-factory.js
const mqtt = require("mqtt");
const { loadPkiCredentials } = require("./pki-service");
const config = require("./pki-config");
const logger = console;
function createSecureMqttClient(deviceIdSuffix) {
  const { host, port, clientIdPrefix } = config.mqttBroker;
  const { ca, cert, key } = loadPkiCredentials();
  const clientId = `${clientIdPrefix}${deviceIdSuffix}`;
  const connectUrl = `${host}:${port}`;
  const options = {
    clientId,
    ca,
    cert,
    key,
    rejectUnauthorized: config.tls.rejectUnauthorized,
    minVersion: config.tls.minVersion,
};
  logger.info(`[MQTT] Підключення до брокера ${connectUrl} як ${clientId}`);
  const client = mqtt.connect(connectUrl, options);
  client.on("connect", () => {
    logger.info(`[MQTT] Клієнт ${clientId} успішно аутентифікований через PKI та підключений`);
  });
  client.on("error", (err) => {
    logger.error(`[MQTT] Помилка підключення клієнта ${clientId}: ${err.message}`);
  });
  client.on("close", () => {
    logger.warn(`[MQTT] З'єднання клієнта ${clientId} закрито`);
  });
  return client;
}
module.exports = {
  createSecureMqttClient,
```

```
};
function canonicalizeMessage(topic, payload, clientId) {
  return JSON.stringify({
    topic,
    clientId,
    ts: Date.now(),
    nonce: crypto.randomUUID(),
    payload
  });
}

module.exports = { canonicalizeMessage };
const crypto = require("crypto");
const fs = require("fs");
const path = require("path");
const PRIVATE_KEY_PATH = path.join(__dirname, "certs", "client-key.pem");
function signMessage(canonicalMessage) {
  const privateKey = fs.readFileSync(PRIVATE_KEY_PATH);
  const signature = crypto.sign(null, Buffer.from(canonicalMessage), privateKey);
  return signature.toString("base64");
}

module.exports = { signMessage };
const crypto = require("crypto");
const fs = require("fs");
const path = require("path");
const CLIENT_CERT_PATH = path.join(__dirname, "certs", "client.pem");
function verifySignature(canonicalMessage, signatureBase64) {
  const cert = fs.readFileSync(CLIENT_CERT_PATH);
  const publicKey = crypto.createPublicKey(cert);
  const signature = Buffer.from(signatureBase64, "base64");
  return crypto.verify(null, Buffer.from(canonicalMessage), publicKey, signature);
}
module.exports = { verifySignature };
const client = require("./secure-mqtt-client");
const { canonicalizeMessage } = require("./canonicalizer");
const { signMessage } = require("./signer");
const TOPIC = "secure/data";
function publishSecure(payload) {
```

```
  const canonical = canonicalizeMessage(TOPIC, payload, client.options.clientId);
  const signature = signMessage(canonical);
  const meta = {
    canonical,
    signature
  };
  client.publish(TOPIC, JSON.stringify(meta), { qos: 1 });
}
publishSecure({ temperature: 22.5, status: "OK" });
const crypto = require("crypto");

class Block {
  constructor({ index, timestamp, eventType, clientId, topic, dataHash, verifyStatus, prevHash }) {
    this.index = index;
    this.timestamp = timestamp;
    this.eventType = eventType;
    this.clientId = clientId;
    this.topic = topic;
    this.dataHash = dataHash;
    this.verifyStatus = verifyStatus;
    this.prevHash = prevHash;
    this.hash = this.calculateHash();
  }

  calculateHash() {
    const content = JSON.stringify({
      index: this.index,
      timestamp: this.timestamp,
      eventType: this.eventType,
      clientId: this.clientId,
      topic: this.topic,
      dataHash: this.dataHash,
      verifyStatus: this.verifyStatus,
      prevHash: this.prevHash
    });
    return crypto.createHash("sha256").update(content).digest("hex");
  }
}
```

```
module.exports = Block;
const Block = require("./block");

class JournalChain {
 constructor() {
  this.chain = [this.createGenesisBlock()];
 }

 createGenesisBlock() {
  return new Block({
   index: 0,
   timestamp: Date.now(),
   eventType: "GENESIS",
   clientId: "SYSTEM",
   topic: "none",
   dataHash: "0",
   verifyStatus: "valid",
   prevHash: "0"
  });
 }

 getLatestBlock() {
  return this.chain[this.chain.length - 1];
 }

 addBlock(event) {
  const prevBlock = this.getLatestBlock();

  const newBlock = new Block({
   index: prevBlock.index + 1,
   timestamp: Date.now(),
   eventType: event.eventType,
   clientId: event.clientId,
   topic: event.topic,
   dataHash: event.dataHash,
   verifyStatus: event.verifyStatus,
   prevHash: prevBlock.hash
  });
```

```
    this.chain.push(newBlock);
    return newBlock;
  }

  validateChain() {
    for (let i = 1; i < this.chain.length; i++) {
      const current = this.chain[i];
      const previous = this.chain[i - 1];

      if (current.hash !== current.calculateHash()) return false;
      if (current.prevHash !== previous.hash) return false;
    }
    return true;
  }
}

module.exports = JournalChain;
const crypto = require("crypto");
const JournalChain = require("./chain");
const journal = new JournalChain();
function recordEvent(eventType, clientId, topic, canonicalMessage, verifyStatus) {
  const dataHash = crypto.createHash("sha256").update(canonicalMessage).digest("hex");

  const block = journal.addBlock({
    eventType,
    clientId,
    topic,
    dataHash,
    verifyStatus
  });
  console.log("[BLOCKCHAIN] Додано блок:", block.hash);
}
module.exports = { recordEvent, journal };
```

Додаток В. Ілюстративний матеріал

Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKL, блокчейн-аудиту та електронного підписування для забезпечення цілісності

Виконав: студент 2 курсу, групи КІТС-24М, Марцун Н.М.

Керівник: к.т.н., доцент Грицак А.В.

Вступ

- Системи Інтернету речей стрімко інтегруються у корпоративні та промислові середовища, але їх безпека все частіше стає критичною проблемою. Протокол MQTT, який широко використовується для передачі телеметрії між пристроями, у базовому вигляді не забезпечує достатнього рівня захисту — зберігаються ризики підробки даних, компрометації пристроїв і маніпуляцій журналами подій.
- Метою моєї роботи є розробка вдосконаленої моделі захисту MQTT-комунікацій, яка поєднує PKI-автентифікацію, електронне підписування повідомлень та блокчейн-аудит подій. Такий підхід дозволяє гарантувати автентичність пристроїв, цілісність даних і неможливість непомітної модифікації журналів. У ході дослідження створено повністю функціональний програмний прототип, проведено тестування та оцінено ефективність інтегрованого рішення.

Актуальність та мета

МЕТА

Розробити вдосконалену модель захисту MQTT-комунікацій, що поєднує PKI-автентифікацію, електронне підписування повідомлень і блокчейн-аудит для забезпечення цілісності, автентичності та прозорості переданих даних.

АКТУАЛЬНІСТЬ

- MQTT широко використовується в IoT, але не має вбудованих механізмів захисту. Це створює високі ризики підміни даних, несанкціонованого доступу та маніпуляції журналами подій, що є критичним для корпоративних і промислових систем.

Загальна структура IoT-комунікацій у середовищі MQTT

Аналіз існуючих методів захисту MQTT-комунікацій у системах IoT

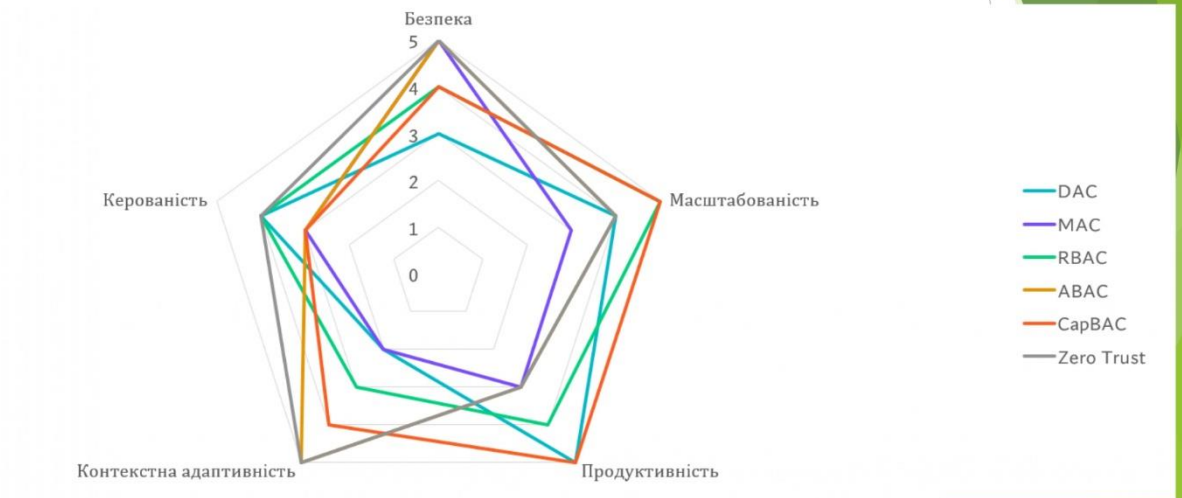
Огляд моделей контролю доступу

Модель	Принцип дії	Переваги	Недоліки
DAC (Discretionary Access Control)	Власник ресурсу сам визначає, хто має право доступу.	Простота реалізації, гнучкість.	Високий ризик помилок конфігурації, слабкий захист від внутрішніх загроз.
MAC (Mandatory Access Control)	Система централізовано визначає рівні доступу та політики.	Високий рівень безпеки, суворий контроль.	Низька гнучкість, складність оновлення політик.
RBAC (Role-Based Access Control)	Доступ базується на ролях користувачів у системі.	Масштабованість, легкість адміністрування.	Потребує регулярного оновлення ролей у динамічних середовищах.
ABAC (Attribute-Based Access Control)	Доступ визначається на основі набору атрибутів користувача, ресурсу й середовища.	Контекстна адаптивність, висока точність політик.	Складність реалізації та потреба в метаданих.
CapBAC (Capability-Based Access Control)	Дозвіл доступу реалізується через унікальні токени (capabilities).	Висока продуктивність, простота перевірки прав.	Необхідність управління та ротації токенів.
Zero Trust Model	Довіра не передбачається — кожен запит перевіряється незалежно.	Максимальна безпека, постійна автентифікація.	Висока вартість і складність інтеграції.

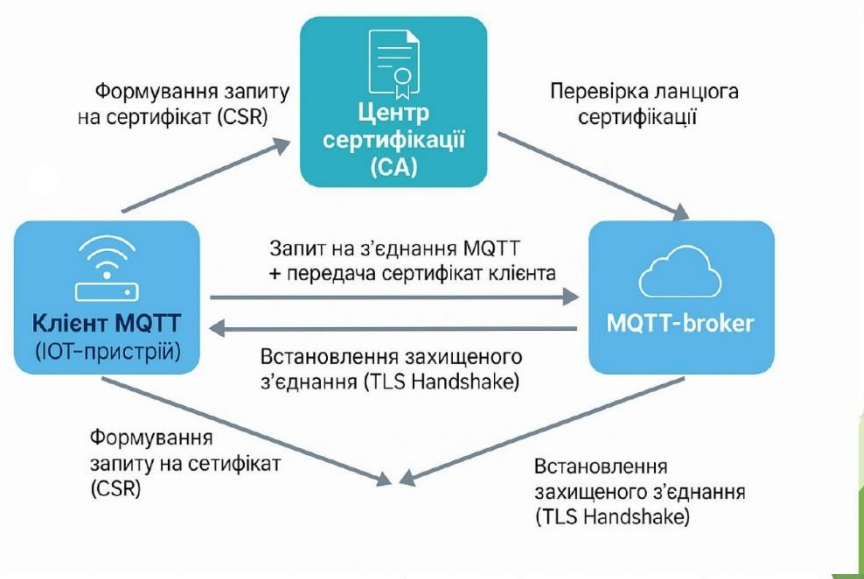
Порівняння моделей контролю доступу в системах

Модель	Безпека	Масштабованість	Продуктивність	Контекстна адаптивність	Керованість
DAC	3	4	5	2	4
MAC	5	3	3	2	3
RBAC	4	5	4	3	4
ABAC	5	4	3	5	3
CapBAC	4	5	5	4	4
Zero Trust	5	4	3	5	3

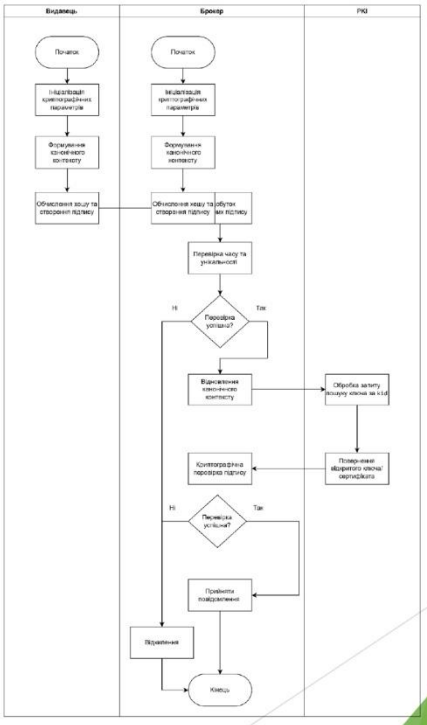
Інтегральна оцінка моделей контролю доступу за основними критеріями

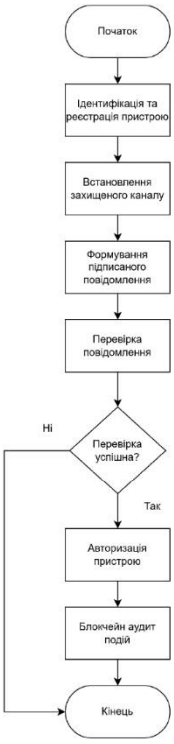


Розроблена модель інтеграції сертифікатів PKI у процес автентифікації MQTT-пристроїв



Розроблема діаграма процесу електронного підписування та перевірки MQTT-повідомлень



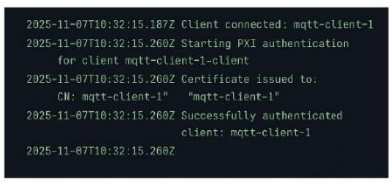


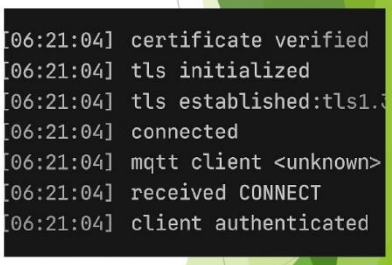
Блок-схема алгоритму підвищеної захищеності MQTT-комунікацій

Середовище розробки і мова програмування

Visual Studio Code

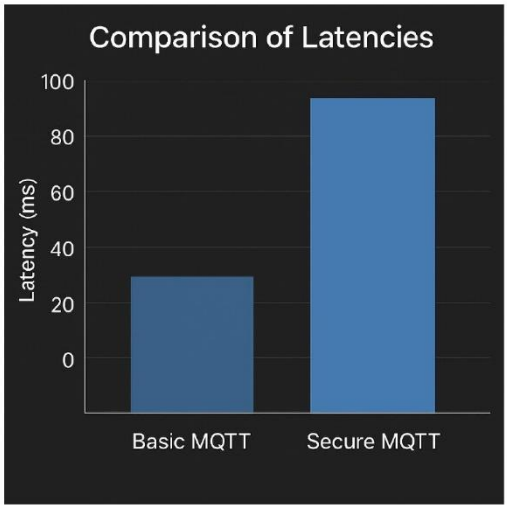
Тестування системи





```
[06:22:35] certificate verified
[06:22:35] tls initialized
[06:22:35] tls established:tls1.3
[06:22:35] connected
[06:22:35] mqtt client <unknown>
[06:22:35] received CONNECT
[06:22:35] ERROR block add to
           chain failed
```

```
[06:21:05] HTTP/1.1 201 Created
content-type: application/json
<blockchain_audit:
  timestamp:'2025-11-21T04:21:04Z',
  client_id:'device123'
  topic:'sensor/data'
  hash:'4f8d9c84bf0d5aadc72fa69edd
  cd46c401acfa21ad1bb4e68e48fb466
  9e746d33
```



Висновок

- Розроблена модель суттєво підвищує безпеку MQTT-комунікацій, забезпечуючи автентифікацію пристроїв, цілісність повідомлень і прозорий аудит подій. Інтеграція PKI, електронного підпису та блокчейн-журналу довела свою ефективність під час тестування й може бути впроваджена в реальні корпоративні та промислові IoT-системи. Запропонований підхід створює надійну основу для подальшого розвитку та масштабування захищених IoT-платформ.

Дякую за увагу!

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Підвищення захищеності MQTT-комунікацій IoT-пристроїв на основі сертифікатів PKL, блокчейн-аудиту та електронного підписування для забезпечення цілісності

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **1,10 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ доц. Грицак А.В.

Здобувач _____ Марцун Н.М.