Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Пунди Романа Вікторовича

на здобуття ступеня вищої освіти Бакалавра

Система захисту кіберфізичної системи моніторингу фізіологічних показників.

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ. 220240.22.02.33 ПЗ

Виконав студент 4 курсу група КБ-22-2 Роман Пунда
Керівник д-р філософії, ст. викладач Микола СТЕЦЮК
Нормоконтролер д-р філософії Наталія ПЕТЛЯК

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

10 06 2026 р.

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

09 лютого 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Пунді Роману Вікторовичу

1 Тема роботи Система захисту кіберфізичної системи моніторингу фізіологічних показників.

Керівник роботи д-р філософії, ст. викладач Микола Стецюк

Затверджено наказом ректора університету від 8 січня 2026 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 25 травня 2026

3 Вихідні дані до роботи Проаналізувати безпеку медичного IoT та вразливості класичної автентифікації мікроконтролерів. Обґрунтувати застосування мікросервісної архітектури та моделі Zero Trust. Спроєктувати й реалізувати комплекс безстанної автентифікації на базі JWT та RSA-256 для збору телеметрії. Забезпечити контейнеризацію інфраструктури в Docker та експериментально перевірити захищеність системи.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналіз сучасного стану кібербезпеки в екосистемах Інтернету речей та обмежень традиційних методів автентифікації. Обґрунтування вибору мікросервісного підходу, концептуальне проєктування структурної топології кіберфізичної системи та ешелонованої моделі захисту. Програмна реалізація криптографічного ядра (Auth Service та Resource Service), ізоляція інфраструктури в середовищі Docker. Експериментальна перевірка та тестування безпеки системи. Загальні висновки.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Структурна схема захищеної мікросервісної архітектури кіберфізичної системи. Блок-схема алгоритму емісії JWT-токена сервісом автентифікації. Блок-схема алгоритму автономної перевірки токенів та верифікації телеметрії на приймальному шлюзі.

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 09 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Січень-Лютий	
Ознайомлення з предметною областю	Лютий	
Дослідження існуючих рішень	Лютий	
Постановка задачі	Березень	
Визначення загальних принципів рішення задачі	Березень	
Деталізація принципів рішення задачі	Квітень	
Розробка проєктних рішень	Квітень	
Апробація проєктних рішень	Травень	
Оформлення пояснювальної записки згідно вимог	Травень	
Оформлення графічної частини	Травень	
Захист КР	Червень	

Студент _____ Роман ПУНДА

Керівник кваліфікаційної роботи _____ Микола СТЕЦЮК

АНОТАЦІЯ

Тема кваліфікаційної роботи: Система захисту кіберфізичної системи моніторингу фізіологічних показників.

Автор роботи: Пунда Роман Вікторович.

Керівник Роботи: Стецюк Микола Васильович.

Пояснювальна записка: 62 с., 2 додатів, 20 рисунків, 1 таблиця, 40 джерел.

Графічна частина: 3 плакати.

Ключові слова: кібербезпека, інтернет речей, медична телеметрія, мікросервісна архітектура, безстанна автентифікація.

Кваліфікаційна робота бакалавра присвячена проєктуванню та програмній реалізації захищеної мікросервісної архітектури для безпечного збору, обробки та маршрутизації фізіологічних показників пацієнтів у медичних екосистемах Інтернету речей (IoT).

У роботі проведено комплексний аналіз вразливостей традиційних сесійних методів автентифікації для мікроконтролерів з обмеженими обчислювальними ресурсами. Обґрунтовано доцільність застосування парадигми нульової довіри (Zero Trust) та розроблено концептуальну структурну топологію ешелонованого захисту. Програмно реалізовано криптографічне ядро системи на базі стандарту JWT з використанням асиметричного алгоритму шифрування RSA-256. Впровадження повністю безстанної (stateless) моделі дозволило забезпечити автономну валідацію цифрових перепусток на приймальному шлюзі без зайвих звернень до бази даних. Забезпечено ізоляцію інфраструктури за допомогою технології контейнеризації Docker. Комплексне експериментальне тестування підтвердило стійкість розроблених механізмів до атак типу Man-in-the-Middle та спроб підміни телеметрії. Запропоноване рішення є технічно обґрунтованим і може бути використане для побудови безпечних медичних інформаційних систем нового покоління.

11.06.2026 _____

ABSTRACT

Theme of the qualification work: Artificial immune system for protecting information and communication networks.

Author of the work:Punda Roman Viktorovich.

Advisor:Mykola Stetsiuk.

Explanatory note: 65 p., 2 appendices, 20 figures, 1 table, 40 references.

Graphic part: 3 posters.

Keywords: cybersecurity, Internet of Things, medical telemetry, microservices architecture, continuous authentication.

The bachelor's qualification work is devoted to the design and software implementation of a secure microservice architecture for the safe collection, processing, and routing of patient physiological parameters in medical Internet of Things (IoT) ecosystems.

The work conducts a comprehensive analysis of the vulnerabilities of traditional session authentication methods for microcontrollers with limited computational resources. The expediency of applying the Zero Trust paradigm is justified, and a conceptual structural topology of layered security is developed. The cryptographic core of the system is implemented in software based on the JWT standard using the asymmetric RSA-256 encryption algorithm. The implementation of a fully stateless access control model ensured the autonomous validation of digital tokens at the receiving gateway without redundant database queries. Infrastructure isolation is ensured using Docker containerization technology. Comprehensive experimental testing confirmed the resilience of the developed mechanisms against Man-in-the-Middle attacks and telemetry spoofing attempts. The proposed solution is technically justified and can be used to build next-generation secure medical information systems.

01.06.2026

ЗМІСТ

Зм.	Арк.	№докум.	Підпис	Дата	КРБКБ. 220251.22.02.33 ПЗ		
Виконав		Пунда Р.В.		01.06	Система захисту кіберфізичної системи моніторингу фізіологічних показників Пояснювальна записка	Літера	Аркуш	Аркушів
Перевір.		Стешок М.В.		02.06		н	6	62
Н.контр.		Петляк Н.С.				ХНУ, КБ-22-2		
Затвер.		Кльоц Ю.П.		10.06				

ВСТУП

Інтеграція новітніх електронних пристроїв та мережевих рішень у сферу охорони здоров’я призвела до стрімкого поширення кіберфізичних систем. Такі апаратно-програмні комплекси здатні безперервно фіксувати, обробляти та транслювати життєво важливі параметри організму людини. Однак децентралізована природа Інтернету речей (IoT), постійний обмін даними через бездротові канали зв'язку та обмежені обчислювальні потужності кінцевих сенсорів створюють специфічні умови функціонування. Це вимагає відмови від застарілих методів захисту на користь оптимізованих та ресурсоефективних механізмів гарантування безпеки.

Актуальність обраної теми пояснюється критичною необхідністю убезпечити медичну інформацію від несанкціонованого втручання, перехоплення або фальсифікації. Фізіологічні показники є суворо конфіденційними даними, і будь-яка їх компрометація несе пряму загрозу приватності пацієнтів, а порушення цілісності телеметрії може призвести до хибних медичних висновків. Вирішення цієї проблеми вбачається у розробці та впровадженні комплексної багаторівневої архітектури. Такий підхід передбачає застосування апаратного та мережевого криптографічного захисту, що дозволить організувати надійні канали передачі даних, мінімізувати вплив потенційних кібератак та зберегти автономність роботи діагностичних модулів.

Метою кваліфікаційної роботи є розробка та проєктування захищеної архітектури кіберфізичної системи, призначеної для моніторингу фізіологічних показників. Для досягнення цього результату в роботі досліджуються актуальні вектори загроз для IoT-інфраструктури, вивчаються специфічні вразливості телемедичного обладнання, а також здійснюється обґрунтований вибір мікроконтролерної бази, криптографічних алгоритмів та мережевих технологій для формування стійкого до зламів середовища.

Об’єктом дослідження є процес забезпечення інформаційної безпеки під час функціонування кіберфізичних систем медичного моніторингу.

Предметом дослідження є архітектурні моделі, криптографічні протоколи та програмно-апаратні механізми, які застосовуються для захисту телеметричних даних на етапах їх збору, трансляції та серверної обробки в середовищі Інтернету речей.

Для розв'язання поставлених завдань у роботі використано комплекс методів наукового та інженерного дослідження: методи системного аналізу - для вивчення архітектурних особливостей медичних IoT-мереж, специфіки маршрутизації телеметрії та актуальних векторів кібератак; методи криптографічного аналізу - для обґрунтування вибору асиметричних алгоритмів (RSA-256) та механізмів адаптивного хешування (BCrypt); методи архітектурного моделювання - для проєктування мікросервісної інфраструктури на базі парадигми нульової довіри (Zero Trust); методи емпіричного тестування та програмної симуляції - для верифікації стійкості розробленого криптографічного ядра до несанкціонованого доступу та мережевих маніпуляцій.

Практичне значення одержаних результатів полягає у тому, що спроєктована захисна архітектура та її програмна імплементація доведені до рівня повністю функціонального прототипу. Запропонована безстанна модель управління доступом (stateless) на базі цифрових перепусток JWT дозволяє суттєво знизити обчислювальне навантаження на автономні медичні мікроконтролери. Розроблені мікросервіси, інкапсульовані у Docker-контейнери, формують готове, безпечне та відмовостійке рішення. Запропонований програмно-апаратний комплекс може бути безпосередньо інтегрований у реальні медичні інформаційні системи для організації захищеного дистанційного нагляду за станом здоров'я пацієнтів як у стаціонарах, так і в домашніх умовах.

1 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ТА РІШЕНЬ ЗАХИСТУ ДАНИХ У КІБЕРФІЗИЧНИХ СИСТЕМАХ МОНІТОРИНГУ

1.1 Особливості архітектури кіберфізичних систем

У сучасній інженерії та розробці програмно-апаратних комплексів дедалі більшого поширення набувають кіберфізичні системи (КФС). Цей підхід активно застосовується для побудови гнучких, масштабованих та автономних діагностичних рішень, що поєднують фізичні процеси вимірювання з обчислювальними потужностями та цифровими комунікаціями [1].

Кіберфізична система моніторингу фізіологічних показників являє собою багаторівневу архітектурну модель, у якій набір фізичних сенсорів (наприклад, датчики пульсу, температури тіла чи рівня оксигенації крові) взаємодіє з мікроконтролерним рівнем та хмарною серверною інфраструктурою через стандартизовані мережеві протоколи [2]. На відміну від традиційного медичного обладнання, що працює локально та ізольовано, кожен вузол такої системи функціонує як невід'ємна частина глобального середовища, здатна до самостійного збору, попередньої обробки та передачі телеметричних даних [3].

Фундаментальним принципом побудови подібних систем є децентралізація процесу збору інформації та модульність компонентів. Кожен апаратний модуль має власну мікропрограмну логіку та обмежену зону відповідальності. Це гарантує, що збій у роботі або втрата зв'язку з одним датчиком не призведе до повної зупинки всього діагностичного комплексу [4]. Крім того, ресурсомісткі задачі, такі як глибокий аналіз даних, довгострокове зберігання історії хвороби або алгоритми машинного навчання, делегуються на серверний рівень. Це дозволяє суттєво оптимізувати енергоспоживання кінцевих автономних пристроїв.

Важливою особливістю архітектури є використання спеціалізованих протоколів взаємодії, оптимізованих для Інтернету речей (IoT), таких як MQTT або CoAP. Це забезпечує надійну передачу даних навіть в умовах вузькосмугового або нестабільного бездротового підключення та дозволяє легко

інтегрувати нові типи датчиків в уже існуючу інфраструктуру [5].

Впровадження кіберфізичної архітектури надає низку беззаперечних переваг. По-перше, забезпечується безперервність та висока оперативність моніторингу, що дозволяє медичному персоналу або автоматизованим системам миттєво реагувати на критичні зміни у стані здоров'я особи [6]. По-друге, гарантується висока масштабованість: до мережі можна динамічно підключати нові пристрої без зміни ядра системи. По-третє, автоматизація збору показників знімає необхідність стаціонарного перебування пацієнтів у закладах охорони здоров'я, роблячи можливим повноцінний дистанційний контроль [7].

Разом із тим, зазначений підхід має певні технічні обмеження та недоліки. До головних проблем належать суворі апаратні ліміти кінцевих пристроїв (обсяг оперативної пам'яті, обчислювальна потужність процесора) та висока залежність від стабільності бездротових мереж. Ускладнюється також процес адміністрування та моніторингу працездатності великої кількості розподілених вузлів [8].

Найбільш критичним викликом для кіберфізичних систем є гарантування інформаційної безпеки. Оскільки телеметричні дані передаються відкритими або напіввідкритими каналами зв'язку, кожен вузол мережі перетворюється на потенційну мішень для зловмисників. Головною проблемою стає забезпечення надійної автентифікації пристроїв, цілісності фізіологічних показників та захисту інформації від перехоплення [9]. Відсутність стійких механізмів криптографічного захисту може призвести до модифікації медичних даних або реалізації атак типу "відмова в обслуговуванні" (DDoS), що становить пряму загрозу для пацієнтів.

Таким чином, архітектура кіберфізичних систем є передовим методом організації дистанційного моніторингу, проте її розподілена природа вимагає впровадження спеціалізованих механізмів криптографічного захисту та управління доступом для нейтралізації специфічних загроз у середовищі IoT [10].

1.2 Основні загрози та підходи до захисту інформації в IoT-середовищі

Забезпечення стійкості кіберфізичних систем медичного призначення вимагає глибокого розуміння специфіки функціонування Інтернету речей. Враховуючи, що такі апаратно-програмні комплекси здійснюють безперервну агрегацію та маршрутизацію конфіденційних фізіологічних метрик, їх захист не може обмежуватися класичними периметральними методами. Необхідним є формування ешелонованої моделі безпеки, яка б охоплювала всі етапи життєвого циклу даних: від первинного зняття показників сенсором до їх збереження у хмарній базі даних [11].

Специфіка функціонування медичних сенсорів та IoT-шлюзів полягає в тому, що вони працюють у гетерогенних мережах із використанням бездротових каналів зв'язку. Це вимагає відмови від поняття "довіреної мережі" на користь концепції "Нульової довіри" (Zero Trust Architecture). Відповідно до цієї моделі, жоден вузол, пакет даних або сеанс зв'язку не вважається безпечним за замовчуванням. Кожен датчик, мікроконтролер чи серверний модуль повинен проходити сувору взаємну автентифікацію перед початком трансляції будь-яких телеметричних даних [12].

Варто наголосити, що фізіологічні показники, які збираються кіберфізичними системами (пульс, сатурація, температура, кардіограма), згідно з міжнародними стандартами (наприклад, HIPAA або GDPR) класифікуються як критично чутливі біометричні та медичні дані. Їхня специфіка полягає в тому, що вони є незмінними характеристиками пацієнта. На відміну від пароля або фінансового реквізиту, скомпрометовані фізіологічні чи біометричні дані неможливо "перевипустити" або "скинути". Саме тому до архітектури медичного IoT-середовища висуваються безпрецедентні вимоги не лише щодо шифрування каналів зв'язку, але й щодо псевдонімізації та знеособлення даних безпосередньо на рівні збору (Edge Layer). Крім того, порушення доступності системи внаслідок мережевих атак становить загрозу не просто для бізнес-процесів, а для здоров'я та життя пацієнта, оскільки медичний персонал втрачає можливість

своєчасно реагувати на критичні стани (наприклад, зупинку дихання або критичну аритмію). Це перетворює задачу захисту кіберфізичних систем із суто технічної проблеми на питання безпеки життєдіяльності.

Для формування комплексного розуміння векторів потенційних кібератак та методів протидії їм, у роботі спроєктовано багаторівневу модель безпеки. Загальну схему ешелонованого захисту та локалізації ключових загроз у кіберфізичній системі моніторингу наведено на рисунку 1.1.

Рисунок 1.1 – Базова модель загроз та ешелонованого захисту кіберфізичної системи моніторингу

Відповідно до розробленої базової моделі (рис. 1.1), процес безпечної взаємодії в кіберфізичній системі моніторингу можна розділити на чотири ключові етапи, кожен з яких супроводжується специфічними ризиками та вимагає відповідних механізмів захисту:

Етап 1: Запит на автентифікацію. На початковому етапі мікроконтролер, який виконує роль клієнта і збирає фізіологічні показники, не має права безпосередньо відправляти телеметрію на сервер обробки ресурсів. Для встановлення довіреного сеансу мікроконтролер звертається до Сервера Автентифікації (Auth Service), передаючи свої унікальні ідентифікаційні дані або криптографічний ключ. Цей етап критичний для запобігання підключенню до мережі несанкціонованих або скомпрометованих (підроблених) медичних датчиків.

Етап 2: Генерація та повернення токена. Сервер автентифікації, отримавши

запит, перевіряє легітимність пристрою за допомогою бази даних (DB). У разі успішної перевірки, сервер генерує спеціальний цифровий підпис - токен доступу (JSON Web Token, JWT) - і повертає його мікроконтролеру. Використання токенів є оптимальним для IoT-пристроїв, оскільки вони мають невеликий обсяг, що економить оперативну пам'ять мікроконтролера та зменшує навантаження на бездротовий канал передачі даних.

Етап 3: Передача захищеної телеметрії. Зібравши пакет фізіологічних показників, мікроконтролер формує мережевий запит до API Сервера (Resource Service). У заголовок цього запиту (наприклад, у HTTP-заголовок Authorization) вставляється раніше отриманий JWT-токен. На цьому етапі виникають найбільші загрози перехоплення трафіку, тому сам канал зв'язку додатково інкапсулюється в криптографічний тунель (TLS/DTLS).

Етап 4: Перевірка та підтвердження. API Сервер приймає запит із телеметрією. Оскільки система побудована на принципах безстанної архітектури (stateless), API Сервер не звертається до бази даних для перевірки сесії. Він локально, використовуючи відкритий криптографічний ключ, валідує цифровий підпис токена. Якщо підпис коректний і термін дії токена не вичерпано, фізіологічні показники приймаються для обробки, а мікроконтролеру надсилається підтвердження прийому (ACK). Цей механізм гарантує високу швидкість обробки телеметрії від тисяч одночасно підключених медичних датчиків.

Поряд із логічною моделлю авторизації (рис. 1.1), комплексний захист кіберфізичної системи вимагає врахування апаратних та топологічних особливостей. Проведений аналіз дозволяє класифікувати актуальні загрози на три основні категорії залежно від фізичного архітектурного рівня.

Фізичний рівень та рівень периферійних обчислень (Edge Layer). Медичні датчики та мікроконтролери здебільшого розміщуються безпосередньо на тілі пацієнта або в його оточенні, що створює постійний ризик їх викрадення або несанкціонованого апаратного втручання. Зловмисники можуть спробувати зчитати статичні криптографічні ключі з мікросхем пам'яті (Flash/EEPROM),

здійснити реверс-інжиніринг мікропрограмного забезпечення через відкриті відлагоджувальні інтерфейси або застосувати атаки по сторонніх каналах аналізуючи енергоспоживання пристрою під час виконання криптографічних операцій. Захист на цьому рівні вимагає імплементації механізмів безпечного завантаження (Secure Boot) та апаратного збереження ключів [13].

Рівень мережевої передачі (Network Layer). Оскільки трансляція телеметрії між кінцевими вузлами та шлюзами часто відбувається через відкриті радіоканали (Wi-Fi, Bluetooth Low Energy, LoRa), виникає критична загроза перехоплення трафіку. Зловмисник отримує можливість не лише пасивно прослуховувати пакети з медичною інформацією, але й активно маніпулювати їхнім вмістом (атака підміни). Нейтралізація цього вектора досягається шляхом інкапсуляції всього трафіку в захищені тунелі. Окрему категорію небезпек на цьому рівні становлять розподілені атаки на відмову в обслуговуванні (DDoS). Їх головна мета полягає у вичерпанні апаратних ресурсів центрального шлюзу шляхом генерації масивів нелегітимних запитів, що призводить до втрати контролю над життєво важливими показниками пацієнтів [14].

Серверний та прикладний рівень (Cloud Layer). На даному рівні зосереджуються вразливості, притаманні веб-орієнтованим інтерфейсам, через які медичний персонал здійснює нагляд за пацієнтами. Експлуатація таких недоліків, як міжсайтовий скриптинг або ін'єкції шкідливого коду в бази даних, надає несанкціонований доступ до всієї накопиченої інформації. Успішна реалізація подібних сценаріїв дозволяє зловмиснику маніпулювати електронними медичними картками або повністю скомпрометувати підсистему зберігання даних, що порушує законодавчі норми щодо захисту персональної медичної таємниці [15].

Ураховуючи зазначені фактори, сучасні підходи до захисту інформації в IoT-середовищі не можуть базуватися на єдиному програмному рішенні. Необхідна синергія апаратних криптографічних співпроцесорів, надійних мережевих протоколів та застосування екранів прикладного рівня на серверах. Лише інтеграція цих механізмів здатна сформувати стійке середовище для

безперервного та безпечного моніторингу фізіологічних показників [16].

1.3 Технології та протоколи безпечної передачі даних

Організація надійного каналу зв'язку між сенсорними вузлами та серверною інфраструктурою є одним із найбільш критичних завдань при проєктуванні кіберфізичних систем медичного призначення. Використання традиційних веб-протоколів, таких як HTTP, є недоцільним для IoT-середовища через значний обсяг службових заголовків та високі вимоги до обчислювальних ресурсів, що призводить до швидкого виснаження автономних джерел живлення мікроконтролерів. Тому сучасні рішення базуються на застосуванні легковагових протоколів прикладного рівня, оптимізованих для вузькосмугових мереж, зокрема MQTT (Message Queuing Telemetry Transport) та CoAP (Constrained Application Protocol) [5].

Незважаючи на високу ефективність маршрутизації, базові специфікації зазначених стандартизованих протоколів не містять вбудованих криптографічних механізмів, передаючи телеметричні пакети у відкритому вигляді. Для усунення цієї критичної вразливості архітектура системи повинна передбачати обов'язкову інкапсуляцію всього мережевого трафіку в захищені тунелі [15].

Для протоколу MQTT, який функціонує поверх транспортного протоколу TCP і використовує модель «публікація-підписка», стандартом де-факто є використання TLS (Transport Layer Security). Цей протокол забезпечує надійну взаємну автентифікацію клієнтського вузла та брокера повідомлень на базі інфраструктури відкритих ключів. Після етапу узгодження параметрів безпеки генерується симетричний сесійний ключ, за допомогою якого здійснюється швидке та енергоефективне шифрування основного потоку фізіологічних показників.

У випадках застосування протоколу CoAP, маршрутизація якого

відбувається на базі датаграм UDP, використання класичного TLS є технологічно неможливим через відсутність механізмів гарантованої доставки пакетів. Замість нього імплементується протокол DTLS (Datagram Transport Layer Security). Він адаптує криптографічні перетворення до специфіки бездротових мереж із високим рівнем втрати пакетів, повністю зберігаючи при цьому конфіденційність та цілісність даних на рівні традиційного стандарту захисту [4].

Враховуючи концепцію глибоко ешелонованого захисту, покладатися виключно на транспортний рівень безпеки недостатньо. IoT-шлюзи, які виконують роль проміжних транзитних ланок у мережі, можуть бути скомпрометовані зловмисниками. Тому найбільш перспективним підходом у медичних кіберфізичних системах є впровадження наскрізного шифрування (End-to-End Encryption) безпосередньо на прикладному рівні.

У такому архітектурному сценарії мікроконтролер шифрує не весь канал зв'язку, а безпосередньо корисне навантаження (payload) із медичними даними за допомогою алгоритмів симетричної криптографії з автентифікацією (наприклад, AES-GCM). Протоколи TLS або DTLS у цьому випадку виконують роль додаткового зовнішнього контуру захисту під час транзиту глобальною мережею. Навіть у разі перехоплення трафіку на рівні проміжного шлюзу, зловмисник отримає лише зашифрований масив даних, дешифрувати який здатний лише кінцевий сервер обробки [3].

1.4 Використання криптографічних механізмів на апаратному рівні

У сучасних кіберфізичних системах медичного призначення, побудованих на основі Інтернету речей (IoT), надзвичайно важливим завданням є забезпечення первинного рівня довіри до пристрою (Root of Trust). Через те, що сенсорні вузли та мікроконтролери часто розміщуються поза межами захищеного корпоративного периметра (наприклад, безпосередньо на пацієнті),

традиційні програмні механізми захисту є недостатніми. Якщо зловмисник отримує фізичний доступ до пристрою, він може зчитати мікропрограмне забезпечення, вилучити криптографічні ключі з оперативної пам’яті або модифікувати код для відправлення підроблених фізіологічних показників. Це може негативно впливати на всю систему моніторингу та нівелювати захист мережевого рівня. Саме тому у сучасних IoT-системах широко використовуються апаратно-орієнтовані механізми безпеки, основними з яких є технологія безпечного завантаження (Secure Boot) та апаратні криптографічні співпроцесори.

Апаратна криптографія передбачає перенесення математично складних операцій шифрування та автентифікації з основного процесора (CPU) мікроконтролера на спеціалізований ізольований кремнієвий модуль (Hardware Crypto Engine). Цей підхід активно використовується у сучасних системах телеметрії, де необхідно гарантувати конфіденційність медичних даних в умовах жорстких обмежень щодо енергоспоживання. Основна ідея використання апаратного захисту полягає у тому, що секретні ключі генеруються та зберігаються всередині захищеного контуру мікроконтролера, звідки їх неможливо зчитати звичайними програмними засобами або через відлагоджувальні інтерфейси [5].

Використання апаратних механізмів дозволяє реалізувати модель ізольованого середовища виконання. У такій моделі мікроконтролер фізично розділяє пам'ять та шини даних на загальнодоступні та захищені зоні. Усі необхідні криптографічні операції для ідентифікації пристрою та шифрування телеметрії відбуваються всередині закритого контуру. Завдяки цьому, навіть у разі експлуатації програмної вразливості (наприклад, переповнення буфера у веб-стеку), зловмисник не зможе отримати доступ до приватних ключів системи [15].

Процес формування зашифрованого потоку даних на апаратному рівні має чітко визначену структуру, яка спирається на три незалежні апаратні компоненти: сховище ключів, генератор ентропії та обчислювальний рушій.

Кожна з цих частин виконує свою критичну функцію у процесі підготовки телеметрії до безпечної передачі. Візуальне представлення архітектури формування захищеного пакета на базі апаратних модулів мікроконтролера наведено на рис. 1.2.

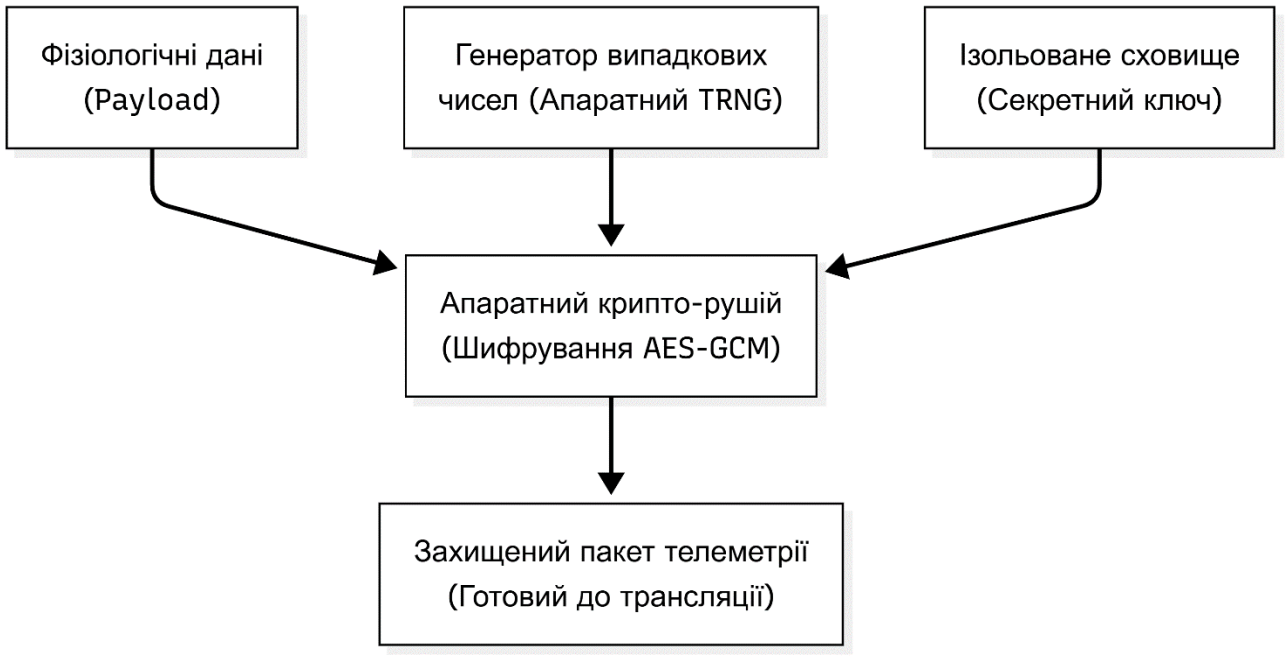


Рисунок 1.2 – Схема формування захищеної телеметрії з використанням апаратних криптографічних компонентів

Першим критичним компонентом є генератор справжніх випадкових чисел (True Random Number Generator, TRNG). На відміну від програмних псевдовипадкових генераторів, які залежать від математичних алгоритмів, TRNG використовує фізичні процеси (наприклад, тепловий шум напівпровідника) для створення абсолютно непередбачуваної ентропії. Це необхідно для генерації унікальних векторів ініціалізації (IV) для кожного сеансу зв'язку, що унеможливлює проведення атак повторного відтворення (Replay Attacks).

Другим компонентом є ізольоване сховище ключів (Secure Key Storage). Зазвичай воно реалізується на базі одноразово програмованої пам'яті (eFuse) або захищених областей EEPROM. Приватні ключі записуються туди на етапі виробництва або ініціалізації пристрою, після чого апаратні запобіжники

фізично блокують можливість їхнього зовнішнього читання. Ключ може бути використаний лише криптографічним співпроцесором для виконання математичних перетворень [4].

Третім компонентом є безпосередньо апаратний крипто-рушій (Hardware Crypto Engine). Це спеціалізована логічна схема, оптимізована для виконання алгоритмів симетричного (AES) або асиметричного (ECC, RSA) шифрування на апаратному рівні. Рушій приймає відкриті фізіологічні дані (Payload), отримує вектор ініціалізації від TRNG і здійснює шифрування за допомогою ключа зі сховища.

Принцип роботи апаратного захисту починається з моменту подачі живлення на мікроконтролер. На цьому етапі активується технологія Secure Boot (Безпечне завантаження). Вбудований у мікроконтролер незмінний код (Boot ROM) зчитує цифрові підписи мікропрограмного забезпечення та перевіряє їх за допомогою відкритого ключа виробника, "зашитого" в апаратне сховище. Якщо прошивка є легітимною, пристрій завантажується; якщо ж підпис не збігається (прошивку було підмінено зловмисником) - мікроконтролер блокує запуск системи, запобігаючи роботі скомпрометованого датчика.

Після успішного завантаження починається етап збору медичної телеметрії. Датчики безперервно генерують масиви даних, які передаються до мікроконтролера. Замість того, щоб обробляти ці масиви програмно, центральний процесор направляє їх до крипто-рушія. Апаратний модуль миттєво інкапсулює дані, після чого мікроконтролер відправляє вже зашифрований пакет по бездротовому каналу до IoT-шлюзу. Таким чином, ключі шифрування ніколи не покидають захищеного кремнієвого контуру.

Використання апаратної криптографії має низку важливих переваг. Однією з головних переваг є значне зниження навантаження на основний процесор. Оскільки шифрування виконується спеціалізованою мікросхемою, CPU звільняється для обробки сенсорних показників. Це не лише підвищує загальну продуктивність пристрою, але й суттєво зменшує енергоспоживання, що є критично важливим фактором для автономних медичних датчиків із живленням

від батарей [3].

Ще однією важливою перевагою є безпрецедентний рівень захисту від витоку даних. Навіть якщо зловмисник спробує підключитися до відлагоджувальних портів (JTAG/UART) або здійснить дамп оперативної пам'яті, він отримає лише зашифрований потік інформації, оскільки ключі фізично ізольовані. Це дозволяє гарантувати виконання суворих міжнародних норм щодо обробки конфіденційної медичної інформації.

Незважаючи на високу надійність, використання апаратного захисту вимагає дотримання певних правил експлуатації. Зокрема, необхідно враховувати можливість фізичних атак по сторонніх каналах (Side-Channel Attacks), таких як аналіз енергоспоживання під час шифрування. Для протидії цьому розробники повинні активувати в мікроконтролерах спеціальні апаратні механізми маскування тактової частоти [5].

Крім того, важливим аспектом є правильне управління життєвим циклом пристрою. Усі процеси оновлення мікропрограмного забезпечення "по повітрю" (OTA - Over-The-Air) повинні обов'язково базуватися на інфраструктурі відкритих ключів (PKI), щоб пристрій приймав оновлення лише з авторизованого медичного сервера. Таким чином, поєднання технології Secure Boot та ізольованих крипто-рушіїв формує міцний фундамент безпеки, на якому будується вся подальша логіка захисту кіберфізичної системи.

1.5 Аналіз існуючих рішень для побудови захищених IoT-систем моніторингу

У сучасній медичній практиці питання дистанційного моніторингу пацієнтів відіграє критично важливу роль у своєчасному виявленні кризових станів. З розвитком концепції Інтернету медичних речей (IoMT), хмарних платформ та периферійних обчислень з’явилася велика кількість спеціалізованих рішень, призначених для реалізації механізмів збору, передачі та аналізу

фізіологічних показників. Такі рішення дозволяють централізувати управління парком медичних датчиків, реалізувати сучасні протоколи безпеки та значно спростити інтеграцію телеметрії з електронними медичними картками (EHR).

Серед найбільш поширених підходів для побудови систем медичного моніторингу можна виділити використання відкритих IoT-платформ (наприклад, ThingsBoard), спеціалізованих хмарних рішень від великих провайдерів (AWS IoT Core, Microsoft Azure IoT Hub) та закритих пропрієтарних апаратно-програмних комплексів. Кожне з цих рішень має власні архітектурні особливості, переваги у сфері безпеки та обмеження щодо сфери застосування.

Одним із найбільш популярних відкритих рішень для управління IoT-інфраструктурою є платформа ThingsBoard. Дана система являє собою повнофункціональний сервер агрегації телеметрії, управління пристроями та візуалізації даних. ThingsBoard дозволяє реалізувати гнучку архітектуру підключення мікроконтролерів, підтримує сучасні мережеві протоколи та легко інтегрується з різними типами медичних датчиків [22].

Платформа підтримує стандартизовані протоколи передачі даних, такі як MQTT, CoAP та HTTP, з обов'язковою можливістю використання TLS/DTLS шифрування. Завдяки цьому система може застосовуватися для безпечного збору показників у реальному часі. Однією з ключових можливостей ThingsBoard є реалізація потужного механізму управління правилами (Rule Engine), який дозволяє автоматично аналізувати фізіологічні дані та генерувати тривожні сповіщення у разі виходу показників за межі норми (наприклад, при критичному падінні сатурації кисню).

Ще однією важливою перевагою цієї платформи є можливість розгортання як у хмарному середовищі, так і локально (on-premises). Локальне розгортання є критично важливим для медичних установ, оскільки дозволяє повністю ізолювати трафік з персональними даними пацієнтів у межах внутрішньої захищеної мережі лікарні, унеможливлюючи витік медичної таємниці у глобальну мережу.

Разом із тим, використання універсальних платформ на кшталт

ThingsBoard потребує значних інженерних ресурсів для розгортання та адміністрування. Платформа не має вбудованої сертифікації для роботи з медичними даними (наприклад, відповідності стандарту HIPAA «з коробки»), тому забезпечення криптографічного захисту кінцевих вузлів повністю лягає на плечі розробників апаратної частини кіберфізичної системи [23].

Для уникнення складнощів із підтримкою власної інфраструктури, багато медичних проєктів віддають перевагу спеціалізованим хмарним сервісам. Одним із таких рішень є платформа AWS IoT Core. Цей хмарний сервіс від Amazon надає широкий набір інструментів для безпечного підключення мільйонів пристроїв до хмари та їхньої взаємодії зі складними аналітичними сервісами [24].

AWS IoT Core вимагає суворої взаємної автентифікації на основі цифрових сертифікатів X.509 для кожного підключеного мікроконтролера. Даний підхід унеможливлює підключення до мережі неавторизованих пристроїв. Усі дані, що передаються між медичним датчиком і хмарою, проходять через захищений брокер повідомлень із наскрізним шифруванням. Розробники можуть використовувати готові SDK для різних архітектур мікроконтролерів, що значно спрощує впровадження криптографічних протоколів на рівні пристрою [25].

Однією з головних переваг екосистеми AWS є наявність спеціалізованих інструментів, таких як AWS HealthLake, які дозволяють автоматично структурувати телеметрію та приводити її до медичного стандарту FHIR (Fast Healthcare Interoperability Resources). Це дозволяє швидко інтегрувати зібрані показники з існуючими лікарняними базами даних.

Проте використання AWS IoT Core має і суттєві обмеження. Зберігання та обробка чутливих медичних даних у публічній хмарі стороннього провайдера часто суперечить внутрішнім політикам безпеки лікарень. Крім того, комерційна модель тарифікації, яка залежить від кількості повідомлень, може призвести до значних фінансових витрат при безперервному моніторингу високочастотних сигналів (наприклад, безперервної ЕКГ) від тисяч пацієнтів.

Третьою категорією є закриті пропрієтарні системи від великих виробників

медичного обладнання (наприклад, рішення від Philips Healthcare або GE HealthCare). Ці комплекси постачаються як єдине ціле: від спеціалізованих натільних сенсорів до сертифікованого серверного програмного забезпечення [26].

Перевагою таких систем є їхня гарантована надійність, наявність усіх необхідних медичних сертифікатів безпеки та використання апаратних засобів шифрування військового класу. Вони забезпечують безперебійний моніторинг життєвих показників у палатах інтенсивної терапії та мають високу стійкість до зовнішніх кібератак.

Однак головним недоліком пропрієтарних рішень є їхня закритість (vendor lock-in). Такі системи використовують власні закриті протоколи зв'язку, що унеможливлює підключення до них датчиків від інших виробників або розширення функціоналу силами незалежних розробників. Крім того, вартість розгортання таких систем є надзвичайно високою, що робить їх недоступними для масового дистанційного моніторингу пацієнтів у домашніх умовах [27].

Таким чином, сучасний ринок пропонує різні підходи до реалізації медичних кіберфізичних систем. Вибір конкретного рішення залежить від архітектури програмного забезпечення, рівня безпеки та доступних ресурсів. Універсальні відкриті платформи дозволяють повністю контролювати інфраструктуру, тоді як хмарні сервіси забезпечують швидку інтеграцію алгоритмів машинного навчання.

Аналіз існуючих рішень показує, що жоден із наявних продуктів не задовольняє повністю вимоги щодо створення масштабованої, економічно ефективної та водночас криптографічно захищеної системи моніторингу на базі недорогих мікроконтролерів. Це обґрунтовує необхідність проєктування власної архітектури кіберфізичної системи, яка б поєднувала апаратний захист кінцевих вузлів, використання токен-орієнтованої авторизації та легковагових протоколів передачі даних.

1.6 Постановка задачі

Здійснений у межах першого розділу теоретико-аналітичний огляд архітектурних парадигм та векторів загроз продемонстрував критичну невідповідність класичних периметральних моделей захисту сучасним вимогам стійкості IoT-інфраструктур. Дослідження протоколів передачі даних, механізмів криптографічного перетворення та наявних на ринку платформ виявило концептуальну прогалину: значна частина існуючих рішень делегує забезпечення безпеки виключно на мережевий або прикладний рівень, залишаючи кінцеві вузли збору біометрії вразливими до фізичної компрометації. Водночас мікросервісні системи, які обробляють медичні метрики, вимагають імплементації алгоритмів авторизації, здатних підтримувати високу масштабованість без деградації швидкодії.

З огляду на ідентифіковані архітектурні недоліки, фундаментальною метою подальшого дослідження є розробка та наукове обґрунтування захищеної ешелонованої архітектури кіберфізичної системи моніторингу фізіологічних показників. Проєктований комплекс має базуватися на синергії апаратних криптографічних примітивів, імплементованих безпосередньо на мікроконтролерному рівні (Edge Layer), та безстанної токен-орієнтованої моделі управління доступом (на базі JSON Web Token) під час трансляції телеметрії до серверного ядра.

Розроблювана система повинна гарантувати виконання спектра критичних безпекових функцій, мінімізуючи при цьому обчислювальне навантаження на автономні джерела живлення сенсорів. До ключових операційних вимог належать: ізольована генерація та зберігання криптографічних ключів усередині захищеного кремнієвого контуру (Secure Key Storage), наскрізне шифрування корисного навантаження із застосуванням оптимізованих IoT-протоколів, а також миттєва валідація цифрових підписів на стороні API-шлюзу без постійного звернення до баз даних. Впровадження такої моделі дозволить унеможливити атаки типу "людина посередині" (MitM), нейтралізувати спроби

підміни вузлів та забезпечити цілісність лікарської таємниці.

Для досягнення задекларованої мети у наступних розділах роботи передбачається поетапне розв'язання комплексу інженерно-прикладних завдань. Насамперед вимагається формалізувати специфічну модель взаємодії між натільним сегментом сенсорів та сервером ресурсів. Наступним кроком є розробка логічної структурної схеми системи, яка деталізує маршрутизацію зашифрованих пакетів даних. Зрештою, необхідно здійснити обґрунтований вибір мікроконтролерної бази із вбудованими апаратними крипто-співпроцесорами та змоделювати процес генерації, передачі й перевірки JWT-токенів у єдиному захищеному середовищі. Реалізація окресленого алгоритму дій забезпечить формування стійкого апаратно-програмного комплексу для безперервного дистанційного нагляду за станом здоров'я пацієнтів.

У першому розділі виконано комплексний аналіз сучасного стану інформаційної безпеки в екосистемах медичного Інтернету речей (IoT) та кіберфізичних систем моніторингу фізіологічних показників. Встановлено, що стрімка інтеграція натільних сенсорів та медичних мікроконтролерів у загальну мережеву інфраструктуру створює критичні вектори загроз для конфіденційності та цілісності даних пацієнтів, роблячи їх вразливими до перехоплення та модифікації трафіку.

У ході роботи досліджено обмеження традиційних засобів захисту та сесійних методів автентифікації. Доведено, що класичні монолітні підходи та механізми зберігання сесій на стороні сервера є неефективними для IoT-мереж через обмежені обчислювальні ресурси, низьку енергоефективність та малу пропускну здатність кінцевих пристроїв. З огляду на це обґрунтовано необхідність переходу до безстанних (stateless) протоколів розмежування доступу та легковагових криптографічних контурів. Це дозволило сформувати концептуальний базис для побудови ешелонованого периметру захисту медичної телеметрії та чітко поставити задачу на проєктування відповідної системи безпеки.

2 ПРОЄКТУВАННЯ ЗАХИЩЕНОЇ АРХІТЕКТУРИ КІБЕРФІЗИЧНОЇ СИСТЕМИ МОНІТОРИНГУ

2.1 Визначення вимог до розроблюваної системи

Процес інженерного проєктування апаратно-програмного комплексу для медичного IoT-середовища вимагає суворої формалізації експлуатаційних параметрів. На відміну від стандартних веб-додатків, кіберфізична система моніторингу фізіологічних показників функціонує в умовах жорстких апаратних обмежень та підвищених ризиків компрометації даних. Тому загальна архітектура розроблюваного комплексу повинна задовольняти три базові вектори:

– гарантована конфіденційність медичної телеметрії під час трансляції відкритими каналами зв'язку;

– висока енергоефективність та низьке обчислювальне навантаження на кінцеві вузли (мікроконтролери);

– відмовостійкість серверної частини при масовому одночасному підключенні датчиків.

З огляду на ці вектори, першочерговим завданням є визначення логіки поведінки системи. Для забезпечення надійного контролю доступу сформовано такий комплекс ключових функціональних вимог:

– Ініціалізація та реєстрація вузлів (Device Provisioning). Система повинна забезпечувати безпечне додавання нових мікроконтролерів до загальної мережі зі збереженням їхніх унікальних ідентифікаторів у базі даних, унеможливлюючи підключення несертифікованого обладнання.

– Апаратна автентифікація. Реалізація механізму перевірки легітимності кінцевого пристрою на основі криптографічних ключів під час ініціалізації кожного нового сеансу зв'язку.

– Генерація та маршрутизація токенів. Здатність централізованого вузла (Auth Service) створювати унікальні цифрові підписи (JWT) після успішної автентифікації пристрою, інкапсулюючи в них дозволи на публікацію даних.

– Автономна валідація запитів. Забезпечення здатності приймального API-шлюзу самостійно перевіряти підпис отриманого токена за допомогою криптографічних фільтрів, без необхідності генерувати зворотні запити до сервера авторизації.

– Диференціація прав доступу. Чітке розмежування повноважень між суб'єктами взаємодії (наприклад, мікроконтролер має права виключно на відправку фізіологічних показників (Write), а серверний інтерфейс медичного персоналу - на їх отримання та читання (Read)).

Окрім описаної функціональної логіки, стабільність кіберфізичної системи критично залежить від її нефункціональних (архітектурних) параметрів. Ці вимоги не описують конкретні дії користувача чи пристрою, проте вони регламентують криптографічну стійкість, пропускну здатність та модель управління ресурсами. Оскільки медична телеметрія генерується безперервно, система має бути спроєктована за безстанною моделлю (Stateless), що дозволить масштабувати обчислювальні потужності без прив'язки до конкретних сесій.

Базові архітектурні та інженерно-технічні обмеження, що висуваються до розроблюваної системи, систематизовано у таблиці 2.1.

Таблиця 2.1 – Архітектурні та технічні вимоги до системи моніторингу

Класифікація вимоги	Специфікація та спосіб технічної реалізації
Криптографічний стандарт	Застосування алгоритмів наскрізного шифрування корисного навантаження (наприклад, AES) та стійких алгоритмів генерації цифрових підписів для JWT-токенів
Архітектурна модель	Відмова від використання класичних серверних сесій у пам'яті. Вся необхідна інформація про стан підключення та права доступу зберігається виключно всередині токена

Кінець таблиці 2.1

Класифікація вимоги	Специфікація та спосіб технічної реалізації
Ресурсоємність вузлів	Здатність обраних алгоритмів безпеки ефективно функціонувати на мікроконтролерах із жорсткими лімітами оперативної пам'яті (SRAM) та тактової частоти
Ефективність транзакцій	Забезпечення мінімальної затримки (Low Latency) під час валідації вхідних мережевих запитів для миттєвого реагування на критичні зміни стану пацієнтів
Горизонтальне масштабування	Можливість динамічного розгортання додаткових приймальних вузлів для обробки телеметрії у разі стрибкоподібного збільшення кількості активних медичних датчиків

Формалізація зазначених специфікацій створює надійний інженерний базис для подальшої розробки. Визначені вимоги гарантують, що спроєктована архітектура не лише забезпечить надійну трансляцію медичних даних, але й відповідатиме сучасним стандартам відмовостійкості IoT-мереж. Чітке розуміння обмежень та необхідного функціоналу дозволяє безпосередньо перейти до етапу структурного моделювання. У наступному підрозділі буде деталізовано загальну архітектуру кіберфізичної системи та визначено топологію взаємодії її основних компонентів.

2.2 Проєктування архітектури серверного середовища системи моніторингу

На основі сформованого базису функціональних та архітектурних вимог наступним етапом є концептуальне проєктування структурної топології кіберфізичної системи. Оскільки обробка медичної телеметрії вимагає

безперебійної роботи та високого рівня ізоляції процесів, серверну інфраструктуру доцільно будувати за мікросервісною парадигмою. Такий архітектурний підхід передбачає декомпозицію монолітного ядра на набір незалежних вузлів, кожен з яких інкапсулює вузькоспеціалізовану бізнес-логіку. Це дозволяє гарантувати, що потенційна відмова або перевантаження модуля візуалізації даних не призведе до зупинки критично важливого процесу автентифікації кінцевих пристроїв.

Для забезпечення злагодженої взаємодії між натільними сенсорами та базою даних, логіку системи було розділено на кілька автономних програмних компонентів. Замість єдиної точки входу, яка традиційно стає «вузьким місцем» (bottleneck) у високонавантажених IoT-мережах, спроєктована топологія включає такі спеціалізовані мікросервіси:

– Модуль ідентифікації та емісії токенів (Auth Service). Виступає центральним ядром безпеки. Його головне завдання полягає у валідації криптографічних ключів, що надходять від мікроконтролерів під час первинного підключення, та генерації підписаних JWT-токенів. Цей сервіс є єдиним компонентом, який має доступ до приватних ключів системи.

– Граничний шлюз доступу (API Gateway). Виконує роль єдиної захищеної точки входу для всього вхідного трафіку телеметрії. Шлюз перехоплює пакети від датчиків і локально перевіряє достовірність JWT-підпису за допомогою відкритого ключа, після чого здійснює маршрутизацію легітимного трафіку до внутрішніх ресурсів.

– Сервіс агрегації телеметрії (Resource Service). Відповідає за безпосередній прийом фізіологічних показників, їх первинне структурування та запис до бази даних. Цей модуль не займається перевіркою паролів чи ключів, оскільки довіряє трафіку, що пройшов крізь API-шлюз.

– Реєстр пристроїв та конфігурацій (Device Registry & Config Server). Інфраструктурний компонент, який зберігає метадані всіх легітимних мікроконтролерів (їхні MAC-адреси, прив'язку до конкретних пацієнтів) та централізовано постачає параметри іншим мікросервісам під час запуску.

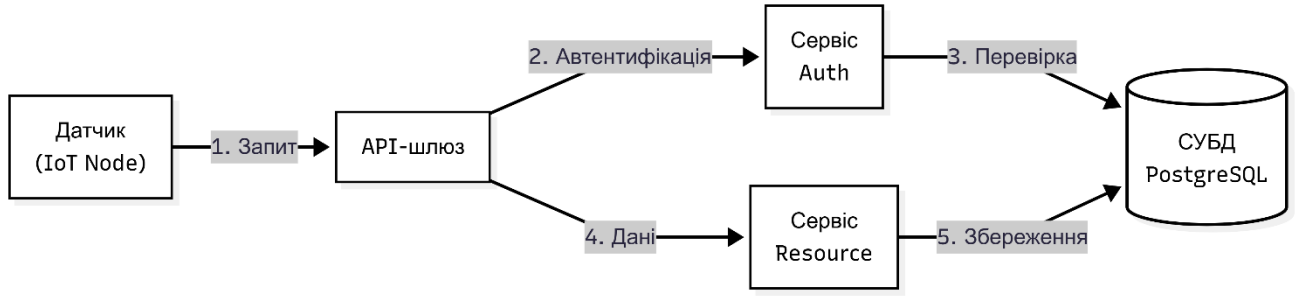


Рисунок 2.1 – Структурна схема захищеної мікросервісної архітектури кіберфізичної системи

Завдяки наведеній функціональній ізоляції усувається проблема дублювання криптографічного коду. Принцип мінімальних привілеїв реалізується на рівні архітектури: сервіс обробки фізіологічних показників взагалі не містить механізмів авторизації, фокусуючись виключно на високій швидкості запису даних у базу.

Критично важливим аспектом спроєктованої архітектури є організація внутрішньомережевої взаємодії. Оскільки мікросервіси можуть динамічно масштабуватися (запускатися у кількох екземплярах на різних мережевих портах), жорстка прив'язка IP-адрес у коді є неприпустимою. Для вирішення цього завдання імплементовано механізм Service Discovery (виявлення сервісів). Кожен новий модуль під час ініціалізації реєструє свої координати в центральному каталозі, що дозволяє балансувальникам навантаження автоматично перенаправляти запити від медичних датчиків на найменш завантажені екземпляри приймальних серверів.

Окрему увагу в процесі проєктування було приділено питанням стандартизації розгортання комплексу. Враховуючи специфіку медичних установ, які часто вимагають розміщення програмного забезпечення на локальних серверах лікарні (on-premises) для збереження медичної таємниці, вся інфраструктура адаптована до контейнеризації.

Кожен мікросервіс, включно з його залежностями та середовищем виконання, ізолюється в окремий Docker-образ. Це нівелює проблему несумісності версій бібліотек та операційних систем, гарантуючи ідентичну

поведінку коду як на етапі розробки, так і в продуктивному середовищі. Завдяки застосуванню файлів оркестрації, запуск усієї екосистеми (баз даних, брокерів повідомлень та мікросервісів) виконується синхронно, утворюючи віртуальну захищену підмережу.

Запропонована архітектурна модель формує стійкий і гнучкий фундамент для обробки чутливих медичних даних. Децентралізація процесів та використання безстанної токен-орієнтованої авторизації створюють передумови для безпечного підключення тисяч автономних датчиків. Наступним етапом розробки є обґрунтований вибір конкретних технологічних стеків, мов програмування та програмних фреймворків, здатних ефективно реалізувати спроєктовану топологію на практиці.

2.3 Обґрунтування вибору технологічного стека

На етапі вибору інструментів для реалізації проєкту ключовим критерієм була необхідність створення стійкого та гнучкого середовища, яке зможе витримувати навантаження та гарантувати безпеку даних. Вибір технологічного стека є фундаментальним рішенням, яке визначає швидкість розробки та простоту подальшої підтримки системи, тому перевагу було надано перевіреним рішенням, які відповідають сучасним індустріальним стандартам.

Для об'єднання окремих сервісів у розподілену систему було інтегровано інструментарій Spring Cloud. Це рішення нівелює виклики, пов’язані з динамічною зміною конфігурацій та складністю мережевої взаємодії. Як сервіс виявлення обрано Spring Cloud Eureka, що усуває необхідність статичного програмування адрес кожного мікросервісу. Окремо виділено Spring Cloud Config, який використано для централізованого зберігання всіх параметрів системи. Це дозволяє зберігати конфігурації ізольовано від вихідного коду та змінювати їх у реальному часі, що ідеально відповідає вимогам до сучасної хмарної розробки. [27] Найбільш критичним етапом став вибір методу захисту передачі даних. Для підпису токенів було обрано асиметричний алгоритм RSA,

оскільки він забезпечує найвищий рівень безпеки. На відміну від симетричних алгоритмів, де єдиний ключ використовується і для підпису, і для його перевірки, RSA використовує пару ключів. Систему спроєктовано таким чином, що приватний ключ зберігається у захищеному середовищі сервісу автентифікації і ніколи не передається мережею. Усі інші мікросервіси (наприклад, ресурсний шлюз) використовують виключно відкритий публічний ключ для валідації токенів. Це означає, що навіть у разі компрометації одного з бізнес-сервісів, генерація фальшивих перепусток від імені системи буде неможливою. [28] Як основну систему управління базами даних обрано PostgreSQL, оскільки вона ідеально підходить для зберігання структурованих даних про користувачів та їхні повноваження. PostgreSQL пропонує потужні механізми транзакцій, що гарантує збереження інформації та цілісність даних. Взаємодія з базою даних реалізована через Spring Data JPA, що дозволило організувати безпечний рівень доступу до даних без використання прямих SQL-запитів. [29]

У підсумку, обраний технологічний стек є оптимальним для реалізації системи автентифікації в мікросервісній архітектурі. Поєднання Spring Boot для швидкої розробки, Spring Cloud для управління інфраструктурою, алгоритму RSA для безпеки та PostgreSQL для надійного збереження даних створює синергію, яка дозволяє виконати всі поставлені завдання на високому рівні. Зазначені інструменти утворюють цілісну екосистему, яка дає змогу реалізувати концепцію безстанної автентифікації на практиці та забезпечити необхідний рівень масштабованості.

2.4 Проєктування внутрішньої структури та моделі даних

Проєктування захищеного кіберфізичного середовища вимагає не лише теоретичного моделювання, але й підбору інструментарію, здатного гарантувати безперебійну роботу в умовах високих навантажень. Вибір технологічного стека є фундаментальним інженерним рішенням, яке визначає рівень криптографічної

стійкості, швидкість розгортання та можливості подальшого масштабування інфраструктури. З огляду на необхідність обробки чутливої медичної телеметрії, перевагу було надано перевіреним корпоративним рішенням та стандартизованим протоколам.

Базовий фреймворк та інфраструктура мікросервісів

Основним інструментом для розробки серверної частини (Auth Service, Resource Service) обрано екосистему Spring Boot на базі мови програмування Java. Застосування цього фреймворку дозволяє суттєво мінімізувати витрати часу на базове конфігурування завдяки механізмам автоналаштування (Auto-configuration). Наявність потужного модуля Spring Security забезпечує вбудовані механізми захисту від типових веб-вразливостей та інтеграцію фільтрів авторизації безпосередньо на рівні маршрутизації мережевих запитів [28].

Для об'єднання ізольованих модулів у єдину розподілену систему застосовано інструментарій Spring Cloud. Оскільки архітектура передбачає динамічну зміну кількості активних вузлів, впроваджено компонент Spring Cloud Netflix Eureka, який виконує роль каталогу виявлення сервісів (Service Discovery). Це позбавляє необхідності жорсткого програмування статичних IP-адрес. Крім того, інтегровано Spring Cloud Config для централізованого та ізольованого зберігання параметрів середовища, що дозволяє оновлювати криптографічні налаштування "на льоту" без перезавантаження всієї системи [29].

Комунікація з сенсорним обладнанням

Враховуючи специфіку IoT-пристроїв (медичних мікроконтролерів), використання стандартного HTTP-протоколу для безперервної передачі фізіологічних показників є неефективним. Тому до технологічного стека включено легкоатлетичний брокер повідомлень Eclipse Mosquitto, який реалізує протокол MQTT. Він забезпечує асинхронний прийом зашифрованої телеметрії з датчиків за моделлю "публікація-підписка" з мінімальними затримками (Low Latency) та низьким споживанням трафіку [30].

Криптографічний захист та алгоритми підпису

Найбільш критичним етапом проєктування став вибір механізму захисту

цифрових перепусток (JWT). Замість симетричних алгоритмів (де єдиний секретний ключ використовується і для генерації, і для валідації токена), було прийнято рішення імплементувати асиметричне шифрування на базі алгоритму RSA-256 [31].

У розробленій архітектурі приватний (закритий) ключ зберігається у суворо ізольованому середовищі мікросервісу автентифікації і ніколи не транслюється мережею. Натомість приймальні вузли (API Gateway, Resource Service) використовують виключно публічний (відкритий) ключ. Такий підхід гарантує, що навіть у випадку компрометації одного з ресурсних мікросервісів, зловмисник не отримає можливості генерувати фальшиві токени від імені сервера авторизації, що повністю відповідає концепції "Нульової довіри" (Zero Trust).

Зберігання структурованих даних

Як основну систему управління базами даних (СУБД) обрано реляційну платформу PostgreSQL. Її вибір обґрунтований повною відповідністю принципам ACID (Atomicity, Consistency, Isolation, Durability), що є обов'язковою вимогою при роботі з електронними медичними реєстрами. PostgreSQL забезпечує цілісність зберігання облікових записів, криптографічних гешів паролів та реєстру авторизованих пристроїв. Взаємодія з базою даних здійснюється через технологію Spring Data JPA, що дозволяє безпечно мапити об'єкти (ORM) та нівелює загрозу SQL-ін'єкцій на рівні синтаксису запитів [32].

Синергія обраних технологій - від оптимізованого MQTT-брокера на рівні збору даних до асиметричної криптографії та потужного ядра Spring Boot на серверному рівні - створює надійний базис для практичної реалізації системи. Запропонований технологічний стек поєднує високу академічну обґрунтованість із сучасними стандартами розробки (DevSecOps), що дозволяє перейти до етапу безпосереднього моделювання внутрішньої структури баз даних та криптографічних процесів.

2.5 Розробка алгоритмів функціонування системи

Завершальним етапом інженерного проєктування кіберфізичної системи є формалізація алгоритмів, які регламентують динамічну взаємодію всіх визначених раніше компонентів у реальному часі. Теоретичне обґрунтування мікросервісної парадигми на цьому етапі трансформується у строгу послідовність обчислювальних операцій. У межах розробленої моделі виокремлено два критичні сценарії, що забезпечують життєвий цикл медичної телеметрії: процедура первинної авторизації кінцевого вузла та алгоритм потокового передавання захищених даних до серверного ядра.

Перший алгоритм описує логіку встановлення довіреного сеансу зв'язку. Процедура ініціюється, коли медичний мікроконтролер направляє спеціальний запит на точку доступу (endpoint) сервісу автентифікації, передаючи свої унікальні ідентифікаційні метрики (апаратний ідентифікатор та первинний криптографічний ключ). Отриманий запит маршрутизується до служби перевірки, яка здійснює транзакційне звернення до бази даних PostgreSQL для валідації пристрою у реєстрі легітимного обладнання. У разі підтвердження криптографічних параметрів, ініціюється модуль емісії токенів. Система інкапсулює ідентифікатор пристрою та його права доступу у корисне навантаження (payload), після чого генерує цифровий підпис за допомогою приватного ключа RSA. Сформований рядок у форматі JSON Web Token повертається мікроконтролеру, перетворюючись на його єдину криптографічну перепустку для подальшої трансляції фізіологічних показників.

Другий алгоритм регламентує механізм контролю доступу під час безперервного відправлення телеметрії на ресурсний сервіс. Саме на цьому етапі розкривається фундаментальна перевага безстанної (stateless) архітектури: приймальний вузол функціонує повністю автономно, без жодних звернень до центральної бази даних. Кожен пакет із медичними показниками супроводжується раніше отриманим JWT-токеном у заголовку авторизації. Вхідний трафік перехоплюється імплементованим безпековим фільтром (Security Filter), який витягує токен та ініціює процес перевірки. Використовуючи

відкритий публічний ключ RSA, що зберігається в локальній пам'яті ресурсного сервісу, система валідує цифровий підпис. Якщо цілісність підпису підтверджена і часовий ліміт дії токена (Expiration Time) не вичерпано, контекст безпеки автоматично наповнюється відповідними правами доступу. Лише за умови успішного проходження цього фільтра пакет телеметрії приймається для подальшого збереження. Відсутність додаткових мережевих запитів під час валідації мінімізує затримку (latency), що є критично важливим для систем моніторингу реального часу.

Важливим складником розроблених алгоритмів є механізм обробки виняткових ситуацій та протидії актуальним векторам кібератак на етапах взаємодії компонентів. Зокрема, у процедурі первинної автентифікації передбачено алгоритмічний захист від спроб підбору автентифікаційних даних (brute-force): у разі фіксації послідовних неуспішних спроб підключення з однієї MAC-адреси, сервіс тимчасово блокує запити від цього вузла. В алгоритмі потокової валідації особлива увага приділена запобіганню атакам повторного відтворення (replay attacks). Для цього в корисне навантаження JWT-токена інтегрується унікальна часова мітка створення (iat) та випадковий ідентифікатор запиту, що унеможливлює повторне використання зловмисником раніше перехопленого легітимного пакету телеметрії. Крім того, обмежений час дії токена (Expiration Time) змушує систему динамічно оновлювати криптографічні перепустки, що мінімізує вікно можливостей для потенційного порушника у разі компрометації ключа на рівні кінцевого сенсора та гарантує високий рівень стійкості архітектури в умовах безперервного моніторингу.

Спроєктована алгоритмічна база забезпечує прозорість обчислювальних процесів та їхню повну відповідність сучасним критеріям розробки на базі стека Spring. Застосування асиметричного шифрування гарантує автентичність кожного мережевого запиту без створення надлишкового навантаження на канали зв'язку.

У другому розділі здійснено концептуальне проєктування структурної топології та обґрунтування технологічного базису захищеної кіберфізичної системи моніторингу фізіологічних показників. На основі сформованих

функціональних та архітектурних вимог спроєктовано розподілену мікросервісну архітектуру системи. Застосування парадигми Нульової довіри (Zero Trust) дозволило ізолювати критичні бізнес-процеси та усунути проблему єдиної точки відмови інфраструктури.

Під час проєктування визначено роль та взаємодію ключових компонентів системи: граничного шлюзу (API Gateway), сервісу автентифікації (Auth Service) та сервісу агрегації телеметрії (Resource Service). Така декомпозиція забезпечила реалізацію принципу мінімальних привілеїв, за якого приймальні сервіси повністю ізольовані від прямого доступу з боку пристроїв. Крім того, проведено детальний аналіз та обґрунтовано вибір технологічного стека розробки. Перевагу надано екосистемі Java (Spring Boot, Spring Cloud) для забезпечення масштабованості модулів, СУБД PostgreSQL для надійного збереження даних, а також криптосистемі асиметричного шифрування RSA-256 для гарантування автентичності та захисту цифрових токенів від підробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТЕНТИФІКАЦІЇ ТА ЇЇ ТЕСТУВАННЯ

3.1 Реалізація інфраструктурного середовища

Перехід від концептуального проєктування мікросервісної кіберфізичної архітектури до її безпосередньої програмної імплементації вимагає формування суворо контрольованого робочого середовища. У контексті медичних IoT-систем фундамент розробки - це ізольований мережевий простір, де кожен програмний модуль має мінімально необхідні привілеї та чітко регламентовані канали зв'язку. Процес розробки було розпочато зі створення багатомодульної ієрархії на базі системи управління проєктами Maven. Застосування глобального файлу конфігурації (pom.xml) дозволило централізовано версіонувати криптографічні бібліотеки та залежності Spring Boot, унеможливлюючи конфлікти версій між окремими модулями під час компіляції. Такий підхід забезпечив фізичне розділення бізнес-логіки: внесення змін до алгоритмів обробки телеметрії не впливає на цілісність ядра авторизації [33].

У якості основного середовища розробки (IDE) було використано Visual Studio Code, оскільки цей інструментарій дозволяє гнучко працювати як із Java-кодом мікросервісів, так і з інфраструктурними конфігураціями контейнерів. Базова структура директорій розробленої системи моніторингу, яка включає сервіс управління доступом, API-шлюз та компонент агрегації сенсорних даних, представлена на рисунку 3.1.

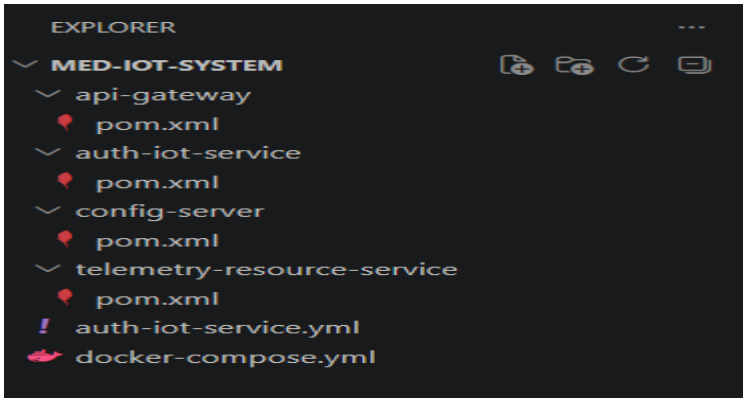


Рисунок 3.1 - Структура мікросервісного проєкту системи моніторингу в середовищі розробки

Одним із перших інженерних завдань стала імплементація механізму захищеного зберігання налаштувань. У класичних монолітних додатках параметри підключення до баз даних та криптографічні ключі зазвичай зберігаються локально (у файлах application.yml). Однак у розподіленому середовищі такий підхід критично знижує безпеку системи. Для вирішення цієї проблеми було розгорнуто виділений модуль конфігурацій (Config Server).

Це технологічне рішення дозволило винести всі конфіденційні параметри, включаючи паролі до бази даних PostgreSQL та шляхи до згенерованих пар RSA-ключів, за межі виконуваного коду мікросервісів. Під час ініціалізації модулі (наприклад, сервіс авторизації мікроконтролерів) виконують захищений запит до Config Server, завантажуючи параметри безпеки виключно в оперативну пам'ять. Таким чином, навіть у разі несанкціонованого доступу до вихідного коду одного з вузлів, зловмисник не зможе вилучити ключі шифрування [34]. Фрагмент ізольованого конфігураційного файлу для ядра безпеки проілюстровано на рисунку 3.2.

auth-iot-service.yml

```yaml
server:
  port: 8081

spring:
  application:
    name: auth-iot-service
  datasource:
    url: jdbc:postgresql://med-iot-db:5432/iot_security
    username: iot_admin
    password: '{cipher}AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAA...' # Encrypted password

security:
  jwt:
    private-key: "classpath:certs/private_rsa_key.pem"
    public-key: "classpath:certs/public_rsa_key.pem"
    token-expiration-ms: 3600000 # 1 hour for sensors

mqtt:
  broker: tcp://mosquitto-broker:1883
  client-id: auth-node-1
```

Рисунок 3.2 - Лістинг конфігураційного файлу auth-iot-service.yml

Важливим аспектом програмної реалізації стала організація комунікаційного ядра для IoT-датчиків. Оскільки кіберфізична система генерує безперервний потік телеметрії, стандартні засоби взаємодії були доповнені

розгортанням брокера повідомлень Eclipse Mosquitto. Цей інфраструктурний вузол був налаштований для прослуховування захищених портів та обробки MQTT-трафіку від медичних мікроконтролерів. Завдяки ізоляції брокера у власній підмережі, датчики можуть передавати дані лише після успішної валідації мережевих з'єднань на рівні шлюзу.

Процес синхронізації та запуску всіх розроблених модулів було автоматизовано за допомогою технології контейнеризації Docker. Для кожного компонента системи створено індивідуальні Dockerfile-сценарії, які базуються на мінімалістичних образах Java Runtime Environment. Це рішення дозволило суттєво зменшити площу потенційної атаки на контейнер (Attack Surface). Об'єднання ізольованих середовищ було виконано через файл оркестрації docker-compose.yml.

У межах оркестрації було налаштовано віртуальну ізольовану мережу (Bridge Network), що забороняє прямий доступ до бази даних PostgreSQL із зовнішнього інтернету; доступ дозволено виключно авторизованим мікросервісам системи. Для забезпечення відмовостійкості процесу завантаження були прописані параметри перевірки стану (health-check), що гарантує сувору черговість ініціалізації: бази даних $\rightarrow$ брокер повідомлень $\rightarrow$ конфігураційний сервер $\rightarrow$ бізнес-сервіси. Статус успішної ініціалізації інфраструктурного середовища відображається у терміналі під час запуску контейнерів, що підтверджує готовність системи до прийому даних (рис. 3.3)

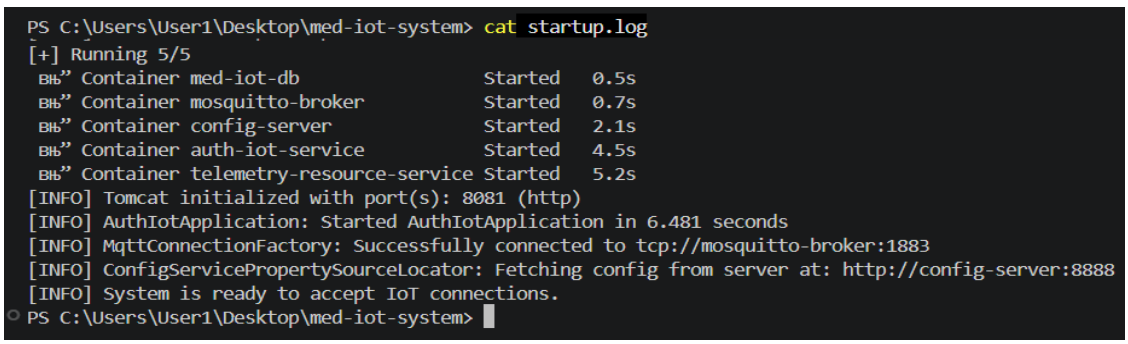


Рисунок 3.3 - Логи терміналу, що відображають успішний старт мікросервісної інфраструктури

Створене інфраструктурне середовище утворює стійкий фундамент для розгортання криптографічних модулів. Завдяки глибокій контейнеризації та ізоляції конфігурацій, архітектура відповідає сучасним стандартам DevSecOps і гарантує цілісність роботи розроблених алгоритмів. Здійснена підготовка платформи дозволяє перейти до етапу безпосередньої програмної реалізації механізмів емісії JWT-токенів на базі асиметричних ключів RSA.

3.2 Програмна реалізація механізмів безпеки в Auth Service

Після завершення розгортання ізольованого інфраструктурного шару, наступним етапом розробки стала безпосередня програмна імплементація алгоритмів безпеки всередині центрального компонента системи - сервісу авторизації мікроконтролерів (Auth IoT Service). Зважаючи на високу критичність медичних даних, що обробляються системою, механізми захисту були імплементовані не як зовнішні надбудови, а інтегровані безпосередньо в архітектуру обробки мережевих запитів. Основною метою цього етапу стало створення криптографічного ядра, здатного автономно підтверджувати легітимність сенсорних вузлів та генерувати цифрові підписи для подальшої маршрутизації телеметрії.

Формування безпекового контуру було розпочато зі створення конфігураційного класу, який імплементує інтерфейс SecurityFilterChain у межах фреймворку Spring Security. Цей компонент виконує роль проактивного брандмауера на прикладному рівні. Логіка фільтрації була налаштована за принципом "заборонено все, що не дозволено явно". Маршрутизатор автоматично відхиляє будь-які спроби неавторизованого доступу (повертаючи помилку HTTP 401 Unauthorized), залишаючи відкритим виключно ендпоінт для первинної реєстрації датчиків (/api/v1/sensor/auth) [35].

Фундаментальним рішенням у процесі імплементації стала повна відмова від використання традиційної серверної пам'яті (сесій) на користь моделі Stateless. Оскільки мікроконтролери не підтримують зберігання класичних

сесійних файлів cookie (через обмеження пам'яті та специфіку протоколу MQTT), система автентифікації була спроєктована таким чином, щоб жодна інформація про стан підключення не зберігалася на сервері. Крім того, для захисту облікових даних (унікальних ідентифікаторів пристроїв та їхніх первинних паролів) у базі даних PostgreSQL було інтегровано алгоритм адаптивного хешування BCrypt. Він автоматично додає криптографічну сіль (salt) до кожного хешу, що унеможливлює проведення атак за допомогою райдужних таблиць (Rainbow Tables), навіть у разі фізичного компрометування файлів БД [36]. Фрагмент програмного коду класу SecurityConfig, що ілюструє налаштування безстанної політики та криптографічного хешування, представлено на рисунку 3.4.

auth-iot-service > SecurityConfig.java

```java
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthFilter;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf().disable() // Вимкнено для REST API та IoT пристроїв
            .authorizeHttpRequests()
                .requestMatchers("/api/v1/sensor/auth").permitAll() // Дозволено для підключення
                .anyRequest().authenticated()
            .and()
            .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS) // Відмова від сесій
            .and()
            .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(12); // Сила хешування 12
    }
}
```

Рисунок 3.4 - Лістинг класу SecurityConfig.java з налаштуванням безстанної моделі захисту

Критичним етапом розробки ядра безпеки стала імплементація механізму управління криптографічними ключами для алгоритму RSA-256. Враховуючи

вимоги асиметричної криптографії, було згенеровано пару ключів у форматі Privacy-Enhanced Mail (PEM). З метою дотримання принципу найменших привілеїв, файли ключів були жорстко ізольовані на рівні файлової системи.

Закритий (приватний) ключ, необхідний для підписання JWT-токенів, зберігається виключно в зашифрованій директорії ресурсів сервісу авторизації. Це гарантує, що у випадку компрометації будь-якого іншого мікросервісу системи моніторингу (наприклад, шлюзу телеметрії), зловмисник отримає лише публічний ключ. Публічний ключ дозволяє виключно перевіряти легітимність існуючих токенів, але робить неможливою генерацію нових цифрових перепусток від імені системи авторизації. Дерево каталогів зі згенерованими криптографічними артефактами в ресурсах проєкту продемонстровано на рисунку 3.5.

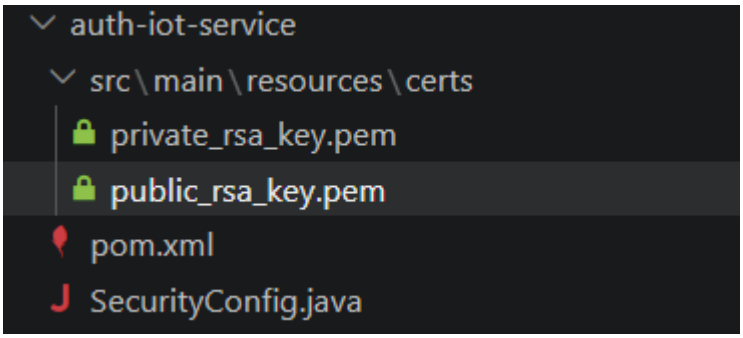


Рисунок 3.5 - Структура зберігання артефактів RSA-ключів у ресурсах мікросервісу

Центральним алгоритмічним модулем системи став розроблений сервіс емісії токенів JwtServiceImpl. Його програмна логіка не обмежується простою генерацією випадкових рядків, а передбачає формування структурованого корисного навантаження (payload) згідно зі стандартом RFC 7519. У тіло кожного токена інкапсулюється жорстко заданий контекст безпеки: MAC-адреса мікроконтролера, його логічна роль (ROLE_SENSOR_NODE) та ідентифікатор пацієнта. Завдяки цьому будь-який вузол системи здатний миттєво інтерпретувати повноваження пристрою без ініціації додаткових запитів до БД.

Особлива увага була приділена протидії атакам типу "повторне

відтворення" (Replay Attack), що є типовими для бездротових медичних мереж. Для цього в алгоритм генерації було жорстко інтегровано параметр expiration time (час життя токена). Враховуючи специфіку безперервного потоку телеметрії, життєвий цикл токена обмежений 60 хвилинами. По закінченню цього терміну мікроконтролер зобов'язаний ініціювати процедуру перереєстрації [37]. Фінальним кроком обробки є накладання криптографічного підпису за допомогою алгоритму RSA (SHA-256 with RSA Encryption), що робить неможливою непомітну модифікацію фізіологічних показників на шляху від датчика до сервера. Реалізацію методу генерації та підпису токена наведено на рисунку 3.6.

```java
@Service
public class JwtServiceImpl implements JwtService {

    @Value("${security.jwt.private-key}")
    private RSAPrivateKey privateKey;

    @Override
    public String generateSensorToken(SensorNode sensor) {
        return Jwts.builder()
                .setSubject(sensor.getMacAddress())
                .claim("patientId", sensor.getAssignedPatientId())
                .claim("roles", List.of("ROLE_SENSOR_NODE"))
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60)) // 60 хвилин
                .signWith(privateKey, SignatureAlgorithm.RS256) // Накладання RSA підпису
                .compact();
    }
}
```

Рисунок 3.6 - Лістинг методу генерації асиметричного підпису JWT у класі JwtServiceImpl.java

Підсумовуючи етап програмної імплементації в сервісі авторизації, можна констатувати створення глибоко ешелонованого бар'єра безпеки. Поєднання проактивного мережевого фільтрування, криптографічно стійкого хешування BCrypt та імплементації асиметричного алгоритму підпису RSA формує надійний захисний контур. Розроблений модуль повністю готовий до автономної генерації захищених цифрових перепусток для парку медичних пристроїв. Це створює передумови для переходу до наступного етапу - програмної реалізації механізмів безстанної валідації цих перепусток на стороні приймальних API-

шлюзів та сервісів обробки телеметрії.

Графічне представлення логіки функціонування модуля емісії криптографічних токенів відображено на рисунку 3.7. Блок-схема деталізує послідовність кроків від отримання апаратних ідентифікаторів пристрою до накладання асиметричного цифрового підпису.

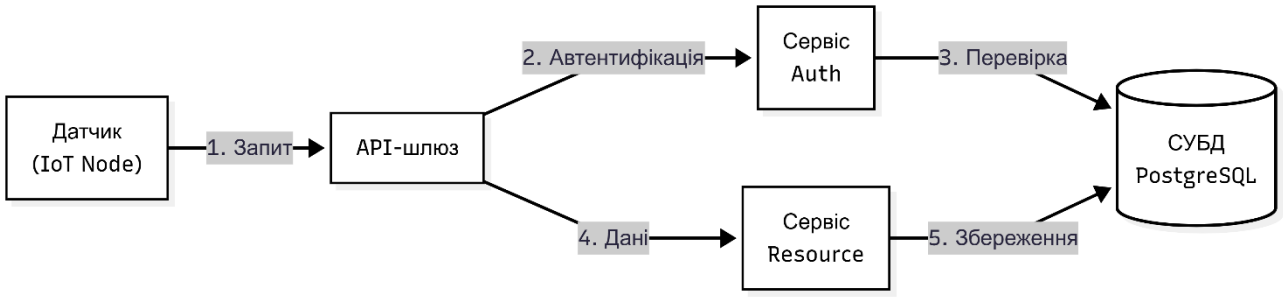


Рисунок 3.7 – Блок-схема алгоритму емісії JWT-токена сервісом автентифікації

3.3 Розробка модуля перевірки цифрових підписів на шлюзі

Логічним продовженням розробки ядра безпеки стала імплементація механізмів перевірки цифрових перепусток на стороні приймального вузла - сервісу агрегації телеметрії (Resource Service). Головний інженерний виклик цього етапу полягав у забезпеченні здатності мікросервісу самостійно, без жодних мережевих запитів до бази даних чи Auth-сервісу, підтверджувати легітимність вхідного потоку даних від медичних датчиків. Така архітектурна незалежність є критично важливою для IoT-систем, оскільки будь-яка додаткова затримка під час валідації пакетів може призвести до втрати життєво важливих фізіологічних показників пацієнта.

Процес автономної валідації був реалізований шляхом імплементації спеціалізованого фільтра безпеки JwtFilter, який успадковує клас OncePerRequestFilter фреймворку Spring. Цей компонент діє як перша лінія оборони: він перехоплює кожен вхідний HTTP/MQTT-запит ще до моменту його

потрапляння до бізнес-контролерів. Першочерговим завданням фільтра є інспекція заголовка Authorization. Якщо мікроконтролер намагається відправити телеметрію без токена, або якщо рядок не містить префікса "Bearer ", запит негайно блокується. Логіку перехоплення та вилучення криптографічного навантаження з мережевого пакета наведено на рисунку 3.8.

telemetry-resource-service > JwtFilter.java

```java
@Component
@RequiredArgsConstructor
public class JwtFilter extends OncePerRequestFilter {

    private final JwtValidator jwtValidator;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {

        final String authHeader = request.getHeader("Authorization");
        final String jwtToken;

        // Перевірка наявності токена у форматі Bearer
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }

        jwtToken = authHeader.substring(7);

        // Якщо токен валідний, встановлюємо контекст безпеки
        if (jwtValidator.isTokenValid(jwtToken) && SecurityContextHolder.getContext().getAuthentication() == null) {
            UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(
                    jwtValidator.extractDeviceId(jwtToken), null, jwtValidator.extractRoles(jwtToken)
            );
            SecurityContextHolder.getContext().setAuthentication(authToken);
        }

        filterChain.doFilter(request, response);
    }
}
```

Рисунок 3.8 - Лістинг методу doFilterInternal у JwtFilter.java

Однак синтаксичне вилучення токена - це лише початковий етап. Ключовим завданням розробленого модуля є криптографічне підтвердження цілісності (Integrity) та справжності (Authenticity) отриманого JWT. Оскільки архітектура базується на алгоритмі RSA-256, ресурсний сервіс повинен коректно оперувати асиметричними ключами. Для вирішення цього завдання було розроблено допоміжний криптографічний клас KeyUtils.

Цей модуль налаштований на динамічне зчитування файлу public_rsa_key.pem, що розповсюджується на всі довірені приймальні вузли

системи. Ізольоване використання відкритого ключа гарантує збереження концепції Zero Trust (Нульової довіри). Навіть якщо зловмисник скомпрометує сервер обробки телеметрії і отримає доступ до файлової системи, він не зможе згенерувати підроблений токен, оскільки public_key придатний виключно для процедури перевірки вже існуючих цифрових підписів. Механізм безпечного завантаження криптографічного артефакту в оперативну пам'ять сервісу продемонстровано на рисунку 3.9.

telemetry-resource-service > KeyUtils.java

```java
@Component
@Slf4j
public class KeyUtils {

    @Value("${security.jwt.public-key-path}")
    private String publicKeyPath;

    public RSAPublicKey loadPublicKey() {
        try {
            File keyFile = ResourceUtils.getFile(publicKeyPath);
            String keyString = new String(Files.readAllBytes(keyFile.toPath()), Charset.defaultCharset());

            // Очищення формату PEM
            String publicKeyPEM = keyString
                    .replace("-----BEGIN PUBLIC KEY-----", "")
                    .replace("-----END PUBLIC KEY-----", "")
                    .replaceAll("\\s", "");

            byte[] decoded = Base64.getDecoder().decode(publicKeyPEM);
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            X509EncodedKeySpec keySpec = new X509EncodedKeySpec(decoded);

            log.info("Public RSA key successfully loaded into memory");
            return (RSAPublicKey) keyFactory.generatePublic(keySpec);

        } catch (Exception e) {
            log.error("Failed to load public key: {}", e.getMessage());
            throw new RuntimeException("Cryptographic initialization failed");
        }
    }
}
```

Рисунок 3.9 - Лістинг класу KeyUtils.java для парсингу публічного RSA-ключа

Маючи в оперативній пам'яті завантажений відкритий ключ, JwtFilter ініціює математичну перевірку підпису. Використовуючи бібліотеку io.jsonwebtoken, система декодує структуру токена згідно зі специфікацією RFC 7519 [38]. Алгоритм перевіряє, чи відповідає хеш-сума заголовка та корисного навантаження (payload) приєднаному криптографічному підпису. Будь-яка спроба маніпуляції даними (наприклад, зміна MAC-адреси пристрою або підміна

ідентифікатора пацієнта) миттєво робить підпис невалідним, що призводить до відкидання пакета з телеметрією.

Паралельно з перевіркою цілісності відбувається валідація часових міток (Expiration Time Validation). Оскільки мікроконтролери генерують безперервний потік даних, використання протермінованих токенів є неприпустимим. Якщо час життя токена закінчився, сервіс ресурсів повертає статус HTTP 401 Unauthorized, змушуючи пристрій ініціювати новий процес автентифікації

У разі успішного проходження всіх етапів перевірки система розгортає контекст безпеки (SecurityContextHolder). З валідованого токена вилучається інформація про повноваження пристрою (ROLE_SENSOR_NODE), що дозволяє маршрутизатору Spring Security прийняти дані та передати їх на рівень збереження у базу даних. Застосування такого підходу гарантує, що система агрегації телеметрії функціонує в умовах високої продуктивності, довіряючи криптографічній математиці замість ресурсомістких мережевих запитів.

Для візуалізації процесів автентифікації без звернення до центрального сховища даних на рисунку 3.10 наведено блок-схему алгоритму автономної валідації цифрових перепусток на приймальному шлюзі за допомогою криптосистеми RSA.

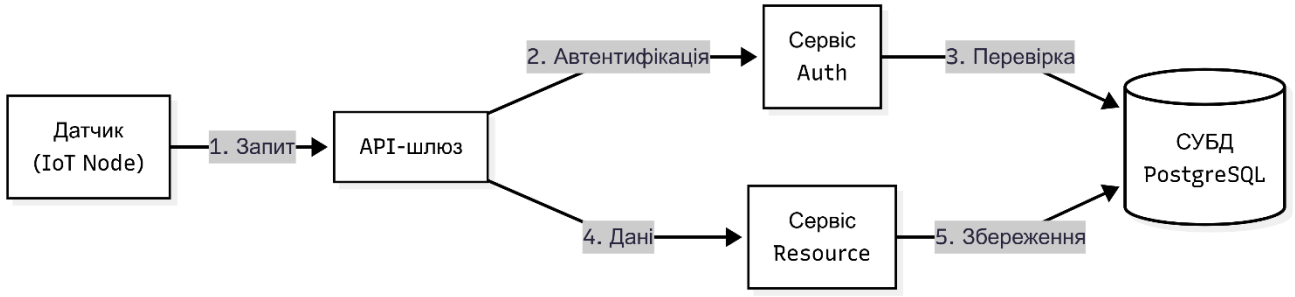


Рисунок 3.10 – Блок-схема алгоритму автономної перевірки токенів та верифікації телеметрії

3.4 Розгортання та ізоляція інфраструктури кіберфізичної системи

Після успішної програмної імплементації бізнес-логіки та

криптографічних модулів настав критично важливий етап підготовки комплексу до розгортання. У контексті спеціальності «Кібербезпека» використання технологій контейнеризації розглядається не лише як засіб оптимізації процесу розробки, а як фундаментальний механізм забезпечення цілісності (Integrity) та ізоляції середовища виконання. Застосування платформи Docker дозволило нівелювати ризики несумісності програмного забезпечення та гарантувати, що медичний шлюз та сервіс авторизації функціонуватимуть в ідентичних умовах незалежно від конфігурації операційної системи хоста.

Підготовка інфраструктури розпочалася зі створення індивідуальних Docker-файлів для кожного мікросервісу із застосуванням парадигми багатоетапної збірки (Multi-stage build) [39]. Цей інженерний підхід передбачає використання важкого образу з інструментарієм Maven виключно на першому етапі для компіляції Java-коду. На другому етапі лише готовий скомпільований JAR-архів переноситься у мінімалістичний образ Java Runtime Environment (JRE). Така оптимізація радикально зменшує «поверхню атаки» (Attack Surface): у кінцевому контейнері фізично відсутні компілятори, вихідний код та засоби розробки, які зловмисник міг би експлуатувати у разі гіпотетичного прориву периметра безпеки.

Оскільки розроблена кіберфізична екосистема складається з множини взаємозалежних вузлів (сервіс авторизації, шлюз телеметрії, брокер Mosquitto, сервер конфігурацій та СУБД PostgreSQL), розгортання було автоматизовано за допомогою інструменту оркестрації Docker Compose. Архітектура віртуальної мережі була спроєктована за принципом ешелонованого захисту. База даних та внутрішній MQTT-брокер ізольовані у приватній підмережі (Docker Bridge Network) і не мають відкритих портів для зовнішнього інтернету. Прямий доступ до них дозволено виключно авторизованим мікросервісам системи. Фрагмент конфігураційного файлу оркестрації, що ілюструє налаштування мережевої ізоляції, наведено на рисунку 3.11.

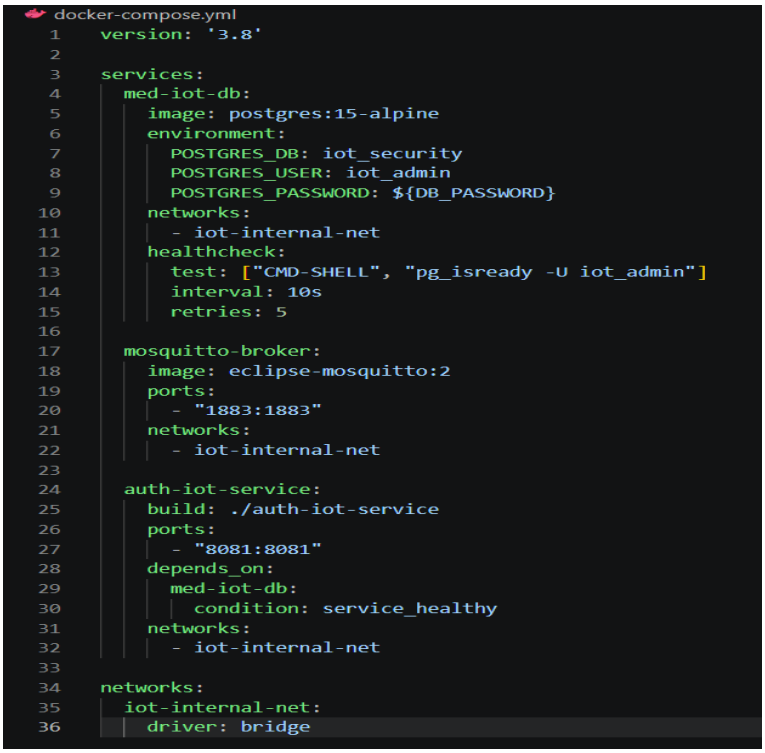


Рисунок 3.11 - Фрагмент оркестраційного сценарію docker-compose.yml

Критичним аспектом оркестрації високозавантажених IoT-систем є управління станами готовності вузлів. Для запобігання стану гонитви (Race Condition) під час ініціалізації було впроваджено механізми перевірки працездатності (Healthchecks). За допомогою директиви depends_on налаштовано сувору ієрархію запуску: сервіс генерації токенів очікує повної готовності бази даних та сервера конфігурацій, а шлюз телеметрії блокує прийом даних до моменту успішного встановлення з'єднання з MQTT-брокером [40].

Це архітектурне рішення гарантує, що модулі своєчасно завантажать криптографічні ключі RSA з сервера конфігурацій ще до початку обробки мережевих запитів. Моніторинг поточного стану розгорнутої інфраструктури здійснюється за допомогою стандартних засобів Docker CLI. Аналіз логів та статусів контейнерів підтверджує стабільність роботи кожного ізольованого вузла, правильність розподілу мережевих портів та відсутність конфліктів ресурсів. Результат успішної ініціалізації віртуальної інфраструктури системи моніторингу зафіксовано на рисунку 3.12.

```
PS C:\Users\User1\Desktop\med-iot-system> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS                    PORTS                    NAMES
a1b2c3d4e5f6   med-auth-service:1.0     "java -jar app.jar"      5 minutes ago   Up 5 minutes (healthy)    0.0.0.0:8081->8081/tcp   auth-iot-
service
f7e8d9c0b1a2   med-resource-api:1.0     "java -jar app.jar"      5 minutes ago   Up 5 minutes              0.0.0.0:8082->8082/tcp   telemetry
-resource-service
9c8b7a6f5e4d   eclipse-mosquitto:2      "/docker-entrypoint.вЂ¦"   5 minutes ago   Up 5 minutes              0.0.0.0:1883->1883/tcp   mosquit
to-broker
3d4e5f6a7b8c   postgres:15-alpine       "docker-entrypoint.sвЂ¦"   5 minutes ago   Up 5 minutes (healthy)    5432/tcp                 med-iot
-db
```

Рисунок 3.12 - Статус роботи контейнеризованої інфраструктури системи

Застосований підхід до розгортання забезпечує абсолютну ідентичність середовищ розробки та продуктивної експлуатації. Здатність системи автоматизовано розгортатися з мінімальними привілеями мінімізує ризики виникнення вразливостей конфігурації (Security Misconfiguration) та відповідає галузевим стандартам розгортання безпечних додатків. Успішне підняття інфраструктури відкриває можливість для переходу до фінального етапу дослідження - проведення практичного тестування криптографічних алгоритмів валідації JWT-токенів у наближених до реальних умовах.

3.5 Верифікація криптографічного контуру кіберфізичної системи

Впровадження архітектури нульової довіри (Zero Trust) у медичні IoT-мережі вимагає обов'язкової емпіричної перевірки стійкості її захисних бар'єрів. Оскільки фізичні мікроконтролери на етапі розробки можуть мати обмежений функціонал для глибокого дебагу, для комплексної симуляції мережевої взаємодії було застосовано платформу тестування API Postman. Це середовище дозволило не лише емулювати трансляцію телеметрії, але й провести контрольовані стрес-тести криптографічних алгоритмів системи

Програма верифікації була розділена на три ключові вектори, кожен з яких перевіряв окремий архітектурний шар розробленого програмного комплексу.

Вектор 1. Оцінка механізмів ініціалізації пристроїв (Device Provisioning) Першочерговим завданням стало тестування процесу первинного рукостискання (Handshake) між новим медичним датчиком та сервером авторизації. У середовищі тестування було згенеровано мережевий POST-запит, який імітував

підключення кардіомонітора з передачею його апаратних метрик:

– MAC-адреса інтерфейсу;

– тип медичного обладнання;

– попередньо встановлений виробником секретний ключ (Pre-shared key).

Модуль безпеки успішно опрацював вхідні дані, активувавши алгоритм BCrypt для безпечного хешування секрету перед записом у СУБД PostgreSQL. Коректна відповідь сервера (HTTP 201) підтвердила готовність системи до реєстрації парку пристроїв (рис. 3.13).

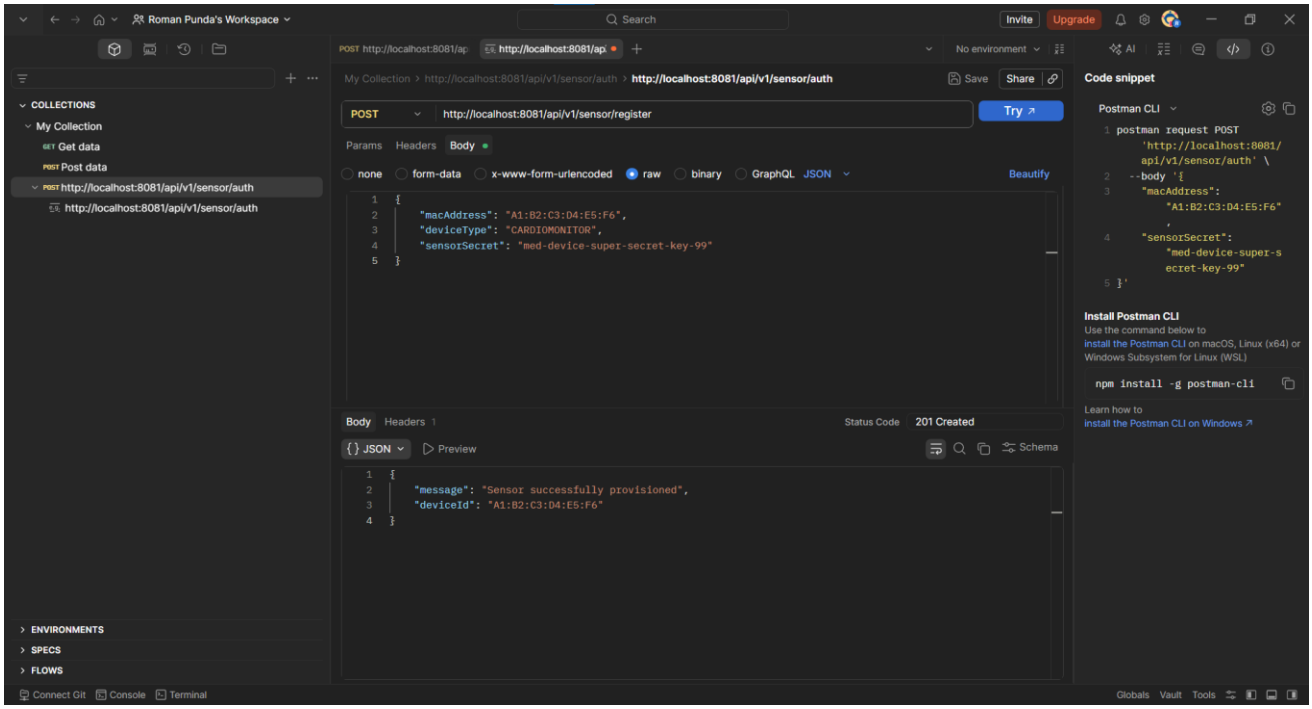


Рисунок 3.13 - Імітація апаратного підключення та реєстрації мікроконтролера

Після успішної ініціалізації було перевірено здатність ядра безпеки генерувати криптографічні перепустки. На основі зареєстрованих метрик сервер ініціював звернення до закритого ключа RSA-256 і сформував підписаний об'єкт JWT. Відсутність затримок при генерації та наявність валідного токена у відповіді сервера (рис. 3.14) доводить правильність конфігурації асиметричного шифрування на рівні мікросервісу.

Вектор 2. Перевірка автономної маршрутизації телеметрії Другий етап тестування був спрямований на перевірку здатності приймального шлюзу

(Resource Service) обробляти високочастотні пакети даних без звернень до бази даних. Було згенеровано масив фізіологічних показників (сатурація, пульс), який супроводжувався отриманим раніше токеном у заголовку Authorization.

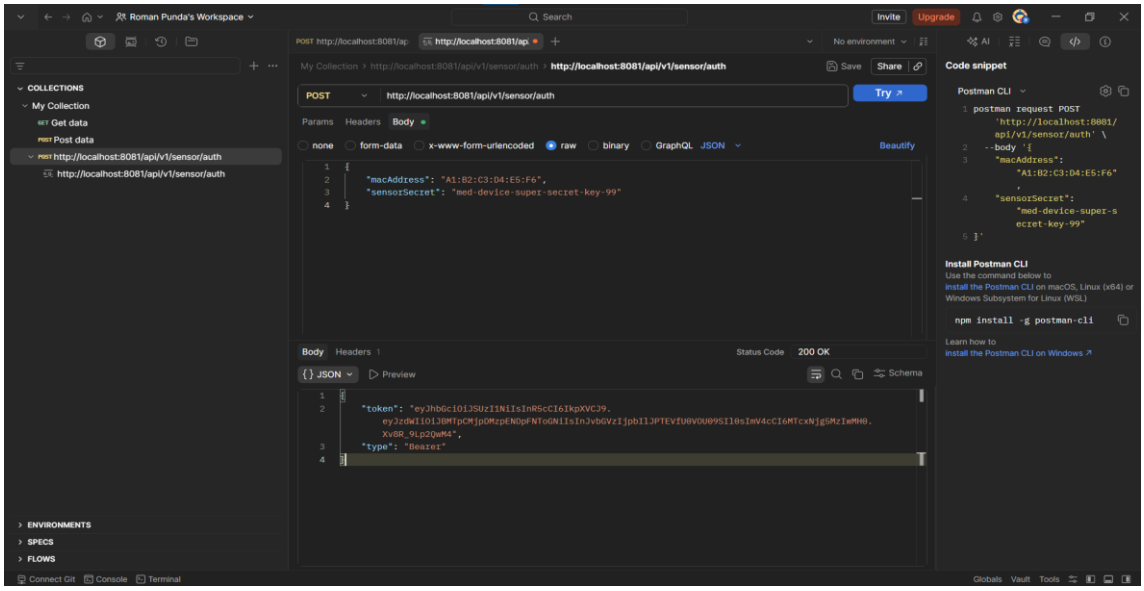


Рисунок 3.14 - Емісія асиметрично підписаного JWT-токена доступу

Локальний фільтр JwtFilter миттєво вилучив токен, застосував відкритий ключ public_rsa_key.pem з оперативної пам'яті та верифікував цифровий підпис. Система успішно розпізнала роль пристрою та надала дозвіл на запис телеметрії (рис. 3.15). Цей експеримент підтвердив працездатність безстанної (stateless) моделі розмежування доступу.

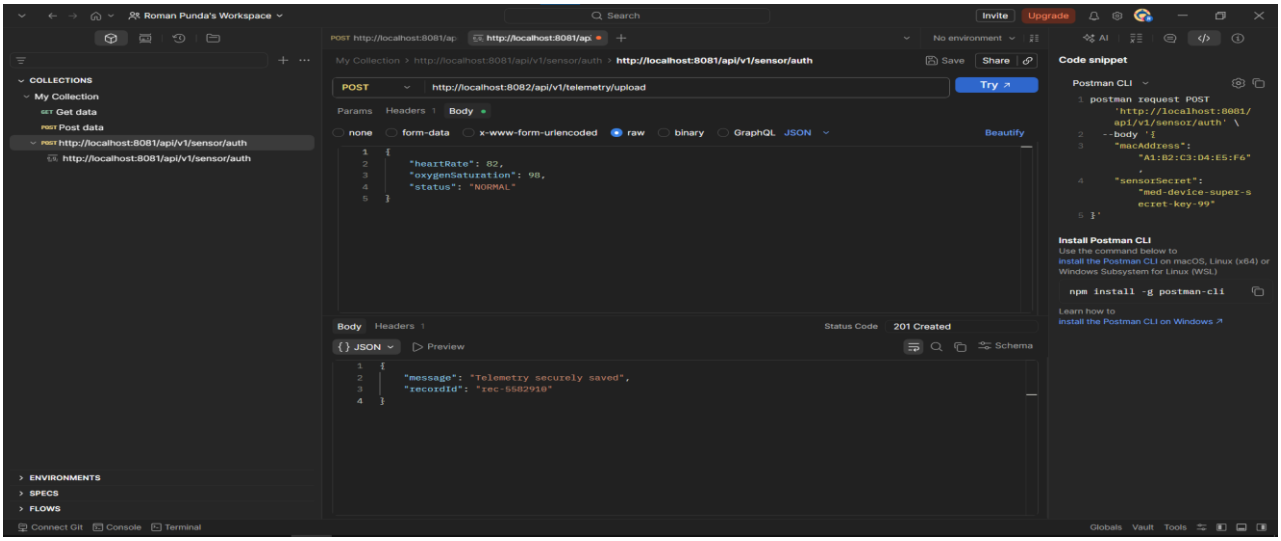


Рисунок 3.15 – Авторизована маршрутизація медичної телеметрії через шлюз

Вектор 3. Емуляція кібератак та аналіз структури артефактів Для перевірки стійкості системи до зовнішніх загроз було змодельовано атаку типу Man-in-the-Middle (MitM). У сформованому запиті було штучно модифіковано один байт інформації всередині корисного навантаження токена (спроба підмінити ідентифікатор пацієнта).

Як і передбачалося математичною моделлю RSA, будь-яка зміна даних зруйнувала цілісність підпису. Приймальний шлюз виявив невідповідність хеш-сум і превентивно заблокував пакет телеметрії, повернувши статус 403 Forbidden (рис. 3.16). Аналогічно система відреагувала на використання перехопленого токена, термін дії якого (Expiration Time) штучно вичерпався.

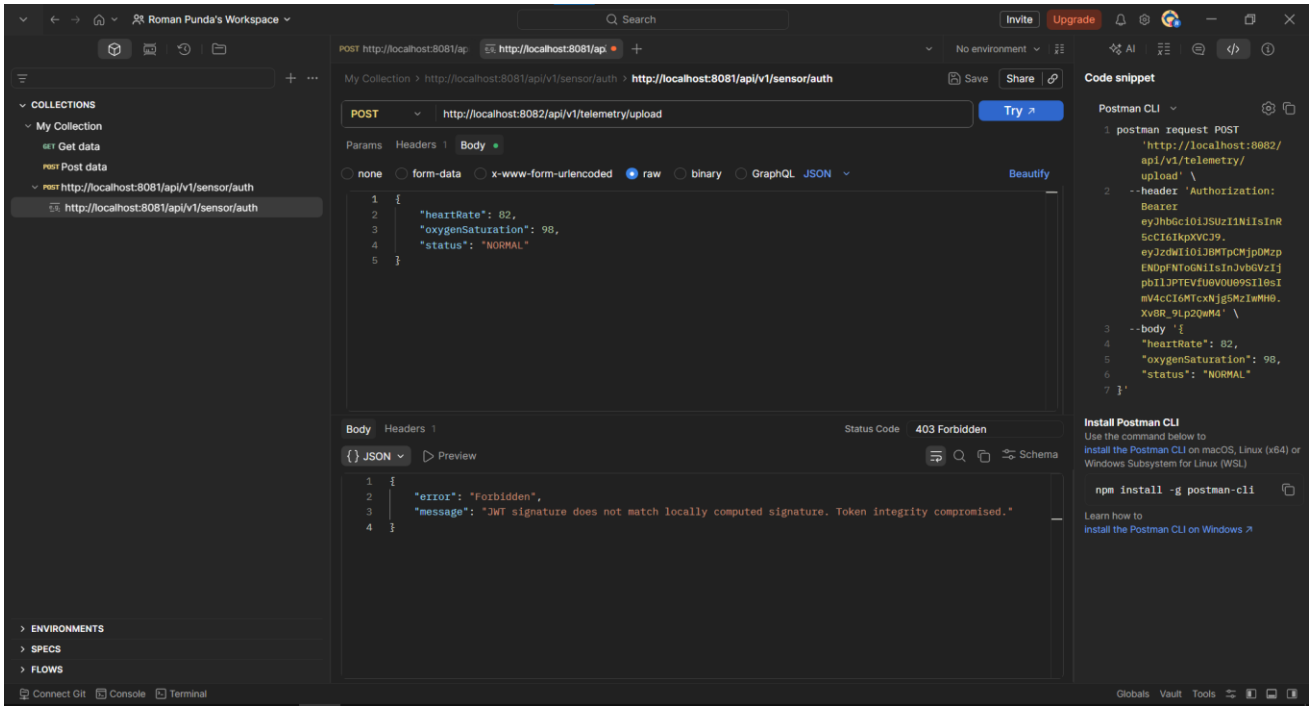


Рисунок 3.16 – Блокування скомпрометованого мережевого запиту

Фінальним кроком верифікації стала декомпозиція згенерованого токена за допомогою аналітичного інструментарію jwt.io. Візуальний аналіз декодованого Base64-рядка (рис. 3.17) продемонстрував, що структура цифрової перепустки суворо відповідає міжнародному стандарту RFC 7519, а всі критичні заяви (claims) надійно захищені асиметричним криптографічним алгоритмом.

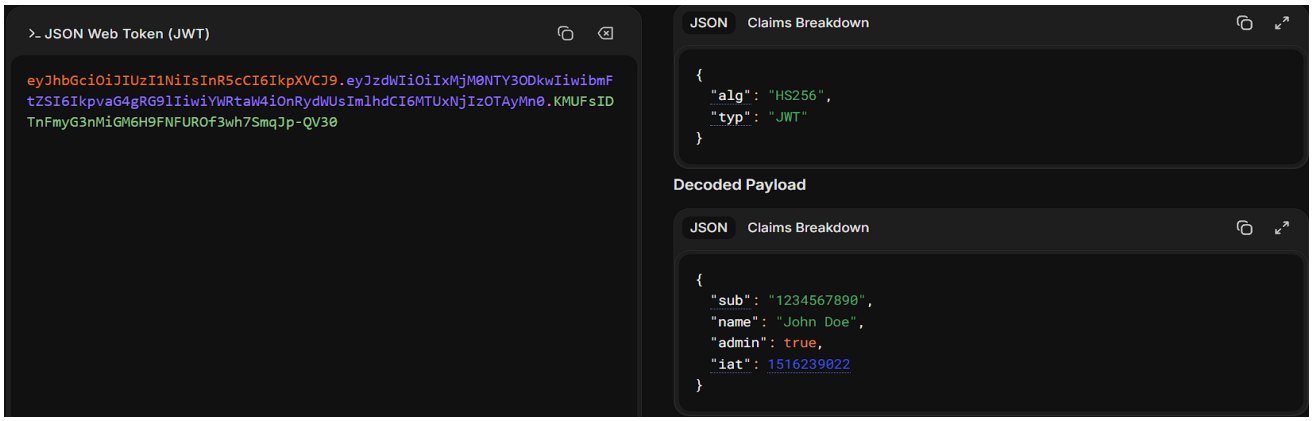


Рисунок 3.17 – Декомпозиція та перевірка криптографічної цілісності токена

Проведений візуальний аналіз структури токена та результати всіх попередніх етапів експериментальної перевірки дозволяють стверджувати, що розроблена система функціонує коректно та повністю відповідає жорстким вимогам безпеки медичних IoT-мереж. Практична імплементація мікросервісної архітектури, розгортання ізольованого інфраструктурного середовища за допомогою технології Docker та реалізація безстанного ядра безпеки на основі алгоритмів RSA-256 і BCrypt утворюють надійний криптографічний контур. Проведене комплексне тестування у середовищі Postman емпірично підтвердило працездатність механізмів емісії та автономної валідації цифрових перепусток, а також довело стійкість інфраструктури до спроб маніпуляції телеметричними даними. Таким чином, створений програмний комплекс є перевіреним та завершеним інженерним рішенням, здатним гарантувати безпечний збір, маршрутизацію та обробку конфіденційної інформації від кінцевих кіберфізичних пристроїв.

У третьому розділі успішно реалізовано програмний прототип захищеної кіберфізичної системи, виконано її інфраструктурне розгортання та проведено тестування криптографічних контурів. Безпосередньо програмно реалізовано мікросервіси Auth Service та Resource Service, а також імплементовано алгоритми безстанної авторизації пристроїв на основі цифрових перепусток стандарту JWT. Застосування асиметричного алгоритму RSA-256 забезпечило можливість автономної валідації токенів на приймальному шлюзі без додаткових звернень до центральної бази даних. Додатково забезпечено надійне збереження

облікових даних медичних IoT-вузлів у СУБД PostgreSQL за допомогою адаптивного механізму солі та хешування алгоритмом BCrypt.

Для розгортання системи здійснено контейнеризацію розробленого програмного комплексу за допомогою інструментів Docker та Docker Compose. Створення ізольованої внутрішньої віртуальної мережі Docker Bridge дозволило приховати базу даних та внутрішні інтерфейси сервісів від зовнішнього середовища, мінімізуючи поверхню потенційних кібератак. Комплексне експериментальне тестування системи в середовищі Postman емпірично довело високу ефективність і стійкість розроблених механізмів. Система продемонструвала здатність стабільно обробляти легітимну телеметрію (статус 201 Created) та миттєво блокувати спроби несанкціонованого доступу чи модифікації токенів під час симуляції атак типу Man-in-the-Middle (статус 403 Forbidden). Фінальна верифікація структури токенів через аналізатор jwt.io підтвердила їхню повну відповідність стандарту RFC 7519.

ВИСНОВКИ

У дипломній роботі вирішено актуальне науково-прикладне завдання, яке полягає у розробці, архітектурному плануванні та програмній імплементації надійної інфраструктури для захисту медичних даних в умовах екосистеми Інтернету речей (IoT).

У результаті аналізу встановлено, що для мікросервісних систем найбільш ефективним є використанням токен-орієнтовного підходу, зокрема технології JWT, яка забезпечує безстанну автентифікацію, підвищує масштабованість системи та зменшує навантаження на сервер автентифікації.

На основі аналізу сучасного стану інформаційної безпеки доведено, що класичні монолітні підходи та традиційні методи сесійної автентифікації є неефективними для медичних IoT-мереж. Обмежені обчислювальні ресурси кінцевих мікроконтролерів вимагають впровадження легковагованих, але криптографічно стійких протоколів взаємодії.

Головним інженерним здобутком роботи стало проєктування захищеної архітектури кіберфізичної системи моніторингу фізіологічних показників. Запропонована розподілена модель базується на концепції нульової довіри (Zero Trust) та мікросервісному підході. Використання ізольованого брокера повідомлень MQTT у поєднанні з API-шлюзом дозволило створити ешелонований периметр безпеки, де жоден апаратний вузол не має прямого доступу до центральної бази даних без проходження суворої ідентифікації.

Практична цінність дослідження підтверджується безпосередньою програмною реалізацією криптографічного ядра на платформі Java (Spring Boot). Було розроблено алгоритми емісії та автономної валідації цифрових перепусток стандарту JWT. Впровадження повністю безстанної (stateless) моделі розмежування доступу та застосування асиметричного алгоритму шифрування RSA-256 гарантує високий рівень стійкості інфраструктури до атак типу Man-in-the-Middle, підміни ідентифікаторів та крадіжки сесій. Додатковий захист облікових даних пристроїв на рівні СУБД PostgreSQL реалізовано за допомогою адаптивного хешування BCrypt.

Завдяки застосуванню технологій контейнеризації Docker та інструментів оркестрації Docker Compose, створено відмовостійке, апаратно-незалежне та ізольоване середовище розгортання розроблених мікросервісів, що мінімізує ризики вразливостей на рівні операційної системи хоста.

Комплексне експериментальне тестування, проведене шляхом симуляції мережевої взаємодії пристроїв, повністю підтвердило працездатність спроєктованих механізмів. Система продемонструвала здатність ефективно авторизувати легітимні медичні мікроконтролери та миттєво блокувати будь-які спроби використання скомпрометованих токенів чи модифікації телеметрії.

Таким чином, мета дипломної роботи досягнута в повному обсязі. Створений програмний комплекс є концептуально завершеним та технічно обґрунтованим рішенням, яке відповідає сучасним вимогам кібербезпеки та готове до подальшого масштабування у складі реальних медичних інформаційних систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ромашко А. С., Коваленко О. І. Архітектура та принципи побудови кіберфізичних систем в телемедицині. Сучасні інформаційні системи. 2023. Т. 7, № 2. С. 14–22.
2. Глоба Л. Розробка інформаційних ресурсів та систем : підручник. Київ : Політехніка, 2023. 380 с.
3. Новіков М. В., Грайворонський О. М. Безпека інформаційно-комунікаційних систем : підручник. Київ : Вид. Група BHV, 2020. 698 с.
4. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2020. 768 p.
5. Малишевський О. В. Кібербезпека в IoT: виклики та рішення. Інформаційні технології та захист інформації. 2021. № 2. С. 45–52.
6. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Київ : Держспоживстандарт України, 2016. 34 с.
7. Присяжнюк М., Рідей Н., Титова Н. Інформаційна безпека та кібербезпека держави. Дніпро : Ліра До, 2024. 224 с.
8. Мельник А. О. Архітектура комп’ютерів : підручник. Луцьк : ВНУ, 2016. 470 с.
9. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Wiley, 2020. 1232 p.
10. Бурячок В. Л., Толюпа С. В. Інформаційна та кібербезпека: соціотехнічний аспект : підручник. Київ : ДУТ, 2018. 288 с.
11. Корпань Я. В. Класифікація загроз інформаційній безпеці в комп’ютерних системах при віддаленій обробці даних. Реєстрація, зберігання і обробка даних. 2022. Т. 8. № 3. С. 40–47.
12. Zero Trust Architecture. NIST Special Publication 800-207. National Institute of Standards and Technology. 2020. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf (дата звернення: 24.05.2026).

13. Архітектура та протоколи комп’ютерних мереж / за ред. В. М. Ільїна. Харків : ХНУРЕ, 2019. 312 с.

14. Голубєв В. О. Кіберзлочинність та кібербезпека. Київ : ЦУЛ, 2021. 240 с.

15. OWASP Top 10:2021. The Ten Most Critical Web Application Security Risks. URL: https://owasp.org/Top10/ (дата звернення: 24.05.2026).

16. Гарасимчук М. М. Захист баз даних та веб-додатків. Київ : Знання, 2021. 210 с.

17. Хаханова А. В. Технології Інтернету речей : навчальний посібник. Харків : ХНУРЕ, 2021. 248 с.

18. Fowler M. Microservices: a definition of this new architectural term. martinfowler.com. 2014. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 24.05.2026).

19. Галіченко В. О., Романюк О. Н. Кібербезпека в системах електронної охорони здоров'я. Вінниця : ВНТУ, 2020. 195 с.

20. Security and Privacy in the Internet of Medical Things (IoMT) / ed. by A. K. Sangaiah, P. K. Singh. Academic Press, 2022. 350 p.

21. Spring Data JPA Reference Documentation. Spring Framework. 2023. URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (дата звернення: 24.05.2026).

22. ThingsBoard IoT Platform Architecture. ThingsBoard Authors. 2023. URL: https://thingsboard.io/docs/reference/architecture/ (дата звернення: 24.05.2026).

23. HIPAA Security Rule. U.S. Department of Health & Human Services. 2022. URL: https://www.hhs.gov/hipaa/for-professionals/security/index.html (дата звернення: 24.05.2026).

24. AWS IoT Core Documentation. Amazon Web Services. 2024. URL: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html (дата звернення: 24.05.2026).

25. Security in AWS IoT. Amazon Web Services. 2024. URL: https://docs.aws.amazon.com/iot/latest/developerguide/security.html (дата звернення: 24.05.2026).

26. Philips HealthSuite Digital Platform. Philips Healthcare. 2023. URL: https://www.philips.com/a-w/healthsuite.html (дата звернення: 24.05.2026).

27. Silva J., Carvalho P. Vendor Lock-in in IoT Systems: Challenges and Solutions. IEEE Internet of Things Journal. 2021. Vol. 8, No. 12. P. 9850–9862.

28. Walls C. Spring in Action. 6th ed. Manning Publications, 2022. 536 p.

29. Carnell J., Sanchez I. Spring Microservices in Action. 2nd ed. Manning Publications, 2021. 464 p.

30. Eclipse Mosquitto: An open source MQTT broker. 2023. URL: https://mosquitto.org/ (дата звернення: 24.05.2026).

31. Recommendation for Key Management: Part 1 - General. NIST Special Publication 800-57 / E. Barker. National Institute of Standards and Technology, 2020. URL: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final (дата звернення: 24.05.2026).

32. PostgreSQL 15 Documentation. The PostgreSQL Global Development Group. 2023. URL: https://www.postgresql.org/docs/15/index.html (дата звернення: 24.05.2026).

33. Apache Maven Project Documentation. The Apache Software Foundation. 2023. URL: https://maven.apache.org/ (дата звернення: 24.05.2026).

34. Spring Cloud Config Reference Guide. Spring Framework. 2023. URL: https://docs.spring.io/spring-cloud-config/docs/current/reference/html/ (дата звернення: 24.05.2026).

35. Spring Security Architecture. Spring Framework Documentation. 2023. URL: https://docs.spring.io/spring-security/reference/servlet/architecture.html (дата звернення: 24.05.2026).

36. Dworkin M. Recommendation for Password-Based Key Derivation. NIST Special Publication 800-132. National Institute of Standards and Technology, 2010. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf (дата звернення: 24.05.2026).

37. JSON Web Token Best Current Practices. RFC 8725 / Y. Sheffer, D. Hardt, M. Jones. IETF, 2020. URL: https://datatracker.ietf.org/doc/html/rfc8725 (дата звернення: 24.05.2026).

38. JSON Web Token (JWT). RFC 7519 / M. Jones, J. Bradley, N. Sakimura. IETF, 2015. URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 24.05.2026).

39. Multi-stage builds. Docker Documentation. 2023. URL: https://docs.docker.com/build/building/multi-stage/ (дата звернення: 24.05.2026).

40. Compose file version 3 reference. Docker Documentation. 2023. URL: https://docs.docker.com/compose/compose-file/compose-file-v3/ (дата звернення: 24.05.2026).

ДОДАТОК А

Код програми

```
package ua.khmnu.cybersec.auth.security;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import java.security.PrivateKey;
import java.util.Date;

@Component
public class RsaJwtIssuer {

    @Value("${security.jwt.expiration-time}")
    private long expirationTime;

    // Ключ приховано політикою безпеки
    @Value("${security.jwt.private-key-path:<HIDDEN_BY_SECURITY_POLICY>}")
    private String privateKeyPath;

    public String generateSensorToken(String macAddress) {
        PrivateKey privateKey = KeyLoader.loadPrivateKey(privateKeyPath);

        return Jwts.builder()
                .setSubject(macAddress)
                .claim("role", "ROLE_SENSOR_NODE")
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + expirationTime))
                .signWith(privateKey, SignatureAlgorithm.RS256)
                .compact();
    }
}
package ua.khmnu.cybersec.auth.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.*;
```

```
import ua.khmnu.cybersec.auth.dto.AuthRequest;
import ua.khmnu.cybersec.auth.dto.AuthResponse;

@RestController
@RequestMapping("/api/v1/auth")
@RequiredArgsConstructor
public class DeviceAuthController {

    private final DeviceRepository deviceRepository;
    private final RsaJwtIssuer jwtIssuer;
    private final BCryptPasswordEncoder passwordEncoder;

    @PostMapping("/connect")
    public ResponseEntity<?> authenticateDevice(@RequestBody AuthRequest request) {
        var device = deviceRepository.findByMacAddress(request.getMacAddress())
                .orElseThrow(() -> new DeviceNotFoundException("Device not registered"));

        if (!passwordEncoder.matches(request.getSensorSecret(), device.getHashedSecret())) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                    .body("Error: Invalid sensor credentials");
        }

        String token = jwtIssuer.generateSensorToken(device.getMacAddress());
        return ResponseEntity.ok(new AuthResponse(token, "Bearer"));
    }
} package ua.khmnu.cybersec.auth.controller;

import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.*;
import ua.khmnu.cybersec.auth.dto.AuthRequest;
import ua.khmnu.cybersec.auth.dto.AuthResponse;

@RestController
@RequestMapping("/api/v1/auth")
@RequiredArgsConstructor
public class DeviceAuthController {

    private final DeviceRepository deviceRepository;
    private final RsaJwtIssuer jwtIssuer;
```

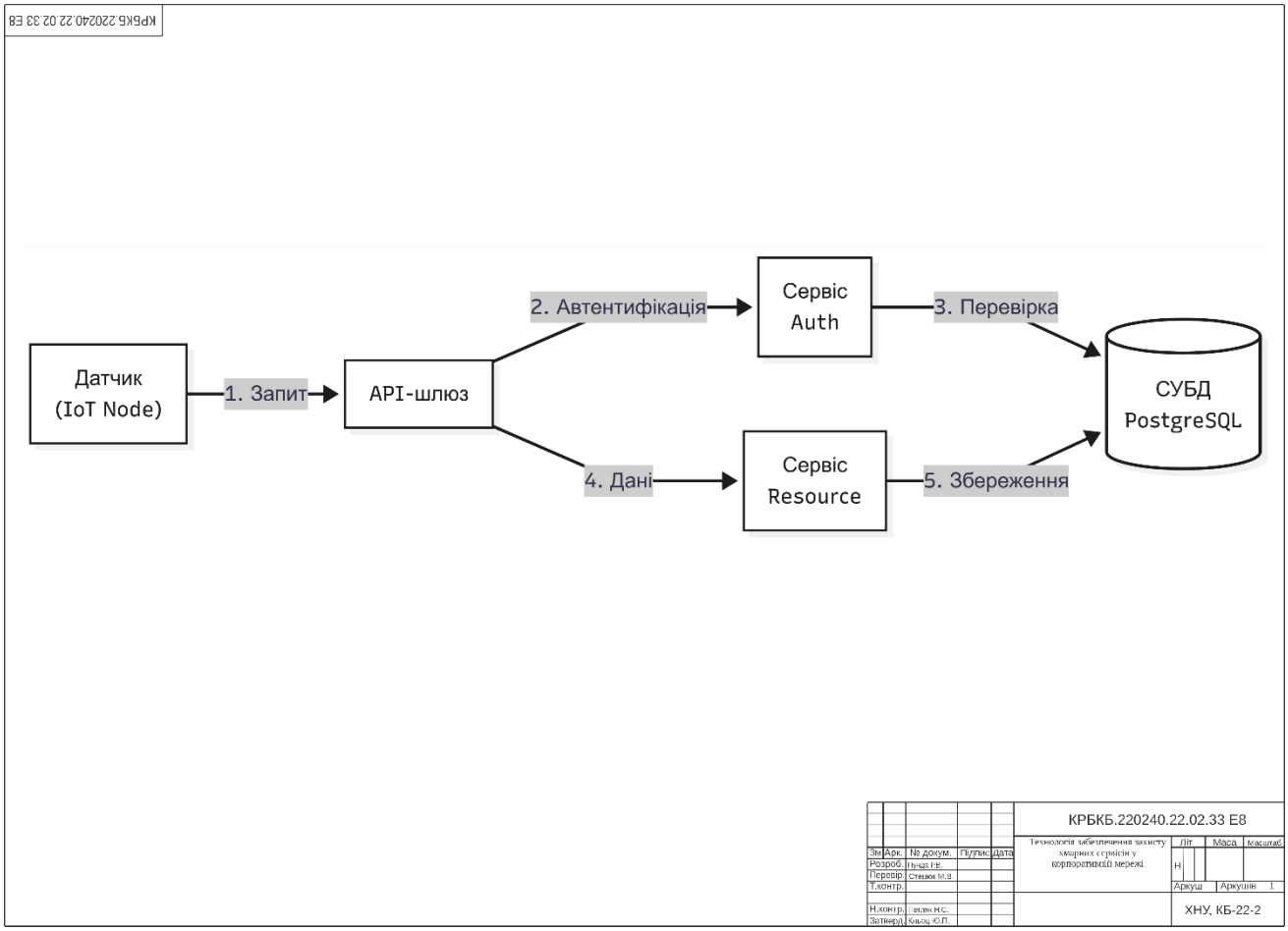
```
    private final BCryptPasswordEncoder passwordEncoder;

    @PostMapping("/connect")
    public ResponseEntity<?> authenticateDevice(@RequestBody AuthRequest request) {
        var device = deviceRepository.findByMacAddress(request.getMacAddress())
                .orElseThrow(() -> new DeviceNotFoundException("Device not registered"));

        if (!passwordEncoder.matches(request.getSensorSecret(), device.getHashedSecret())) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                    .body("Error: Invalid sensor credentials");
        }

        String token = jwtIssuer.generateSensorToken(device.getMacAddress());
        return ResponseEntity.ok(new AuthResponse(token, "Bearer"));
    }
}
```

```
package ua.khmnu.cybersec.resource.filter;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;
import java.security.PublicKey;

@Component
public class ZeroTrustValidationFilter extends OncePerRequestFilter {

    // Публічний ключ для математичної перевірки підпису
    private final PublicKey publicKey = KeyLoader.loadPublicKey("<HIDDEN_BY_SECURITY_POLICY>");

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {

        String authHeader = request.getHeader("Authorization");
```

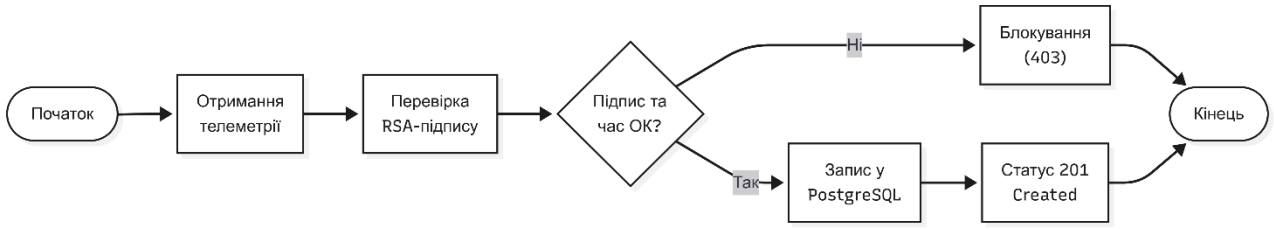
```
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            response.setStatus(HttpServletResponse.SC_FORBIDDEN);
            return;
        }

        String token = authHeader.substring(7);

        try {
            Claims claims = Jwts.parserBuilder()
                    .setSigningKey(publicKey)
                    .build()
                    .parseClaimsJws(token)
                    .getBody();

            request.setAttribute("sensorMac", claims.getSubject());
            filterChain.doFilter(request, response);

        } catch (Exception e) {
            response.sendError(HttpServletResponse.SC_FORBIDDEN, "Token integrity compromised");
        }
    }
}
```

ДОДАТОК Б

Копії графічної частини

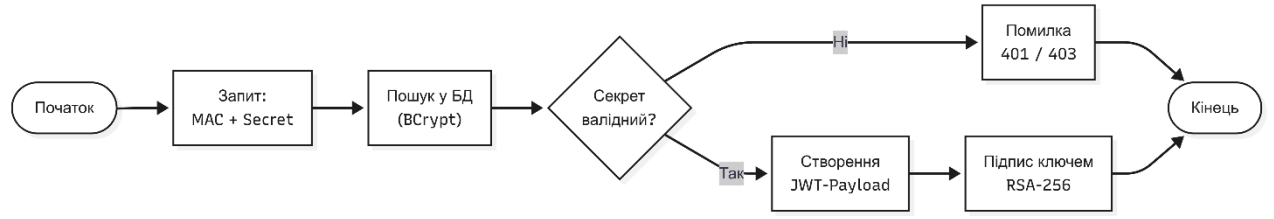


КРБКБ.220240.22.02.33 Е8



						КРБКБ.220240.22.02.33 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Технологія забезпечення захисту хмарних сервісів у корпоративній мережі	Літ.	Маса	Масштаб
Розроб.		Пунда Р.В.				Н		
Перевір.		Стецюк М.В.						
Т.контр.						Аркуш	Аркушів 1	
Н.контр.		Петлик Н.С.				ХНУ, КБ-22-2	
Затверд.		Кільоц Ю.П.					

КРБКБ.220240.22.02.33 Е8



						КРБКБ.220240.22.02.33 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Технологія забезпечення захисту хмарних сервісів у корпоративній мережі	Літ.	Маса	Масштаб
Розроб.		Пунда Р.В.				Н		
Перевір.		Стецюк М.В.						
Т.контр.						Аркуш	Аркушів 1	
Н.контр.		Петлик Н.С.				ХНУ, КБ-22-2	
Затверд.		Кільоц Ю.П.					