МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Сенькович Костянтин Юрійович

VPN-шлюз для безпечного підключення IoT-вузлів / VPN Gateway for Secure IoT Node Connectivity

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
К. Ю. Сенькович

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

Кваліфікаційну роботу
допущено до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____Н.В. Дзюбановська

ТЕРНОПІЛЬ – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Сенькович Костянтин Юрійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: VPN-шлюз для безпечного підключення IoT-вузлів / VPN Gateway for Secure IoT Node Connectivity
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 01 грудня 2025 р. № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- провести аналіз предметної області;
- провести опис предметної області ;
- провести аналіз підходів до захисту трафіку IoT-вузлів;
- сформувати постановку задачі;
- розробити архітектуру VPN-шлюзу;
- розробити алгоритми функціонування основних компонентів;
- розробити інформаційне забезпечення;
- провести вибір програмних і апаратних засобів
- реалізувати ключові модулі;
- провести тестування функціональності.

5. Перелік графічного матеріалу в роботі:
- структурна схема системи;
- автомат станів функціонування VPN-шлюзу;

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ К. Ю. Сенькович
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 54 с., 14 рис., 28 джерел, 4 додатки.

Метою роботи є розробка VPN-шлюзу для безпечного підключення IoT-вузлів через публічні мережі з підтримкою захищеного тунелювання, маршрутизації трафіку, базового контролю доступу, автентифікації вузлів, моніторингу стану та журналювання подій.

В роботі застосовано методи системного аналізу, порівняльного аналізу VPN- та overlay-рішень, мережевого тунелювання і маршрутизації, фільтрації трафіку засобами firewall, віртуалізації мережевого середовища Linux, а також засоби програмної реалізації мережевих сервісів мовами Go, Python і Bash.

Розглянуто предметну область безпечного підключення IoT-вузлів, виконано огляд існуючих рішень

Розроблено архітектуру VPN-шлюзу, яка включає локальний сегмент IoT-вузлів, VPN-шлюз на базі Raspberry Pi, зовнішню мережу, віддалений сервер або адміністратора, а також допоміжні служби DNS і NTP. Розроблено алгоритмічне забезпечення системи. Сформовано інформаційне забезпечення системи. Реалізовано експериментальний прототип VPN-шлюзу.

Проведено тестування функціональності розробленого прототипу.

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ, IoT, VPN-ШЛЮЗ, RASPBERRY PI, WIREGUARD, МАРШРУТИЗАЦІЯ, ЗАХИСТ ТРАФІКУ, IPTABLES, GO, PYTHON, ТЕЛЕМЕТРІЯ, EDGE-ПРИСТРІЙ.

ANNOTATION

Explanatory note to the qualification thesis: 54 pages, 14 figures, 28 references, 4 annexes.

The aim of the thesis is to develop a VPN Gateway for Secure IoT Node Connectivity with support for secure tunneling, traffic routing, basic access control, node authentication, status monitoring, and event logging.

The work applies the methods of system analysis, comparative analysis of VPN and overlay solutions, network tunneling and routing, traffic filtering using firewall tools, virtualization of the Linux network environment, as well as software tools for implementing network services in Go, Python, and Bash.

The subject area of secure IoT node connectivity is considered, and a review of existing solutions is carried out.

A VPN gateway architecture has been developed, including a local IoT node segment, a Raspberry Pi-based VPN gateway, an external network, a remote server or administrator, as well as auxiliary DNS and NTP services. The algorithmic support of the system has been developed. The information support of the system has been formed. An experimental prototype of the VPN gateway has been implemented.

The functionality of the developed prototype has been tested.

Keywords: INTERNET OF THINGS, IoT, VPN GATEWAY, RASPBERRY PI, WIREGUARD, ROUTING, TRAFFIC PROTECTION, IPTABLES, GO, PYTHON, TELEMETRY, EDGE DEVICE.

ЗМІСТ

ВСТУП

Інтернет речей дедалі частіше використовується в побутових, промислових і сервісних системах, де окремі пристрої збирають дані, передають їх мережею та взаємодіють із віддаленими сервісами. Через це на перший план виходить завдання безпечного підключення таких пристроїв до мережі та стабільної передачі даних.

В реальних IoT-системах вузли нерідко розміщуються на різних об’єктах або в окремих мережевих сегментах, тому потребують віддаленого доступу і централізованого контролю. Тому пряме підключення через публічні мережі створює додаткові ризики, пов’язані з несанкціонованим доступом, перехопленням трафіку, підміною даних, ботнет-активністю, а також складністю роботи через приватну адресацію і NAT. Для розв’язання цієї проблеми використовують VPN-шлюз, який забезпечує захищений тунель, маршрутизацію між підмережами, базовий контроль доступу та підтримку віддаленого адміністрування.

Для організації захищеного підключення найчастіше використовують OpenVPN, IPsec, WireGuard, а також overlay-рішення, які спрощують роботу з віддаленими вузлами. Для компактного шлюзу на базі Raspberry Pi доцільно використати WireGuard, тому що він має просту конфігурацію, невеликі накладні витрати та достатню продуктивність для IoT-сценаріїв.

Актуальність зумовлена потребою в створенні доступного, компактного та достатньо надійного VPN-шлюзу, здатного забезпечити безпечне підключення IoT-вузлів через публічні мережі без суттєвого ускладнення інфраструктури. Використання Raspberry Pi як апаратної основи дозволяє поєднати невисоку вартість, гнучкість налаштування та можливість практичної реалізації основних функцій шлюзу, зокрема тунелювання, маршрутизації, контролю доступу, моніторингу стану та журналювання подій.

Метою кваліфікаційної роботи є розробка VPN-шлюзу для безпечного підключення IoT-вузлів через публічні мережі з підтримкою захищеного тунелювання, маршрутизації трафіку, базового контролю доступу, автентифікації вузлів, моніторингу стану та журналювання подій.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати предметну область безпечного підключення IoT-вузлів та розглянути типові сценарії віддаленого доступу;

– виконати огляд і аналіз існуючих VPN- та overlay-рішень для IoT-середовища;

– дослідити підходи до захисту трафіку IoT-вузлів;

– сформувати постановку задачі та визначити вимоги до розроблюваного прототипу;

– розробити архітектуру VPN-шлюзу і визначити склад його основних модулів;

– розробити алгоритмічне забезпечення функціонування системи;

– розробити інформаційне забезпечення;

– обрати програмні та апаратні засоби для практичної реалізації;

– реалізувати експериментальний прототип VPN-шлюзу;

– провести тестування функціональності системи та оцінити її затримку, пропускну здатність, стійкість до збоїв і придатність до edge-застосування.

Об’єктом дослідження є процес підключення IoT-вузлів через публічні мережі із застосуванням VPN-шлюзу.

Предметом дослідження є методи та засоби побудови VPN-шлюзів для захищеного тунелювання, маршрутизації трафіку, контролю доступу в IoT-середовищі.

Практичне значення одержаних результатів полягає в створенні експериментального прототипу VPN-шлюзу на платформі Raspberry Pi або іншому компактному edge-пристрої для захищеного підключення локальних IoT-сегментів до віддаленої інфраструктури.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 СТАН ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Типова IoT-система складається з багатьох кінцевих вузлів, які збирають дані з середовища, передають їх у мережу та, за потреби, виконують команди керування. Такими вузлами можуть бути датчики температури, вологості, освітленості, руху, реле, контролери виконавчих механізмів, камери, лічильники та інші вбудовані пристрої. Для більшості таких вузлів характерні обмежені обчислювальні ресурси, невеликий обсяг пам’яті, залежність від стабільності мережевого каналу та потреба в енергоощадному режимі роботи [1, 2]. Приклад подібної структури наведено в [3] на основі системи розумного будинку, де сенсори та виконавчі пристрої об’єднані в єдину мережу. В такій архітектурі кінцевий вузол виконує окрему прикладну функцію, а координація, зберігання та захист даних переважно переносяться на шлюз або хмарний сервіс. [2, 3].

Для IoT-систем важливими є сценарії віддаленого підключення. На практиці це означає, що вузли можуть працювати не в одній локальній мережі, а бути рознесеними між різними майданчиками, приміщеннями або географічно віддаленими об’єктами. Один із типових варіантів полягає в тому, що група сенсорів передає дані про навколишній світ на локальний шлюз, а вже шлюз пересилає зібрані дані до сервера або хмарної платформи [2].

Інший сценарій передбачає дистанційне керування фізичним обладнанням через проміжний вузол, що представлено в роботі [4], де Raspberry Pi використовується як основа системи віддаленого доступу до лабораторного обладнання. Для VPN-шлюзу важливим є ще один сценарій, описаний в роботі [5], коли потрібно безпечно з’єднати кілька віддалених сегментів мережі або забезпечити захищений доступ окремих клієнтів до локальної інфраструктури через публічну мережу. В такому випадку VPN не просто дає доступ до вузлів, а формує захищений канал для службового трафіку, телеметрії, команд керування і адміністрування (рисунок 1.1).

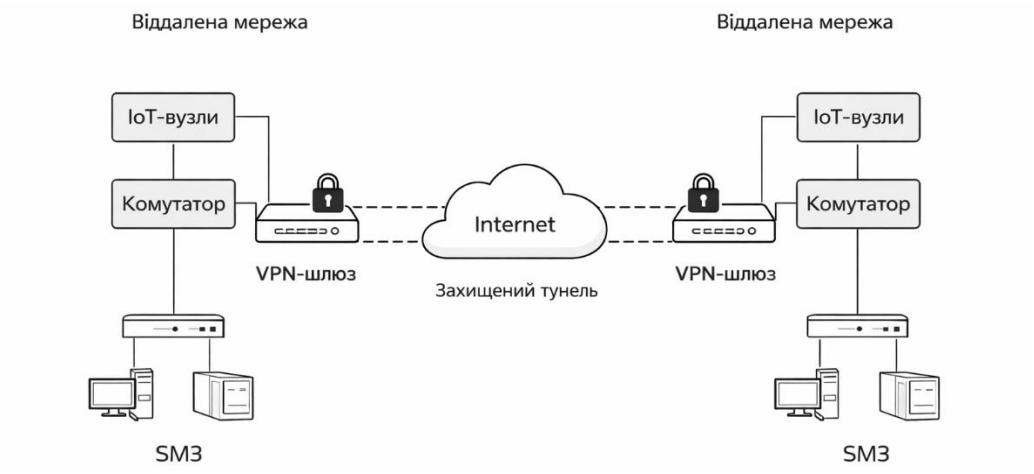


Рисунок 1.1 – Класична схема з’єднання двох віддалених мереж через захищений VPN-тунель

Безпека є однією з головних проблем IoT-систем, так як значна частина пристроїв працює на периферії мережі й часто не має достатнього фізичного та програмного захисту. В роботах [1, 6] показується, що передавання всіх даних безпосередньо в хмару створює додаткові ризики, пов’язані із затримками, перевантаженням каналів, залежністю від центральної інфраструктури та збільшенням площини атаки. Для IoT характерні такі загрози, як несанкціонований доступ до вузлів, перехоплення трафіку, підміна даних, компрометація ключів, відмова в обслуговуванні, ботнет-активність, а також фізичні та апаратні атаки на периферійні пристрої [6, 7]. В роботі [7] окремо показано, що IoT-пристрої можуть використовуватися як частина ботнетів для DoS та DDoS атак, а відомі приклади такі як Mirai доводять, що навіть відносно прості й масові пристрої здатні ставати джерелом серйозних мережевих інцидентів. Тому для безпечного підключення IoT-вузлів важливо розглядати не лише логічний захист каналу, а і питання автентифікації, сегментації мережі, журналювання подій та обмеження доступу до служб керування [1, 6, 8].

В сучасних IoT-системах важливим є периферійний, або туманний, рівень обробки даних. Його використовують для перенесення частини обробки, контролю та захисних функцій ближче до кінцевих пристроїв. В роботах [6] і [8] туманні обчислення розглядаються як проміжний рівень між хмарою та Інтернетом речей, який може виконувати роль шлюзу. Такий підхід дозволяє зменшити затримки,

знизити навантаження на центральний сервер, частково локалізувати обробку даних і швидше реагувати на події в мережі [1, 6]. Для IoT-шлюзу це означає виконання кількох практичних функцій одночасно. Він агрегує трафік від вузлів, може перетворювати протоколи, фільтрує небажані з'єднання, реалізує базові правила доступу, веде облік підключень та передає до вищого рівня вже впорядковані дані [2, 6]. У випадку VPN-шлюзу до цього додається побудова захищеного тунелю між локальним сегментом IoT і зовнішньою мережею, що важливо для віддалених або слабко контрольованих майданчиків [5].

Як апаратна платформа для такого шлюзу Raspberry Pi є одним із найбільш доцільних варіантів. В роботі [2] зазначено, що ця платформа цікава завдяки вдалому співвідношенню обчислювальних можливостей і вартості, високій доступності, поширеності та великій спільноті користувачів. Практична цінність Raspberry Pi також полягає в тому, що вона працює під керуванням Linux, має мережеві інтерфейси, GPIO і підтримує різні способи взаємодії з периферією, що робить її зручною для ролі концентратора, контролера або шлюзу [2, 4, 3]. В роботі [4] вибір Raspberry Pi обґрунтовано як основу для системи дистанційного доступу, а в [3] ця платформа розглядається як доступний базовий вузол для побудови IoT-рішення без дорогого спеціалізованого обладнання. Водночас для задач VPN-шлюзу потрібно враховувати і певні обмеження. Обмежені ресурси одноплатового комп'ютера, навантаження від шифрування, пропускна здатність каналу та стабільність з'єднання безпосередньо впливають на реальну ефективність системи [5]. Тому Raspberry Pi доцільно розглядати не як універсальний сервер, а як компактний edge-пристрій, який добре підходить для побудови локального захищеного вузла доступу до IoT-мережі.

Отже, побудова VPN-шлюзу для IoT-вузлів включає роботу малопотужних пристроїв, захист трафіку, маршрутизацію та контроль віддаленого доступу. З одного боку, існує велика кількість малопотужних IoT-вузлів, які потребують надійного і постійного зв'язку. З іншого боку, відкриті мережі й віддалене розміщення пристроїв створюють ризики безпеці. Саме тому в edge-архітектурі особливого значення набуває шлюз, який поєднує функції доступу, маршрутизації,

локального контролю та захисту трафіку. Raspberry Pi в цьому випадку є придатною платформою для реалізації такого рішення, оскільки поєднує доступність, достатню гнучкість і можливість інтеграції з реальними IoT-вузлами [5, 2, 4].

1.2 Огляд і аналіз існуючих рішень

Сучасні рішення для побудови VPN-з’єднань можна поділити на дві групи. До першої належать класичні протоколи, такі як OpenVPN, IPsec та WireGuard. До другої групи належать сучасні overlay-платформи, які створюють захищений тунель і спрощують виявлення вузлів, обхід NAT та підключення без публічної IP-адреси. До таких рішень відносяться Tailscale, ZeroTier і близькі до них платформи Headscale або Netbird [9, 10] (рисунок 1.2).

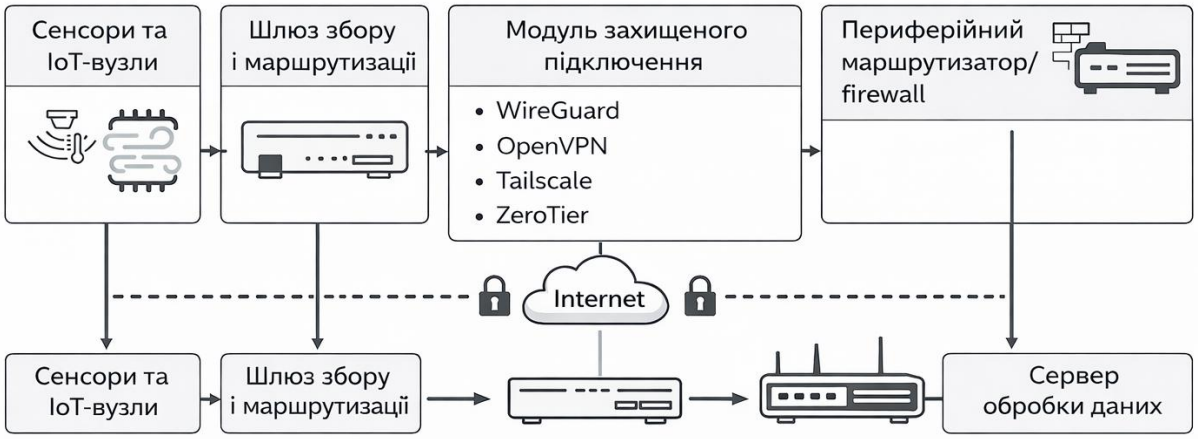


Рисунок 1.2 – Функціональна схема організації захищеного підключення IoT-вузлів через VPN та overlay-рішення

Для безпечного підключення IoT-вузлів важливими є наявність шифрування, простота розгортання на реальному обладнанні, невелика додаткова затримка, стабільність у нестійких мережах і можливість роботи на малопотужних платформах.

IPsec є одним із найстаріших і найпоширеніших рішень. Його перевагою є зрілість, підтримка на рівні операційних систем і можливість використання без стороннього клієнта в багатьох сценаріях. В роботах [11] і [12] відзначено, що IPsec добре підходить для корпоративного середовища в комбінації з IKEv2 або L2TP. Але налаштування є відносно складним, а в окремих конфігураціях додаткова інкапсуляція негативно впливає на швидкість передавання даних [11]. В дослідженні [13] IPsec показав результати за пропускною здатністю, посівши друге місце після WireGuard, але за затримкою він був найкращим рішенням. Це означає, що IPsec є сильним варіантом там, де важливі сумісність, зрілі стандарти і стабільне корпоративне адміністрування, але для компактних IoT-рішень він не завжди є найпростішим.

OpenVPN також широко застосовується завдяки універсальності та підтримці різних платформ. Його перевагою є те, що він давно використовується, добре документований і проходив незалежні перевірки безпеки [11, 14]. Протокол може працювати як через TCP, так і через UDP, що дає гнучкість під час розгортання. Але в більшості розглянутих робіт саме OpenVPN виявився важчим у налаштуванні та обслуговуванні, ніж сучасніші альтернативи [12, 11, 15]. Для IoT-рішень це суттєво, оскільки складніша конфігурація збільшує ймовірність помилок і підвищує вимоги до ресурсів шлюзу. В роботі [16] під час тестів в IoT-середовищі OpenVPN мав значно більшу затримку, ніж WireGuard, а в мережі GSM теж поступався за швидкістю реакції. В роботі [13] OpenVPN показав найнижчу пропускну здатність серед порівнюваних рішень WireGuard, IPsec, Tinc та OpenVPN, тобто OpenVPN є слабким або непридатним.

В більшості розглянутих досліджень WireGuard демонструє найкраще поєднання продуктивності, простоти налаштування та криптографічної стійкості. Його в більшості розглядають, як сучасну альтернативу OpenVPN та IPsec, яка має простішу архітектуру, менший код і фіксований набір сучасних криптографічних примітивів [17, 14]. В роботі [17] показано, що WireGuard підходить для слабких або обмежених пристроїв, тому що має менші накладні витрати та простіше розгортається. Тобто він добре в’яжеться з IoT-середовищем, де пристрої мають

мало пам’яті та не дуже високу обчислювальну потужність. В статті [16] WireGuard показав найменшу затримку і найкращі результати за передаванням трафіку через оптичний канал. В роботі [15] WireGuard дав найкращий середній час відповіді серед кількох VPN-рішень. В дослідженні [13] цей пакет також продемонстрував найвищу пропускну здатність, низький RTT і майже нульовий рівень повторних передавань. Крім того окремою перевагою є криптографічна стійкість. WireGuard використовує сучасні механізми на базі ChaCha20, Poly1305 і Curve25519, що зменшує ризик помилок під час вибору алгоритмів і робить конфігурацію більш передбачуваною [17, 15]. Для IoT це важливо, тому що спрощена конфігурація зменшує ризик помилок під час розгортання та супроводу системи.

Tailscale та ZeroTier можна розглядати як платформи для спрощення підключення вузлів в складних мережевих умовах [16, 10]. Такий підхід є корисним для IoT-сценаріїв, де пристрої працюють в домашніх мережах, за мобільним інтернетом або за CGNAT. В роботі [9] показано, що Tailscale і ZeroTier поступаються WireGuard за продуктивністю, але виграють за зручністю розгортання. Tailscale базується на WireGuard, проте в реальних тестах через додаткову інфраструктуру він не повторює його максимальні показники [9]. ZeroTier теж є простим у використанні та дає зручну логіку віртуальної мережі, але за швидкістю передавання даних зазвичай не був лідером [9, 15]. Водночас в роботі [10] показано, що в умовах великої затримки або втрат пакетів ZeroTier і Headscale можуть поводитися краще, ніж деякі інші mesh-рішення, особливо коли йдеться про TCP-трафік. Тобто їхня перевага не в максимальній швидкості, а в зручності, гнучкості та кращій поведінці в окремих мережевих сценаріях [8].

Якщо оцінювати ці рішення з позиції IoT-середовища, найбільш збалансованим варіантом є WireGuard. Він поєднує просте розгортання, сучасну криптографію, високу пропускну здатність і низьку затримку [9, 17, 13, 15]. OpenVPN варто розглядати тоді, коли важливі сумісність, і підтримка різних платформ, але потрібно прийняти вищі накладні витрати [11, 14]. IPsec підходить для більш класичних корпоративних схем і добре інтегрується в наявну мережеву інфраструктуру, але часто вимагає більш професійного адміністрування [11, 12],

[13]. Tailscale, ZeroTier та подібні платформи підходять тоді, коли пріоритетом є швидке підключення віддалених вузлів без складного налаштування маршрутизаторів [16, 10].

Огляд існуючих рішень показує, що універсального протоколу для всіх випадків немає. Але якщо орієнтуватися саме на VPN-шлюз для безпечного підключення IoT-вузлів на платформі Raspberry Pi, найбільш доцільним базовим рішенням є WireGuard. Він краще відповідає вимогам до низької затримки, невеликого навантаження на систему і простоти підтримки. При цьому Tailscale або ZeroTier можуть розглядатися як практичні альтернативи у випадках, коли потрібно легкість віддаленого підключення і робота за NAT, а OpenVPN та IPsec доцільно залишати для сценаріїв, де важливими є сумісність з наявною інфраструктурою та консервативний підхід до мережевої безпеки [9, 11], [13, 10].

1.3 Аналіз підходів до захисту трафіку IoT-вузлів

В реальній IoT-системі захист трафіку передбачає закриття каналу передавання даних від стороннього доступу, автентифікацію вузлів, роботу через приватні мережі та NAT, а також збереження прийнятної швидкодії на обмежених пристроях. Тому в сучасних IoT-системах використовують поєднання VPN, тунелювання, механізмів автентифікації, сертифікації вузлів і додаткових засобів мережевого доступу [18, 19, 20].

Найбільш поширеним підходом є побудова захищеного тунелю між сегментами мережі або між IoT-шлюзом і серверною частиною. Це дозволяє передавати телеметрію, команди керування та службові повідомлення через публічну мережу так, ніби вузли знаходяться в єдиному захищеному середовищі. В роботі [18] показано, що IPsec VPN залишається одним з базових способів захисту даних в IoT-інфраструктурі (рисунок 1.3), оскільки забезпечує автентифікацію сторін і шифрування трафіку на мережевому рівні. Такий метод можна застосовувати для зв’язку між окремими хостами та для безпечного передавання даних між підмережами. Це робить VPN-шлюз точкою захисту для

групи IoT-вузлів, які працюють за локальним маршрутизатором або edge-пристроєм [18].

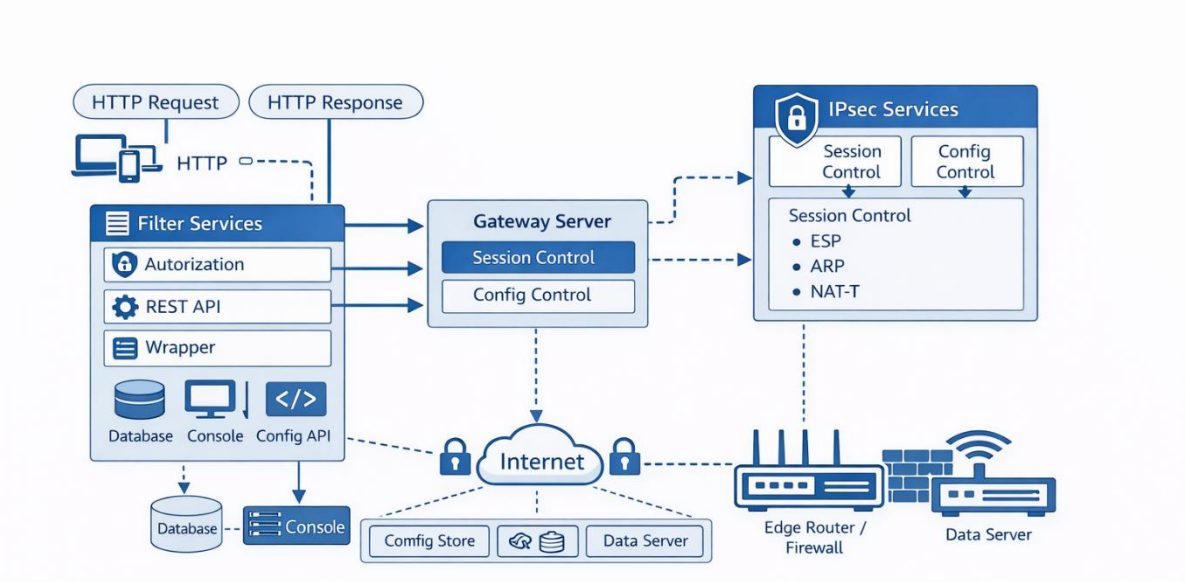


Рисунок 1.3 – Загальна структура IoT-шлюзу для захисту та передавання трафіку

Тунелювання як загальний підхід є ширшим за класичний VPN. Воно включає різні способи інкапсуляції трафіку, коли один тип мережевих пакетів передається всередині іншого каналу. В роботі [21] розглянуто кілька практичних варіантів тунелювання, зокрема WireGuard, GRE разом з IPsec, L2TP разом з IPsec, EoIP разом з IPsec та OpenVPN. З цього випливає що для IoT, сам по собі тунель ще не означає достатній захист. Якщо тунелювання не поєднане з шифруванням і перевіркою сторін, то воно вирішує радше задачу транспортування, а не повноцінної безпеки. Для IoT-вузлів, де обчислювальні ресурси обмежені, найбільший інтерес мають рішення з меншими накладними витратами, але без втрати конфіденційності та цілісності трафіку [21, 19].

Окремою проблемою є робота через приватні мережі та NAT. IoT-пристрої часто підключаються через домашні маршрутизатори, мобільні мережі або провайдерські схеми з приватною адресацією. Через це прямий доступ до вузла ззовні стає складним або неможливим. В роботі [22] показано, що для таких випадків використовують NAT traversal, тобто набір механізмів, які допомагають вузлам знайти один одного і побудувати прямий або напівпрямий захищений канал.

Типові рішення включають STUN, hole punching і relay-підхід. В роботі [23] показано, що hole punching добре працює не завжди, бо поведінка NAT в різних мережах може відрізнятися. Якщо система спирається лише на один спосіб обходу NAT, з'єднання між вузлами не гарантоване. Це важливо, бо вузол може бути фізично доступний, але мережево залишатися недосяжним. Тому потрібна або контрольована топологія, або резервний варіант у вигляді relay-вузла.

Питання маршрутизації між приватними підмережами пов'язане з VPN і NAT traversal. Класичний підхід передбачає з'єднання між двома шлюзами, де кожна сторона знає, які локальні адреси потрібно пропускати через тунель. В сучасних рішеннях [22, 23] також розглядається підхід, за якого поверх фізичної мережі створюється власний логічний адресний простір. Недоліком таких рішень є те, що overlay-мережа додає службовий шар, створює додаткові накладні витрати та залежить від коректної роботи механізмів виявлення вузлів, сертифікатів і NAT traversal.

Захист трафіку починається з шифрування каналу і з механізму довіри до самого вузла. Якщо система не має надійного способу ідентифікації пристрою, то захищений тунель не гарантує, що до мережі під'єднався саме дозволений вузол. В роботі [24] розглянуто керування сертифікатами і введення пристрою в експлуатацію на великому масштабі. Показано, що для IoT важливими є сертифікати X.509 та процедури початкового, оновлення, відкликання та повторної видачі. Також підкреслено, що частину важких операцій доцільно переносити на IoT-шлюз, а не виконувати безпосередньо на кінцевому малопотужному пристрої. Подібну логіку показано в дослідженні [25], де запропоновано підхід до конфіденційного обміну сертифікатами для IoT-пристроїв із використанням IOTA. Практична цінність такого підходу полягає в тому, що сертифікація пов'язує мережевий доступ із перевіреною ідентичністю пристрою.

Для IoT-трафіку важливий також багаторівневий захист. В [19] підкреслено, що використання лише VPN або тунелювання не забезпечує повного захисту, якщо вразливими залишаються бездротовий канал, прикладний протокол або медіапотік. Тому захист трафіку включає такі моменти, як шифрування даних, механізми

автентифікації, захист RTSP-потоків і протидія атакам на бездротове середовище. Для IoT-систем це актуально, бо значна частина вузлів працює саме через Wi-Fi або інші бездротові канали, де ризик перехоплення або придушення сигналу є вищим.

Окреме місце серед сучасних підходів займає QUIC. В роботі [26] показано, що QUIC працює поверх UDP, використовує TLS 1.3, підтримує швидке встановлення сесії та краще пристосований до проходження через NAT і міжмережеві екрани, ніж частина старіших підходів, крім того QUIC може працювати на ресурсно обмежених IoT-платформах.

Отже, захист IoT-трафіку краще будувати, як поєднання кількох механізмів. VPN і тунелювання забезпечують закритий канал. NAT traversal і overlay-підходи допомагають працювати у складних мережевих умовах. Сертифікація вузлів додає перевірену ідентичність. Маршрутизація між приватними підмережами дає змогу об’єднати рознесені сегменти в єдину захищену систему.

1.4 Постановка задачі

Аналіз предметної області та існуючих рішень показав, що для безпечного підключення IoT-вузлів потрібно організувати мережевий доступ до пристроїв та одночасно забезпечити захист трафіку, перевірку справжності вузлів, стабільну роботу в умовах приватних мереж і NAT, а також прийнятну продуктивність на компактній edge-платформі. Для цього доцільно використовувати VPN-шлюз, який виконує роль проміжної довіреної ланки між локальною мережею IoT-пристроїв і зовнішньою інфраструктурою.

Основною метою роботи є розробка VPN-шлюзу на базі Raspberry Pi для безпечного підключення IoT-вузлів через публічні мережі з підтримкою захищеного тунелювання, базового контролю доступу, стабільної маршрутизації трафіку та можливості подальшого адміністрування. Вибір Raspberry Pi зумовлений тим, що ця платформа є доступною, гнучкою для ролі gateway і водночас має обмежені обчислювальні ресурси. Отже, під час розробки потрібно враховувати обмеження за обсягом пам’яті, навантаженням на процесор,

пропускною здатністю каналу та енергоспоживанням, особливо під час шифрування і підтримки кількох з’єднань одночасно.

На основі цього сформовано вимоги до розроблюваного VPN-шлюзу. Функціонально система повинна забезпечувати підключення віддалених IoT-вузлів або локальних сегментів через VPN, маршрутизацію трафіку між приватними підмережами, базову автентифікацію вузлів, збереження та використання конфігурацій, а також моніторинг стану з’єднання.

Шлюз має підтримувати сучасні криптографічні механізми, зберігати працездатність під час типових мережевих збоїв, працювати за NAT і не створювати надмірних затримок під час передавання телеметрії.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Розробити загальну архітектуру VPN-шлюзу та визначити склад його основних модулів.

2. Обрати підхід до захищеного тунелювання, маршрутизації трафіку та роботи через приватні мережі і NAT.

3. Передбачити механізми автентифікації та перевірки довіри до підключених вузлів.

4. Реалізувати прототип VPN-шлюзу та перевірити його роботу в типовому IoT-сценарії.

5. Провести оцінювання стабільності, затримки, пропускної здатності та загальної придатності рішення для edge-застосування.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Розробка архітектури VPN-шлюзу

Архітектура VPN-шлюзу має поєднувати простоту розгортання на Raspberry Pi з достатнім рівнем захисту та керованості мережевого трафіку. З одного боку, система має бути достатньо простою для розгортання на компактній платформі Raspberry Pi. З іншого боку, вона повинна забезпечувати надійний захист трафіку, стабільну маршрутизацію, віддалене адміністрування і коректну роботу в умовах реальної мережі, де можуть бути NAT, нестабільний інтернет-канал і обмежені ресурси. Тому VPN-шлюз має об’єднувати функції безпечного доступу, мережевої взаємодії та базового контролю стану підключених пристроїв.

Загальна структура системи (рисунок 2.1) включає локальний сегмент IoT-вузлів, VPN-шлюз на Raspberry Pi, зовнішню мережу, віддалений сервер або адміністратора, а також допоміжні мережеві сервіси. IoT-вузли можуть бути датчиками, контролерами, виконавчими пристроями або невеликими мікроконтролерними платформами, що передають телеметрію та приймають команди керування. В локальній мережі вони взаємодіють або безпосередньо зі шлюзом, або через проміжний комутатор чи Wi-Fi-точку доступу. Raspberry Pi в цій архітектурі виконує роль довіреного проміжного вузла, через який проходить зовнішній доступ до локального сегмента.

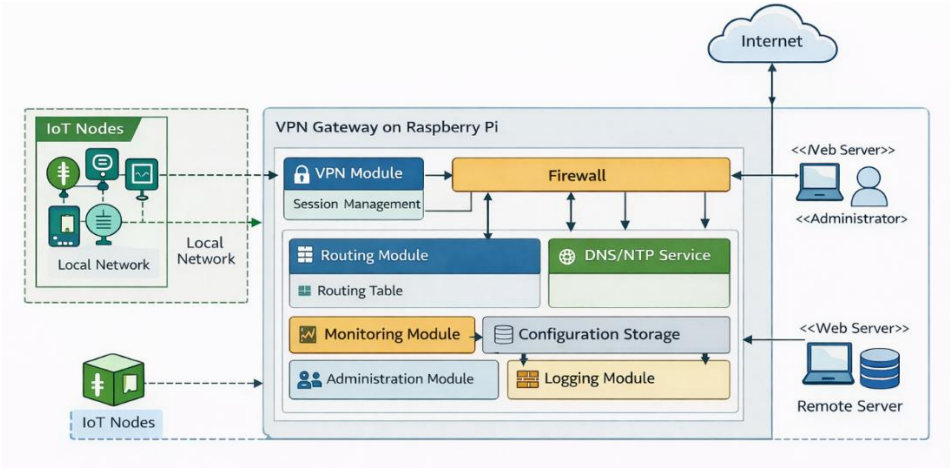


Рисунок 2.1 – Компонентна діаграма архітектури VPN-шлюзу для безпечного підключення IoT-вузлів

На рівні мережевої побудови Raspberry Pi повинен мати щонайменше два логічні напрями взаємодії. Перший напрям пов’язаний із внутрішньою мережею, де розміщені IoT-вузли. Другий напрям пов’язаний із зовнішньою мережею або інтернетом, через які формується захищене VPN-з’єднання з віддаленим сервером або робочим місцем адміністратора. Це дозволяє ізолювати локальний сегмент від прямого виходу в публічну мережу та зосередити функції маршрутизації і захисту в одному місці. В результаті IoT-вузли можуть працювати в приватній адресації, а зовнішній доступ до них відбувається лише через VPN-шлюз.

Ключовим елементом архітектури є VPN-модуль. Він створює захищений тунель між Raspberry Pi і віддаленою стороною. В базовому варіанті віддаленою стороною може бути сервер обробки даних, центральний вузол керування або персональний пристрій адміністратора. Архітектура повинна дозволяти як сценарій site-to-site, коли з’єднуються дві підмережі, так і сценарій remote access, коли до шлюзу підключається окремий адміністратор. Для практичної реалізації можна використати сучасний VPN-протокол з невеликим навантаженням на систему, але сама архітектура має залишатися достатньо гнучкою, щоб в разі потреби можна було замінити конкретну реалізацію тунелювання.

Окрему роль у структурі системи відіграють таблиці маршрутизації. Після встановлення VPN-тунелю Raspberry Pi повинен знати, який трафік потрібно залишати в локальній мережі, який спрямовувати у VPN-інтерфейс, а який блокувати. Таблиця маршрутів визначає, як буде досягатися доступ до IoT-вузлів, віддаленого сервера, служб DNS і NTP, а також служб адміністрування. Для цього архітектура повинна передбачати окремі адресні простори для локальної підмережі IoT, VPN-підмережі та, за потреби, службової підмережі управління. Такий поділ спрощує контроль доступу і дозволяє уникати конфліктів адрес при підключенні кількох віддалених майданчиків.

Для коректної роботи системи також використовуються служби DNS і NTP. DNS потрібен для зручності звернення до віддалених сервісів за іменами та для стабільної роботи засобів оновлення, телеметрії та адміністрування. NTP потрібен для того, щоб на всіх пристроях був правильний час. Без цього важче аналізувати

журнали, шукати помилки та забезпечувати коректну роботу окремих мережевих і захисних механізмів.

Важливим елементом архітектури є міжмережевий екран, який обмежує доступ до внутрішнього сегмента IoT-мережі. Його завданням є відкриття порту VPN, реалізація правил, які визначають, який трафік дозволено для встановлення тунелю, які вузли можуть бути доступні через захищений канал, які сервіси можна використовувати віддалено, а які мають бути доступні тільки локально. Через firewall обмежується прямий доступ до служб керування, і дозволяються тільки необхідні протоколи.

З боку функціональної організації VPN-шлюз поділятиметься на кілька внутрішніх підсистем:

1. Перша підсистема відповідає за приймання і передавання мережевого трафіку між локальною мережею та VPN-тунелем.
2. Друга підсистема відповідає за криптографічне з’єднання, керування ключами та підтримку сесії.
3. Третя підсистема реалізує мережеві правила, трансляцію адрес за потреби та фільтрацію пакетів.
4. Четверта підсистема відповідає за сервісні функції DNS, NTP, журналювання та моніторинг стану.
5. П’ята підсистема забезпечує адміністрування, тобто локальне або віддалене налаштування конфігурації, перегляд стану інтерфейсів, маршрутів, журналів і статистики передавання.

Узагальнено архітектуру VPN-шлюзу можна описати через такий принцип: а) IoT-вузли працюють в локальному приватному сегменті, б) Raspberry Pi виступає центральним edge-шлюзом, який встановлює захищений VPN-тунель, обслуговує маршрутизацію, контролює доступ через firewall і забезпечує взаємодію з DNS та NTP. Віддалений сервер або адміністратор працює лише через захищений канал і не звертається безпосередньо до внутрішніх пристроїв. Це дозволяє зберегти простоту розгортання та ізолювати локальну мережу IoT від публічного середовища.

2.2 Алгоритми функціонування основних компонентів

Основну логіку роботи системи формують алгоритми ініціалізації шлюзу, встановлення VPN-тунелю, автентифікації вузлів, маршрутизації трафіку, повторного підключення, контролю доступу та фіксації подій. Разом вони формують логіку роботи шлюзу та забезпечують передачу даних, стабільність, керованість і базовий рівень захисту.

Алгоритм ініціалізації шлюзу запускається під час ввімкнення Raspberry Pi або перезапуску системи (рисунок 2.2).

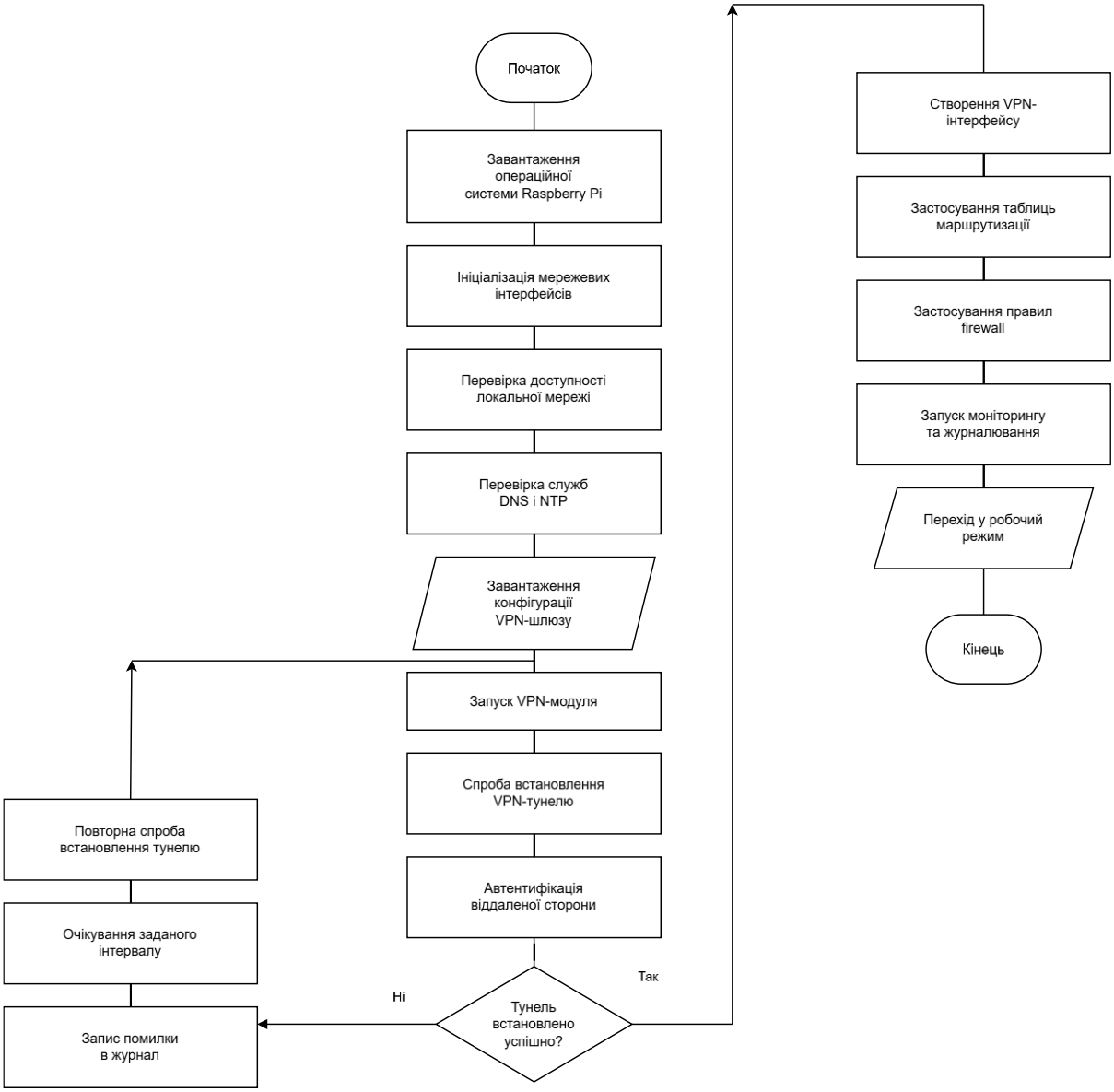


Рисунок 2.2 – Блок-схема алгоритму ініціалізації VPN-шлюзу та встановлення захищеного тунелю

На цьому етапі відбувається завантаження операційної системи, перевірка стану мережевих інтерфейсів, застосування конфігурації VPN-служби, правил маршрутизації та параметрів firewall. Далі система перевіряє доступність локальної мережі, коректність адресації, наявність маршрутів за замовчуванням, а також працездатність служб DNS і NTP. Якщо один із ключових сервісів недоступний, шлюз фіксує помилку в журналі та переходить у режим очікування або повторної спроби. Після успішної ініціалізації вмикаються служби моніторингу та керування, а система переходить до встановлення захищеного з'єднання.

Алгоритм встановлення VPN-тунелю починається з ініціації з'єднання між шлюзом і віддаленою стороною. Спочатку зчитуються параметри конфігурації, які містять адреси вузлів, криптографічні ключі або сертифікати, дозволені мережі та параметри тунелю. Далі шлюз виконує обмін службовими повідомленнями для формування захищеної сесії. Якщо автентифікація сторін успішна, створюється віртуальний VPN-інтерфейс, для якого задається внутрішня адреса і необхідні маршрути. Після цього відкривається передавання службового та прикладного трафіку через захищений канал. Якщо тунель не вдається встановити, система не повинна відкривати доступ до локального сегмента через зовнішню мережу та формується повідомлення про помилку, а модуль повторного підключення запускає нову спробу через заданий інтервал.

Алгоритм автентифікації вузлів потрібен для того, щоб шлюз взаємодіяв лише з довіреними пристроями. В найпростішому варіанті шлюз перевіряє ключі або ідентифікатори вузлів, які намагаються підключитися через VPN. Також можуть використовуватися сертифікати, таблиці дозволених вузлів або списки довірених адрес. Після надходження запиту на підключення шлюз порівнює отримані параметри з локальною конфігурацією. Якщо вузол підтверджено, йому дозволяється подальша взаємодія із заданими ресурсами. Якщо перевірка неуспішна, з'єднання відхиляється, а подія фіксується в журналі.

Алгоритм маршрутизації трафіку визначає, куди спрямовуються пакети після їх надходження на шлюз. Якщо трафік призначений для вузла в локальній підмережі, він передається через локальний інтерфейс. Якщо пакет належить до

віддаленої захищеної мережі, він спрямовується у VPN-тунель. Якщо пакет не відповідає дозволеним маршрутам або порушує правила доступу, він відхиляється. На цьому етапі (рисунок 2.3) можуть застосовуватись правила трансляції адрес, фільтрації службових запитів та ізоляції окремих класів трафіку. Це дозволяє зменшити ризик несанкціонованого доступу і зробити поведінку системи більш передбачуваною.

Рисунок 2.3 –Блок-схема алгоритму обробки трафіку, контролю доступу та відновлення з'єднання

Алгоритм повторного підключення при збоях забезпечує стійкість шлюзу в реальних умовах, коли мережа може тимчасово зникати або працювати нестабільно. Шлюз постійно контролює стан VPN-інтерфейсу, наявність відповіді від віддаленої сторони та коректність обміну службовими пакетами. Якщо виявлено розрив тунелю, перевищення часу очікування або втрату маршруту, активується процедура відновлення. Спочатку виконується повторна перевірка мережевої доступності. Потім система перезапускає VPN-службу або заново проходить процедуру встановлення тунелю. Щоб уникнути перевантаження мережі, повторні спроби доцільно виконувати з невеликою затримкою або зі збільшенням інтервалу між спробами. Після відновлення з'єднання шлюз повторно застосовує маршрути, правила доступу і переходить у нормальний режим роботи.

Алгоритм контролю доступу працює паралельно з маршрутизацією і визначає, які саме запити та сервіси дозволено використовувати. На рівні firewall задаються порти, протоколи, напрямки з'єднань і перелік допустимих джерел. Наприклад, до сервісів адміністрування може бути дозволений доступ лише через VPN, а до окремих IoT-вузлів лише з визначеної підмережі. Якщо пакет або з'єднання не відповідає встановленій політиці, воно блокується. Контроль доступу також може передбачати розділення прав між різними типами користувачів або вузлів, наприклад між звичайним IoT-пристроєм, сервером збору даних та адміністратором.

Алгоритм журналювання подій забезпечує фіксацію всіх важливих станів і дій у системі. До журналу записуються запуск і зупинка служб, спроби встановлення VPN-тунелю, успішні та неуспішні автентифікації, зміни маршрутів, спрацювання правил firewall, відновлення після збоїв і критичні помилки. Журналювання потрібне для подальшого аналізу інцидентів та для звичайної експлуатації системи. За записами можна побачити, коли зникало з'єднання, які вузли намагалися підключитися, чи були спроби несанкціонованого доступу та як поводилася система під навантаженням. Для зручності кожен запис повинен містити час події, тип події, короткий опис і службові параметри, необхідні для діагностики.

В загальному алгоритми роботи основних компонентів визначають послідовність дій VPN-шлюзу від запуску до обробки збоїв і фіксації подій. Ініціалізація підготовлює шлюз до роботи, встановлення тунелю створює захищений канал, автентифікація перевіряє довіреність вузлів, маршрутизація визначає напрями передавання пакетів, повторне підключення підтримує стійкість при збоях, контроль доступу обмежує взаємодію лише дозволеними правилами, а журналювання дає можливість аналізувати стан системи та її події. Саме така узгоджена робота компонентів робить VPN-шлюз придатним для використання в реальному IoT-середовищі.

2.3 Інформаційне забезпечення

Інформаційне забезпечення описує набір даних, з якими працює VPN-шлюз під час налаштування, експлуатації, моніторингу та аналізу подій безпеки. Тому інформаційне забезпечення повинно бути побудоване так, щоб підтримувати початкове налаштування системи та щоденну експлуатацію, моніторинг стану та аналіз подій безпеки.

До вхідних даних належать конфігурації вузлів, криптографічні ключі, IP-адреси, правила маршрутизації, параметри VPN-з’єднання, стани тунелів, службові часові мітки та системні журнали. Конфігурації вузлів містять ідентифікатор пристрою, його роль у системі, допустимі адреси, параметри підключення та перелік доступних ресурсів. Для VPN-шлюзу також важливими є ключі або інші засоби автентифікації, які використовуються для встановлення захищеного тунелю і перевірки довірених вузлів. IP-адреси потрібні як для локального сегмента IoT-вузлів, так і для зовнішнього інтерфейсу шлюзу, VPN-інтерфейсу та віддалених вузлів. Окрему категорію становлять правила маршрутизації, які визначають, через який інтерфейс повинен передаватися трафік, які підмережі є локальними, а які доступні через VPN-тунель.

Основною частиною вхідної інформації є стан тунелів і мережевих інтерфейсів. Система повинна отримувати дані про те, чи активний VPN-зв’язок,

який час встановлення сесії, чи були спроби повторного підключення, яка адреса призначена віртуальному інтерфейсу, а також чи не виникло помилок в роботі маршрутизації або firewall. До вхідної інформації можна також віднести системні журнали, оскільки вони є даними, на основі яких система або адміністратор робить висновок про поточний стан шлюзу, збої чи підозрілі події (рисунок 2.4).

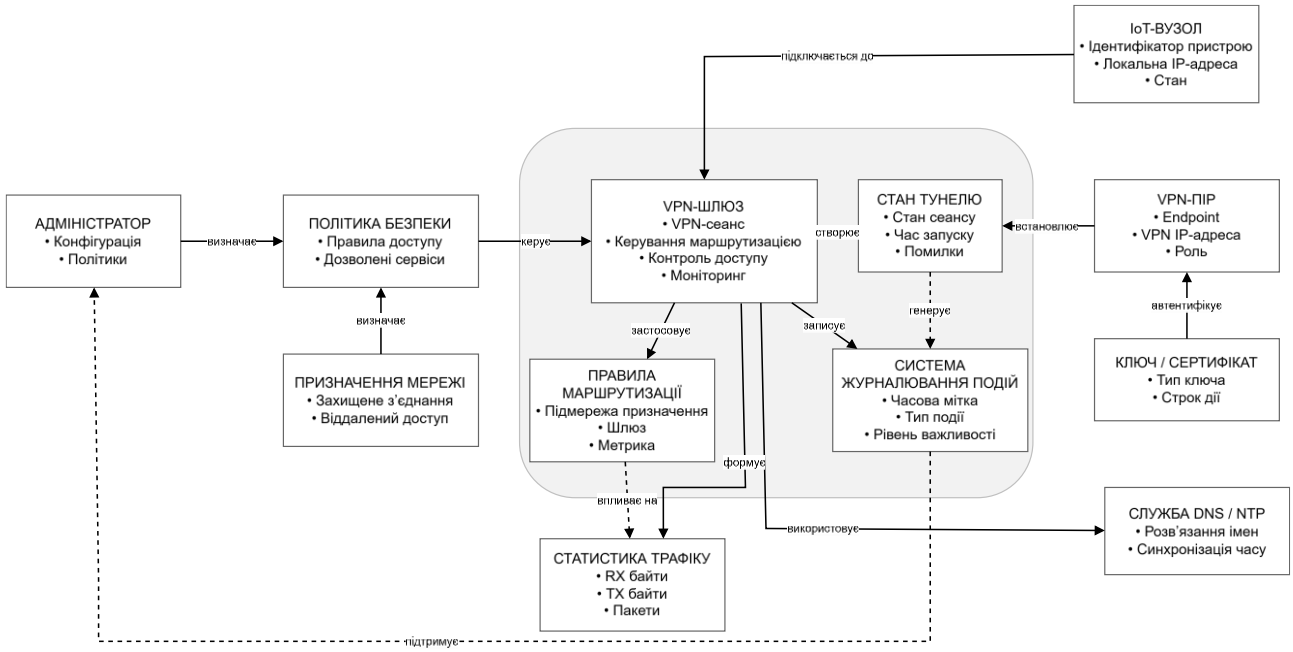


Рисунок 2.4 – Інфологічна модель інформаційного забезпечення VPN-шлюзу

Вихідними даними системи є таблиці стану підключень, журнали безпеки, статистика трафіку та повідомлення про помилки. Таблиці стану підключень показують, які вузли або віддалені сторони зараз підключені до шлюзу, коли було встановлено з'єднання, який статус має тунель, яка адреса використовується та чи є обмеження доступу для цього підключення. Журнали безпеки фіксують спроби входу, успішні та неуспішні автентифікації, зміни конфігурації, спрацювання правил міжмережевого екрана й події відновлення після розриву тунелю. Статистика трафіку потрібна для оцінювання навантаження на систему. Вона може містити обсяг переданих і прийнятих даних, кількість пакетів, середню швидкість передавання, число повторних підключень і частоту збоїв. Повідомлення про помилки використовуються для локального або віддаленого адміністрування і мають інформувати про недоступність VPN-сервера, помилки в ключах, конфлікти

адрес, відсутність маршруту, неправильний час системи або помилки у службових налаштуваннях.

На концептуальному рівні в інфологічній моделі можна визначити кілька основних сутностей. Першою є вузол, який описує IoT-пристрій або віддаленого учасника VPN-мережі. Другою є тунель, що відображає факт існування захищеного каналу між шлюзом і віддаленою стороною. Третьою сутністю є маршрут, тобто правило спрямування трафіку між підмережами та інтерфейсами. Окремо доцільно виділити сутність ключ або облікові дані доступу, яка зберігає інформацію для автентифікації. Ще однією сутністю є журнал подій, що фіксує всі важливі стани системи. Крім цього, логічно ввести сутність статистика трафіку, яка зберігає агреговані показники використання мережі за певний проміжок часу.

Один вузол може мати одну або кілька конфігурацій доступу. Один тунель пов’язаний із конкретним вузлом або групою вузлів. Для кожного тунелю можуть існувати кілька маршрутів, які визначають дозволені напрями передавання трафіку. Кожний вузол або тунель можуть мати пов’язані записи журналу подій та записи статистики. Така інфологічна модель дозволяє не лише зберігати поточні параметри, а й аналізувати історію роботи системи.

На даталогічному рівні цю модель можна реалізувати як реляційну базу даних (рисунок 2.5). Основними таблицями можуть бути nodes, vpn_peers, tunnels, routes, keys, logs і traffic_stats. Таблиця nodes містить ідентифікатор вузла, його назву, тип, роль, локальну адресу, стан та примітки адміністратора. Таблиця vpn_peers описує віддалені сторони підключення, їх VPN-адреси, параметри доступу та прив’язку до конкретного вузла або майданчика. Таблиця tunnels зберігає інформацію про активні або завершені тунелі, час встановлення, статус, причину завершення та службові параметри інтерфейсу.

Таблиця routes містить локальні та віддалені підмережі, шлюз, пріоритет і стан маршруту. Таблиця keys призначена для обліку ключів або записів про сертифікати, часу створення, строку дії та прив’язки до вузла чи VPN-піра.

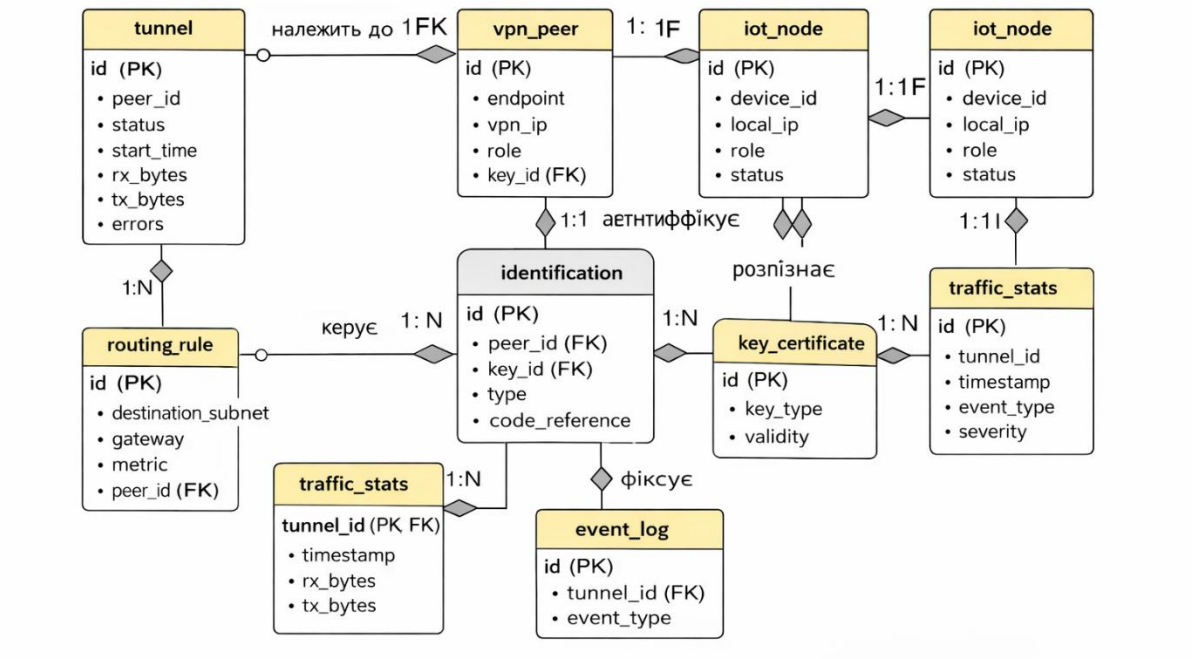


Рисунок 2.5 – ER-діаграма реляційної моделі даних VPN-шлюзу

Таблиця logs зберігає часову мітку, тип події, рівень важливості, опис і посилання на вузол або тунель. Таблиця traffic_stats містить часовий інтервал, обсяг вхідного та вихідного трафіку, кількість пакетів, число помилок і прив’язку до конкретного тунелю або інтерфейсу.

Така організація даних дає змогу зробити систему керованою, придатною до моніторингу і зручною для подальшого аналізу безпеки та стабільності роботи.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Вибір програмних і апаратних засобів

Для реалізації експериментального зразка VPN-шлюзу було обрано програмно-апаратні засоби, які дозволяють відтворити основні функції системи (рисунок 3.1). На етапі реалізації роль апаратної платформи виконувала віртуальна машина з операційною системою Debian 13.

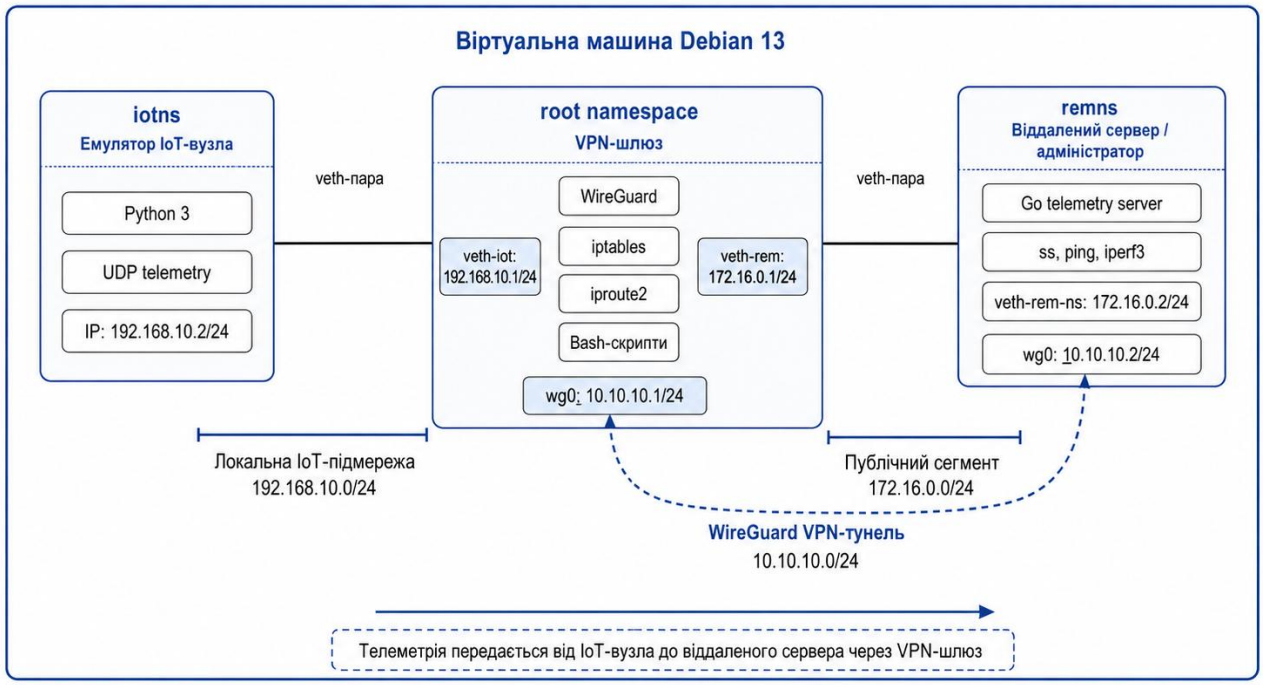


Рисунок 3.1 – Структура експериментального стенда VPN-шлюзу у віртуальному середовищі

Такий підхід дозволив побудувати повноцінний лабораторний стенд, у якому було реалізовано VPN-шлюз, локальний IoT-вузол, віддалений вузол адміністратора та захищений канал зв’язку між ними.

Debian 13 обрано як базову операційну систему, так як вона належить до Linux-сімейства, має стабільне мережеве ядро та підтримує сучасні засоби тунелювання й маршрутизації.

Як основну VPN-технологію обрано WireGuard через просте налаштування, компактну конфігурацію, сучасні криптографічні механізми та високу

продуктивність. WireGuard забезпечив встановлення тунелю між основним середовищем, де працював шлюз, і простором імен remns, який виконував роль віддаленого вузла.

Для реалізації мережевої взаємодії було використано пакет iproute2, а саме команди ip, ip link, ip route та ip netns. Ці засоби дозволили створити окремі мережеві простори імен iotns та remns, зв'язати їх з основним середовищем за допомогою віртуальних інтерфейсів veth, призначити IP-адреси, налаштувати маршрути та організувати передавання трафіку через шлюз.

Для контролю доступу і фільтрації трафіку було використано iptables. За його допомогою реалізовано правило, яке забороняє прямий доступ з публічного сегмента до локальної IoT-підмережі та дозволяє доступ до неї лише через WireGuard-тунель.

Для реалізації серверної частини приймання телеметрії було обрано мову програмування Go, тому що вона підходить для створення мережевих сервісів, має просту модель роботи з сокетами, компілюється в окремий виконуваний файл і не потребує додаткового середовища виконання. В розробленому стенді на Go було реалізовано UDP-сервер, який приймав телеметричні повідомлення на віддаленому вузлі та відображав їх у консольному режимі.

Для створення емулятора IoT-вузла (рисунок 3.2) було використано Python 3. На Python було створено скрипт, який періодично формував телеметричні повідомлення з полями ідентифікатора вузла, часу, температури, вологості та стану пристрою й передавав їх через UDP на віддалений сервер.

Запуск і обслуговування стенда автоматизовано за допомогою Bash-скриптів. Окремі сценарії було створено для старту всього стенду, очищення мережевої конфігурації, запуску сервера телеметрії та запуску емулятора IoT-вузла.

Для перевірки працездатності і вимірювання характеристик системи було використано допоміжні утиліти ping, ss та iperf3.

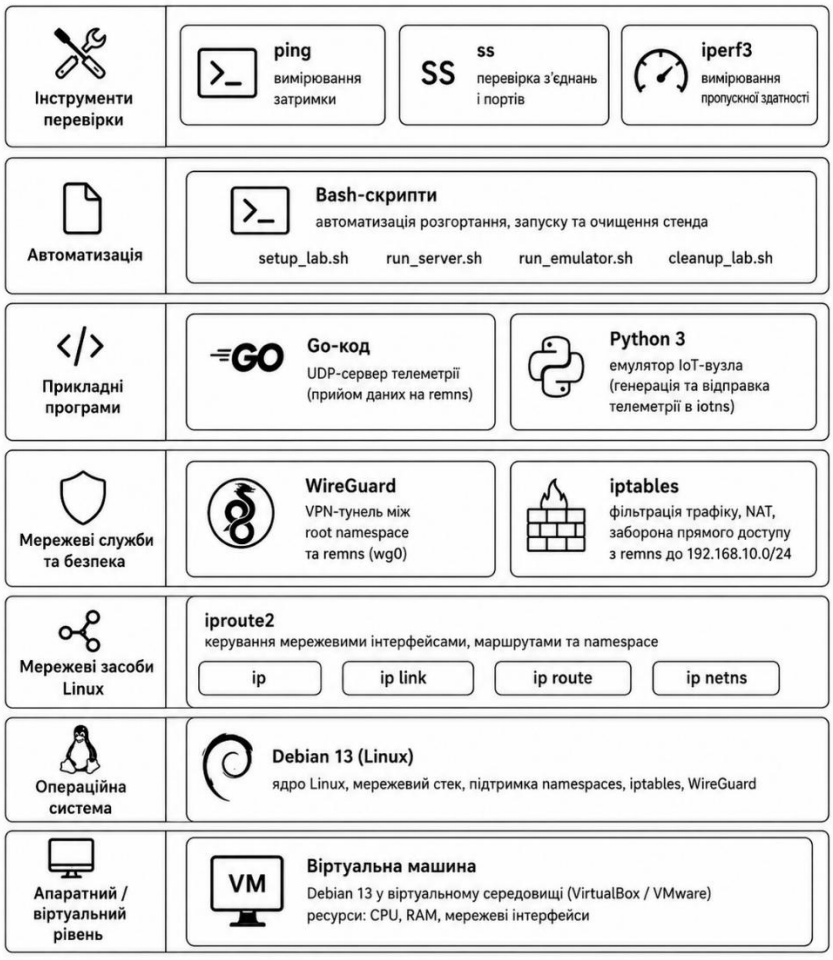


Рисунок 3.2 – Стек програмних засобів експериментальної реалізації VPN-шлюзу

За допомогою ping оцінювалася затримка передавання пакетів через тунель, команда ss дозволяла перевірити прослуховування потрібного порту на віддаленому вузлі, а iperf3 використовувався для вимірювання пропускної здатності захищеного каналу.

3.2 Реалізація ключових модулів

В межах однієї віртуальної машини було побудовано експериментальний стенд, в якому основне середовище виконувало роль VPN-шлюзу, простір імен iotns використовувався як емулятор IoT-вузла, а простір імен remns — як віддалений сервер або вузол адміністратора.

Першим реалізовано модуль підготовки мережевого середовища. Для цього було створено Bash-скрипт setup_lab.sh, який автоматично виконує побудову

всього стенду. У цьому скрипті створюються простори імен iotns і remns, формуються дві пари віртуальних інтерфейсів veth, призначаються IP-адреси і додаються маршрути. Тобто було організовано три логічні сегменти. Локальна IoT-підмережа отримала адресу 192.168.10.0/24, публічний сегмент — 172.16.0.0/24, а WireGuard-тунель — 10.10.10.0/24. Для шлюзу це означає, що він має доступ одночасно до локальної мережі та до віддаленого вузла і може виконувати маршрутизацію між сегментами й виступати центральною точкою безпеки.

Другим компонентом став захищений тунель на основі WireGuard. На етапі налаштування було згенеровано окремі приватні та публічні ключі для шлюзу й віддаленого вузла. Після цього в основному середовищі створювався інтерфейс wg0 з адресою 10.10.10.1/24, а в просторі remns — інтерфейс wg0 з адресою 10.10.10.2/24. Для вузла remns як віддалену точку доступу було задано адресу 172.16.0.1:51820, тобто зовнішній інтерфейс шлюзу. Також було визначено дозволені адреси для тунелю (рисунок 3.3).

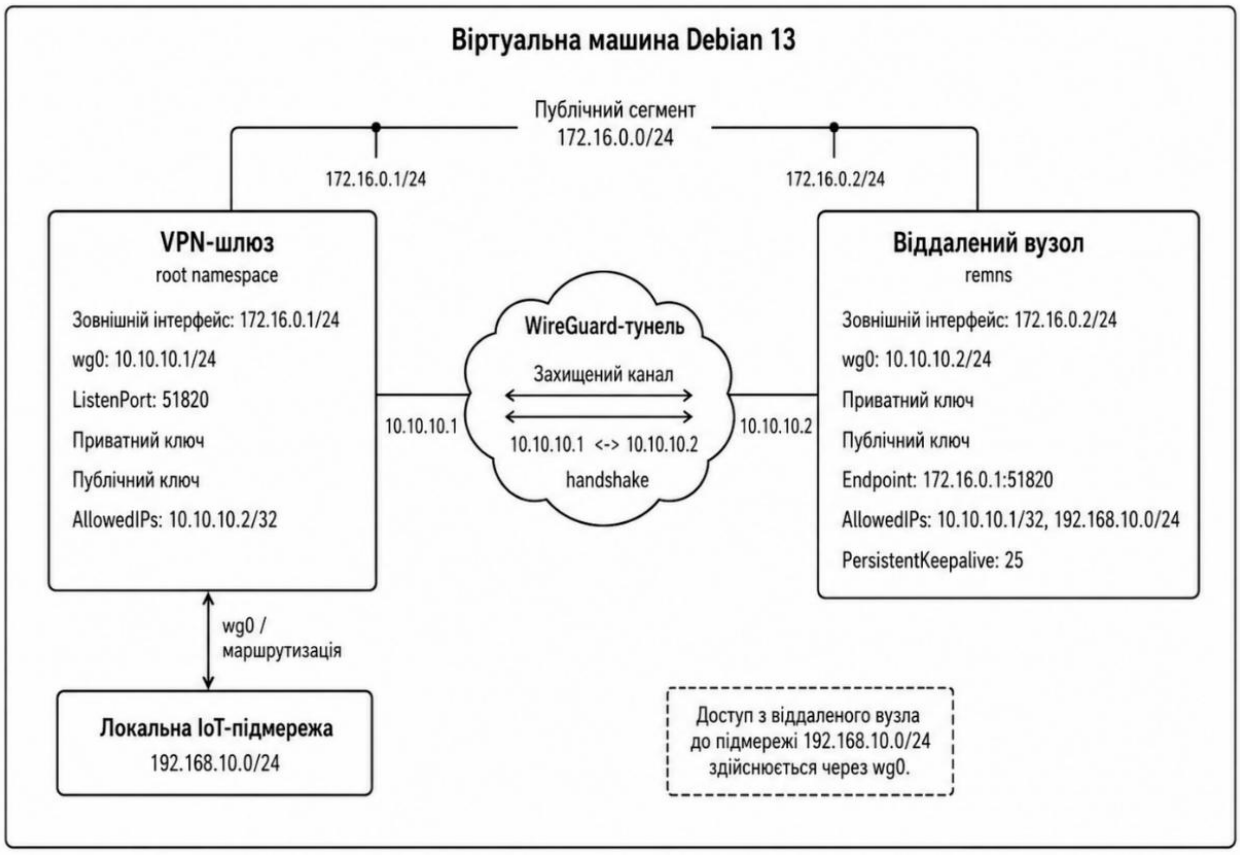


Рисунок 3.3 – Схема налаштування WireGuard-тунелю між шлюзом і віддаленим вузлом

Окремо було реалізовано модуль мережевої маршрутизації. Для цього використовувались засоби iproute2, команди керування маршрутами та інтерфейсами. На шлюзі було ввімкнено net.ipv4.ip_forward, що дозволило пересилати пакети між сегментами. В просторі remns було додано маршрут до мережі 192.168.10.0/24 через інтерфейс wg0. Це забезпечило доступ віддаленого вузла до IoT-підмережі тільки через WireGuard-тунель.

Для контролю доступу було реалізовано окремий firewall-модуль на основі iptables. На рівні правил FORWARD було дозволено трафік між інтерфейсом wg0 і локальним інтерфейсом veth-iot, а також зворотний трафік від IoT-підмережі до VPN-тунелю (рисунок 3.4). Додатково було введено правило, яке забороняє прямий доступ з публічного сегмента через інтерфейс veth-rem до локальної IoT-підмережі. Таке правило забезпечило виконання однієї з основних вимог, а саме доступ до IoT-вузлів має здійснюватися лише через VPN-тунель.

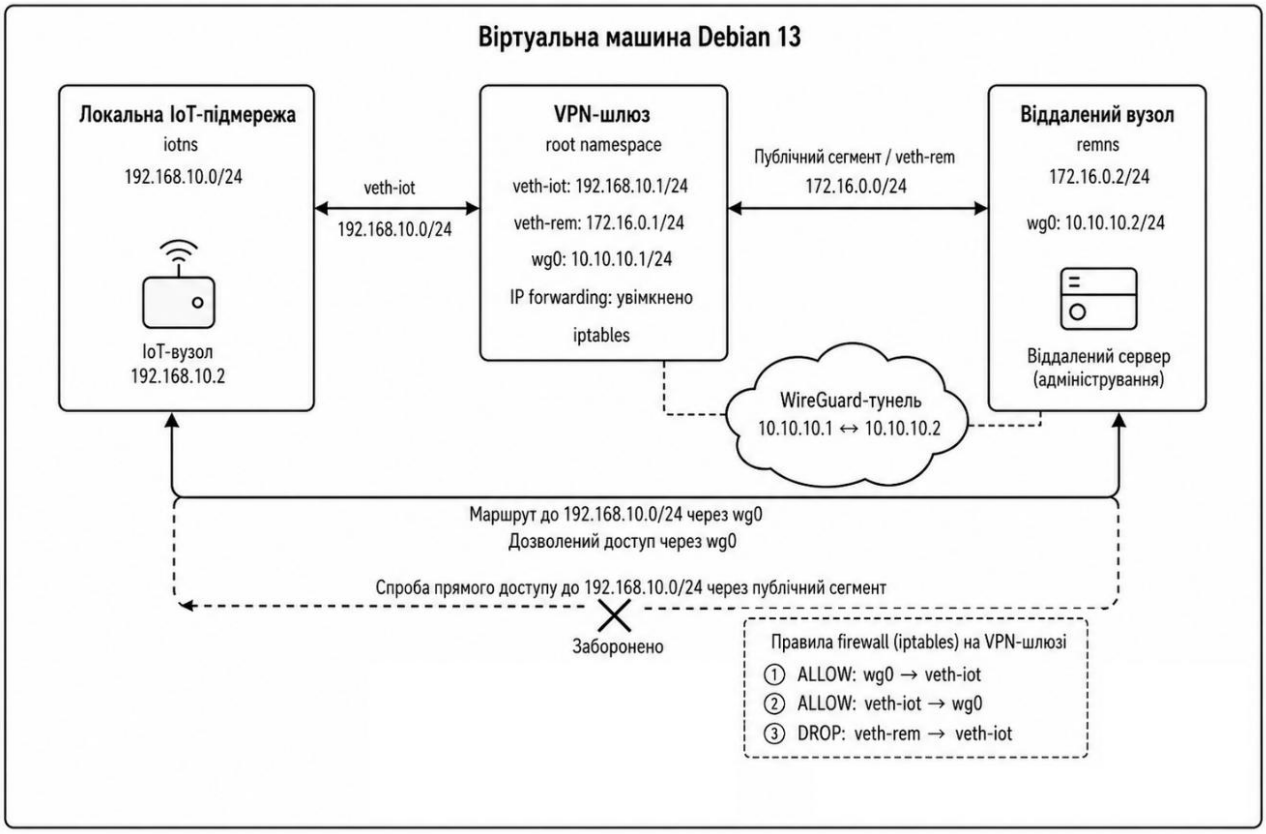


Рисунок 3.4 – Схема маршрутизації і фільтрації трафіку в експериментальному стенді

Для приймання телеметрії було реалізовано серверний модуль мовою Go. Сервер створював UDP-сокет і прослуховував адресу 10.10.10.2:9000 в просторі remns. Після надходження повідомлення він фіксував час приймання, адресу відправника і текст самого повідомлення

Для імітації роботи IoT-пристрою було розроблено Python-модуль iot_emulator.py (рисунок 3.5). В цьому скрипті з певним інтервалом формувалося JSON-повідомлення, яке містило ідентифікатор вузла, часову мітку, температуру, вологість та стан пристрою. Далі це повідомлення передавалося по UDP на сервер 10.10.10.2:9000.

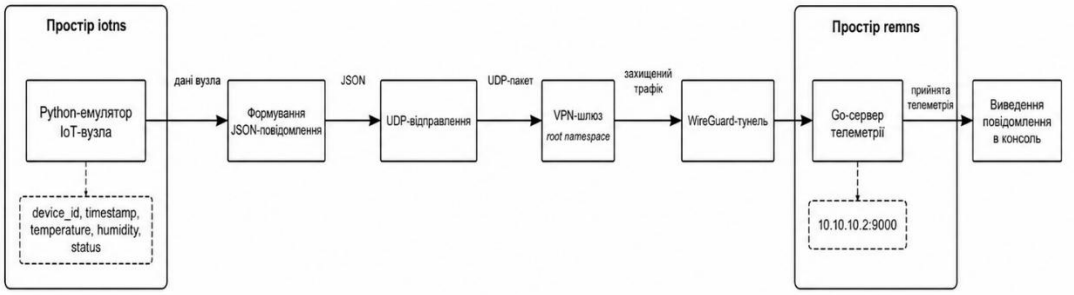


Рисунок 3.5 – Схема взаємодії Go-сервера телеметрії та Python-емулятора IoT-вузла

Для спрощення роботи з стендом додатково створено сервісні скрипти run_server.sh, run_emulator.sh і cleanup_lab.sh. Перший запускає Go-сервер у просторі remns, другий — Python-емулятор у просторі iotns, а третій очищає середовище після завершення роботи, видаляючи інтерфейси, простори імен та правила firewall.

3.3 Тестування функціональності

Після реалізації прототипу перевірено передавання телеметрії, роботу маршрутизації, блокування прямого доступу та відновлення тунелю після збою.

На першому етапі було перевірено базову функціональність передавання телеметрії через VPN-тунель. Після запуску Go-сервера телеметрії у просторі remns

і запуску Python-емулятора в iotns було підтверджено, що телеметричні повідомлення успішно передаються через захищений тунель до віддаленого вузла. Сервер приймав UDP-повідомлення на адресі 10.10.10.2:9000 і виводив їх у консоль разом із часовою міткою, адресою джерела та вмістом повідомлення (рисунок 3.6).

```
root@Debain-  :/home/vpn_gateway_project# ./scripts/setup_lab.sh
[INFO] Creating namespaces...
[INFO] Creating veth pairs...
[INFO] Assigning addresses in root namespace...
[INFO] Assigning addresses in iotns...
[INFO] Assigning addresses in remns...
[INFO] Enabling IP forwarding...
[INFO] Creating WireGuard on gateway...
[INFO] Creating WireGuard in remns...
[INFO] Applying firewall rules...
[INFO] Lab setup completed.
root@Debain-  :/home/vpn_gateway_project# ./scripts/run_server.sh
[START] Telemetry server listening on 10.10.10.2:9000
[2026-05-27 11:33:33] from 192.168.10.2:55775: {"device_id": "iot-node-01", "timestamp": "202
6-05-27T11:33:33", "temperature": 25.18, "humidity": 51.24, "status": "ok"}
[2026-05-27 11:33:36] from 192.168.10.2:55775: {"device_id": "iot-node-01", "timestamp": "202
6-05-27T11:33:36", "temperature": 21.91, "humidity": 58.55, "status": "ok"}
[2026-05-27 11:33:39] from 192.168.10.2:55775: {"device_id": "iot-node-01", "timestamp": "202
6-05-27T11:33:39", "temperature": 26.22, "humidity": 54.39, "status": "ok"}
[2026-05-27 11:33:42] from 192.168.10.2:55775: {"device_id": "iot-node-01", "timestamp": "202
6-05-27T11:33:42", "temperature": 24.29, "humidity": 40.4, "status": "ok"}
```

Рисунок 3.6 – Консольний вивід Go-сервера під час приймання телеметрії через VPN-тунель

На стороні сервера фіксувалась адреса джерела 192.168.10.2, що підтверджувало правильну маршрутизацію трафіку від локального IoT-вузла до віддаленого сервера через VPN-шлюз.

На другому етапі було перевірено, чи може віддалений вузол отримати прямий доступ до локальної IoT-підмережі без використання VPN-маршруту. Для цього в просторі remns було видалено маршрут до мережі 192.168.10.0/24 через інтерфейс wg0. Після цього спроба виконати ping до вузла 192.168.10.2 завершилась повною втратою пакетів. Отриманий результат підтвердив коректну роботу правила міжмережевого екрана, яке блокує прямий доступ з публічного сегмента до локальної IoT-підмережі. Після повернення маршруту через wg0 доступ до вузла був знову успішно відновлений, тобто трафік до IoT-підмережі проходив лише через захищений тунель.

На третьому етапі було перевірено стійкість з’єднання до розриву тунелю. Для моделювання аварійної ситуації інтерфейс wg0 у просторі remns було

переведено в стан DOWN. Після цього нові телеметричні повідомлення перестали надходити на сервер, що підтвердило фактичний розрив з'єднання. Після повторного активування інтерфейсу тунель було відновлено, що підтверджувалось наявністю нового handshake у виводі wg show, а також відновленням передавання телеметрії на сервер.

```
oso@Debain-oso: ~
root@Debain-    :/home/oso# ip netns exec remns ip link set wg0 down
root@Debain-    :/home/oso# ip netns exec remns ip link set wg0 up
root@Debain-    :/home/oso# ip netns exec remns wg show
interface: wg0
  public key: XWUhNjab2EdBfT8xh9R14fiO1CJfOBYdRl4zMdyWAnw=
  private key: (hidden)
  listening port: 38560

peer: /yNCnVXMCIoWvSUSowdrDKCbZNy20wlL27jy5tlCHxQ=
  endpoint: 172.16.0.1:51820
  allowed ips: 10.10.10.1/32, 192.168.10.0/24
  latest handshake: 9 seconds ago
  transfer: 18.92 KiB received, 1.52 KiB sent
  persistent keepalive: every 25 seconds
root@Debain-    :/home/oso#
```

Лістинг 3.2 – Консольний вивід WireGuard після відновлення тунелю

Окремо було проведено вимірювання затримки передавання пакетів через VPN-тунель. Для цього з простору remns було надіслано десять ICMP-пакетів до вузла 192.168.10.2. Усі пакети були успішно доставлені, втрати не спостерігалось. За результатами вимірювання мінімальна затримка становила 1.394 мс, середня — 2.659 мс, максимальна — 4.036 мс. Отримані значення свідчать про стабільну роботу тунелю та невеликий рівень затримки в умовах віртуального стенда.

```
root@Debain-    :/home/oso# ip netns exec remns ping -c 10 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.

--- 192.168.10.2 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9222ms

root@Debain-    :/home/oso#
```

Лістинг 3.3 – Результат вимірювання затримки передавання пакетів через VPN-тунель

Для оцінювання пропускної здатності каналу було використано iperf3. Сервер iperf3 було запущено в просторі iotns, а клієнт — у просторі remns. Під час тесту було підтверджено встановлення TCP-з’єднання з вузлом 192.168.10.2 через VPN-тунель. На стороні відправника було передано 4.54 GBytes даних із середньою пропускною здатністю 3.90 Gbit/s, а на стороні приймача зафіксовано 4.54 GBytes і середню швидкість 3.89 Gbit/s.

```
root@Debain-(   :/home/oso# ip netns exec remns iperf3 -c 192.168.10.2 -t 10
Connecting to host 192.168.10.2, port 5201
[  5] local 10.10.10.2 port 36866 connected to 192.168.10.2 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   495 MBytes  4.15 Gbits/sec  114   1.09 MBytes
[  5]   1.00-2.00   sec   526 MBytes  4.41 Gbits/sec   13    970 KBytes
[  5]   2.00-3.00   sec   518 MBytes  4.34 Gbits/sec    0   1.26 MBytes
[  5]   3.00-4.00   sec   502 MBytes  4.21 Gbits/sec   10   1.12 MBytes
[  5]   4.00-5.00   sec   507 MBytes  4.26 Gbits/sec   20   1003 KBytes
[  5]   5.00-6.00   sec   522 MBytes  4.38 Gbits/sec    0   1.28 MBytes
[  5]   6.00-7.00   sec   541 MBytes  4.54 Gbits/sec    2   1.18 MBytes
[  5]   7.00-8.00   sec   534 MBytes  4.48 Gbits/sec   10   1.03 MBytes
[  5]   8.00-9.00   sec   576 MBytes  4.84 Gbits/sec    0   1.35 MBytes
[  5]   9.00-10.01  sec   515 MBytes  4.28 Gbits/sec    7   1.23 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.01  sec  5.12 GBytes  4.39 Gbits/sec  176             sender
[  5]   0.00-10.01  sec  5.12 GBytes  4.39 Gbits/sec                  receiver

iperf Done.
```

Рисунок 3.4 – Результат вимірювання пропускної здатності каналу через VPN-тунель

Тестування підтвердило, що прототип передає телеметрію через захищений тунель, блокує прямий доступ без VPN, відновлює з’єднання після розриву та забезпечує достатню пропускну здатність для лабораторного IoT-сценарію.

ВИСНОВКИ

При виконанні кваліфікаційної роботи на тему «VPN-шлюз для безпечного підключення IoT-вузлів» було досягнуто таких результатів:

1. Проведено аналіз предметної області безпечного підключення IoT-вузлів, розглянуто особливості їх роботи, типові сценарії віддаленого доступу, основні загрози безпеці, роль шлюзу в edge-архітектурі.

2. Проаналізовано існуючі рішення для організації захищених підключень, таких як WireGuard, OpenVPN, IPsec, Tailscale, ZeroTier та подібні засоби, і виконано їх порівняння за простотою розгортання, продуктивністю, затримкою, криптографічною стійкістю та придатністю для IoT-середовища.

3. Досліджено підходи до захисту IoT-трафіку, VPN-тунелювання, NAT traversal, overlay-мережі, сертифікацію вузлів і багаторівневий контроль доступу.

4. Сформульовано постановку задачі розробки VPN-шлюзу, визначено функціональні та нефункціональні вимоги, обмеження за ресурсами, стабільності та продуктивності.

5. Розроблено архітектуру VPN-шлюзу, яка включає локальний сегмент IoT-вузлів, VPN-шлюз, віддалений сервер або адміністратора, таблиці маршрутизації, firewall, а також допоміжні служби DNS і NTP.

6. Розроблено алгоритмічне забезпечення системи, що включає ініціалізацію шлюзу, встановлення VPN-тунелю, автентифікацію вузлів, маршрутизацію трафіку, контроль доступу, повторне підключення при збоях і логуванні подій.

7. Розроблено інформаційне забезпечення VPN-шлюзу, визначено склад вхідних і вихідних даних, а також сформовано інфологічну та реляційну моделі для збереження конфігурацій.

8. Проведено вибір програмних і апаратних засобів для практичної реалізації. Де за базу вибрано VPN-технологію WireGuard, для маршрутизації використано iproute2, фільтрації трафіку вибрано iptables, серверного модуля сервер на мові Go, для емуляції IoT-вузла — Python, для автоматизації — Bash-скрипти. Для емуляції платформи використано віртуальну машину Debian 13.

9. Реалізовано експериментальний прототип VPN-шлюзу.

10. Для відтворення ізольованих мережевих сегментів створено окремі простори імен для IoT-вузла та віддаленого сервера.

11. Проведено тестування функціональності системи. За результатами тестування середній RTT становив близько 2.659 мс, а середня пропускна здатність каналу за вимірюванням iperf3 досягла 4.39 Gbit/s в межах тестового стенду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fazeldehkordi E., Grønli T.-M. A Survey of Security Architectures for Edge Computing-Based IoT

2. Calvo I. та ін. Building IoT Applications with Raspberry Pi and Low Power IQRF Communication Modules

3. Яременко К. М. Автоматизация планирования умного дома

4. Посашков О., Цимбал О. Розроблення архітектури системи віддаленого доступу до навчального лабораторного обладнання з використанням автоматизованих рішень

5. Gentile A. F. та ін. A VPN Performances Analysis of Constrained Hardware Open Source Infrastructure Deploy in IoT Environment

6. Журило О., Ляшенко О. Архітектура та системи безпеки IoT на основі туманних обчислень

7. Соловей Б. В. Виявлення DoS/DDoS атак в IoT за допомогою машинного навчання

8. Журило О., Ляшенко О., Аветісова К. Огляд рішень з апаратної безпеки кінцевих пристроїв туманних обчислень у Інтернеті речей

9. Hrițcan D.-F. та ін. A Performance Analysis of VPN Technologies Used in an IoT Environment

10. Kjorveziroski V. та ін. Full-mesh VPN performance evaluation for a secure edge-cloud continuum

11. Капустін М. Методика вибору протоколу VPN при організації віддаленого доступу в корпоративних мережах

12. Козюк Ю. Ю., Салієва О. В. Порівняльний аналіз сучасних технологій тунелювання в комп’ютерних мережах

13. Zou Y. Network performance and key management of VPN tunnels between autonomous vehicles

14. Кулібабчук І. П., Азарова А. О. Порівняння можливостей та продуктивності VPN-протоколів

15. Yedla B. K. Performance evaluation of VPN solutions in multi-region kubernetes cluster

16. Hrițcan D.-F. та ін. A Performance Analysis of VPN Technologies Used in an IoT Environment

17. Jumakhan H., Mirzaeinia A. Wireguard: An Efficient Solution for Securing IoT Device Connectivity

18. Yan Jiang та ін. Design and Implementation of IPsec VPN IoT Gateway System in National Secret Algorithm

19. Я. С. Олійник та ін. Методи захисту інформації в технологіях IoT

20. Shailesh Mota. Secure Certificate Management and Device Enrollment at IoT Scale

21. Є. Риндич, А. Боровик, О. Боровик. Дослідження технологій тунелювання в сучасних комп'ютерних мережах

22. David Isaac Wolinsky та ін. Virtual Private Overlays: Secure Group Communication in NAT-Constrained Environments

23. Bruno Marcel Duarte Coscia. Evaluation of Network-Layer Security Technologies for Cloud Platforms

24. Shailesh Mota. Secure Certificate Management and Device Enrollment at IoT Scale

25. П. П. Вовчановський. Забезпечення конфіденційного обміну сертифікатами для пристроїв Інтернету речей з використанням IOTA

26. Lars Eggert. Towards Securing the Internet of Things with QUIC

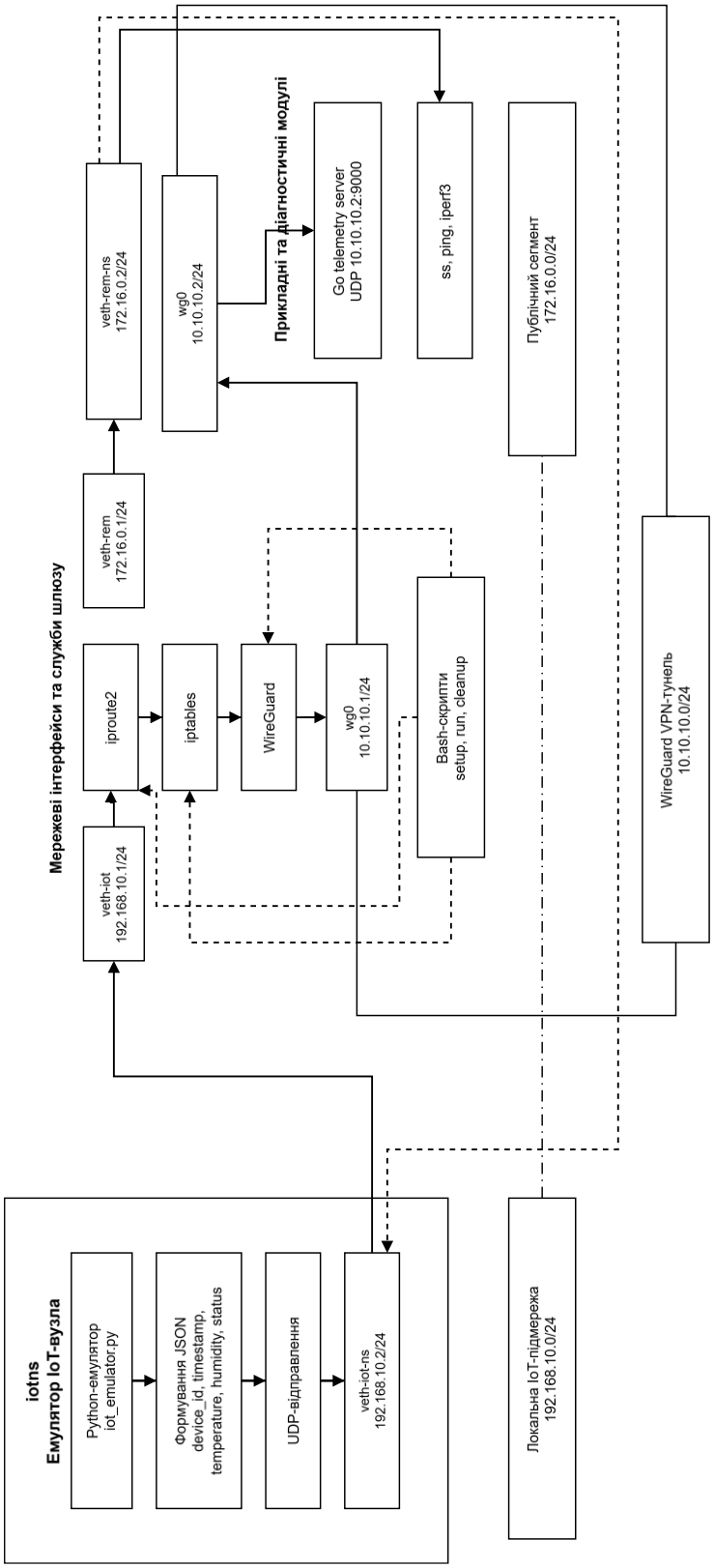
27. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

28. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання

кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

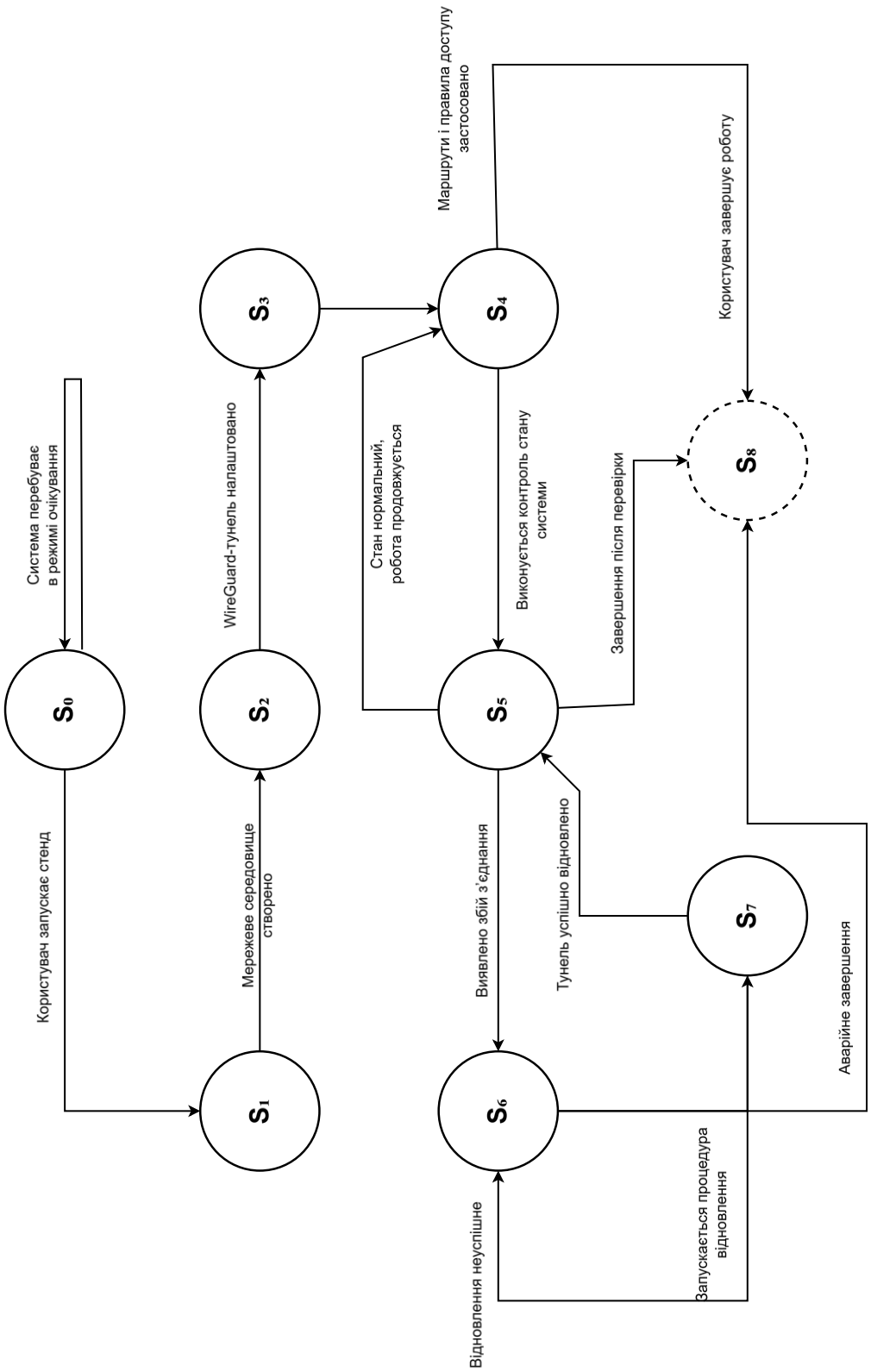
Додаток А

Структурна схема системи



Додаток Б

Автомат станів функціонування VPN-шлюзу



Додаток В

Код програмних модулів

Скрипт розгортання експериментального стенда setup_lab.sh

```
#!/usr/bin/env bash
set -e
export PATH=$PATH:/usr/sbin:/sbin
BASE_DIR="/home/vpn_gateway_project"
WG_DIR="$BASE_DIR/configs/wireguard"
echo "[INFO] Creating namespaces..."
ip netns add iotns
ip netns add remns
echo "[INFO] Creating veth pairs..."
ip link add veth-iot type veth peer name veth-iot-ns
ip link add veth-rem type veth peer name veth-rem-ns
ip link set veth-iot-ns netns iotns
ip link set veth-rem-ns netns remns
echo "[INFO] Assigning addresses in root namespace..."
ip addr add 192.168.10.1/24 dev veth-iot
ip addr add 172.16.0.1/24 dev veth-rem
ip link set veth-iot up
ip link set veth-rem up
echo "[INFO] Assigning addresses in iotns..."
ip netns exec iotns ip addr add 192.168.10.2/24 dev veth-iot-ns
ip netns exec iotns ip link set lo up
ip netns exec iotns ip link set veth-iot-ns up
ip netns exec iotns ip route add default via 192.168.10.1
echo "[INFO] Assigning addresses in remns..."
ip netns exec remns ip addr add 172.16.0.2/24 dev veth-rem-ns
ip netns exec remns ip link set lo up
ip netns exec remns ip link set veth-rem-ns up
ip netns exec remns ip route add default via 172.16.0.1
echo "[INFO] Enabling IP forwarding..."
sysctl -w net.ipv4.ip_forward=1 >/dev/null
echo "[INFO] Creating WireGuard on gateway..."
ip link add wg0 type wireguard
ip addr add 10.10.10.1/24 dev wg0
wg set wg0 \
  listen-port 51820 \
  private-key "$WG_DIR/gateway_private.key" \
  peer "$(cat "$WG_DIR/remote_public.key")" \
  allowed-ips 10.10.10.2/32
ip link set wg0 up
echo "[INFO] Creating WireGuard in remns..."
ip netns exec remns ip link add wg0 type wireguard
ip netns exec remns ip addr add 10.10.10.2/24 dev wg0
ip netns exec remns wg set wg0 \
  private-key "$WG_DIR/remote_private.key" \
  peer "$(cat "$WG_DIR/gateway_public.key")" \
  endpoint 172.16.0.1:51820 \
  allowed-ips 10.10.10.1/32,192.168.10.0/24 \
  persistent-keepalive 25
ip netns exec remns ip link set wg0 up
ip netns exec remns ip route add 192.168.10.0/24 dev wg0
echo "[INFO] Applying firewall rules..."
iptables -I FORWARD 1 -i wg0 -o veth-iot -j ACCEPT
iptables -I FORWARD 2 -i veth-iot -o wg0 -j ACCEPT
iptables -I FORWARD 3 -i veth-rem -o veth-iot -j DROP
echo "[INFO] Lab setup completed."
```

Скрипт очищення середовища cleanup_lab.sh

Лістинг В.2 – Скрипт очищення середовища cleanup_lab.sh

```
#!/usr/bin/env bash
set -e
export PATH=$PATH:/usr/sbin:/sbin
echo "[INFO] Removing iptables FORWARD rules if they exist..."
iptables -D FORWARD -i wg0 -o veth-iot -j ACCEPT 2>/dev/null || true
iptables -D FORWARD -i veth-iot -o wg0 -j ACCEPT 2>/dev/null || true
iptables -D FORWARD -i veth-rem -o veth-iot -j DROP 2>/dev/null || true
echo "[INFO] Removing wg0 from root namespace..."
ip link del wg0 2>/dev/null || true
echo "[INFO] Removing wg0 from remns..."
ip netns exec remns ip link del wg0 2>/dev/null || true
```

```
  echo "[INFO] Removing veth interfaces
from root namespace..."
  ip link del veth-iot 2>/dev/null ||
true
  ip link del veth-rem 2>/dev/null ||
true
  echo "[INFO] Removing namespaces..."
  ip netns del iotns 2>/dev/null ||
true
  ip netns del remns 2>/dev/null ||
true
  echo "[INFO] Cleanup completed."
```

Скрипт запуску сервера телеметрії run_server.sh

```
  #!/usr/bin/env bash
  set -e
  BASE_DIR="/home/vpn_gateway_project"
  ip netns exec remns
"$BASE_DIR/cmd/telemetry_server/teleme
try_server"
```

Скрипт запуску емулятора IoT-вузла run_emulator.sh

```
  #!/usr/bin/env bash
  set -e
  BASE_DIR="/home/vpn_gateway_project"
  ip netns exec iotns python3
"$BASE_DIR/scripts/iot_emulator.py"
```

Код сервера телеметрії мовою Go main.go

```
  package main
  import (
      "fmt"
      "net"
      "os"
      "time"
  )
  func main() {
      addr := "10.10.10.2:9000"
      udpAddr,          err          :=
net.ResolveUDPAddr("udp", addr)
      if err != nil {
          fmt.Println("resolve
error:", err)
          os.Exit(1)
      }
      conn,           err           :=
net.ListenUDP("udp", udpAddr)
      if err != nil {
          fmt.Println("listen error:",
err)
          os.Exit(1)
      }
      defer conn.Close()
      fmt.Println("[START]   Telemetry
server listening on", addr)
      buf := make([]byte, 4096)
      for {
          n,   remoteAddr,   err   :=
conn.ReadFromUDP(buf)
          if err != nil {
              fmt.Println("read
error:", err)
              continue
          }
          now                        :=
time.Now().Format("2006-01-02
15:04:05")
          msg := string(buf[:n])
          fmt.Printf("[%s]   from   %s:
%s\n", now, remoteAddr.String(), msg)
      }
  }
```

Код емулятора IoT-вузла мовою Python iot_emulator.py

```
  #!/usr/bin/env python3
  import json
  import random
  import socket
  import time
  from datetime import datetime
  SERVER_IP = "10.10.10.2"
  SERVER_PORT = 9000
  DEVICE_ID = "iot-node-01"
  sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
  print(f"[START] IoT emulator sending
telemetry                          to
{SERVER_IP}:{SERVER_PORT}")
  while True:
      payload = {
          "device_id": DEVICE_ID,
          "timestamp":
datetime.now().isoformat(timespec="sec
onds"),
          "temperature":
round(random.uniform(20.0, 28.0), 2),
          "humidity":
round(random.uniform(40.0, 65.0), 2),
          "status": "ok"
      }
      message   =   json.dumps(payload,
ensure_ascii=False)
      sock.sendto(message.encode("utf-
8"), (SERVER_IP, SERVER_PORT))
      print("[SENT]", message)

      time.sleep(3)
```

Додаток Г

Апробація отриманих результатів

ДОВІДКА

ПРО ПРИЙНЯТТЯ НАУКОВОЇ РОБОТИ ДО ПУБЛІКАЦІЇ

11.06.2026

Шановний(і) авторе(и):
Сенькович Костянтин Юрійович ,

Організаційний комітет з радістю повідомляє, що наукову роботу «VPN-ШЛЮЗ ДЛЯ БЕЗПЕЧНОГО ПІДКЛЮЧЕННЯ IOT-ВУЗЛІВ» прийнято до публікації в збірнику за матеріалами XI Міжнародної наукової конференції «Традиційні та інноваційні підходи до наукових досліджень» (19.06.2026; м. Луцьк, Україна).

Опублікована робота буде доступна з 19.06.2026 за посиланням:
https://archives.mcnd.org.ua/index.php/conference-proceeding/issue/view/19.06.2026

. . . . . .

Електронні сертифікати учасників конференції будуть доступні з 19 червня.

Конференцію зареєстровано в базі даних науково-технічних заходів УкрІНТЕІ (Посвідчення № 176 від 26.01.2026). Матеріали конференції знаходитимуться в відкритому доступі (Open Access) на умовах ліцензії CC BY-SA 4.0 International.

З повагою,

Віце-президент МЦНД
Голова оргкомітету конференції
НАСТАСІЯ РАБЕЙ

МЦНД
МІЖНАРОДНИЙ ЦЕНТР НАУКОВИХ ДОСЛІДЖЕНЬ

VPN-ШЛЮЗ ДЛЯ БЕЗПЕЧНОГО ПІДКЛЮЧЕННЯ IOT-ВУЗЛІВ

Сенькович Костянтин Юрійович
Студент спеціальності 122 – Комп'ютерні науки
Західноукраїнський національний університет, Україна
Науковий керівник: Осолінський Олександр Романович
ORCID ID: 0000-0002-0136-395X
к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет
Україна

Вступ

Інтернет речей активно використовується у побутових, промислових і сервісних системах, де сенсори, контролери та виконавчі пристрої передають дані до віддалених серверів або отримують команди керування. Значна частина таких вузлів працює у приватних мережах, за домашніми маршрутизаторами, мобільними каналами або в окремих локальних сегментах. Через це пряме підключення до них через публічну мережу ускладнюється, а деколи створює додаткові ризики перехоплення трафіку або підміну даних [1].

Для зменшення цих ризиків можна використовувати проміжний шлюз, який ізолює локальну IoT-мережу від відкритого середовища та організовує захищений канал до віддаленого адміністратора або сервера. Такий вузол повинен створювати тунель, виконувати маршрутизацію між підмережами, фільтрувати небажані з'єднання, контролювати доступ і фіксувати події роботи системи. Для компактних IoT-сценаріїв важливо, щоб рішення мало невеликі накладні витрати і могло працювати на малопотужній edge-платформі [2].

Архітектура та програмна реалізація VPN-шлюзу

Запропонована система побудована за принципом проміжного довіреного вузла. Локальна IoT-підмережа не відкривається напряму в публічний сегмент, а

весь доступ до неї здійснюється через VPN-шлюз. В ролі апаратної основи в роботі розглянуто Raspberry Pi. Це дозволило створити ізольований стенд без додаткового обладнання і перевірити ключові побудову тунелю, передавання телеметрії, фільтрацію трафіку та відновлення після розриву з'єднання.

Також було створено три логічні частини: а) простір імен iotns, який виконував роль IoT-вузла; б) основне середовище root namespace, що працювало як VPN-шлюз; в) простір імен remns, який імітував віддалений сервер або робоче місце адміністратора. Локальна підмережа отримала адресу 192.168.10.0/24, публічний сегмент – 172.16.0.0/24, а захищений WireGuard-тунель – 10.10.10.0/24. Такий поділ дав змогу чітко відокремити локальний трафік від віддаленого доступу і перевірити, чи проходять пакети саме через захищений канал.

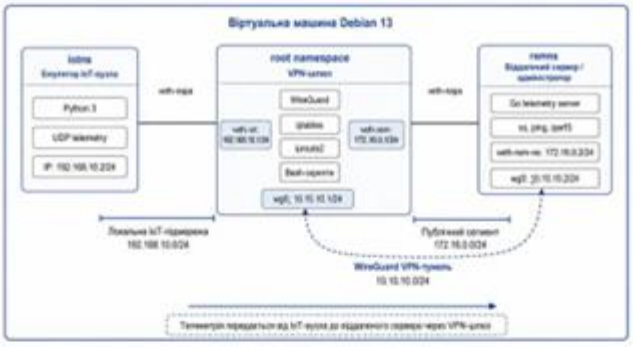


Рис. 1. **Структура експериментального стенда VPN-шлюзу у віртуальному середовищі**

Основною технологією тунелювання обрано WireGuard, бо він має компактну конфігурацію, сучасні криптографічні механізми і може працювати в сценаріях з обмеженими ресурсами [3], [4]. На шлюзі створювався інтерфейс wg0 з адресою 10.10.10.1/24, а на віддаленому вузлі – інтерфейс wg0 з адресою 10.10.10.2/24. Для сторін тунелю були згенеровані окремі приватні та публічні

ключі, а у конфігурації задавалися дозволені адреси, порт прослуховування і параметр підтримання з'єднання.

Маршрутизацію реалізовано засобами iproute2. На шлюзі було увімкнено пересилання IPv4-пакетів, після чого трафік між локальною IoT-підмережею і віддаленим сервером міг проходити через інтерфейс wg0. Для контролю доступу застосовано iptables. Правила міжмережевого екрана дозволяли трафік між VPN-інтерфейсом і локальним інтерфейсом IoT-сегмента, але блокували прямий доступ з публічного сегмента до локального вузла. Завдяки цьому віддалена сторона могла звернутися до IoT-вузла лише через захищений тунель.

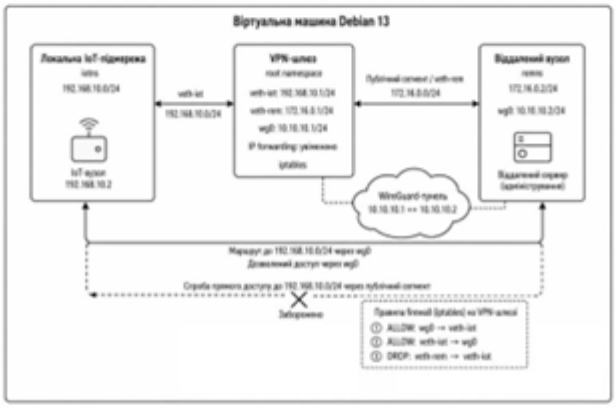


Рис. 2. **Схема маршрутизації та фільтрації трафіку в експериментальному стенді**

Для відтворення прикладного IoT-сценарію реалізовано два програмні модулі. Сервер приймання телеметрії написано мовою Go. Він прослуховував UDP-порт 9000 на віддаленому вузлі та відображав отримані повідомлення з часовою міткою і адресою джерела. Емулятор IoT-вузла створено мовою Python. Він періодично формував JSON-повідомлення з ідентифікатором пристрою, часом, температурою, вологістю та станом і надсилав його на віддалений сервер через VPN-шлюз. Для запуску стенда, сервера, емулятора та очищення середовища підготовлено Bash-скрипти.

Запропонована організація дає змогу розділити службову логіку на рівень підготовки мережевого середовища, встановлення тунелю, маршрутизацію, фільтрацію пакетів, передавання телеметрії та фіксацію стану. Такий підхід є зручним для подальшого перенесення на Raspberry Pi або інший компактний edge-пристрій.

Тестування функціональності

Тестування прототипу проводилося за кількома сценаріями. Спочатку було перевірено передавання телеметрії через WireGuard-тунель. Після запуску Go-сервера в просторі remns і Python-емулятора в просторі iotns повідомлення успішно надходили на адресу 10.10.10.2:9000. На стороні сервера фіксувалася адреса джерела 192.168.10.2, що підтвердило правильну маршрутизацію даних від локального IoT-вузла до віддаленого сервера через шлюз.

Другий тест стосувався перевірки захисту від прямого доступу. Після видалення маршруту до мережі 192.168.10.0/24 через wg0 спроба звернення до IoT-вузла завершилася повною втратою пакетів. Після повернення маршруту доступ було відновлено. Це підтвердило, що правило міжмережевого екрана блокує небажане підключення з публічного сегмента і дозволяє взаємодію лише через VPN.

Окремо змодельовано аварійну ситуацію: інтерфейс wg0 на віддаленому вузлі переводився у стан DOWN. Після цього нові телеметричні повідомлення перестали надходити на сервер, а після повторного активування інтерфейсу з'єднання відновилося. Наявність нового handshake у WireGuard і продовження приймання телеметрії підтвердили працездатність механізму відновлення.

Висновки. Було запропоновано та реалізовано експериментальний прототип VPN-шлюзу для безпечного підключення IoT-вузлів. Система поєднує WireGuard-тунелювання, маршрутизацію між підмережами, фільтрацію трафіку засобами iptables, UDP-сервер телеметрії мовою Go, Python-емулятор IoT-вузла та Bash-скрипти для автоматизації стенда.

Результати тестування показали, що прототип забезпечує передавання телеметрії через захищений тунель, блокує прямий доступ до локальної IoT-підмережі без VPN-маршруту, відновлює роботу після розриву з'єднання та має достатні характеристики для лабораторного IoT-сценарію.

Список використаних джерел:

1. Fazeldehkordi E., Grønli T.-M. A Survey of Security Architectures for Edge Computing-Based IoT.
2. Gentile, A. F., Macrì, D., De Rango, F., Tropea, M., & Greco, E. (2022). A VPN performances analysis of constrained hardware open-source infrastructure deploy in IoT environment. *Future Internet*, 14(9), 264.
3. HRIȚCAN, Daniel-Florin, et al. A Performance Analysis of VPN Technologies Used in an IoT Environment. Advances in Electrical & Computer Engineering, 2025, 25.2.
4. JUMAKHAN, Haseebullah; MIRZAEINIA, Amir. Wireguard: An Efficient Solution for Securing IoT Device Connectivity. *arXiv preprint arXiv*:2402.02093, 2024.
5. Kjorveziroski, V., Bernad, C., Gilly, K., & Filiposka, S. (2024). Full-mesh VPN performance evaluation for a secure edge-cloud continuum. *Software: Practice and Experience*, 54(8), 1543-1564.
6. YEDLA, Bharani Kumar. Performance evaluation of VPN solutions in multi-region kubernetes cluster. 2023.