Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Кібербезпеки
(повна назва)

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження побудови технології VPN у хмарі
(тема)

Виконав:

Здобувач 2 року навчання,

групи ІМІм-24-2

Ілля ТАРАНЕНКО
(власне ім'я, прізвище)

Спеціальність
172 Електронні комунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Освітня програма
Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник доц. Станіслав ІВАНЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ Микола МОСКАЛЕЦЬ
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей заборонених до відкритого публікування.

Здобувач _____/Ілля ТАРАНЕНКО/
(підпис)

Керівник _____/Станіслав ІВАНЕНКО/
(підпис)

Харківський національний університет радіоелектроніки

Факультет Кібербезпеки

Кафедра Інформаційно-мережна інженерія

Рівень вищої освіти другий (магістерський)

Спеціальність 172 Електронні комунікації та радіотехніка

Тип програми освітньо-наукова

Освітня програма Інформаційно-мережна інженерія

ЗАТВЕРДЖУЮ

Зав. кафедри _____
(підпис)

«___»_____20 __ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Тараненку Іллі Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження побудови технології VPN у хмарі

Затверджена наказом по університету від «23» березня 20 26 р. № 232 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «28» травня 20 26 р.

3. Вихідні дані до роботи Дослідити методи побудови технології VPN у хмарі

4. Перелік питань, що потрібно опрацювати в роботі

Вступ

1 АНАЛІЗ ПІДХОДІВ ДО ПОБУДОВИ ТА РОЗГОРТАННЯ VPN

2 АНАЛІЗ VPN ПРОТОКОЛІВ

3 РОЗГОРТАННЯ ТА КОНФІГУРАЦІЯ VPN-ІНФРАСТРУКТУРИ

4 МЕТОДИКА ЕКСПЕРИМЕНТАЛЬНОГО ТЕСТУВАННЯ

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри) _____

Слайди у форматі Power Point (назва та мета роботи, Концепція VPN, Архітектурні моделі VPN-з'єднань, Порівняння моделей розгортання VPN, IKEv2/IPsec, WireGuard, OpenVPN, Методика експерименту, Отримані результати, Результати оптимізації, Висновки).

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Пошук і аналіз літератури з теми кваліфікаційної роботи	03.04.2026	виконано
2	Виконання розділу 1	15.04.2026	виконано
3	Виконання розділу 2	25.04.2026	виконано
4	Виконання розділу 3	05.05.2026	виконано
5	Виконання розділу 4	05.05.2026	виконано
	Виконання розділу 5	15.05.2026	виконано
6	Підготовка доповіді	20.05.2026	виконано

Дата видачі завдання 23 березня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Станіслав ІВАНЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 102 с., 23 рис., 8 табл., 6 додатків, 27 джерел.

Об’єкт дослідження – технологія побудови VPN у хмарному середовищі.

Предмет дослідження – VPN-рішення IKEv2/IPsec, WireGuard і OpenVPN у self-hosted хмарній інфраструктурі на базі VPS.

Мета роботи – дослідити побудову VPN у хмарному середовищі та оцінити практичну придатність розглянутих VPN-протоколів для організації захищеного віддаленого доступу.

У роботі проаналізовано моделі розгортання VPN, особливості протоколів IKEv2/IPsec, WireGuard і OpenVPN, виконано їх конфігурацію на VPS та проведено експериментальне тестування. Оцінювання здійснено за показниками пропускної здатності, джиттера та втрат пакетів.

У результаті дослідження визначено переваги, обмеження та доцільність використання розглянутих VPN-рішень у self-hosted хмарній інфраструктурі.

IKEV2/IPSEC, WIREGUARD, OPENVPN, VPN, SELF-HOSTED VPN

ABSTRACT

Explanatory slip: 102 p., 23 fig., 8 tab., 6 app., 27 sources.

The object of the research is the technology of VPN deployment in a cloud environment.

The subject of the research is VPN solutions IKEv2/IPsec, WireGuard, and OpenVPN in a self-hosted cloud infrastructure based on a VPS.

The purpose of the work is to study VPN deployment in a cloud environment and to evaluate the practical applicability of the considered VPN protocols for organizing secure remote access.

The work analyzes VPN deployment models, the features of IKEv2/IPsec, WireGuard, and OpenVPN protocols, performs their configuration on a VPS, and conducts experimental testing. The evaluation is carried out according to throughput, jitter, and packet loss indicators.

The results of the research identified the advantages, limitations, and feasibility of using the VPN solutions under consideration in a self-hosted cloud infrastructure.

IKEV2/IPSEC, WIREGUARD, OPENVPN, VPN, SELF-HOSTED VPN

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AEAD (Authenticated Encryption with Associated Data) – режим автентифікованого шифрування з асоційованими даними;

AWS (Amazon Web Services) – платформа хмарних сервісів Amazon;

BGP (Border Gateway Protocol) – протокол граничної маршрутизації;

CA (Certificate Authority) – центр сертифікації;

CPU (Central Processing Unit) – центральний процесор;

CRL (Certificate Revocation List) – список відкликаних сертифікатів;

DCO (Data Channel Offload) – механізм винесення обробки каналу даних OpenVPN у ядро операційної системи;

DNS (Domain Name System) – система доменних імен;

ESP (Encapsulating Security Payload) – протокол IPsec для захисту корисного навантаження IP-пакетів;

GCP (Google Cloud Platform) – платформа хмарних сервісів Google;

HA (High Availability) – висока доступність;

HMAC (Hash-based Message Authentication Code) – код автентифікації повідомлення на основі хеш-функції;

IaC (Infrastructure as Code) – підхід до керування інфраструктурою за допомогою програмного коду;

IKEv2 (Internet Key Exchange version 2) – протокол обміну ключами версії 2;

IPsec (Internet Protocol Security) – набір протоколів для захисту IP-трафіку;

MAC (Message Authentication Code) – код автентифікації повідомлення;

MTU (Maximum Transmission Unit) – максимальний розмір блоку даних, що може бути переданий через мережний інтерфейс;

NAT (Network Address Translation) – механізм трансляції мережних адрес;

OCSP (Online Certificate Status Protocol) – протокол перевірки статусу сертифіката;

OSI (Open Systems Interconnection) – еталонна модель взаємодії відкритих систем;

P2S (Point-to-Site) – модель VPN-підключення окремого клієнта до мережі;

SA (Security Association) – асоціація безпеки;

SAD (Security Association Database) – база асоціацій безпеки;

SPD (Security Policy Database) – база політик безпеки;

SPI (Security Parameters Index) – індекс параметрів безпеки;

SSH (Secure Shell) – протокол захищеного віддаленого доступу;

TLS (Transport Layer Security) – протокол захисту транспортного рівня;

UDP (User Datagram Protocol) – протокол користувацьких дейтаграм;

VPC (Virtual Private Cloud) – віртуальна приватна хмара;

VPN (Virtual Private Network) – віртуальна приватна мережа;

VPS (Virtual Private Server) – віртуальний приватний сервер.

ВСТУП

У сучасних умовах хмарна інфраструктура, віддалений доступ і розподілені мережні сервіси стали важливими складовими функціонування інформаційних систем. Передавання даних через публічні мережі супроводжується ризиками перехоплення трафіку, несанкціонованого доступу, аналізу метаданих і втручання в мережну взаємодію [1].

Одним із поширених засобів захисту мережної взаємодії є віртуальна приватна мережа. VPN дозволяє створити захищений тунель поверх недовіреної мережі, забезпечити автентифікацію сторін, шифрування трафіку, контроль цілісності та маршрутизацію даних через визначений VPN-шлюз. Такі властивості роблять VPN актуальним інструментом для організації віддаленого доступу, захисту службового трафіку та об’єднання мережних сегментів.

Практичне розгортання VPN у хмарному середовищі потребує вибору моделі розгортання, VPN-протоколу, параметрів маршрутизації, NAT, правил фаєрвола, внутрішньої адресації та способу автентифікації. Керовані хмарні VPN-рішення спрощують частину цих завдань, однак обмежують рівень контролю над інфраструктурою та створюють залежність від постачальника послуг. Тому актуальним є дослідження self-hosted підходу, за якого VPN-шлюз розгортається на базі VPS, а адміністратор самостійно керує операційною системою, VPN-програмним забезпеченням і мережною конфігурацією. Способи реалізації такого підходу можуть суттєво відрізнятися: від ручного налаштування безпосередньо в операційній системі до автоматизації за допомогою концепції Infrastructure as Code (IaC) чи контейнеризації (Docker). Важливим є не лише теоретичне порівняння VPN-протоколів, а й їх практичне розгортання, перевірка працездатності та експериментальне оцінювання характеристик трафіку в реальних умовах мережі Інтернет.

Актуальність роботи зумовлена потребою практичного аналізу побудови VPN у хмарному середовищі та вибору доцільного рішення для self-hosted VPN-

інфраструктури. Метою кваліфікаційної роботи є дослідження побудови технології VPN у хмарному середовищі на основі self-hosted моделі та оцінювання практичної придатності VPN-рішень IKEv2/IPsec, WireGuard і OpenVPN для організації захищеного віддаленого доступу.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

- проаналізувати концепцію VPN;
- розглянути архітектурні моделі VPN-з’єднань і моделі розгортання VPN-інфраструктури;
- проаналізувати особливості VPN-протоколів;
- виконати конфігурацію IKEv2/IPsec, WireGuard та OpenVPN;
- розробити методику експериментального тестування VPN-рішень;
- провести вимірювання пропускної здатності, джиттера та втрат пакетів;
- проаналізувати отримані результати та визначити доцільність використання кожного VPN-рішення.

Об’єктом дослідження є технологія побудови VPN у хмарному середовищі.

Предметом дослідження є моделі розгортання, конфігураційні особливості та експериментальні характеристики VPN-рішень IKEv2/IPsec, WireGuard і OpenVPN у self-hosted хмарній інфраструктурі на базі VPS.

Апробація результатів роботи здійснена шляхом підготовки публікації «Конфігурація приватного VPN-серверу», у якій розглянуто практичні аспекти налаштування приватного VPN-сервера, ризики перехоплення даних у недовірених мережах, та використання WireGuard як сучасного засобу побудови захищених каналів зв’язку [1].

1 АНАЛІЗ ПІДХОДІВ ДО ПОБУДОВИ ТА РОЗГОРТАННЯ VPN

1.1 Концепція та принципи функціонування VPN

Віртуальна приватна мережа, або VPN, є способом організації ізольованого та захищеного обміну даними поверх наявної мережної інфраструктури. Її основне призначення полягає в організації безпечного доступу до приватних ресурсів через потенційно небезпечне мережне середовище. Такий доступ може надаватися окремим користувачам, вузлам або цілим мережним сегментам. Найчастіше таким середовищем є Інтернет, однак VPN може використовуватися і в межах приватних або операторських мереж, якщо потрібно обмежити доступ до службового трафіку чи відокремити його від іншої мережної взаємодії [2].

Потреба у VPN виникає через те, що передавання даних через публічні мережі супроводжується ризиками перехоплення, модифікації трафіку, несанкціонованого доступу та розкриття службової інформації про взаємодію між вузлами. Замість використання фізично виділених каналів VPN дозволяє застосувати спільну транспортну інфраструктуру, але додає до неї механізми ізоляції, автентифікації та криптографічного захисту. Такий підхід зменшує залежність від окремих телекомунікаційних ліній і дає змогу організувати захищену взаємодію між віддаленими користувачами, серверами або приватними підмережами [3].

Базовим принципом роботи VPN є тунелювання. Його сутність полягає в тому, що початковий трафік не передається через зовнішню мережу безпосередньо, а інкапсулюється в зовнішній транспортний пакет: внутрішній пакет містить адреси та дані приватної взаємодії, а зовнішній заголовок використовується для доставлення через транспортну мережу. На стороні отримувача VPN-клієнт або шлюз виконує декапсуляцію, після чого відновлені пакети спрямовуються до вузла призначення. Маршрутизація в зовнішній мережі при цьому виконується переважно на основі зовнішнього заголовка [2,4].

Інкапсуляція створює технічну основу VPN-тунелю, але сама по собі не гарантує захисту даних. Вона забезпечує перенесення трафіку через іншу мережу та логічне відокремлення внутрішньої взаємодії від транспортного середовища. Для використання VPN у публічних або недовірених мережах тунелювання має доповнюватися криптографічними механізмами. До них належать шифрування, автентифікація сторін, контроль цілісності та захист від повторного передавання пакетів. Саме ці механізми перетворюють тунелювання з механізму передавання трафіку на захищений канал зв'язку [2].

Криптографічний захист у VPN виконує кілька функцій:

– шифрування приховує вміст переданих даних від сторонніх осіб;
– автентифікація підтверджує, що з'єднання встановлюється з дозволеним клієнтом, сервером або шлюзом;
– контроль цілісності забезпечує криптографічну перевірку того, що дані не були змінені під час передавання.

Для цього можуть використовуватися механізми на основі MAC/HMAC або режими автентифікованого шифрування, зокрема AEAD. Захист від повторного передавання пакетів не дозволяє зловмиснику повторно використати раніше перехоплені пакети як коректний елемент нового обміну [2].

Окрім захисту вмісту, VPN впливає на логіку маршрутизації трафіку. Після встановлення з'єднання на клієнті або шлюзі з'являються маршрути, які визначають пакети, що мають проходити через VPN-інтерфейс. Якщо через тунель передається лише трафік до визначених приватних підмереж або сервісів, використовується модель split tunneling (роздільного тунелювання) (рис. 1.1). Вона зменшує навантаження на VPN-сервер і дозволяє звичайному інтернет-трафіку проходити через локальний шлюз користувача. Якщо ж увесь трафік клієнта спрямовується через VPN, використовується модель full tunneling (повного тунелювання) (рис. 1.2). Цей режим забезпечує централізований контроль мережного трафіку, але збільшує вимоги до пропускної здатності, затримки, правил NAT, правил фаєрвола та моніторингу [2].

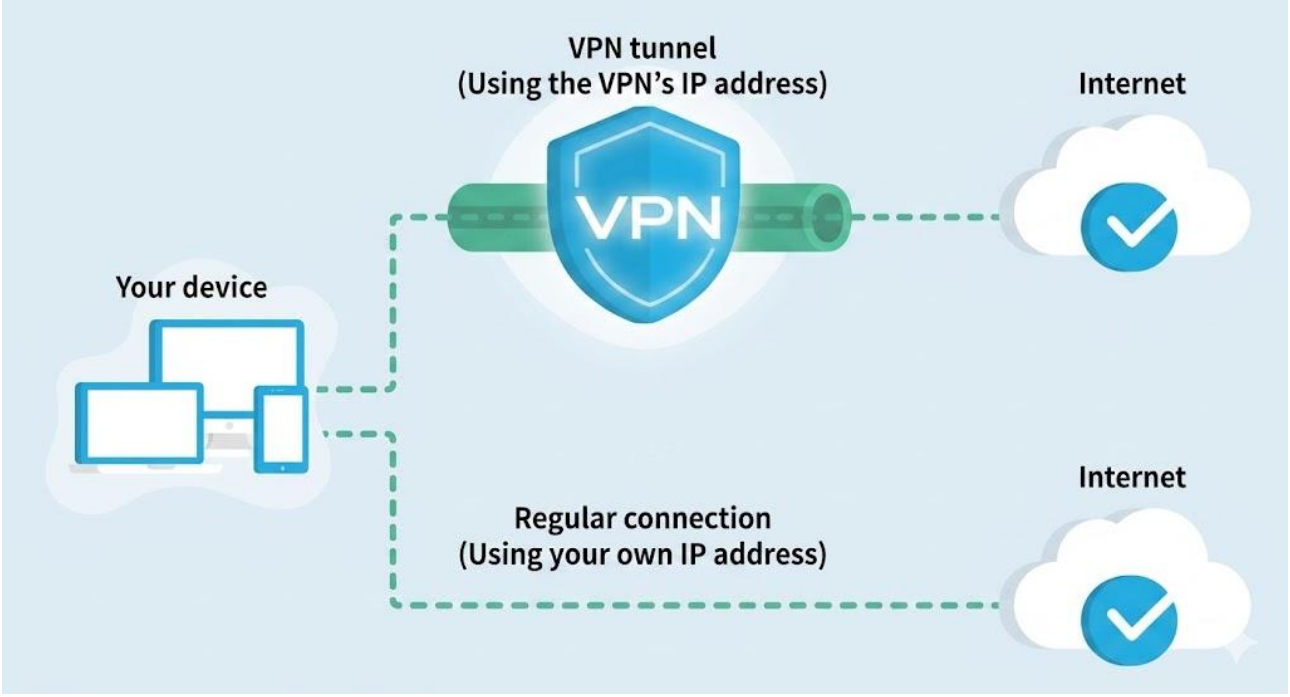

Рисунок 1.1 – Модель Split Tunneling

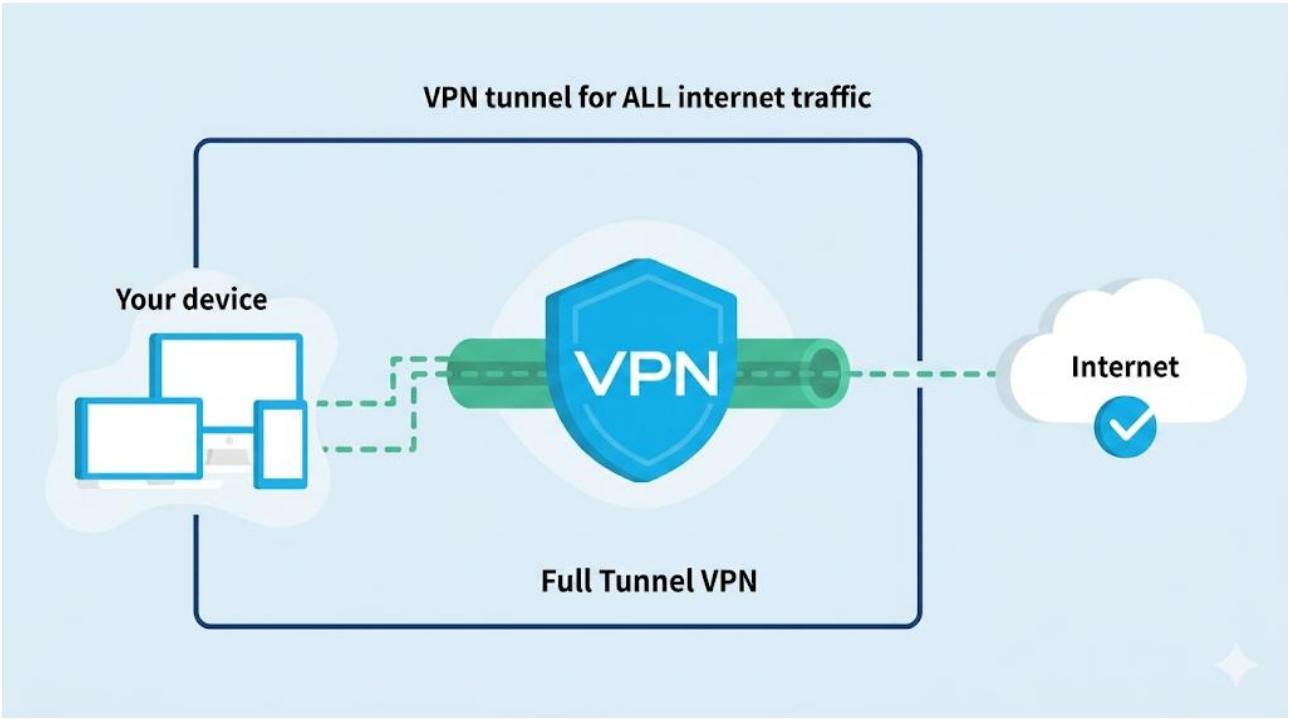

Рисунок 1.2 – Модель Full Tunneling

У межах VPN-з’єднання можуть визначатися:

– які користувачі мають право підключатися;
– які ресурси їм доступні;
– які маршрути додаються після встановлення з’єднання та які типи трафіку дозволено передавати.

Тому практичне розгортання VPN потребує узгодження криптографічних параметрів, адресації, маршрутизації, правил фаєрвола, логування та політик доступу. Некоректне налаштування хоча б одного з цих компонентів може призвести до надмірного доступу, витоку трафіку поза тунелем або недоступності потрібних сервісів [2].

Таким чином, VPN є базовим механізмом організації захищеної мережної взаємодії в умовах використання спільної або недовіреної транспортної інфраструктури. Його робота ґрунтується на поєднанні тунелювання, інкапсуляції, криптографічного захисту, автентифікації, контролю цілісності та маршрутизації.

У загальній архітектурі мережі VPN дають змогу надати віддалений або міжмережний доступ до приватних ресурсів без прямого відкриття цих ресурсів у публічному середовищі. VPN широко застосовується для організації захищеного віддаленого доступу, міжмережної взаємодії та контролю доступу до приватної інфраструктури.

1.2 Архітектурні моделі побудови VPN-з’єднань

Архітектурна модель VPN визначає, між якими об’єктами встановлюється захищене з’єднання та на яку частину мережного шляху поширюється VPN-захист. Залежно від цього VPN може використовуватися для підключення окремого користувача до приватної мережі, для об’єднання двох віддалених мережних сегментів або для захисту взаємодії між двома конкретними вузлами.

Серед базових моделей VPN зазвичай виділяють Remote Access VPN, Site-to-Site VPN та Host-to-Host VPN [2].

Remote Access VPN призначений для підключення окремого користувача або пристрою до приватної мережі організації. У такій моделі на стороні організації розміщується VPN-шлюз, а пристрій користувача виконує роль VPN-клієнта. Після автентифікації між клієнтом і шлюзом створюється захищене з'єднання, через яке користувач отримує доступ до внутрішніх ресурсів: серверів, служб адміністрування, файлових сховищ, внутрішніх вебсервісів або інших корпоративних систем. Така архітектура використовується тоді, коли користувач перебуває поза межами захищеної мережі, але повинен працювати з її ресурсами без прямого відкриття цих ресурсів у публічному середовищі, приклад такої моделі показано на рис. 1.3 [2].

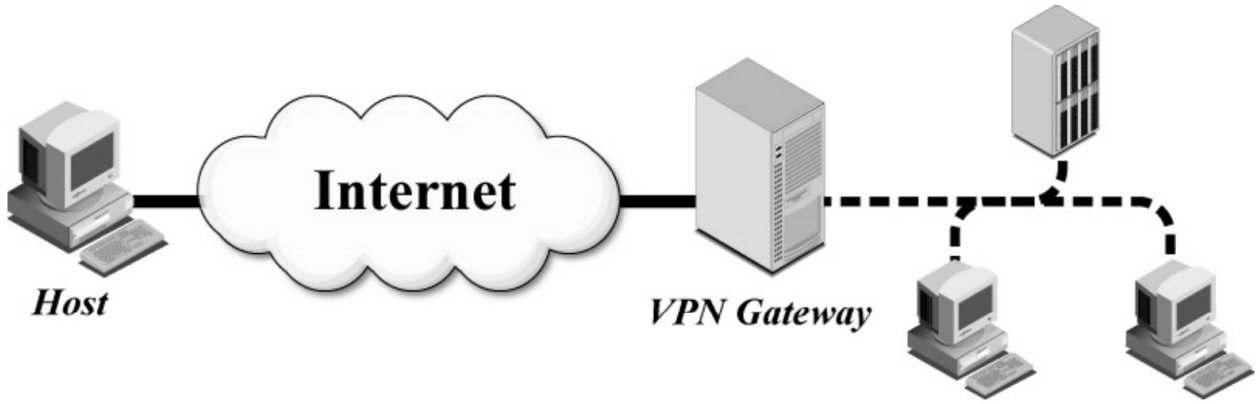

Рисунок 1.3 – Приклад архітектурної моделі Remote Access

У такій моделі VPN захищає насамперед ділянку між клієнтським пристроєм і VPN-шлюзом. Подальший рух трафіку всередині приватної мережі залежить від локальної маршрутизації, правил фаєрвола та політик доступу.

Під час проєктування такої архітектури важливо враховувати не лише сам факт встановлення тунелю, а й те, які маршрути отримує клієнт, які підмережі стають доступними та як виконується контроль користувацьких прав. Така модель забезпечує гнучкість, оскільки користувач може підключатися з різних мереж. Водночас адміністрування ускладнюється через потребу керувати обліковими даними, клієнтським програмним забезпеченням, параметрами

автентифікації та режимами маршрутизації трафіку.

Site-to-Site VPN використовується для захищеного з’єднання двох або більше мережних сегментів. У цій моделі VPN-тунель встановлюється між шлюзами, які розміщені на межі відповідних мереж. Самі кінцеві пристрої зазвичай не створюють VPN-з’єднання самостійно: їхній трафік автоматично спрямовується через локальний шлюз відповідно до таблиць маршрутизації.

Типовими прикладами є з’єднання головного офісу з філією, об’єднання двох дата-центрів або організація захищеного обміну між віддаленими приватними мережами. Приклад архітектурної моделі показано на рис. 1.4. У NIST така модель описується як gateway-to-gateway, тобто з’єднання між двома VPN-шлюзами [2].

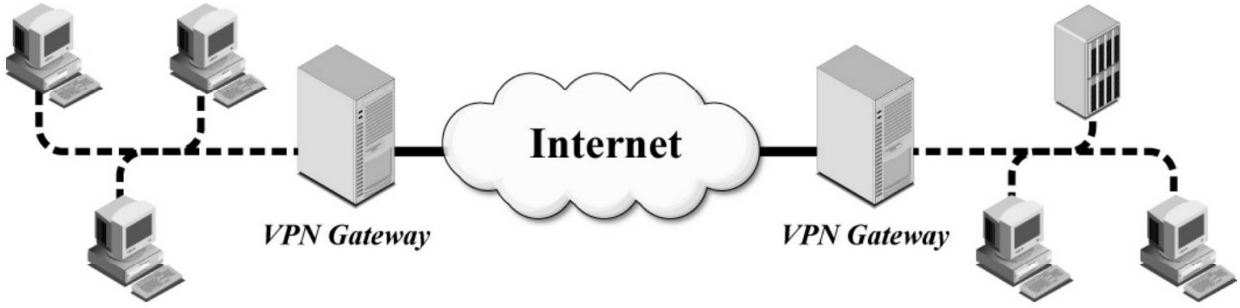

Рисунок 1.4 – Приклад архітектурної моделі Site-to-Site

Перевага Site-to-Site VPN полягає в прозорості для кінцевих вузлів. Якщо маршрутизація налаштована коректно, хости в одній мережі можуть взаємодіяти з хостами в іншій мережі без встановлення VPN-клієнта на кожному пристрої. Один захищений тунель між шлюзами може обслуговувати трафік багатьох вузлів або цілих підмереж. Водночас така архітектура захищає лише ту частину шляху, яка проходить між VPN-шлюзами. Ділянки від хоста до локального шлюзу та від віддаленого шлюзу до цільового хоста мають контролюватися засобами локальної інфраструктури. Через це важливими елементами проєктування є розміщення шлюзів, правила маршрутизації, фільтрація трафіку,

резервування та контроль точок входу до приватної мережі.

Host-to-Host VPN передбачає встановлення захищеного з'єднання безпосередньо між двома окремими вузлами. На відміну від попередніх архітектур, Host-to-Host VPN використовується переважно для спеціалізованих сценаріїв захисту взаємодії між окремими системами. Така модель зазвичай використовується для захисту критичних службових з'єднань, адміністрування окремих серверів або організації ізольованої взаємодії між визначеними довіреними хостами, приклад архітектурної моделі показано на рис. 1.5. У цьому випадку сервер може приймати з'єднання не з усієї мережі, а лише від хоста, який пройшов VPN-автентифікацію [2].

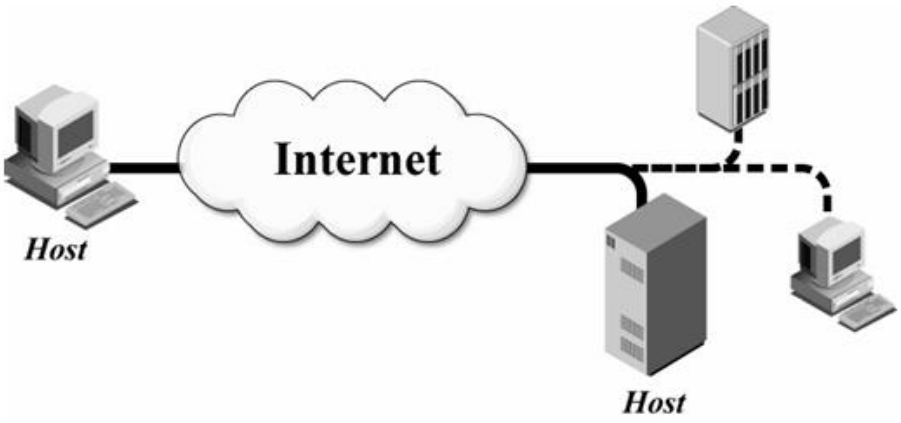

Рисунок 1.5 – Приклад архітектурної моделі Host-to-Host

Сильною стороною Host-to-Host VPN є наскрізний захист, тобто захист трафіку безпосередньо між двома кінцевими вузлами. Це зменшує ризик перехоплення або модифікації даних у проміжній мережі та дозволяє приховати службові порти від неавторизованих підключень. Водночас така модель складніше масштабується. Якщо кількість вузлів зростає, для кожного з них потрібно керувати параметрами підключення, ключами або сертифікатами, правилами доступу та оновленнями програмного забезпечення. Крім того, наскрізний захист між хостами може ускладнювати перевірку трафіку проміжними засобами безпеки, зокрема системами виявлення вторгнень або

мережними фільтрами.

Вибір архітектурної моделі VPN визначає межі захисту, спосіб маршрутизації, складність адміністрування та характер доступу до приватних ресурсів. Remote Access VPN орієнтований на підключення окремих користувачів або пристроїв до приватної мережі. Site-to-Site VPN забезпечує захищену взаємодію між мережними сегментами через VPN-шлюзи. Host-to-Host VPN застосовується тоді, коли потрібно захистити обмін між двома конкретними системами.

1.3 Порівняльний аналіз моделей розгортання VPN

Архітектурна модель VPN визначає характер взаємодії між вузлами та межі захисту трафіку. Проте практична реалізація VPN також залежить від того, де розміщується VPN-шлюз, хто адмініструє інфраструктуру та які ресурси використовуються для підтримки тунелю.

Порівняння моделей розгортання VPN доцільно виконувати за критеріями керування конфігурацією, масштабованості, складності адміністрування, вартості експлуатації, розподілу відповідальності за безпеку, залежності від постачальника та придатності до автоматизації. За цими ознаками можна виділити три основні моделі: On-premises VPN, Cloud-managed VPN та Self-hosted cloud VPN.

1.3.1 On-premises VPN

On-premises VPN передбачає розміщення VPN-шлюзу у власній локальній інфраструктурі організації, зокрема в офісній мережі, серверному приміщенні або приватному дата-центрі. У такій схемі VPN-шлюз функціонує в середовищі, яке адмініструється самою організацією або власником інфраструктури. Основні параметри роботи визначаються локально. Приклад моделі показано на рис. 1.6.

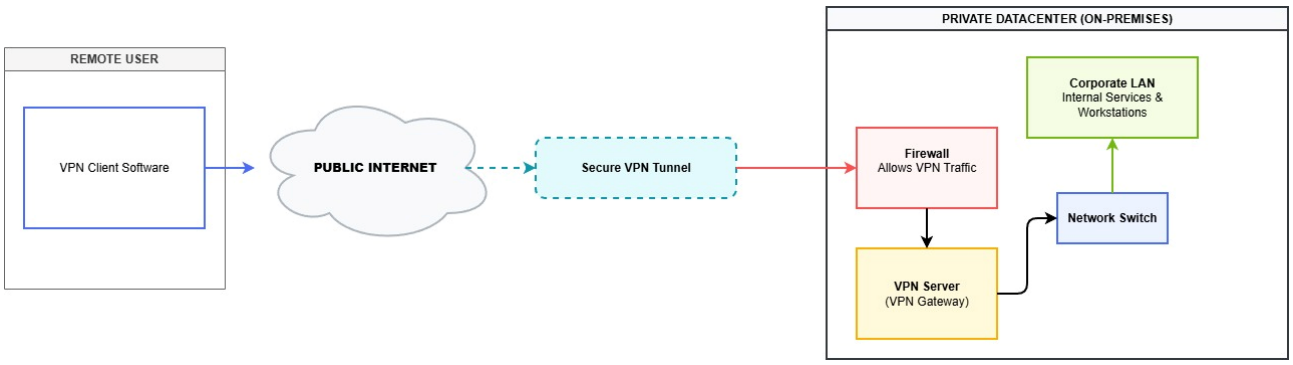

Рисунок 1.6 – Приклад моделі On-premises

On-premises VPN характерна для організацій, які вже мають власну мережну інфраструктуру та використовують VPN для контрольованого доступу до внутрішніх ресурсів: файлових сховищ, адміністративних систем, баз даних або внутрішніх вебсервісів. У такій архітектурі VPN-шлюз виконує роль захищеної точки входу до приватної мережі та дозволяє не відкривати внутрішні сервіси безпосередньо в Інтернет. Подібний підхід також може застосовуватися в малих приватних мережах і лабораторних середовищах [5].

З експлуатаційної точки зору On-premises VPN потребує самостійного супроводу інфраструктури. До зони відповідальності адміністратора входять обладнання, резервування каналів зв’язку, електроживлення, оновлення програмного забезпечення, моніторинг, захист периметра та відновлення після збоїв. Масштабування також пов’язане з додатковими витратами: збільшення кількості користувачів або VPN-тунелів може вимагати модернізації обладнання, зміни мережної топології або впровадження додаткових VPN-шлюзів.

On-premises VPN може бути доцільною моделлю в ситуаціях, коли власна інфраструктура вже існує, а вимоги до ізоляції, локального адміністрування та контролю мережного периметра є пріоритетними. Водночас ця модель менш зручна для сценаріїв, де потрібно швидко змінювати параметри розгортання або масштабувати ресурси без придбання й обслуговування додаткового обладнання.

1.3.2 Cloud-managed VPN

Модель Cloud-managed VPN передбачає використання VPN-шлюзу, який надається хмарним провайдером як керований сервіс. До таких рішень належать AWS Site-to-Site VPN, Google Cloud VPN та Azure VPN Gateway. У цій моделі провайдер забезпечує роботу VPN-шлюзу на своєму боці, його інтеграцію з віртуальними мережами та підтримку типових сценаріїв site-to-site або point-to-site підключення. Приклад моделі показано на рис. 1.7.

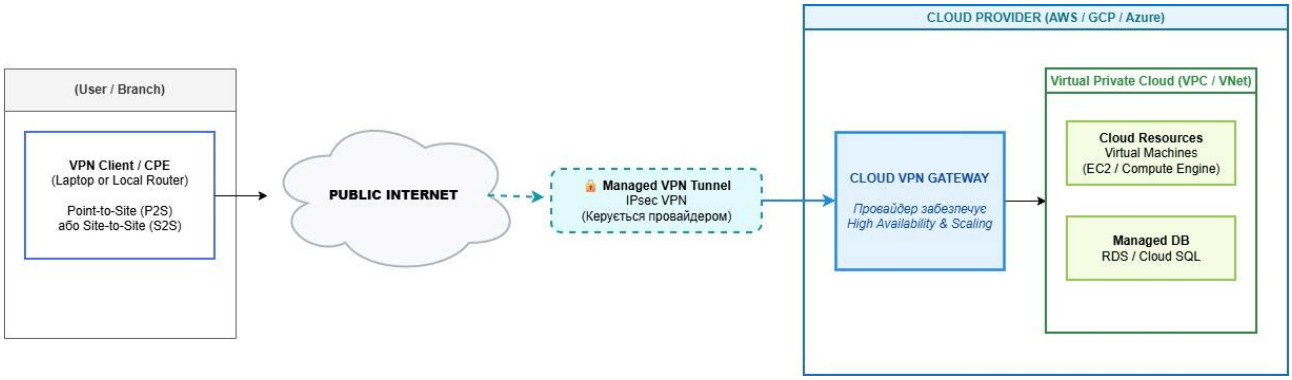

Рисунок 1.7 – Приклад моделі Cloud-managed VPN

AWS Site-to-Site VPN використовується для побудови з'єднання між віддаленою мережею та AWS VPC, Transit Gateway або Cloud WAN. Тарифікація базується на погодинній оплаті за активне VPN-з'єднання. Для регіону US East (Ohio) AWS вказує вартість 0,05 дол. США за годину для стандартного Site-to-Site VPN-з'єднання, що становить приблизно 36 дол. США за 30 днів безперервної роботи. Для 5 Gbps Site-to-Site VPN ціна становить 0,60 дол. США за годину, тобто приблизно 432 дол. США за 30 днів. Окремо може оплачуватися Transit Gateway attachment, наприклад 0,05 дол. США за годину в прикладі AWS [6].

Google Cloud VPN реалізує кероване IPsec з'єднання. У HA (High availability) VPN тарифікація базується на погодинній оплаті за кожен тунель, а також на оплаті IPsec-трафіку. Google Cloud вказує 0,05 дол. США за годину за кожен тунель. Для досягнення 99,99% доступності потрібно налаштувати два

тунелі, а при роботі з AWS peer gateway – чотири тунелі. Отже, мінімальна HA-конфігурація з двома тунелями коштує приблизно 72 дол. США за 30 днів безперервної роботи без урахування трафіку [7].

Azure VPN Gateway використовує більш фіксовану модель тарифікації через вибір SKU шлюзу. На відміну від AWS і Google Cloud, де витрати масштабуються разом із кількістю тунелів, додатковими мережними компонентами та обсягом трафіку, в Azure основна вартість визначається класом VPN Gateway. Наприклад, VpnGw1 коштує 138,70 дол. США на місяць і має пропускну здатність 650 Mbps, а VpnGw2 коштує 357,70 дол. США на місяць із пропускною здатністю 1 Gbps. Такий підхід спрощує прогнозування базових витрат, але перехід до продуктивніших SKU суттєво збільшує щомісячну вартість [8].

Різні провайдери використовують різні підходи до розрахунку вартості. В AWS базовою одиницею є активне VPN-з'єднання, у Google Cloud – VPN-тунель, а в Azure – клас VPN Gateway. Через це пряме порівняння цін потребує обережності: однакові суми не завжди відповідають однаковій архітектурі, пропускній здатності або рівню доступності.

Таблиця 1.1 – Базові витрати на Cloud-managed VPN

Провайдер / сервіс	Одиниця тарифікації	Базова ціна	Орієнтовно за 30 днів 24/7	Коментар
AWS Site-to-Site VPN 1,25 Gbps	VPN-з'єднання	0,05 $/год	≈ 36 $/міс	Стандартне VPN-з'єднання
Google Cloud HA VPN	1 тунель	0,05 $/год	≈ 36 $/міс	Один VPN-тунель
Google Cloud HA VPN	2 тунелі	0,10 $/год	≈ 72 $/міс	Мінімальна HA-конфігурація
Azure VPN Gateway VpnGw1	SKU шлюзу	–	138,70 $/міс	–

Окремо слід враховувати витрати, які не є власне оплатою VPN-сервісу, але впливають на кінцеву вартість рішення. До них належать вихідний трафік, IPsec-трафік, Transit Gateway attachment, зовнішні IP-адреси та інші мережні компоненти.

Таблиця 1.2 – Додаткові витрати Cloud-managed VPN

Провайдер	Додатковий компонент	Як впливає на вартість
AWS	Transit Gateway attachment	Оплачується окремо від Site-to-Site VPN; у прикладі AWS – 0,05 $/год
AWS	Data transfer out	Суттєво впливає на загальну вартість при великих обсягах вихідного трафіку
AWS	Accelerated VPN / Global Accelerator	Додає окремі мережні витрати у сценаріях прискореного VPN
Google Cloud	IPsec traffic	Тарифікується окремо від погодинної оплати тунелю
Google Cloud	Зовнішні IP-адреси	Можуть тарифікуватися, якщо IP-адреса призначена gateway, але не використовується тунелем
Azure	Data transfer	Передавання даних не входить у базову вартість VPN Gateway
Azure	Додаткові тунелі / P2S-підключення	Для частини SKU кількість підключень обмежена параметрами шлюзу

Важливою особливістю Cloud-managed VPN є модель розділеної відповідальності. Хмарний провайдер відповідає за роботу керованого VPN-сервісу, його оновлення, інфраструктурну доступність і підтримку платформи. Користувач, у свою чергу, відповідає за коректність мережної конфігурації: вибір параметрів тунелю, налаштування маршрутизації, правил доступу, BGP, security groups, фаєрвол політик і взаємодії з локальною мережею. Тобто провайдер зменшує обсяг робіт із підтримки самого сервісу, але не усуває відповідальність користувача за архітектуру підключення.

Cloud-managed VPN зменшує обсяг робіт із розгортання та супроводу VPN-шлюзу. Адміністратору не потрібно встановлювати VPN-програмне забезпечення, підтримувати системні служби або вручну інтегрувати шлюз із віртуальною мережею провайдера. Такі сервіси орієнтовані на експлуатаційні сценарії, де важливі інтеграція з хмарною мережею, підтримка BGP, централізоване керування та використання механізмів доступності, передбачених платформою.

З економічної точки зору Cloud-managed VPN є доцільним тоді, коли організації потрібна інтеграція з конкретною хмарною платформою, офіційна

підтримка провайдера, висока доступність і менший обсяг ручного адміністрування. Для невеликої кількості site-to-site підключень базова ціна AWS або Google Cloud може бути помірною, особливо якщо обсяг трафіку невеликий. Проте при збільшенні кількості тунелів, пропускної здатності або значного вихідного трафіку кінцева вартість зростає. У таких умовах вартість визначається вже не тільки самим VPN, а всією мережною архітектурою.

Порівняно з AWS і Google Cloud, Azure VPN Gateway має більш фіксовану модель витрат, оскільки користувач обирає конкретний SKU шлюзу. Це спрощує прогнозування бюджету, але навіть базові продуктивні рівні, наприклад VpnGw1 або VpnGw2, коштують дорожче за мінімальні конфігурації AWS або Google Cloud. Тому Azure VPN Gateway економічно виправданий переважно тоді, коли VPN є частиною ширшої Azure-інфраструктури або коли потрібні можливості конкретного SKU.

Cloud-managed VPN є доцільним у сценаріях, де важливі інтеграція з хмарною платформою, керована доступність і зменшення операційного навантаження. Для невеликих лабораторних або експериментальних середовищ, де потрібне безпосереднє керування конфігурацією та мінімізація постійних витрат, така модель може бути менш раціональною, особливо якщо порівнювати її з self-hosted cloud VPN на VPS.

1.3.3 Self-hosted cloud VPN

Модель Self-hosted cloud VPN передбачає розгортання VPN-шлюзу на віртуальній машині, орендованому VPS. У цьому випадку VPN-шлюз реалізується програмними засобами операційної системи.

На відміну від Cloud-managed VPN, self-hosted модель не надає готового хмарного VPN-шлюзу. Конфігурація VPN визначає параметри маршрутизації, фільтрації трафіку, адресації та автентифікації. Це означає, що VPN-шлюз налаштовується засобами операційної системи та вибраного програмного забезпечення. Приклад моделі показано на рис. 1.8.

У self-hosted cloud VPN модель розділеної відповідальності зміщується в бік адміністратора. VPS відповідає переважно за фізичну інфраструктуру,

віртуалізацію та доступність сервера, тоді як адміністратор відповідає за ОС, фаєрвол, ключі, журнали, політики доступу та оновлення [9].

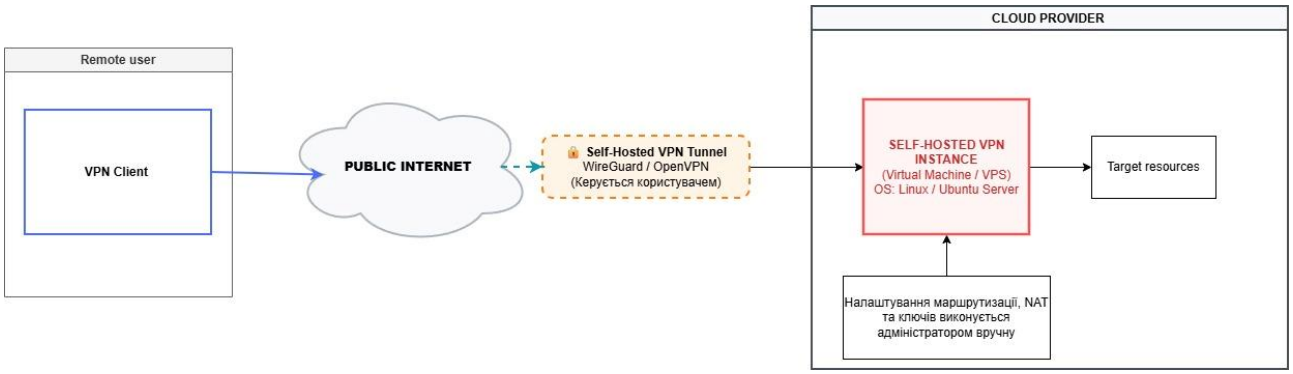

Рисунок 1.8 – Приклад моделі Self-hosted cloud VPN

Окремим технічним фактором self-hosted VPN є пропускна здатність. Вона залежить не лише від тарифу VPS або ліміту мережного порту, а й від продуктивності CPU під час шифрування трафіку. Отже, заявлена пропускна здатність VPS не гарантує аналогічної швидкості всередині VPN-тунелю. На фактичну продуктивність впливають протокол, алгоритми шифрування та MTU.

Self-hosted cloud VPN є придатною моделлю для автоматизації розгортання. Створення сервера, встановлення залежностей, генерація ключів, налаштування мережних правил і запуск VPN-сервісу можуть бути формалізовані за допомогою Infrastructure as Code, скриптів або контейнеризації. Це зменшує кількість ручних дій і спрощує відтворення однакової конфігурації в різних середовищах.

Економічна модель self-hosted cloud VPN відрізняється від managed-рішень. Основними статтями витрат є оренда VPS, публічна IP-адреса, дискові ресурси та мережний трафік відповідно до тарифів провайдера. Окремий спеціалізований VPN-сервіс або кожен VPN-тунель як окрема сутність зазвичай не тарифікуються. Один сервер може поєднувати функції VPN-шлюзу, контейнерного хоста та тестового сервера, якщо це відповідає вимогам до продуктивності та безпеки [9].

Недоліком self-hosted cloud VPN є більший обсяг адміністративних завдань. Потрібно оновлювати операційну систему, контролювати відкриті порти, обмежувати SSH-доступ, налаштовувати фаєрвол, стежити за журналами, створювати резервні копії конфігурацій і забезпечувати відновлення після збоїв. Також необхідно враховувати продуктивність сервера, навантаження на CPU під час шифрування, пропускну здатність мережного інтерфейсу та обмеження VPS.

1.3.4 Узагальнене порівняння моделей

Узагальнене порівняння моделей розгортання VPN наведено в таблиці 1.3.

Таблиця 1.3 – Порівняння моделей розгортання VPN

Критерій	On-premises VPN	Cloud-managed VPN	Self-hosted cloud VPN
Розміщення VPN-шлюзу	Власна локальна інфраструктура	Інфраструктура хмарного провайдера	VPS (інфраструктура хмарного провайдера)
Рівень контролю над конфігурацією	Високий	Обмежене можливостями сервісу	Високий
Складність адміністрування	Висока	Низька	Середня
Вибір VPN-протоколу	Вільний	Обмежений сервісом	Вільний
Масштабування	Через власне обладнання	Через механізми хмарного провайдера	Вертикальне або горизонтальне
Відмовостійкість	Реалізується самостійно	Частково забезпечується провайдером	Реалізується самостійно
Вартість	Обладнання, підтримка, канали зв’язку	VPN-шлюз, тунелі, обсяг трафіку, додаткові мережні сервіси	Сервер, IP-адреса, трафік
Відповідальність за безпеку ПЗ	Адміністратор або власник інфраструктури	Провайдер відповідає за VPN-сервіс; користувач за мережну конфігурацію	Адміністратор відповідає за ОС, VPN-ПЗ, патчинг і вразливості
Логування	Визначаються власником інфраструктури	Залежать від моделі керування провайдера	Визначаються адміністратором сервера
Пропускна здатність	Залежить від обладнання та каналу зв’язку	Визначається параметрами сервісу	Залежить від типу VPS, продуктивності CPU та мережних лімітів
Залежність від провайдера	Низька	Висока	Середня: можлива міграція на інший VPS
Придатність до IaC/DevOps	Середня, залежить від підтримки API	Висока (Terraform, CloudFormation)	Дуже висока (Terraform, Ansible)

Порівняння моделей розгортання VPN показує, що кожна з них має різний баланс між контролем, вартістю та складністю адміністрування. On-premises VPN забезпечує найбільшу незалежність від зовнішніх провайдерів і високий рівень керування інфраструктурою, однак потребує власного обладнання, супроводу мережного периметра та самостійного масштабування. Cloud-managed VPN зменшує операційне навантаження та спрощує інтеграцію з хмарною мережею, але водночас підвищує залежність від конкретного провайдера і має складнішу структуру витрат. Self-hosted cloud VPN займає проміжну позицію: вона не потребує власної фізичної інфраструктури, але зберігає можливість адміністрування VPN на рівні операційної системи, мережної конфігурації та засобів автоматизації.

Отже, вибір моделі розгортання VPN є компромісом між рівнем контролю, операційною складністю, вартістю та залежністю від постачальника. Для сценаріїв, де важлива керована доступність і інтеграція з конкретною хмарною платформою, доцільною є Cloud-managed VPN. Для середовищ, у яких потрібно досліджувати параметри тунелювання, маршрутизації, фаєрвол політик, контейнеризації та автоматизації розгортання, більш придатною є Self-hosted cloud VPN.

За результатами порівняння моделей розгортання для практичної частини роботи обрано self-hosted cloud VPN, оскільки ця модель забезпечує достатній рівень контролю над конфігурацією, дає змогу самостійно налаштовувати VPN-протоколи, маршрутизацію, NAT і фаєрвол.

2 АНАЛІЗ VPN ПРОТОКОЛІВ

У роботі розглядаються три поширені VPN-протоколи: IKEv2/IPsec, WireGuard та OpenVPN. Вони відрізняються архітектурою, способом автентифікації, моделлю керування ключами, рівнем складності конфігурації та особливостями проходження через NAT і фаєрволи.

Розгляд цих протоколів дозволяє визначити їх переваги й обмеження для подальшого використання у VPN-інфраструктурі.

2.1 IKEv2/IPsec

IKEv2/IPsec є стандартизованим підходом до побудови захищених VPN-з’єднань на мережному рівні. Його особливість полягає в тому, що він поєднує два компоненти: IPsec та IKEv2. IPsec відповідає за безпосередній захист IP-трафіку, тоді як IKEv2 виконує функції автентифікації сторін, узгодження криптографічних параметрів, формування сеансових ключів та керування асоціаціями безпеки. Такий поділ дозволяє відокремити службовий процес встановлення захищеного з’єднання від подальшого передавання користувацького трафіку [2].

IPsec не є одним окремим протоколом, а являє собою архітектуру захисту IP-трафіку. Вона визначає набір механізмів, за допомогою яких може забезпечуватися конфіденційність, цілісність, автентифікація джерела даних і захист від повторного відтворення пакетів.

IPsec працює на мережному рівні та застосовує захист безпосередньо до IP-пакетів, а не до окремої прикладної сесії. Завдяки цьому один VPN-тунель може захищати різні типи трафіку без окремого налаштування шифрування для кожного застосунку. Така модель робить IPsec придатним для побудови захищених каналів між вузлами, мережами або клієнтом і VPN-шлюзом [2].

Основою функціонування IPsec є поняття Security Association, або асоціації

безпеки. Асоціація безпеки визначає набір параметрів, які використовуються для захисту певного потоку трафіку. До таких параметрів належать криптографічні алгоритми, ключі, напрям передавання, час життя асоціації, ідентифікатори сторін, режим роботи IPsec та політики обробки пакетів.

Security Association є односпрямованою, тому для повноцінного двостороннього обміну даними створюються щонайменше дві асоціації: одна для вхідного, а інша для вихідного трафіку.

У межах IPsec використовуються дві важливі логічні бази: Security Policy Database та Security Association Database. Security Policy Database визначає, як саме має оброблятися трафік: бути захищеним за допомогою IPsec, передаватися без захисту або блокуватися. Security Association Database містить параметри вже створених асоціацій безпеки, які використовуються для фактичної обробки пакетів. Такий підхід дозволяє не застосовувати шифрування до всього трафіку без розбору, а керувати захистом відповідно до визначених політик [10].

Політики IPsec визначають не лише факт застосування захисту, а й умови, за яких він має використовуватися. Наприклад, політика може встановлювати, що трафік між певними IP-адресами або підмережами повинен передаватися тільки через захищену асоціацію, тоді як інший трафік може передаватися звичайним способом. Якщо пакет не відповідає заданим політикам, він може бути відкинутий або оброблений без IPsec-захисту. Це дає змогу гнучко керувати тим, які саме мережні потоки потрапляють у VPN-тунель [10].

У зв’язці IKEv2/IPsec використовуються два основні типи асоціацій безпеки: IKE SA та CHILD SA. IKE SA створюється для захисту службового обміну між сторонами. Через неї виконуються автентифікація, узгодження параметрів, оновлення ключів і створення додаткових асоціацій. CHILD SA використовується вже для захисту користувацького трафіку за допомогою IPsec. Таким чином, IKE SA відповідає за керування з’єднанням, а CHILD SA за безпосередній захист переданих даних [11].

Для визначення трафіку, який має бути захищений конкретною CHILD SA, в IKEv2 використовуються Traffic Selectors. Вони описують діапазони IP-адрес,

протоколи та порти, до яких застосовується IPsec захист. Одна асоціація може бути створена для всього трафіку між двома підмережами, а інша – лише для окремого типу мережної взаємодії.

Завдяки Traffic Selectors IKEv2/IPsec може підтримувати різні сценарії маршрутизації трафіку, зокрема повне тунелювання або вибіркове передавання лише певних потоків через захищений канал.

Traffic Selectors також виконують важливу роль під час узгодження параметрів між сторонами. Кожна сторона може запропонувати власний діапазон трафіку, який має бути захищений, після чого сторони узгоджують спільний допустимий набір. Це запобігає ситуації, коли одна сторона очікує захисту ширшого діапазону трафіку, ніж підтримує або дозволяє інша. У практичних VPN-конфігураціях некоректне узгодження Traffic Selectors може призводити до ситуацій, коли тунель формально встановлений, але частина трафіку через нього не проходить [11].

Основним протоколом захисту трафіку в сучасних IPsec-рішеннях є ESP (Encapsulating Security Payload). ESP забезпечує шифрування даних, контроль цілісності, автентифікацію джерела даних і захист від повторного відтворення пакетів. Перехоплений трафік не може бути прочитаний без відповідних ключів, а спроби змінити або повторно використати пакети можуть бути виявлені. ESP є основним механізмом, який забезпечує реальний криптографічний захист даних у більшості реалізацій IPsec.

ESP може забезпечувати як шифрування, так і контроль цілісності даних. Залежно від обраних алгоритмів, ці функції можуть реалізовуватися окремо або в межах автентифікованого шифрування. Для VPN-з’єднань це має принципове значення, оскільки недостатньо лише приховати зміст переданих даних; необхідно також переконатися, що пакет не був змінений під час передавання. Саме поєднання конфіденційності та контролю цілісності робить ESP основним інструментом захисту IPsec-трафіку [10].

Під час роботи ESP використовується SPI (Security Parameters Index). Це числовий ідентифікатор, який дозволяє отримувачу визначити, до якої асоціації

безпеки належить конкретний пакет. На основі SPI, IP-адреси призначення та протоколу отримувач знаходить відповідний запис у Security Association Database і застосовує потрібні параметри обробки.

Також ESP використовує порядкові номери пакетів, які застосовуються для захисту від атак повторного відтворення (replay attack). Якщо зловмисник повторно надсилає раніше перехоплений пакет, механізм anti-replay дозволяє виявити таку спробу [10].

IPsec може працювати у двох основних режимах: transport mode та tunnel mode. У транспортному режимі захищається корисне навантаження IP-пакета, тоді як початковий IP-заголовок залишається відкритим. Такий режим зазвичай використовується для захисту взаємодії між двома кінцевими вузлами. У тунельному режимі весь початковий IP-пакет інкапсулюється в новий IP-пакет. Саме tunnel mode найчастіше використовується у VPN, оскільки він дозволяє передавати внутрішній трафік через зовнішню мережу, приховуючи початкову структуру пакета.

Тунельний режим має особливе значення для побудови VPN-з’єднань. У цьому режимі початковий пакет розглядається як корисне навантаження нового пакета, який передається між IPsec-вузлами. Завдяки цьому можна організувати захищений канал між клієнтом і шлюзом або між двома шлюзами. Така модель дозволяє передавати трафік приватних адресних просторів через публічні мережі, не розкриваючи внутрішню адресацію кінцевих вузлів зовнішнім учасникам маршрутизації.

У transport mode зовнішні маршрутизатори бачать початкові IP-адреси відправника й отримувача, оскільки IP-заголовок не інкапсулюється повністю. У tunnel mode зовнішня мережа бачить лише адреси IPsec-шлюзів або кінцевих точок тунелю, тоді як внутрішній пакет залишається прихованим усередині інкапсульованого навантаження. Саме тому tunnel mode краще відповідає логіці VPN: він дозволяє створити окремий захищений канал поверх існуючої мережної інфраструктури [10].

Процес встановлення IKEv2/IPsec-з’єднання починається з обміну

IKE_SA_INIT. На цьому етапі сторони узгоджують набір криптографічних алгоритмів, обмінюються параметрами Діффі–Геллмана та формують початковий ключовий матеріал. Цей обмін створює основу для подальшого захищеного службового каналу, але ще не завершує автентифікацію сторін.

Сторони можуть сформувати спільний сеансовий ключ навіть тоді, коли сам обмін відбувається через незахищену мережу [11].

Під час IKE_SA_INIT сторони також узгоджують так звані криптографічні пропозиції. До них належать алгоритми шифрування, контролю цілісності, псевдовипадкової функції та група Діффі–Геллмана. Якщо сторони не мають спільного набору підтримуваних параметрів, з’єднання не може бути встановлене. Це є важливим практичним аспектом, оскільки несумісність криптографічних профілів є однією з типових причин помилок під час налаштування IKEv2/IPsec.

Після IKE_SA_INIT виконується обмін IKE_AUTH. На цьому етапі сторони підтверджують свою ідентичність і завершують створення IKE SA.

Автентифікація може виконуватися за допомогою попередньо спільного ключа, цифрових сертифікатів або EAP-механізмів. Також під час IKE_AUTH створюється перша CHILD SA, яка використовується для захисту користувацького IP-трафіку. Після завершення цього етапу VPN-з’єднання фактично готове до передавання даних.

На етапі IKE_AUTH важливою є перевірка ідентичності сторін. Ідентифікатор учасника з’єднання має відповідати використаному методу автентифікації та політикам довіри. Наприклад, у разі сертифікатної автентифікації перевіряється не лише сам факт наявності сертифіката, а й ланцюг довіри, термін дії сертифіката та відповідність ідентифікатора очікуваному вузлу або користувачу. Це дозволяє знизити ризик підключення неавторизованої сторони [11].

У процесі роботи IKEv2 використовує додаткові службові обміни. CREATE_CHILD_SA застосовується для створення нових CHILD SA або оновлення ключів для вже наявних асоціацій. Це дозволяє підтримувати

довготривалі захищені з’єднання без необхідності повного повторного встановлення VPN-тунелю. INFORMATIONAL використовується для передавання службових повідомлень, повідомлень про помилки, видалення асоціацій безпеки та підтримання стану з’єднання. Завдяки цьому IKEv2 забезпечує повний життєвий цикл керування IPsec-з’єднанням [11].

Оновлення ключів, або rekeying, є важливим елементом безпечної роботи IKEv2/IPsec. Воно передбачає періодичне створення нового ключів для IKE SA або CHILD SA. Це зменшує ризики, пов’язані з тривалим використанням одного й того самого ключа, і дозволяє підтримувати криптографічну стійкість з’єднання протягом часу [11].

Надто часте оновлення може створювати додаткове навантаження, а надто рідке збільшувати час використання одних сеансових ключів.

IKEv2 також підтримує механізми виявлення недоступності іншої сторони. Це важливо для ситуацій, коли один із вузлів раптово втрачає з’єднання, змінює адресу або перестає відповідати. У таких випадках службові механізми IKEv2 дозволяють визначити, чи залишається асоціація актуальною, або її потрібно видалити й створити заново. Завдяки цьому протокол краще пристосований до нестабільних мережних умов, ніж підходи, де стан тунелю контролюється лише вручну.

IKEv2 підтримує кілька методів автентифікації сторін. Найпростішим варіантом є PSK (Pre Shared Key). Його перевагою є простота налаштування, однак у більших системах такий підхід ускладнює масштабування і безпечну зміну секретів. Іншим варіантом є автентифікація за допомогою сертифікатів X.509, яка дозволяє будувати інфраструктуру довіри, відкликати скомпрометовані сертифікати та не використовувати один спільний секрет для всіх учасників. Також IKEv2 підтримує EAP, що корисно для сценаріїв віддаленого доступу користувачів [11].

Сертифікатна автентифікація є більш складною в адмініструванні, але має важливі переваги з погляду безпеки. Вона дозволяє окремо ідентифікувати кожного клієнта або шлюз, застосовувати терміни дії сертифікатів і відкликати

доступ без зміни загального секрету для всієї системи. У випадку використання PSK компрометація одного секрету може поставити під загрозу більшу частину інфраструктури, тоді як сертифікати дозволяють точніше контролювати довіру між учасниками з’єднання [11].

Окремою важливою можливістю IKEv2/IPsec є підтримка NAT traversal. У багатьох реальних мережах клієнти перебувають за NAT, що створює проблему для класичного ESP, оскільки ESP не використовує TCP або UDP-порти так, як це роблять звичайні транспортні протоколи. Для вирішення цієї проблеми ESP-трафік може інкапсулюватися в UDP. Це дозволяє IPsec-з’єднанню проходити через NAT-пристрої та фаєрволи, які орієнтуються на транспортні порти під час обробки трафіку [11].

На практиці IKEv2 зазвичай використовує UDP-порт 500 для початкового IKE-обміну та UDP-порт 4500 у разі використання NAT traversal. Це має значення для налаштування фаєрволів і мережних політик, оскільки блокування цих портів може повністю унеможливити встановлення VPN-з’єднання. Також потрібно враховувати, що деякі мережі можуть обмежувати або фільтрувати UDP-трафік, через що стабільність IKEv2/IPsec може залежати не лише від конфігурації сервера й клієнта, а й від проміжної мережної інфраструктури [11].

Ще однією перевагою IKEv2 є підтримка мобільності за допомогою розширення MOBIKE. Це розширення дозволяє змінювати IP-адреси, пов’язані з IKEv2 та IPsec tunnel mode Security Associations, без необхідності повного перевстановлення VPN-з’єднання. Така можливість є корисною для мобільних клієнтів, які можуть переходити між різними мережами, наприклад з Wi-Fi на мобільний Інтернет. Завдяки цьому VPN-сесія може залишатися активною навіть після зміни мережного інтерфейсу або зовнішньої IP-адреси клієнта.

З погляду безпеки IKEv2/IPsec забезпечує кілька ключових властивостей. Конфіденційність досягається завдяки шифруванню трафіку за допомогою ESP. Цілісність забезпечується криптографічною перевіркою даних, що дозволяє виявити зміну пакетів під час передавання. Автентифікація дозволяє сторонам перевірити одна одну перед створенням захищеного каналу. Anti-replay-

механізми захищають від повторного використання раніше перехоплених пакетів. У сукупності ці механізми забезпечують комплексний захист IP-трафіку [10].

До переваг IKEv2/IPsec можна віднести стандартизованість, підтримку різних методів автентифікації, можливість роботи в тунельному режимі, підтримку NAT traversal і мобільності через MOBIKE. Також важливою перевагою є широка підтримка в операційних системах і мережному обладнанні. Завдяки цьому IKEv2/IPsec може застосовуватися в різних сценаріях: від віддаленого доступу окремих користувачів до побудови захищених тунелів між мережними сегментами.

Серед практичних особливостей IKEv2/IPsec варто виділити велику кількість взаємопов’язаних параметрів. До них належать криптографічні пропозиції, тип автентифікації, ідентифікатори сторін, Traffic Selectors, режими IPsec, час життя асоціацій безпеки, правила NAT traversal та політики маршрутизації. Це забезпечує гнучкість і можливість адаптації до різних мережних сценаріїв, але така кількість параметрів підвищує вимоги до коректності проєктування та налаштування VPN.

Таким чином, IKEv2/IPsec є надійним і стандартизованим механізмом побудови VPN-з’єднань, який поєднує керування асоціаціями безпеки з криптографічним захистом IP-трафіку. Його сильними сторонами є підтримка різних методів автентифікації, захист на мережному рівні, NAT traversal, MOBIKE та широка сумісність із клієнтськими платформами. Водночас практична реалізація IKEv2/IPsec потребує уважного налаштування політик, маршрутів, автентифікації та параметрів безпеки.

2.2 WireGuard

WireGuard є сучасним VPN-протоколом, призначеним для створення захищених тунелів на мережному рівні. Його основна ідея полягає у поєднанні простої архітектури, високої продуктивності та сучасної криптографії. На

відміну від складніших VPN-рішень, таких як IKEv2/IPsec, WireGuard має мінімалістичну модель роботи й не потребує великої кількості параметрів для встановлення захищеного з’єднання. Протокол працює як віртуальний мережний інтерфейс, через який передаються зашифровані IP-пакети між налаштованими вузлами [12].

WireGuard реалізує однорангову модель взаємодії. На рівні протоколу немає жорсткого поділу на клієнт і сервер, кожен учасник розглядається як peer. Кожен peer має власну пару криптографічних ключів: приватний і публічний. Приватний ключ зберігається локально й не передається мережею, а публічний ключ використовується іншими вузлами для ідентифікації цього peer. Така модель спрощує встановлення довіри між сторонами, оскільки для базової взаємодії достатньо обмінятися публічними ключами [12].

Поділ на сервер і клієнт у WireGuard є радше практичним, ніж протокольним. Вузол зі статичною публічною IP-адресою та відкритим UDP-портом часто виконує роль центрального peer, до якого підключаються інші учасники. Проте з погляду самого протоколу всі вузли залишаються рівноправними peer. Це відрізняє WireGuard від VPN-систем, де архітектура жорстко залежить від окремого сервера автентифікації або централізованого шлюзу [12].

Важливою архітектурною особливістю WireGuard є його тісна інтеграція з мережним стеком операційної системи. У Linux WireGuard від початку проєктувався як високопродуктивний мережний тунель, що працює на рівні ядра. Такий підхід дозволяє зменшити накладні витрати, пов’язані з передаванням пакетів між простором користувача та простором ядра. У порівнянні з VPN-рішеннями, які значною мірою працюють у просторі користувача, це знижує кількість перемикань контексту та позитивно впливає на продуктивність обробки пакетів [13].

Робота на рівні ядра має значення не лише для швидкодії, а й для загальної простоти інтеграції. WireGuard у Linux поводиться як звичайний мережний інтерфейс, наприклад wg0, тому до нього можуть застосовуватися стандартні

механізми маршрутизації, фаєрвола, таблиць маршрутів і системного адміністрування. Це спрощує включення WireGuard у наявну мережну інфраструктуру, оскільки адміністратор працює не з окремим прикладним тунельним процесом, а з повноцінним мережним інтерфейсом [14].

Ключовим принципом WireGuard є Cryptokey Routing. Його суть полягає в тому, що публічний ключ peer пов’язується з набором IP-адрес або підмереж, які дозволено передавати через тунель для цього вузла. Таким чином, публічний ключ у WireGuard виконує подвійну функцію: він є криптографічним ідентифікатором peer і водночас пов’язується з маршрутизацією трафіку. Якщо IP-пакет має бути переданий до адреси, що належить до AllowedIPs певного peer, WireGuard шифрує цей пакет для відповідного вузла й надсилає його через UDP-тунель [12].

Параметр AllowedIPs є центральним елементом конфігурації WireGuard. Він визначає, які IP-адреси або підмережі пов’язані з конкретним peer і який трафік має бути спрямований через VPN-інтерфейс.

Наприклад, окрема адреса може використовуватися для точкового з’єднання між двома вузлами, підмережа для доступу до групи хостів, а маршрути 0.0.0.0/0 для повного тунелювання IPv4- або IPv6-трафіку. У цьому полягає гнучкість WireGuard: один і той самий механізм може застосовуватися для point-to-point, site-to-site і remote access сценаріїв [14].

Типовою помилкою конфігурації WireGuard є некоректне визначення AllowedIPs. Якщо вказати занадто вузький діапазон, частина потрібного трафіку не потрапить у тунель. Якщо вказати занадто широкий діапазон, через VPN може піти більше трафіку, ніж передбачено, аж до повного перенаправлення всього інтернет-трафіку клієнта [14].

WireGuard працює поверх UDP. Усі службові й користувацькі пакети передаються через UDP, що спрощує реалізацію протоколу та зменшує накладні витрати. На практиці часто використовується порт 51820/UDP, хоча конкретний порт може бути змінений адміністратором [15].

Використання UDP дозволяє уникнути проблеми “TCP over TCP”, яка

може виникати при тунелюванні TCP трафіку через TCP з’єднання. Проблема характеризується суттєвим зниженням продуктивності мережі.

Передавання через UDP означає, що WireGuard не гарантує доставку пакетів на рівні самого тунелю. Це відповідає логіці IP-мереж: якщо всередині VPN передається TCP-трафік, надійність забезпечується самим TCP; якщо UDP-трафік – він передається без додаткового механізму повторної доставки. Такий підхід зменшує затримки, оскільки VPN-протокол не дублює функції транспортного рівня [15].

Робота поверх UDP і додавання криптографічних заголовків впливають на доступний розмір корисного навантаження пакета. Через це для WireGuard важливим є правильне налаштування MTU. Якщо MTU встановлено занадто високо, можуть виникати фрагментація або проблеми з доступом до окремих ресурсів, особливо коли ICMP-повідомлення про необхідність фрагментації блокуються проміжними мережами [15].

У практичних конфігураціях часто використовується значення MTU близько 1420, хоча оптимальне значення залежить від конкретного каналу, провайдера, додаткової інкапсуляції та мережного середовища.

Криптографічна модель WireGuard побудована на основі фіксованого набору сучасних алгоритмів. Протокол використовує Curve25519 для обміну ключами, ChaCha20 для симетричного шифрування, Poly1305 для автентифікації повідомлень, BLAKE2s для хешування, SipHash для хеш-таблиць і HKDF для отримання похідних ключів.

Відсутність вибору криптографічних алгоритмів є свідомим проєктним рішенням WireGuard. У традиційних VPN-протоколах адміністратор часто має самостійно визначати набір алгоритмів, групи обміну ключами та параметри автентифікації. Це збільшує гнучкість, але водночас підвищує ризик неправильної або застарілої конфігурації. WireGuard натомість використовує один сучасний криптографічний набір, що робить поведінку протоколу передбачуванішою й зменшує простір для помилок [13].

Обмін ключами у WireGuard базується на протоколі Noise_IK із Noise

Protocol Framework. Під час встановлення з’єднання сторони здійснюють взаємну автентифікацію на основі статичних відкритих ключів, у межах якого автентифікують одна одну за допомогою довгострокових публічних ключів і формують тимчасові сеансові ключі. Ці ключі використовуються для шифрування транспортних даних, але не зберігаються як постійний секрет. Такий підхід забезпечує Perfect Forward Secrecy, тобто компрометація довгострокового ключа в майбутньому не повинна автоматично розкривати зміст раніше переданого трафіку [15].

Процес встановлення захищеного з'єднання у WireGuard є значно простішим, ніж у IKEv2/IPsec. Він не передбачає складного багатоступеневого узгодження політик, наборів алгоритмів або окремих асоціацій безпеки. Замість цього WireGuard використовує наперед визначений криптографічний набір і мінімальний службовий обмін. Якщо peer має коректний публічний ключ іншої сторони, правильно визначений endpoint і відповідні AllowedIPs, він може ініціювати захищене з’єднання без окремої інфраструктури сертифікації [15].

Після успішного встановлення з’єднання WireGuard створює сеансові ключі для передавання даних. Ці ключі періодично оновлюються, що дозволяє обмежити обсяг трафіку, захищеного одним і тим самим ключовим матеріалом.

На відміну від IPsec, де адміністратор часто явно налаштовує параметри часу життя SA та rekeying, у WireGuard значна частина цієї логіки прихована всередині протоколу. Це спрощує експлуатацію, оскільки адміністратор не повинен вручну керувати великою кількістю параметрів життєвого циклу ключів [15].

WireGuard не використовує традиційну модель користувачів, логінів або паролів. Ідентичність вузла визначається його публічним ключем. Якщо публічний ключ peer відсутній у конфігурації, пакети від такого вузла не будуть прийняті як довірені. Такий підхід зменшує залежність від централізованих механізмів автентифікації, однак покладає відповідальність за безпечне зберігання приватних ключів і правильний обмін публічними ключами на адміністратора системи [12].

Захист приватного ключа є критично важливим для безпеки WireGuard. Оскільки саме приватний ключ визначає ідентичність peer, його компрометація дозволяє зловмиснику імітувати відповідний вузол. Тому приватні ключі мають зберігатися з обмеженими правами доступу, не передаватися відкритими каналами та не використовуватися повторно в різних незалежних середовищах без потреби. У разі компрометації ключа адміністратор повинен видалити відповідний публічний ключ з конфігурації інших peer і згенерувати нову пару ключів [14].

Особливістю WireGuard є мінімальний обсяг інформації про стан, який протокол зберігає для кожного вузла. Якщо трафік між сторонами відсутній, з'єднання фактично не підтримується як постійна активна сесія у класичному розумінні. WireGuard виконує процедуру автентифікації тоді, коли виникає потреба передати трафік або оновити ключі. Це робить протокол ефективним для середовищ, де з'єднання може бути переривчастим або де peer не завжди має постійну мережну активність [13].

Ця властивість також важлива для мобільних пристроїв. Оскільки WireGuard не потребує постійного активного службового обміну за відсутності трафіку, він може бути енергоефективнішим для смартфонів і ноутбуків.

Для підтримання з'єднання через NAT у WireGuard використовується параметр PersistentKeepalive. Він змушує peer періодично надсилати невеликі службові пакети, щоб NAT-пристрій не видаляв відповідний запис трансляції. Це особливо важливо у випадках, коли один із peer перебуває за NAT і не має постійної публічної IP-адреси. Наприклад, для клієнта за домашнім маршрутизатором PersistentKeepalive дозволяє зберігати можливість отримання пакетів від віддаленого вузла [14].

WireGuard не має окремого складного механізму NAT traversal, подібного до IKEv2/IPsec. Оскільки протокол працює поверх UDP, його проходження через NAT часто є простішим, але не повністю автоматичним. Якщо обидва peer перебувають за NAT і не мають доступного endpoint, може знадобитися проміжний вузол із публічною адресою або додаткові механізми організації

з’єднання. У типовій VPN-топології цю роль виконує центральний peer із відкритим UDP-портом [14].

WireGuard не має вбудованого механізму динамічного призначення IP-адрес, подібного до DHCP усередині VPN. Адреси інтерфейсів і AllowedIPs задаються в конфігурації. З одного боку, це робить поведінку тунелю передбачуваною та простою для аналізу. З іншого боку, адміністратор повинен самостійно стежити, щоб IP-адреси peer не конфліктували між собою, а маршрути були визначені коректно. У складніших інфраструктурах це може вимагати додаткової автоматизації конфігурації [14].

З практичного погляду конфігурація WireGuard складається з двох основних блоків: Interface та Peer. У секції Interface задаються параметри локального вузла, зокрема приватний ключ, адреса VPN-інтерфейсу та порт прослуховування. У секції Peer вказуються параметри віддаленого вузла: його публічний ключ, AllowedIPs, Endpoint і за потреби PersistentKeepalive. Така структура є значно простішою, ніж конфігурації IPsec, де окремо налаштовуються політики, асоціації безпеки, криптографічні пропозиції та методи автентифікації [14].

Важливим елементом WireGuard є параметр Endpoint. Він визначає зовнішню адресу та порт peer, на які потрібно надсилати UDP-пакети. При цьому WireGuard може оновлювати фактичний endpoint peer після отримання автентичного пакета з іншої адреси. Це корисно для мобільних або динамічних клієнтів, у яких змінюється IP-адреса. Така поведінка дозволяє WireGuard працювати в умовах зміни мережі без складних окремих механізмів мобільності [15].

WireGuard забезпечує конфіденційність, цілісність і автентифікацію трафіку. Конфіденційність досягається за рахунок симетричного шифрування, цілісність за рахунок автентифікації повідомлень, а автентифікація peer через використання публічних ключів.

Протокол має захист від атак повторного відтворення (replay attack), оскільки пакети містять лічильники, які дозволяють виявляти повторно надіслані

повідомлення. У сукупності ці механізми забезпечують захист IP-пакетів, що передаються через тунель.

Захист від replay-атак у WireGuard має практичне значення для безпеки тунелю. Якщо зловмисник перехопить зашифрований пакет і спробує надіслати його повторно, протокол не повинен сприйняти такий пакет як новий коректний трафік. Це доповнює шифрування, оскільки захищає не лише вміст пакетів, а й саму логіку їх приймання [13].

До переваг WireGuard належать простота конфігурації, невеликий обсяг кодової бази, висока продуктивність і використання сучасних криптографічних примітивів.

Мінімалістична архітектура зменшує кількість параметрів, які може неправильно налаштувати адміністратор, і спрощує аудит безпеки. Крім того, завдяки роботі як звичайного мережного інтерфейсу WireGuard добре інтегрується зі стандартними механізмами маршрутизації та фаєрвола [13].

Ще однією перевагою WireGuard є відсутність складного узгодження криптографічних алгоритмів. У багатьох VPN-протоколах сторони повинні домовитися про набір алгоритмів, і помилка в такій конфігурації може призвести або до неможливості встановлення з’єднання. WireGuard натомість використовує фіксований криптографічний набір, що робить поведінку протоколу більш передбачуваною [15].

Серед практичних обмежень WireGuard варто зазначити відсутність вбудованої системи керування користувачами, рольової моделі доступу та інфраструктури публічних ключів.

Протокол працює з peer і публічними ключами, а тому питання обліку користувачів, відкликання доступу, видачі конфігурацій і контролю життєвого циклу ключів мають вирішуватися зовнішніми засобами. Для невеликої кількості вузлів це спрощує адміністрування, але у великих середовищах може потребувати додаткового рівня керування [14].

На практиці цей недолік часто компенсується сторонніми надбудовами над WireGuard. Наприклад, Tailscale використовує WireGuard як основу для

захищених з’єднань і додає контрольну площину для керування пристроями, ключами та доступом. Netmaker також використовує WireGuard для створення зашифрованих тунелів і автоматизує конфігурацію та маршрутизацію.

Такі рішення не змінюють базову криптографічну модель WireGuard, але додають рівень адміністрування, який відсутній у самому протоколі.

Проте варто враховувати, що використання цих платформ має сенс для корпоративних користувачів, де потреба в автоматизації виправдовує додаткові витрати.

Таким чином, WireGuard є сучасним VPN-протоколом, який поєднує просту архітектуру, продуктивність і сильну криптографічну модель. Його робота базується на публічних ключах, Cryptokey Routing, фіксованому наборі криптографічних алгоритмів і передаванні зашифрованих IP-пакетів через UDP.

Основними перевагами WireGuard є простота налаштування і діагностики, менша кількість параметрів, висока швидкодія, робота як мережного інтерфейсу та підтримка Perfect Forward Secrecy.

Основними недоліками WireGuard є потреба в ручному або зовнішньо автоматизованому керуванні ключами і менша гнучкість у виборі криптографічних алгоритмів.

2.3 OpenVPN

OpenVPN є відкритим VPN-рішенням, яке використовується для створення захищених тунелів між клієнтами, серверами або окремими мережними сегментами. На відміну від IPsec, який реалізує захист безпосередньо на рівні IP, OpenVPN працює як окремий VPN-демон і формує тунель поверх транспортного протоколу UDP або TCP. Основою його моделі безпеки є використання SSL/TLS для автентифікації сторін, обміну ключовим матеріалом і захисту керувального каналу. Завдяки цьому OpenVPN поєднує криптографічну модель TLS із власним механізмом інкапсуляції тунельного трафіку [16].

Архітектура OpenVPN логічно поділяється на control channel і data channel.

Control channel відповідає за службову частину роботи з’єднання: встановлення TLS-сесії, автентифікацію сторін, узгодження параметрів і формування ключів. Data channel використовується для передавання основного VPN-трафіку, тобто IP-пакетів або Ethernet-кадрів, які інкапсулюються в тунель і захищаються симетричними криптографічними механізмами. Такий поділ дозволяє відокремити логіку встановлення й керування з’єднанням від передавання користувацьких даних [17].

Використання TLS у OpenVPN має принципове значення, оскільки саме TLS забезпечує захищений механізм первинної автентифікації та обміну ключами.

В сучасних версіях OpenVPN може використовуватися TLS 1.3, який скорочує кількість службових обмінів під час встановлення з’єднання і вилучає низку застарілих криптографічних механізмів, характерних для попередніх версій TLS. Для OpenVPN це означає швидше формування захищеного control channel і зменшення ризику використання небажаних криптографічних параметрів під час встановлення з’єднання [18, 16].

Вибір між UDP і TCP визначає транспортну поведінку VPN-тунелю. У режимі UDP OpenVPN передає тунельні пакети без додаткового механізму гарантованої доставки на рівні самого VPN-протоколу, що зменшує затримки й накладні витрати. Якщо всередині тунелю передається TCP-трафік, надійність забезпечується самим TCP усередині VPN. У режимі TCP OpenVPN легше провести через мережі з жорсткою фільтрацією, однак при передаванні TCP-трафіку всередині TCP-тунелю виникає ефект “TCP over TCP”: два рівні одночасно виконують повторні передавання та контроль перевантаження, що може погіршувати продуктивність [16].

Використання інтерфейсів TUN і TAP визначає, на якому рівні мережної моделі працюватиме тунель.

Інтерфейс TUN формує L3-тунель і передає IP-пакети, тому застосовується для маршрутизованих VPN-з’єднань між клієнтом і сервером або між окремими мережами.

Інтерфейс TAP працює на L2-рівні та передає Ethernet-кадри, що дозволяє будувати мостові з’єднання між віддаленими сегментами.

Практично це означає, що TUN є доцільнішим для більшості VPN-сценаріїв, оскільки не передає зайвий широкомовний трафік, тоді як TAP потрібен лише у випадках, де необхідна саме L2-прозорість [16].

У маршрутизованому режимі OpenVPN створює окрему IP-підмережу для VPN-клієнтів. Клієнт отримує адресу з цієї підмережі, після чого маршрути визначають, який трафік має проходити через тунель. Сервер може передавати клієнтам маршрути до внутрішніх мереж або налаштовувати повне тунелювання всього трафіку [19].

Такий підхід дозволяє гнучко керувати тим, які ресурси доступні через VPN, але потребує правильного налаштування IP forwarding, NAT і правил фаєрвола на стороні сервера.

У мостовому режимі на основі TAP OpenVPN може об’єднувати віддалені сегменти на канальному рівні. Це дозволяє передавати Ethernet-кадри, включно з широкомовним трафіком або не-IP-протоколами. Однак така модель створює більші накладні витрати, гірше масштабується та складніше контролюється з погляду маршрутизації. Тому TAP доцільно використовувати лише тоді, коли потрібно зберегти поведінку єдиного L2-сегмента, а для типового VPN-доступу практичнішим є TUN [16].

Криптографічна модель OpenVPN базується на поєднанні TLS для керувального каналу та симетричного шифрування для data channel. Після встановлення TLS-сесії сторони отримують ключовий матеріал, який використовується для захисту основного тунельного трафіку. У сучасних конфігураціях перевага надається AEAD-шифрам, зокрема AES-GCM або ChaCha20-Poly1305, які одночасно забезпечують шифрування та автентифікацію даних. Це зменшує ризик помилок, пов’язаних із роздільним налаштуванням шифрування та контролю цілісності [17].

У старіших або сумісних конфігураціях OpenVPN може використовувати окреме шифрування та HMAC для перевірки цілісності. У такій схемі HMAC

дозволяє виявляти модифікацію або підробку пакетів. Якщо ж застосовується AEAD-режим, окремий HMAC для data channel зазвичай не потрібен, оскільки автентифікація вже входить до складу самого режиму шифрування. Таким чином, сучасна конфігурація OpenVPN поступово зміщується в бік простіших і стійкіших криптографічних схем [16, 17].

Для автентифікації сторін OpenVPN зазвичай використовує інфраструктуру відкритих ключів на основі сертифікатів X.509. У такій моделі центр сертифікації підписує сертифікати сервера та клієнтів, а сторони перевіряють належність сертифікатів до довіреної інфраструктури. Це дозволяє не використовувати один спільний секрет для всіх клієнтів і дає змогу відкликати доступ окремого вузла без зміни всієї системи автентифікації. Такий підхід є зручним для середовищ, де потрібно керувати доступом багатьох клієнтів [20].

Механізм додаткової користувацької автентифікації в OpenVPN дозволяє відокремити ідентифікацію пристрою від ідентифікації користувача. Сертифікат X.509 підтверджує, що клієнтський пристрій або конфігурація належить до довіреної інфраструктури, тоді як логін, пароль або другий фактор підтверджують особу користувача. Така схема зменшує ризик несанкціонованого доступу у випадку втрати конфігураційного файлу або компрометації пристрою, оскільки для підключення недостатньо лише мати сертифікат [16, 21].

Підтримка плагінів і зовнішніх систем автентифікації дає змогу інтегрувати OpenVPN із механізмами багатофакторної автентифікації. У такому випадку користувач може проходити додаткову перевірку через одноразовий код, зовнішній провайдер ідентифікації або іншу систему контролю доступу. Це робить OpenVPN придатним для сценаріїв, де VPN має захищати не лише мережний канал, а й сам процес доступу конкретних користувачів до інфраструктури [16, 22].

Механізми tls-auth і tls-crypt виконують роль попереднього захисту каналу керування. Tls-auth додає HMAC-перевірку до службових TLS-пакетів, тому сервер може відкинути неавторизований трафік ще до виконання повної TLS-

обробки. Це знижує навантаження від небажаних пакетів і ускладнює просте сканування OpenVPN-сервісу. Tls-crypt додатково шифрує службові пакети control channel, ускладнюючи ідентифікацію OpenVPN-трафіку сторонніми спостерігачами [23].

На рівні власного протоколу OpenVPN використовує формат пакетів, у межах якого передаються як службові повідомлення, так і тунельні дані. Для control channel передбачені службові пакети, підтвердження отримання, повторне передавання та підтримання стану з'єднання. Це потрібно тому, що TLS очікує надійної доставки службових повідомлень, а OpenVPN може працювати поверх UDP. Водночас data channel не дублює функцію гарантованої доставки: якщо всередині тунелю передається TCP, за надійність відповідає сам TCP [16].

OpenVPN може працювати в режимі статичного ключа (Static Key mode) або в TLS-режимі. Static Key mode є простішим, але погано масштабується, оскільки потребує попереднього розповсюдження одного спільного ключа між сторонами. TLS-режим краще підходить для реальних VPN-сценаріїв, оскільки дозволяє використовувати сертифікати, окрему автентифікацію клієнтів, відкликання доступу та централізоване керування довірою. Тому в інфраструктурах із кількома клієнтами TLS-режим є основним практичним варіантом [16].

Велика кількість параметрів OpenVPN є наслідком його гнучкої архітектури. Адміністратор може окремо керувати транспортом, типом тунельного інтерфейсу, TLS-параметрами, сертифікатами, шифрами data channel, маршрутами, DNS-параметрами, push-опціями, MTU та додатковою автентифікацією. Це дозволяє адаптувати OpenVPN до різних мережних сценаріїв, але одночасно збільшує кількість місць, де помилка конфігурації може призвести до проблем із підключенням, маршрутизацією або безпекою [16].

Маршрутизація в OpenVPN визначає, який саме трафік буде передаватися через VPN-тунель. Сервер може передавати клієнтам маршрути до внутрішніх підмереж або змінювати маршрут за замовчуванням для повного тунелювання.

Якщо TLS-з’єднання встановлюється успішно, але користувацький трафік не проходить, причина часто пов’язана не з криптографією, а з маршрутами, NAT, IP forwarding або правилами фаєрвола. Тому перевірка маршрутизації є обов’язковою частиною практичного налаштування OpenVPN [19].

Параметр MTU має важливе значення для стабільності OpenVPN-тунелю, оскільки інкапсуляція, шифрування та службові заголовки зменшують доступний розмір корисного навантаження. Якщо MTU налаштовано неправильно, можуть виникати фрагментація, зависання окремих TCP-з’єднань або недоступність частини ресурсів. Для зменшення таких проблем в OpenVPN може використовуватися параметр mssfix, який коригує TCP MSS і тим самим впливає саме на TCP-трафік.

Водночас mssfix не вирішує проблеми фрагментації для UDP-трафіку, оскільки UDP не використовує MSS. Тому для комплексного налаштування розміру пакетів також застосовується параметр tun-mtu, який визначає MTU самого тунельного інтерфейсу. Поєднання mssfix і tun-mtu дозволяє точніше контролювати розмір пакетів у тунелі та зменшити ризик фрагментації в різних типах трафіку [16].

Однією з практичних переваг OpenVPN є можливість роботи в мережах із жорсткою фільтрацією. Якщо UDP-трафік блокується, OpenVPN може працювати поверх TCP, зокрема на портах, які часто дозволені в корпоративних або публічних мережах.

Наприклад, використання TCP-порту 443 іноді дозволяє пройти через обмеження, орієнтовані лише на веб-трафік. Водночас така конфігурація є компромісною, оскільки TCP-режим збільшує накладні витрати та може погіршувати продуктивність тунелю [16].

Класична реалізація OpenVPN має певні продуктивні обмеження через роботу значної частини обробки у просторі користувача. Пакети проходять через VPN демон, що створює витрати на копіювання даних і перемикання між user space та kernel space. У порівнянні з протоколами, які тісніше інтегровані з ядром операційної системи, це може збільшувати затримки та знижувати пропускну

здатність, особливо при великій кількості трафіку [16].

DCO (Data Channel Offload) змінює спосіб обробки користувацького трафіку. DCO переносить обробку data channel на рівень ядра, завдяки чому зменшується кількість контекстних перемикань і накладні витрати на проходження пакетів через VPN-тунель. Це не змінює ролі TLS у керувальному каналі, але оптимізує шлях передавання основного трафіку [25].

У контексті порівняння з WireGuard це важливо, оскільки OpenVPN із DCO зменшує частину накладних витрат, характерних для класичної обробки OpenVPN, зберігаючи при цьому гнучку модель автентифікації та керування доступом.

З погляду безпеки OpenVPN забезпечує кілька основних властивостей: конфіденційність, цілісність, автентифікацію сторін і захист службового каналу. Конфіденційність data channel забезпечується симетричним шифруванням, цілісність - HMAC або AEAD-механізмами, автентифікація - TLS і сертифікатами, а попередній захист control channel може реалізовуватися через tls-auth або tls-crypt. Кінцева безпека залежить не лише від самого протоколу, а й від актуальності версії OpenVPN, коректності TLS-параметрів, захисту приватних ключів і якості адміністрування PKI [23].

До переваг OpenVPN можна віднести зрілість, відкритість, широку підтримку операційними системами, гнучкість конфігурації, підтримку UDP і TCP, роботу через TUN/TAP, сертифікатну модель автентифікації, можливість використання логіна, пароля та 2FA, а також розвиток у напрямі TLS 1.3 і Data Channel Offload. Ці властивості роблять OpenVPN придатним для різних сценаріїв: від простого віддаленого доступу до інфраструктур із централізованим контролем користувачів і політик доступу.

Основними обмеженнями OpenVPN є складніша конфігурація порівняно з мінімалістичними VPN-протоколами, потреба в супроводі PKI, велика кількість параметрів і потенційно вищі накладні витрати в класичній user-space-реалізації. Адміністратор повинен контролювати сертифікати, ключі, CRL, TLS-параметри, маршрути, NAT, фаєрвол, MTU та додаткові механізми автентифікації. Тому

OpenVPN є гнучким і функціональним рішенням, але потребує уважного налаштування та регулярного супроводу [16].

Діагностику функціонування OpenVPN доцільно розподіляти на два послідовні рівні, що дозволяє чітко локалізувати виникнення мережних несправностей. На першому етапі, у разі неможливості встановлення з'єднання з боку клієнта, першочерговому аналізу підлягають доступність транспортних портів (UDP або TCP), валідність цифрових сертифікатів, процедура узгодження параметрів TLS та критерії автентифікації. Другий рівень діагностики застосовується за умови успішної ініціалізації тунелю, але за відсутності пакетної передачі даних. У такому випадку фокус дослідження зміщується на перевірку таблиць маршрутизації, механізмів трансляції адрес (NAT), активації пересилання IP-пакетів (IP forwarding), конфігурації правил фаєрвола, відповідності параметрів MTU та налаштувань віртуального тунельного інтерфейсу. Така декомпозиція процесу дозволяє ефективно диференціювати збої на етапі побудови захищеного каналу від проблем у підсистемі транспортування користувацького трафіку [24].

Таким чином, OpenVPN є зрілим і гнучким VPN рішенням, яке використовує SSL/TLS для автентифікації та обміну ключами, a data channel для передавання зашифрованого тунельного трафіку.

Сильними сторонами є широка сумісність, підтримка різних режимів роботи, сертифікатна модель автентифікації, інтеграція з логіном/паролем і 2FA, підтримка TLS 1.3 та розвиток DCO.

Основними недоліками є більша складність конфігурації, потреба в якісному супроводі PKI та вищі адміністративні вимоги порівняно з простішими VPN-протоколами.

2.4 Порівняння VPN-протоколів

Розглянуті VPN-протоколи виконують спільну задачу створення захищеного каналу передавання трафіку через недовірене мережне середовище.

Водночас IKEv2/IPsec, WireGuard та OpenVPN суттєво відрізняються архітектурою, способом автентифікації, моделлю керування ключами, продуктивністю та складністю адміністрування.

Для подальшого узагальнення доцільно порівняти ці протоколи за такими критеріями: рівень роботи, транспорт, автентифікація, керування ключами, NAT traversal, мобільність, продуктивність, складність конфігурації та типові проблеми практичного розгортання.

Таблиця 2.1 – Порівняння VPN протоколів

Критерій	IKEv2/IPsec	WireGuard	OpenVPN
Основна архітектура	IPsec + IKEv2; захист IP-трафіку на мережному рівні	Мінімалістичний L3 VPN на основі peer-to-peer-моделі	VPN-демон із поділом на control channel і data channel
Рівень інкапсуляції	L3	L3	L3 (TUN) або L2 (TAP)
Транспорт	ESP для data plane; IKEv2 через UDP 500/4500 (NAT-T)	UDP	UDP або TCP
Автентифікація	PSK, X.509, EAP	Криптографічна ідентифікація peer за статичними публічними ключами	X.509, логін/пароль, 2FA
Керування ключами	IKEv2, IKE SA, CHILD SA, rekeying	NoiseIK, автоматична ротація ephemeral session keys	TLS handshake, формування ключів data channel, PKI-based certificate revocation (CRL/OCSP)
Криптографічна модель	Узгодження криптографічних пропозицій	Фіксований криптографічний стек на основі Noise Protocol Framework	TLS + налаштовувані шифри data channel
NAT traversal	NAT-T, інкапсуляція ESP в UDP	PersistentKeepalive	Працює через UDP або TCP без окремого NAT-T механізму
Мобільність	MOBIKE	Built-in roaming між endpoint	Обмежена; повторне встановлення тунелю залежить від транспортної сесії
Продуктивність	Висока	Висока	Нижча в класичній user-space реалізації; покращується за рахунок DCO

Продовження таблиці 2.1.

Критерій	IKEv2/IPsec	WireGuard	OpenVPN
Складність конфігурації	Висока	Низька	Середня або висока
Типові проблеми практичного розгортання	Помилки SA, Traffic Selectors, блокування UDP 500/4500	Некоректна маршрутизація через AllowedIPs, NAT, MTU	PKI, TLS, CRL/OCSP, NAT, MTU

Порівняння VPN-протоколів показує, що IKEv2/IPsec є найбільш формалізованим і стандартизованим рішенням серед розглянутих підходів. Його перевагами є сумісність із мережним обладнанням, підтримка PSK, X.509 та EAP-автентифікації, а також наявність MOBIKE для мобільних клієнтів. Водночас практичне розгортання IKEv2/IPsec потребує точного налаштування політик IPsec, асоціацій безпеки, Traffic Selectors і механізмів NAT traversal.

WireGuard має найбільш мінімалістичну архітектуру та найменшу кількість конфігураційних параметрів. Його модель базується на публічних ключах, Cryptokey Routing, фіксованому криптографічному стеку на основі Noise Protocol Framework і передаванні трафіку через UDP. Завдяки цьому WireGuard є ефективним для self-hosted VPN, швидкого розгортання та сценаріїв, де критичними є продуктивність і простота адміністрування. Водночас WireGuard не має вбудованої системи централізованого керування користувачами та політиками доступу.

OpenVPN є найбільш гнучким щодо автентифікації та адаптації до різних мережних умов. Його перевагами є підтримка TLS, сертифікатної моделі X.509, логіна/пароля, 2FA, UDP або TCP, а також режимів TUN і TAP. Водночас така гнучкість супроводжується вищою складністю супроводу, оскільки адміністратор повинен підтримувати PKI, CRL/OCSP, TLS-параметри, маршрутизацію, MTU, NAT і правила фаєрвола.

Таким чином, вибір VPN-протоколу залежить не лише від криптографічної стійкості, а й від вимог до адміністрування, продуктивності, масштабування та контролю доступу. Для self-hosted VPN у хмарному середовищі WireGuard є одним із найбільш практичних варіантів завдяки простій архітектурі, високій

продуктивності та невеликій складності розгортання при збереженні сучасної криптографічної моделі.

OpenVPN доцільно використовувати у випадках, де потрібні розвинена модель автентифікації, підтримка TLS, 2FA, сертифікатів і робота в мережах із жорсткою фільтрацією трафіку. IKEv2/IPsec варто розглядати як стандартизоване рішення для сценаріїв, у яких важливими є сумісність із мережним обладнанням, мобільність клієнтів і підтримка класичної IPsec-архітектури.

3 РОЗГОРТАННЯ ТА КОНФІГУРАЦІЯ VPN-ІНФРАСТРУКТУРИ

У цьому розділі розглянуто практичну реалізацію VPN-інфраструктури за моделлю self-hosted cloud VPN. Вибір цієї моделі зумовлений тим, що вона дає змогу розгорнути VPN сервер у хмарному середовищі, але при цьому зберегти повний контроль над операційною системою, програмним забезпеченням і параметрами обробки мережного трафіку.

Експериментальне середовище реалізовано за моделлю «клієнт – VPN-сервер – зовнішня мережа». Як серверний вузол використовується VPS з операційною системою Debian 12, публічною IPv4-адресою та одним основним мережним інтерфейсом. Розгортання виконується за одновузловою моделлю, тобто всі VPN рішення розгортаються та перевіряються на одному серверному вузлі.

Клієнтська частина експериментального середовища представлена двома операційними системами: Windows 11 та Linux Mint. Обидва клієнти розгорнуті у вигляді віртуальних машин з однаковими апаратними параметрами, що забезпечує однакові умови тестування на клієнтському боці. Це дозволяє оцінити роботу VPN-клієнтів у різних операційних системах без додаткового впливу відмінностей у конфігурації віртуальних машин.

У межах практичної частини на VPS послідовно розгортаються IKEv2/IPsec, WireGuard та OpenVPN. Для всіх VPN-рішень використовується однаковий функціональний сценарій: клієнт встановлює з’єднання із сервером через мережу Інтернет, отримує внутрішню адресу, після чого клієнтський трафік маршрутизується через тунель відповідно до конфігурації VPN-сервера.

Застосування self-hosted cloud VPN є доцільним для цієї роботи, оскільки дає змогу безпосередньо аналізувати ті компоненти VPN-інфраструктури, які у керованих VPN-сервісах частково приховані від користувача. До таких компонентів належать операційна система сервера, VPN-програмне забезпечення, внутрішня адресація, параметри автентифікації, механізми

пересилання IP пакетів і взаємодія VPN-тунелю з мережним стеком сервера.

Далі розглядається конфігурація окремих VPN-рішень: IKEv2/IPsec, WireGuard та OpenVPN. Для кожного VPN-рішення описуються основні елементи розгортання: програмне забезпечення, конфігураційні файли, внутрішня адресація, спосіб автентифікації та особливості підключення клієнта.

Практична частина роботи виконується в межах self-hosted cloud підходу, за якого VPN-шлюз розгортається на віртуальному сервері з публічною IPv4-адресою.

Отримане експериментальне середовище використовується як основа для подальшого оцінювання складності розгортання, особливостей адміністрування та практичної придатності IKEv2/IPsec, WireGuard і OpenVPN у self-hosted середовищі на базі VPS.

3.1 Загальна схема розгортання

У межах практичної реалізації послідовно налаштовуються IKEv2/IPsec, WireGuard та OpenVPN. Використання одного VPS для всіх варіантів дає змогу зберегти однакові інфраструктурні умови та надалі оцінювати відмінності між VPN-рішеннями без впливу різних серверних платформ.

Повні скрипти налаштування винесено в додатки. У межах основного розділу розглядаються ключові параметри розгортання: спосіб автентифікації, внутрішня адресація, службові порти, маршрутизація клієнтського трафіку, взаємодія VPN-тунелю з мережним стеком сервера та особливості підключення клієнтів.

Для всіх VPN-рішень використовується єдиний функціональний сценарій. Клієнтський пристрій встановлює з’єднання із VPN-сервером через мережу Інтернет, проходить автентифікацію та отримує внутрішню VPN-адресу. Після цього клієнтський трафік передається через захищений тунель, а на стороні VPS обробляється відповідно до конфігурації конкретного VPN-рішення та мережних параметрів сервера.

Незважаючи на спільні умови розгортання, досліджувані VPN-рішення мають різну конфігураційну модель. IKEv2/IPsec потребує налаштування strongSwan, параметрів IKE та IPsec, автентифікації клієнта і правил обробки IPsec-трафіку. WireGuard використовує модель на основі публічних ключів, віртуального мережного інтерфейсу та параметра AllowedIPs, який одночасно визначає доступні адреси й маршрутизацію через тунель. OpenVPN функціонує як окрема серверна служба, що використовує механізми SSL/TLS, серверний конфігураційний файл і клієнтські профілі підключення.

Під час опису кожного VPN-рішення основна увага приділяється не процесу встановлення пакетів, а структурі робочої конфігурації. Для кожного варіанта визначаються параметри автентифікації, внутрішній адресний простір, службові порти та особливості клієнтського підключення. Такий підхід дозволяє надалі порівнювати IKEv2/IPsec, WireGuard та OpenVPN за складністю розгортання, прозорістю конфігурації та зручністю адміністрування.

3.2 Конфігурація IKEv2/IPsec

Для реалізації IKEv2/IPsec на серверному вузлі використано strongSwan із конфігурацією через swanctl. Повний скрипт розгортання, включно з командами встановлення пакетів, створенням сертифікатів, налаштуванням XFRM-інтерфейсу, правилами фаєрвола і NAT, наведено в додатку А.

У конфігурації використовується IKEv2-з’єднання, у якому сервер приймає підключення від клієнтів із довільних зовнішніх адрес. На сервері створюється локальний сертифікат, підписаний власним центром сертифікації, а автентифікація серверної сторони виконується за відкритим ключем. Для клієнтської сторони використовується EAP-MSCHAPv2, що дозволяє виконувати автентифікацію користувача за обліковими даними. Така схема є зручною для сценарію remote access, оскільки клієнт перевіряє сертифікат сервера, а користувач проходить автентифікацію за логіном і паролем.

Для клієнтів визначено окремий внутрішній адресний простір

10.10.10.0/24, а як адреса VPN-шлюзу на сервері використовується 10.10.10.254. Після успішного встановлення з’єднання клієнт отримує адресу з цього пулу, а його трафік обробляється відповідно до параметрів IKEv2/IPsec-конфігурації. У ролі DNS-серверів для клієнтів задано зовнішні DNS-сервери Cloudflare (1.1.1.1) та Google Public DNS (8.8.8.8), що дозволяє забезпечити коректне розв’язання доменних імен після підключення до VPN.

Особливістю реалізації є використання XFRM-інтерфейсу ipsec0, пов’язаного з основним зовнішнім інтерфейсом VPS. Для цього в конфігурації задано ідентифікатор інтерфейсу if_id = 42, який використовується strongSwan для прив’язки IPsec-трафіку до відповідного XFRM-інтерфейсу. Після узгодження IKE SA та CHILD SA правила обробки трафіку передаються до підсистеми XFRM ядра Linux, яка забезпечує застосування політик IPsec. Такий підхід дозволяє працювати з IPsec трафіком як із трафіком окремого мережного інтерфейсу, що спрощує маршрутизацію, фільтрацію та застосування NAT.

Використання XFRM-інтерфейсу спрощує подальшу роботу з тунельним трафіком, оскільки для нього можна застосовувати стандартні механізми Linux, зокрема маршрутизацію, фільтрацію та NAT за інтерфейсом ipsec0. Такий підхід є зручнішим порівняно з класичною policy-based IPsec-конфігурацією, де трафік обробляється переважно через набір IPsec-політик без представлення тунелю як окремого мережного інтерфейсу.

Для роботи VPS як VPN-шлюзу в операційній системі вмикається пересилання IP пакетів. Це необхідно, оскільки сервер має не лише приймати зашифрований трафік від клієнта, а й передавати його до зовнішньої мережі через основний мережний інтерфейс. Додатково вимикаються окремі параметри, які можуть заважати проходженню тунельного трафіку, зокрема reverse path filtering. Це важливо для IPsec, де маршрутизація пакетів може відрізнятися від звичайної схеми обробки трафіку.

У конфігурації IKEv2 криптографічні параметри задаються окремо для службового обміну IKE та ESP-захисту користувацького трафіку. Після встановлення IKE SA формується дочірня асоціація безпеки CHILD SA, яка в цій

конфігурації працює в тунельному режимі та використовується для передавання користувацького IP-трафіку. Значення traffic selector 0.0.0.0/0 визначає, що під дію IPsec-політик може потрапляти весь IPv4-трафік клієнта, який відповідно до маршрутів спрямовується через VPN.

Оскільки клієнтам призначаються приватні адреси з внутрішнього VPN-пулу, для їхнього виходу до зовнішньої мережі застосовується NAT. На сервері налаштовується правило MASQUERADE, яке перетворює адресу джерела клієнтського трафіку в публічну IPv4-адресу VPS. Завдяки цьому відповіді із зовнішньої мережі повертаються на сервер, після чого передаються назад клієнту через IPsec-тунель.

Для забезпечення доступності IKEv2/IPsec на фаєрволі відкриваються UDP-порти 500 і 4500. Порт 500 використовується для початкового IKE обміну, а порт 4500 для NAT Traversal, який потрібний у випадках, коли клієнт перебуває за NAT. Окремо дозволяється маршрутизація трафіку між XFRM-інтерфейсом ipsec0 та зовнішнім інтерфейсом VPS. Це забезпечує проходження пакетів між VPN-клієнтом і зовнішньою мережею.

Після завершення конфігурації перевіряється стан служби strongSwan, завантаження параметрів через swanctl та створення захищених асоціацій під час клієнтського підключення. Успішна робота IKEv2/IPsec підтверджується встановленням IKE SA і CHILD SA, призначенням клієнту внутрішньої адреси з VPN-пулу, проходженням трафіку через інтерфейс ipsec0 та виходом клієнта до зовнішньої мережі через публічну адресу VPS.

Для перевірки працездатності IKEv2/IPsec-тунелю використовувалися команди swanctl --list-conns та swanctl --list-sas. Перша команда дозволяє перевірити завантажену конфігурацію strongSwan: тип з’єднання IKEv2, локальний і віддалений ідентифікатори, методи автентифікації, режим тунелю та селектори трафіку. У наведеному виводі (рис. 3.1) видно, що з’єднання ikev2-vpn використовує автентифікацію сервера за відкритим ключем і клієнтську автентифікацію EAP-MSCHAPv2, а для дочірнього тунелю net задано локальний селектор 0.0.0.0/0 і віддалений селектор dynamic. Команда swanctl --list-sas

підтверджує фактичне встановлення тунелю: для з'єднання ikev2-vpn створено IKE SA у стані ESTABLISHED, а дочірня асоціація CHILD SA має стан INSTALLED. Наявність лічильників переданих байтів і пакетів у напрямах in та out свідчить про проходження трафіку через IPsec-тунель.

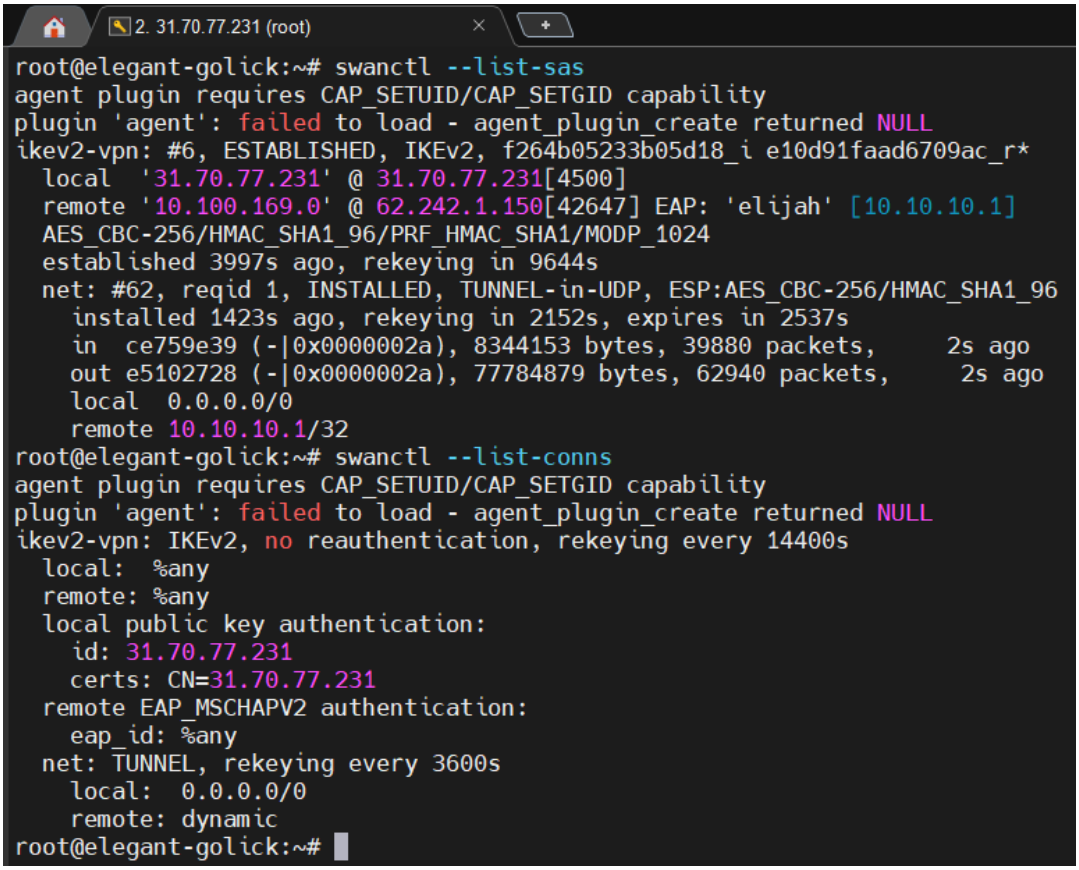

Рисунок 3.1 – Стан тунелю IPsec/IKEv2

Отже, конфігурація IKEv2/IPsec у цій роботі базується на strongSwan, сертифікатній автентифікації сервера, EAP-MSCHAPv2 для клієнта, XFRM-інтерфейсі ipsec0, внутрішньому адресному пулі 10.10.10.0/24 і NAT на зовнішньому інтерфейсі VPS. Така схема дозволяє реалізувати повноцінний сценарій віддаленого доступу з маршрутизацією клієнтського трафіку через IPsec-тунель.

3.3 Конфігурація WireGuard

Для реалізації WireGuard на серверному вузлі використано пакети WireGuard та WireGuard-tools. WireGuard використовує спрощену модель конфігурації та працює як віртуальний мережний інтерфейс операційної системи. У межах практичної реалізації створено інтерфейс wg0, через який передається захищений VPN-трафік між клієнтом і сервером. Повний скрипт розгортання WireGuard наведено в додатку Б.

Серверний інтерфейс WireGuard отримує адресу 10.8.0.1/24, а для клієнта визначено адресу 10.8.0.2/32. Така схема відповідає моделі віддаленого доступу, у якій клієнт підключається до VPN-сервера через мережу Інтернет, після чого його трафік маршрутизується через VPS. Для взаємодії між сторонами використовується UDP-порт 51820, який відкривається на фаєрволі сервера.

Автентифікація у WireGuard базується на парі криптографічних ключів. Під час конфігурації створюється окрема пара ключів для сервера та окрема пара ключів для клієнта. Приватний ключ зберігається локально на відповідній стороні, а публічний ключ додається до конфігурації іншої сторони. Таким чином, сервер приймає трафік лише від вузла, публічний ключ якого внесено до секції Peer.

У серверній конфігурації основними параметрами є адреса інтерфейсу wg0, порт прослуховування 51820, приватний ключ сервера та публічний ключ клієнта. Для клієнта в параметрі AllowedIPs на стороні сервера задано адресу 10.8.0.2/32. Це означає, що сервер пов’язує вказаний публічний ключ саме з цією внутрішньою VPN-адресою клієнта. У WireGuard параметр AllowedIPs виконує подвійну функцію: визначає допустимі адреси для конкретного вузла і одночасно впливає на правила маршрутизації через тунель.

Клієнтська конфігурація містить приватний ключ клієнта, внутрішню адресу 10.8.0.2/32, DNS-сервери Cloudflare (1.1.1.1) та Google Public DNS (8.8.8.8), а також параметри віддаленого вузла. У секції Peer задається публічний ключ сервера, зовнішня адреса VPN-сервера та параметр AllowedIPs = 0.0.0.0/0.

Таке значення означає, що весь IPv4-трафік клієнта спрямовується через WireGuard-тунель.

Для підтримання з’єднання в умовах NAT у клієнтській конфігурації використовується параметр PersistentKeepalive = 25. Він забезпечує періодичне надсилання службових пакетів до сервера, що допомагає зберігати активний запис трансляції адрес на проміжних NAT-пристроях. Це особливо важливо для клієнтів, які підключаються з домашніх або мобільних мереж.

Для роботи VPS як VPN-шлюзу в операційній системі вмикається пересилання IPv4-пакетів. Також вимикається механізм перевірки зворотного маршруту (rp_filter), що зменшує ризик відкидання пакетів у сценаріях, де трафік проходить через окремий тунельний інтерфейс. Після цього сервер може приймати трафік від клієнта через wg0 і передавати його до зовнішньої мережі через основний мережний інтерфейс.

Оскільки клієнт використовує приватну адресу з діапазону 10.8.0.0/24, для виходу до зовнішньої мережі застосовується NAT. У конфігурації WireGuard для цього використовуються директиви PostUp і PostDown. Після запуску інтерфейсу wg0 додаються правила iptables, які дозволяють передавання трафіку між wg0 та зовнішнім інтерфейсом VPS, а також виконують трансляцію адрес за допомогою правила MASQUERADE. Після зупинки інтерфейсу ці правила видаляються, що дозволяє підтримувати узгоджений стан конфігурації фаєрвола.

Запуск WireGuard виконується через службу wg-quick@wg0, яка зчитує параметри з файлу /etc/WireGuard/wg0.conf, створює інтерфейс wg0, призначає йому адресу та застосовує параметри конфігурації. Автоматичний запуск служби забезпечує відновлення VPN-інтерфейсу після перезавантаження сервера.

Працездатність з’єднання надалі перевіряється за станом інтерфейсу wg0, наявністю активного вузла, часом останнього обміну ключами, а також проходженням трафіку через тунель, стан інтерфейсу показано на рис. 3.2. Успішна робота підтверджується встановленням обміну між клієнтом і сервером, збільшенням лічильників переданих та отриманих даних, доступністю VPN-шлюзу 10.8.0.1 і виходом клієнта до зовнішньої мережі через публічну

адресу VPS.

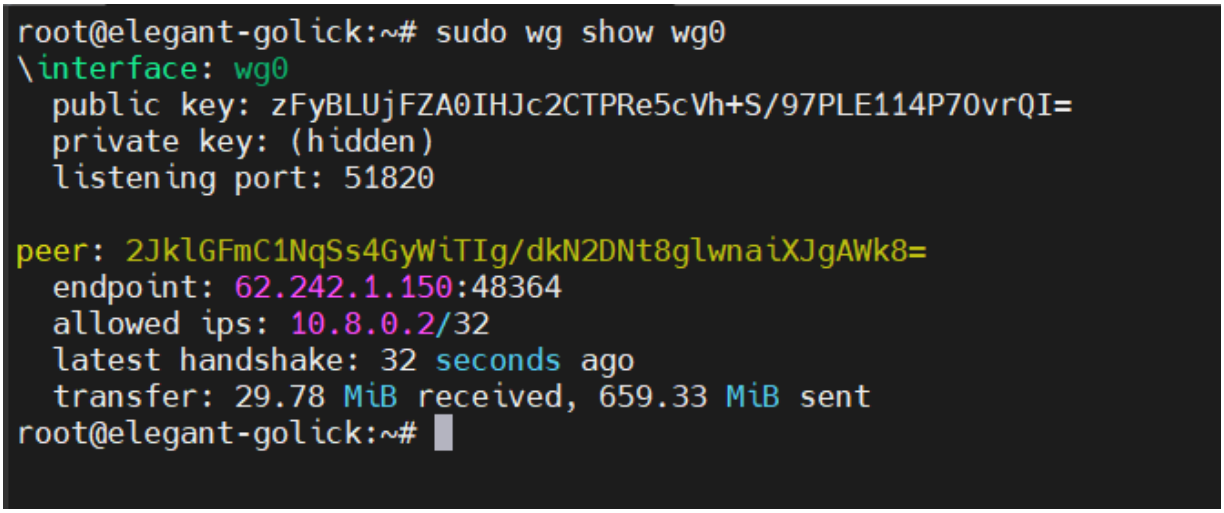

Рисунок 3.2 – Стан інтерфейсу wg0 після підключення клієнта

Функціонування VPN-сервера WireGuard забезпечується налаштуванням віртуального інтерфейсу wg0 із прив'язкою до відповідного UDP-порту та автентифікацією клієнтів за публічними ключами. Локальний сегмент тунелю розгорнуто у внутрішньому адресному просторі 10.8.0.0/24. Завдяки директиві AllowedIPs = 0.0.0.0/0 організовано повне тунелювання клієнтського IPv4-трафіку з його подальшою трансляцією (NAT) через зовнішній інтерфейс VPS. Описана схема є оптимальною для побудови простих і прозорих з'єднань віддаленого доступу.

3.4 Конфігурація OpenVPN

Для реалізації OpenVPN на серверному вузлі використано пакети openvpn та easy-rsa. На відміну від WireGuard, OpenVPN працює як окрема серверна служба і використовує модель SSL/TLS автентифікації з власною інфраструктурою відкритих ключів (PKI). Повний скрипт розгортання OpenVPN, включно зі створенням сертифікатів, серверної конфігурації, клієнтського профілю, правил NAT і фаєрвола, наведено в додатку В.

У межах конфігурації OpenVPN створюється тунельний інтерфейс tun0, призначений для передавання трафіку між клієнтом і сервером. Сервер працює за протоколом UDP на порту 1194, що є типовим варіантом для OpenVPN. Застосування UDP дозволяє зменшити накладні витрати на передавання трафіку та уникнути проблем, пов’язаних із тунелюванням TCP-трафіку всередині TCP-з’єднання.

Для внутрішньої адресації VPN-клієнтів використовується мережа 10.9.0.0/24. У серверній конфігурації параметр server визначає адресний простір тунелю та дозволяє серверу призначати клієнтам внутрішні VPN-адреси.

Автентифікація в архітектурі OpenVPN реалізується на основі сертифікатів. Для цього за допомогою інструментарію Easy-RSA розгортається локальний центр сертифікації (CA), який генерує необхідні сертифікати для сервера та клієнтів. На стороні VPN-сервера для перевірки автентичності та ініціалізації сесії використовуються файли ca.crt, server.crt, server.key, а також параметри Діффі-Геллмана (dh.pem). Відповідно, клієнтський конфігураційний профіль містить сертифікат клієнта, його приватний ключ та сертифікат центру сертифікації. Зазначена інфраструктура відкритих ключів (PKI) забезпечує надійну взаємну автентифікацію суб'єктів взаємодії на етапі встановлення TLS-з'єднання.

Конфігурація VPN-сервера містить криптографічні параметри для забезпечення конфіденційності та цілісності даних. Для перевірки цілісності повідомлень використовується алгоритм хешування SHA-256, а для шифрування даних AES-256-GCM. Через директиву data-ciphers також визначено перелік додаткових сумісних шифрів (AES-128-GCM, ChaCha20-Poly1305). Це дозволяє реалізувати механізм динамічного узгодження параметрів (NCP) між клієнтом і сервером безпосередньо під час встановлення з'єднання.

Для реалізації режиму повного тунелювання в серверній конфігурації задано директиву «push "redirect-gateway def1 bypass-dhcp"». Вона передає клієнту маршрутні параметри, за яких основний IPv4-трафік спрямовується через OpenVPN-тунель. Також клієнту передаються DNS-сервери Cloudflare

(1.1.1.1) та Google Public DNS (8.8.8.8).Для роботи VPS як VPN-шлюзу в операційній системі вмикається пересилання IPv4-пакетів. Також вимикається механізм перевірки зворотного маршруту rp_filter, що зменшує ризик відкидання пакетів у сценаріях, де трафік проходить через окремий тунельний інтерфейс. Після цього сервер може приймати трафік від клієнта через інтерфейс tun0 і передавати його до зовнішньої мережі через основний мережний інтерфейс.

Оскільки клієнтам призначаються приватні адреси з мережі 10.9.0.0/24, для виходу до зовнішньої мережі застосовується NAT. На сервері налаштовується правило MASQUERADE, яке транслює адресу джерела клієнтського трафіку в публічну IPv4-адресу VPS. Окремо додаються правила FORWARD, які дозволяють передавання трафіку між інтерфейсом tun0 та зовнішнім інтерфейсом сервера. Для збереження цих правил після перезапуску створюється окрема systemd-служба openvpn-nat.service.

На фаєрволі відкривається порт 1194/udp, необхідний для приймання клієнтських OpenVPN-підключень. Додатково дозволяється маршрутизований трафік між VPN інтерфейсом і зовнішнім інтерфейсом VPS. Це забезпечує проходження клієнтського трафіку до зовнішньої мережі.

Запуск OpenVPN виконується через службу openvpn-server@server, яка використовує конфігураційний файл /etc/openvpn/server/server.conf. Після запуску служби створюється інтерфейс tun0, активується серверний процес OpenVPN і стає можливим підключення клієнта за згенерованим профілем client1.ovpn. Клієнтський профіль містить адресу сервера, порт, протокол, сертифікати, приватний ключ клієнта та ключ tls-auth, що дозволяє використовувати його як самодостатній файл конфігурації підключення.

Конфігурація розгорнутого сервера OpenVPN містить параметри віртуального інтерфейсу tun0 та UDP-порту 1194, а ідентифікація користувачів здійснюється за допомогою сертифікатної автентифікації на базі інфраструктури Easy-RSA. Локальний сегмент мережі організовано в межах адресного простору 10.9.0.0/24. Реалізована схема забезпечує повне тунелювання IPv4-трафіку користувачів та його маршрутизацію в зовнішню мережу через механізм NAT на

стороні VPS. Такий підхід гарантує надійний віддалений доступ на основі стандартних TLS-протоколів OpenVPN.

4 МЕТОДИКА ЕКСПЕРИМЕНТАЛЬНОГО ТЕСТУВАННЯ

Методика експериментального тестування побудована на порівнянні чотирьох сценаріїв тестування: передавання трафіку без VPN, через IKEv2/IPsec, WireGuard та OpenVPN. Основна увага приділяється умовам вимірювання, параметрам тестових запусків, уніфікації MTU (Maximum Transmission Unit) та метрикам оцінювання якості з'єднань. Усі вимірювання виконуються для IPv4 трафіку, оскільки конфігурації VPN рішень, маршрутизація клієнтського трафіку та правила NAT у межах експерименту побудовані саме для IPv4. Такий підхід забезпечує оцінювання впливу кожного VPN рішення на досяжну пропускну здатність, стабільність передавання трафіку, джиттер і втрати пакетів.

Експеримент проводиться в реальних умовах мережі Інтернет. Тестовий трафік проходить через інфраструктуру комерційних провайдерів, проміжні маршрутизатори та магістральні мережні сегменти. Це наближує результати до практичного сценарію використання VPN і враховує зміну доступної пропускної здатності, коливання затримки та вплив проміжної мережної інфраструктури.

Для забезпечення порівнюваності результатів умови тестування уніфіковано для всіх VPN-рішень. У сценарії без VPN використовується стандартне значення MTU 1500 байт, яке відповідає типовому Ethernet-середовищу. Для VPN-сценаріїв значення MTU було уніфіковано на рівні 1420 байт для тунельних інтерфейсів, через які проходить клієнтський трафік. Для WireGuard це інтерфейс wg0, для OpenVPN – tun0, а для IKEv2/IPsec – XFRM-інтерфейс ipsec0, створений у межах route-based IPsec-конфігурації strongSwan.

Використання XFRM-інтерфейсу для IKEv2/IPsec обрано для наближення моделі IPsec до інтерфейсно-орієнтованої логіки WireGuard та OpenVPN. Це спрощує уніфікацію маршрутизації, фільтрації трафіку та MTU між різними VPN-рішеннями. Водночас слід враховувати, що IPsec у Linux може працювати і в policy-based режимі без окремого тунельного інтерфейсу, тому використання ipsec0 є особливістю саме обраної route-based конфігурації.

Для забезпечення порівнюваності результатів експерименту та мінімізації впливу відмінностей у параметрах тунельних інтерфейсів значення MTU для всіх досліджуваних VPN-рішень було уніфіковано на рівні 1420 байт: wg0 для WireGuard, tun0 для OpenVPN та ipsec0 для IKEv2/IPsec.

Вибір цього значення зумовлений наявністю додаткового службового навантаження, яке виникає під час інкапсуляції, шифрування та додавання протокольних заголовків у VPN-тунелі. За базового Ethernet MTU 1500 байт розрахунковий розмір внутрішнього MTU для досліджуваних рішень орієнтовно перебуває в діапазоні 1412-1446 байт залежно від конкретного протоколу, режиму інкапсуляції та криптографічних параметрів. Зокрема, для WireGuard службова інформація становить близько 76 байт, для OpenVPN у режимі tun поверх UDP приблизно 68–88 байт, а для IKEv2/IPsec з ESP та NAT-T близько 54–69 байт. Параметр MTU для інтерфейсів показано на рис. 4.1.

Отже, MTU 1420 байт було обрано як єдине значення, що забезпечує резерв для службових заголовків VPN-протоколів і водночас не призводить до надмірного зменшення корисного навантаження в одному пакеті.

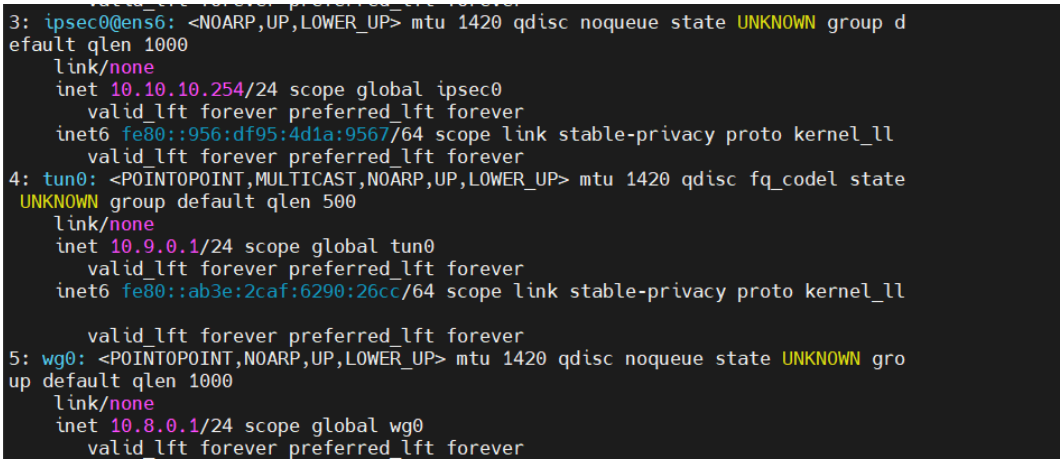

Рисунок 4.1 – Верифікація значень MTU VPS

Таке налаштування не розглядається як універсально оптимальне для будь-яких мережних умов, однак у межах експерименту воно дозволяє зменшити ймовірність IP-фрагментації та забезпечити коректніше порівняння пропускної

здатності, затримки, джитера й втрат пакетів.

Результати експерименту слід розглядати як характеристики конкретної тестової конфігурації. На продуктивність VPN впливають не лише протокольні накладні витрати, а й ресурси VPS, продуктивність CPU під час шифрування, реалізація VPN клієнта в конкретній операційній системі та поточний стан мережі. Використання одного серверного вузла для всіх VPN-рішень зменшує вплив різних серверних платформ на порівняння, однак не усуває повністю залежність результатів від апаратного та мережного середовища.

Для вимірювання характеристик з’єднань використовується утиліта Iperf3. Вона підтримує TCP і UDP тестування та доступна на всіх клієнтських операційних системах, задіяних у дослідженні. Це формує єдиний інструментальний підхід для Windows 11 та Linux Mint.

Утиліта Iperf3 є оптимальним галузевим стандартом. Вона повністю ізолює процес вимірювання від швидкості накопичувачів та специфіки файлових систем. Шляхом генерації синтетичного трафіку в оперативній пам'яті (RAM) інструмент дозволяє оцінити максимально можливу швидкість передавання даних для досліджуваного каналу [26].

Усі вимірювання виконуються в напрямку клієнт – сервер без використання reverse mode, щоб зберегти однакову схему генерації трафіку для всіх клієнтських платформ і VPN-сценаріїв. TCP-тестування проводиться в режимі чотирьох потоків. Такий режим обрано для масштабування обчислень, запобігаючи обмеженню параметрів мережі продуктивністю одного ядра. Параметри TCP congestion control, socket buffer size та інші низькорівневі параметри TCP стеку на клієнтських операційних системах додатково не змінювалися. Тестування виконувалося зі стандартними налаштуваннями відповідної ОС, що відповідає практичному сценарію використання VPN звичайним клієнтським пристроєм.

Для кожного сценарію тестування виконується серія зі 100 послідовних ітерацій. Така кількість повторів зменшує вплив випадкових коливань мережі, короткочасного перевантаження проміжних вузлів, фонової активності

провайдерів та інших неконтрольованих факторів, характерних для реального інтернет-середовища. Порядок виконання тестів періодично змінювався для зменшення впливу добових коливань навантаження мережі.

Кожна ітерація складається з активного вимірювання тривалістю 5 секунд і паузи тривалістю 5 секунд. Тривалість активного вимірювання обрана як компроміс між тривалістю одного тесту та необхідністю виконати достатню кількість повторів для кожного сценарію. Стабільність результатів оцінюється не за одним окремим запуском, а за серією зі 100 ітерацій. Пауза між ітераціями зменшує безпосередній вплив попереднього запуску на наступний, дає змогу завершити обробку попереднього потоку та зменшує ефект накопичення короткочасного навантаження.

Тестування поділяється на два типи вимірювань:

- TCP-тест використовується для оцінювання досяжної пропускної здатності з’єднання в заданих умовах експерименту. У цьому режимі Iperf3 формує TCP-потік, швидкість якого визначається роботою TCP-стеку та поточним станом каналу, зокрема алгоритмами керування перевантаженням, параметрами TCP-вікна, затримкою і втратами пакетів. У звітах Iperf3 фіксуються показники як на стороні відправника, так і на стороні отримувача, однак основним значенням для подальшого аналізу використовується пропускна здатність на стороні отримувача;
- UDP-тест використовується для оцінювання якості та стабільності каналу. Він запускається з параметрами -u -b 50M, тобто формує UDP-потік із фіксованою інтенсивністю 50 Мбіт/с. Це значення обрано як контрольоване навантаження, достатнє для імітації інтенсивного мультимедійного трафіку, але водночас нижче за очікувану пропускну здатність каналу без VPN. Завдяки цьому тест дає змогу оцінювати джиттер і втрати пакетів без навмисного перевантаження каналу самим генератором трафіку.

Значення 50 Мбіт/с розглядається як фіксований рівень навантаження для перевірки стабільності передавання. Якщо для окремого VPN-рішення фактична

досяжна пропускна здатність виявляється нижчою за це значення, результати UDP-тесту інтерпретуються як реакція тунелю на перевантаження, що також є важливою характеристикою практичної роботи VPN. Тестування виконувалося зі стандартними параметрами утиліти.

Вибір швидкості генерації UDP-потоку на рівні 50 Мбіт/с обумовлений необхідністю моделювання реального критичного навантаження на корпоративні або хмарні канали зв'язку. Такий показник відповідає одночасному високоінтенсивному використанню мережі: передачі потокового відео високої чіткості, проведенню групових відеоконференцій без втрати якості (VoIP/Telepresence) та фоновому обміну даними [27].

UDP тест використано для оцінювання параметрів передавання, які під час TCP-тесту можуть бути частково компенсовані механізмами транспортного рівня, зокрема повторною передачею втрачених сегментів, керуванням перевантаженням і регулюванням швидкості передавання. Оскільки UDP не виконує повторну передачу втрачених пакетів і не має вбудованого механізму керування перевантаженням, такий режим дозволяє визначити рівень втрат пакетів і джиттер. Це особливо важливо для аналізу тунелів у контексті трафіку реального часу, зокрема VoIP, відеоконференцій та інших мультимедійних сервісів, чутливих до затримок і втрат.

Основними метриками експерименту є пропускна здатність, джиттер і втрати пакетів. Пропускна здатність характеризує обсяг даних, який фактично передається через VPN-тунель за одиницю часу. Для VPN-рішень ця метрика відображає сумарний вплив шифрування, інкапсуляції, обробки пакетів у клієнтській і серверній операційних системах, а також накладних витрат конкретного протоколу.

Джиттер характеризує нерівномірність затримки між послідовними пакетами. Для інтерактивних і мультимедійних сервісів цей показник має не менше значення, ніж пропускна здатність, оскільки навіть за достатньої швидкості каналу значні коливання затримки можуть погіршувати якість голосового або відеозв’язку. У контексті VPN джиттер використовується для

оцінювання стабільності обробки трафіку в реальному часі.

Втрати пакетів відображають частку пакетів, які не були доставлені до отримувача. Для VPN-тунелів ця метрика є важливою, оскільки втрати можуть виникати не лише через якість фізичного або провайдерського каналу, а й через перевантаження тунелю, некоректний MTU, фрагментацію, обмеження CPU під час шифрування або особливості роботи конкретного VPN рішення. Низький рівень втрат свідчить про стабільність каналу та коректність налаштування інфраструктури.

За результатами 100 ітерацій для кожного сценарію розраховується середнє значення кожного показника. Це дозволяє зменшити вплив випадкових короткочасних відхилень і отримати узагальнену оцінку роботи VPN-рішення в межах заданої експериментальної конфігурації. Такий підхід є придатним для прикладного порівняння WireGuard, OpenVPN та IKEv2/IPsec за пропускною здатністю, джитером і рівнем втрат пакетів.

Для забезпечення відтворюваності експерименту основні параметри тестування зведено в таблицю 4.1.

Таблиця 4.1 – Основні параметри експериментального тестування

Параметр	Значення
Кількість ітерацій	100
Тривалість одного тесту	5 с
Пауза між тестами	5 с
TCP-тест	Iperf3 -P 4
UDP-тест	Iperf3 -u -b 50M -P 4
Метрики	throughput, jitter, packet loss
Сценарії	без VPN, IKEv2/IPsec, WireGuard, OpenVPN
MTU	однакове значення для всіх VPN-рішень

Обмеженням експерименту є використання одного VPS і визначеної кількості клієнтських середовищ, тому отримані числові значення не слід розглядати як універсальні для всіх хмарних платформ.

Таким чином, обрана методика забезпечує оцінювання VPN-рішень за трьома напрямами: максимально досяжною пропускною здатністю, стабільністю доставки пакетів і якістю передавання UDP-трафіку. Поєднання TCP та UDP

тестів, багаторазових ітерацій, пауз між вимірюваннями, чергування послідовності тестів та уніфікованого MTU формує порівнювані умови для IKEv2/IPsec, WireGuard та OpenVPN у реальному сценарії використання self-hosted VPN на базі VPS.

5 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У цьому розділі проаналізовано результати експериментального тестування VPN-рішень, отримані за методикою, описаною в попередньому розділі. Порівняння виконується для чотирьох сценаріїв тестування: передавання трафіку без VPN, через IKEv2/IPsec, WireGuard та OpenVPN.

Аналіз результатів проводиться за трьома основними групами показників: пропускна здатність TCP-з’єднання, джиттер UDP-трафіку та втрати UDP-пакетів. Пропускна здатність використовується для оцінювання ефективності передавання даних через VPN-тунель, тоді як джиттер і втрати пакетів характеризують стабільність каналу та якість передавання трафіку, чутливого до затримок. Такий поділ дозволяє оцінити VPN-рішення не лише за швидкістю, а й за якістю роботи в умовах реального мережного середовища.

Метою аналізу є не лише порівняння числових значень пропускної здатності, джиттера та втрат пакетів, а й визначення того, яке VPN-рішення є найбільш доцільним для розгортання в хмарній self-hosted інфраструктурі за сукупністю технічних і експлуатаційних критеріїв.

Узагальнені результати експериментального тестування наведено в таблиці 5.1.

Таблиця 5.1 – Узагальнені результати експериментального тестування VPN-рішень

Операційна система клієнта	Сценарій тестування	Пропускна здатність (Мбіт/с)	Джиттер (мс)	Втрата пакетів (%)
Windows 11	Без VPN	303,64	0,300	0,01
	WireGuard (wg0)	292,31	0,313	0,12
	OpenVPN (tun0)	100,41	0,197	7,04
	IKEv2/IPsec (ipsec0)	102,84	0,426	0,02
Linux Mint	Без VPN	322,73	0,203	0,04
	WireGuard (wg0)	285,52	0,217	0,35
	OpenVPN (tun0)	83,17	0,226	8,07
	IKEv2/IPsec (ipsec0)	251	0,194	0,35

Аналіз наведених результатів показує, що вплив VPN-тунелю на характеристики з’єднання істотно відрізняється залежно від обраного VPN-рішення та клієнтської операційної системи.

За показником TCP-пропускної здатності найкращі результати в обох середовищах демонструє WireGuard. У Linux Mint його пропускна здатність становить 285,52 Мбіт/с, у Windows 11 292,31 Мбіт/с, що значно перевищує показники конкурентів. IKEv2/IPsec займає другу позицію: 251 Мбіт/с у Linux Mint та 102,84 Мбіт/с у Windows 11. OpenVPN показує найнижчу пропускну здатність 83,17 Мбіт/с у Linux Mint і 100,41 Мбіт/с у Windows 11, що свідчить про суттєве зниження продуктивності. Такі результати можуть пояснюватися особливостями TLS-орієнтованої архітектури OpenVPN, роботою через TUN-інтерфейс, додатковою обробкою сертифікатів і загальними накладними витратами серверної служби.

Показники джиттера демонструють неоднозначну картину. У Linux Mint найнижчий джиттер зафіксовано для IKEv2/IPsec 0,194 мс, що є найкращим значенням серед усіх Linux-сценаріїв. WireGuard має джиттер 0,217 мс, OpenVPN 0,226 мс. У Windows 11, навпаки, найменший джиттер показав OpenVPN 0,20 мс, тоді як WireGuard 0,31 мс, а IKEv2/IPsec 0,426 мс. Таким чином, IKEv2/IPsec демонструє мінімальні коливання затримки в Linux, але в Windows його джиттер виявився найвищим, що свідчить про залежність стабільності затримок від клієнтської реалізації.

За показником втрат пакетів також спостерігаються суттєві відмінності. У Linux Mint WireGuard та IKEv2/IPsec мають однаково низький рівень втрат по 0,35 %, тоді як OpenVPN демонструє вкрай високі втрати 8,07 %. У Windows 11 найкращий результат за цим критерієм показав IKEv2/IPsec лише 0,02 %, далі йде WireGuard із 0,12 %, а OpenVPN знову має найгірший показник 7,04 %. Отже, OpenVPN у всіх сценаріях суттєво програє за стабільністю UDP-трафіку.

Порівняння результатів показує, що жодне VPN-рішення не є безумовно найкращим за всіма метриками. WireGuard лідирує за пропускною здатністю в обох ОС, однак у Linux Mint IKEv2/IPsec забезпечує нижчий джиттер при тих самих втратах, а у Windows 11 IKEv2/IPsec демонструє найнижчі втрати, хоча й поступається за джиттером. OpenVPN програє за швидкістю та втратами в обох середовищах, однак у Windows показав найменший джиттер.

Для вибору оптимального VPN-рішення було застосовано багатокритеріальний підхід, який враховує не лише пропускну здатність, а й якісні характеристики передавання трафіку. Пропускна здатність розглядалася як критерій, що має максимізуватися, тоді як джиттер і втрата пакетів як критерії, що мають мінімізуватися. Для порівнюваності показників було виконано нормування: для пропускної здатності найвище значення отримувало оцінку 1, найнижче 0; для джиттера та втрат застосовано обернену логіку (меншому значенню відповідає вища оцінка).

Оцінку розраховано з урахуванням вагових коефіцієнтів: 0,35 для пропускної здатності, 0,20 для джиттера та 0,45 для втрати пакетів. Найбільшу вагу надано втраті пакетів, оскільки цей показник безпосередньо впливає на стабільність передавання трафіку через VPN-тунель.

За результатами нормалізації для Linux Mint найвищу оцінку отримав WireGuard 0,788. Це зумовлено поєднанням найвищої пропускної здатності, дуже низького джиттера та мінімальних втрат пакетів (0,35%). IKEv2/IPsec отримав оцінку 0,766, він має дещо нижчу пропускну здатність порівняно з WireGuard, що знизило його підсумковий бал. OpenVPN отримав оцінку 0,102 через найнижчу пропускну здатність та найвищі втрати пакетів (8,07 %).

У середовищі Windows 11 найвищу оцінку також отримав WireGuard 0,845. Його перевага забезпечена найбільшою пропускною здатністю та дуже низькими втратами (0,12%), хоча джиттер у нього не найменший. IKEv2/IPsec отримав оцінку 0,572 попри абсолютний мінімум втрат (0,02 %), він суттєво поступився за пропускною здатністю та показав найвищий джиттер. OpenVPN отримав 0,228, що відображає його слабкі сторони за швидкістю та втратами, незважаючи на найнижчий джиттер.

Отримані результати характеризують продуктивність VPN-рішень у межах конкретного тестового середовища. На значення пропускної здатності, джиттера та втрати пакетів могли впливати параметри VPS, завантаження процесора під час шифрування, проходження трафіку через мережу Інтернет, особливості клієнтської операційної системи, драйверів мережних інтерфейсів і поточний стан каналу зв’язку. Тому результати не слід розглядати як універсальну абсолютну характеристику протоколів, але вони дозволяють оцінити їхню ефективність у досліджуваній self-hosted cloud VPN-інфраструктурі.

Таким чином, багатокритеріальна оцінка показує, що WireGuard є найкращим рішенням як для Linux Mint, так і для Windows 11 за заданих вагових коефіцієнтів (0,35; 0,2; 0,45). Його висока пропускна здатність і низькі втрати пакетів забезпечують стійку перевагу над IKEv2/IPsec та OpenVPN в обох середовищах. IKEv2/IPsec показує конкурентоспроможні результати в Linux (0,766), але поступається WireGuard в обох ОС. OpenVPN є найменш ефективним через суттєві втрати пакетів і низьку швидкість.

ВИСНОВКИ

У кваліфікаційній роботі досліджено побудову технології VPN у хмарному середовищі. Розглянуто принципи функціонування VPN, архітектурні моделі VPN-з’єднань, підходи до розгортання VPN-інфраструктури та особливості сучасних VPN-протоколів.

У результаті аналізу моделей розгортання встановлено, що для дослідження доцільною є модель self-hosted cloud VPN. Вона дозволяє розгорнути VPN-шлюз на базі VPS і зберегти контроль над операційною системою, VPN-програмним забезпеченням, маршрутизацією, NAT, MTU та правилами фаєрвола. Такий підхід є гнучкішим за cloud-managed VPN, хоча потребує більшого обсягу адміністрування.

У роботі проаналізовано три VPN-рішення: IKEv2/IPsec, WireGuard та OpenVPN. IKEv2/IPsec є стандартизованим і гнучким рішенням, але потребує складнішого налаштування. WireGuard має просту конфігураційну модель і високу ефективність обробки трафіку. OpenVPN є зрілим TLS-орієнтованим рішенням, однак має більші накладні витрати та потребує розгортання PKI і клієнтських профілів.

У практичній частині на VPS було послідовно розгорнуто IKEv2/IPsec на базі strongSwan, WireGuard та OpenVPN. Для кожного рішення визначено внутрішню адресацію, спосіб автентифікації, службові порти, правила маршрутизації, NAT і параметри клієнтського підключення. Повні конфігураційні скрипти винесено в додатки.

Для перевірки роботи VPN-рішень розроблено методику експериментального тестування. Вимірювання виконувалися для чотирьох сценаріїв: без VPN, через IKEv2/IPsec, WireGuard та OpenVPN. Основними метриками були пропускна здатність TCP-з’єднання, джиттер UDP-трафіку та втрати UDP-пакетів. Для вимірювань використано Iperf3, а для забезпечення порівнюваності результатів було уніфіковано MTU для VPN-сценаріїв.

Результати експерименту показали, що VPN-тунелювання впливає на характеристики з’єднання через накладні витрати на інкапсуляцію, шифрування, маршрутизацію та обробку трафіку. Найвищу пропускну здатність серед досліджуваних VPN-рішень продемонстрував WireGuard. OpenVPN показав нижчі результати за швидкістю, а IKEv2/IPsec зайняв проміжну позицію, зберігаючи переваги стандартизованості та гнучкості конфігурації.

Аналіз джиттера та втрат пакетів підтвердив, що вибір VPN-рішення не може ґрунтуватися лише на пропускній здатності. Для практичного застосування важливими є також стабільність передавання, рівень втрат, складність розгортання та зручність адміністрування. Тому вибір оптимального VPN-рішення доцільно виконувати за багатокритеріальним підходом.

За сукупністю отриманих результатів WireGuard можна вважати найбільш доцільним рішенням для self-hosted VPN-інфраструктури на базі VPS у сценаріях, де пріоритетними є висока пропускна здатність, проста конфігурація та низька складність адміністрування. IKEv2/IPsec доцільний у середовищах, де важливі стандартизованість і гнучке керування параметрами IPsec. OpenVPN залишається придатним варіантом за потреби широкої підтримки клієнтів і TLS-орієнтованої моделі автентифікації.

Практичне значення роботи полягає в розробленні та перевірці підходу до побудови self-hosted VPN-інфраструктури в хмарному середовищі. Отримані результати можуть бути використані під час вибору VPN-рішення для організації захищеного віддаленого доступу на базі VPS.

ПЕРЕЛІК ДЖЕРЕЛ

1. Тараненко І. О. Конфігурація приватного VPN-серверу // 30-й Міжнародний молодіжний форум «Радіоелектроніка та інновації». Харків : ХНУРЕ, 2026. С. 143–144.

2. Guide to IPsec VPNs / E. Barker та ін. Gaithersburg, MD : National Institute of Standards and Technology, 2020. URL: https://doi.org/10.6028/nist.sp.800-77r1 (дата звернення: 15.05.2026).

3. Guide to SSL VPNs / S. E. Frankel та ін. Gaithersburg, MD : National Institute of Standards and Technology, 2008. URL: https://doi.org/10.6028/nist.sp.800-113 (дата звернення: 15.05.2026).

4. Generic Routing Encapsulation (GRE) / S. Hanks та ін. RFC Editor, 1994. URL: https://doi.org/10.17487/rfc1701 (дата звернення: 15.05.2026).

5. Khan A. On-Prem vs Cloud-Hosted VPN Servers: Which Is Right for Your IT Team?. LinkedIn. URL: https://www.linkedin.com/pulse/on-prem-vs-cloud-hosted-vpn-servers-which-right-your-team-awais-khan-18e3f (дата звернення: 15.05.2026).

6. AWS VPN pricing. URL: https://aws.amazon.com/vpn/pricing/ (дата звернення: 15.05.2026).

7. Cloud VPN. Google Cloud. URL: https://cloud.google.com/vpn/pricing (дата звернення: 15.05.2026).

8. VPN Gateway pricing. Microsoft Azure: Cloud Computing Services. URL: https://azure.microsoft.com/en-us/pricing/details/vpn-gateway/ (дата звернення: 15.05.2026).

9. Best Self-Hosted Open-Source VPN Solutions | Colonel Server. Colonel Server - Infrastructure Server Hosting. URL: https://colonelserver.com/blog/open-source-vpn/ (дата звернення: 15.05.2026).

10. Kent S., Seo K. Security Architecture for the Internet Protocol. RFC 4301.

Internet Engineering Task Force, 2005. URL: **https://www.rfc-editor.org/rfc/rfc4301** (дата звернення: 15.05.2026).

11. Kaufman C. та ін. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296. Internet Engineering Task Force, 2014. URL: **https://www.rfc-editor.org/rfc/rfc7296** (дата звернення: 15.05.2026).

12. WireGuard: fast, modern, secure VPN tunnel. WireGuard. URL: https://www.WireGuard.com/#conceptual-overview (дата звернення: 15.05.2026).

13. Donenfeld J. WireGuard: Next Generation Kernel Network Tunnel. WireGuard. URL: https://www.WireGuard.com/papers/WireGuard.pdf (дата звернення: 15.05.2026).

14. Quick Start - WireGuard. WireGuard. URL: https://www.WireGuard.com/quickstart (дата звернення: 15.05.2026).

15. Protocol & Cryptography - WireGuard. WireGuard. URL: https://www.WireGuard.com/protocol/ (дата звернення: 15.05.2026).

16. OpenVPN 2.6 Manual. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/community-articles/openvpn-2-6-manual.html (дата звернення: 15.05.2026).

17. OpenVPN Cryptographic Layer. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/openvpn-cryptographic-layer.html (дата звернення: 15.05.2026).

18. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3 : RFC 8446 / Internet Engineering Task Force. 2018. URL: https://www.rfc-editor.org/rfc/rfc8446 (дата звернення: 15.05.2026).

19. Routing All Client Traffic (Including Web Traffic) Through the VPN. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/routing-all-client-traffic--including-web-traffic--through-the-vpn.html (дата звернення: 15.05.2026).

20. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile : RFC 5280 / D. Cooper та ін. ; Internet Engineering

Task Force. 2008. URL: https://www.rfc-editor.org/rfc/rfc5280 (дата звернення: 15.05.2026).

21. Setting Up Your Own Certificate Authority (CA) and Generating Certificates and Keys for an OpenVPN Server and Multiple Clients. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/setting-up-your-own-certificate-authority--ca--and-generating-certificates-and-keys-for-an-openvpn-server-and-multiple-clients.html (дата звернення: 15.05.2026).

22. Using Alternative Authentication Methods. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/using-alternative-authentication-methods.html (дата звернення: 15.05.2026).

23. Hardening OpenVPN Security. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/community-docs/hardening-openvpn-security.html (дата звернення: 15.05.2026).

24. Troubleshooting Guide for Access Server. Secure VPN Solutions for Business & Remote Access | OpenVPN. URL: https://openvpn.net/as-docs/troubleshooting.html (дата звернення: 15.05.2026).

25. OpenVPN Data Channel Offload (DCO). *Secure VPN Solutions for Business & Remote Access | OpenVPN*. URL: https://openvpn.net/as-docs/openvpn-data-channel-offload.html (дата звернення: 16.05.2026).

26. Assessment of iPerf as a tool for LAN throughput prediction / A. H. Al-Asadi, H. M. Al-Sabbagh, A. S. Al-Khayyat et al. International Journal of Engineering Technology (IJET). 2023. Vol. 12, No. 4. P. 154–165.

27. Bitstream-based Model Standard for 4K/UHD: ITU-T P.1204.3 – Model Details, Evaluation, Analysis, and Open Source Implementation / R. Rao, S. Göring, P. List та ін. IEEE Communications Magazine. 2020. Т. 58, № 6. С. 88–94.