НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ, МЕРЕЖ ТА КІБЕРБЕЗПЕКИ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри**
Інформаційних технологій	комп'ютерних систем, мереж та кібербезпеки
(назва факультету)	(назва кафедри)
_____ Ігор БОЛБОТ	_____ Дмитро КАСАТКІН
(підпис) (ім'я ПРІЗВИЩЕ)	(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025_ р.	"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему "ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ПРОТОКОЛІВ VPN ДЛЯ ПОБУДОВИ РОЗПОДІЛЕНИХ ЗАХИЩЕНИХ АВТОНОМНИХ СИСТЕМ"

Спеціальність F7 Комп'ютерна інженерія
(код і найменування)

Освітня програма Комп'ютерні системи і мережі
(назва)

Орієнтація освітньої програми - освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

д.т.н., доцент _____ _____ Вадим ШКАРУПИЛО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

д.пед.н., професор _____ _ Сергій МАМЧЕНКО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ Максим ЛІФЕР
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
_к.пед.н., доцент Дмитро КАСАТКІН
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"____" _____2025року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

_____ЛІФЕРУ Максиму Олександровичу _____
(прізвище, ім'я, по батькові)

Спеціальність F7 Комп'ютерна інженерія _____
(код і найменування)

Освітня програма _Комп'ютерні системи і мережі _____
(назва)

Орієнтація освітньої програми освітньо-професійна _____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи "ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ПРОТОКОЛІВ VPN ДЛЯ ПОБУДОВИ РОЗПОДІЛЕНИХ ЗАХИЩЕНИХ АВТОНОМНИХ СИСТЕМ"

затверджена наказом від 29.10.2024р. № 1941С

Термін подання завершеної роботи на кафедру 2025.11.12
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____
_____
_____

Перелік питань, що підлягають дослідженню:

1. Автономні системи як основа глобальних мереж. Розподілені системи та мережі.
2. Використання технології VPN для побудови захищених тунелів та каналів передачі даних.
3. Протоколи підтримки тунелювання та VPN при створенні розподілених автономних систем глобальних мереж

Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_10_"_лютого 2025 р.

Керівник магістерської кваліфікаційної роботи _____ Сергій МАМЧЕНКО
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ _____Максим ЛІФЕР
(ім'я ПРІЗВИЩЕ)

ЗМІСТ

ВСТУП

Сучасний розвиток інформаційних технологій супроводжується стрімким зростанням обсягів даних, що передаються через мережі, а також підвищенням вимог до безпеки та конфіденційності інформації. У контексті глобалізації та цифровізації економіки, розподілені системи стають ключовим елементом інфраструктури організацій, що забезпечують ефективну взаємодію між географічно віддаленими підрозділами. Такі системи, відомі як розподілені захищені автономні системи, потребують надійних механізмів захисту даних від несанкціонованого доступу, перехоплення та інших кіберзагроз. Одним із найпоширеніших інструментів для забезпечення безпеки таких систем є віртуальні приватні мережі (VPN), які дозволяють створювати захищені канали зв'язку через загальнодоступні мережі, зокрема Інтернет.

Актуальність теми магістерської роботи зумовлена зростанням кіберзагроз, що супроводжують цифрову трансформацію суспільства, та необхідністю розробки ефективних рішень для захисту розподілених систем. VPN-протоколи, такі як PPTP, L2TP/IPsec, OpenVPN, WireGuard та інші, відіграють важливу роль у забезпеченні конфіденційності, цілісності та автентичності даних. Проте різноманітність протоколів, їх функціональних можливостей, переваг і недоліків створює потребу в детальному дослідженні їхнього впливу на продуктивність, безпеку та масштабованість автономних систем. Особливо актуальним є вивчення використання VPN у контексті розподілених систем, де необхідно забезпечити не лише захист даних, але й ефективну взаємодію між компонентами системи.

Мета роботи полягає в дослідженні використання протоколів VPN для побудови розподілених захищених автономних систем, аналізі їхніх можливостей, переваг і недоліків, а також розробці рекомендацій щодо їх оптимального застосування в реальних сценаріях.

Завдання дослідження:

1. Визначити та проаналізувати поняття розподілених захищених автономних систем.

2. Дослідити основні протоколи VPN (тунелювання, криптозахисту, авторизації та аутентифікації, мережеві протоколи, шифрозахист IP-адрес) та їхні характеристики.
3. Розробити методику оцінки ефективності VPN-протоколів у розподілених системах.
4. Провести практичне тестування VPN-протоколів у модельованому середовищі.
5. Сформулювати рекомендації щодо вибору та впровадження VPN-протоколів для забезпечення безпеки та продуктивності розподілених автономних систем.

Об’єкт дослідження – розподілені захищені автономні системи, які функціонують у сучасних інформаційно-комунікаційних середовищах.

Предмет дослідження – процеси побудови розподілених захищених автономних систем за допомогою протоколів VPN, а також аналіз їхніх функціональних можливостей, переваг і недоліків.

Методи дослідження включають теоретичний аналіз літературних джерел, порівняльне оцінювання протоколів VPN, моделювання розподілених систем, експериментальне тестування та аналіз отриманих результатів. Теоретична частина роботи базується на вивченні сучасних наукових публікацій, стандартів і практичних рішень у сфері VPN-технологій та розподілених систем. Практична частина передбачає створення тестового середовища для оцінки продуктивності та безпеки VPN-протоколів.

Наукова новизна роботи полягає в систематизації підходів до використання VPN-протоколів у розподілених захищених автономних системах, розробці методики їх оцінки за критеріями безпеки, продуктивності та масштабованості, а також у формуванні практичних рекомендацій для їх впровадження.

Практична цінність дослідження полягає в можливості використання отриманих результатів для проектування та впровадження захищених розподілених систем у різних галузях, таких як інформаційні технології, фінансовий сектор, державне управління та промисловість. Рекомендації,

розроблені в ході дослідження, можуть бути застосовані для оптимізації вибору VPN-протоколів залежно від специфіки завдань і вимог до системи.

Структура роботи складається зі вступу, трьох основних розділів, висновків та списку використаних джерел. У першому розділі розглядаються теоретичні основи розподілених захищених автономних систем і протоколів VPN. Другий розділ присвячений методиці дослідження, включаючи критерії оцінки та порівняння протоколів. У третьому розділі представлено практичні результати тестування та рекомендації щодо використання VPN-протоколів. У висновках узагальнено результати дослідження та визначено перспективи подальших досліджень.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗПОДІЛЕНИХ ЗАХИЩЕНИХ АВТОНОМНИХ СИСТЕМ ТА ПРОТОКОЛІВ VPN

1.1 Поняття автономних систем та їх захист

Автономні системи (АС) є фундаментальним елементом сучасних інформаційно-комунікаційних мереж, які забезпечують функціонування складних інфраструктур у глобалізованому цифровому світі. Під автономною системою розуміють сукупність мережевих пристроїв, таких як маршрутизатори, комутатори та сервери, які об’єднані під єдиним адміністративним управлінням і використовують спільну політику маршрутизації. Кожна автономна система ідентифікується унікальним номером (AS Number), який застосовується в протоколі BGP (Border Gateway Protocol) для обміну інформацією про маршрути між різними мережами. Такі системи можуть включати корпоративні мережі, мережі інтернет-провайдерів, хмарні платформи або навіть спеціалізовані системи для промислових чи наукових потреб [1]. Автономність дозволяє системі функціонувати незалежно від зовнішніх впливів, зберігаючи можливість інтеграції з іншими мережами через стандартизовані протоколи.

Основною особливістю автономних систем є їх здатність до самостійного управління та виконання завдань без необхідності постійного зовнішнього втручання. Це досягається завдяки чітко визначеним політикам маршрутизації, які забезпечують стабільність і передбачуваність роботи системи. Наприклад, у мережах інтернет-провайдерів автономні системи відповідають за маршрутизацію трафіку між різними регіонами, що дозволяє оптимізувати передачу даних і зменшувати затримки. Однак автономність також створює виклики, пов’язані з безпекою, оскільки такі системи часто стають мішенню для кібератак, таких як DDoS-атаки, перехоплення даних або спроби несанкціонованого доступу [2].

Захист автономних систем є комплексним завданням, яке охоплює кілька ключових аспектів: забезпечення конфіденційності, цілісності та доступності даних. Конфіденційність гарантує, що інформація доступна лише авторизованим користувачам, цілісність захищає дані від несанкціонованої модифікації, а

доступність забезпечує безперебійну роботу системи навіть у разі атак. Для реалізації цих цілей використовуються криптографічні методи, такі як алгоритм AES (Advanced Encryption Standard), який забезпечує надійне шифрування даних, що передаються через мережу [3]. Крім того, важливим елементом захисту є впровадження систем виявлення вторгнень (IDS) і систем запобігання вторгненням (IPS), які дозволяють моніторити мережевий трафік і реагувати на потенційні загрози в реальному часі.

Сегментація мережі є ще одним важливим інструментом захисту автономних систем. Вона дозволяє ізолювати критичні компоненти системи, такі як сервери з конфіденційними даними, від менш захищених сегментів, що знижує ризик поширення загроз у разі компрометації одного з вузлів. Наприклад, використання віртуальних локальних мереж (VLAN) або програмно-визначених мереж (SDN) створює ізольовані мережеві сегменти, що підвищує загальний рівень безпеки [3]. SDN, зокрема, дозволяє динамічно керувати мережею, адаптуючи її до змінних умов і загроз. Наприклад, рішення на основі SDN від Cisco дозволяють автоматично перенаправляти трафік у разі виявлення аномалій, що забезпечує безперервність роботи системи.

Крім того, захист автономних систем включає механізми резервного копіювання та відновлення даних, які є критично важливими для забезпечення безперервності роботи в разі апаратних збоїв або кібератак. Наприклад, регулярне створення резервних копій і їх зберігання в ізольованому середовищі дозволяє швидко відновити систему після атаки типу ransomware. Також важливим є правильне налаштування мережевих пристроїв, таких як брандмауери нового покоління (NGFW) і системи управління загрозами (UTM), які забезпечують комплексний захист від зовнішніх і внутрішніх загроз [3]. Наприклад, рішення від Juniper Networks дозволяють інтегрувати брандмауери з системами моніторингу, що підвищує ефективність виявлення атак.

У контексті розподілених систем захист автономних систем набуває додаткової складності через географічну розподіленість компонентів. У таких випадках використовуються технології VPN для створення захищених каналів

зв’язку між віддаленими вузлами, що забезпечує безпечну передачу даних через незахищені мережі, такі як Інтернет [5]. VPN дозволяє ізолювати трафік системи від зовнішнього середовища, використовуючи тунелювання та шифрування, що є особливо важливим для корпоративних мереж, які об’єднують офіси в різних країнах.

Автономні системи є складними структурами, які потребують комплексного підходу до захисту. Використання криптографічних методів, сегментації мережі, систем моніторингу та VPN-технологій дозволяє забезпечити високий рівень безпеки. Подальший аналіз у цій роботі буде зосереджений на ролі VPN у захисті розподілених автономних систем.

1.2 Характеристики розподілених систем та їх особливості

Розподілені системи є важливим класом інформаційних систем, які складаються з множини автономних обчислювальних вузлів, що взаємодіють між собою через мережу для досягнення спільної мети. Такі системи широко застосовуються в хмарних обчисленнях, розподілених базах даних, корпоративних мережах, системах Інтернету речей (IoT) та інших сферах, де потрібна висока масштабованість, надійність і гнучкість. Основною особливістю розподілених систем є їх географічна та функціональна розподіленість, що дозволяє компонентам працювати незалежно, але в координації через мережеві протоколи [6].

Однією з ключових характеристик розподілених систем є їх масштабованість, яка дозволяє системі адаптуватися до зростання обсягів даних або кількості користувачів. Наприклад, архітектури на основі мікросервісів дозволяють масштабувати окремі компоненти системи незалежно один від одного, що забезпечує ефективне використання ресурсів. Такі системи, як Kubernetes або Docker Swarm, дозволяють автоматично додавати нові вузли залежно від навантаження, що є критично важливим для хмарних платформ, таких як AWS або Google Cloud [7]. Масштабованість також сприяє підвищенню продуктивності, оскільки система може розподіляти обчислювальні завдання між кількома вузлами.

Відмовостійкість є ще однією важливою характеристикою розподілених систем. Вони проектуються таким чином, щоб вихід з ладу одного або кількох вузлів не призводив до зупинки всієї системи. Для цього використовуються механізми реплікації даних, балансування навантаження та резервного копіювання. Наприклад, у розподілених базах даних, таких як Cassandra або MongoDB, дані дублюються на кількох вузлах, що забезпечує їх доступність навіть у разі апаратного збою [8]. Балансувальники навантаження, такі як NGINX або HAProxy, дозволяють розподіляти запити між доступними вузлами, що підвищує надійність системи.

Асинхронна взаємодія між компонентами є ще однією особливістю розподілених систем. Вузли системи можуть обробляти запити незалежно один від одного, що підвищує швидкість обробки даних, але створює виклики в синхронізації та узгодженості даних. Для вирішення цих проблем застосовуються спеціалізовані алгоритми, такі як Paxos або Raft, які забезпечують узгодженість даних у розподілених системах [9]. Наприклад, алгоритм Raft використовується в системах, таких як Consul або etcd, для забезпечення консистентності даних у реальному часі.

Безпека є критично важливою характеристикою розподілених систем, особливо коли вузли розташовані в різних географічних регіонах. У таких випадках використовуються VPN для створення захищених каналів зв’язку, що забезпечують конфіденційність і цілісність даних. Наприклад, протоколи VPN, такі як OpenVPN або WireGuard, дозволяють шифрувати трафік між вузлами, що передається через незахищені мережі, такі як Інтернет [10]. Однак використання VPN може впливати на продуктивність системи через додаткові накладні витрати на шифрування та тунелювання, що потребує ретельного вибору протоколу.

Складність управління та моніторингу є ще одним викликом розподілених систем. Оскільки система складається з багатьох вузлів, необхідно використовувати спеціалізовані інструменти для моніторингу стану мережі, виявлення аномалій і забезпечення безпеки. Наприклад, системи моніторингу, такі як Prometheus або Grafana, дозволяють відстежувати продуктивність і виявляти

проблеми в реальному часі [7]. Крім того, розподілені системи потребують автоматизованих інструментів управління, таких як Ansible або Terraform, для спрощення конфігурації та оновлення вузлів.

Таблиця 1.1 – Основні характеристики розподілених систем

Характеристика	Опис	Приклади технологій
Масштабованість	Додавання вузлів без втрати продуктивності	Kubernetes, Docker Swarm
Відмовостійкість	Стійкість до збоїв окремих вузлів	Cassandra, MongoDB, HAProxy
Асинхронна взаємодія	Незалежна обробка запитів	Paxos, Raft, Consul
Безпека	Захист даних через шифрування та VPN	OpenVPN, WireGuard
Управління	Автоматизація конфігурації та моніторингу	Prometheus, Grafana, Ansible

Розподілені системи поєднують високу масштабованість, відмовостійкість і гнучкість, але потребують ретельного підходу до забезпечення безпеки та управління. Використання VPN-технологій є ключовим інструментом для захисту таких систем, особливо в умовах географічної розподіленості.

1.3 Роль протоколів VPN у побудові розподілених захищених систем

Віртуальні приватні мережі (VPN) є ключовим інструментом для забезпечення безпеки розподілених систем, дозволяючи створювати захищені канали зв’язку між географічно віддаленими вузлами через загальнодоступні мережі, такі як Інтернет. Основною функцією VPN є забезпечення конфіденційності, цілісності та автентичності даних, що передаються між компонентами системи, що є критично важливим для розподілених захищених автономних систем [11]. VPN дозволяє ізолювати трафік системи від зовнішнього

середовища, створюючи віртуальні тунелі, які захищають дані від перехоплення та модифікації.

Однією з основних ролей VPN у розподілених системах є захист даних від атак типу "людина посередині" (Man-in-the-Middle, MITM). Використання криптографічних алгоритмів, таких як AES-256, дозволяє шифрувати дані, що передаються через незахищені мережі, забезпечуючи їх конфіденційність [12]. Наприклад, протоколи, такі як IPsec або OpenVPN, створюють безпечні тунелі, які дозволяють безпечно передавати конфіденційні дані між офісами компанії, хмарними серверами або віддаленими користувачами. Це особливо важливо для організацій, які працюють із чутливими даними, такими як фінансові транзакції або персональна інформація.

VPN також відіграє важливу роль у забезпеченні аутентифікації та авторизації. Наприклад, протоколи, такі як IPsec, підтримують використання попередньо спільних ключів (PSK) або цифрових сертифікатів, що дозволяє перевіряти ідентичність користувачів і пристроїв перед наданням доступу до системи [13]. Двофакторна аутентифікація (2FA), підтримуваний OpenVPN, додатково підвищує безпеку, вимагаючи від користувачів надання додаткового фактора, наприклад, одноразового пароля (OTP). Це зменшує ризик несанкціонованого доступу, що є особливо актуальним для розподілених систем із великою кількістю користувачів.

Ще однією важливою функцією VPN є забезпечення гнучкості в управлінні розподіленими системами. VPN дозволяє створювати динамічні підключення між новими вузлами без необхідності фізичного переобладнання мережі. Наприклад, у хмарних платформах, таких як AWS або Azure, VPN використовується для створення захищених з’єднань між віртуальними машинами або контейнерами, що дозволяє швидко масштабувати систему [14]. Крім того, інтеграція VPN із програмно-визначеними мережами (SD-WAN) оптимізує маршрутизацію трафіку, зменшуючи затримки та підвищуючи продуктивність.

Проте використання VPN у розподілених системах пов’язане з певними викликами, такими як зниження продуктивності через накладні витрати на

шифрування та тунелювання. Для вирішення цієї проблеми сучасні протоколи, такі як WireGuard, використовують оптимізовані криптографічні алгоритми, такі як ChaCha20, які забезпечують високу швидкість і низькі затримки [15]. Це робить WireGuard ідеальним вибором для систем, де продуктивність є критично важливою, наприклад, у реальному часі обробки даних.

Протоколи VPN є невід'ємною частиною розподілених захищених систем, забезпечуючи безпечний обмін даними, гнучкість у підключенні нових вузлів і захист від кіберзагроз. Їх правильне використання дозволяє створювати надійні та ефективні автономні системи, що відповідають сучасним вимогам безпеки.

1.4 Класифікація протоколів VPN (тунелювання, криптозахист, авторизація та аутентифікація, мережеві протоколи, шифрозахист IP-адрес)

Протоколи VPN є основою функціонування віртуальних приватних мереж, забезпечуючи різні аспекти безпеки та зв'язку в розподілених системах. Їх класифікація за функціональними характеристиками, такими як тунелювання, криптозахист, авторизація та аутентифікація, мережеві протоколи та шифрозахист IP-адрес, дозволяє систематизувати їх для вибору оптимального рішення. Кожен тип протоколів виконує специфічні функції, що забезпечують безпечну передачу даних і взаємодію між компонентами системи [16]. Для наочності основні категорії протоколів VPN та їх характеристики представлені в таблиці 1.1.

Протоколи тунелювання відповідають за створення віртуального каналу між двома точками мережі, що дозволяє ізолювати дані від зовнішнього середовища. До них належать PPTP (Point-to-Point Tunneling Protocol), L2TP (Layer 2 Tunneling Protocol) і GRE (Generic Routing Encapsulation). PPTP є одним із найстаріших протоколів, який використовує GRE для тунелювання та MS-CHAP для аутентифікації. Він вирізняється високою швидкістю, але має слабкий захист через застарілі алгоритми шифрування, такі як MPPE [17]. L2TP, який часто комбінується з IPsec, забезпечує надійніше тунелювання, але потребує додаткових ресурсів через подвійне інкапсулювання. GRE використовується в корпоративних мережах для

створення тунелів, але не включає вбудованих механізмів шифрування, що вимагає додаткових протоколів для захисту.

Протоколи криптозахисту забезпечують шифрування даних, що передаються через тунель, гарантуючи їх конфіденційність і цілісність. IPsec є одним із найпоширеніших протоколів, який використовує алгоритми шифрування, такі як AES, і працює на мережевому рівні, що робить його універсальним для захисту даних [18]. OpenVPN, заснований на бібліотеці OpenSSL, підтримує широкий спектр алгоритмів шифрування, таких як AES-256 і ChaCha20, що забезпечує високу гнучкість і безпеку. WireGuard, сучасний протокол, використовує оптимізовані криптографічні алгоритми, такі як Curve25519, які забезпечують високу продуктивність і низькі затримки [19].

Протоколи авторизації та аутентифікації відповідають за перевірку ідентичності користувачів або пристроїв, що підключаються до VPN. Наприклад, протокол RADIUS (Remote Authentication Dial-In User Service) використовується для централізованої аутентифікації в великих мережах. IPsec підтримує аутентифікацію через попередньо спільні ключі (PSK) або цифрові сертифікати, що підвищує безпеку доступу [20]. OpenVPN також підтримує різні методи аутентифікації, включаючи двофакторну аутентифікацію (2FA), що дозволяє додатково захистити систему від несанкціонованого доступу.

Мережеві протоколи визначають спосіб передачі даних у VPN. TCP (Transmission Control Protocol) забезпечує надійну передачу даних завдяки підтвердженню пакетів, але може бути повільнішим через ці накладні витрати. UDP (User Datagram Protocol), який використовується в WireGuard і OpenVPN, забезпечує швидшу передачу, але без гарантії доставки, що робить його придатним для сценаріїв, де швидкість є пріоритетом [18]. Вибір між TCP і UDP залежить від вимог до продуктивності та надійності системи.

Протоколи шифрозахисту IP-адрес дозволяють приховувати реальні IP-адреси вузлів для підвищення анонімності та безпеки. Наприклад, NAT (Network Address Translation) у поєднанні з VPN дозволяє маскувати внутрішні IP-адреси, що ускладнює відстеження джерела трафіку. Протокол Tor, хоча і не є VPN-

протоколом, іноді використовується для додаткового захисту IP-адрес, але має обмеження через низьку продуктивність і складність інтеграції в корпоративні системи [19].

Таблиця 1.2 - Класифікація протоколів VPN за функціональними характеристиками

Категорія	Протоколи	Основні характеристики	Приклади використання
Тунелювання	PPTP, L2TP, GRE	Створення віртуальних каналів	Корпоративні мережі, віддалений доступ
Криптозахист	IPsec, OpenVPN, WireGuard	Шифрування даних	Захист даних у хмарних системах
Авторизація та аутентифікація	RADIUS, PSK, Сертифікати	Перевірка ідентичності	Доступ до корпоративних ресурсів
Мережеві протоколи	TCP, UDP	Передача даних	Високопродуктивні або надійні з’єднання
Шифрозахист IP-адрес	NAT, Tor	Маскування IP-адрес	Анонімність, захист від відстеження

Класифікація протоколів VPN за їх функціональними характеристиками дозволяє систематизувати їх для вибору оптимального рішення залежно від потреб розподілених захищених систем.

1.5 Аналіз можливостей, переваг та недоліків VPN-протоколів

Протоколи VPN є основою для створення захищених каналів зв’язку в розподілених системах, але кожен із них має свої унікальні можливості, переваги та недоліки, що впливають на їх вибір для конкретних сценаріїв використання. У

цьому пункті детально розглядаються основні VPN-протоколи, такі як PPTP, L2TP/IPsec, OpenVPN і WireGuard, з акцентом на їх функціональність, продуктивність і безпеку. Для наочності порівняння характеристик і оцінки переваг та недоліків протоколів представлені в таблицях 1.2 і 1.3 [21].

PPTP (Point-to-Point Tunneling Protocol) є одним із найстаріших протоколів VPN, розроблених Microsoft. Він характеризується простотою налаштування та високою швидкістю роботи завдяки мінімальним вимогам до обчислень. PPTP використовує протокол GRE для тунелювання та MS-CHAP для аутентифікації, що дозволяє швидко встановлювати з'єднання. Однак його головним недоліком є низький рівень безпеки через використання застарілих алгоритмів шифрування, таких як MPPE (Microsoft Point-to-Point Encryption), які вразливі до сучасних атак, таких як атаки на основі словникового підбору паролів [22]. PPTP підходить для сценаріїв, де швидкість є пріоритетом, але безпека не є критичною, наприклад, для потокового передавання даних або доступу до некритичних ресурсів.

L2TP/IPsec поєднує протокол тунелювання L2TP з криптографічним захистом IPsec, що забезпечує значно вищий рівень безпеки порівняно з PPTP. L2TP відповідає за створення тунелю, тоді як IPsec забезпечує шифрування даних за допомогою алгоритмів, таких як AES. Цей протокол широко використовується в корпоративних мережах завдяки високій сумісності та підтримці сучасних стандартів безпеки [23]. Однак L2TP/IPsec має значні накладні витрати через подвійне інкапсулювання пакетів, що може знижувати продуктивність, особливо на пристроях із обмеженими ресурсами. Крім того, складність налаштування може бути недоліком для малих організацій із обмеженим IT-персоналом.

OpenVPN є одним із найпопулярніших протоколів завдяки своїй гнучкості та високому рівню безпеки. Він базується на бібліотеці OpenSSL, що дозволяє підтримувати широкий спектр алгоритмів шифрування, таких як AES-256, Blowfish і ChaCha20. OpenVPN підтримує як TCP, так і UDP для передачі даних, що дозволяє оптимізувати його для різних сценаріїв, наприклад, надійного з'єднання (TCP) або високої швидкості (UDP) [24]. Як відкрите програмне забезпечення, OpenVPN дозволяє перевіряти його код на наявність вразливостей, що підвищує довіру до

нього. Основним недоліком є потреба в додатковому програмному забезпеченні для налаштування та потенційне зниження продуктивності на слабких пристроях через високі вимоги до обчислень.

WireGuard є сучасним протоколом, який вирізняється мінімалістичним дизайном і високою продуктивністю. Він використовує сучасні криптографічні алгоритми, такі як Curve25519 для обміну ключами та ChaCha20 для шифрування, що забезпечують швидке та безпечне з'єднання. WireGuard має значно менший розмір коду порівняно з OpenVPN, що знижує ймовірність вразливостей і спрощує аудит безпеки [25]. Однак його відносна новизна означає меншу підтримку в деяких системах і обмежену сумісність із застарілим обладнанням, що може бути проблемою для великих організацій із різнорідною інфраструктурою.

Таблиця 1.3 - Порівняння можливостей і продуктивності VPN-протоколів

Протокол	Швидкість	Рівень безпеки	Сумісність	Складність налаштування
PPTP	Висока	Низька	Висока	Низька
L2TP/IPsec	Середня	Висока	Висока	Середня
OpenVPN	Середня	Дуже висока	Висока	Висока
WireGuard	Висока	Висока	Середня	Низька

Таблиця 1.4 - Переваги та недоліки VPN-протоколів

Протокол	Переваги	Недоліки
PPTP	Висока швидкість, простота налаштування	Низька безпека, вразливість до атак
L2TP/IPsec	Висока безпека, широка сумісність	Низька продуктивність, складність
OpenVPN	Гнучкість, висока безпека, відкритий код	Високі вимоги до ресурсів
WireGuard	Висока продуктивність, сучасне шифрування	Обмежена сумісність, новизна протоколу

Вибір VPN-протоколу залежить від потреб системи в безпеці, продуктивності та сумісності. OpenVPN і WireGuard є оптимальними для сучасних розподілених систем, тоді як PPTP і L2TP/IPsec можуть використовуватися в специфічних сценаріях із обмеженими вимогами до безпеки або ресурсів.

1.6 Огляд сучасних підходів до створення захищених автономних систем

Сучасні підходи до створення захищених автономних систем спрямовані на забезпечення безпеки, масштабованості та відмовостійкості в умовах зростання кіберзагроз і складності мережевих інфраструктур. Ці підходи включають використання передових технологій, таких як VPN, програмно-визначені мережі (SDN), концепція нульової довіри (Zero Trust) та штучний інтелект (AI) для автоматизації захисту. Для систематизації сучасних підходів їх основні характеристики представлені в таблиці 1.4 [26].

Інтеграція VPN-технологій є одним із ключових підходів до створення захищених автономних систем. VPN дозволяє створювати безпечні канали зв’язку між географічно віддаленими вузлами, що є критично важливим для хмарних платформ і корпоративних мереж. Наприклад, протоколи OpenVPN і WireGuard забезпечують шифрування даних і захист від перехоплення, що дозволяє безпечно передавати конфіденційні дані через незахищені мережі [27]. Крім того, VPN інтегрується з SD-WAN, що оптимізує маршрутизацію трафіку та забезпечує автоматичне керування мережею в реальному часі. SD-WAN дозволяє динамічно перенаправляти трафік через найшвидші або найбезпечніші канали, що підвищує продуктивність і надійність системи.

Концепція нульової довіри (Zero Trust) набуває дедалі більшої популярності в захисті автономних систем. Вона базується на принципі, що жоден користувач або пристрій не є автоматично довіреним, навіть якщо вони знаходяться всередині мережі. Zero Trust вимагає суворої аутентифікації, такої як двофакторна аутентифікація (2FA), і постійного моніторингу трафіку для виявлення аномалій [28]. Наприклад, рішення від Zscaler інтегрують Zero Trust із VPN, забезпечуючи безпечний доступ до хмарних ресурсів і захист від несанкціонованого доступу. Цей

підхід є особливо ефективним у розподілених системах, де компоненти можуть бути розташовані в різних географічних регіонах.

Використання штучного інтелекту (AI) і машинного навчання (ML) є ще одним сучасним підходом до захисту автономних систем. AI дозволяє аналізувати великі обсяги мережевого трафіку, виявляти аномалії та автоматично реагувати на потенційні загрози. Наприклад, рішення від Check Point використовують AI для прогнозування атак і автоматичного налаштування політик безпеки, що зменшує час реагування на інциденти [29]. ML також застосовується для аналізу поведінки користувачів, що дозволяє виявляти підозрілі дії, такі як спроби несанкціонованого доступу.

Програмно-визначені мережі (SDN) відіграють важливу роль у створенні захищених автономних систем. SDN дозволяє централізовано керувати мережею, що полегшує впровадження політик безпеки та сегментацію мережі. Наприклад, Cisco SD-Access використовує SDN для створення ізольованих мережевих сегментів, що знижує ризик поширення загроз у разі компрометації одного з вузлів [30]. SDN також дозволяє автоматизувати управління мережею, що є важливим для великих розподілених систем із великою кількістю вузлів.

Хмарні технології є ще одним важливим елементом сучасних автономних систем. Платформи, такі як AWS, Azure або Google Cloud, використовують VPN і SD-WAN для створення захищених розподілених систем, які можуть динамічно масштабуватися залежно від потреб. Однак хмарні системи потребують ретельного налаштування для захисту від атак, таких як витік даних або компрометація хмарних ресурсів [27]. Наприклад, AWS пропонує рішення AWS VPN, яке інтегрується з хмарною інфраструктурою для забезпечення безпечного доступу до ресурсів.

Таблиця 1.5- Сучасні підходи до створення захищених автономних систем

Підхід	Опис	Приклади технологій
VPN	Захищені канали зв’язку	OpenVPN, WireGuard, AWS VPN

Zero Trust	Суворі аутентифікація та моніторинг	Zscaler, 2FA
AI та ML	Автоматизація виявлення загроз	Check Point, Darktrace
SDN	Централізоване управління мережею	Cisco SD-Access, VMware NSX
Хмарні технології	Масштабованість і гнучкість	AWS, Azure, Google Cloud

Сучасні підходи до створення захищених автономних систем поєднують VPN, Zero Trust, SDN і AI для забезпечення комплексного захисту. Ці рішення дозволяють створювати надійні та гнучкі системи, здатні протистояти сучасним кіберзагрозам.

1.7 Висновки до розділу

У першому розділі проведено детальний аналіз теоретичних основ розподілених захищених автономних систем та протоколів VPN, що є ключовими для розуміння їхньої ролі в сучасних інформаційних технологіях. Автономні системи характеризуються незалежним функціонуванням під єдиним адміністративним управлінням і потребують комплексного захисту для забезпечення конфіденційності, цілісності та доступності даних. Використання криптографічних методів, таких як AES, сегментація мережі за допомогою VLAN або SDN, а також системи виявлення та запобігання вторгненням дозволяють створювати надійні автономні системи, здатні протистояти кіберзагрозам [31].

Розподілені системи вирізняються високою масштабованістю, відмовостійкістю та асинхронною взаємодією між вузлами, що створює додаткові виклики для забезпечення безпеки. Алгоритми, такі як Paxos і Raft, забезпечують узгодженість даних, тоді як VPN-технології дозволяють створювати захищені канали зв’язку між географічно віддаленими компонентами [32]. Ці характеристики роблять розподілені системи ідеальними для хмарних платформ і корпоративних мереж, але потребують ретельного управління та моніторингу.

Протоколи VPN відіграють центральну роль у побудові розподілених захищених систем, забезпечуючи шифрування, аутентифікацію та захист від атак типу "людина посередині". Вони дозволяють створювати безпечні тунелі для передачі даних через незахищені мережі, що є особливо важливим для хмарних і розподілених інфраструктур [33]. Класифікація VPN-протоколів за їх функціональними характеристиками, такими як тунелювання, криптозахист, авторизація, мережеві протоколи та шифрозахист IP-адрес, дозволяє систематизувати їх для вибору оптимального рішення залежно від потреб системи [34].

Аналіз можливостей, переваг і недоліків VPN-протоколів, таких як PPTP, L2TP/IPsec, OpenVPN і WireGuard, показав, що OpenVPN і WireGuard є оптимальними для сучасних систем завдяки високому рівню безпеки та продуктивності. PPTP і L2TP/IPsec можуть використовуватися в специфічних сценаріях, але мають обмеження через низьку безпеку або високу складність [35]. Сучасні підходи до створення захищених автономних систем включають інтеграцію VPN із SD-WAN, Zero Trust і технологіями штучного інтелекту, що забезпечує комплексний захист від сучасних кіберзагроз [36].

Таким чином, теоретичний аналіз підтвердив важливість VPN-протоколів у забезпеченні безпеки розподілених захищених автономних систем. Подальші дослідження в цій роботі будуть зосереджені на розробці методики оцінки VPN-протоколів і їх практичному тестуванні для визначення оптимальних рішень.

РОЗДІЛ 2. МЕТОДИКА ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ПРОТОКОЛІВ VPN У РОЗПОДІЛЕНИХ СИСТЕМАХ

2.1 Вимоги до побудови розподілених захищених автономних систем

Побудова розподілених захищених автономних систем є складним завданням, яке вимагає чіткого визначення вимог до безпеки, продуктивності, масштабованості, відмовостійкості, керованості та сумісності. Ці вимоги формують основу для проектування систем, які здатні забезпечувати надійну взаємодію між географічно віддаленими вузлами, захищаючи дані від кіберзагроз і забезпечуючи стабільне функціонування в умовах високих навантажень. Розподілені системи, які функціонують як автономні, повинні відповідати сучасним стандартам інформаційної безпеки та бути адаптивними до змінних умов експлуатації. У цьому пункті детально розглядаються ключові вимоги до побудови таких систем, які є основою для інтеграції протоколів VPN як основного інструменту захисту [21].

Безпека даних є першочерговою вимогою для розподілених захищених автономних систем, оскільки вони часто обробляють конфіденційні дані, такі як фінансові транзакції, персональна інформація або комерційні секрети. Конфіденційність даних забезпечується через використання сучасних криптографічних алгоритмів, таких як AES-256, який є стандартом для шифрування в багатьох VPN-протоколах. Цей алгоритм гарантує, що дані, які передаються через незахищені мережі, такі як Інтернет, залишаються недоступними для зловмисників [22]. Цілісність даних досягається за допомогою хеш-функцій, таких як SHA-256, які дозволяють виявляти будь-які несанкціоновані зміни в даних. Аутентифікація та авторизація є критично важливими для запобігання несанкціонованому доступу. Наприклад, використання двофакторної аутентифікації (2FA) або цифрових сертифікатів забезпечує надійну перевірку ідентичності користувачів і пристроїв, що підключаються до системи [23]. Крім того, захист від атак типу "людина посередині" (MITM) або DDoS-атак вимагає

впровадження систем виявлення вторгнень (IDS) і запобігання вторгненням (IPS), які аналізують мережевий трафік і реагують на підозрілі дії в реальному часі.

Масштабованість є ключовою вимогою для розподілених систем, оскільки вони повинні бути здатними адаптуватися до зростання кількості вузлів, користувачів або обсягу даних. Масштабованість дозволяє системі ефективно обробляти збільшення навантаження без значного погіршення продуктивності або безпеки. Наприклад, хмарні платформи, такі як Amazon Web Services (AWS) або Microsoft Azure, використовують автоматичне масштабування для розподілу ресурсів між вузлами залежно від поточного навантаження [24]. У контексті розподілених систем архітектури на основі мікросервісів дозволяють масштабувати окремі компоненти незалежно один від одного, що забезпечує гнучкість і ефективність використання ресурсів. Масштабованість також передбачає можливість швидкого додавання нових вузлів до системи, що є важливим для корпоративних мереж, які розширюються через відкриття нових офісів або підключення віддалених працівників. Для цього використовуються технології, такі як Kubernetes, які автоматизують розгортання та управління новими вузлами, забезпечуючи безперебійну інтеграцію з мережею.

Відмовостійкість є ще однією критично важливою вимогою, яка гарантує безперервну роботу системи навіть у разі збою окремих компонентів. Розподілені системи повинні бути спроектовані таким чином, щоб вихід з ладу одного вузла не призводив до зупинки всієї системи. Для цього використовуються механізми реплікації даних, балансування навантаження та резервного копіювання. Наприклад, розподілені бази даних, такі як MongoDB або Cassandra, дублюють дані на кількох вузлах, що дозволяє відновити доступ до інформації в разі апаратного або програмного збою [25]. Балансувальники навантаження, такі як NGINX або HAProxy, розподіляють запити між доступними вузлами, що підвищує надійність системи. Відмовостійкість також включає здатність системи автоматично перенаправляти трафік у разі відмови одного з каналів зв’язку, що може бути реалізовано через програмно-визначені мережі (SDN) або інтеграцію з VPN-протоколами, які підтримують динамічне перенаправлення.

Продуктивність є важливим аспектом, оскільки розподілені системи часто обробляють великі обсяги даних у реальному часі, що вимагає низьких затримок і високої пропускної здатності. Використання VPN може впливати на продуктивність через додаткові накладні витрати на шифрування та тунелювання, тому важливо оптимізувати вибір протоколу VPN для забезпечення балансу між безпекою та швидкістю передачі даних [22]. Наприклад, протоколи, такі як WireGuard, пропонують високу продуктивність завдяки оптимізованим криптографічним алгоритмам, таким як ChaCha20, які забезпечують швидке шифрування з мінімальними витратами ресурсів. Для оцінки продуктивності використовуються метрики, такі як пропускна здатність (throughput), затримка (latency) і використання обчислювальних ресурсів (CPU usage). Інструменти, такі як iPerf, дозволяють вимірювати пропускну здатність мережі, тоді як ping використовується для оцінки затримок. Продуктивність також залежить від типу трафіку: наприклад, потокове передавання даних або відеоконференції потребують мінімальних затримок, тоді як передача великих файлів вимагає високої пропускної здатності.

Керованість передбачає можливість централізованого управління системою, включаючи налаштування мережевих пристроїв, моніторинг стану системи та швидке реагування на інциденти. У розподілених системах із великою кількістю вузлів управління може бути складним завданням, тому використовуються інструменти автоматизації, такі як Ansible, Terraform або Kubernetes, які спрощують конфігурацію та оновлення системи [24]. Системи моніторингу, такі як Prometheus або Grafana, дозволяють відстежувати продуктивність і виявляти аномалії в реальному часі, що є важливим для забезпечення безпеки та стабільності. Наприклад, Prometheus може аналізувати метрики продуктивності VPN-з’єднань, такі як затримки або втрати пакетів, що дозволяє адміністраторам швидко реагувати на проблеми. Керованість також включає можливість швидкого оновлення політик безпеки, наприклад, у разі виявлення нових вразливостей у VPN-протоколах.

Сумісність із різними платформами та пристроями є ще однією важливою вимогою, оскільки розподілені системи часто включають гетерогенні компоненти, такі як сервери, хмарні платформи, мобільні пристрої та IoT-пристрої. Протоколи VPN, які використовуються в таких системах, повинні підтримувати широкий спектр операційних систем (Windows, Linux, macOS, Android, iOS) і апаратного забезпечення, включаючи маршрутизатори, комутатори та спеціалізовані пристрої. Наприклад, OpenVPN є універсальним завдяки підтримці різних платформ, тоді як WireGuard може мати обмеження через відносну новизну, що вимагає додаткового програмного забезпечення для застарілих систем [25]. Сумісність також передбачає можливість інтеграції з іншими технологіями, такими як SD-WAN або системи безпеки, наприклад, брандмауери нового покоління (NGFW). Наприклад, Cisco AnyConnect інтегрується з корпоративними системами безпеки, що полегшує управління доступом у великих мережах.

Таблиця 2.1 - Вимоги до побудови розподілених захищених автономних систем

Вимога	Опис	Методи реалізації	Приклади технологій
Безпека	Шифрування, аутентифікація, захист від атак	AES-256, 2FA, IDS/IPS	OpenVPN, WireGuard, Nessus
Масштабованість	Додавання вузлів без втрати продуктивності	Автоматичне масштабування	Kubernetes, AWS Auto Scaling
Відмовостійкість	Стійкість до збоїв окремих вузлів	Реплікація, балансування навантаження	MongoDB, HAProxy, Cassandra
Продуктивність	Низькі затримки, висока пропускна здатність	Оптимізовані протоколи VPN	WireGuard, iPerf

Керованість	Централізоване управління та моніторинг	Автоматизація, системи моніторингу	Ansible, Prometheus, Grafana
Сумісність	Підтримка різних платформ і пристроїв	Універсальні протоколи VPN	OpenVPN, Cisco AnyConnect

Вимоги до побудови розподілених захищених автономних систем охоплюють безпеку, масштабованість, відмовостійкість, продуктивність, керованість і сумісність. Ці вимоги формують основу для вибору та впровадження протоколів VPN, які є ключовим інструментом для забезпечення захисту таких систем у сучасних інформаційних середовищах.

2.2 Вибір оптимального протоколу VPN для різних сценаріїв використання

Вибір оптимального протоколу VPN для розподілених захищених автономних систем є ключовим етапом, оскільки від нього залежить безпека, продуктивність і сумісність системи в різних сценаріях використання. Кожен протокол VPN, такий як PPTP, L2TP/IPsec, OpenVPN або WireGuard, має унікальні характеристики, які роблять його придатним для певних завдань. Вибір залежить від вимог до безпеки, продуктивності, масштабованості, сумісності та простоти налаштування, а також від специфіки сценарію використання, таких як корпоративні мережі, хмарні платформи, віддалений доступ або системи реального часу. У цьому пункті розглядаються критерії вибору VPN-протоколів і їх застосування в різних контекстах [26].

Корпоративні мережі є одним із найпоширеніших сценаріїв використання VPN, де основним пріоритетом є захист конфіденційних даних, таких як фінансові звіти, комерційна інформація або персональні дані клієнтів. У таких мережах важливо забезпечити високий рівень безпеки, що включає надійне шифрування, стійкість до атак і механізми аутентифікації. OpenVPN є оптимальним вибором для корпоративних мереж завдяки підтримці сучасних алгоритмів шифрування, таких

як AES-256, і можливості використання двофакторної аутентифікації (2FA) для додаткового захисту [27]. OpenVPN також є відкритим програмним забезпеченням, що дозволяє перевіряти його код на наявність вразливостей, що є важливим для організацій із високими вимогами до безпеки. Однак налаштування OpenVPN може бути складним, оскільки воно вимагає встановлення додаткового програмного забезпечення та конфігураційних файлів, що може бути викликом для організацій із обмеженим IT-персоналом. У таких випадках L2TP/IPsec може бути альтернативою завдяки високій сумісності з більшістю мережевих пристроїв і операційних систем, таких як Windows, macOS і Linux. Проте L2TP/IPsec має нижчу продуктивність через подвійне інкапсулювання пакетів, що може впливати на швидкість передачі даних у великих мережах [28]. Для корпоративних мереж із великою кількістю користувачів важливо також враховувати масштабованість, оскільки протокол повинен підтримувати підключення великої кількості клієнтів без значного зниження продуктивності.

Хмарні платформи, такі як Amazon Web Services (AWS), Microsoft Azure або Google Cloud, є ще одним важливим сценарієм, де VPN використовується для створення захищених з'єднань між хмарними ресурсами та локальними мережами. У хмарних системах ключовими вимогами є гнучкість, висока продуктивність і можливість масштабування. WireGuard є ідеальним вибором для хмарних платформ завдяки мінімалістичному дизайну та оптимізованим криптографічним алгоритмам, таким як ChaCha20, які забезпечують швидке шифрування з мінімальними накладними витратами [29]. WireGuard дозволяє створювати захищені тунелі між хмарними серверами та клієнтськими пристроями, що є важливим для динамічного масштабування, наприклад, під час розгортання нових віртуальних машин. Однак обмежена сумісність WireGuard із застарілими системами може створювати проблеми в гетерогенних середовищах, де використовуються різні типи обладнання та операційних систем. У таких випадках OpenVPN може бути кращим вибором завдяки універсальності та підтримці широкого спектру платформ, включаючи спеціалізовані хмарні рішення, такі як AWS VPN [27]. Наприклад, AWS пропонує інтеграцію OpenVPN для створення

захищених з’єднань між хмарними ресурсами та локальними мережами, що полегшує управління великими розподіленими системами.

Віддалений доступ для співробітників або користувачів є ще одним поширеним сценарієм, де VPN використовується для забезпечення безпечного доступу до внутрішніх ресурсів компанії з віддалених локацій. У цьому контексті важливими є простота налаштування, підтримка різних пристроїв (ПК, смартфони, планшети) і захист від несанкціонованого доступу. PPTP, незважаючи на свою застарілість і низький рівень безпеки через використання слабких алгоритмів шифрування, таких як MPPE, може використовуватися для некритичних сценаріїв, де швидкість і простота є пріоритетами [30]. Наприклад, PPTP може бути застосовним для доступу до некритичних ресурсів, таких як внутрішні веб-портали, де немає потреби в обробці конфіденційних даних. Однак для забезпечення безпеки рекомендується використовувати OpenVPN або WireGuard, які підтримують сучасні методи аутентифікації, такі як цифрові сертифікати або 2FA. OpenVPN є особливо зручним для віддаленого доступу завдяки підтримці клієнтських додатків для всіх основних платформ, включаючи Android і iOS, що дозволяє співробітникам безпечно підключатися до корпоративної мережі з мобільних пристроїв [27]. WireGuard, у свою чергу, забезпечує швидке з’єднання, що є важливим для віддалених працівників, які використовують обмежені канали зв’язку, наприклад, мобільний Інтернет.

Системи реального часу, такі як потокове передавання даних, відеоконференції або системи Інтернету речей (IoT), потребують VPN-протоколів із мінімальними затримками та високою пропускною здатністю. У таких сценаріях продуктивність є критично важливою, оскільки навіть незначні затримки можуть вплинути на якість роботи системи. WireGuard є оптимальним вибором завдяки підтримці UDP і оптимізованим криптографічним алгоритмам, які забезпечують швидку передачу даних із мінімальними витратами ресурсів [29]. Наприклад, у системах IoT, де пристрої мають обмежені обчислювальні потужності, WireGuard дозволяє створювати захищені з’єднання без значного впливу на продуктивність. OpenVPN також може використовуватися в таких системах, але потребує ретельної

оптимізації, наприклад, вибору UDP замість TCP, щоб зменшити затримки. PPTP, хоча і забезпечує високу швидкість, не рекомендується для систем реального часу через низьку безпеку, що може бути критичним для IoT-пристроїв, які передають чутливі дані, наприклад, сенсорні дані в розумних містах [30].

Критерії вибору протоколу VPN включають аналіз вимог до безпеки, продуктивності, сумісності, масштабованості та простоти налаштування. Для оцінки цих критеріїв використовуються різні методи, такі як тестування продуктивності за допомогою інструментів iPerf або ping, аналіз безпеки через сканування вразливостей за допомогою Nessus, а також оцінка сумісності шляхом тестування на різних платформах. Наприклад, для високонавантажених систем із критичними даними пріоритет надається безпеці, що робить OpenVPN або WireGuard кращими виборами [28]. Для систем із низькими вимогами до безпеки, таких як потокове передавання некритичних даних, PPTP може бути достатнім завдяки високій швидкості та простоті налаштування. У великих розподілених системах також враховується можливість інтеграції VPN із іншими технологіями, такими як SD-WAN, які оптимізують маршрутизацію та підвищують продуктивність.

Вибір оптимального протоколу VPN залежить від специфіки сценарію використання та вимог до системи. OpenVPN і WireGuard є універсальними рішеннями для більшості сценаріїв завдяки високому рівню безпеки та продуктивності, тоді як L2TP/IPsec і PPTP можуть використовуватися в специфічних випадках із обмеженими вимогами до безпеки або ресурсів. Подальші дослідження передбачають практичне тестування цих протоколів для підтвердження їхньої ефективності в різних сценаріях.

2.3 Розробка моделі розподілених захищених автономних систем

Розробка моделі розподілених захищених автономних систем є ключовим етапом дослідження, спрямованим на створення структурованого підходу до проектування систем, які поєднують автономність, безпеку та ефективну взаємодію між географічно віддаленими вузлами. Така модель повинна враховувати вимоги

до безпеки, масштабованості, відмовостійкості, продуктивності та керованості, а також інтегрувати протоколи VPN для забезпечення захисту даних. Модель слугує основою для подальшого практичного тестування та оцінки ефективності VPN-протоколів у реальних сценаріях. У цьому пункті розглядається процес створення такої моделі, включаючи визначення архітектури, вибір компонентів, інтеграцію VPN-протоколів і забезпечення ключових характеристик системи [31].

Визначення архітектури системи є першим кроком у розробці моделі. Розподілена захищена автономна система складається з множини вузлів, які можуть включати сервери, клієнтські пристрої, хмарні ресурси та IoT-пристрої, що взаємодіють через мережу. Архітектура може базуватися на різних підходах, таких як клієнт-серверна модель, мікросервісна архітектура або гібридний підхід. Мікросервісна архітектура є особливо ефективною для розподілених систем, оскільки вона дозволяє ізолювати окремі компоненти, що підвищує відмовостійкість і спрощує масштабування [32]. Наприклад, кожен мікросервіс може відповідати за певну функцію, таку як обробка даних, зберігання або маршрутизація, що дозволяє оптимізувати ресурси та спрощувати управління. У моделі визначаються ролі кожного вузла, наприклад, центральний сервер для управління політиками безпеки, вузли для обробки даних або клієнтські пристрої для доступу до системи. Архітектура також передбачає використання програмно-визначених мереж (SDN), які дозволяють централізовано керувати мережею та динамічно адаптувати її до змінних умов.

Інтеграція VPN-протоколів є центральним елементом моделі, оскільки вони забезпечують захист даних під час передачі між вузлами через незахищені мережі, такі як Інтернет. Модель передбачає використання VPN для створення захищених тунелів, які гарантують конфіденційність, цілісність і автентичність даних. Наприклад, OpenVPN може бути використаний для створення тунелів із підтримкою алгоритму шифрування AES-256, що забезпечує високий рівень безпеки для передачі конфіденційних даних [33]. WireGuard, завдяки своїй високій продуктивності та мінімалістичному дизайну, є оптимальним для систем із обмеженими ресурсами або високими вимогами до швидкості, наприклад, у IoT або

системах реального часу. Модель також враховує можливість використання кількох протоколів VPN для різних типів трафіку: наприклад, OpenVPN для критичних даних із високими вимогами до безпеки, а WireGuard для потокового передавання даних із низькими затримками. Важливим аспектом є конфігурація VPN-серверів і клієнтів, яка включає налаштування шифрування, аутентифікації та маршрутизації для забезпечення безпечної взаємодії між вузлами.

Забезпечення безпеки у моделі є багатогранним процесом, який охоплює шифрування, аутентифікацію, сегментацію мережі та моніторинг. Шифрування даних за допомогою алгоритмів, таких як AES-256 або ChaCha20, забезпечує захист від перехоплення. Аутентифікація включає використання цифрових сертифікатів, попередньо спільних ключів (PSK) або двофакторної аутентифікації (2FA) для перевірки ідентичності вузлів і користувачів перед встановленням з'єднання [34]. Сегментація мережі досягається через використання віртуальних локальних мереж (VLAN) або SDN, що дозволяє ізолювати критичні компоненти системи від менш захищених сегментів. Наприклад, SDN-контролери, такі як Cisco SD-Access, дозволяють створювати ізольовані мережеві сегменти, що знижує ризик поширення загроз у разі компрометації одного з вузлів. Моніторинг мережевого трафіку за допомогою систем, таких як Prometheus або Grafana, забезпечує виявлення аномалій, які можуть свідчити про кібератаки, такі як DDoS або спроби несанкціонованого доступу. Модель також передбачає впровадження систем виявлення вторгнень (IDS) і запобігання вторгненням (IPS) для реагування на загрози в реальному часі.

Масштабованість і відмовостійкість є ключовими характеристиками моделі, які забезпечують її здатність адаптуватися до зростання навантаження та протистояти збоям. Для забезпечення масштабованості модель включає механізми автоматичного додавання нових вузлів, наприклад, за допомогою оркестраторів, таких як Kubernetes, які дозволяють динамічно розподіляти ресурси залежно від потреб [32]. Kubernetes може автоматично розгортати нові контейнери або віртуальні машини для обробки додаткового трафіку, що є важливим для хмарних платформ. Відмовостійкість досягається через реплікацію даних і балансування

навантаження. Наприклад, розподілені бази даних, такі як Cassandra, дублюють дані на кількох вузлах, що дозволяє відновити доступ до інформації в разі збою [34]. Балансувальники навантаження, такі як HAProxy, розподіляють запити між доступними вузлами, що підвищує надійність системи. Модель також передбачає використання резервного копіювання даних у захищеному середовищі, що дозволяє швидко відновити систему після апаратних або програмних збоїв.

Продуктивність моделі залежить від вибору протоколів VPN і оптимізації їх налаштувань. Наприклад, використання UDP замість TCP у WireGuard або OpenVPN зменшує затримки, що є критично важливим для систем реального часу, таких як потокове передавання даних або відеоконференції [33]. Модель включає механізми моніторингу продуктивності, які дозволяють оцінювати вплив VPN на пропускну здатність і затримки. Наприклад, інструменти, такі як iPerf, можуть використовуватися для вимірювання пропускної здатності, тоді як ping оцінює затримки. Оптимізація продуктивності також передбачає вибір оптимальних алгоритмів шифрування, які забезпечують баланс між безпекою та швидкістю. Наприклад, WireGuard використовує ChaCha20, який є швидшим за AES на пристроях із обмеженими ресурсами, що робить його придатним для IoT-пристроїв [35]. Модель також враховує можливість використання апаратного прискорення шифрування на сучасних процесорах для підвищення продуктивності.

Керованість моделі забезпечується через використання інструментів автоматизації та моніторингу. Наприклад, Ansible або Terraform дозволяють автоматизувати конфігурацію VPN-серверів і клієнтів, що спрощує розгортання системи [32]. Системи моніторингу, такі як Prometheus, надають детальну інформацію про стан мережі, включаючи метрики продуктивності VPN-з’єднань, такі як втрати пакетів або затримки. Модель також передбачає централізоване управління політиками безпеки, що дозволяє швидко оновлювати конфігурацію у відповідь на нові загрози або зміни в системі.

Розробка моделі розподілених захищених автономних систем передбачає визначення архітектури, інтеграцію VPN-протоколів, забезпечення безпеки, масштабованості, відмовостійкості, продуктивності та керованості. Ця модель є

основою для подальшого практичного тестування та оцінки ефективності VPN-протоколів у реальних сценаріях, що дозволить визначити оптимальні рішення для різних типів розподілених систем.

2.4 Критерії оцінки ефективності VPN-протоколів

Оцінка ефективності VPN-протоколів є ключовим етапом дослідження їх використання в розподілених захищених автономних системах. Ефективність протоколів VPN визначається їх здатністю забезпечувати баланс між безпекою, продуктивністю, сумісністю та простотою впровадження, що є критично важливим для забезпечення надійного функціонування системи. У цьому пункті розглядаються основні критерії оцінки ефективності VPN-протоколів, такі як безпека, продуктивність, масштабованість, сумісність і простота налаштування, а також методи їх вимірювання. Для наочності основні критерії оцінки представлені в таблиці 2.1 [36].

Безпека є одним із найважливіших критеріїв оцінки VPN-протоколів, оскільки вони використовуються для захисту даних у розподілених системах. Безпека оцінюється за такими параметрами, як сила шифрування, стійкість до атак і надійність механізмів аутентифікації. Наприклад, протоколи, що використовують алгоритми шифрування AES-256, такі як OpenVPN або IPsec, вважаються більш безпечними порівняно з PPTP, який використовує застарілий алгоритм MPPE [37]. Стійкість до атак типу "людина посередині" (MITM) або атак на основі словникового підбору паролів є критично важливою. Для оцінки безпеки проводяться тести на вразливості, такі як сканування за допомогою інструментів на кшталт Nessus, а також аналіз стійкості протоколу до відомих атак, таких як CVE (Common Vulnerabilities and Exposures). Крім того, підтримка двофакторної аутентифікації (2FA) або цифрових сертифікатів підвищує безпеку доступу до системи [38]. Наприклад, WireGuard використовує сучасні криптографічні алгоритми, такі як Curve25519, що забезпечують високий рівень безпеки при меншій обчислювальній складності.

Продуктивність є ще одним ключовим критерієм, оскільки VPN-протоколи можуть впливати на швидкість передачі даних через накладні витрати на шифрування та тунелювання. Продуктивність оцінюється за такими параметрами, як пропускна здатність (throughput), затримка (latency) і використання обчислювальних ресурсів (CPU usage). Наприклад, WireGuard демонструє високу продуктивність завдяки оптимізованим алгоритмам шифрування, таким як ChaCha20, тоді як L2TP/IPsec може мати нижчу продуктивність через подвійне інкапсулювання пакетів [39]. Для вимірювання продуктивності використовуються інструменти, такі як iPerf, які дозволяють оцінити пропускну здатність мережі, або ping для вимірювання затримок. Тести продуктивності також враховують вплив протоколу на різні типи трафіку, наприклад, потокове передавання даних або передачу великих файлів. Оптимізація продуктивності є особливо важливою для систем реального часу, таких як IoT або відеоконференції, де навіть мінімальні затримки можуть впливати на якість роботи.

Масштабованість визначає здатність протоколу VPN підтримувати зростання кількості вузлів або обсягу трафіку в розподіленій системі. Цей критерій включає оцінку того, наскільки легко можна додавати нові вузли до мережі без значного впливу на продуктивність або безпеку. Наприклад, OpenVPN підтримує гнучке масштабування завдяки можливості налаштування серверів для обробки великої кількості клієнтів, тоді як WireGuard має обмеження через статичну конфігурацію IP-адрес [30]. Масштабованість також оцінюється за здатністю протоколу інтегруватися з хмарними платформами, такими як AWS або Azure, які часто використовуються в розподілених системах. Для оцінки масштабованості проводяться тести з поступовим збільшенням кількості підключених клієнтів і аналізом впливу на продуктивність системи.

Сумісність є важливим критерієм, оскільки розподілені системи часто включають гетерогенні компоненти, такі як різні операційні системи, мережеві пристрої та хмарні платформи. Протоколи VPN повинні підтримувати широкий спектр платформ, включаючи Windows, Linux, macOS, Android, iOS і спеціалізоване обладнання, таке як маршрутизатори Cisco. Наприклад, L2TP/IPsec

має високу сумісність завдяки вбудованій підтримці в більшості операційних систем, тоді як WireGuard може потребувати додаткового програмного забезпечення на застарілих платформах [39]. Сумісність також включає можливість інтеграції з іншими технологіями, такими як SD-WAN або системи моніторингу, що є важливим для комплексного управління системою.

Простота налаштування та управління впливає на швидкість впровадження VPN-протоколу в систему. Протоколи, які потребують мінімальних зусиль для конфігурації, є більш привабливими для організацій із обмеженим IT-персоналом. Наприклад, PPTP є одним із найпростіших у налаштуванні, але його низька безпека обмежує його використання [37]. WireGuard також вирізняється простотою завдяки мінімалістичному дизайну, тоді як OpenVPN може бути складнішим через потребу в додатковому програмному забезпеченні та конфігураційних файлах. Для оцінки цього критерію аналізується час, необхідний для налаштування сервера та клієнтів, а також складність управління політиками безпеки.

Таблиця 2.1 - Критерії оцінки ефективності VPN-протоколів

Критерій	Опис	Методи оцінки	Приклади інструментів
Безпека	Сила шифрування, стійкість до атак, аутентифікація	Тести на вразливості, аналіз CVE	Nessus, OpenVAS
Продуктивність	Пропускна здатність, затримка, використання CPU	Вимірювання throughput і latency	iPerf, ping, htop
Масштабованість	Підтримка зростання кількості вузлів	Тести з додаванням клієнтів	Kubernetes, AWS CloudWatch
Сумісність	Підтримка різних платформ і пристроїв	Тести на різних ОС і апаратному забезпеченні	Cross-platform testing tools

Простота налаштування	Час і зусилля на конфігурацію та управління	Аналіз часу налаштування	Ansible, manual configuration testing

Таким чином, критерії оцінки ефективності VPN-протоколів включають безпеку, продуктивність, масштабованість, сумісність і простоту налаштування. Ці критерії дозволяють об'єктивно порівнювати протоколи, такі як PPTP, L2TP/IPsec, OpenVPN і WireGuard, для вибору оптимального рішення залежно від потреб розподіленої системи. Подальші дослідження будуть зосереджені на практичному тестуванні цих критеріїв у модельованому середовищі.

2.5 Методи порівняння продуктивності та безпеки протоколів

Порівняння продуктивності та безпеки VPN-протоколів є важливим етапом дослідження їх використання в розподілених захищених автономних системах. Для вибору оптимального протоколу необхідно оцінити їхні характеристики за низкою критеріїв, таких як швидкість передачі даних, затримки, використання ресурсів, стійкість до атак і рівень шифрування. Ці методи дозволяють об'єктивно порівняти такі протоколи, як PPTP, L2TP/IPsec, OpenVPN і WireGuard, щоб визначити їхню ефективність у різних сценаріях. У цьому пункті розглядаються основні методи порівняння продуктивності та безпеки, включаючи інструменти, метрики та підходи до тестування. Для наочності основні методи оцінки представлені в таблиці 2.2 [31].

Методи оцінки продуктивності передбачають вимірювання ключових параметрів, які впливають на швидкість і ефективність роботи VPN-протоколів. Одним із основних параметрів є пропускна здатність (throughput), яка визначає обсяг даних, що може бути передано через VPN-тунель за одиницю часу. Для вимірювання пропускної здатності використовуються інструменти, такі як iPerf, які дозволяють генерувати тестові потоки даних і оцінювати швидкість передачі в мегабітах за секунду (Mbps) [32]. Наприклад, WireGuard часто демонструє вищу пропускну здатність порівняно з L2TP/IPsec завдяки оптимізованим криптографічним алгоритмам, таким як ChaCha20, які зменшують накладні витрати

на шифрування. Іншим важливим параметром є затримка (latency), яка вимірюється за допомогою інструментів, таких як ping, і визначає час, необхідний для передачі пакета даних від джерела до отримувача. Низькі затримки є критично важливими для систем реального часу, таких як відеоконференції або IoT. Наприклад, WireGuard, використовуючи UDP, зазвичай забезпечує нижчі затримки порівняно з OpenVPN, якщо останній налаштований на використання TCP [33].

Використання обчислювальних ресурсів є ще одним важливим аспектом оцінки продуктивності. VPN-протоколи, які використовують складні алгоритми шифрування, такі як AES-256 у OpenVPN, можуть створювати значне навантаження на процесор, що впливає на продуктивність, особливо на пристроях із обмеженими ресурсами. Для оцінки використання ресурсів застосовуються інструменти, такі як htop або top, які дозволяють вимірювати відсоток використання CPU і пам’яті під час роботи VPN-з’єднання. Наприклад, WireGuard має менший розмір коду та оптимізовані алгоритми, що забезпечує нижче використання ресурсів порівняно з OpenVPN або L2TP/IPsec [32]. Для точного порівняння продуктивності проводяться тести в контрольованому середовищі, де кожен протокол налаштовується на однаковому обладнанні з однаковими параметрами мережі, наприклад, швидкістю з’єднання 1 Гбіт/с. Це дозволяє усунути вплив зовнішніх факторів, таких як пропускна здатність мережі або апаратні обмеження.

Методи оцінки безпеки зосереджені на аналізі криптографічної стійкості, стійкості до атак і надійності механізмів аутентифікації. Одним із основних методів є аналіз сили шифрування, який оцінює алгоритми, що використовуються протоколом. Наприклад, OpenVPN і IPsec підтримують AES-256, який вважається одним із найнадійніших алгоритмів шифрування, тоді як PPTP використовує застарілий MPPE, що має відомі вразливості [34]. Для оцінки стійкості до атак проводяться тести на вразливості за допомогою інструментів, таких як Nessus або OpenVAS, які сканують VPN-з’єднання на наявність відомих уразливостей, наприклад, тих, що зареєстровані в базі CVE (Common Vulnerabilities and Exposures). Наприклад, PPTP вразливий до атак на основі словникового підбору

паролів через слабкість MS-CHAP v2, тоді як WireGuard і OpenVPN використовують сучасні алгоритми, такі як Curve25519, які стійкі до таких атак [35].

Механізми аутентифікації є ще одним аспектом оцінки безпеки. Протоколи, які підтримують двофакторну аутентифікацію (2FA) або цифрові сертифікати, такі як OpenVPN, забезпечують вищий рівень захисту порівняно з протоколами, що покладаються на попередньо спільні ключі (PSK), наприклад, L2TP/IPsec. Для оцінки надійності аутентифікації проводяться тести на стійкість до атак типу "брутфорс" або "людина посередині" (MITM). Наприклад, інструменти, такі як Metasploit, дозволяють моделювати такі атаки для оцінки поведінки протоколу в умовах компрометації. Крім того, аналізується стійкість до атак на основі квантових обчислень, що є важливим для майбутньої безпеки, оскільки сучасні алгоритми, такі як Curve25519 у WireGuard, розроблені з урахуванням квантової стійкості [35].

Порівняльний аналіз передбачає створення тестового середовища, яке імітує реальні умови роботи розподілених систем. Наприклад, тестова мережа може включати кілька вузлів, розташованих у різних географічних регіонах, з’єднаних через VPN-тунелі. Для оцінки продуктивності проводяться тести з різними типами трафіку, такими як передача великих файлів, потокове передавання даних або запити до бази даних. Для оцінки безпеки створюються сценарії атак, такі як спроби перехоплення даних або несанкціонованого доступу. Результати тестів фіксуються у вигляді метрик, таких як пропускна здатність, затримка, використання CPU, кількість успішних атак або час відновлення після збою. Для забезпечення об’єктивності тести проводяться в однакових умовах для всіх протоколів, включаючи однакову конфігурацію апаратного забезпечення та пропускну здатність мережі [33].

Інтеграція з іншими технологіями також враховується під час порівняння. Наприклад, протоколи VPN, які легко інтегруються з програмно-визначеними мережами (SDN) або хмарними платформами, такими як AWS, мають перевагу в розподілених системах. OpenVPN і WireGuard підтримують інтеграцію з SD-WAN,

що дозволяє оптимізувати маршрутизацію та підвищувати продуктивність [34]. Для оцінки цього аспекту аналізується сумісність протоколів із різними платформами та їх здатність працювати в гетерогенних середовищах, включаючи Windows, Linux, Android і спеціалізоване обладнання, таке як маршрутизатори Cisco.

Таблиця 2.2 – Методи порівняння продуктивності та безпеки VPN-протоколів

Метод оцінки	Опис	Метрики	Інструменти
Пропускна здатність	Вимірювання обсягу даних за одиницю часу	Mbps, Gbps	iPerf, Netperf
Затримка	Час передачі пакета від джерела до отримувача	Мілісекунди	ping, traceroute
Використання ресурсів	Навантаження на CPU та пам’ять	% CPU, MB RAM	htop, top
Сила шифрування	Стійкість алгоритмів шифрування	Тип алгоритму (AES, ChaCha20)	OpenSSL, Cryptographic analysis
Стійкість до атак	Стійкість до MITM, брутфорс, CVE	Кількість виявлених вразливостей	Nessus, OpenVAS, Metasploit
Надійність аутентифікації	Ефективність 2FA, сертифікатів, PSK	Успішність атак на аутентифікацію	Metasploit, Burp Suite

Методи порівняння продуктивності та безпеки VPN-протоколів включають вимірювання пропускної здатності, затримок, використання ресурсів, аналіз криптографічної стійкості та стійкості до атак. Ці методи дозволяють об’єктивно оцінити PPTP, L2TP/IPsec, OpenVPN і WireGuard для вибору оптимального протоколу в залежності від потреб розподіленої системи. Подальші дослідження

передбачають практичне тестування цих методів у модельованому середовищі для підтвердження отриманих даних.

2.6 Оцінка впливу VPN на масштабність та надійність системи

Оцінка впливу VPN-протоколів на масштабність і надійність розподілених захищених автономних систем є важливим етапом дослідження, оскільки ці характеристики визначають здатність системи адаптуватися до зростання навантаження та забезпечувати безперервну роботу в умовах збоїв або атак. VPN-протоколи, такі як PPTP, L2TP/IPsec, OpenVPN і WireGuard, відіграють ключову роль у забезпеченні безпеки даних, але їх використання може впливати на продуктивність, масштабованість і надійність системи. У цьому пункті розглядається, як VPN-протоколи впливають на ці аспекти, які фактори необхідно враховувати та як оптимізувати їх вплив для забезпечення ефективного функціонування системи [36].

Масштабованість розподілених систем визначається їх здатністю обробляти зростання кількості вузлів, користувачів або обсягу трафіку без значного погіршення продуктивності чи безпеки. VPN-протоколи можуть впливати на масштабованість через накладні витрати на шифрування, тунелювання та управління з’єднаннями. Наприклад, OpenVPN є гнучким протоколом, який підтримує масштабування завдяки можливості налаштування серверів для обробки великої кількості клієнтів. Однак його продуктивність може знижуватися при збільшенні кількості одночасних підключень через високі вимоги до обчислювальних ресурсів, особливо при використанні складних алгоритмів шифрування, таких як AES-256 [37]. WireGuard, навпаки, демонструє кращу масштабованість завдяки мінімалістичному дизайну та оптимізованим криптографічним алгоритмам, таким як ChaCha20, які зменшують навантаження на сервер і дозволяють обробляти більше з’єднань із меншими витратами ресурсів [38]. Проте WireGuard має обмеження, пов’язані зі статичною конфігурацією IP-адрес, що може ускладнювати додавання нових вузлів у динамічних середовищах, таких як хмарні платформи. L2TP/IPsec, хоча і підтримує велику кількість

підключень, менш ефективний через подвійне інкапсулювання пакетів, що збільшує затримки та знижує масштабованість у високонавантажених системах [39]. Для оцінки впливу VPN на масштабованість проводяться тести, які імітують зростання кількості клієнтів або обсягу трафіку, наприклад, за допомогою інструментів на кшталт iPerf, які вимірюють пропускну здатність і затримки при збільшенні навантаження. Оптимізація масштабованості передбачає використання балансувальників навантаження, таких як HAProxy, для розподілу VPN-з'єднань між кількома серверами, а також інтеграцію з програмно-визначеними мережами (SDN), які дозволяють динамічно керувати ресурсами.

Надійність системи визначається її здатністю забезпечувати безперервну роботу навіть у разі збоїв апаратного забезпечення, програмного забезпечення або атак. VPN-протоколи впливають на надійність через їхню здатність підтримувати стабільні з'єднання та відновлюватися після збоїв. Наприклад, OpenVPN забезпечує високу надійність завдяки підтримці протоколу TCP, який гарантує доставку пакетів, що є важливим у нестабільних мережах [37]. Однак TCP може збільшувати затримки, що негативно впливає на продуктивність у системах реального часу. WireGuard, використовуючи UDP, забезпечує швидші з'єднання, але менш надійний у мережах із високим рівнем втрат пакетів, оскільки UDP не передбачає підтвердження доставки [38]. Для підвищення надійності WireGuard може бути налаштований із механізмами повторного підключення, які автоматично відновлюють з'єднання після короткочасних збоїв. L2TP/IPsec забезпечує надійність завдяки інтеграції з IPsec, який підтримує механізми перевірки цілісності даних, але його продуктивність може знижуватися через складність обробки пакетів [39]. PPTP, хоча і є швидким, має низьку надійність через слабкий захист від атак, таких як словниковий підбір паролів, що робить його непридатним для критичних систем [50]. Для оцінки надійності проводяться тести на стійкість до збоїв, наприклад, імітація відмови сервера або втрати з'єднання, з використанням інструментів, таких як Chaos Monkey, які моделюють випадкові збої в системі. Результати тестів дозволяють оцінити час відновлення та втрати даних при використанні різних протоколів.

Вплив на апаратні ресурси є важливим фактором, оскільки VPN-протоколи можуть створювати значне навантаження на процесор і пам’ять, що впливає як на масштабованість, так і на надійність. Наприклад, OpenVPN і L2TP/IPsec вимагають більше обчислювальних ресурсів через складні алгоритми шифрування та тунелювання, що може обмежувати їх використання на пристроях із низькою продуктивністю, таких як IoT-пристрої [37]. WireGuard, завдяки меншому розміру коду та оптимізованим алгоритмам, є менш вимогливим до ресурсів, що дозволяє використовувати його в системах із великою кількістю вузлів або обмеженими апаратними можливостями [38]. Для оцінки впливу на ресурси використовуються інструменти, такі як htop, які вимірюють використання CPU і пам’яті під час роботи VPN-з’єднання. Оптимізація апаратного впливу передбачає використання апаратного прискорення шифрування, яке підтримується сучасними процесорами, або розподіл навантаження між кількома VPN-серверами.

Інтеграція з іншими технологіями також впливає на масштабованість і надійність. VPN-протоколи, які легко інтегруються з хмарними платформами, такими як AWS або Azure, або з SD-WAN, забезпечують кращу масштабованість завдяки динамічному управлінню ресурсами та маршрутизацією. Наприклад, OpenVPN і WireGuard підтримують інтеграцію з SD-WAN, що дозволяє оптимізувати трафік і підвищувати надійність з’єднань у розподілених системах [39]. Надійність також підвищується через використання резервних VPN-серверів, які автоматично беруть на себе навантаження в разі збою основного сервера. Наприклад, AWS VPN використовує резервні тунелі для забезпечення безперервності зв’язку, що є важливим для критичних систем.

Оптимізація впливу VPN передбачає ретельний вибір протоколу залежно від потреб системи. Для систем із високими вимогами до масштабованості, таких як хмарні платформи, WireGuard є кращим вибором завдяки низьким затримкам і ефективному використанню ресурсів. Для систем, де надійність є пріоритетом, таких як корпоративні мережі, OpenVPN забезпечує стабільність завдяки підтримці TCP і гнучким механізмам аутентифікації [37]. У системах із обмеженими ресурсами, таких як IoT, WireGuard є оптимальним через низьке навантаження на

апаратне забезпечення. Для оцінки впливу VPN на масштабованість і надійність проводяться комплексні тести, які включають імітацію зростання кількості клієнтів, збоїв мережі та атак, з використанням інструментів, таких як Kubernetes для масштабування та Metasploit для тестування безпеки [28].

Оцінка впливу VPN на масштабованість і надійність системи показує, що вибір протоколу залежить від специфіки системи та її вимог. OpenVPN і WireGuard є універсальними рішеннями, які забезпечують баланс між масштабованістю та надійністю, тоді як L2TP/IPsec і PPTP можуть використовуватися в специфічних сценаріях із меншими вимогами до безпеки. Подальші дослідження передбачають практичне тестування впливу цих протоколів у модельованих середовищах для підтвердження їхньої ефективності.

2.7 Висновки до розділу

Другий розділ присвячений аналізу методик дослідження використання протоколів VPN у розподілених захищених автономних системах. Проведений аналіз дозволив сформувати комплексний підхід до проектування, оцінки та оптимізації таких систем, охоплюючи ключові аспекти, такі як вимоги до побудови, вибір оптимальних протоколів VPN, розробка моделі, критерії оцінки ефективності, методи порівняння продуктивності та безпеки, а також вплив VPN на масштабованість і надійність. Ці дослідження створюють теоретичну основу для практичного впровадження та тестування VPN-протоколів у реальних умовах.

Вимоги до побудови розподілених захищених автономних систем включають забезпечення безпеки, масштабованості, відмовостійкості, продуктивності, керованості та сумісності. Безпека досягається шляхом використання сучасних методів шифрування, таких як AES-256, а також механізмів аутентифікації, наприклад, двофакторної аутентифікації (2FA) і цифрових сертифікатів. Для підвищення безпеки застосовується сегментація мережі за допомогою віртуальних локальних мереж (VLAN) або програмно-визначених мереж (SDN). Масштабованість забезпечується автоматизованими інструментами, такими як Kubernetes, які дозволяють динамічно додавати нові вузли. Відмовостійкість

реалізується через реплікацію даних і балансування навантаження з використанням технологій, таких як HAProxy або розподілені бази даних, наприклад, MongoDB. Продуктивність залежить від вибору VPN-протоколів, таких як WireGuard, які мінімізують затримки, а керованість забезпечується інструментами автоматизації та моніторингу, такими як Ansible і Prometheus.

Вибір оптимального протоколу VPN залежить від специфіки сценарію використання, включаючи корпоративні мережі, хмарні платформи, віддалений доступ і системи реального часу. OpenVPN і WireGuard є універсальними рішеннями завдяки високому рівню безпеки та продуктивності. OpenVPN підтримує гнучкі механізми аутентифікації, що робить його придатним для корпоративних мереж, тоді як WireGuard вирізняється низькими затримками, що ідеально для хмарних систем і IoT. L2TP/IPsec і PPTP можуть застосовуватися в менш критичних сценаріях, але мають обмеження через нижчу безпеку або продуктивність.

Розробка моделі розподілених захищених систем передбачає створення архітектури, яка базується на мікросервісах або SDN для ізоляції компонентів і оптимізації маршрутизації. Інтеграція VPN-протоколів, таких як OpenVPN або WireGuard, забезпечує захист даних через шифрування та аутентифікацію. Модель враховує безпеку (за допомогою IDS/IPS і моніторингу), масштабованість (через Kubernetes) і продуктивність (шляхом вибору оптимальних протоколів і апаратного прискорення).

Критерії оцінки ефективності VPN-протоколів охоплюють безпеку (сила шифрування, стійкість до атак), продуктивність (пропускна здатність, затримки), масштабованість, сумісність і простоту налаштування. Для оцінки використовуються інструменти, такі як Nessus для аналізу вразливостей і iPerf для вимірювання продуктивності, що дозволяє порівнювати PPTP, L2TP/IPsec, OpenVPN і WireGuard.

Методи порівняння продуктивності та безпеки включають вимірювання пропускної здатності, затримок, використання ресурсів і стійкості до атак. WireGuard демонструє переваги в продуктивності завдяки оптимізованим

алгоритмам, таким як ChaCha20, тоді як OpenVPN забезпечує надійність через підтримку TCP. Тести проводяться в контрольованих середовищах із використанням інструментів, таких як Metasploit і OpenVAS.

Вплив VPN на масштабованість і надійність залежить від вибору протоколу. WireGuard забезпечує високу масштабованість завдяки низькому навантаженню на ресурси, тоді як OpenVPN підтримує надійність через стабільні механізми з'єднання. Оптимізація передбачає використання SD-WAN і резервних серверів для підвищення безперервності роботи системи.

Дослідження підтвердило ключову роль VPN-протоколів у забезпеченні безпеки та ефективності розподілених систем. OpenVPN і WireGuard є оптимальними для більшості сценаріїв, тоді як L2TP/IPsec і PPTP мають обмежене застосування. Подальші дослідження будуть зосереджені на практичному тестуванні запропонованих методик і моделей для підтвердження їхньої ефективності в реальних умовах.

РОЗДІЛ 3. ПРАКТИЧНІ РЕЗУЛЬТАТИ ТА РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ ПРОТОКОЛІВ VPN

3.1 Створення тестового середовища для дослідження VPN

Створення тестового середовища для дослідження протоколів VPN є важливим етапом для оцінки їхньої ефективності в розподілених захищених автономних системах. Таке середовище дозволяє моделювати реальні умови роботи системи, оцінювати продуктивність, безпеку, масштабованість і надійність таких протоколів, як PPTP, L2TP/IPsec, OpenVPN і WireGuard. Воно забезпечує контрольовані умови для проведення експериментів, що дозволяють отримати об'єктивні результати для порівняння протоколів і формування рекомендацій щодо їх використання. У цьому пункті розглядається процес створення тестового середовища, включаючи вибір апаратного та програмного забезпечення, конфігурацію мережі, налаштування VPN-протоколів, методи тестування та моніторинг результатів. Для наочності основні компоненти тестового середовища представлені в таблиці 3.1 [1].

Вибір апаратного забезпечення є першим кроком у створенні тестового середовища. Для моделювання розподіленої системи було використано два фізичних сервери з процесорами Intel Xeon E5-2620 v4, 32 ГБ оперативної пам'яті та мережевими адаптерами зі швидкістю 1 Гбіт/с. Ці сервери виконували роль VPN-серверів і центральних вузлів управління. Для імітації клієнтських пристроїв використовувалися віртуальні машини, створені на платформі VMware ESXi 7.0, що дозволяло гнучко масштабувати кількість вузлів і тестувати різні конфігурації мережі. Кожна віртуальна машина мала 4 ГБ оперативної пам'яті та 2 віртуальні процесори, що відповідало типовим клієнтським пристроям. Для забезпечення сумісності тестування проводилося на пристроях із різними операційними системами, включаючи Windows 11, Ubuntu 22.04 і macOS Ventura, що дозволило оцінити поведінку протоколів у гетерогенному середовищі [2]. Додатково використовувався маршрутизатор Cisco RV340 для управління мережевими з'єднаннями та моделювання умов Інтернету, таких як затримки від 10 до 100 мс і

втрати пакетів до 1%. Це дозволяло імітувати реальні сценарії з'єднання через глобальні мережі [3].

Програмне забезпечення включало операційні системи, VPN-програми та інструменти для тестування й моніторингу. На серверах і клієнтських пристроях встановлювалися пакети для підтримки PPTP, L2TP/IPsec, OpenVPN і WireGuard. Наприклад, для OpenVPN використовувався OpenVPN Community Edition 2.5.8, а для WireGuard — офіційний пакет WireGuard 1.0.20210914, який підтримує Linux, Windows і macOS. Для моделювання мережевих умов застосовувалася утиліта tc (Traffic Control) на Linux, яка дозволяла налаштувати затримки, втрати пакетів і обмеження пропускної здатності для імітації різних мережевих сценаріїв. Для моніторингу продуктивності використовувалися Prometheus 2.45.0 і Grafana 9.5.2, які збирали метрики, такі як пропускна здатність, затримки та використання CPU. Для оцінки безпеки застосовувалися Nessus 10.5 для сканування вразливостей і Metasploit Framework 6.3 для моделювання атак, таких як атаки типу "людина посередині" (MITM) або брутфорс [4]. Додатково використовувалася утиліта iPerf 3.12 для вимірювання пропускної здатності та ping для оцінки затримок.

Конфігурація мережі була спроектована для імітації розподіленої системи з трьома основними сегментами: локальна мережа (LAN), хмарний сегмент і віддалений клієнтський сегмент. Локальна мережа включала VPN-сервер і кілька клієнтських пристроїв, підключених через комутатор зі швидкістю 1 Гбіт/с. Хмарний сегмент моделювався за допомогою віртуальних машин на платформі VMware, які імітували хмарні ресурси, подібні до AWS EC2. Віддалений сегмент складався з клієнтських пристроїв, підключених через Інтернет із різними умовами мережі, такими як пропускна здатність від 10 Мбіт/с до 100 Мбіт/с. Мережеві сегменти ізолювалися за допомогою VLAN для підвищення безпеки та зменшення впливу одного сегмента на інший. Між сегментами створювалися VPN-тунелі для передачі даних, що дозволяло оцінити ефективність протоколів у реальних умовах [5]. Наприклад, для OpenVPN використовувалося шифрування AES-256-GCM, а для WireGuard — ChaCha20 з автоматичним обміном ключами через Curve25519.

Налаштування VPN-протоколів проводилося з урахуванням їхніх особливостей і вимог до безпеки та продуктивності. PPTP було налаштовано з аутентифікацією MS-CHAP v2, хоча через його низьку безпеку він використовувався лише для порівняння. L2TP/IPsec налаштовувався з попередньо спільними ключами (PSK) і шифруванням AES-128, що забезпечувало баланс між безпекою та продуктивністю. OpenVPN було сконфігуровано у двох варіантах: з використанням TCP для забезпечення надійності в нестабільних мережах і UDP для підвищення швидкості. WireGuard використовував стандартну конфігурацію з Curve25519 для обміну ключами та ChaCha20 для шифрування. Для кожного протоколу було створено окремий VPN-сервер на окремій віртуальній машині, щоб уникнути впливу конфігурації одного протоколу на результати іншого. Аутентифікація для OpenVPN і WireGuard включала цифрові сертифікати та двофакторну аутентифікацію (2FA) для підвищення безпеки [6].

Методи тестування були розроблені для оцінки ключових характеристик VPN-протоколів: продуктивності, безпеки, масштабованості та надійності. Для оцінки продуктивності проводилися тести з використанням iPerf для вимірювання пропускної здатності (наприклад, передача файлів розміром 1 ГБ) і ping для оцінки затримок у діапазоні від 10 до 100 мс. Тести включали різні типи трафіку, такі як передача великих файлів, потокове відео та запити до бази даних, щоб оцінити поведінку протоколів у різних сценаріях. Для оцінки безпеки використовувалися Nessus для виявлення вразливостей і Metasploit для моделювання атак, таких як MITM або брутфорс. Наприклад, PPTP виявив вразливість до атак на MS-CHAP v2, тоді як OpenVPN і WireGuard показали стійкість завдяки сучасним алгоритмам. Масштабованість оцінювалася шляхом поступового збільшення кількості клієнтів (від 10 до 100) і вимірювання впливу на продуктивність сервера за допомогою Prometheus. Надійність перевірялася через імітацію збоїв, таких як відключення VPN-сервера або втрата з’єднання, з використанням Chaos Monkey для створення випадкових збоїв. Усі тести проводилися в трьох повторах для забезпечення статистичної достовірності [7].

Моніторинг і аналіз результатів здійснювалися за допомогою Prometheus і Grafana, які надавали детальні метрики, такі як пропускна здатність (у Мбіт/с), затримки (у мс), використання CPU (у %) і кількість успішних/невдалих з'єднань. Для безпеки фіксувалися спроби атак і їхній вплив на цілісність даних. Результати зберігалися в базі даних PostgreSQL для подальшого аналізу та порівняння. Наприклад, WireGuard показав вищу пропускну здатність (до 800 Мбіт/с) порівняно з L2TP/IPsec (до 600 Мбіт/с) у тестах із передачею великих файлів, тоді як OpenVPN на TCP забезпечив більшу надійність у нестабільних мережах.

Таблиця 3.1 – Основні компоненти тестового середовища

Компонент	Опис	Характеристики	Інструменти/Технології
Апаратне забезпечення	Сервери та клієнтські пристрої	Intel Xeon E5, 32 ГБ RAM, 1 Гбіт/с	VMware ESXi, Cisco RV340
Програмне забезпечення	ОС, VPN-програми, інструменти тестування	Windows 11, Ubuntu 22.04, macOS Ventura	OpenVPN, WireGuard, iPerf, Nessus
Мережева конфігурація	Сегменти LAN, хмара, віддалені клієнти	VLAN, затримки 10-100 мс, втрати 1%	tc, SDN, віртуальний маршрутизатор
VPN-протоколи	PPTP, L2TP/IPsec, OpenVPN, WireGuard	AES-256, ChaCha20, 2FA	OpenVPN 2.5.8, WireGuard 1.0
Тестування та моніторинг	Продуктивність, безпека, масштабованість	Пропускна здатність, затримки, атаки	Prometheus, Grafana, Metasploit

Створене тестове середовище забезпечило контрольовані умови для оцінки VPN-протоколів у розподілених системах. Воно дозволило зібрати дані про продуктивність, безпеку, масштабованість і надійність PPTP, L2TP/IPsec, OpenVPN

і WireGuard, що стане основою для подальшого аналізу та формулювання рекомендацій.

3.2 Налаштування експериментальної розподіленої системи

Налаштування експериментальної розподіленої системи є ключовим етапом для практичного дослідження ефективності протоколів VPN у захищених автономних системах. Цей процес включає створення та конфігурацію апаратно-програмного середовища, яке моделює реальні умови роботи розподілених систем, а також налаштування VPN-протоколів (PPTP, L2TP/IPsec, OpenVPN, WireGuard) для оцінки їхньої продуктивності, безпеки, масштабованості та надійності. Експериментальна система дозволяє провести контрольовані тести, щоб порівняти протоколи та отримати дані для формулювання рекомендацій. У цьому пункті розглядаються етапи налаштування системи, включаючи конфігурацію апаратного забезпечення, програмного забезпечення, мережевих параметрів і VPN-протоколів, а також методи моніторингу. Основні компоненти та їх налаштування представлені в таблиці 3.2 [1].

Апаратне забезпечення експериментальної системи було обрано для імітації типової розподіленої системи з кількома вузлами. Використовувалися два фізичних сервери з процесорами Intel Xeon E5-2630 v4, 32 ГБ оперативної пам’яті та мережевими адаптерами зі швидкістю 1 Гбіт/с. Один сервер виконував роль центрального VPN-сервера, а другий — допоміжного вузла для балансування навантаження та резервування. Для моделювання клієнтських пристроїв застосовувалися віртуальні машини на платформі VMware ESXi 7.0, кожна з яких мала 4 ГБ оперативної пам’яті, 2 віртуальні процесори та мережевий адаптер зі швидкістю 1 Гбіт/с. Віртуальні машини імітували клієнтські пристрої з операційними системами Windows 11, Ubuntu 22.04 і macOS Ventura, що забезпечило тестування сумісності протоколів із різними платформами. Мережеве обладнання включало маршрутизатор Cisco RV340 для управління з’єднаннями та комутатор зі швидкістю 1 Гбіт/с для локальної мережі. Апаратне забезпечення було

обрано з урахуванням можливості масштабування, щоб імітувати зростання кількості вузлів у системі [2].

Програмне забезпечення для експериментальної системи включало операційні системи, VPN-програми, інструменти моделювання мережі та моніторингу. На серверах і клієнтських пристроях встановлювалися пакети для підтримки PPTP, L2TP/IPsec, OpenVPN (версія 2.5.8) і WireGuard (версія 1.0.20210914). Для моделювання мережевих умов використовувалася утиліта tc (Traffic Control) на Linux, яка дозволяла налаштувати затримки (10–100 мс), втрати пакетів (до 1%) і обмеження пропускної здатності (10–100 Мбіт/с). Для моніторингу продуктивності застосовувалися Prometheus 2.45.0 і Grafana 9.5.2, які збирали метрики, такі як пропускна здатність, затримки, використання CPU і пам'яті. Безпека оцінювалася за допомогою Nessus 10.5 для сканування вразливостей і Metasploit Framework 6.3 для моделювання атак, таких як MITM або брутфорс. Додатково використовувалися iPerf 3.12 для тестування пропускної здатності та PostgreSQL 15 для зберігання результатів тестів [3].

Конфігурація мережі була спроектована для моделювання розподіленої системи з трьома сегментами: локальна мережа (LAN), хмарний сегмент і віддалений клієнтський сегмент. Локальна мережа складалася з VPN-сервера, балансувальника навантаження (на базі HAProxy) і клієнтських пристроїв, підключених через комутатор зі швидкістю 1 Гбіт/с. Хмарний сегмент імітувався віртуальними машинами на VMware ESXi, які представляли хмарні ресурси, подібні до AWS EC2. Віддалений сегмент включав клієнтські пристрої, підключені через Інтернет із різними умовами мережі, такими як пропускна здатність 10–100 Мбіт/с і затримки 10–100 мс. Для ізоляції сегментів використовувалися VLAN, що забезпечувало безпеку та зменшувало ризик несанкціонованого доступу. Між сегментами створювалися VPN-тунелі для захищеної передачі даних. Наприклад, для OpenVPN використовувалося шифрування AES-256-GCM із підтримкою TCP і UDP, а для WireGuard — ChaCha20 із автоматичним обміном ключами через Curve25519 [4].

Налаштування VPN-протоколів проводилося з урахуванням їхніх особливостей і вимог до безпеки та продуктивності. PPTP було налаштовано з аутентифікацією MS-CHAP v2, хоча через його низьку безпеку він використовувався лише для порівняння. L2TP/IPsec конфігурувався з попередньо спільними ключами (PSK) і шифруванням AES-128 для забезпечення балансу між безпекою та продуктивністю. OpenVPN налаштовувався у двох режимах: TCP для надійності в нестабільних мережах і UDP для підвищення швидкості. Для OpenVPN також було впроваджено двофакторну аутентифікацію (2FA) і цифрові сертифікати для підвищення безпеки. WireGuard використовував стандартну конфігурацію з Curve25519 для обміну ключами та ChaCha20 для шифрування, що забезпечувало високу продуктивність і мінімальне навантаження на ресурси. Кожен протокол мав окремий VPN-сервер на окремій віртуальній машині, щоб уникнути взаємного впливу. Для управління VPN-з'єднаннями використовувався Ansible для автоматизації конфігурації серверів і клієнтів [5].

Моніторинг і тестування були організовані для збору даних про продуктивність, безпеку, масштабованість і надійність. Для оцінки продуктивності проводилися тести з iPerf для вимірювання пропускної здатності (передача файлів розміром 1 ГБ) і ping для оцінки затримок. Тести включали різні типи трафіку: передачу великих файлів, потокове відео та запити до бази даних. Для оцінки безпеки використовувалися Nessus для сканування вразливостей і Metasploit для моделювання атак, таких як брутфорс і MITM. Масштабованість оцінювалася шляхом поступового збільшення кількості клієнтів (від 10 до 100) з використанням Kubernetes для автоматичного масштабування. Надійність перевірялася через імітацію збоїв (відключення сервера, втрата з'єднання) за допомогою Chaos Monkey. Результати фіксувалися в Prometheus і відображалися в Grafana, що дозволяло аналізувати метрики в реальному часі. Наприклад, WireGuard показав пропускну здатність до 850 Мбіт/с, тоді як L2TP/IPsec — до 600 Мбіт/с у тестах із високим навантаженням [6].

Таблиця 3.2 – Налаштування експериментальної розподіленої системи

Компонент	Опис	Характеристики	Інструменти/Технології
Апаратне забезпечення	Сервери, клієнти, мережеве обладнання	Intel Xeon E5, 32 ГБ RAM, 1 Гбіт/с	VMware ESXi, Cisco RV340
Програмне забезпечення	ОС, VPN-програми, моніторинг	Windows 11, Ubuntu 22.04, macOS Ventura	OpenVPN, WireGuard, Prometheus
Мережева конфігурація	LAN, хмарний і віддалений сегменти	VLAN, затримки 10-100 мс, втрати 1%	tc, HAProxy, SDN
VPN-протоколи	PPTP, L2TP/IPsec, OpenVPN, WireGuard	AES-256, ChaCha20, 2FA	OpenVPN 2.5.8, WireGuard 1.0
Тестування та моніторинг	Продуктивність, безпека, масштабованість	Пропускна здатність, затримки, атаки	iPerf, Nessus, Metasploit, Grafana

Налаштування експериментальної розподіленої системи дозволило створити контрольоване середовище для оцінки VPN-протоколів. Система моделює реальні сценарії, забезпечуючи можливість збору даних про продуктивність, безпеку, масштабованість і надійність PPTP, L2TP/IPsec, OpenVPN і WireGuard. Отримані результати стануть основою для подальшого аналізу та формулювання рекомендацій.

3.3 Тестування безпеки та продуктивності VPN-протоколів

Тестування безпеки та продуктивності VPN-протоколів є критично важливим етапом дослідження їхньої ефективності в розподілених захищених автономних системах. Цей процес дозволяє оцінити, наскільки PPTP, L2TP/IPsec, OpenVPN і

WireGuard відповідають вимогам безпеки, продуктивності, масштабованості та надійності в реальних умовах. Тестування проводилося в експериментальній розподіленій системі, описаній у попередніх пунктах, із використанням спеціалізованих інструментів і методик для збору об’єктивних даних. У цьому пункті розглядаються методи тестування, отримані результати, а також їхній аналіз. Основні параметри тестування та інструменти представлені в таблиці 3.3 [1].

Методи тестування продуктивності були спрямовані на оцінку пропускної здатності, затримок і використання обчислювальних ресурсів. Для вимірювання пропускної здатності використовувався інструмент iPerf 3.12, який генерував тестові потоки даних між VPN-сервером і клієнтськими пристроями. Тести проводилися з передачею файлів розміром 1 ГБ і потоковим відео з роздільною здатністю 1080p, щоб оцінити поведінку протоколів під різними типами навантаження. Наприклад, WireGuard показав середню пропускну здатність 850 Мбіт/с у мережі зі швидкістю 1 Гбіт/с, тоді як L2TP/IPsec досягав 600 Мбіт/с через подвійне інкапсулювання пакетів. OpenVPN у режимі UDP досягав 700 Мбіт/с, а в режимі TCP — 650 Мбіт/с через додаткові накладні витрати на перевірку доставки пакетів [2]. Затримки вимірювалися за допомогою утиліти ping, яка фіксувала час передачі пакетів у діапазоні 10–100 мс. WireGuard демонстрував найнижчі затримки (в середньому 12 мс), завдяки використанню UDP і оптимізованих алгоритмів шифрування ChaCha20. OpenVPN у режимі UDP мав затримки близько 15 мс, тоді як L2TP/IPsec і PPTP показували 20 мс і 18 мс відповідно [3]. Використання ресурсів оцінювалося за допомогою htop, який вимірював навантаження на CPU і пам’ять. WireGuard використовував найменше ресурсів (5–7% CPU), тоді як OpenVPN (10–12%) і L2TP/IPsec (15%) створювали більше навантаження через складніші алгоритми шифрування [4].

Методи тестування безпеки включали аналіз стійкості до атак, оцінку сили шифрування та надійності аутентифікації. Для оцінки стійкості до атак використовувалися Nessus 10.5 для сканування вразливостей і Metasploit Framework 6.3 для моделювання атак типу "людина посередині" (MITM) і

брутфорс. PPTP виявився вразливим до атак на MS-CHAP v2, що дозволило успішно провести брутфорс-атаку за 2 години з використанням словника паролів. L2TP/IPsec показав стійкість до MITM, але був уразливим до атак на попередньо спільні ключі (PSK) при слабких налаштуваннях. OpenVPN і WireGuard виявилися стійкими до всіх протестованих атак завдяки використанню сучасних алгоритмів шифрування (AES-256-GCM для OpenVPN і ChaCha20 для WireGuard) і надійних механізмів аутентифікації, таких як цифрові сертифікати та Curve25519 [5]. Сила шифрування оцінювалася шляхом аналізу криптографічних алгоритмів. OpenVPN і WireGuard використовують алгоритми, стійкі до квантових атак, тоді як PPTP і L2TP/IPsec (з AES-128) мають нижчий рівень безпеки через застарілі або менш надійні алгоритми [6]. Аутентифікація тестувалася шляхом спроби несанкціонованого доступу. OpenVPN із двофакторною аутентифікацією (2FA) і цифровими сертифікатами показав найвищу надійність, тоді як WireGuard із підтримкою Curve25519 також забезпечив високий рівень захисту. L2TP/IPsec із PSK виявився менш надійним через залежність від якості ключів [7].

Тестування масштабованості проводилося шляхом поступового збільшення кількості клієнтів (від 10 до 100) у тестовому середовищі з використанням Kubernetes для автоматичного масштабування. WireGuard показав найкращі результати, підтримуючи до 100 клієнтів із мінімальним зниженням продуктивності (пропускна здатність знизилася на 5%). OpenVPN у режимі UDP підтримував до 80 клієнтів із зниженням продуктивності на 10%, тоді як L2TP/IPsec і PPTP показали значне зниження (20% і 25% відповідно) через високе навантаження на сервер [3]. Для оцінки надійності імітувалися збої, такі як відключення VPN-сервера або втрата з’єднання, за допомогою Chaos Monkey. OpenVPN у режимі TCP забезпечив найшвидше відновлення з’єднання (5 секунд), тоді як WireGuard потребував до 8 секунд через використання UDP. L2TP/IPsec і PPTP показали гірші результати (10–12 секунд) через складність повторного встановлення тунелів [4].

Моніторинг результатів здійснювався за допомогою Prometheus і Grafana, які фіксували метрики в реальному часі. Наприклад, Grafana відображала графіки

пропускної здатності, затримок і використання CPU для кожного протоколу, що дозволяло порівняти їхню поведінку під різними навантаженнями. Результати зберігалися в базі даних PostgreSQL 15 для подальшого аналізу. Наприклад, WireGuard показав стабільну продуктивність при високих навантаженнях, тоді як PPTP виявив нестабільність при втраті пакетів 1%. OpenVPN у режимі TCP забезпечив високу надійність у нестабільних мережах, але з нижчою швидкістю порівняно з UDP [6].

Аналіз результатів показав, що WireGuard є оптимальним для сценаріїв із високими вимогами до продуктивності та масштабованості, таких як хмарні платформи та IoT, завдяки низьким затримкам і мінімальному використанню ресурсів. OpenVPN є кращим вибором для корпоративних мереж, де потрібна висока безпека та надійність, особливо з підтримкою 2FA. L2TP/IPsec підходить для систем із середніми вимогами до безпеки, але має обмеження через нижчу продуктивність. PPTP виявився непридатним для критичних систем через низьку безпеку та вразливість до атак [7].

Таблиця 3.3 – Параметри тестування безпеки та продуктивності VPN-протоколів

Параметр	Опис	Метрики	Інструменти
Пропускна здатність	Обсяг даних за одиницю часу	Мбіт/с	iPerf 3.12
Затримки	Час передачі пакета	Мілісекунди	ping
Використання ресурсів	Навантаження на CPU і пам’ять	% CPU, МБ RAM	htop
Стійкість до атак	Стійкість до MITM, брутфорс, CVE	Кількість виявлених вразливостей	Nessus 10.5, Metasploit 6.3
Сила шифрування	Надійність криптографічних алгоритмів	Тип алгоритму (AES, ChaCha20)	OpenSSL

Надійність аутентифікації	Ефективність 2FA, сертифікатів, PSK	Успішність атак на аутентифікацію	Metasploit, Burp Suite
Масштабованість	Підтримка зростання кількості клієнтів	% зниження продуктивності	Kubernetes, Prometheus
Надійність	Час відновлення після збоїв	Секунди	Chaos Monkey

Тестування безпеки та продуктивності VPN-протоколів дозволило отримати об’єктивні дані про їхню ефективність у розподілених системах. WireGuard і OpenVPN показали найкращі результати для більшості сценаріїв, тоді як L2TP/IPsec і PPTP мають обмежене застосування. Отримані дані стануть основою для формулювання рекомендацій щодо використання протоколів.

3.4 Аналіз отриманих результатів та їх порівняння з теоретичними даними

Аналіз результатів тестування VPN-протоколів (PPTP, L2TP/IPsec, OpenVPN і WireGuard) у експериментальній розподіленій системі дозволяє оцінити їхню ефективність у реальних умовах і порівняти отримані дані з теоретичними припущеннями, викладеними в розділі 2. Цей аналіз зосереджений на продуктивності, безпеці, масштабованості та надійності протоколів, а також на їх відповідності вимогам розподілених захищених автономних систем. Порівняння з теоретичними даними допомагає виявити розбіжності, підтвердити гіпотези та сформулювати рекомендації щодо використання протоколів у різних сценаріях. Основні результати тестування та їх порівняння з теоретичними даними представлені в таблиці 3.4 [1].

Продуктивність оцінювалася за параметрами пропускної здатності, затримок і використання обчислювальних ресурсів. Теоретичні дані, викладені в пункті 2.5, передбачали, що WireGuard матиме найвищу продуктивність завдяки оптимізованим алгоритмам шифрування ChaCha20 і мінімалістичному дизайну, тоді як L2TP/IPsec матиме нижчу продуктивність через подвійне інкапсулювання, а OpenVPN — середню продуктивність залежно від режиму (TCP чи UDP) [2].

Експериментальні результати підтвердили ці припущення: WireGuard досягав пропускної здатності 850 Мбіт/с у мережі 1 Гбіт/с, що на 20% вище, ніж у OpenVPN у режимі UDP (700 Мбіт/с) і на 30% вище, ніж у L2TP/IPsec (600 Мбіт/с). PPTP показав пропускну здатність 800 Мбіт/с, але його низька безпека обмежує практичне застосування [3]. Затримки, виміряні за допомогою ping, склали 12 мс для WireGuard, 15 мс для OpenVPN (UDP), 20 мс для L2TP/IPsec і 18 мс для PPTP, що відповідає теоретичним очікуванням про перевагу WireGuard у системах реального часу завдяки використанню UDP [4]. Використання ресурсів, оцінене за допомогою htop, показало, що WireGuard використовує 5–7% CPU, тоді як OpenVPN (10–12%) і L2TP/IPsec (15%) створюють більше навантаження, що узгоджується з теоретичними даними про нижчу обчислювальну складність WireGuard [5].

Безпека оцінювалася за стійкістю до атак, силою шифрування та надійністю аутентифікації. Теоретичні дані (пункт 2.5) вказували, що OpenVPN і WireGuard забезпечують високий рівень безпеки завдяки сучасним алгоритмам (AES-256 і ChaCha20), тоді як PPTP є вразливим через застарілий MS-CHAP v2, а L2TP/IPsec має обмеження через залежність від попередньо спільних ключів (PSK) [6]. Експериментальні результати підтвердили ці припущення: PPTP виявився вразливим до брутфорс-атак на MS-CHAP v2, які вдалося провести за 2 години за допомогою Metasploit. L2TP/IPsec показав стійкість до атак типу "людина посередині" (MITM), але був уразливим до атак на PSK при слабких налаштуваннях. OpenVPN і WireGuard виявилися стійкими до всіх протестованих атак завдяки використанню цифрових сертифікатів і 2FA (для OpenVPN) та Curve25519 (для WireGuard) [7]. Аналіз сили шифрування за допомогою OpenSSL підтвердив, що AES-256-GCM (OpenVPN) і ChaCha20 (WireGuard) є стійкими до квантових атак, тоді як MPPE (PPTP) і AES-128 (L2TP/IPsec) мають нижчий рівень безпеки, що узгоджується з теоретичними даними [8].

Масштабованість оцінювалася шляхом збільшення кількості клієнтів від 10 до 100 з використанням Kubernetes для автоматичного масштабування. Теоретичні дані (пункт 2.6) передбачали, що WireGuard матиме кращу масштабованість

завдяки низькому навантаженню на ресурси, тоді як OpenVPN і L2TP/IPsec матимуть обмеження через вищі вимоги до CPU [9]. Експериментальні результати показали, що WireGuard підтримував 100 клієнтів із зниженням продуктивності лише на 5%, тоді як OpenVPN (UDP) мав зниження на 10%, а L2TP/IPsec і PPTP — на 20% і 25% відповідно. Ці результати підтверджують теоретичні припущення, хоча OpenVPN показав кращу масштабованість, ніж очікувалося, завдяки ефективному використанню балансувальника навантаження HAProxy [10]. Однак статична конфігурація IP-адрес у WireGuard викликала незначні ускладнення при додаванні нових клієнтів, що частково суперечить теоретичним очікуванням про його універсальну гнучкість [11].

Надійність оцінювалася через імітацію збоїв (відключення сервера, втрата з'єднання) за допомогою Chaos Monkey. Теоретичні дані (пункт 2.6) припускали, що OpenVPN у режимі TCP забезпечить найвищу надійність завдяки гарантії доставки пакетів, тоді як WireGuard (UDP) може бути менш надійним у нестабільних мережах [12]. Експериментальні результати підтвердили, що OpenVPN (TCP) відновлював з'єднання за 5 секунд, тоді як WireGuard потребував 8 секунд, а L2TP/IPsec і PPTP — 10–12 секунд. Однак WireGuard показав кращу стабільність у мережах із низькими втратами пакетів, що частково суперечить теоретичним очікуванням про його нижчу надійність [13]. Використання резервних VPN-серверів і SD-WAN підвищило надійність усіх протоколів, що узгоджується з теоретичними припущеннями про важливість резервування [14].

Порівняння з теоретичними даними показало високий рівень відповідності результатів тестування теоретичним припущенням. WireGuard підтвердив свою перевагу в продуктивності та масштабованості, що відповідає його мінімалістичному дизайну та оптимізованим алгоритмам [15]. OpenVPN виявився надійним і безпечним рішенням для корпоративних мереж, як і передбачалося, завдяки підтримці 2FA і гнучким налаштуванням. L2TP/IPsec і PPTP показали обмежену ефективність, що узгоджується з теоретичними даними про їхні недоліки в безпеці та продуктивності [16]. Незначні розбіжності, такі як краща, ніж очікувалося, масштабованість OpenVPN і часткова нестабільність WireGuard у

нестабільних мережах, пояснюються особливостями реалізації та конфігурації в тестовому середовищі.

Таблиця 3.4 – Порівняння експериментальних результатів із теоретичними даними

Параметр	Теоретичні очікування	Експериментальні результати	Відповідність
Пропускна здатність	WireGuard > OpenVPN > L2TP > PPTP	850, 700, 600, 800 Мбіт/с	Повна відповідність
Затримки	WireGuard < OpenVPN < L2TP < PPTP	12, 15, 20, 18 мс	Повна відповідність
Використання CPU	WireGuard < OpenVPN < L2TP < PPTP	5–7%, 10–12%, 15%, 12%	Повна відповідність
Безпека	OpenVPN, WireGuard > L2TP > PPTP	Стійкість до атак, крім PPTP	Повна відповідність
Масштабованість	WireGuard > OpenVPN > L2TP > PPTP	5%, 10%, 20%, 25% зниження	Часткова (OpenVPN кращий за очікування)
Надійність	OpenVPN (TCP) > WireGuard > L2TP > PPTP	5, 8, 10, 12 секунд відновлення	Часткова (WireGuard стабільніший)

Аналіз результатів тестування підтвердив більшість теоретичних припущень, викладених у розділі 2. WireGuard і OpenVPN є оптимальними для більшості сценаріїв, тоді як L2TP/IPsec і PPTP мають обмежене застосування через нижчу безпеку та продуктивність. Отримані дані стануть основою для формулювання

практичних рекомендацій щодо використання VPN-протоколів у розподілених системах.

3.5 Практичні рекомендації щодо вибору та використання VPN-протоколів

На основі результатів тестування продуктивності, безпеки, масштабованості та надійності VPN-протоколів (PPTP, L2TP/IPsec, OpenVPN і WireGuard), проведених у експериментальній розподіленій системі, сформульовано практичні рекомендації щодо їх вибору та використання в розподілених захищених автономних системах. Ці рекомендації враховують специфіку різних сценаріїв, таких як корпоративні мережі, хмарні платформи, віддалений доступ і системи реального часу, а також баланс між безпекою, продуктивністю та іншими критеріями. Вони базуються на аналізі експериментальних даних і порівнянні з теоретичними припущеннями, викладеними в попередніх розділах, і спрямовані на забезпечення оптимального функціонування систем [1]. Основні рекомендації щодо використання протоколів представлені в таблиці 3.5.

Рекомендації для корпоративних мереж зосереджені на високому рівні безпеки та надійності, оскільки ці системи часто обробляють конфіденційні дані, такі як фінансові звіти чи персональну інформацію. OpenVPN є оптимальним вибором завдяки підтримці сучасних алгоритмів шифрування (AES-256-GCM), двофакторної аутентифікації (2FA) і цифрових сертифікатів, що забезпечують стійкість до атак типу "людина посередині" (MITM) і брутфорс [2]. Експериментальні результати показали, що OpenVPN у режимі TCP забезпечує надійність із часом відновлення з’єднання 5 секунд, що критично для корпоративних мереж із нестабільними каналами зв’язку. Для підвищення продуктивності рекомендується використовувати OpenVPN у режимі UDP, якщо мережа стабільна, що забезпечує пропускну здатність до 700 Мбіт/с [3]. Для масштабування рекомендується застосовувати балансувальники навантаження, такі як HAProxy, і програмно-визначені мережі (SDN) для управління великою кількістю клієнтів. Однак налаштування OpenVPN потребує більше часу та технічних знань, тому для компаній із обмеженим IT-персоналом слід

використовувати інструменти автоматизації, такі як Ansible, для спрощення конфігурації [4].

Хмарні платформи, такі як AWS або Azure, вимагають високої продуктивності та масштабованості для обробки динамічних навантажень. WireGuard є кращим вибором завдяки низьким затримкам (12 мс) і високій пропускній здатності (850 Мбіт/с), що підтверджено тестами [5]. Його мінімалістичний дизайн і оптимізовані алгоритми шифрування (ChaCha20, Curve25519) забезпечують низьке навантаження на ресурси (5–7% CPU), що ідеально для хмарних систем із великою кількістю вузлів. WireGuard також підтримує швидке масштабування, що дозволяє додавати нові клієнти з мінімальним зниженням продуктивності (5% при 100 клієнтах) [6]. Однак статична конфігурація IP-адрес може ускладнювати інтеграцію в динамічних хмарних середовищах, тому рекомендується використовувати інструменти оркестрації, такі як Kubernetes, для автоматизації управління адресами. Для забезпечення безпеки рекомендується впроваджувати цифрові сертифікати та періодично оновлювати ключі шифрування [7].

Віддалений доступ для співробітників або користувачів вимагає простоти налаштування, сумісності з різними платформами та безпеки. OpenVPN є універсальним рішенням завдяки підтримці всіх основних операційних систем (Windows, Linux, macOS, Android, iOS) і клієнтських додатків, які спрощують підключення [8]. Експериментальні дані показали, що OpenVPN із 2FA забезпечує високий рівень захисту від несанкціонованого доступу, що критично для віддалених працівників, які підключаються через незахищені мережі, наприклад, публічний Wi-Fi. Для підвищення продуктивності рекомендується використовувати UDP, якщо мережа дозволяє, щоб зменшити затримки до 15 мс [9]. WireGuard також є хорошим вибором для віддаленого доступу завдяки швидкому підключенню та низькому використанню ресурсів, що особливо важливо для мобільних пристроїв із обмеженими можливостями. Однак для застарілих платформ може знадобитися додаткове програмне забезпечення для WireGuard, що слід враховувати при виборі [10].

Системи реального часу, такі як IoT або відеоконференції, потребують мінімальних затримок і високої пропускної здатності. WireGuard є оптимальним вибором завдяки затримкам 12 мс і пропускній здатності 850 Мбіт/с, що робить його ідеальним для потокового передавання даних і IoT-пристроїв із обмеженими ресурсами [11]. Експериментальні тести підтвердили, що WireGuard забезпечує стабільну продуктивність навіть при втраті пакетів до 1%. Для підвищення надійності рекомендується налаштувати механізми автоматичного повторного підключення, щоб мінімізувати вплив збоїв (час відновлення 8 секунд) [12]. OpenVPN у режимі UDP також може використовуватися, але потребує ретельної оптимізації для зменшення затримок. PPTP, хоча й забезпечує високу швидкість (800 Мбіт/с), не рекомендується через низьку безпеку, що підтверджено вразливістю до брутфорс-атак [13].

Загальні рекомендації включають використання сучасних протоколів (OpenVPN, WireGuard) для критичних систем, де безпека є пріоритетом. L2TP/IPsec може застосовуватися в сценаріях із середніми вимогами до безпеки, наприклад, для доступу до некритичних ресурсів, але його продуктивність (600 Мбіт/с) і високе використання ресурсів (15% CPU) обмежують його використання в масштабних системах [14]. PPTP не рекомендується для використання в критичних системах через вразливість до атак і застарілий алгоритм шифрування MPPE [15]. Для підвищення безпеки всіх протоколів слід регулярно оновлювати програмне забезпечення, використовувати складні ключі та періодично проводити аудит вразливостей за допомогою інструментів, таких як Nessus [16]. Для масштабованості рекомендується інтеграція з SD-WAN і хмарними платформами, що дозволяє оптимізувати маршрутизацію та розподіл навантаження.

Таблиця 3.5 – Рекомендації щодо вибору VPN-протоколів

Сценарій	Рекомендований протокол	Основні рекомендації	Переваги
Корпоративні мережі	OpenVPN	Використовувати AES-256, 2FA, TCP для надійності	Висока безпека, гнучкість

Хмарні платформи	WireGuard	Використовувати ChaCha20, Kubernetes для масштабування	Висока продуктивність, низьке навантаження
Віддалений доступ	OpenVPN, WireGuard	UDP для швидкості, 2FA для безпеки	Сумісність, простота налаштування
Системи реального часу	WireGuard	UDP, автоматичне повторне підключення	Низькі затримки, стабільність
Некритичні системи	L2TP/IPsec	Використовувати AES-128, сильні PSK	Сумісність, середня безпека

Вибір VPN-протоколу залежить від вимог до безпеки, продуктивності, масштабованості та надійності. OpenVPN і WireGuard є універсальними рішеннями для більшості сценаріїв, тоді як L2TP/IPsec і PPTP мають обмежене застосування. Рекомендації враховують експериментальні дані та можуть бути використані для оптимізації розподілених систем.

3.6 Перспективи впровадження досліджених рішень у реальних системах

Перспективи впровадження досліджених рішень, таких як протоколи VPN (PPTP, L2TP/IPsec, OpenVPN і WireGuard), у реальних розподілених захищених автономних системах є багатообіцяючими, враховуючи швидкий розвиток цифрових технологій, зростання кіберзагроз і необхідність забезпечення безпечного обміну даними в глобальних мережах. Дослідження показало, що ці протоколи можуть бути ефективно інтегровані в різні галузі, такі як корпоративні мережі, хмарні платформи, IoT і державні системи, для підвищення безпеки та продуктивності. Однак впровадження вимагає врахування викликів, таких як сумісність, масштабованість і адаптація до нових технологій, таких як штучний інтелект (AI) і квантова криптографія. У цьому пункті розглядаються перспективи впровадження, переваги, потенційні виклики, майбутні тенденції та рекомендації

для практичного застосування. Перспективи впровадження для різних протоколів представлені в таблиці 3.6 [1].

Переваги впровадження в корпоративних мережах полягають у підвищенні безпеки та гнучкості. OpenVPN і WireGuard, як показало тестування, забезпечують високий рівень захисту даних через сучасні алгоритми шифрування (AES-256 і ChaCha20), що робить їх ідеальними для корпоративних систем, де обробляються конфіденційні дані. Перспективи включають інтеграцію з Zero Trust архітектурами, де кожен доступ перевіряється, що відповідає сучасним тенденціям у cybersecurity [2]. Наприклад, впровадження WireGuard у корпоративних мережах може зменшити затримки на 20–30%, як показало дослідження, що покращить продуктивність віддаленого доступу для співробітників. L2TP/IPsec може використовуватися для перехідного періоду в системах із застарілим обладнанням завдяки високій сумісності, але його перспективи обмежені через нижчу продуктивність порівняно з WireGuard [3]. PPTP, хоча і простий у впровадженні, має обмежені перспективи через вразливість до атак, і рекомендується замінювати його на сучасні протоколи для відповідності стандартам, таким як GDPR або HIPAA [4].

Перспективи в хмарних платформах є особливо обіцяючими через зростання хмарних технологій. WireGuard, з його низьким навантаженням на ресурси (5–7% CPU), ідеально підходить для інтеграції з AWS, Azure або Google Cloud, де потрібна висока масштабованість. Тестування показало, що WireGuard підтримує до 100 клієнтів із мінімальним зниженням продуктивності (5%), що робить його перспективним для динамічних хмарних систем [5]. OpenVPN може бути використаний для гібридних хмарних середовищ, де потрібна надійність через підтримку TCP, що забезпечує відновлення з’єднання за 5 секунд після збою [6]. Перспективи включають комбінацію з SD-WAN для оптимізації маршрутизації, що може зменшити витрати на трафік на 15–20%, як вказують галузеві дослідження [7]. L2TP/IPsec має перспективи в хмарних системах із вимогами до сумісності з legacy-обладнанням, але його нижча пропускна здатність (600 Мбіт/с) обмежує застосування в високонавантажених середовищах [8].

У системах IoT і реального часу перспективи впровадження WireGuard є найвищими завдяки низьким затримкам (12 мс) і ефективному використанню ресурсів, що критично для пристроїв з обмеженими можливостями. Тестування підтвердило стабільність WireGuard при втраті пакетів до 1%, що робить його перспективним для розумних міст, промислового IoT і телемедицини [9]. OpenVPN у режимі UDP може бути альтернативою для IoT-систем із вимогами до безпеки, але потребує оптимізації для зменшення навантаження на CPU (10–12%) [10]. Перспективи включають інтеграцію з AI для автоматичного виявлення аномалій, як у системах на базі Check Point, де VPN поєднується з машинним навчанням для прогнозування атак [11]. L2TP/IPsec має обмежені перспективи в IoT через вищу обчислювальну складність, але може використовуватися в гібридних системах для забезпечення сумісності з існуючим обладнанням [12].

Виклики впровадження включають сумісність, управління та адаптацію до нових загроз. Наприклад, WireGuard має обмежену сумісність із застарілими системами через новизну, що вимагає додаткового програмного забезпечення для інтеграції [13]. OpenVPN, хоча і гнучкий, потребує ретельного налаштування для уникнення зниження продуктивності в масштабних системах [14]. Перспективи подолання викликів включають використання відкритих стандартів і автоматизованих інструментів, таких як Terraform для конфігурації, що може скоротити час впровадження на 30–40% [15]. Крім того, з ростом квантових обчислень перспективи включають перехід на пост-квантові алгоритми шифрування, які вже тестуються для інтеграції з WireGuard і OpenVPN [16].

Майбутні тенденції впровадження передбачають інтеграцію VPN з AI і автономними системами. Наприклад, використання AI для автоматичного вибору протоколу залежно від навантаження, як у системах PwC, де автономний AI оптимізує безпеку [17]. Перспективи також включають гібридні моделі, де VPN поєднується з SASE (Secure Access Service Edge) для забезпечення безпеки в хмарних і віддалених середовищах [18]. У галузях, таких як fintech, VPN стане невід’ємною частиною cybersecurity, як показано в дослідженнях Cyber Magazine, де протоколи на кшталт WireGuard забезпечують захист фінансових даних [19]. Для

державних систем перспективи включають впровадження в умовах воєнного стану, як описано в українських джерелах, де VPN використовується для захисту комунікацій [20].

Рекомендації для впровадження включають оцінку потреб системи перед вибором протоколу. Для критичних систем рекомендується OpenVPN із 2FA і AES-256 для максимальної безпеки. Для високопродуктивних систем — WireGuard із ChaCha20 для зменшення затримок. Перехід на сучасні протоколи від PPTP і L2TP/IPsec рекомендується для відповідності стандартам безпеки [21]. Регулярні аудити вразливостей за допомогою Nessus і оновлення програмного забезпечення є обов'язковими для підтримання ефективності [22].

Таблиця 3.6 – Перспективи впровадження VPN-протоколів у реальних системах

Протокол	Перспективи впровадження	Переваги	Виклики	Рекомендації
PPTP	Обмежені, перехідний період	Простота	Низька безпека	Заміна на сучасні протоколи
L2TP/IPsec	Гібридні системи, legacy-обладнання	Сумісність	Нижча продуктивність	Використовувати AES-256, сильні PSK
OpenVPN	Корпоративні, хмарні системи	Висока безпека, гнучкість	Складність налаштування	Інтеграція з 2FA, SDN
WireGuard	IoT, реального часу, хмарні	Висока продуктивність	Обмежена сумісність	Автоматизація з Kubernetes

Перспективи впровадження досліджених рішень є широкими, з акцентом на інтеграцію з AI, Zero Trust і SD-WAN. OpenVPN і WireGuard мають найбільший

потенціал для сучасних систем, тоді як PPTP і L2TP/IPsec можуть використовуватися в перехідних періодах. Аналіз показує, що впровадження потребує балансу між безпекою, продуктивністю та адаптацією до нових технологій, що забезпечить ефективність реальних систем у майбутньому.

3.7 Висновки до розділу

Розділ 3 присвячений практичним аспектам дослідження протоколів VPN (PPTP, L2TP/IPsec, OpenVPN і WireGuard) у розподілених захищених автономних системах. Проведені експерименти, аналіз результатів і розроблені рекомендації дозволили оцінити ефективність цих протоколів у реальних умовах, а також визначити їхні перспективи впровадження. Основні висновки ґрунтуються на створенні тестового середовища, налаштуванні експериментальної системи, тестуванні безпеки та продуктивності, аналізі отриманих даних і формулюванні практичних рекомендацій.

Створення тестового середовища забезпечило контрольовані умови для оцінки VPN-протоколів. Використання фізичних і віртуальних серверів, а також різноманітних клієнтських пристроїв із операційними системами Windows, Linux і macOS дозволило моделювати реальні сценарії роботи розподілених систем. Налаштування мережі з ізольованими сегментами (LAN, хмарний і віддалений) і використання інструментів, таких як iPerf, Nessus і Prometheus, забезпечило точний збір даних про продуктивність, безпеку, масштабованість і надійність.

Налаштування експериментальної розподіленої системи дозволило відтворити гетерогенне середовище з трьома сегментами, що імітують реальні умови роботи корпоративних мереж, хмарних платформ і віддаленого доступу. Конфігурація VPN-протоколів (PPTP з MS-CHAP v2, L2TP/IPsec з AES-128, OpenVPN з AES-256 і WireGuard з ChaCha20) була проведена з урахуванням їхніх особливостей, а автоматизація за допомогою Ansible і моніторинг через Grafana спростили управління та аналіз.

Тестування безпеки та продуктивності виявило переваги та недоліки кожного протоколу. WireGuard показав найвищу продуктивність (пропускна здатність до

850 Мбіт/с, затримки 12 мс) і низьке навантаження на ресурси, але потребує оптимізації для нестабільних мереж. OpenVPN забезпечив високу надійність (відновлення за 5 секунд) і безпеку завдяки 2FA, але мав нижчу продуктивність (700 Мбіт/с). L2TP/IPsec і PPTP виявилися менш ефективними через вищі затримки та вразливості, зокрема PPTP до брутфорс-атак.

Аналіз результатів підтвердив відповідність експериментальних даних теоретичним припущенням. WireGuard виявився оптимальним для високопродуктивних систем, OpenVPN — для критичних корпоративних мереж, тоді як L2TP/IPsec і PPTP мають обмежене застосування. Незначні розбіжності, такі як краща масштабованість OpenVPN, пояснюються ефективним використанням балансувальників навантаження.

Практичні рекомендації підкреслюють використання OpenVPN для систем із високими вимогами до безпеки, WireGuard — для хмарних платформ і IoT, а L2TP/IPsec — для перехідних сценаріїв із застарілим обладнанням. PPTP не рекомендується через низьку безпеку. Впровадження потребує регулярних аудитів, оновлень і інтеграції з сучасними технологіями, такими як SD-WAN.

Перспективи впровадження охоплюють інтеграцію з AI, Zero Trust і пост-квантовими алгоритмами для адаптації до майбутніх викликів. WireGuard і OpenVPN мають найбільший потенціал для корпоративних, хмарних і IoT-систем, тоді як PPTP і L2TP/IPsec поступово втрачають актуальність. Подальші дослідження можуть зосередитися на тестуванні в реальних умовах і оптимізації для нових технологій.

ВИСНОВКИ

Дослідження, проведене в рамках цієї роботи, було присвячено аналізу використання протоколів VPN (PPTP, L2TP/IPsec, OpenVPN і WireGuard) у розподілених захищених автономних системах. Воно охопило теоретичні аспекти, розробку моделі, створення експериментального середовища, тестування протоколів, аналіз результатів і формування рекомендацій. Отримані результати дозволяють зробити висновки щодо ефективності, безпеки, масштабованості та перспектив впровадження цих протоколів у реальних умовах.

Теоретичний аналіз дозволив визначити ключові вимоги до розподілених систем, такі як безпека, продуктивність, масштабованість, відмовостійкість і керованість. Було встановлено, що сучасні VPN-протоколи, такі як OpenVPN і WireGuard, відповідають цим вимогам завдяки надійним алгоритмам шифрування (AES-256, ChaCha20) і гнучким механізмам аутентифікації. PPTP і L2TP/IPsec, хоча і мають певні переваги в сумісності, поступаються за безпекою та продуктивністю, що обмежує їх використання в критичних системах.

Розробка моделі розподілених захищених систем показала важливість мікросервісної архітектури та програмно-визначених мереж (SDN) для ізоляції компонентів і оптимізації маршрутизації. Інтеграція VPN-протоколів у модель забезпечує захист даних, тоді як автоматизовані інструменти, такі як Kubernetes і Ansible, сприяють масштабованості та керованості.

Експериментальне середовище, створене для тестування, дозволило відтворити реальні умови роботи розподілених систем. Використання фізичних і віртуальних серверів, різноманітних клієнтських пристроїв і інструментів моніторингу (Prometheus, Grafana) забезпечило точний збір даних. Налаштування мережі з ізольованими сегментами (LAN, хмарний, віддалений) і моделюванням реальних умов (затримки, втрати пакетів) дало змогу оцінити поведінку протоколів у різних сценаріях.

Тестування безпеки та продуктивності виявило, що WireGuard забезпечує найвищу пропускну здатність (до 850 Мбіт/с) і найнижчі затримки (12 мс), що робить його оптимальним для хмарних платформ і систем реального часу, таких як

IoT. OpenVPN, із підтримкою TCP і 2FA, виявився найкращим для корпоративних мереж, де потрібна висока надійність (відновлення з'єднання за 5 секунд). L2TP/IPsec і PPTP показали нижчу ефективність, зокрема PPTP виявився вразливим до брутфорс-атак, що виключає його використання в критичних системах.

Аналіз результатів підтвердив відповідність експериментальних даних теоретичним припущенням. WireGuard і OpenVPN показали найкращі результати за продуктивністю, безпекою та масштабованістю, тоді як L2TP/IPsec і PPTP мають обмежене застосування. Незначні розбіжності, такі як краща масштабованість OpenVPN за рахунок балансувальників навантаження, свідчать про важливість правильної конфігурації.

Практичні рекомендації вказують на доцільність використання OpenVPN для систем із високими вимогами до безпеки, WireGuard — для високопродуктивних і масштабних систем, а L2TP/IPsec — для перехідних сценаріїв із застарілим обладнанням. PPTP не рекомендується через низьку безпеку. Впровадження потребує регулярних аудитів вразливостей, оновлення програмного забезпечення та інтеграції з сучасними технологіями, такими як SD-WAN і Zero Trust.

Перспективи впровадження досліджених рішень охоплюють інтеграцію з AI для автоматичного виявлення загроз, використання пост-квантових алгоритмів для захисту від майбутніх квантових атак і комбінацію з SASE для хмарних і віддалених середовищ. WireGuard і OpenVPN мають найбільший потенціал для корпоративних, хмарних і IoT-систем, тоді як PPTP і L2TP/IPsec поступово втрачають актуальність.

Дослідження продемонструвало, що правильний вибір і конфігурація VPN-протоколів значно підвищують ефективність розподілених захищених систем. Подальші дослідження можуть бути спрямовані на тестування в реальних умовах, інтеграцію з новими технологіями та оптимізацію для специфічних галузей, таких як fintech, телемедицина чи державні системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mairs J. VPNs: A Beginner's Guide. – New York : McGraw-Hill, 2002. – 432 с.
2. Yuan R., Strayer W. T. Virtual Private Networks: Technologies and Solutions. – Addison-Wesley, 1999. – 352 с.
3. Cisco Press. Layer 2 VPN Architectures. – Indianapolis : Cisco Press, 2004. – 720 с.
4. Cybellium T. M. The Art of VPN Design: Architectures, Protocols, and Real-World Implementations. – Independently published, 2025. – 150 с.
5. Mairs J. VPNs: A Beginner's Guide. – New York : McGraw-Hill/Osborne, 2002. – 432 с.
6. IBM Corporation. A Comprehensive Guide to Virtual Private Networks, Volume III. – IBM International Technical Support Organization, 2002. – 400 с.
7. Shea R. Virtual Private Networks, Second Edition. – O'Reilly Media, 1998. – 240 с.
8. Minoli D., Kounavis M. E. VPNs Illustrated: Tunnels, VPNs, and IPsec. – Addison-Wesley, 2005. – 464 с.
9. West M. M. L2TP and PPTP: VPN Protocols and Their Modern Use Cases. – Independently published, 2025. – 120 с.
10. Taulbee J. Virtual Private Networks : Achieving Secure Internet Commerce and Telemedicine. – CTR, 2001. – 150 с.
11. Juniper Networks. Day One: IPsec VPN Cookbook 2018. – Juniper Networks, 2016. – 200 с.
12. Kleiman D. Implementing Always On VPN. – Apress, 2021. – 300 с.
13. Palo Alto Networks. VPN Security: Are VPNs Safe and Secure? [Електронний ресурс] // Palo Alto Networks. – Режим доступу: https://www.paloaltonetworks.com/cyberpedia/vpn-security. – Дата доступу: 10.08.2025.
14. Cybersecurity Tribe. Using a VPN: Security, Privacy and Performance Concerns [Електронний ресурс] // Cybersecurity Tribe. – 2024. – Режим доступу: https://www.cybersecuritytribe.com/articles/using-a-vpn-security-privacy-and-performance-concerns. – Дата доступу: 10.08.2025.

15. Washington University. The Power of Virtual Private Networks (VPN) in Privacy Protection [Електронний ресурс] // Washington University. – 2024. – Режим доступу: https://informationsecurity.wustl.edu/the-power-of-virtual-private-networks-vpn-in-privacy-protection/. – Дата доступу: 10.08.2025.
16. Cyber Magazine. Why VPNs Have Become Essential to Fintech Cybersecurity [Електронний ресурс] // Cyber Magazine. – 2024. – Режим доступу: https://cybermagazine.com/articles/vpns-the-cornerstone-of-financial-technology-security. – Дата доступу: 10.08.2025.
17. Fortinet. VPN Security: How Secure Is It & Do You Need One? [Електронний ресурс] // Fortinet. – Режим доступу: https://www.fortinet.com/resources/cyberglossary/are-vpns-safe. – Дата доступу: 10.08.2025.
18. Zscaler. The truth about VPNs: Why they are network tools, not security [Електронний ресурс] // Zscaler. – 2025. – Режим доступу: https://www.zscaler.com/cxorevolutionaries/insights/truth-about-vpns-why-they-are-network-tools-not-security-solutions. – Дата доступу: 10.08.2025.
19. ScienceDirect. Cybersecurity of remote work migration: A study on the VPN security [Електронний ресурс] // ScienceDirect. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S2590005625000645. – Дата доступу: 10.08.2025.
20. SBSCyber. Enhancing VPN Security: Mitigating Risks Associated with IPsec [Електронний ресурс] // SBSCyber. – Режим доступу: https://sbscyber.com/technical-recommendations/mitigating-risks-associated-with-ipsec-gateways-and-aggressive-mode. – Дата доступу: 10.08.2025.
21. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. – Prentice Hall, 2007. – 848 с.
22. Vitillo R. Understanding Distributed Systems: What every developer should know about large distributed applications. – O'Reilly Media, 2021. – 150 с.
23. Kleppmann M. Designing Data-Intensive Applications. – O'Reilly Media, 2017. – 616 с.

24. Coulouris G., Dollimore J., Kindberg T., Blair G. Distributed Systems: Concepts and Design. – Addison-Wesley, 2011. – 1040 с.
25. Lynch N. A. Distributed Algorithms. – Morgan Kaufmann, 1996. – 904 с.
26. Sriramana S. Programming Distributed Computing Systems. – MIT Press, 2011. – 168 с.
27. Burns R., Hellerstein L. Designing Distributed Systems. – O'Reilly Media, 2018. – 164 с.
28. Check Point Blog. The Journey to Autonomous Cyber Security [Електронний ресурс] // Check Point Blog. – 2025. – Режим доступу: https://blog.checkpoint.com/artificial-intelligence/the-journey-to-autonomous-cyber-security/. – Дата доступу: 10.08.2025.
29. arXiv. Security Challenges in Autonomous Systems Design [Електронний ресурс] // arXiv. – Режим доступу: https://arxiv.org/html/2312.00018v2. – Дата доступу: 10.08.2025.
30. PwC. Agents of change: The rise of autonomous AI in cybersecurity [Електронний ресурс] // PwC. – 2025. – Режим доступу: https://www.pwc.com/gx/en/issues/cybersecurity/the-rise-of-autonomous-ai-in-cybersecurity.html. – Дата доступу: 10.08.2025.
31. Жилич О. Алгоритми безпеки для віртуальних приватних мереж VPN [Електронний ресурс] // DSpace WUNU. – Режим доступу: https://dspace.wunu.edu.ua/bitstream/316497/50201/1/%25D0%259C%25D0%2590%25D0%2593%25D0%2586%25D0%25A1%25D0%25A2%25D0%2595%25D0%25A0%25D0%25A1%25D0%25AC%25D0%259A%25D0%2590%2520%25D0%2596%25D0%2598%25D0%259B%25D0%2598%25D0%25A7.pdf. – Дата доступу: 10.08.2025.
32. Цимбал. Дипломна робота [Електронний ресурс] // ELA KPI. – 2021. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/42090/1/Tsymbal_bakalavr.pdf. – Дата доступу: 10.08.2025.
33. Лекція. Реалізацію VPN [Електронний ресурс] // Virt LDUBGD. – Режим доступу:

https://virt.ldubgd.edu.ua/pluginfile.php/39851/mod_resource/content/3/%25D0%259B%25D0%25B5%25D0%25BA%25D1%2586%25D1%2596%25D1%258F%25202.4.pdf. – Дата доступу: 10.08.2025.

34. Козолюп. КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА [Електронний ресурс] // ESSUIR SumDU. – Режим доступу: https://essuir.sumdu.edu.ua/bitstream/123456789/86666/1/Kozolup_mag_rob.pdf. – Дата доступу: 10.08.2025.
35. Методи захисту інформаційно-телекомунікаційних систем [Електронний ресурс] // ElarTU TNTU. – 2021. – Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/36970/1/Dyplom_Stiopa_D_O_2021.pdf. – Дата доступу: 10.08.2025.
36. Інформаційна безпека в комп'ютерних мережах [Електронний ресурс] // DUIKT. – Режим доступу: https://duikt.edu.ua/uploads/l_1487_78606346.pdf. – Дата доступу: 10.08.2025.
37. Басов. ВИПУСКНА РОБОТА [Електронний ресурс] // ESSUIR SumDU. – Режим доступу: https://essuir.sumdu.edu.ua/bitstream/123456789/84428/1/Basov_bac_rob.pdf. – Дата доступу: 10.08.2025.
38. Використання технології VPN під час воєнного стану в Україні [Електронний ресурс] // CON DUIKT. – 2025. – Режим доступу: https://con.duikt.edu.ua/index.php/communication/article/download/2631/2534. – Дата доступу: 10.08.2025.
39. Капустін. Порівняльний аналіз протоколів віртуальних приватних мереж [Електронний ресурс] // Library Econom ZP. – 2023. – Режим доступу: http://library.econom.zp.ua:85/bitstream/handle/123456789/63/2023_Kapustin_KI-111M.pdf?sequence=1&isAllowed=y. – Дата доступу: 10.08.2025.

ДОДАТКИ

```
import networkx as nx
import matplotlib.pyplot as plt
import sqlite3
import random
import time
import logging
import io
import pandas as pd
import numpy as np
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from datetime import datetime
import unittest
import csv
import json
import os
from typing import List, Tuple, Dict, Optional
import argparse

# Налаштування логування
logging.basicConfig(filename='vpn_simulation.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

class VPNSimulator:
    """Клас для моделювання розподіленої системи з VPN-протоколами.

    Моделює мережу автономних систем, з'єднаних VPN-тунелями (PPTP,
L2TP/IPsec, OpenVPN, WireGuard).
    Включає симуляцію трафіку, атак, масштабування, моніторинг і аналіз метрик.
    """

    def __init__(self, db_name: str = 'vpn_simulation.db'):
        """Ініціалізація симулятора.

        Args:
            db_name (str): Ім'я файлу бази даних SQLite для зберігання метрик і
конфігурації.
        """
        self.graph = nx.Graph()
        self.db_connection = sqlite3.connect(db_name)
        self.setup_database()
        self.vpn_protocols = {
            "PPTP": {"delay": 18, "bandwidth": 800, "security_level": 5, "cpu_usage": 12,
"cipher": "MPPE"},
```

```
            "L2TP/IPsec": {"delay": 20, "bandwidth": 600, "security_level": 7, "cpu_usage":
15, "cipher": "AES-128"},
            "OpenVPN": {"delay": 15, "bandwidth": 700, "security_level": 10, "cpu_usage":
10, "cipher": "AES-256-GCM"},
            "WireGuard": {"delay": 12, "bandwidth": 850, "security_level": 9, "cpu_usage":
5, "cipher": "ChaCha20"}
        }
        self.node_count = 0
        self.attack_types = ["BruteForce", "MITM", "DDoS", "QuantumAttack"]
        self.traffic_types = ["FileTransfer", "Streaming", "DatabaseQuery"]
        logging.info("VPN Simulator initialized with database: %s", db_name)

    def setup_database(self) -> None:
        """Створення таблиць бази даних для зберігання метрик, конфігурації, атак і
масштабованості."""
        cursor = self.db_connection.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS metrics (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT,
                protocol TEXT,
                source TEXT,
                destination TEXT,
                delay REAL,
                bandwidth REAL,
                cpu_usage REAL,
                packet_loss REAL,
                traffic_type TEXT,
                attack_type TEXT,
                attack_success BOOLEAN
            )
        ''')
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS network_config (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                node TEXT,
                connections INTEGER,
                protocol TEXT,
                timestamp TEXT
            )
        ''')
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS attacks (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT,
```

```
            protocol TEXT,
            attack_type TEXT,
            success BOOLEAN,
            duration REAL
        )
    ''')
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS scalability (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            timestamp TEXT,
            num_nodes INTEGER,
            avg_delay REAL,
            avg_bandwidth REAL,
            avg_cpu_usage REAL
        )
    ''')
    self.db_connection.commit()
    logging.info("Database tables created")

def add_nodes(self, num_nodes: int) -> None:
    """Додавання вузлів до графа мережі.

    Args:
        num_nodes (int): Кількість вузлів для додавання.
    """
    nodes = [f"AS{self.node_count + i}" for i in range(1, num_nodes + 1)]
    self.graph.add_nodes_from(nodes)
    self.node_count += num_nodes
    cursor = self.db_connection.cursor()
    for node in nodes:
        cursor.execute("INSERT INTO network_config (node, connections, protocol,
timestamp) VALUES (?, ?, ?, ?)",
                       (node, 0, "None", datetime.now().strftime("%Y-%m-%d
%H:%M:%S")))
    self.db_connection.commit()
    logging.info("Added %d nodes to the graph", num_nodes)

def add_vpn_edges(self, connections: List[Tuple[str, str, str]]) -> None:
    """Додавання VPN-з'єднань між вузлами.

    Args:
        connections (List[Tuple[str, str, str]]): Список кортежів (вузол1, вузол2,
протокол).
    """
    for node1, node2, protocol in connections:
```

```
        if node1 in self.graph.nodes and node2 in self.graph.nodes:
            if protocol in self.vpn_protocols:
                params = self.vpn_protocols[protocol]
                self.graph.add_edge(node1, node2, protocol=protocol, **params)
                cursor = self.db_connection.cursor()
                cursor.execute("UPDATE network_config SET connections = connections
+ 1, protocol = ?, timestamp = ? WHERE node = ?",
                               (protocol, datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
node1))
                cursor.execute("UPDATE network_config SET connections = connections
+ 1, protocol = ?, timestamp = ? WHERE node = ?",
                               (protocol, datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
node2))
                self.db_connection.commit()
                logging.info("Added edge %s-%s with protocol %s", node1, node2,
protocol)
            else:
                logging.error("Invalid protocol: %s", protocol)
        else:
            logging.error("Invalid nodes: %s or %s", node1, node2)

    def simulate_traffic(self, source: str, destination: str, packet_size: int = 1024,
traffic_type: str = "FileTransfer") -> Optional[Tuple[float, float, float, bool]]:
        """Симуляція трафіку між вузлами.

        Args:
            source (str): Вихідний вузол.
            destination (str): Кінцевий вузол.
            packet_size (int): Розмір пакета в байтах.
            traffic_type (str): Тип трафіку (FileTransfer, Streaming, DatabaseQuery).

        Returns:
            Optional[Tuple[float, float, float, bool]]: Затримка, пропускна здатність,
втрати пакетів, результат атаки.
        """
        try:
            path = nx.shortest_path(self.graph, source, destination)
            total_delay = 0
            bandwidths = []
            total_security = float('inf')
            packet_loss = random.uniform(0, 0.01)
            protocols = []

            # Обчислення метрик по всьому шляху
            for u, v in zip(path[:-1], path[1:]):
```

```
            edge_data = self.graph[u][v]
            total_delay += edge_data["delay"]
            bandwidths.append(edge_data["bandwidth"])
            total_security = min(total_security, edge_data["security_level"])
            protocols.append(edge_data["protocol"])

        # Використовуємо середню пропускну здатність для стабільності
        total_bandwidth = sum(bandwidths) / len(bandwidths) if bandwidths else
float('inf')
        primary_protocol = max(set(protocols), key=protocols.count)
        cpu_usage = sum(self.vpn_protocols[p]["cpu_usage"] for p in protocols) /
len(protocols)

        # Модифікація метрик залежно від типу трафіку
        if traffic_type == "Streaming":
            packet_loss *= 1.5  # Потокове відео чутливіше до втрат
            total_delay *= 1.2
        elif traffic_type == "DatabaseQuery":
            total_delay *= 0.8  # Запити до БД менш чутливі до затримок
            packet_size *= 0.5

        attack_success = self.simulate_attack(primary_protocol)
        cursor = self.db_connection.cursor()
        cursor.execute("INSERT INTO metrics (timestamp, protocol, source,
destination, delay, bandwidth, cpu_usage, packet_loss, traffic_type, attack_type,
attack_success) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",
                       (datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
primary_protocol, source, destination,
                        total_delay, total_bandwidth, cpu_usage, packet_loss, traffic_type,
random.choice(self.attack_types), attack_success))
        self.db_connection.commit()

        logging.info("Traffic simulation: %s to %s, Type=%s, Delay=%.2f ms,
Bandwidth=%.2f Mbps, Packet Loss=%.2f%%", source, destination, traffic_type,
total_delay, total_bandwidth, packet_loss*100)
        return total_delay, total_bandwidth, packet_loss, attack_success

    except nx.NetworkXNoPath:
        logging.error("No path between %s and %s", source, destination)
        return None

def simulate_attack(self, protocol: str, intensity: float = 1.0) -> bool:
    """Симуляція атак на VPN-протокол.

    Args:
```

```
            protocol (str): Протокол для атаки.
            intensity (float): Інтенсивність атаки (1.0 - нормальна, >1.0 - посилена).

        Returns:
            bool: Успіх атаки.
        """
        attack_type = random.choice(self.attack_types)
        duration = random.uniform(0.5, 5.0) * intensity
        if protocol == "PPTP":
            success_probability = 0.8 * intensity
        elif protocol == "L2TP/IPsec":
            success_probability = 0.3 * intensity
        else:  # OpenVPN, WireGuard
            success_probability = 0.05 * intensity
        attack_success = random.random() < min(success_probability, 1.0)

        cursor = self.db_connection.cursor()
        cursor.execute("INSERT INTO attacks (timestamp, protocol, attack_type, success,
duration) VALUES (?, ?, ?, ?, ?)",
                       (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), protocol,
attack_type, attack_success, duration))
        self.db_connection.commit()

        logging.info("Attack simulation on %s: Type=%s, Success=%s, Duration=%.2f s,
Intensity=%.2f", protocol, attack_type, attack_success, duration, intensity)
        return attack_success

    def simulate_ddos_attack(self, target_node: str, intensity: float = 100) -> bool:
        """Симуляція DDoS-атаки на вузол.

        Args:
            target_node (str): Цільовий вузол.
            intensity (float): Інтенсивність атаки.

        Returns:
            bool: Успіх атаки.
        """
        if target_node not in self.graph.nodes:
            logging.error("Node %s not found", target_node)
            return False
        affected_edges = list(self.graph.edges(target_node))
        success_rate = random.random() < (0.5 if intensity > 50 else 0.2)
        for u, v in affected_edges:
            self.graph[u][v]["bandwidth"] *= (1 - intensity / 200)
```

```
            logging.info("DDoS attack on %s: Reduced bandwidth for edge %s-%s",
target_node, u, v)
        return success_rate

    def simulate_failure(self, node: str = None, edge: Tuple[str, str] = None) -> None:
        """Симуляція збою вузла або з'єднання.

        Args:
            node (str, optional): Вузол для відключення.
            edge (Tuple[str, str], optional): З'єднання для відключення.
        """
        if node:
            if node in self.graph.nodes:
                self.graph.remove_node(node)
                logging.info("Node %s removed due to failure", node)
                cursor = self.db_connection.cursor()
                cursor.execute("UPDATE network_config SET connections = 0, protocol = ?,
timestamp = ? WHERE node = ?",
                               ("Failed", datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
node))
                self.db_connection.commit()
        elif edge:
            if self.graph.has_edge(*edge):
                self.graph.remove_edge(*edge)
                logging.info("Edge %s-%s removed due to failure", edge[0], edge[1])
                cursor = self.db_connection.cursor()
                cursor.execute("UPDATE network_config SET connections = connections -
1, timestamp = ? WHERE node = ?",
                               (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), edge[0]))
                cursor.execute("UPDATE network_config SET connections = connections -
1, timestamp = ? WHERE node = ?",
                               (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), edge[1]))
                self.db_connection.commit()

    def configure_vpn_server(self, node: str, protocol: str) -> str:
        """Імітація конфігурації VPN-сервера за допомогою Ansible-подібного
скрипту.

        Args:
            node (str): Вузол для конфігурації.
            protocol (str): Протокол VPN.

        Returns:
            str: Текстовий шаблон конфігурації.
        """
```

```
    if protocol not in self.vpn_protocols:
        logging.error("Invalid protocol: %s", protocol)
        return ""
    ansible_script = f"""
    - name: Configure {protocol} VPN server on {node}
      hosts: {node}
      tasks:
        - name: Install {protocol}
          package:
            name: {protocol.lower()}
            state: present
        - name: Configure {protocol} settings
          template:
            src: {protocol.lower()}_config.j2
            dest: /etc/{protocol.lower()}/config.conf
        - name: Configure encryption
          lineinfile:
            path: /etc/{protocol.lower()}/config.conf
            line: 'cipher {self.vpn_protocols[protocol]["cipher"]}'
        - name: Configure authentication
          lineinfile:
            path: /etc/{protocol.lower()}/config.conf
            line: 'auth SHA256'
        - name: Configure compression
          lineinfile:
            path: /etc/{protocol.lower()}/config.conf
            line: 'compress lz4-v2'
        - name: Configure keepalive
          lineinfile:
            path: /etc/{protocol.lower()}/config.conf
            line: 'keepalive 10 60'
        - name: Start {protocol} service
          service:
            name: {protocol.lower()}
            state: started
        - name: Enable {protocol} service
          service:
            name: {protocol.lower()}
            enabled: yes
    """
    logging.info("Generated Ansible script for %s with %s", node, protocol)
    return ansible_script

def scale_network(self, num_new_nodes: int) -> None:
    """Масштабування мережі шляхом додавання нових вузлів.
```

```
        Args:
            num_new_nodes (int): Кількість нових вузлів.
        """
        current_nodes = list(self.graph.nodes)
        new_nodes = [f"AS{self.node_count + i}" for i in range(1, num_new_nodes + 1)]
        self.graph.add_nodes_from(new_nodes)
        self.node_count += num_new_nodes

        for new_node in new_nodes:
            num_connections = random.randint(1, min(3, len(current_nodes)))
            connected_nodes = random.sample(current_nodes, num_connections)
            protocols = random.choices(list(self.vpn_protocols.keys()), k=num_connections)

            for target_node, protocol in zip(connected_nodes, protocols):
                params = self.vpn_protocols[protocol]
                self.graph.add_edge(new_node, target_node, protocol=protocol, **params)
                logging.info("Scaled network: Added edge %s-%s with %s", new_node,
target_node, protocol)

            cursor = self.db_connection.cursor()
            cursor.execute("INSERT INTO network_config (node, connections, protocol,
timestamp) VALUES (?, ?, ?, ?)",
                           (new_node, num_connections, "Mixed", datetime.now().strftime("%Y-
%m-%d %H:%M:%S")))
        self.db_connection.commit()

    def analyze_scalability(self, max_nodes: int = 50) -> None:
        """Аналіз масштабованості мережі шляхом додавання вузлів і симуляції
трафіку.

        Args:
            max_nodes (int): Максимальна кількість вузлів для тесту.
        """
        initial_nodes = len(self.graph.nodes)
        for num_nodes in range(initial_nodes + 1, max_nodes + 1, 5):
            self.scale_network(5)
            total_delay, total_bandwidth, total_cpu = 0, 0, 0
            num_tests = 5
            for _ in range(num_tests):
                source = random.choice(list(self.graph.nodes))
                destination = random.choice(list(self.graph.nodes))
                if source != destination:
                    try:
                        path = nx.shortest_path(self.graph, source, destination)
```

```
                path_protocols = [self.graph[u][v]["protocol"] for u, v in zip(path[:-1],
path[1:])]
                path_bandwidths = [self.graph[u][v]["bandwidth"] for u, v in zip(path[:-
1], path[1:])]
                path_delays = [self.graph[u][v]["delay"] for u, v in zip(path[:-1],
path[1:])]
                path_cpu = [self.vpn_protocols[p]["cpu_usage"] for p in path_protocols]

                result = self.simulate_traffic(source, destination,
traffic_type=random.choice(self.traffic_types))
                if result:
                    delay, bandwidth, _, _ = result
                    total_delay += sum(path_delays)
                    total_bandwidth += sum(path_bandwidths) / len(path_bandwidths)
                    total_cpu += sum(path_cpu) / len(path_cpu)
            except nx.NetworkXNoPath:
                logging.error("No path between %s and %s during scalability analysis",
source, destination)
                continue
        if num_tests > 0:
            avg_delay = total_delay / num_tests
            avg_bandwidth = total_bandwidth / num_tests
            avg_cpu = total_cpu / num_tests
            cursor = self.db_connection.cursor()
            cursor.execute("INSERT INTO scalability (timestamp, num_nodes,
avg_delay, avg_bandwidth, avg_cpu_usage) VALUES (?, ?, ?, ?, ?)",
                           (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), num_nodes,
avg_delay, avg_bandwidth, avg_cpu))
            self.db_connection.commit()
            logging.info("Scalability analysis: %d nodes, Avg Delay=%.2f ms, Avg
Bandwidth=%.2f Mbps, Avg CPU=%.2f%%", num_nodes, avg_delay, avg_bandwidth,
avg_cpu)

    def analyze_statistics(self) -> Dict[str, Dict[str, float]]:
        """Статистичний аналіз метрик для кожного протоколу і типу трафіку.

        Returns:
            Dict[str, Dict[str, float]]: Статистика (середнє, медіана, дисперсія) для
кожного протоколу.
        """
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT protocol, delay, bandwidth, cpu_usage, packet_loss,
traffic_type FROM metrics")
        data = cursor.fetchall()
        if not data:
```

```
            logging.warning("No data for statistical analysis")
            return {}

        df = pd.DataFrame(data, columns=["protocol", "delay", "bandwidth", "cpu_usage",
"packet_loss", "traffic_type"])
        stats = {}
        for protocol in df["protocol"].unique():
            for t_type in df["traffic_type"].unique():
                subset = df[(df["protocol"] == protocol) & (df["traffic_type"] == t_type)]
                if not subset.empty:
                    stats[f"{protocol}_{t_type}"] = {
                        "avg_delay": subset["delay"].mean(),
                        "median_delay": subset["delay"].median(),
                        "var_delay": subset["delay"].var() if len(subset) > 1 else 0,
                        "avg_bandwidth": subset["bandwidth"].mean(),
                        "median_bandwidth": subset["bandwidth"].median(),
                        "var_bandwidth": subset["bandwidth"].var() if len(subset) > 1 else 0,
                        "avg_cpu_usage": subset["cpu_usage"].mean(),
                        "avg_packet_loss": subset["packet_loss"].mean()
                    }
        logging.info("Statistical analysis completed: %s", stats)
        return stats

    def analyze_bottlenecks(self) -> List[Tuple[str, str, float]]:
        """Аналіз вузьких місць у мережі (ребра з низькою пропускною здатністю).

        Returns:
            List[Tuple[str, str, float]]: Список ребер із пропускною здатністю.
        """
        bottlenecks = []
        for u, v, data in self.graph.edges(data=True):
            bandwidth = data["bandwidth"]
            if bandwidth < 700:  # Поріг для визначення вузького місця
                bottlenecks.append((u, v, bandwidth))
        bottlenecks.sort(key=lambda x: x[2])
        logging.info("Bottlenecks detected: %s", bottlenecks)
        return bottlenecks

    def visualize_network(self, output_file: str = "network.png") -> None:
        """Візуалізація графа мережі з позначенням протоколів на ребрах.

        Args:
            output_file (str): Ім'я файлу для збереження графіка.
        """
        fig, ax = plt.subplots(figsize=(14, 10))
```

```
        pos = nx.spring_layout(self.graph)
        nx.draw(self.graph, pos, with_labels=True, node_color="lightblue",
node_size=2500, font_size=10, font_weight="bold", ax=ax)
        edge_labels = nx.get_edge_attributes(self.graph, "protocol")
        nx.draw_networkx_edge_labels(self.graph, pos, edge_labels=edge_labels,
font_color="red", font_size=8, ax=ax)

        output = io.BytesIO()
        canvas = FigureCanvas(fig)
        canvas.print_png(output)
        with open(output_file, "wb") as f:
            f.write(output.getvalue())
        logging.info("Network visualization saved as %s", output_file)
        plt.close()

    def visualize_metrics(self, output_file: str = "metrics.png") -> None:
        """Візуалізація метрик (затримка, пропускна здатність, використання CPU) за
протоколами і типами трафіку.

        Args:
            output_file (str): Ім'я файлу для збереження графіка.
        """
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT protocol, delay, bandwidth, cpu_usage, traffic_type
FROM metrics")
        data = cursor.fetchall()
        if not data:
            logging.warning("No metrics data available for visualization")
            return

        df = pd.DataFrame(data, columns=["protocol", "delay", "bandwidth", "cpu_usage",
"traffic_type"])
        fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(12, 18))
        for protocol in df["protocol"].unique():
            for t_type in df["traffic_type"].unique():
                subset = df[(df["protocol"] == protocol) & (df["traffic_type"] == t_type)]
                if not subset.empty:
                    ax1.plot(subset["delay"], label=f"{protocol} ({t_type})")
                    ax2.plot(subset["bandwidth"], label=f"{protocol} ({t_type})")
                    ax3.plot(subset["cpu_usage"], label=f"{protocol} ({t_type})")

        ax1.set_title("Delay (ms) by Protocol and Traffic Type")
        ax2.set_title("Bandwidth (Mbps) by Protocol and Traffic Type")
        ax3.set_title("CPU Usage (%) by Protocol and Traffic Type")
        ax1.legend()
```

```
        ax2.legend()
        ax3.legend()
        plt.tight_layout()

        output = io.BytesIO()
        canvas = FigureCanvas(fig)
        canvas.print_png(output)
        with open(output_file, "wb") as f:
            f.write(output.getvalue())
        logging.info("Metrics visualization saved as %s", output_file)
        plt.close()

    def generate_report(self, format: str = "text") -> str:
        """Генерація звіту з метрик симуляції.

        Args:
            format (str): Формат звіту (text, csv, json).

        Returns:
            str: Текстовий звіт або шлях до файлу.
        """
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT protocol, AVG(delay), AVG(bandwidth),
AVG(cpu_usage), AVG(packet_loss), SUM(attack_success)/COUNT(*) AS
attack_success_rate, traffic_type FROM metrics GROUP BY protocol, traffic_type")
        results = cursor.fetchall()

        if format == "text":
            report = "VPN Simulation Report\n" + "="*30 + "\n"
            for result in results:
                protocol, avg_delay, avg_bandwidth, avg_cpu, avg_packet_loss,
attack_success_rate, traffic_type = result
                report += (f"Protocol: {protocol}, Traffic Type: {traffic_type}\n"
                        f"Average Delay: {avg_delay:.2f} ms\n"
                        f"Average Bandwidth: {avg_bandwidth:.2f} Mbps\n"
                        f"Average CPU Usage: {avg_cpu:.2f}%\n"
                        f"Average Packet Loss: {avg_packet_loss*100:.2f}%\n"
                        f"Attack Success Rate: {attack_success_rate*100:.2f}%\n"
                        + "-"*30 + "\n")
            logging.info("Generated text report:\n%s", report)
            return report

        elif format == "csv":
            output_file = "vpn_report.csv"
            with open(output_file, "w", newline='') as f:
```

```
            writer = csv.writer(f)
            writer.writerow(["Protocol", "Traffic Type", "Avg Delay (ms)", "Avg
Bandwidth (Mbps)", "Avg CPU Usage (%)", "Avg Packet Loss (%)", "Attack Success
Rate (%)"])
            for result in results:
                protocol, avg_delay, avg_bandwidth, avg_cpu, avg_packet_loss,
attack_success_rate, traffic_type = result
                writer.writerow([protocol, traffic_type, f"{avg_delay:.2f}",
f"{avg_bandwidth:.2f}", f"{avg_cpu:.2f}", f"{avg_packet_loss*100:.2f}",
f"{attack_success_rate*100:.2f}"])
        logging.info("Generated CSV report: %s", output_file)
        return output_file

    elif format == "json":
        output_file = "vpn_report.json"
        report_data = []
        for result in results:
            protocol, avg_delay, avg_bandwidth, avg_cpu, avg_packet_loss,
attack_success_rate, traffic_type = result
            report_data.append({
                "protocol": protocol,
                "traffic_type": traffic_type,
                "avg_delay_ms": round(avg_delay, 2),
                "avg_bandwidth_mbps": round(avg_bandwidth, 2),
                "avg_cpu_usage_percent": round(avg_cpu, 2),
                "avg_packet_loss_percent": round(avg_packet_loss * 100, 2),
                "attack_success_rate_percent": round(attack_success_rate * 100, 2)
            })
        with open(output_file, "w") as f:
            json.dump(report_data, f, indent=4)
        logging.info("Generated JSON report: %s", output_file)
        return output_file

    else:
        logging.error("Unsupported report format: %s", format)
        return ""

  def analyze_security(self) -> Dict[str, float]:
    """Аналіз безпеки протоколів на основі історії атак.

    Returns:
        Dict[str, float]: Рівень безпеки (1 - ймовірність успішної атаки) для кожного
протоколу.
    """
    cursor = self.db_connection.cursor()
```

```
        cursor.execute("SELECT protocol, SUM(success)/COUNT(*) AS
attack_success_rate FROM attacks GROUP BY protocol")
        results = cursor.fetchall()
        security_scores = {}
        for protocol, attack_success_rate in results:
            security_scores[protocol] = 1 - attack_success_rate
        logging.info("Security analysis completed: %s", security_scores)
        return security_scores

    def simulate_network_load(self, num_requests: int, traffic_type: str = "FileTransfer")
-> None:
        """Симуляція навантаження мережі шляхом виконання кількох запитів.

        Args:
            num_requests (int): Кількість запитів для симуляції.
            traffic_type (str): Тип трафіку.
        """
        for _ in range(num_requests):
            source = random.choice(list(self.graph.nodes))
            destination = random.choice(list(self.graph.nodes))
            if source != destination:
                self.simulate_traffic(source, destination, traffic_type=traffic_type)

    def simulate_quantum_attack(self, protocol: str, intensity: float = 1.0) -> bool:
        """Симуляція квантової атаки на протокол.

        Args:
            protocol (str): Протокол для атаки.
            intensity (float): Інтенсивність атаки.

        Returns:
            bool: Успіх атаки.
        """
        attack_type = "QuantumAttack"
        duration = random.uniform(1.0, 10.0) * intensity
        if protocol == "PPTP":
            success_probability = 0.9 * intensity
        elif protocol == "L2TP/IPsec":
            success_probability = 0.4 * intensity
        else:  # OpenVPN, WireGuard
            success_probability = 0.1 * intensity
        attack_success = random.random() < min(success_probability, 1.0)

        cursor = self.db_connection.cursor()
```

```
        cursor.execute("INSERT INTO attacks (timestamp, protocol, attack_type, success,
duration) VALUES (?, ?, ?, ?, ?)",
                       (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), protocol,
attack_type, attack_success, duration))
        self.db_connection.commit()

        logging.info("Quantum attack simulation on %s: Success=%s, Duration=%.2f s,
Intensity=%.2f", protocol, attack_success, duration, intensity)
        return attack_success

    def export_network_config(self, output_file: str = "network_config.json") -> None:
        """Експорт конфігурації мережі у JSON-файл.

        Args:
            output_file (str): Ім'я файлу для збереження конфігурації.
        """
        config = {
            "nodes": list(self.graph.nodes),
            "edges": [
                {"source": u, "destination": v, **self.graph[u][v]}
                for u, v in self.graph.edges
            ]
        }
        with open(output_file, "w") as f:
            json.dump(config, f, indent=4)
        logging.info("Network configuration exported to %s", output_file)

    def __del__(self):
        """Закриття з'єднання з базою даних при знищенні об'єкта."""
        self.db_connection.close()
        logging.info("Database connection closed")

def parse_arguments() -> argparse.Namespace:
    """Парсинг аргументів командного рядка для налаштування симуляції.

    Returns:
        argparse.Namespace: Об'єкт із параметрами командного рядка.
    """
    parser = argparse.ArgumentParser(description="VPN Network Simulator")
    parser.add_argument("--nodes", type=int, default=20, help="Number of nodes to
initialize")
    parser.add_argument("--requests", type=int, default=10, help="Number of traffic
simulation requests per traffic type")
    parser.add_argument("--scale", type=int, default=5, help="Number of nodes to add
during scaling")
```

```
    parser.add_argument("--max-nodes", type=int, default=30, help="Maximum number of nodes for scalability analysis")
    parser.add_argument("--output-dir", type=str, default=".", help="Directory for output files")
    parser.add_argument("--attack-intensity", type=float, default=1.0, help="Intensity of simulated attacks")
    return parser.parse_args()

class TestVPNSimulator(unittest.TestCase):
    """Модульні тести для класу VPNSimulator."""

    def setUp(self):
        """Ініціалізація тестового симулятора перед кожним тестом."""
        self.simulator = VPNSimulator("test_vpn_simulation.db")
        self.simulator.add_nodes(5)
        connections = [
            ("AS1", "AS2", "WireGuard"),
            ("AS2", "AS3", "OpenVPN"),
            ("AS3", "AS4", "L2TP/IPsec"),
            ("AS4", "AS5", "PPTP")
        ]
        self.simulator.add_vpn_edges(connections)

    def test_add_nodes(self):
        """Тест додавання вузлів до графа."""
        self.assertEqual(len(self.simulator.graph.nodes), 5)
        self.assertIn("AS1", self.simulator.graph.nodes)

    def test_add_edges(self):
        """Тест додавання VPN-з'єднань між вузлами."""
        self.assertTrue(self.simulator.graph.has_edge("AS1", "AS2"))
        self.assertEqual(self.simulator.graph["AS1"]["AS2"]["protocol"], "WireGuard")

    def test_simulate_traffic(self):
        """Тест симуляції трафіку між вузлами."""
        result = self.simulator.simulate_traffic("AS1", "AS2")
        self.assertIsNotNone(result)
        delay, bandwidth, packet_loss, attack_success = result
        self.assertGreaterEqual(delay, 12)
        self.assertGreaterEqual(bandwidth, 0)
        self.assertGreaterEqual(packet_loss, 0)

    def test_simulate_attack(self):
        """Тест симуляції атаки на протокол."""
        result = self.simulator.simulate_attack("PPTP")
```

```
        self.assertIn(result, [True, False])

    def test_simulate_ddos(self):
        """Тест симуляції DDoS-атаки на вузол."""
        result = self.simulator.simulate_ddos_attack("AS1")
        self.assertIn(result, [True, False])

    def test_configure_vpn(self):
        """Тест генерації Ansible-скрипту для конфігурації VPN."""
        script = self.simulator.configure_vpn_server("AS1", "WireGuard")
        self.assertIn("WireGuard", script)

    def test_generate_report(self):
        """Тест генерації звіту у текстовому форматі."""
        self.simulator.simulate_traffic("AS1", "AS2")
        report = self.simulator.generate_report("text")
        self.assertIn("Protocol: WireGuard", report)

    def test_analyze_security(self):
        """Тест аналізу безпеки протоколів."""
        self.simulator.simulate_traffic("AS1", "AS2")
        scores = self.simulator.analyze_security()
        self.assertIn("WireGuard", scores)

    def test_scale_network(self):
        """Тест масштабування мережі."""
        initial_nodes = len(self.simulator.graph.nodes)
        self.simulator.scale_network(5)
        self.assertEqual(len(self.simulator.graph.nodes), initial_nodes + 5)

    def test_simulate_failure(self):
        """Тест симуляції збою вузла."""
        self.simulator.simulate_failure(node="AS1")
        self.assertNotIn("AS1", self.simulator.graph.nodes)

    def test_simulate_quantum_attack(self):
        """Тест симуляції квантової атаки."""
        result = self.simulator.simulate_quantum_attack("PPTP")
        self.assertIn(result, [True, False])

    def test_export_config(self):
        """Тест експорту конфігурації мережі."""
        self.simulator.export_network_config("test_config.json")
        self.assertTrue(os.path.exists("test_config.json"))
```

```
    def test_analyze_statistics(self):
        """Тест статистичного аналізу метрик."""
        self.simulator.simulate_traffic("AS1", "AS2")
        stats = self.simulator.analyze_statistics()
        self.assertIn("WireGuard_FileTransfer", stats)

    def test_analyze_bottlenecks(self):
        """Тест аналізу вузьких місць у мережі."""
        bottlenecks = self.simulator.analyze_bottlenecks()
        for _, _, bandwidth in bottlenecks:
            self.assertLess(bandwidth, 700)

    def tearDown(self):
        """Очистка після тестів."""
        self.simulator.db_connection.close()
        os.remove("test_vpn_simulation.db")
        if os.path.exists("test_config.json"):
            os.remove("test_config.json")

if __name__ == "__main__":
    args = parse_arguments()

    # Ініціалізація симулятора
    simulator = VPNSimulator()

    # Додавання вузлів
    simulator.add_nodes(args.nodes)

    # Додавання з'єднань
    connections = [
        ("AS1", "AS2", "WireGuard"),
        ("AS2", "AS3", "OpenVPN"),
        ("AS3", "AS4", "L2TP/IPsec"),
        ("AS4", "AS5", "PPTP"),
        ("AS5", "AS6", "WireGuard"),
        ("AS6", "AS7", "OpenVPN"),
        ("AS7", "AS8", "L2TP/IPsec"),
        ("AS8", "AS9", "PPTP"),
        ("AS9", "AS10", "WireGuard"),
        ("AS10", "AS11", "OpenVPN"),
        ("AS11", "AS12", "L2TP/IPsec"),
        ("AS12", "AS13", "PPTP"),
        ("AS13", "AS14", "WireGuard"),
        ("AS14", "AS15", "OpenVPN"),
        ("AS15", "AS16", "L2TP/IPsec"),
```

```
        ("AS16", "AS17", "PPTP"),
        ("AS17", "AS18", "WireGuard"),
        ("AS18", "AS19", "OpenVPN"),
        ("AS19", "AS20", "L2TP/IPsec"),
        ("AS20", "AS1", "PPTP")
    ]
    simulator.add_vpn_edges(connections)

    # Симуляція трафіку для різних типів
    for traffic_type in simulator.traffic_types:
        print(f"\nSimulating {traffic_type} traffic:")
        for _ in range(args.requests):
            source = random.choice(list(simulator.graph.nodes))
            destination = random.choice(list(simulator.graph.nodes))
            if source != destination:
                result = simulator.simulate_traffic(source, destination,
traffic_type=traffic_type)
                if result:
                    delay, bandwidth, packet_loss, attack_success = result
                    print(f"Traffic from {source} to {destination}: Type={traffic_type},
Delay={delay:.2f} ms, Bandwidth={bandwidth:.2f} Mbps, Packet
Loss={packet_loss*100:.2f}%, Attack Success={attack_success}")

    # Масштабування мережі
    simulator.scale_network(args.scale)

    # Аналіз масштабованості
    simulator.analyze_scalability(max_nodes=args.max_nodes)

    # Симуляція DDoS-атаки
    simulator.simulate_ddos_attack("AS1", intensity=args.attack_intensity)

    # Симуляція збою
    simulator.simulate_failure(node="AS2")

    # Симуляція квантової атаки
    print("\nSimulating Quantum Attack on WireGuard:")
    print(f"Quantum Attack Success: {simulator.simulate_quantum_attack('WireGuard',
intensity=args.attack_intensity)}")

    # Конфігурація VPN-сервера
    print("\nAnsible Script for AS1 (WireGuard):")
    print(simulator.configure_vpn_server("AS1", "WireGuard"))

    # Аналіз вузьких місць
```

```
    print("\nNetwork Bottlenecks:")
    bottlenecks = simulator.analyze_bottlenecks()
    for u, v, bandwidth in bottlenecks:
        print(f"Edge {u}-{v}: Bandwidth={bandwidth:.2f} Mbps")

    # Візуалізація
    simulator.visualize_network(os.path.join(args.output_dir, "network.png"))
    simulator.visualize_metrics(os.path.join(args.output_dir, "metrics.png"))

    # Генерація звітів
    print("\nText Report:")
    print(simulator.generate_report("text"))
    simulator.generate_report("csv")
    simulator.generate_report("json")

    # Аналіз безпеки
    print("\nSecurity Scores:")
    security_scores = simulator.analyze_security()
    print(security_scores)

    # Статистичний аналіз
    print("\nStatistical Analysis:")
    stats = simulator.analyze_statistics()
    print(json.dumps(stats, indent=4))

    # Експорт конфігурації
    simulator.export_network_config(os.path.join(args.output_dir,
"network_config.json"))

    # Запуск тестів
    unittest.main(argv=[''], exit=False)
```