SECTION 5. COMPUTER ENGINEERING

DOI: 10.46299/ISG.2025.MONO.TECH.3.5.1

5.1 Методи створення відмовостійкості та доступності серверів ліцензування

5.1.1 Сучасні системи ліцензування з використанням апаратних засобів захисту

У сучасному світі програмне забезпечення стало одним із найважливіших ресурсів, що визначають розвиток технологій, бізнесу, науки та освіти. Його створення потребує значних інтелектуальних і фінансових витрат, тому захист результатів програмної діяльності є одним із ключових завдань розробників і компаній, які постачають ПЗ на ринок. У цьому контексті ліцензування програмного забезпечення виступає не лише юридичним інструментом регулювання відносин між розробником і користувачем, а й технічним механізмом, що забезпечує контроль за використанням програмного продукту.

Поняття «ліцензія» в контексті програмного забезпечення означає надання користувачеві певних прав на встановлення, використання або розповсюдження програми згідно з умовами розробника чи власника авторських прав. Ліцензійна угода визначає кількість дозволених інсталяцій, тип доступу (персональний, корпоративний, мережевий), термін дії, а також функціональні можливості, що залежать від рівня підписки або купленої редакції. Саме через механізми ліцензування виробник має змогу контролювати обсяг використання ПЗ, захищати свій продукт від несанкціонованого копіювання, а також отримувати дохід, необхідний для подальшого розвитку[117].

Значення ліцензування особливо зростає у галузях, де програмне забезпечення є не просто інструментом, а критичною складовою виробничих процесів. Це системи автоматизованого проєктування (CAD/CAM/CAE), інженерні комплекси, геоінформаційні системи, фінансові застосунки, медичне та наукове ПЗ. Такі продукти, як правило, мають високу вартість і потребують

спеціального контролю за доступом до функціоналу. Відсутність належного захисту призводить до значних економічних втрат розробників через піратство та нелегальне використання ліцензій.

Історично перші системи ліцензування реалізовувалися у вигляді програмних ключів – унікальних кодів, що активували програму на конкретному комп'ютері. Проте з поширенням глобальних мереж та віртуалізації цей підхід виявився недостатньо безпечним: ключі можна було копіювати або зламувати. У відповідь на ці виклики були створені апаратні засоби ліцензування, або апаратні ключі (рис. 1). Вони забезпечують апаратний рівень захисту, що робить несанкціоноване копіювання програмного забезпечення значно складнішим або практично неможливим.

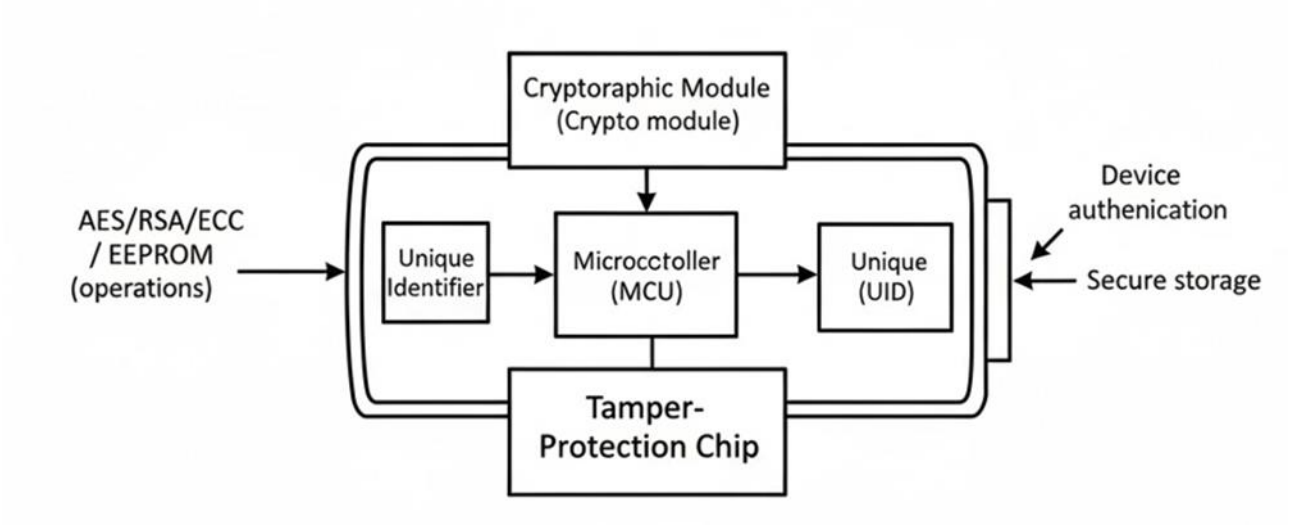

Рисунок 1. Схема апаратного ключа [авторська розробка]

Апаратний ключ виконує роль фізичного носія ліцензії – він містить зашифровану інформацію про права користувача, унікальний ідентифікатор, криптографічні ключі та алгоритми перевірки автентичності. Такий пристрій зазвичай підключається до комп'ютера через USB-інтерфейс або інтегрується у локальну мережу як ліцензійний сервер. Програма звертається до ключа при запуску або виконанні певних функцій, перевіряючи наявність і дійсність ліцензії. У разі відсутності ключа або невідповідності даних виконання програми блокується.

Із технічного погляду, сучасні системи ліцензування з апаратними ключами виконують кілька важливих функцій:

- ідентифікація користувача або робочого місця кожен ключ має унікальний ідентифікатор,

- контроль кількості активних ліцензій для мережевих продуктів обмежується кількість одночасних підключень,

- захист від підроблення або емуляції за рахунок криптографічних протоколів та апаратного шифрування,

- автоматичне відключення при порушеннях, наприклад, під час спроби перенесення ключа без авторизації.

Таким чином, ліцензування виконує подвійну функцію економічну (захист комерційних інтересів розробника) та технічну (забезпечення безпеки програмного продукту). Його роль стає ще важливішою у корпоративних середовищах, де від безперервності роботи ліцензійних серверів залежить функціонування багатьох бізнес-процесів.

В умовах глобальної цифровізації та поширення контейнеризованих інфраструктур виникає потреба у нових підходах до організації ліцензування. Сучасні компанії прагнуть інтегрувати апаратні ключі у віртуальні середовища, розподілені кластери та хмарні платформи. Це відкриває нові можливості, але водночас створює низку проблем, пов’язаних із відмовостійкістю, масштабованістю та безпекою доступу до фізичних пристроїв[118].

Отже, ліцензування у сучасних програмних продуктах є фундаментальним механізмом не лише захисту авторських прав, а й забезпечення стабільної, контрольованої та безпечної експлуатації програмних систем. Його розвиток безпосередньо впливає на економічну ефективність, надійність і конкурентоспроможність програмних рішень, а тому дослідження методів удосконалення ліцензування, зокрема в контейнеризованих середовищах, є актуальним завданням сучасної інформаційної інженерії.

Системи ліцензування з використанням апаратних засобів захисту є одним із найефективніших інструментів протидії несанкціонованому використанню

програмного забезпечення. Їхня поява стала логічним етапом еволюції засобів захисту авторських прав – від простих програмних ключів і серійних номерів до апаратних пристроїв, що реалізують складні криптографічні алгоритми. Такі рішення використовуються в багатьох комерційних, інженерних та промислових продуктах, де втрати від піратства можуть перевищувати витрати на впровадження захисних технологій.

Основна ідея апаратного захисту полягає в тому, що для запуску або роботи програмного забезпечення необхідна фізична наявність спеціального пристрою – апаратного ключа. Цей ключ, як правило, підключається через інтерфейс USB або використовується як мережевий модуль, який зберігає зашифровану інформацію про ліцензію, криптографічні ключі та алгоритми перевірки автентичності. Програма взаємодіє з ним за допомогою спеціального API або драйвера, виконуючи перевірку перед початком роботи.

Переваги систем ліцензування з апаратними засобами захисту:

- високий рівень безпеки. На відміну від програмних ключів, які можуть бути скопійовані або зламані, апаратний ключ містить криптографічно захищену інформацію, до якої неможливо отримати доступ без фізичного пристрою. Більшість сучасних ключів підтримують шифрування з використанням алгоритмів AES, RSA, ECC, а також реалізують механізми захисту від реверс-інжинірингу та апаратного злому (апарат виявлення, самознищення даних при спробі зламу тощо);
- надійна ідентифікація користувача. Кожен ключ має унікальний ідентифікатор, що дозволяє точно визначати власника ліцензії, кількість активних копій програми та права доступу. Це особливо важливо для корпоративних і мережевих середовищ, де контроль за кількістю одночасних користувачів є обов’язковим;
- захист від підроблення та емуляції. Апаратні ключі використовують унікальні криптографічні обчислення (наприклад, challenge–response), що унеможливлює створення програмного аналога. Навіть у разі копіювання програмного коду, відсутність справжнього ключа блокує роботу ПЗ;

- незалежність від мережевих збоїв або серверів авторизації. У локальному режимі роботи апаратний ключ забезпечує автономну перевірку ліцензії без потреби постійного з'єднання з сервером, що знижує ризики зупинки програмного продукту через проблеми з мережею;

- довготривала експлуатація. Більшість сучасних апаратних ключів мають термін служби понад 10 років, не потребують складного обслуговування і сумісні з різними операційними системами.

Недоліки систем ліцензування з апаратними засобами захисту:

- залежність від фізичного пристрою. Основною проблемою є необхідність фізичного підключення ключа. У разі його пошкодження, втрати або виходу з ладу користувач може повністю втратити доступ до програмного забезпечення, доки не буде отримано заміну. Це створює ризики простою, особливо у випадку віддалених або критично важливих систем;

- складність інтеграції у контейнеризовані та хмарні середовища. Апаратні ключі розроблялися для традиційних фізичних серверів або локальних робочих станцій. У контейнерах або віртуальних машинах їх підключення потребує спеціальних механізмів, таких як USB-over-IP або емуляція пристроїв, що часто призводить до проблем з продуктивністю, безпекою чи сумісністю;

- обмежена масштабованість. На відміну від програмних ліцензій, які можна легко масштабувати шляхом активації додаткових ключів у базі даних, апаратні рішення вимагають наявності фізичних пристроїв. Це ускладнює швидке розширення інфраструктури або міграцію на інші платформи;

- висока вартість впровадження. Вартість апаратних ключів, драйверів і серверів ліцензій є суттєвою. Для великих організацій із сотнями користувачів це може призвести до значних початкових витрат (CAPEX), а також витрат на підтримку (OPEX). Необхідність резервування та управління.

Для забезпечення відмовостійкості потрібне дублювання ключів або побудова кластерних ліцензійних серверів, що вимагає додаткової інфраструктури, моніторингу та адміністрування.

Таким чином, системи ліцензування з апаратними засобами захисту мають високий рівень безпеки, стабільність та надійність, що робить їх оптимальними для критично важливих застосунків і комерційного програмного забезпечення з високою вартістю. Проте їх основними недоліками залишаються залежність від фізичного пристрою, складність інтеграції у сучасні контейнеризовані інфраструктури та висока вартість володіння.

Ці фактори зумовлюють актуальність подальших досліджень і розроблення методів підвищення відмовостійкості та доступності серверів ліцензування, які поєднували б переваги апаратного захисту з гнучкістю сучасних хмарних і контейнерних технологій.

5.1.2 Підходи до побудови відмовостійких сервісів із використанням апаратних ключів

Із розвитком інформаційних технологій, хмарних платформ і контейнеризації особливої актуальності набули питання забезпечення відмовостійкості та високої доступності серверних систем. Для ліцензійних серверів, що використовують апаратні ключі як засіб автентифікації та контролю доступу до програмного забезпечення, ці завдання мають специфічний характер, оскільки такі ключі є фізичними пристроями, які мають обмеження у масштабуванні, переміщенні й дублюванні. Забезпечення безперервної роботи подібних систем потребує комплексного підходу, який враховує як програмну, так і апаратну складові.

Традиційно найпростішим способом побудови ліцензійного сервера є локальне підключення апаратного ключа до фізичного вузла, де виконується програма-сервер. Така архітектура має мінімальні затримки, не потребує складних мережевих налаштувань і є ефективною для невеликих підприємств або лабораторій. Проте її головним недоліком є наявність єдиної точки відмови: у разі виходу з ладу сервера або самого ключа вся система стає недоступною. Для великих корпоративних середовищ це неприпустимо, тому виникла потреба у розподілених архітектурних рішеннях [119].

Одним із підходів є використання мережевих апаратних ключів, які функціонують як окремі сервери ліцензій. У цьому випадку кілька користувачів або застосунків можуть одночасно підключатися до спільного ключа через локальну мережу. Така архітектура дозволяє централізовано керувати ліцензіями, відслідковувати їхнє використання та забезпечувати контроль доступу на рівні користувачів. Водночас вона також створює ризики – у разі відмови вузла з ключем або мережевого інтерфейсу система втрачає доступ до ліцензій, що може призвести до зупинки критичних процесів. Для мінімізації таких ризиків застосовуються механізми резервування, дублювання серверів і використання балансувальників навантаження.

Сучасні архітектури з високою доступністю (High Availability) реалізуються за схемами active-passive або active-active (рис. 2). У першому випадку основний сервер має ексклюзивний доступ до апаратного ключа, тоді як резервний перебуває у режимі очікування. У разі збою система автоматично перемикається на резервний вузол. Такий підхід забезпечує передбачувану поведінку системи, однак потребує коректної синхронізації станів і механізмів “fencing”, щоб уникнути ситуацій, коли два сервери одночасно намагаються звертатися до одного ключа.

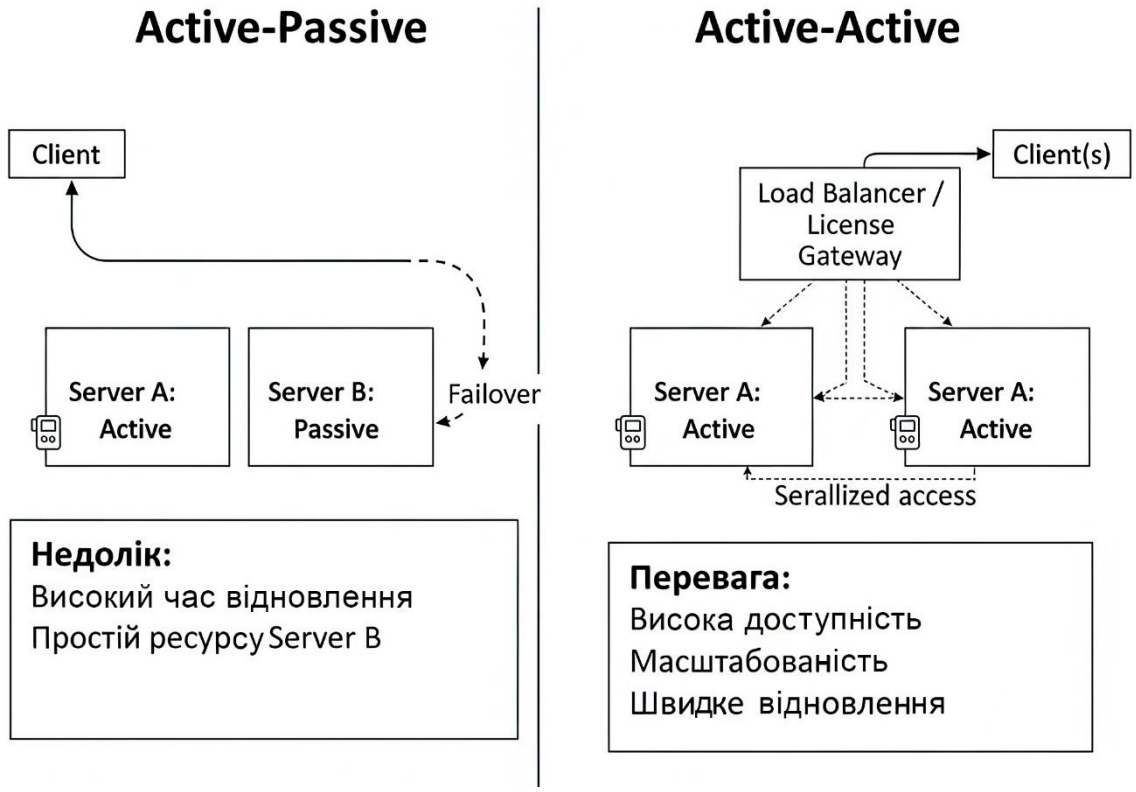

Рисунок 2. Архітектури з високою доступністю на схемах взаємодії active-active, active-passive [авторська розробка]

У схемі active-active декілька серверів або контейнерів можуть одночасно обслуговувати клієнтів, але звернення до апаратного ключа відбувається через спеціальний програмний шлюз або сервіс-посередник. Цей шлюз виконує серіалізацію запитів, забезпечує контроль черги, а також веде журнал операцій видачі й повернення ліцензій. Така модель більш гнучка, вона дозволяє масштабувати клієнтське навантаження, однак вимагає особливої уваги до надійності самого шлюзу, який стає критичним елементом системи.

В умовах контейнеризації, коли сервіси розгортаються у середовищах на зразок Kubernetes, з’являються додаткові інструменти для реалізації відмовостійкості. Для підключення апаратних ключів у таких середовищах використовують Device Plugin (рис. 3) – спеціальний компонент, який інформує оркестратор про наявність доступного апаратного ресурсу.

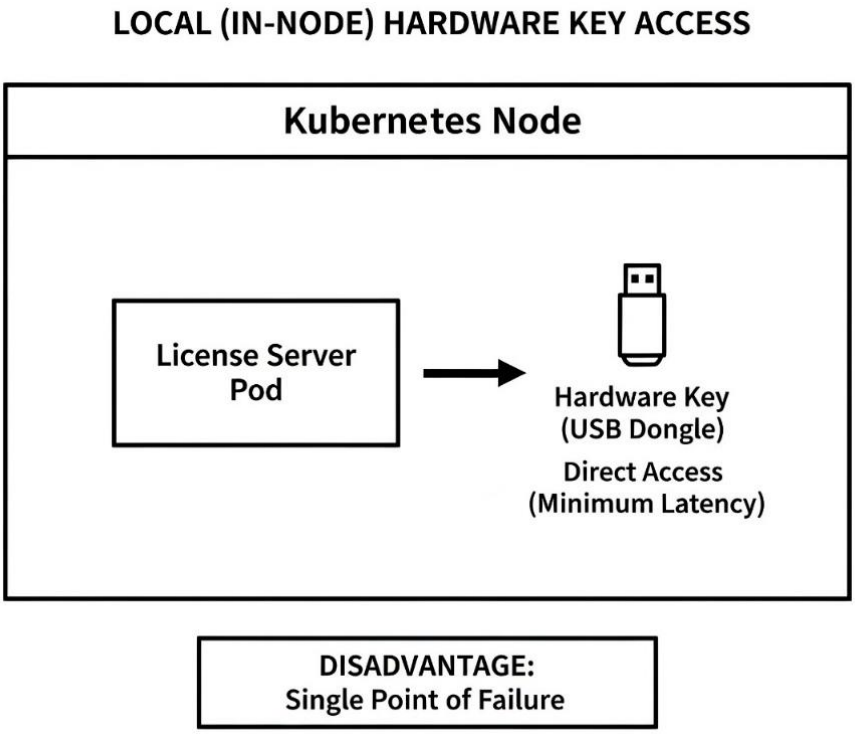

Рисунок 3. Схема підключення Device Plugin у кластер Kubernetes [авторська розробка]

Завдяки цьому можна забезпечити, щоб контейнери, які потребують доступу до ключа, розміщувалися виключно на вузлах, де такий ключ підключено. Додаткові механізми, як-от PodDisruptionBudget, допомагають обмежити кількість одночасно недоступних контейнерів під час оновлень або перезапусків, а Topology Spread Constraints забезпечують рівномірний розподіл реплік по вузлах кластера, що знижує ймовірність одночасного збою.

Особливу роль у побудові відмовостійких систем відіграють health-checks та watchdog-сервіси, які постійно перевіряють стан апаратного ключа, драйвера та ліцензійного демона. У разі втрати зв’язку або збоїв вони ініціюють автоматичне перезапускання контейнера або перемикання на резервний вузол. У поєднанні з механізмами leader election це дозволяє досягти автоматичного відновлення працездатності без втручання адміністратора [121].

Важливим аспектом також є безпека. Оскільки апаратний ключ є джерелом істини для ліцензійної системи, доступ до нього має бути захищений на всіх рівнях – від фізичного до мережевого. Для цього використовуються протоколи

взаємної автентифікації (mTLS), ролева модель доступу (RBAC), ізольовані мережеві простори (NetworkPolicy), а також секрет-менеджери, які відповідають за безпечне зберігання криптографічних параметрів. Таким чином, забезпечується як цілісність процесу ліцензування, так і захист від внутрішніх і зовнішніх загроз.

У розподілених інфраструктурах, де доступ до апаратних ключів здійснюється через мережу, важливим завданням є мінімізація затримок і забезпечення стабільного каналу зв'язку. Для цього застосовуються технології VPN, Zero-Trust Network Access, а також системи моніторингу, що відстежують продуктивність ліцензійних запитів, рівень помилок та час відгуку (latency). За допомогою метрик типу p95 або p99 адміністратори можуть вчасно виявляти деградацію продуктивності та запобігати аваріям.

Також одним із ключових аспектів побудови відмовостійких сервісів є забезпечення високої доступності системи зберігання даних. Оскільки апаратний ключ є лише одним із компонентів комплексної інфраструктури захисту, стабільність роботи програмного сервісу значною мірою залежить від доступності бази даних, у якій зберігаються метадані ліцензій, журнали аудиту та конфігураційні параметри. У разі недоступності бази втрачається можливість фіксації операцій, виконання перевірок ліцензій або обробки транзакцій, що може призвести до блокування сервісу навіть за наявності справного апаратного ключа. Тому архітектура системи повинна враховувати механізми реплікації, автоматичного перемикання між вузлами та розподілення навантаження [120].

Залежно від вимог до узгодженості даних, продуктивності та масштабованості можуть застосовуватися різні типи баз даних.

Традиційні реляційні системи керування базами даних (PostgreSQL, MySQL, SQL Server) забезпечують високу цілісність і транзакційність, що є критичним для ліцензійних операцій, де неприпустимі дублікати або некоректні стани. Для забезпечення доступності ці системи підтримують синхронну та асинхронну реплікацію, режим primary–standby, а також кворумні кластери на основі групової реплікації.

Нереляційні бази даних (MongoDB, Cassandra, Redis), натомість, краще підходять для високонавантажених і географічно розподілених систем, де важливо мінімізувати латентність і забезпечити горизонтальне масштабування.

У кластерних середовищах із використанням апаратних ключів доцільно комбінувати кілька типів сховищ залежно від задачі: реляційні СУБД використовуються для зберігання критичних параметрів ліцензування та аудиту, тоді як нереляційні сховища можуть застосовуватись для кешування частих запитів, зниження навантаження на основну базу та прискорення обробки операцій. Забезпечення високої доступності реалізується шляхом використання операторів Kubernetes (наприклад, Patroni для PostgreSQL або MongoDB Operator), механізмів автоматичного відновлення (self-healing), розподілених транзакцій та балансування навантаження. Таким чином система отримує здатність продовжувати роботу навіть за умов відмови окремих вузлів або цілих сегментів інфраструктури, що є фундаментальним для проєктування відмовостійких сервісів із апаратними ключами.

Отже, архітектура відмовостійких сервісів із використанням апаратних ключів має поєднувати кілька взаємопов’язаних рівнів – фізичний, програмний та оркестраційний. Ефективність таких систем визначається не лише наявністю резервних вузлів чи дублюванням обладнання, а й узгодженою роботою механізмів моніторингу, балансування навантаження, серіалізації доступу до ключа та контролю безпеки. Архітекрута Active-Active (рис 1.3) повністю відповідає вимогам до доступності та оптимізує використання обчислювальних ресурсів (серверів). Кількість серверів можна збільшувати відповідно до навантаження що забезпечить високу доступність. Отже для подови програмного продукту буде доцільно обрати саме її.

5.1.3 Вимоги до відмовостійкості систем

Відмовостійкість – це здатність системи зберігати свою функціональність у разі виникнення збоїв окремих компонентів, забезпечуючи безперервність виконання основних процесів. Для серверів ліцензування, які взаємодіють з

апаратними ключами, вимоги до відмовостійкості є критично важливими, оскільки будь-яка втрата доступу до ключа призводить до зупинки роботи програмного забезпечення, що може спричинити значні фінансові та операційні втрати [122].

Загальні вимоги до відмовостійких систем визначаються на основі трьох ключових критеріїв: доступності (availability), надійності (reliability) та відновлюваності (recoverability) (рис. 4).

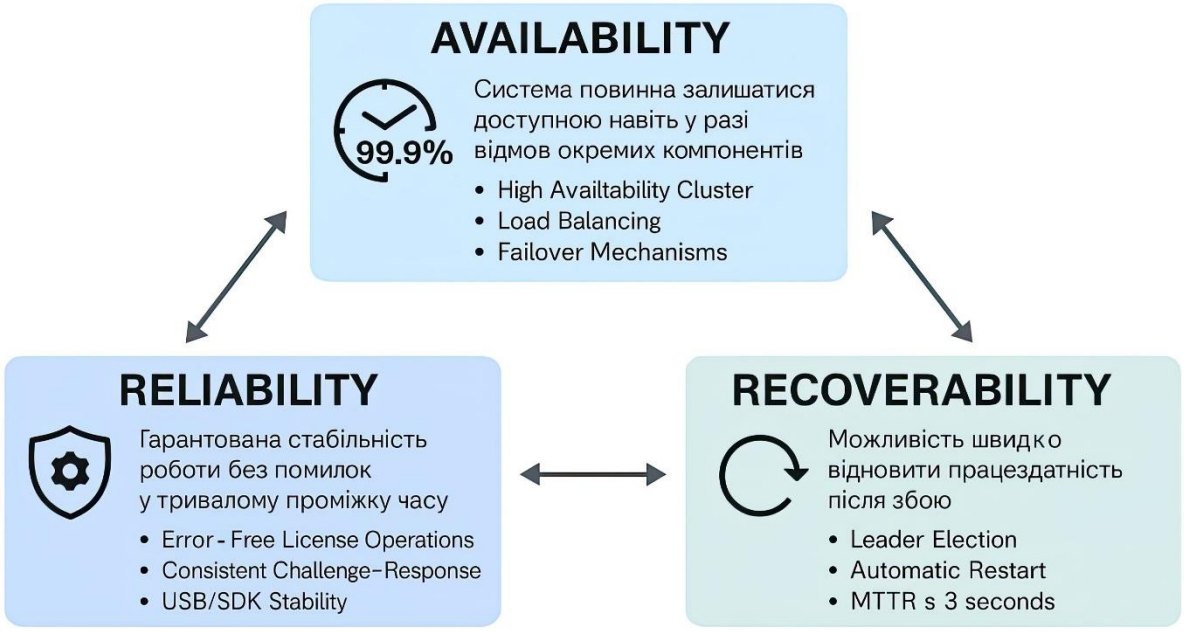

Рисунок 4. Складові відмовостійкості системи ліцензування з апаратними ключами [авторська розробка]

Доступність характеризує відсоток часу, протягом якого система працює безперебійно. Для критичних сервісів, таких як ліцензійні сервери, типовими орієнтирами є показники 99.9% (так звана “три дев’ятки”) або вище. Це означає, що сумарний час простою не повинен перевищувати кількох хвилин на місяць на табл. 1. [123]

Надійність визначається здатністю системи виконувати свої функції без збоїв протягом певного періоду часу, а відновлюваність – часом, необхідним для повернення системи до робочого стану після відмови.

Таблиця 1. Рівні доступності сервісу та відповідний час простою

Відсоток безвідмовної роботи	Щомісячний простій	Щорічний простій
99.5%	3,6 години	43,8 години
99,9%	43 хвилини	8,76 години
99,99%	4,3 хвилини	52,6 хвилини
99,999%	26,3 секунди	5,256 хвилини

Для забезпечення таких характеристик у системах із апаратними ключами мають бути реалізовані певні технічні й організаційні вимоги.

По-перше, необхідно передбачити фізичне резервування компонентів – використання декількох вузлів або серверів, які можуть дублювати функції основного у разі його виходу з ладу. Якщо ключ є єдиним фізичним пристроєм, то слід передбачити наявність резервного ключа з ідентичними правами або використання схеми “primary-secondary” із можливістю швидкого перемикання доступу.

По-друге, система повинна мати механізми автоматичного виявлення відмов і перемикання (failover). Це передбачає безперервний моніторинг стану апаратного ключа, ліцензійного демона, мережевого з’єднання та контейнерів, у яких вони працюють. При втраті зв’язку або виявленні помилки система має самостійно ініціювати перезапуск або переключення на резервний вузол без втручання оператора.

По-третє, важливим є забезпечення цілісності даних та синхронізації станів між активними й резервними компонентами. Усі зміни ліцензійних даних, лічильників або журналів доступу повинні зберігатися у транзакційному вигляді та передаватися в резервну копію для уникнення втрати інформації при відмові.

Ще однією вимогою є ізоляція відмов, тобто запобігання каскадному впливу збоїв. Вихід з ладу одного вузла або контейнера не повинен впливати на стабільність усього кластера. Для цього використовуються механізми оркестрації, такі як PodDisruptionBudget, anti-affinity, node taints та topology

spread constraints, які обмежують кількість одночасно недоступних компонентів і забезпечують їх розподіл по різних фізичних або віртуальних хостах.

Важливо також визначити вимоги до моніторингу та спостережуваності. Система має збирати показники часу відгуку, кількості успішних та неуспішних запитів, стану ключа, продуктивності мережі, використання ресурсів та інших параметрів, що характеризують її стабільність. Наявність таких метрик дозволяє проактивно виявляти проблеми та вживати заходів до настання критичних відмов.

Не менш значущим є питання безпеки, яке тісно пов’язане з відмовостійкістю. Будь-яке втручання в роботу апаратного ключа, спроби несанкціонованого підключення або підміни пристрою можуть призвести до зупинки системи, тому важливо реалізувати контроль автентичності компонентів (mTLS).

Щодо експлуатаційних вимог, система повинна підтримувати оновлення без простоїв (rolling updates) і мати механізми rollback у разі невдалого оновлення програмного забезпечення. Це особливо актуально для середовищ, у яких обслуговування ліцензійного сервера відбувається без зупинки критичних бізнес-процесів.

Крім того, важливо забезпечити географічну відмовостійкість – можливість роботи системи у разі втрати цілого дата-центру або мережевого сегмента. Для цього застосовується реплікація ключів або ліцензійних серверів у різних зонах доступності (Availability Zones), синхронізація станів та глобальне балансування навантаження з автоматичним маршрутизацією трафіку [124].

Також однією з ключових вимог до відмовостійкості апаратного ключа є його здатність продовжувати роботу у випадку часткової втрати доступу до окремих компонентів або інфраструктурних сервісів. Оскільки апаратний ключ є фізичним носієм логіки ліцензування, його відмова безпосередньо впливає на доступність програмного забезпечення. Тому пристрій має підтримувати автономний режим функціонування, зберігати працездатність при тимчасовій втраті мережевого з’єднання, забезпечувати захищену обробку запитів та

гарантувати цілісність криптографічних операцій незалежно від стану зовнішніх ресурсів. До базових вимог входять також захист від фізичного доступу, стійкість до перебоїв живлення та здатність до автоматичного відновлення після перезапуску.

Окремим аспектом відмовостійкості апаратного ключа є наявність GPS-модуля, який забезпечує контроль місцезнаходження пристрою та виступає додатковим механізмом виявлення аномалій. У випадку втрати супутникового сигналу, появи надмірних похибок координат або повної недоступності GPS, ключ повинен переходити в безпечний режим роботи, у якому криптографічні операції виконуються з використанням резервних параметрів або обмеженої функціональності. Це запобігає ситуаціям, коли тимчасові порушення у прийомі GPS-сигналу можуть призвести до збою ліцензійної системи або помилкового блокування користувачів. Важливою вимогою є також наявність допустимого діапазону похибки, який дозволяє уникнути помилкових спрацювань захисту під час природних флуктуацій супутникових даних.

GPS-модуль виконує також функцію сенсора безпеки, що дозволяє виявляти спроби фізичного переміщення або викрадення ключа. Відповідно, вимоги до відмовостійкості включають механізм поведінки пристрою у разі різкої зміни координат або виходу за межі дозволеного геоперіметру. За таких ситуацій апаратний ключ повинен негайно активувати захисні процедури: перейти у режим часткового або повного блокування, змінити параметри шифрування, сформувати подію безпеки для серверної інфраструктури та передати відповідні сповіщення до системи моніторингу. Це забезпечує можливість оперативного реагування та мінімізує ризик компрометації, навіть якщо фізичний доступ до пристрою був отриманий зловмисником.

Отже, узагальнюючи, основними вимогами до відмовостійких систем ліцензування з апаратними ключами є:

- наявність резервування фізичних і програмних компонентів,
- автоматичне виявлення відмов і перемикання на резерв,
- забезпечення цілісності й узгодженості даних,

- ізоляція збоїв та обмеження їхнього впливу,
- безперервний моніторинг, аудит і звітність,
- гарантії безпеки доступу до ключів,
- підтримка оновлень без зупинки роботи,
- можливість географічного відновлення після катастрофічних відмов.

Виконання зазначених вимог є передумовою побудови надійної системи ліцензування, здатної функціонувати в умовах сучасних контейнеризованих середовищ, забезпечуючи високу доступність, мінімальний час простою та стабільну роботу критичних сервісів.

5.1.4 Метод забезпечення відмовостійкості та доступності серверів ліцензування з апаратними ключами в контейнеризованому середовищі

Метод забезпечення відмовостійкості серверів ліцензування ґрунтується на необхідності адаптації традиційних підходів до роботи з апаратними ключами в умовах сучасних контейнеризованих і хмарних середовищ. Головна проблема полягає у тому, що фізичний апаратний ключ, будучи матеріальним носієм ліцензії, не може бути безпосередньо масштабований або дубльований як програмний ресурс. Тому такий метод має забезпечити логічне розширення доступності такого ключа за рахунок розподілу обчислювальних навантажень, організації керованого доступу до нього та використання механізмів автоматичного відновлення у разі збоїв.

Сутність методу забезпечення відмовостійкості серверів ліцензування полягає у поєднанні програмно-апаратного резервування із засобами оркестрації контейнерів. При цьому, основна ідея – це відокремити апаратний ресурс (ключ) від програмних компонентів, які його використовують, і надати до нього доступ через спеціалізований сервіс-шлюз. Цей сервіс працює як єдина точка взаємодії між контейнерами, що потребують ліцензії, та фізичним ключем, забезпечуючи контроль черги запитів, автентифікацію клієнтів, перевірку стану пристрою та автоматичне перемикання у разі збою.

На рис. 5 предаставлено реалізацію вище описаного методу із використанням трирівневої архітектури.

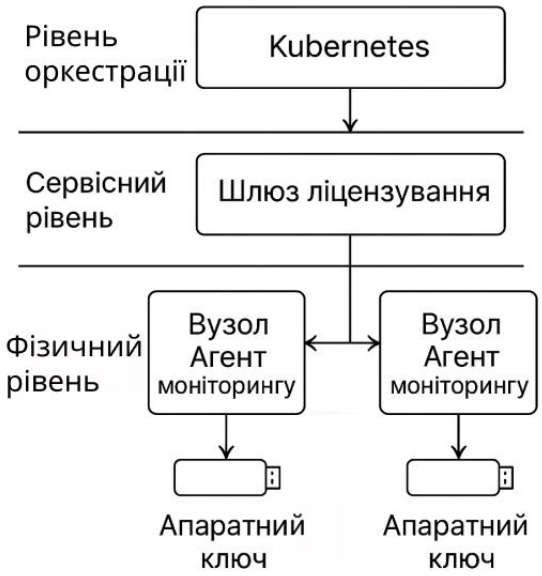

Рисунок 5. Методу доступу до апаратного ключа на базі трирівневої архітектури [авторська розробка]

Перший рівень – фізичний, до складу якого входять вузли з під'єднаними апаратними ключами. На цьому рівні реалізуються функції апаратного моніторингу, виявлення несправностей та відновлення роботи USB-портів. Для цього використовується агент спостереження, який контролює стан ключів, виконує періодичне опитування, фіксує підключення та відключення, а також передає метрики у систему моніторингу.

Другий рівень – сервісний, який реалізує програмну логіку взаємодії контейнерів з апаратним ключем. У межах цього рівня створюється окремий контейнер-шлюз (License Gateway), який працює у режимі active-active або active-passive. Шлюз взаємодіє з фізичним ключем за допомогою драйвера або SDK від постачальника (наприклад, CodeMeter, Sentinel або FlexNet) і надає інтерфейс REST/gRPC для інших мікросервісів. У разі відмови основного вузла система автоматично обирає новий активний програмий вузол за допомогою механізму leader election, що мінімізує час простою.

Третій рівень – оркестраційний, який забезпечує керування контейнерами, моніторинг стану системи, автоматичне масштабування та балансування навантаження. Для реалізації цього рівня використовується ПЗ Kubernetes, що дозволяє розподіляти сервіси між вузлами, проводити оновлення без простоїв (rolling updates), а також реалізовувати політики доступності за допомогою таких інструментів, як PodDisruptionBudget, node affinity/anti-affinity та topology spread constraints.

Ключовим елементом методу є впровадження механізму динамічного виявлення та перепідключення ключів, який у разі фізичного від'єднання пристрою або втрати зв'язку ініціює повторне підключення без потреби зупинки контейнерів. Це досягається завдяки спеціальному драйверу-посереднику, який взаємодіє із системними службами операційної системи та перехоплює події USB.

Ще одним важливим компонентом є система моніторингу та виявлення помилок у системі, що інтегрується з Prometheus або Grafana. Вона відстежує метрики доступності ключа, затримку запитів, кількість активних ліцензій, частоту збоїв і тривалість їхнього усунення. Зібрані дані дозволяють оцінювати фактичний рівень доступності сервісу (SLA, SLO) та своєчасно реагувати на потенційні відмови [125].

У методі доступу до апаратного ключа на базі трирівневої архітектури також реалізується механізм резервного доступу до нього. Це стає можливим завдяки створенню двох вузлів – основного і резервного, між якими налаштовано реплікацію метаданих ліцензій і синхронізацію станів. У разі відмови основного вузла система автоматично переведе навантаження на резервний за допомогою протоколів Keepalived або VRRP, що дозволяє зберегти безперервність обслуговування клієнтів.

Для підвищення безпеки метод передбачає використання взаємної автентифікації (mTLS) між контейнерами та шлюзом, контролю доступу (RBAC) на рівні кластеру та керування секретами через системи Vault або KMS. Таким чином забезпечується захист як переданих даних, так і апаратних ресурсів від несанкціонованого доступу (рис. 6).

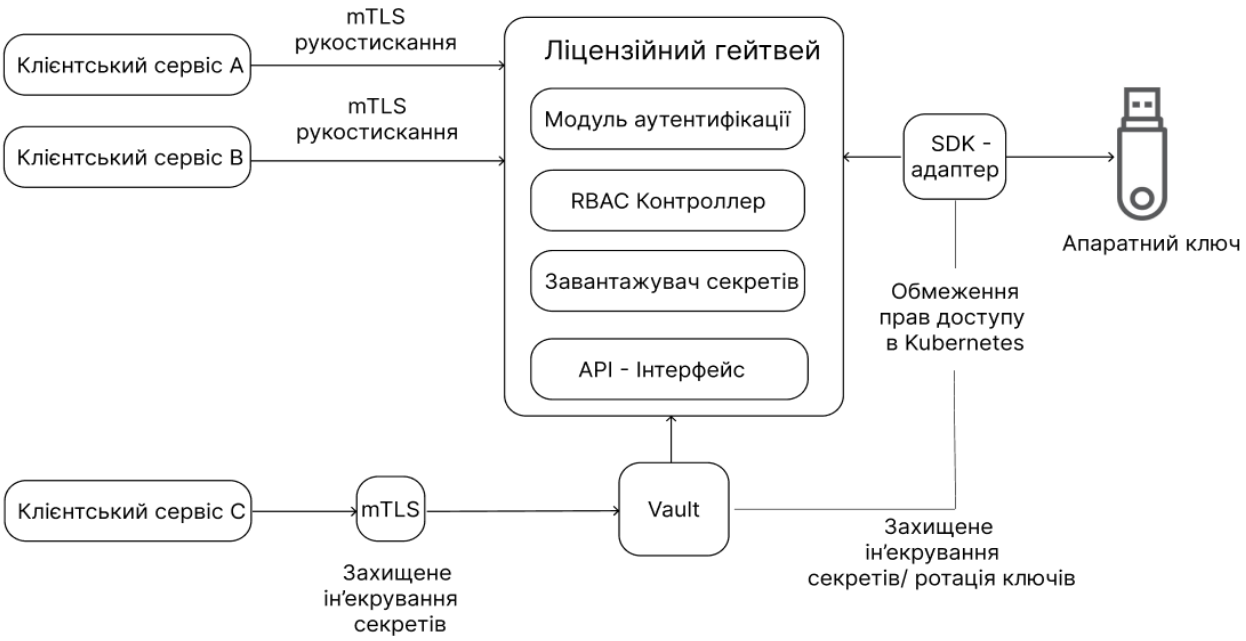

Рисунок 6. Запропонована схема захищеної комунікації між контейнерами та сервісом-шлюзом [авторська розробка]

Метод доступу до апаратного ключа на базі трирівневої архітектури поєднує переваги апаратного захисту з можливостями контейнерної оркестрації, що дозволяє досягти високого рівня відмовостійкості, гнучкості й масштабованості систем ліцензування. Його впровадження дає змогу зменшити час простою ліцензійних серверів, мінімізувати ризики втрати доступу до апаратних ключів і підвищити ефективність управління ліцензійною інфраструктурою у сучасних корпоративних середовищах.

5.1.5 Апаратна частина ключа в контейнеризованому середовищі

Проєктування апаратної частини апаратного ключа є одним із ключових етапів створення надійної системи ліцензування, оскільки саме на цьому рівні формується фізичний носій криптографічної інформації та механізм її безпечної обробки. Апаратний ключ виконує функції зберігання секретних параметрів, виконання криптографічних операцій і забезпечення автентичності, виступаючи центральною ланкою між програмним забезпеченням та захищеною ліцензійною логікою.

Апаратною основою ключа доцільно обрати мікрокомп'ютер Raspberry Pi 5 (рис. 7), який поєднує високу обчислювальну продуктивність, доступність та широкий набір інтерфейсів, необхідних для реалізації кастомного USB-пристрою.

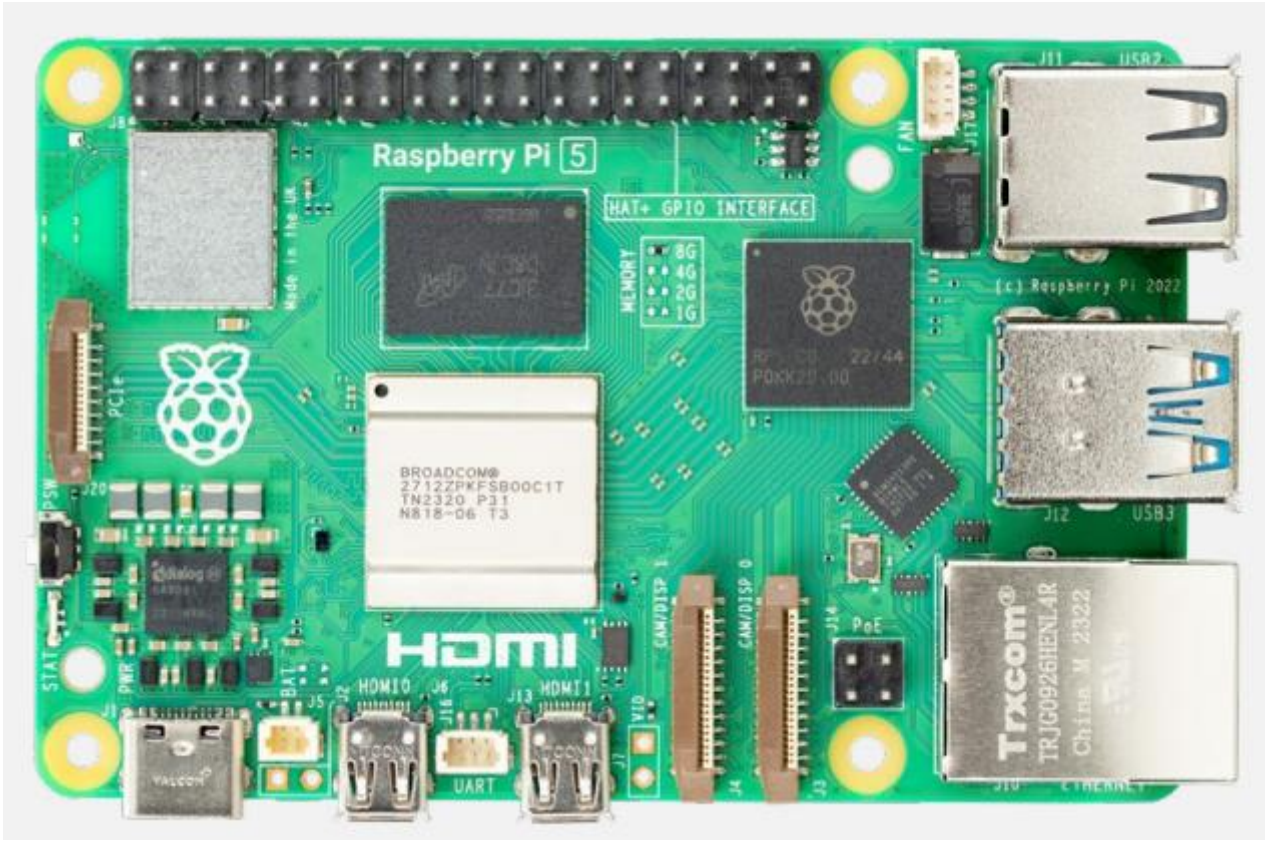

Рисунок 7. Мікрокомп'ютер Raspberry Pi 5 [126]

Пристрій оснащений портами USB та Ethernet, що розширює можливості його інтеграції в інфраструктуру. Порт USB використовується для підключення GPS-модуля, який забезпечує визначення координат у процесі роботи, тоді як інтерфейс Ethernet слугує для підключення апаратного ключа до серверної мережі. Це дозволяє програмному забезпеченню, встановленому на мікроконт-ролері, отримувати власну IP-адресу та забезпечувати стабільну взаємодію з іншими компонентами системи.

Для розміщення апаратного ключа на основі Raspberry Pi 5 із підключеним GPS-модулем NEO-8M та інтерфейсом Ethernet використовується спеціалізований промисловий корпус GEEKOM IT 13 Mini PC (рис 8).

Рисунок 8. Промисловий корпус GEEKOM IT 13 Mini PC для оболонки апаратного ключа [127]

Такий корпус забезпечує захист пристрою від механічних впливів, електромагнітних завад та несанкціонованого фізичного доступу, що є критично важливим у контексті розробки засобів апаратної безпеки. Алюмінієві моделі корпусів також виконують функцію пасивного радіатора, відводячи тепло від високопродуктивного процесора Raspberry Pi 5 та забезпечуючи стабільність його роботи під час виконання криптографічних операцій.

Корпус повинен мати достатній внутрішній простір для розміщення самої плати Raspberry Pi 5, GPS-модуля NEO-8M та необхідних комунікаційних інтерфейсів. Наявність зовнішніх отворів під роз’єми Ethernet і USB забезпечує можливість прямої інтеграції апаратного ключа до серверної інфраструктури та підключення GPS-антени.

З огляду на те, що робота GPS-модуля потребує якісного прийому супутникового сигналу, конструкція корпусу передбачає використання зовнішньої SMA-антени або встановлення внутрішньої керамічної антени у разі

застосування пластикового корпусу з низьким рівнем екранування. Розміри такі: ширина – 12см, довжина – 11см та висота – 5см, та вага 0,652 кг.

Особливу увагу приділено теплотехнічним та інженерним характеристикам корпусу. Raspberry Pi 5 має високий тепловий пакет, тому корпус має забезпечувати можливість пасивного або комбінованого (пасивно-активного) охолодження. Алюмінієві промислові корпуси, такі як GEEKOM IT 13 Mini PC, дозволяють знизити температуру процесора на 10–20 °C, підвищуючи надійність роботи пристрою. У комплексі такі корпуси забезпечують оптимальні умови для стабільної роботи Ethernet-інтерфейсу, GPS-модуля та криптографічних алгоритмів, зберігаючи при цьому компактність і захищеність апаратного ключа в умовах реальної експлуатації [128].

Для підвищення рівня безпеки до структури апаратного ключа інтегрується GPS-модуль (рис. 9), який забезпечує постійний контроль фактичного місцезнаходження пристрою.

Рисунок 9. GPS модуль NEO-8M [129]

Як навігаційний компонент доцільно використати модуль NEO-8M, що характеризується високою точністю визначення координат та швидким отриманням сигналу від супутникових систем.

Застосування GPS-модуля дає змогу реалізувати механізми географічного прив’язування криптографічних операцій, виявляти несанкціоноване переміщення ключа та автоматично активувати захисні режими у разі виходу за визначений

геоперіметр, що суттєво підвищує стійкість системи до фізичних атак. Хоча точність визначення координат модуля становить до двох метрів, з практичних міркувань програмної реалізації доцільно збільшити припустиму похибку до п’ятдесяти метрів. Такий підхід запобігає помилковим спрацюванням захисних механізмів у разі тимчасових збоїв супутникового сигналу та гарантує безперебійну роботу програмного забезпечення навіть за умов нестабільного GPS-покриття.

Структурна модель апаратного ключа (рис. 10) являє собою апаратно-програмний комплекс, що поєднує кілька функціональних модулів, інтегрованих у єдиний пристрій. Основою конструкції є міні-комп’ютер GEEKOM IT13 Mini PC, корпус якого забезпечує достатній внутрішній простір для розміщення плати Raspberry Pi 5, GPS-модуля NEO-8M, додаткових сенсорів та допоміжних комунікаційних інтерфейсів. Завдяки компактності та продуманій конструкції корпус дозволяє розмістити апаратні компоненти таким чином, щоб забезпечити захист від фізичного доступу, стабільне охолодження та можливість виведення зовнішніх інтерфейсів.

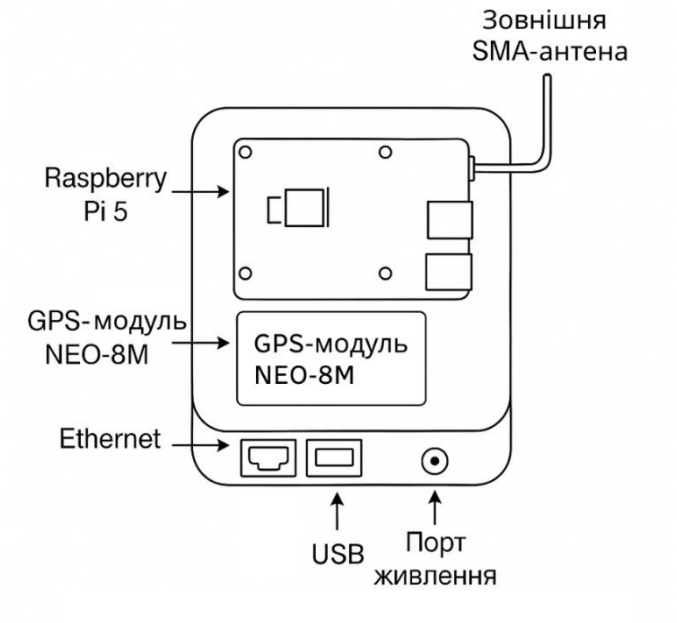

Рисунок 10. Структурна модель апаратного ключа [авторська розробка]

Розглянемо будову окреми модулів апаратного ключа.

У конструкцію апаратного ключа входить міні-комп'ютер Raspberry Pi 5, який виконує основну обчислювальну логіку: обробку криптографічних операцій, виконання ліцензійних алгоритмів, взаємодію з сервером через Ethernet та керування периферійними модулями. Плата під'єднується до внутрішнього живлення корпусу та має прямий доступ до мережевого інтерфейсу, що дозволяє мікрокомп'ютеру отримувати статичну або динамічну IP-адресу та працювати в клієнтсько-серверній інфраструктурі.

Другим ключовим елементом моделі є GPS-модуль NEO-8M, який відповідає за визначення географічних координат і контроль місцезнаходження апаратного ключа. Він з'єднується з Raspberry Pi 5 через USB або UART та може використовувати як внутрішню керамічну антену, так і зовнішню SMA-антену, використання якої забезпечує стабільний прийом сигналу всередині приміщень або в умовах значного екранування. GPS-модуль виконує функції захисту, включно з визначенням аномального переміщення, ініціюванням захисного режиму при зміні координат та виявленням втрати супутникового сигналу.

Корпус GEEKOM IT13 Mini PC використовується як зовнішній захисний контейнер, який поєднує компактність, міцність і достатній простір для встановлення обладнання. Він оснащений вентиляційними прорізами, монтажними отворами та можливістю фіксації додаткових компонентів. На корпус виводяться ключові інтерфейси:

– Ethernet – для підключення апаратного ключа до серверної мережі;

– USB – для інтеграції GPS-модуля та можливого обслуговування пристрою;

– SMA-роз'єм – для зовнішньої GPS-антени;

– порт живлення – для стабільної роботи усіх внутрішніх модулів.

Структурна модель також передбачає наявність внутрішньої схеми моніторингу, яка забезпечує контроль температури, живлення, працездатності модулів та стану GPS-сигналу.

Усі події передаються до серверної інфраструктури, що дозволяє системі ліцензування реагувати на будь-які відхилення у роботі апаратного ключа.

Захист апаратної частини реалізується шляхом використання комплексу інженерно-технічних та програмно-криптографічних заходів. Корпус апаратного ключа виготовляється з антивандальних матеріалів, а кристал мікросхеми додатково герметизується компаундом, що унеможливлює доступ до внутрішніх елементів та суттєво ускладнює фізичний реверс-інжиніринг.

Окрім фізичного захисту, критично важливою є безпека прошивки мікроконтролера. Для цього застосовуються такі механізми:

- блокування інтерфейсів відлагодження (SWD/JTAG) після встановлення прошивки, що унеможливлює її пряме зчитування;
- шифрування прошивки перед записом у пам’ять мікроконтролера, що робить недоступним її аналіз навіть у разі фізичного доступу до пам'яті;
- використання контрольних сум та цифрових підписів, які забезпечують верифікацію цілісності прошивки під час запуску.

Для запобігання витоку критичних даних і ключової логіки, у мікросхемі реалізовано механізм самознищення інформації при спробі несанкціонованого доступу. Такий механізм активується у випадках:

- виявлення доступу через заблокований SWD/JTAG інтерфейс;
- спроби зчитування пам’яті за допомогою сторонніх програматорів;
- аномальної поведінки живлення або тактової частоти, що може свідчити про використання атак типу fault-injection;
- багаторазового введення некоректного адміністративного пароля.

У разі активації механізму захисту мікроконтролер здійснює безповоротне очищення пам’яті, включно з криптографічними ключами, частиною логіки обчислень та службовими даними. Це робить неможливим відтворення алгоритмів, що зберігалися на апаратному ключі, та повністю нівелює спроби реверс-інжинірингу або фізичного копіювання пристрою. Використання таких багаторівневих заходів захисту забезпечує високий рівень стійкості апаратного ключа до фізичного злому, аналізу прошивки та крадіжки інтелектуальної власності.

Для запобігання фізичній підміні апаратного ключа застосовуються програмні механізми захисту. У випадку, коли ключ виконує лише бінарне повернення результату (true/false), що інтерпретується як авторизований або неавторизований стан, його функціональність може бути відносно легко емульована сторонніми засобами. Натомість розміщення на апаратному ключі критично важливих обчислювальних процедур, необхідних для коректного функціонування програмного забезпечення, робить підміну фактично неможливою. Відтворення таких алгоритмів вимагало б повного дублювання апаратної логіки, що є технічно вкрай складним та економічно недоцільним. У разі спроби фізичного втручання працює tamper-система, яка блокує пристрій або стирає секретні дані. Контролер може виконувати самотестування, перевірку CRC, контроль напруги та температури, що гарантує стабільну роботу навіть в умовах коливань живлення або температурних перепадів.

Окремим типом загроз є фізичне проникнення зловмисника до дата-центру з метою викрадення апаратного ключа або флеш-модуля, що містить критично важливі дані та логіку шифрування. У такому випадку система захисту переходить у режим підвищеної безпеки та активує низку захисних механізмів:

По-перше, апаратний ключ оснащений вбудованим GPS-модулем який постійно контролює своє місцезнаходження. У разі зміни координат, що не відповідає дозволеним зонам експлуатації (геоперіметру), або раптового зникнення живлення у захищеному середовищі, пристрій автоматично переходить у режим порушення безпеки. У цьому стані ключ змінює внутрішні параметри роботи та формує альтернативну криптографічну послідовність, відмінну від тієї, що використовувалася у штатному режимі. Внаслідок цього всі подальші операції шифрування та дешифрування стають некоректними для легітимної системи, що унеможливлює відновлення первинної логіки обчислень навіть при фізичному доступі до пристрою.

По-друге, програмна інфраструктура, що взаємодіє з ключем, здійснює постійний моніторинг його стану. При втраті зв’язку з датчиком місцезнаходження або при фіксації аномальних координат система генерує

аварійні сповіщення для адміністраторів. Такі сповіщення надходять у реальному часі через системи моніторингу та журналювання (наприклад, Prometheus Alertmanager або SIEM-платформи), що дозволяє оперативно виявити факт викрадення.

По-третє, у разі підтвердження викрадення апаратного ключа спеціальний модуль безпеки на сервері ініціює ревокацію криптографічних матеріалів, що були пов’язані з цим ключем. Це включає генерацію нових ключів шифрування, перезапис конфігурації системи та блокування будь-яких операцій, пов’язаних із скомпрометованим пристроєм.

Таким чином, навіть у випадку фізичного доступу до дата-центру та викрадення флеш-модуля, зловмисник не отримує можливості зламати систему або відновити логіку її роботи. Завдяки поєднанню GPS-контролю, автоматичного переходу в захисний режим, механізмів алертингу та криптографічної ревокації забезпечується високий рівень стійкості до фізичних атак та захист від компрометації інтелектуальної власності.

У представленому методі особливу увагу приділено забезпеченню сумісності апаратного ключа з контейнеризованими середовищами. Апаратний ключ під’єднується до серверної інфраструктури через Ethernet-інтерфейс, отримуючи статичну або динамічну IP-адресу, що дозволяє використовувати його як повноцінний мережевий пристрій у кластері.

Після отримання IP-адреси ключ стає доступним для сервісів Kubernetes через внутрішню мережеву взаємодію, що значно підвищує його гнучкість і зменшує залежність від фізичного розміщення. Для інтеграції апаратного ключа у Kubernetes-кластер використовується механізм Device Plugin, який автоматично позначає вузли, що мають доступ до ключа, та направляє відповідні поди саме на ці вузли. Такий підхід забезпечує гарантований доступ сервісів до апаратного ключа незалежно від того, на якому фізичному обладнанні вони розгорнуті. Крім того, апаратний ключ може працювати як через локальний USB, так і через USB-over-IP, що дозволяє розміщувати пристрій окремо від вузлів кластера без втрати функціональності (рис. 11).

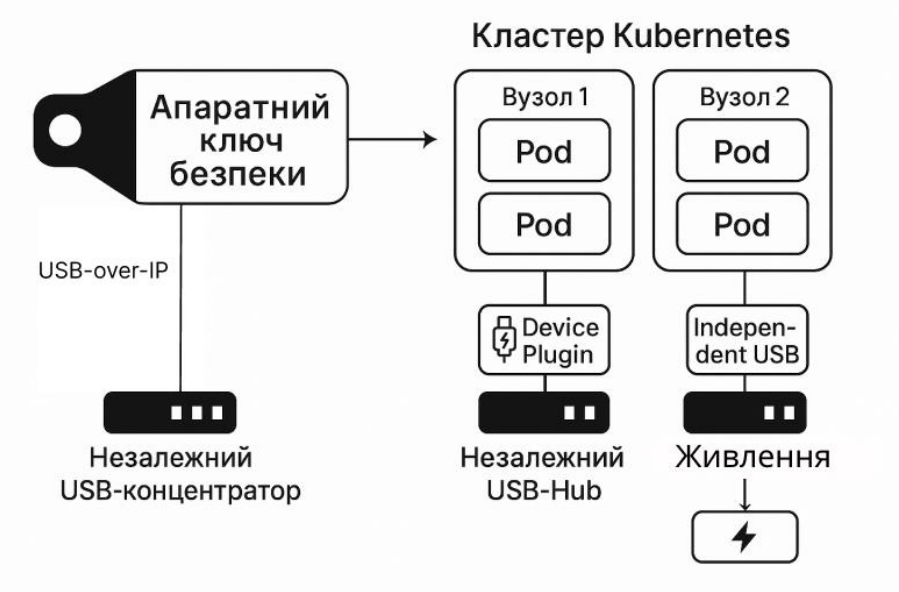

Рисунок 11. Інтеграція апаратного ключа з Kubernetes-кластерами засобами USB / USB-over-IP та механізм Device-Plugin [авторська розробка]

Для підвищення відмовостійкості інфраструктура доповнюється незалежними USB-хабами, резервованим живленням та можливістю гарячої заміни апаратного ключа без зупинки сервісів. Це забезпечує стабільну роботу системи навіть за умови часткових збоїв обладнання.

Розглянута апаратна архітектура на базі Raspberry Pi 5 забезпечує високу гнучкість, захищеність та сумісність із сучасними обчислювальними середовищами, дозволяючи реалізувати всі необхідні функції ліцензійного апаратного ключа та відповідати вимогам безпеки, автономності й надійності.

5.1.6 Програмна реалізація доступу до апаратного ключа

Розглянемо програмну модель доступу до апаратного ключа, що представлено на рис. 12, яка покликана ізолювати фізичну специфіку апаратного ключа від прикладних сервісів та забезпечити керований, спостережуваний і відмовостійкий доступ до ліцензійного механізму.

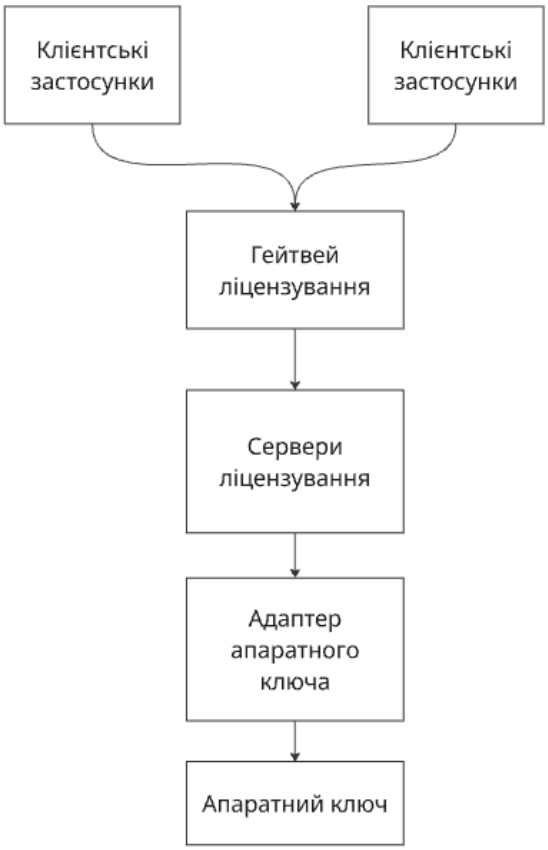

Рисунок 12. Програмна модель доступу до апаратного ключа [авторська розробка]

Основною концепцією є впровадження проміжного шару сервісу-шлюзу ліцензування, який інкапсулює взаємодію з вендорським SDK, реалізує політики керування чергою запитів, механізми ідемпотентності операцій та аудит доступу, а також надає стандартизований інтерфейс взаємодії (REST/gRPC) для всіх споживачів у кластері. Такий підхід гарантує, що прикладні мікросервіси ніколи не звертаються до апаратного ключа напряму та залишаються незалежними від його фізичного розміщення, типу драйвера, особливостей встановлення або змін у вендорській реалізації [130].

Додатковою перевагою такого підходу є можливість забезпечення рівномірного розподілу навантаження між кількома апаратними ключами, у випадку якщо інфраструктура містить декілька вузлів, кожен з яких має власний фізичний носій ліцензії. Шлюз ліцензування може виконувати балансування запитів, автоматичне перемикання між ключами при відмовах, а також централізований моні-

торинг стану кожного пристрою. Це значно підвищує надійність роботи системи та мінімізує ризик простою сервісів у випадку збоїв на рівні окремого вузла.

Оскільки представлений метод забезпечує максимально захищене, керо-ване та масштабоване використання апаратного ключа, клієнтам, що впровад-жують подібну систему захисту свого програмного забезпечення, рекомендовано застосовувати саме таку архітектуру. Вона дозволяє досягти високої ефектив-ності використання ключа, мінімізувати вплив на бізнес-логіку при збоях апаратної частини, а також забезпечує централізоване адміністрування, швидке розширення інфраструктури та відповідність вимогам інформаційної безпеки. Логічна модель складається з трьох шарів (рис 13).

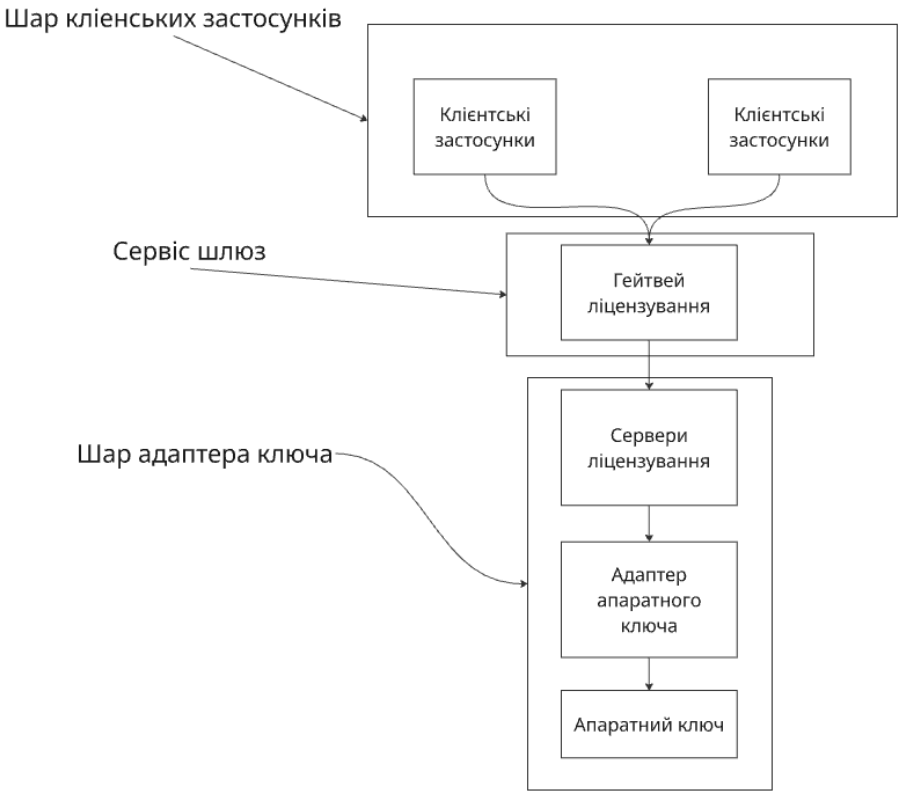

Рисунок 13. Логічна модель доступу до апаратного ключа [авторська розробка]

На периферії знаходиться шар клієнтських застосунків та мікросервісів бізнес-домену, що виконують запити на отримання або звільнення ліцензії в межах транзакції роботи користувача чи обчислювальної задачі.

У центрі розташовано сервіс-шлюз, який приймає запити, автентифікує клієнтів за допомогою взаємних сертифікатів, проводить авторизацію за ролями

та політиками, застосовує обмеження швидкості, формує чергу на доступ до ключа і гарантує коректне завершення життєвого циклу “checkout–use–return”.

У нижньому шарі функціонує адаптер до апаратного ключа, що інкапсулює вендорське SDK, керує сесіями, виконує challenge–response та проводить локальну самодіагностику стану пристрою і драйверів.

Щоб забезпечити високу доступність, сервіс-шлюз розгортається у кількох репліках, між якими реалізовано механізм вибору лідера. Лідер має ексклюзивний доступ до адаптера ключа, тоді як інші інстанси обробляють автентифікацію, черги та метрики, але перенаправляють критичні виклики до активного екземпляра через внутрішній RPC. Такий підхід дозволяє масштабувати фронт оброблення запитів і водночас уникати конкуренції за доступ до фізичного пристрою. Перемикання лідера відбувається автоматично за наявності ознак деградації (втрата heartbeat, збій SDK, зростання p95 часу `checkout()`), що мінімізує час простою.

Комунікація між компонентами будується на принципах нульової довіри. Усі виклики до шлюзу підписуються і шифруються (mTLS), сертифікати мають короткий строк дії та автоматично ротується через механізм видавця сертифікатів у кластері.

Авторизація здійснюється на основі атрибутів запиту: типу продукту, редакції, кількості одночасних місць, контексту проєкту та дозволених дій. Політики конфігуруються декларативно і зберігаються у централізованому сховищі конфігурацій, що дає змогу виконувати зміну правил без перезапуску сервісів і без впливу на працюючі сесії. Стан системи розмежовується. Джерелом істини щодо кількості доступних ліцензій виступає сам апаратний ключ або вендорський демон; будь-які проміжні кеші слугують лише для телеметрії та оптимізації з’єднань.

Для уникнення подвійних списань кожна операція `checkout()` має ідемпотентний ідентифікатор, а у разі повторного виклику в межах тайм-вікна шлюз повертає поточний стан без додаткового споживання ліцензії. У випадках втрати зв’язку реалізується механізм автоматичного відновлення сесії: при

короткочасних збоях застосовується повтор із експоненційною затримкою, при триваліших – примусове повернення “завислих” оренд після контрольного тайм-ауту з подальшим аудитом [131].

Спостережуваність закладається як невід’ємна властивість архітектури. Кожен виклик фіксується у журналах із кореляційним ідентифікатором, метрики експортуються у систему моніторингу та включають рівень успішності, затримку на різних етапах обробки, глибину черги, кількість активних сесій, частоту відмов SDK і події апаратного рівня (detach/attach, помилки шини). На основі цих показників визначаються SLI та SLO, формується error budget і налаштовуються попереджувальні сповіщення про вичерпання ліцензій, деградацію p95/p99 затримок і флапінг пристрою. Для розбору інцидентів передбачено знеособлений трейсинг запитів із мінімально необхідним набором полів.

З міркувань безпеки та керованості секрети (ключі шифрування, токени адміністрування, приватні сертифікати) зберігаються у зовнішньому сховищі секретів із журнальним доступом і політиками мінімальних привілеїв. Конфігурація під’єднання до вендорських сервісів, параметри тайм-аутів, обмеження на довжину сесії та профілі продуктивності постачаються через централізовані конфіг-джерела і підхоплюються під час виконання. Оновлення сервісів відбуваються за схемою без простоїв: для шлюзу застосовується стратегії поступового виведення з експлуатації, для адаптера – контрольована передача лідерства і від’єднання з коректним поверненням усіх активних оренд [132].

Особлива увага приділяється тестуванню стійкості. Архітектура враховує сценарії хаос-експериментів: раптове від’єднання USB, штучні затримки на мережевому шляху, збої у вендорському SDK, падіння лідера та відмови вузла. Для кожного сценарію визначено очікувану поведінку (непорушність інваріантів, гарантоване повернення ліцензії, межі допустимої деградації затримок) і порядок відновлення. Функціональні тести перевіряють коректність

бізнес-правил видачі, а навантажувальні оцінюють пікові `checkout rate` і стабільність під тривалим тиском з різними профілями запитів.

Взаємодія з прикладними командами спрощується за рахунок стабільного контракту API і офіційних SDK-обгорток, які інкапсулюють усі вимоги безпеки, генерацію ідемпотентних ключів, повтори з бюджетом і телеметрію.

Таким чином, розробники бізнес-функцій отримують прозорий сервіс ліцензування як платформенну можливість, не занурюючись у деталі роботи з фізичним пристроєм. У підсумку архітектура забезпечує керованість, відмовостійкість і масштабованість, дозволяючи експлуатувати апаратний ключ як надійний ресурс у контейнеризованому середовищі без компромісів щодо безпеки та доступності.

5.1.7 Метод підключення до апаратного ключа в кластерному середовищі

Проєктування методу доступу до апаратного ключа в кластерному середовищі є одним із ключових етапів розроблення підходу до забезпечення відмовостійкості, адже саме від нього залежить стабільність і безперервність роботи всієї системи ліцензування. Основною метою створення методу є розподіл запитів на використання апаратного ключа між кількома контейнерами або вузлами без порушення цілісності даних, уникнення одночасного доступу до одного ресурсу, а також забезпечення автоматичного відновлення зв’язку у випадку збоїв.

У кластерному середовищі, де програмні сервіси розподілені між різними вузлами, апаратний ключ залишається фізично унікальним пристроєм. Це створює певну асиметрію: програмна інфраструктура може масштабуватись майже необмежено, тоді як апаратний ресурс є обмеженим і має підтримувати лише один активний канал взаємодії в певний момент часу. Для вирішення цього протиріччя було розроблено підхід до централізованого доступу з механізмом серіалізації запитів, який реалізує логіку “один активний доступ – багато клієнтів”.

Процес уведення апаратного ключа в експлуатацію починається ще на етапі його фізичної доставки до датацентру й завершується повноцінною інтеграцією в кластер та системи моніторингу.

Розглянемо поетапне виконання методу підключення до апаратного ключа в кластерному середовищі (рис. 14).

Рисунок 14. Етапи методу підключення до апаратного ключа в кластерному середовищі [авторська розробка]

Етап 1. Доставка та первинна перевірка. Апаратний ключ доставляється до датацентру в опломбованій упаковці. Адміністратор розпаковує пристрій, виконує візуальну перевірку цілісності корпусу, пломб та відсутності механічних пошкоджень. За потреби результати перевірки фіксуються в журналі приймання обладнання.

Етап 2. Підключення до інфраструктури живлення та мережі. Пристрій під’єднується до джерела живлення згідно з технічною документацією. Після цього апаратний ключ підключається до мережевої інфраструктури датацентру через порт Ethernet відповідного серверного сегмента (наприклад, захищеної VLAN з доступом лише до сервісів ліцензування).

Етап 3. Початкова ініціалізація та введення адміністративного пароля. Після увімкнення живлення апаратний ключ переходить у початковий режим ініціалізації, у якому вимагає введення адміністративного пароля. Адміністратор за допомогою консольного доступу (SSH, серійний порт або веб-інтерфейс керування) задає або змінює початковий пароль на унікальний, що відповідає політикам інформаційної безпеки організації. На цьому етапі також можуть створюватися додаткові облікові записи з обмеженими правами.

Етап 4. Налаштування мережевих параметрів та отримання IP-адреси. Далі виконується конфігурація мережі: пристрій може отримувати IP-адресу через DHCP або мати статичну адресу, зарезервовану в межах виділеного сегмента. Після присвоєння IP-адреси проводиться тестовий обмін пакетами (ping, перевірка доступу до службового порту) для підтвердження коректності мережевих налаштувань.

Етап 5. Реєстрація апаратного ключа в сервісі-шлюзі ліцензування. Отримана IP-адреса, ідентифікатор пристрою, серійний номер та інші атрибути вносяться в конфігурацію сервісу-шлюзу ліцензування. На цьому етапі ключ реєструється як окремий вузол у внутрішньому реєстрі ресурсів (наприклад, у конфігураційній базі або сховищі параметрів кластера), після чого шлюз отримує можливість здійснювати до нього контрольований доступ.

Етап 6. Інтеграція із системами моніторингу та виявлення помилок. Наступним кроком є підключення апаратного ключа до систем спостереження (моніторинг здоров’я сервісів, доступності, часу відповіді), а також налаштування правил сповіщень. Адміністратор конфігурує систему виявлення помилок на випадок втрати зв’язку з пристроєм, зміни його координат (згідно з GPS-модулем), повторних помилок автентифікації, некоректної роботи ліцензійного SDK тощо. Це забезпечує оперативне виявлення інцидентів безпеки та збоїв.

Етап 7. Інтеграція з кластером контейнеризації. На завершальному етапі апаратний ключ логічно “прив’язується” до кластерного середовища. У конфігурації Kubernetes або іншої системи оркестрації налаштовується сервіс

ліцензування (gateway), який знає про IP-адресу ключа, політики доступу до нього, таймаути та параметри повторних спроб. Для вузлів, на яких розгортається сервіс-шлюз, можуть бути задані спеціальні мітки (labels), node affinity або tolerations, якщо ключ фізично прив'язаний до конкретного сегмента мережі або вузла.

На цьому етапі програмне забезпечення може повноцінно використовувати можливості апаратного ключа у своїй роботі. Для цього мікросервіси надсилають запити на IP-адресу, яку було встановлено під час початкової конфігурації пристрою. Оскільки взаємодія здійснюється в межах локальної мережі датацентру, затримки при обміні даними є мінімальними, що забезпечує високу швидкість обробки операцій ліцензування. Проте, така конфігурація має суттєве обмеження: якщо програмний компонент, який взаємодіє з ключем, розгорнутий на одному фізичному сервері з пристроєм, то його відмова призведе до втрати доступності всієї системи ліцензування. Це суперечить вимогам до відмовостійкості, тому наступним етапом є розгортання повноцінного кластерного середовища.

Архітектурно метод доступу до апаратного ключа ґрунтується на концепції лідер–репліка, згідно з якою лише один вузол (лідер) у будь-який момент часу має ексклюзивний доступ до апаратного ключа (рис. 15). Усі інші вузли кластера перебувають у пасивному режимі та виконують роль реплік, готових перебрати функціональність лідера у разі його недоступності. Такий підхід розв'язує проблему єдиного фізичного ресурсу – апаратного ключа – забезпечуючи водночас масштабованість і високу доступність програмної інфраструктури.

Визначення активного вузла здійснюється за допомогою механізму leader election, що реалізується засобами Kubernetes через використання `ConfigMap`, `Lease API` або спеціалізованих контролерів.

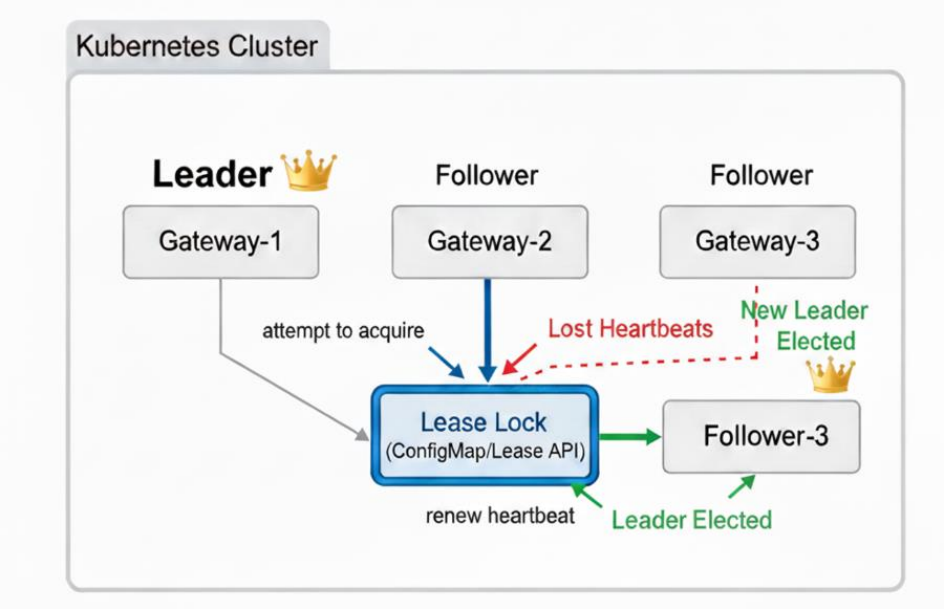

Рисунок 15. Реалізація концепції лідер – репліка для доступу до апаратного ключа [авторська розробка]

Після обрання лідера всі запити до апаратного ключа спрямовуються виключно до цього вузла через внутрішній сервіс-шлюз, що відповідає за серіалізацію запитів, контроль доступу та взаємодію з вендорським SDK. У разі збою лідера Kubernetes автоматично ініціює процедуру переобрання, і новий вузол миттєво перебирає роль активного. Це забезпечує безперервність роботи та мінімізує ризик простою ліцензійної системи навіть за умов часткових відмов інфраструктури.

Коли користувач або інший мікросервіс надсилає запит на отримання ліцензії, запит надходить у чергу шлюзу доступу до ключа. Кожному запиту присвоюється унікальний ідентифікатор та часовий штамп. Алгоритм перевіряє стан ключа, доступність лідера та відсутність активних транзакцій. Якщо ключ вільний, шлюз ініціює операцію `checkout()` через SDK виробника апаратного ключа, отримує токен доступу та передає його запитувачу. Токен має обмежений термін дії, після чого виконується автоматичне `return()` або подовження сесії при продовженні роботи користувача

Для уникнення колізій у черзі запитів застосовується стратегія ідемпотентності (рис. 16): повторні запити з тим самим ідентифікатором не породжують нових сесій. Це виключає дублювання ліцензій і гарантує, що навіть при мережевих затримках або повторних викликах клієнт отримає один і той самий результат.

Рисунок 16. Граф-схема досягнення ідемпотентності запитів [авторська розробка]

У разі виникнення збоїв передбачено кілька сценаріїв відновлення. Якщо втрачається з’єднання між ліцензійним шлюзом і апаратним ключем, активна сесія переходить у режим очікування з періодичними спробами

перепідключення. Якщо впродовж певного тайм-ауту зв’язок не відновлюється, система ініціює процедуру failover: призначається новий лідер, а попередній вузол від’єднується від ключа через механізм fencing, що унеможливлює одночасний доступ із двох джерел. Новий лідер відновлює роботу сервісу без втрати даних, виконуючи повторну автентифікацію та синхронізацію стану з попередніми сесіями.

Додатково метод реалізує автоматичне балансування навантаження на рівні клієнтів (рис. 17).

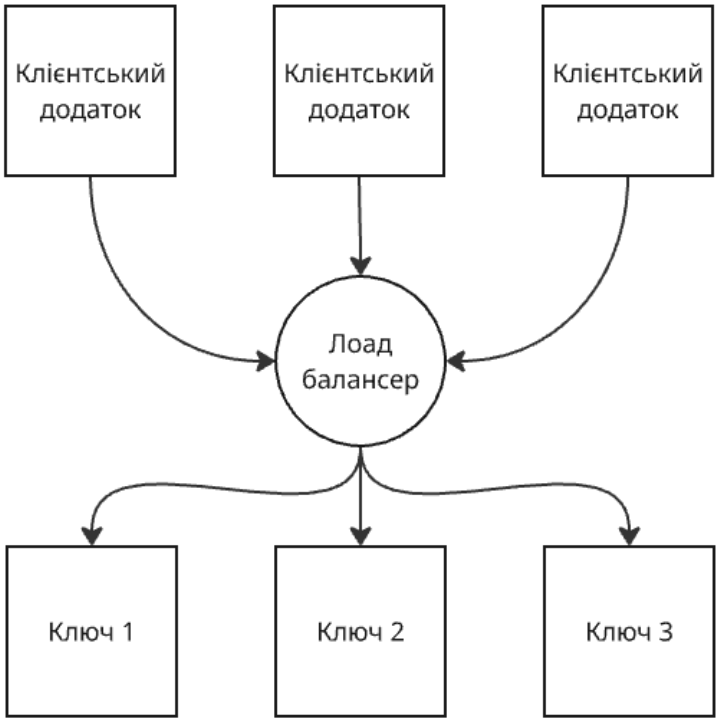

Рисунок 17. Схема балансування навантаження на рівні клієнтів [авторська розробка]

Якщо система має кілька незалежних ключів або мережевих шлюзів, запити рівномірно розподіляються між ними з урахуванням поточної кількості активних сесій і часу відгуку. Це дозволяє оптимізувати використання ресурсів і уникнути перевантаження одного вузла.

Важливою складовою методу є захист від несанкціонованого доступу. Кожна транзакція супроводжується обов’язковою автентифікацією клієнта через

протокол mTLS, а обмін даними з ключем виконується за зашифрованим каналом. Усі дії користувачів і системних процесів журналюються, що забезпечує повний аудит використання ліцензій і допомагає виявляти аномальну активність.

Таким чином, запропонований метод забезпечує:

- контрольований одночасний доступ до апаратного ключа без порушення цілісності даних;
- автоматичне відновлення після збоїв або втрати вузла-лідера;
- динамічне балансування навантаження в розподіленому середовищі;
- криптографічно захищену комунікацію;
- ідемпотентність операцій та аудит усіх транзакцій.

Поєднання цих характеристик дозволяє ефективно інтегрувати апаратний ключ у кластерну контейнеризовану архітектуру, забезпечуючи безперервність роботи системи ліцензування та підвищуючи її надійність і масштабованість.

Розглянутий метод дозволяє значно підвищити рівень відмовостійкості, мінімізувати час простою та забезпечити стабільну роботу ліцензійних серверів навіть у випадку часткових збоїв інфраструктури.

Крім того, створена модель є універсальною та може бути адаптована для різних типів апаратних ключів і корпоративних середовищ, що робить її практично цінною для впровадження в сучасних системах ліцензування.