УДК 681.12

Я. М. Ясько
А. О. Азарова

Розроблення методу забезпечення відмовостійкості серверів ліцензування

Вінницький національний технічний університет

Анотація

У роботі досліджено методи забезпечення відмовостійкості та доступності серверів ліцензування в контейнеризованому середовищі шляхом використання апаратних ключів. Розглянуто архітектурні принципи, механізми розподілу навантаження, серіалізації доступу до унікального апаратного ресурсу та засоби захисту від фізичних і програмних атак. Запропоноване рішення поєднує контейнеризацію Kubernetes, мережеві протоколи безпечної взаємодії та апаратне криптографічне ядро, що дає змогу підвищити надійність, безпеку та стабільність системи ліцензування. Отримані результати можуть бути використані у критично важливих інформаційних системах, де необхідно гарантувати цілісність та доступність ліцензійних даних.

Ключові слова: апаратний ключ, відмовостійкість, контейнеризація, Kubernetes, криптографія, високодоступні системи, безпека даних, ліцензування.

Ensuring visibility and availability of license servers with hardware keys in a containerized environment (Docker/Kubernetes)

Abstract

This work investigates methods for ensuring the fault tolerance and availability of licensing servers in a containerized environment through the use of hardware security keys. The study examines architectural principles, request serialization mechanisms, load distribution approaches, and protection techniques against physical and software-based attacks. The proposed solution integrates Kubernetes container orchestration, secure communication protocols, and a hardware cryptographic module, enabling increased reliability, security, and stability of the licensing infrastructure. The findings can be applied in critical information systems where data integrity and availability of licensing services are essential.

Keywords: hardware key, fault tolerance, containerization, Kubernetes, cryptography, high availability, data security, licensing systems.

Розроблення методу 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 запропонований 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 розробленого методу полягає у поєднанні програмно-апаратного резервування із засобами оркестрації контейнерів. Основна ідея – 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. 1 розглянуто реалізацію запропонованого у МКР методу на базі використання трирівневої архітектури.

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 спос стереження, 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

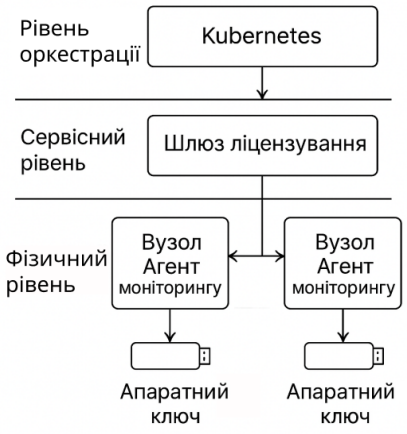

Рисунок 1 – Методу доступу до апаратного ключа
на базі трирівневої архітектури

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.

У межах розробленого методу також реалізується механізм резервного доступу до апаратного ключа. Для цього створюються два вузли – основний і резервний, 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. 2).

Запропонований метод 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.

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.

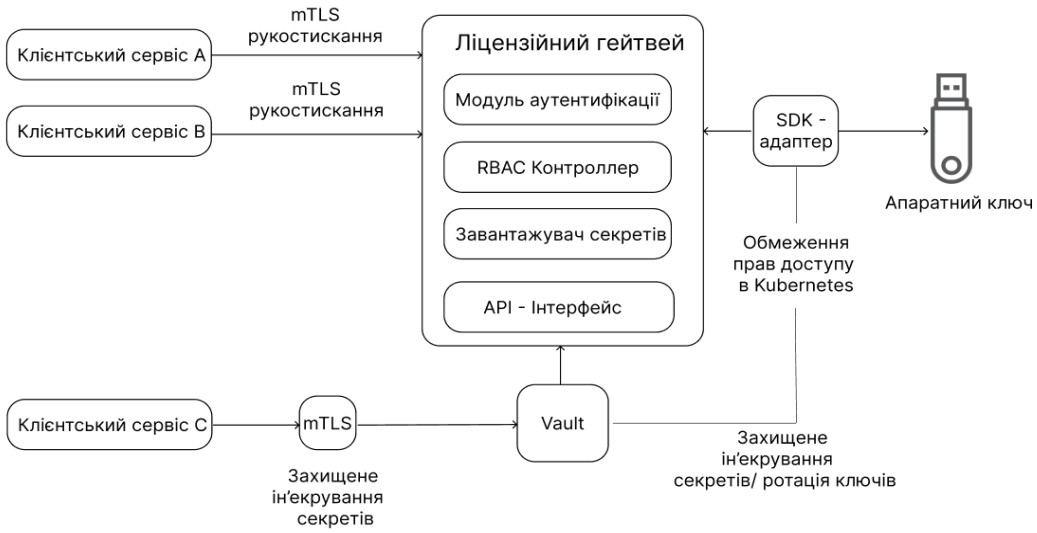

Рисунок 2 – Запропонована схема захищеної комунікації між контейнерами та сервісом-шлюзом

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

Пристрій оснащено 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 мікроконтролері, отримувати власну IP-адресу та забезпечувати стабільну взаємодію з іншими компонентами системи.

Структурна модель апаратного ключа (рис. 3) 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.

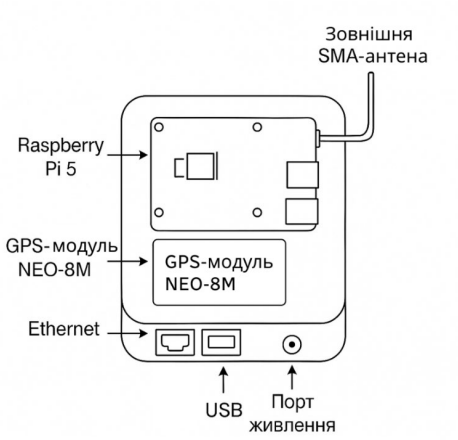

Рисунок 6 – Структурна модель апаратного ключа

Розглянемо будову окремих 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.

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 (алерти) 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 розробці 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 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. Розроблена 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.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Martin Kleppmann, розробка додатків із інтенсивним використанням даних: великі ідеї, що лежать в основі надійних, масштабованих і зручних систем / Martin Kleppmann — розробка додатків, 2017. — 231 с.

2. Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud. URL: https://aws.amazon.com/ru/blogs/architecture/disaster-recovery-dr-architecture-on-aws-part-i-strategies-for-recovery-in-the-cloud/ (Дата звернення 13.12.2025).

Яcько Яків Михайлович — студент групи 2КІ-24м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: james.yasko.2002@gmail.com

Азарова Анжеліка Олексіївна — професор кафедри обчислювальної техніки, Вінницький національний технічний університет.

Yasko Yakiv Mykhailovych — student of group 2KI-24m, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: **james.yasko.2002@gmail.com**

Azarova Anzhelika Oleksiivna — Professor of the Department of Computer Engineering, Vinnytsia National Technical University.