УДК 004.42 https://doi.org/10.32515/2664-262X.2026.13(44).73-82

А. О. Нічепорук, доц., канд. техн. наук., **В.Ю. Гурський**
Хмельницький національний університет, м. Хмельницький, Україна
e-mail: andrey.nicheporuk@gmail.com, nighttima@gmail.com

Архітектура віддаленої верифікації контейнеризованих застосунків Kubernetes для периферійних пристроїв на базі Raspberry Pi

Запропонована архітектура спрямована на віддалену верифікацію контейнеризованих застосунків (подів) Kubernetes кластеру у середовищах периферійних обчислень, де обмежені ресурси, доступність фізичних пристроїв та підвищені ризики компрометації вимагають ретельного контролю цілісності виконання. Пропонована архітектура інтегрується в Kubernetes без змін ядра, використовуючи стандартні механізми розширюваності, такі як CRD та користувацькі контролери. Завдяки цьому функції верифікації реалізуються як складова частина логіки керування кластером і виконуються в межах його звичайних процесів.

верифікація, периферійні пристрої, TPM-модуль, Raspberry Pi, архітектура вимірювання цілісності

Постановка проблеми. Швидкий розвиток концепції периферійних обчислень та масштабне впровадження edge computing архітектур у промислових, транспортних і IoT-системах зумовлюють зростаючу потребу у забезпеченні довіри до розподіленої інфраструктури. Вона розміщується географічно близько до джерел даних, часто у фізично незахищених локаціях, з обмеженим контролем з боку адміністраторів. Використання Kubernetes як де-факто стандарту оркестрації контейнеризованих застосунків поширюється за межі традиційних центрів обробки даних на edge-пристрої з обмеженими ресурсами, зокрема на одноплатні комп'ютери Raspberry Pi, що створює нові виклики у сфері кібербезпеки [1, 2].

Відсутність нативних механізмів віддаленої верифікації у Kubernetes формує критичну прогалину у забезпеченні цілісності виконуваних застосунків. Компрометація периферійного вузла може призвести до витоку конфіденційних даних, маніпуляції результатами обробки інформації або використання інфраструктури для проведення подальших атак [3, 4]. Особливо гостро ця проблема проявляється у багатоорендному середовищі, де різні організації спільно використовують edge-інфраструктуру хмарного провайдера, що вимагає надійних криптографічних гарантій того, що застосунки виконуються на довірених вузлах без несанкціонованих модифікацій коду чи конфігурацій.

Існуючі фреймворки віддаленої атестації (верифікації), такі як Keylime [5], забезпечують верифікацію цілісності фізичних вузлів через апаратні модулі довіри TPM, однак оперують на рівні всієї операційної системи. Тобто рішення, які лежать в основі цих фреймворків не дозволяють розрізнити окремі контейнери (поди). Як наслідок, такий недолік, зокрема в контексті мікросервісних архітектур, де на одному вузлі може виконуватись десятки подів, призводить до відсутності можливості реагування на компроментацію конретних застосунків, що виконуються в подах кластеру.

Аналіз останніх досліджень і публікацій. Стрімке поширення контейнеризації в периферійних середовищах обумовлює критичну потребу у віддаленій верифікації

Kubernetes-кластерів, проте разом із тим відомі традиційні фреймворки, такі як Keylime, орієнтовані переважно на цілісність усього вузла і не забезпечують необхідної гранулярності на рівні окремих подів. У граничних (edge) системах ця проблема ускладнюється обмеженими ресурсами пристроїв, нестабільністю мережі та потребою в перевірці конкретних навантажень без критичного впливу на продуктивність. Сучасні дослідження зосереджені на інтеграції апаратних коренів довіри (TPM) з механізмами оркестрації, де провідні архітектури базуються на використанні підсистеми IMA та TPM 2.0 для динамічної атестації [6]. Зокрема, для ідентифікації конкретних подів пропонується модифікація ядра Linux через додавання атрибута cgroup-path до шаблону вимірювань. Розвиток цих підходів включає використання конфіденційних середовищ на базі Kata Containers [7], автоматизацію процесів через систему KubeTEE [8] та адаптацію Virtual Kubelet для низькоресурсних пристроїв класу Raspberry Pi [9, 10].

В Україні активно розвиваються суміжні напрямки граничних обчислень та інфраструктури для мобільних інтелектуальних систем, зокрема з інтеграцією edge та cloud-обчислень, балансуванням навантаження та кіберзахистом. Зокрема, запропоновано підходи до розгортання інфраструктури мобільних інтелектуальних систем на базі груп безпілотних літальних апаратів з акцентом на адаптивне управління ресурсами в гібридних edge-cloud середовищах [11]. Також досліджено інтелектуалізацію граничних обчислень Інтернету речей, де акцентовано на підвищенні ефективності обробки даних на периферії з урахуванням обмежених ресурсів та динамічних умов [12]. Ці роботи підкреслюють актуальність легких, адаптивних архітектур для edge-пристроїв, що створює передумови для подальшої інтеграції механізмів верифікації на рівні Kubernetes-подів.

Постановка завдання. Метою даного дослідження є проведення перевірки цілісності на рівні окремих подів у розподілених кластерах з обмеженими обчислювальними ресурсами, а також зменшення часу проведення процесу верифікації шляхом дослідження та проєктування архітектури віддаленої верифікації Kubernetes подів для edge-пристроїв на базі Raspberry Pi з використанням апаратного TPM-модуля Infineon OPTIGA SLB 9670.

Основні завдання дослідження: 1) проаналізувати традиційні підходи та фреймворки віддаленої верифікації і виокремити їхні переваги та недоліки в межах мікросервісних архітектур; 2) запропонувати архітектуру віддаленої верифікації Kubernetes подів для пристроїв на базі Raspberry Pi, що заснована на інтеграції апаратних модулів TPM 2.0, використанні підсистеми IMA з модифікованим шаблоном вимірювань cgroup-path, а також застосуванні декларативної моделі керування через механізми CRD та користувацьких контролерів; 3) провести експериментальне дослідження щодо часової ефективності запропонованої системи та порівняти її продуктивність із існуючим рішеннями.

Виклад основного матеріалу. Запропонована архітектура спрямована на забезпечення віддаленої верифікації Kubernetes подів у середовищі edge-обчислень, де обмежені ресурси, фізична доступність пристроїв і підвищений ризик компрометації вимагають посиленого контролю цілісності виконання. Пропонована архітектура дозволяє нативно інтегруватись у кластер Kubernetes без внесення змін до його ядра. Вона спирається на стандартні засоби розширення платформи, зокрема механізм визначення власних типів ресурсів і спеціалізовані керуючі компоненти. Завдяки цьому функції верифікації реалізуються як складова частина логіки керування кластером і виконуються в межах його звичайних процесів.

В рамках даної архітектуру розглядається Kubernetes кластер, у якому одноплатні комп’ютери Raspberry Pi, виконують роль робочих вузлів, які розміщуються

безпосередньо на периферії мережі та відповідають за виконання контейнеризованих застосунків.

Контрольні та верифікаційні компоненти архітектури розміщуються на головному вузлі Kubernetes (control plane), який може бути реалізований як окремий сервер або більш потужний edge-шлюз. Вони відповідають за координацію процесів верифікації, зберігання довірених еталонів, перевірку криптографічних доказів та прийняття рішень щодо довіреності подів. Взаємодія між головним вузлом та edge-вузлами здійснюється виключно через Kubernetes API, що забезпечує узгодженість із декларативною моделлю керування та спрощує масштабування системи.

На кожній одноплатній комп'ютерній системі Raspberry Pi здійснюється розгортання спеціального агенту, який працює з розширеними правами доступу та взаємодіє з апаратними й програмними засобами забезпечення довіри пристрою. У межах цієї роботи Raspberry Pi розглядається як периферійна платформа, що може виконувати роль верифікатора, до якого під'єднаний апаратний модуль TPM 2.0, що виступає як корінь довіри, а також підсистему контролю цілісності Linux для збору даних про виконуваний код в самих подах. Це дає змогу формувати криптографічно захищені підтвердження цілісності, які відображають фактичний стан ізольованих програмних середовищ на периферійному вузлі кластеру.

Особливість пропонованої архітектури є представлення станів вузлів кластеру, самих подів, а також модулів, що реалізують процеси верифікації у вигляді власних типів ресурсів. Такий підхід дозволяє інтегрувати механізм віддаленої верифікації в подієву модель Kubernetes і забезпечує автоматичну реакцію системи на зміни стану edge-вузлів або подів. У результаті верифікація подів стає безперервним процесом, тісно пов'язаним із життєвим циклом застосунків, що особливо важливо для edge-сценаріїв із динамічним навантаженням і частими оновленнями.

Загальну схему роботи пропонованої архітектури віддаленої верифікації kubernetes подів можна подати через наступу послідовність дій:

1. Под запускається на робочому вузлі, що представлений одноплатною комп'ютерною системою Raspberry Pi.
2. Агент на Raspberry Pi:
- читає дані з TPM;
- перевіряє цілісність коду пода;
3. Raspberry Pi надсилає доказ до керуючого вузла.
4. На керуючому вузлі:
- перевіряється доказ;
- вирішує: довірений под чи не довірений под.
5. Kubernetes кластер виконує реагування, тобто дозволяє або блокує або зупиняє под.

Таким чином, запропонована архітектура формує наскрізний ланцюг довіри від апаратної платформи Raspberry Pi до окремого Kubernetes pod'а, поєднуючи апаратні механізми безпеки, програмні засоби контролю цілісності та декларативну модель керування Kubernetes. Це робить можливим застосування віддаленої верифікації подів у розподілених інфраструктурах, що функціонують на периферії мережі (edge), і підвищує рівень довіри до виконання контейнеризованих застосунків на периферійних пристроях.

З точки зору компонентів пропонована архітектура віддаленої верифікації kubernetes подів для edge-пристроїв на базі Raspberry Pi складається із наступних компонентів (рис. 1): модуля реєстратора, обробника робочих вузлів, модулів агентів (на кожному робочому вузлі), агенту стану, контролера стану кластера, обробника

подів, об'єкту CRD, що виконує запит на атестацію, верифікатора, підсистеми зберігання білого списку, підсистеми моніторингу подів.

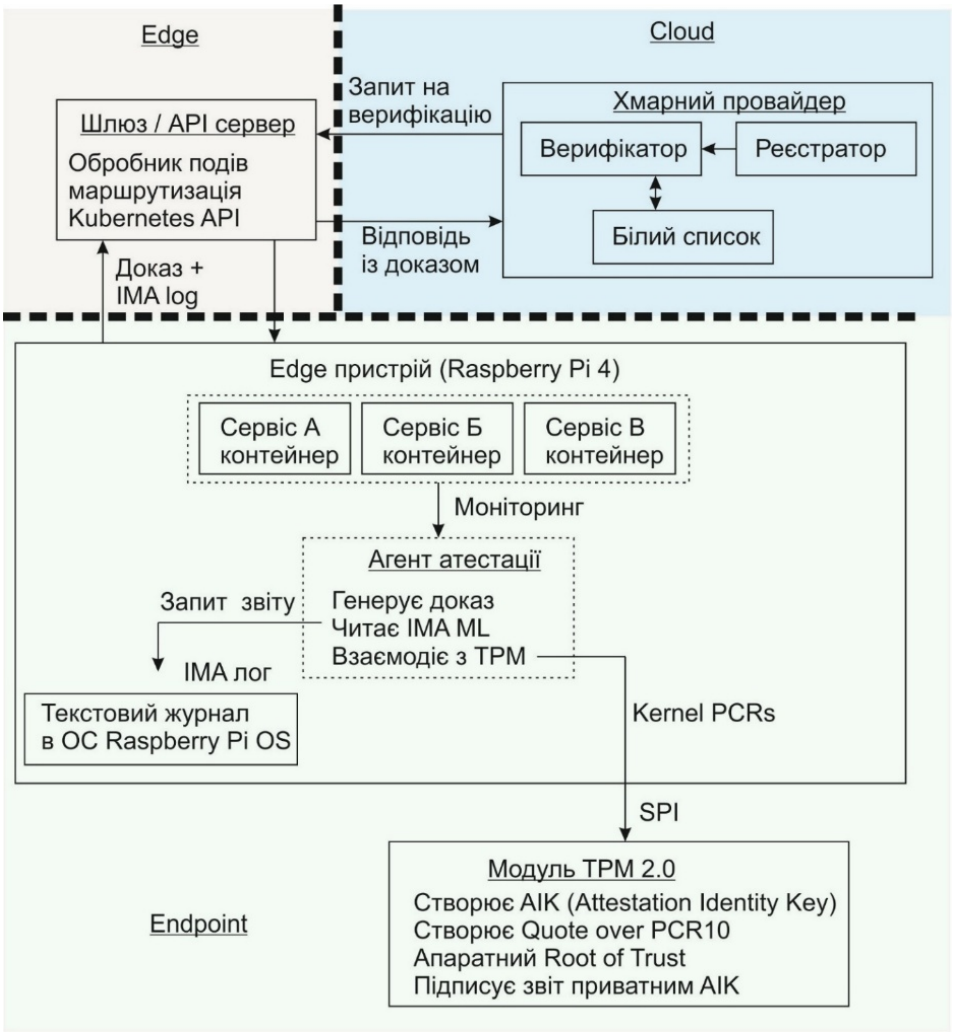

Рисунок 1– Узагальнена архітектура віддаленої верифікації Kubernetes подів для edge-пристроїв на базі Raspberry Pi

Джерело: розроблено авторами

Розглянемо детальніше складові компоненти пропонованої архітектури. Одним із головних компонентів у пропонованій архітектурі є модуль реєстратора. Модуль реєстратора розгортається на вузлі керуючої площини і керує асиметричними ключами та сертифікатами, необхідними для забезпечення безпечного виконання системних операцій. Основними функціями реєстратора є [6]:

– реєстрація робочих вузлів у кластері шляхом зберігання їхніх відповідних ключів ідентифікації атестації (Attestation Identity Key, AIK), що дозволяє валідувати екземпляри доказів, подані модулем агента, розгорнутим на цьому робочому вузлі;

– надання сервісу, який інші компоненти можуть використовувати для взаємодії з ключами орендарів та робочих вузлів для верифікації підписів, обчислених за допомогою відповідних приватних частин;

– зберігання інформації про виробників TPM, включаючи проміжні та кореневі сертифікати центрів сертифікації, що дозволяє верифікувати сертифікати EK TPM, надані новими робочими вузлами під час їхньої реєстрації;

– ідентифікація залучених орендарів і збереження публічного ключа, який наданий ними під час процесу реєстрації в хмарній інфраструктурі; останній використовується для валідування підписів над запитаними діями, тим самим зберігаючи конфіденційність і запобігаючи несанкціонованому доступу до хмарних ресурсів.

Даний модуль розгортаєтсья на керуючій площині, а також керує чотирма ключовими типами даних, забезпечуючи централізований доступ без дублювання компетентностями у інших компонентах.

Спочатку, для орендарів – зареєстрованих користувачів кластера – фіксуються унікальний ідентифікатор UUID (для прив'язки подів до власника), загальна назва та публічний ключ, наданий постачальнику хмари для аутентифікації запитів. Це дозволяє відстежувати поди без розкриття конфіденційної інформації.

Далі, для робочих вузлів – довірених елементів кластера – зберігаються UUID (отриманий під час реєстрації), ім'я хоста та ключ AIK, доведений як належний TPM вузла. AIK слугує для верифікації доказів атестації і криптозвітів (quote), гарантуючи апаратну автентичність.

Щодо виробників TPM, модуль реєстратор утримує список визнаних TCG-виробників з комерційними назвами та 4-байтними ідентифікаторами, які повертаються TPM і використовуються в ланцюгу сертифікатів EK для перевірки властивостей чіпа.

Нарешті, для сертифікатів виробників TPM зберігаються кореневі та проміжні CA: з полями загальної назви та самим сертифікатом. Це охоплює ієрархію, де кореневі CA підписують проміжні, а ті – сертифікати для сімейств пристроїв, забезпечуючи повну валідність ланцюга.

Реєстратор надає уніфікований сервіс для інших компонентів: вони надсилають підписи для верифікації, уникаючи самостійного керування ключами чи перевірки ланцюгів. Такий підхід мінімізує ризики, як-от man-in-the-middle-атаки з підміною ключів, і оптимізує безпеку, зберігаючи всі матеріали в одному захищеному місці.

Ще одним компонентом є обробник робочих вузлів. Обробник робочих вузлів розгортається на головному вузлі і керує процесом реєстрації робочих вузлів, що приєднуються до кластера, завершуючись зберіганням UUID, який унікально ідентифікує кожен робочий вузол і пов'язує його з відповідним AIK у модулі реєстратора. Процес реєстрації робочого вузла включає верифікацію сертифіката EK TPM, демонстрацію резидентності отриманого AIK та верифікацію довіреного завантаження.

Реєстрація робочого вузла в системі верифікації виконується перед процесом приєднання вузла до кластера, оскільки необхідно створити на ньому агента для взаємодії з його TPM. Після завершення реєстрації обробник робочих вузлів комунікує з модулем реєстратора для зберігання інформації про новододаний вузол і асоціює з ним екземпляр агента стану. Також даний модуль виконує розподіл на новозареєстровані робочі вузли публічний ключ модуля верифікатора, необхідний для аутентифікації запитів атестації.

Іншим компонентом, який автоматично розгортається на кожному робочому вузлі та відповідає за керування взаємодією з TPM, журналом вимірювань IMA та цільовим середовищем верифікації є модуль агенту. Основна його функція полягає у створенні AIK ключа та розв'язанні активаційних викликів для останнього під час

реєстрації робочого вузла кластера, зборі тверджень, генерації відповідних доказів та поданні цих даних модулю верифікатора для оцінки.

Ще одним компонентом є агент стану CRD – спеціальний ресурс у Kubernetes, який відстежує стан довіри подів орендарів та робочого вузла, на якому вони виконуються. Екземпляр цього ресурсу створюється для кожного робочого вузла для точного асоціювання кожного пода з вузлом, на якому він розгорнутий. Даний модуль діє як централізований реєстр інформації про довіру, оновлюваний виключно модулем верифікатора у відповідь на результат верифікації доказів. Варто відзначити, що модифікований стан довіри, асоційований з робочим вузлом та кожним подом, розгорнутим на ньому, використовується для збереження останнього результату атестації та надання посилання на нього іншим компонентам.

Модуль контролера стану кластера розгортається на головному вузлі та виконує цикл керування для моніторингу стану робочих вузлів та подів, записаних у кожному агенті CRD. Основними його функціями є виявлення змін, які можуть вимагати коригувальних дій, таких як видалення пода або вилучення робочого вузла, при умові якщо вони були визначені модулем верифікатора як не довірені. Контролер діє як сторона, що покладається на атестацію, забезпечуючи виконання відповідних дій для збереження цілісності кластера на основі змінного стану довіри робочих вузлів та подів.

Іншим компонентом є обробник подів, що розгортається на головному вузлі й надає орендарям інтерфейс для видачі аутентифікованих запитів на розгортання подів та атестацію. Обробник подів виконує наступні функції: верифікує підпис запиту орендаря за допомогою модуля реєстратора для забезпечення його автентичності та незмінності, а потім обробляє його; розгортає поди в просторах імен (namespaces), увімкнених для атестації; перевіряє, чи под, для якого запитана атестація, належить орендарю, що запитує, а потім видає екземпляр об’єкту CRD, що виконує запит на атестацію для запуску модулем верифікатора процесу атестації з залученим модулем агента.

Об’єкт CRD, що виконує запит на атестацію є спеціальним ресурсом, який визначає всю інформацію, необхідну для запуску атестації над цільовим подом, споживаний модулем верифікатора. Даний об’єкт надає інформацію для взаємодії із модулем агента робочого вузла, на якому виконується под. Також об’єкт CRD, що виконує запит на атестацію включає HMAC, обчислений над параметрами запиту за допомогою секрету, спільного з модулем верифікатора, для запобігання обробки останнім неавтентичних запитів атестації.

Іншим компонентом є модуль верифікатора, який розгортається на головному вузлі та відповідає за проведення атестації подів відповідно до вхідних екземплярів об’єкту CRD, що виконує запит на атестацію. Модуль верифікатора здійснює: виконання циклу керування для моніторингу вхідних запитів атестації, використання секрету, що спільний з обробником подів, для валідування отриманого HMAC, тим самим забезпечуючи цілісність запиту атестації, запускає процес атестації з Агентом робочого вузла, що хостить цільовий под, реалізує політику оцінки для оцінювання отриманих доказів та оновлює атрибути агенту CRD цільового робочого вузла відповідно до результату валідування.

Підсистема зберігання білого списку розгортається на головному вузлі та визначає репозиторій затверджених та довірених еталонних значень для всіх сервісів, конфігурацій робочих вузлів та контейнеризованих застосунків, що виконуються в подах. Основними функціями підсистеми зберігання білого списку є: надання модулю верифікатора сервісу для порівняння довірених вимірювань з твердженнями (Claims)

доказів, поданих модулем агентом, централізація керування затвердженим програмним забезпеченням та конфігураціями в одному компоненті, забезпечуючи зберігання кожного запису разом з відповідними еталонними вимірюваннями.

Останнім компонентом пропонованої системи є підсистема моніторингу подів, яка також розгортається на головному вузлі й виявляє створення та видалення подів у кластері. Основними функціями підсистеми моніторингу є: реалізація циклу керування для моніторингу подів, розгорнутих або видалених з усіх робочих вузлів кластера, оновлення відповідного екземпляру агенту стану залученого робочого вузла для відстеження подів, що виконуються та можуть бути предметом верифікації.

Експериментальні дослідження та оцінка ефективності запропонованих рішень. Пропонована архітектура досліджувалась у кластері Kubernetes, який складався із двох елементів – головного та робочого вузла. На головному вузлі було встановлено Ubuntu Server 22.04.3 LTS з ядром версії 5.15.0-91-generic, оскільки ця конфігурація забезпечувала найкращу сумісність з Kubernetes v1.34 та підтримку необхідних kernel features для IMA. Робочий вузол отримав Raspberry Pi OS Lite (Debian Bookworm) з модифікованим ядром 6.1.0, до якого було застосовано патч для підтримки шаблону ima-cgpath. До робочого вузла через SPI шину було підключено апаратний TPM-модуля Infineon OPTIGA SLB 9670.

Процес модифікації ядра Linux для робочого вузла Kubernetes розглядався як необхідна умова коректної роботи механізмів вимірювання цілісності в контейнеризованому середовищі. У стандартній реалізації IMA всі вимірювання формуються на рівні вузла і не містять контексту контейнера або пода, у межах якого виконується процес. За відсутності такого контексту неможливо однозначно співвіднести зафіксовані вимірювання з конкретними подами, що унеможливлює подальшу атестацію та аналіз безпеки на рівні контейнерів у Kubernetes кластері.

Для усунення цього обмеження було реалізовано модифікацію ядра Linux, спрямовану на розширення структури вимірювань IMA додатковим атрибутом, який відображає приналежність процесу до конкретної контрольної групи. Оскільки в Kubernetes кожен Pod та контейнер ієрархічно представлений у вигляді cgroup, повний шлях cgroup у файловій ієрархії ядра є надійним ідентифікатором виконуваного Pod'a. Саме тому до шаблону вимірювань IMA було додано нове поле cgroup-path, яке зберігає цей шлях у текстовому вигляді. Технічно модифікація була реалізована шляхом внесення змін до файлу security/integrity/ima/ima_template.c, який відповідає за формування та відображення полів шаблону IMA.

Для реалізації пропонованої архітектури всі компоненти були розроблені як контейнеризовані сервіси, розгорнуті в Kubernetes кластері. Такі модулі як модуль реєстратора, обробник робочих вузлів, верифікатор, обробник подів та підсистема зберігання білого списку виконувалися як розгортання (тип ресурсу Deployment) із однією реплікою на головному вузлі та були написані на мові Go із використанням бібліотеки controller-runtime для реалізації власних контролерів. Модулі агента розгорталися як DaemonSet на кожному робочому вузлі (що представляв собою Raspberry Pi) і були реалізовані на мові Python із використанням бібліотек tpm2-pytss та gRPC. Всі взаємодії між компонентами здійснювалися через Kubernetes API та внутрішній gRPC-канал (для передачі підтвердження від агента до модуля верифікатора).

Оцінка ефективності розробленої системи верифікації проводилася через порівняння з базовою конфігурацією Kubernetes кластера без механізмів верифікації, а також з reference implementation на базі Keylime. В процесі експериментальних досліджень було проведено заміри часових характеристик, що включали виконання

повного циклу верифікації. Вимірювання виконувалися на ідентичному апаратному забезпеченні Raspberry Pi 3 Model B з 4GB RAM для забезпечення коректного порівняння. Для пода із двома контейнерами на базі образу Redis розмір IMA логу складав приблизно 18000 записів, з яких 247 були релевантні для цього Pod'a (рис. 2). Час верифікації одного такого Pod'a становив 4.2 секунди, з розподілом: генерація доказу на робочому вузлі (включаючи TPM quote) – 1.8 секунди, передача по мережі – 0.3 секунди, валідація на модулі верифікації (включаючи парсинг IMA log та запити до підсистеми зберігання білого списку) – 2.1 секунди. При збільшенні кількості одночасних атестацій до 10 Подів середній час зростав до 6.7 секунд за рахунок конкуренції за TPM ресурс, який підтримує лише послідовний доступ.

Порівняння з Keylime для верифікації показало перевагу пропонованого спеціалізованого рішення. Зокрема Keylime потребувала 7.3 секунди для верифікації одного робочого вузла (що включало под неявно), але не могла розрізняти окремі поди, що робило неможливим вибіркове видалення скомпрометованого пода без перезавантаження всього вузла кластера, тоді як пропонована архітектура мала підтримку точкового виявлених порушень безпеки (на рівні окремих компонентів).

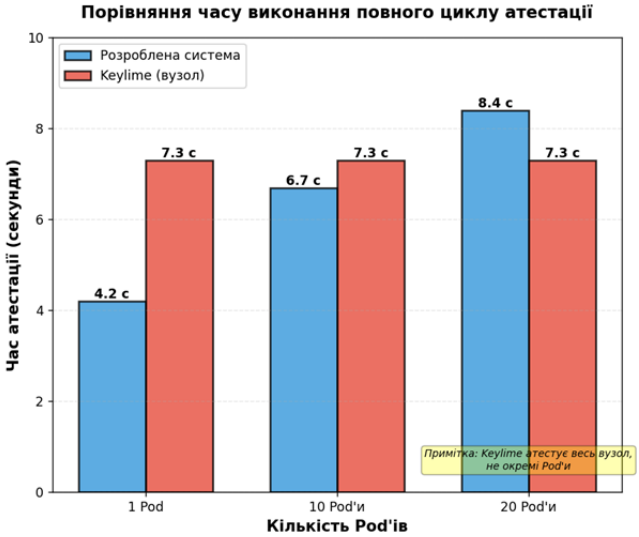

Рисунок 2 – Порівняння часу виконання повного циклу верифікації на базі пропонованої архітектури та Keylime

Джерело: розроблено авторами

Висновки. Таким чином, проведене дослідження показало наявні переваги та проблемні аспекти реалізації віддаленої верифікації подів на периферійних пристроях. У розрізі цього було запропоновано архітектуру віддаленої верифікації Kubernetes подів, яка нативно інтегрується в Kubernetes кластер, без змін ядра, використовуючи стандартні механізми розширюваності, такі як CRD та користувацькі контролери. Запропонована архітектура дає змогу контролювати стан контейнерів на одноплатних комп'ютерних системах Raspberry Pi та подібних периферійних пристроях. Для цього використовувались апаратний TPM-модуль Infineon OPTIGA SLB 9670, який виступав джерелом довіри, та IMA, яка збирала вимірювання з урахуванням того, як саме працюють поди. Основними елементами архітектури стали спеціальні CRD для відображення станів верифікації, контролери на головному вузлі для координації процесів та агент, що розташовується на кожному робочому вузлі кластеру для створення криптографічних доказів.

Для проведення експериментальних досліджень було здійснено імплементацію пропонованої архітектури в розгорнути Kubernetes кластер. Порівняння здійснювалось із фреймворком Keylime. Було встановлено, що для пропонованого рішення час повного циклу верифікації одно поду склав 4,2, що є швидшим за схожий процес, реалізований через Keylime.

Список літератури

1. Mo W., Wang T., Zhang S., Zhang J. An active and verifiable trust evaluation approach for edge computing. *J. Cloud Comput*. 2020. Vol. 9. P. 51. doi: 10.1186/s13677-020-00202-w
2. Rostampour S., Bagheri N., Bendavid Y., Safkhani M., Kumari S., Rodrigues J. J. An authentication protocol for next generation of constrained Iot systems. *IEEE Internet Things Journal.* 2022. Vol. 9. P. 21493–21504. doi: 10.1109/JIOT.2022.3184293
3. Chen B., Wan J., Celesti A., Li D., Abbas H., Zhang Q. Edge Computing in IoT-Based Manufacturing. *IEEE Commun. Mag.* 2018. № 56. P. 103–109.
4. Rupanetti D., Kaabouch N. Combining Edge Computing-Assisted Internet of Things Security with Artificial Intelligence: Applications, Challenges, and Opportunities. *Applied Science* 2024. № 14. P. 7104.
5. Keylime: remote boot attestation and runtime integrity management solution. Cloud Native Computing Foundation (CNCF). URL: https://keylime.dev/ (дата звернення: 10.02.2026)
6. Zaritto F. Kubernetes Pods Remote Attestation : PhD Thesis. Politecnico di Torino, 2024. 102 p.
7. Zaritto F., Bravi E., Sisinni S., Lioy A. Extending Kubernetes for Pods Integrity Verification. *Journal of Network and Systems Management*. 2026. Vol 34, 14.
8. Scopelliti G., Amir-Mohammadian S., Csallner C. Trusting the Cloud-Native Edge: Remotely Attested Kubernetes Workers. 2024. arxiv.org/abs/2405.10131.
9. Goethals T., De Turck F., Volckaert B. Extending Kubernetes Clusters to Low-Resource Edge Devices Using Virtual Kubelets. IEEE Transactions on Cloud Computing. 2020, vol. 10, no. 4, pp. 2623-2636.
10. Fernandez G. P., Brito A. Secure container orchestration in the cloud: policies and implementation. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY : ACM, 2019. P. 138–145.
11. Косаревський Б. В., Тецький А. Г. Сучасні підходи до розгортання інфраструктури мобільних інтелектуальних систем. *Сучасний стан наукових досліджень та технологій в промисловості*. 2025. № 2(32), с. 33–48.
12. Коваленко О. Є. Інтелектуалізація граничних обчислень Інтернету речей. *Математичні машини і системи*. 2024. № 3–4. С. 50–68.

References

1. Mo, W., Wang T., Zhang S., Zhang J. (2020) An active and verifiable trust evaluation approach for edge computing. *J. Cloud Comput*. 9, 51. doi: 10.1186/s13677-020-00202-w
2. Rostampour, S., Bagheri, N., Bendavid, Y., Safkhani, M., Kumari, S., Rodrigues, J. J. (2022) An authentication protocol for next generation of constrained Iot systems. *IEEE Internet Things Journal.* 9, 21493-21504. doi: 10.1109/JIOT.2022.3184293
3. Chen, B., Wan, J., Celesti, A., Li, D., Abbas, H., Zhang, Q. (2018) Edge Computing in IoT-Based Manufacturing. *IEEE Commun. Mag.* 56, 103–109.
4. Rupanetti, D., Kaabouch, N. (2024) Combining Edge Computing-Assisted Internet of Things Security with Artificial Intelligence: Applications, Challenges, and Opportunities. *Applied Science.* 14, 7104.
5. Keylime: remote boot attestation and runtime integrity management solution. (2026, February 10). *Cloud Native Computing Foundation (CNCF)*. https://keylime.dev/
6. Zaritto, F. Kubernetes Pods Remote Attestation (2024): PhD Thesis. Politecnico di Torino.
7. Zaritto, F., Bravi, E., Sisinni, S., Lioy, A. (2026) Extending Kubernetes for Pods Integrity Verification. *Journal of Network and Systems Management*. 34, 14.
8. Scopelliti, G., Amir-Mohammadian, S., Csallner, C. (2024) Trusting the Cloud-Native Edge: Remotely Attested Kubernetes Workers. *arxiv.org/abs/2405.10131*.
9. Goethals, T., De Turck, F., Volckaert, B. Extending Kubernetes Clusters to Low-Resource Edge Devices Using Virtual Kubelets (2020). *IEEE Transactions on Cloud Computing*. 10 (4), 2623-2636.

10. Fernandez, G. P., Brito, A. Secure container orchestration in the cloud: policies and implementation (2019). *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY : ACM 138-145.
11. Kosarevskyi, B. V., Tetskyi, A. H. (2025) Suchasni pidkhody do rozghortannia infrastruktury mobilnykh intelektualnykh system. *Suchasnyi stan naukovykh doslidzhen ta tekhnolohii v promyslovosti*. 2(32), 33–48.
12. Kovalenko, O. Ye. (2024) Intelektualizatsiia hranychnykh obchyslen Internetu rechei. *Matematychni mashyny i systemy*. 3–4, 50–68.

Andrii Nicheporuk, Assoc. Prof., PhD tech. sci.,, **Vadym Hurskyi**
Khmelnytskyi National University, Khmelnytskyi, Ukraine

Architecture of Remote Verification of Kubernetes Containerized Applications for Edge Devices Based on Raspberry Pi

This study addresses the security gap in edge computing environments by proposing a specialized architecture for the remote verification of Kubernetes pods. While traditional frameworks like Keylime focus on node-level integrity, they fail to provide the granularity required for microservice-oriented infrastructures where multiple isolated workloads (pods) run on a single node.

The proposed architecture natively integrates into the Kubernetes ecosystem using standard extensibility mechanisms, specifically Custom Resource Definitions (CRDs) and custom controllers, allowing verification logic to operate as a seamless part of the cluster's control plane without requiring changes to the core Kubernetes code. The system is specifically optimized for resource-constrained edge devices, such as the Raspberry Pi 3 Model B, utilizing the Infineon OPTIGA SLB 9670 hardware TPM 2.0 module as a hardware root of trust.

To achieve granular pod-level attestation, the architecture leverages the Linux Integrity Measurement Architecture (IMA). A key technical innovation includes a kernel modification that introduces the cgroup-path attribute to the IMA measurement template, enabling the system to uniquely associate cryptographic measurements with specific Kubernetes pods. The architecture consists of several core components: a Registrar module for key management, a Verifier for evaluating evidence against trusted whitelists, and a Node Agent that generates cryptographic quotes via gRPC.

Experimental results obtained from a deployed cluster show that the proposed solution completes a full verification cycle in 4.2 seconds for a single pod, significantly outperforming the 7.3 seconds required by the Keylime framework for a similar process. Furthermore, unlike traditional tools, this architecture enables precise remediation, such as terminating only the compromised pod without the need to reboot or isolate the entire worker node, thus preserving the availability of other edge services.

Verification process, Edge devices, TPM module, Raspberry Pi, Integrity Measurement Architecture (IMA)

Одержано (Received) 12.02.2026 *Прорецензовано (Reviewed) 18.02.2026*
Прийнято до друку (Approved) 24.02.2026