Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМіТ

Марина ЄСІНА

"Допущено до захисту"

« »_____2025р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Аналіз механізмів автентифікації користувачів та пристроїв в рамках архітектури нульової довіри»

оцінка « _____ »

Керівник: д.т.н., професор Єсін В.І.

Голова ЕК
Мичуда Л.З.

Рецензент: к.т.н., професор Качко О.Г.

Виконавець: студент групи КБ-41
Семенюк Є.С.

Харків 2025

РЕФЕРАТ

Пояснювальна записка до роботи бакалавра містить 59 сторінок, 10 рисунків, 3 таблиці, 8 додатків, 46 посилань на джерела.

Мета роботи полягає в аналізі механізмів автентифікації користувачів та пристроїв у рамках архітектури нульової довіри, оцінка їхньої ефективності, виявлення сильних і слабких сторін, а також визначення перспективних напрямів розвитку технологій автентифікації у сучасних умовах.

Об’єкт дослідження — процеси автентифікації користувачів і пристроїв у інформаційних системах.

Предмет дослідження — механізми автентифікації, що застосовуються в сучасних інформаційних системах та відповідають принципам нульової довіри.

Методами дослідження є аналіз теоретичних засад, вивчення існуючих рішень, оцінку їхньої ефективності та ідентифікацію потенційних загроз. Методи дослідження включають як якісний, так і кількісний аналіз.

У роботі досліджено: ключові принципи ZTA, багатофакторну, контекстно-орієнтовану й безперервну автентифікацію, а також механізми перевірки ідентичності пристроїв у розподілених і IoT-середовищах.

Результати роботи можуть бути використані як основа для подальших досліджень у сфері кібербезпеки, а також для розробки практичних рішень, спрямованих на підвищення рівня захисту сучасних інформаційних систем.

Ключові слова: ZERO TRUST ARCHITECTURE, АВТЕНТИФІКАЦІЯ, IOT, MTLS, OAUTH 2.0, OPENID CONNECT, SAML, MFA, RBAC, ABAC, ПОВЕДІНКОВА БІОМЕТРІЯ, КОНТЕКСТНА АВТЕНТИФІКАЦІЯ, ЛЕГКОВАГОВІ ПРОТОКОЛИ, TPM, PUF, RMAC

ABSTRACT

The explanatory note to the bachelor's thesis contains 59 pages, 10 figures, 3 tables, 8 appendices, 46 references to sources.

The purpose of the work is to analyze user and device authentication mechanisms within the framework of the zero trust architecture, assess their effectiveness, identify strengths and weaknesses, and determine promising areas for the development of authentication technologies in modern conditions.

The object of the study is the processes of user and device authentication in information systems.

The subject of the study is authentication mechanisms used in modern information systems and comply with the principles of zero trust.

The research methods are the analysis of theoretical foundations, the study of existing solutions, the assessment of their effectiveness, and the identification of potential threats. The research methods include both qualitative and quantitative analysis.

The paper explores: key principles of ZTA, multifactor, context-based and continuous authentication, as well as mechanisms for verifying device identity in distributed and IoT environments.

The results of the paper can be used as a basis for further research in the field of cybersecurity, as well as for developing practical solutions aimed at increasing the level of protection of modern information systems.

Keywords: ZERO TRUST ARCHITECTURE, AUTHENTICATION, IOT, MTLS, OAUTH 2.0, OPENID CONNECT, SAML, MFA, RBAC, ABAC, BEHAVIORAL BIOMETRICS, CONTEXTUAL AUTHENTICATION, LIGHTWEIGHT PROTOCOLS, TPM, PUF, RMAC.

ЗМІСТ

ВСТУП

Актуальність роботи. Процедура встановлення особи користувача або пристрою перед наданням доступу до ресурсів залишається надзвичайно важливою складовою інформаційної безпеки сучасних IT-інфраструктур. Традиційні методи автентифікації, що базуються на паролях, апаратних токенах чи біометрії, дедалі більше виявляють свої обмеження: статичні облікові дані піддаються атакам методом brute-force, фішингу та словниковим атакам, а людський фактор часто призводить до використання простих або повторно застосованих паролів. Крім того, після початкової перевірки звичайні системи рідко здійснюють повторну валідацію протягом сеансу, що робить їх вразливими до викрадення активних сесій.

Поширення хмарних сервісів, Інтернету речей (IoT) та віддаленої роботи призвело до розмивання периметра корпоративних мереж і появи нових точок доступу, які неможливо ефективно захистити за допомогою традиційних засад «довіра всередині, недовіра зовні». Саме тому концепція нульової довіри (Zero Trust), запропонована Дж. Кіндервагом у 2010 році, відмовляється від автоматичної довіри до будь-якого користувача чи пристрою незалежно від їхнього розташування та вимагає безперервної оцінки ризику кожного запиту [1].

Подальший розвиток підходу Zero Trust отримав імпульс зі зростанням кібератак і опублікуванням у 2020 році моделі нульової довіри від NIST, яка сформувала практичні рекомендації щодо реалізації багатофакторної автентифікації, моніторингу поведінкових факторів та адаптивних політик доступу [2]. Врахування контексту сеансу, зміни поведінкових характеристик користувача та характеристик пристрою дозволяє підвищити стійкість систем до внутрішніх і зовнішніх загроз, що робить дослідження механізмів автентифікації в рамках Zero Trust особливо актуальним для сучасних інформаційних систем.

Мета дослідження. Аналіз механізмів автентифікації користувачів та пристроїв у рамках архітектури нульової довіри, оцінка їхньої ефективності,

виявлення сильних і слабких сторін, а також визначення перспективних напрямів розвитку технологій автентифікації у сучасних умовах.

Завдання дослідження. Для досягнення поставленої мети необхідно:

1. Дослідити традиційні підходи до автентифікації, визначити їх основні проблеми, недолікі та обмеження.

2. Проаналізувати можливості концепції нульової довіри, її ключових принципів та вимог, які пред'являються до автентифікаційних механізмів у межах архітектури а також.

3. Проаналізувати механізми автентифікації користувачів і пристроїв, особливо в умовах Інтернету речей та децентралізованих середовищ

4. Визначити можливі напрямки розвитку технологій автентифікації у контексті нульової довіри.

5. Сформулювати рекомендації щодо ефективного і безпечного використання механізмів автентифікації, які відповідатимуть вимогам сучасних інформаційних систем та архітектури нульової довіри.

Очікувані результати дослідження. Очікується, що аналіз існуючих підходів дозволить виявити їхні переваги та недоліки, а також сприятиме визначенню основних загроз і ризиків, пов'язаних із впровадженням нових технологій автентифікації, та формуванню рекомендацій щодо їхньої мінімізації. Отримані результати можуть бути використані як основа для подальших досліджень у сфері кібербезпеки, а також для розробки практичних рішень, спрямованих на підвищення рівня захисту сучасних інформаційних систем.

Об'єктом дослідження є процеси автентифікації користувачів і пристроїв у інформаційних системах.

Предметом дослідження є механізми автентифікації, що застосовуються в сучасних інформаційних системах та відповідають принципам нульової довіри.

Методи дослідження. Дослідження механізмів автентифікації користувачів охоплює аналіз теоретичних засад, вивчення існуючих рішень, оцінку їхньої ефективності та ідентифікацію потенційних загроз. Дослідження включають як якісний, так і кількісний аналіз.

1 АНАЛІЗ ОСНОВНИХ ПРИНЦИПІВ ТА МЕТОДІВ АВТЕНТИФІКАЦІЇ В РАМКАХ АРХІТЕКТУРИ НУЛЬОВОЇ ДОВІРИ

1.1 Архітектура нульової довіри

Архітектура нульової довіри (Zero Trust Architecture, ZTA) — це підхід до побудови інформаційно-комунікаційних систем, який ґрунтується на принципах концепції Zero Trust (ZT). Основою побудови архітектури є постулат ZT "ніколи не довіряй, завжди перевіряй", тобто відсутність автоматичної довіри навіть до користувачів і пристроїв, які перебувають усередині корпоративної мережі. Це кардинально відрізняється від класичних систем безпеки, захист яких передбачає надійність внутрішнього середовища. Національний інститут стандартів і технологій США (NIST) визначає Zero Trust як модель, що вимагає автентифікації, авторизації та безперервного моніторингу всіх суб’єктів доступу, незалежно від їхнього розташування [2].

Проблема теперішнього захисту більшості IT-інфраструктур полягає в тому, що вони передбачають чіткий поділ на "довірену" внутрішню мережу та "недовірену" зовнішню. Однак сучасні загрози, такі як атаки зсередини, викрадення облікових даних, злам хмарних середовищ і використання скомпрометованих пристроїв, роблять цей підхід застарілим. Згідно з даними компанії Hideez, внутрішні загрози були пов'язані з 65% витоків даних у 2023 році [3]. ZTA передбачає, що будь-яка мережа — потенційно небезпечна, а отже, всі запити на доступ мають піддаватися ретельній перевірці та контролю.

Критично важливими елементами інформаційної безпеки у світі є ідентифікація та автентифікація. В існуючих рішеннях безпеки перевірка виконується один раз, під час початку роботи із системою, що створює значні ризики, оскільки після успішного входу зловмисник може діяти практично безперешкодно. У свою чергу концепція ZT вимагає, щоб процес підтвердження тривав постійно, допоки користувач або пристрій залишаються в мережі. Це означає, що кожна взаємодія з ресурсами організації повинна підлягати повторному аналізу. Для цього можуть застосовуватись різні методи, зокрема

багатофакторна автентифікація (MFA), поведінковий аналіз або контроль контексту запиту [4]. Так, якщо співробітник завжди працював з однієї географічної локації, але раптово намагається отримати доступ до інфраструктури з іншої країни або з технічного засобу, якого раніше не використовувала, система повинна розцінювати це як потенційну загрозу та вимагати додаткового підтвердження особи.

Крім контролю користувачів, Zero Trust передбачає жорстке регулювання доступу до ресурсів. Це реалізується через принцип найменших привілеїв (Least Privilege Access) — надання кожному суб'єкту лише мінімально необхідних дозволів для виконання його функцій. Такий підхід запобігає ситуаціям, коли компрометований обліковий запис може бути використаний для горизонтального переміщення мережею та отримання доступу до критично важливих даних. Навіть якщо співробітник має доступ до певних корпоративних систем, це не означає, що він автоматично може отримати доступ до всіх даних організації. Кожен запит розглядається окремо, а доступ надається лише після підтвердження відповідності політикам безпеки.

У межах корпоративної мережі ZTA вимагає впровадження мікросегментації та розділення зон довіри, що означає поділ інфраструктури на невеликі, ізольовані підмережі. Кожен компонент—сервер, база даних чи прикладний сервіс—взаємодіє з іншими лише через ретельно налаштовані та постійно моніторовані канали зв'язку. На практиці це означає, що навіть якщо зловмиснику вдасться скомпрометувати один із серверів, без повторної автентифікації та авторизації він не зможе «перейти» до інших сегментів. Така архітектура створює кілька рівнів захисту, де кожен новий запит на доступ до даних чи системи проходить сувору перевірку незалежно від ознак відправника.

Застосування принципу "ніколи не довіряй, завжди перевіряй" дозволяє значно підвищити рівень безпеки організації. Однак впровадження цього підходу вимагає суттєвих змін у корпоративній інфраструктурі, включаючи оновлення політик безпеки, модернізацію механізмів контролю доступу та інтеграцію з аналітичними системами моніторингу [2]. Водночас це дозволяє мінімізувати

ризики, пов’язані з витоками даних, атаками зсередини та використанням вразливостей у системах безпеки. Порівняння традиційного підходу захисту із концепцією нульової довіри надано у таблиці 1.1.

Таблиця 1.1 — Порівняння традиційного підходу захисту із концепцією нульової довіри

Критерій	Традиційний підхід захисту	Zero Trust
Підхід	Довіряй, але перевіряй	нікому не довіряй, завжди перевіряй
Границя довіри	Зовнішні (недовірені) та внутрішні (довірені) мережі	Мікросегментація, немає довірених зон
Контроль доступу	Базується на IP (порт, протокол)	Орієнтований на дані, мінімально необхідний доступ
Шифрування трафіку	Тільки для зовнішнього трафіку, внутрішній може бути незашифрований	Шифрування всього трафіку без виключень
Автентифікація	Одноразова перевірка під час входу	Перевірка перед кожним доступом та постійний моніторинг
Політика безпеки	Заздалегідь визначені правила та загальні політики	Гнучкі правила, адаптивні політики, аналіз ризиків
Моніторинг та управління	Індивідуальний моніторинг та ручне управління	Автоматизація, поведінковий аналіз, контроль пристроїв та сервісів
Модель довіри	Довіряє пристроям і користувачам всередині корпоративної мережі	Нульова довіра, кожен запит підлягає перевірці
Захист пристроїв	Захист обмежується корпоративними пристроями	Контроль стану всіх пристроїв, незалежно від місця підключення
Виявлення загроз	Реактивний підхід, виявлення після атаки	Проактивний підхід, передбачення загроз та швидке реагування
Масштабованість	Обмежена, складно інтегрувати з хмарними технологіями	Гнучка, легко масштабується для хмарних середовищ та віддаленої роботи

1.2 Автентифікація користувачів

Автентифікація користувачів відіграє центральну роль у реалізації архітектури нульової довіри. Оскільки Zero Trust ґрунтується на принципі постійної перевірки кожного запиту незалежно від його походження, ефективна автентифікація є основним механізмом забезпечення безпеки доступу до ресурсів. Традиційні системи безпеки здебільшого покладаються на одноразову автентифікацію під час входу до системи. Однак такий підхід має суттєві

вразливості: після початкового входу зловмисник, який отримав контроль над обліковим записом, може безперешкодно використовувати привілеї користувача. За статистикою, значна частина кібератак здійснюється саме через використання скомпрометованих паролів або облікових записів [4]. Періодична зміна паролів частково вирішує ці проблеми, однак такий метод не гарантує повного захисту. В умовах нових рішень безпеки перевірка особи має відбуватися на кожному етапі доступу, перевіряючи не тільки правильність введених даних, але й контекст взаємодії: місцезнаходження користувача, пристрій, з якого здійснюється запит, час доступу та поведінкові патерни [5].

Однофакторна автентифікація – один із перших методів підтвердження особи користувача. Цей метод є найпростішим, але водночас найменш захищеним. Основна ідея базується на одному чиннику перевірки, найчастіше паролі або PIN-коді. Це було зручно, але з часом стало зрозуміло, що такий спосіб є вразливим до атак, оскільки паролі можуть бути зламані, викрадені, або вгадані, у випадку використання слабких комбінацій.

Багатофакторна автентифікація (Multi-Factor Authentication, MFA) є значно надійнішим методом і є обов’язковим компонентом Zero Trust [6]. Вона передбачає використання щонайменше двох незалежних факторів перевірки з різних категорій: щось, що користувач знає (пароль, PIN-код, секретне питання); щось, що користувач має (мобільний токен, смарт-карта, USB-ключ); щось, що у користувача є (біометричні дані: відбиток пальця, розпізнавання обличчя, сканування райдужної оболонки ока) [7].

Комбінація цих факторів значно ускладнює компрометацію облікового запису, оскільки навіть у разі викрадення пароля зловмиснику буде важко пройти автентифікацію без додаткового підтвердження особи.

Останні тенденції у MFA включають використання FIDO2-технологій, які дозволяють здійснювати автентифікацію без введення паролів. Під час реєстрації на сервер передається лише публічний ключ, а приватний генерується й зберігається в ізольованому модулі пристрою, доступ до якого розблоковується локально. При спробі входу (див. рисунок 1.1) сервер надсилає одноразове

випадкове завдання (challenge), яке підписується приватним ключем у захищеному сховищі і повертається назад. Перевіривши цей підпис за допомогою збереженого публічного ключа та переконавшись, що відповідь належить поточній сесії й конкретному пристрою, сервер підтверджує особу користувача. Така схема гарантує, що секретні дані ніколи не покидають пристрій, а перевірка домену запиту унеможливлює успішні фішингові атаки.

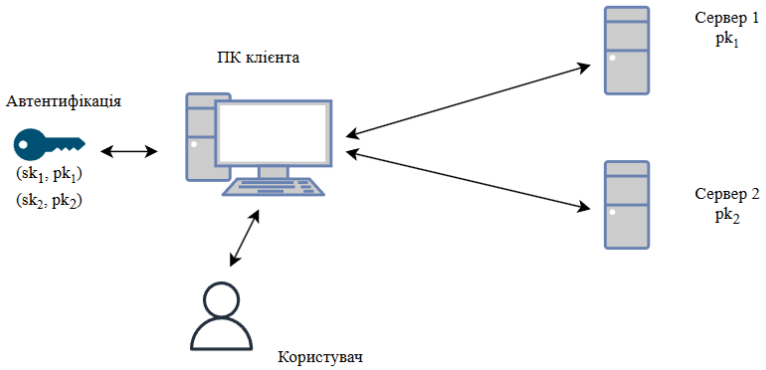

Рисунок 1.1 — Автентифікація користувача з використанням технології FIDO2

Такі підходи хоч і значно підвищили рівень безпеки, у порівнянні із однофакторними методами, проте все ще не усувають ризик компрометації облікового запису після успішного проходження перевірки. Саме тому в Zero Trust активно використовуються нові підходи, зокрема контекстна (Context-Aware) та безперервна автентифікація (Continuous Authentication), які забезпечують динамічний контроль доступу на основі аналізу поведінки користувача, характеристик пристрою та інших факторів ризику.

Контекстна автентифікація базується на тому, що доступ користувача до ресурсів визначається не лише його ідентифікаційними даними, такими як ім’я користувача чи пароль, а й цілим рядом інших факторів, що стосуються поточної ситуації, в якій здійснюється підключення. Параметрами можуть бути географічне місцезнаходження користувача, тип пристрою, з якого здійснюється доступ, мережа, до якої здійснюється доступ, або час доби. Однак, отримавши ці дані лише один раз не можна гарантувати безпеку всього сеансу, оскільки контекстні

умови можуть змінюватися в режимі реального часу. Тому безперервна перевірка є наступним кроком у розвитку автентифікаційних механізмів, оскільки дозволяє здійснювати динамічну оцінку поведінки користувача протягом усього часу його роботи в системі. Система захисту повинна аналізувати активність користувача в реальному часі: місцезнаходження, послідовність натискання клавіш, швидкість руху миші, взаємодію з інтерфейсом, тощо [8]. Застосування цих технологій у межах Zero Trust дозволяє організаціям не тільки запобігати атакам на етапі автентифікації, а й оперативно реагувати на загрози, які виникають під час активного використання ресурсів. Поєднання контекстної автентифікації та безперервного моніторингу створює надійний рівень захисту, що значно перевершує традиційні методи ідентифікації користувачів.

Примусовий контроль доступу є ключовим механізмом у ZTA, що забезпечує суворе дотримання політик безпеки під час кожного запиту користувача або пристрою. Ця модель реалізується через спеціалізовані компоненти: Policy Enforcement Point (PEP) та Policy Decision Point (PDP) (див. рисунок 1.2). PEP відповідає за перехоплення всіх запитів на доступ і передає їх у PDP для аналізу. PDP, у свою чергу, оцінює запит на основі визначених політик безпеки, контексту доступу, рівня ризику та інших параметрів. Далі формується рішення про надання або відхилення доступу.

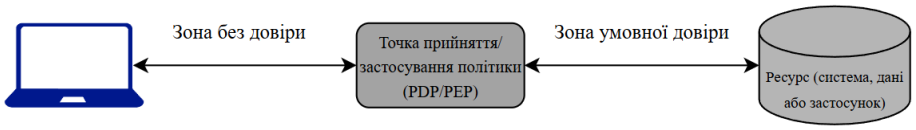

Рисунок 1.2 — Схема контролю доступу в архітектурі нульової довіри

1.3 Автентифікація пристроїв

Як і у випадку із користувачами, архітектура нульової довіри передбачає, що жоден пристрій не повинен отримувати доступу до ресурсів мережі за замовчуванням, навіть якщо він фізично знаходиться у корпоративній інфраструктурі. Підходи, які використовують сьогодні для забезпечення безпеки пристроїв мають ряд недоліків, серед яких можна виділити: відсутність перевірки

цілісності пристроїв перед їх підключенням, що дозволяє зловмисникам використовувати скомпрометовані або заражені пристрої; залежність від статичних ідентифікаторів, таких як MAC-адреси, які можуть бути підмінені; вразливість до атак «людина посередині» (MITM) та повторного використання викрадених облікових даних [7]; відсутність адаптивної перевірки контексту підключення, що не дозволяє виявляти аномальну активність [7].

Враховуючи ці проблеми, особливо важливо впроваджувати та використовувати нові методи безпеки, які пропонуються при реалізації архітектури нульової довіри. А саме, криптографічні сертифікати, TPM, механізми поведінкового аналізу — лише низка інструментів для забезпечення динамічної автентифікації та мінімізації ризиків компрометації.

Цифрові сертифікати створюють ефективний механізм взаємної перевірки між пристроями та сервісами без необхідності залучення централізованих довірених посередників. Вони дозволяють встановлювати надійні з'єднання між різними елементами мережі, а також інтегрувати в них метаінформацію про рівень доступу до кожного ресурсу. Завдяки цьому, система може автономно визначати допустимість підключення, спираючись на локальну базу політик безпеки.

Важливим компонентом також є Trusted Platform Module (TPM). Цей апаратний модуль гарантує безпеку криптографічних операцій і захищає зберігання ключів автентифікації. Завдяки здатності перевіряти цілісність пристрою під час завантаження операційної системи, TPM захищає систему навіть у разі фізичного доступу зловмисників. Цей модуль може генерувати та зберігати криптографічні ключі, підписувати дані та перевіряти автентичність програмного середовища, що значно підвищує рівень довіри до пристрою.

Довірені платформи (Trusted Execution Environments, TEE) доповнюють цифрові сертифікати та TPM, надаючи безпечне середовище для виконання критичних процесів. Вони забезпечують ізольоване середовище (див. рисунок 1.3), у якому можуть виконуватись криптографічні операції, зберігатись конфіденційні дані та виконуватись перевірка автентичності пристроїв. У

поєднанні з механізмами взаємної перевірки сертифікатів та TPM, TEE дозволяє мінімізувати ризик атак на рівні пристроїв та унеможливлює перехоплення даних під час автентифікації.

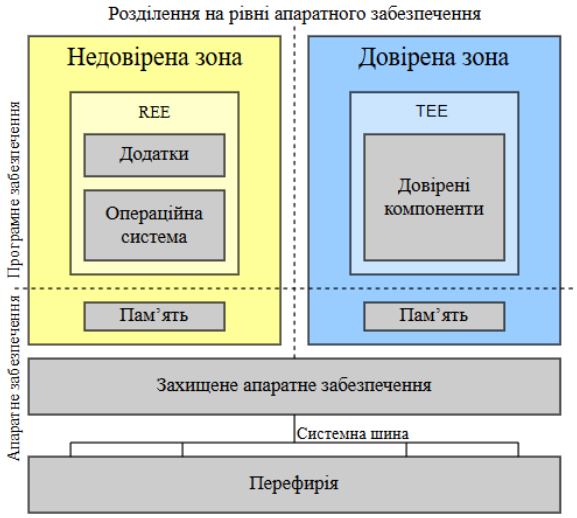

Рисунок 1.3 — Архітектура поділу довіреного та недовіреного середовищ у пристрої [9]

1.4 Протоколи та моделі автентифікації у ZTA

1.4.1 Сертифікат-орієнтовані протоколи

У сертифікат-орієнтованих протоколах автентифікація в Zero Trust ґрунтується на криптографічній довірі, яку забезпечують цифрові сертифікати X.509 та інфраструктура відкритих ключів (PKI). Застосування цих механізмів дозволяє точно ідентифікувати кожен елемент мережі, водночас виконуючи детальну валідацію перед будь-якою операцією доступу.

Сертифікати X.509 виконують роль «паспортів» для користувачів і пристроїв: вони містять публічний ключ, інформацію про видавця (цілісний або проміжний CA), термін дії і набір атрибутів, які політика Zero Trust може використовувати у процесі ухвалення рішення. У базовому сценарії перевірка обмежується запитом до CRL або OCSP і дозволяє виявити відкликані чи

скомпрометовані ключі — цього достатньо для захисту від очевидних загроз, але не враховує різної надійності самої інфраструктури CA [2].

Для більш тонкої диференціації довіри до сертифіката доцільно вводити градацію на основі його метаданих [2]. Наприклад, атрибут Issuer дозволяє відокремити кореневі CA високого рівня довіри (enterprise root) від проміжних або публічних CA, а поле Signature Algorithm — розрізняти сучасні (SHA-256/ECDSA) й застарілі (SHA-1/RSA) підписи. Аналіз Certificate Policy OID та Key Usage допомагає визначити, чи призначений сертифікат лише для TLS, або для автентифікації клієнта, чи для цифрового підпису. Підписані сильнішим алгоритмом ієрархічно вищі сертифікати можна зарахувати до вищої «категорії довіри», а менш надійні — карати зменшенням рівня довіри або додатковими перевірками .

Для відповідності критеріям ZTA ці атрибути повинні інтегруватись у політику на основі ризиків: кожному полю сертифіката присвоюється нормалізоване значення і вага, після чого обчислюється сертифікаційний вклад у загальний рівень довіри. Це дозволяє поєднати статус «дійсний/відкликаний» із градацією за видавцем, алгоритмом і призначенням ключа, а також динамічно коригувати правила доступу в залежності від поточного рівня загроз або контексту запиту. Хоча прості перевірки дійсний/відкликаний мінімізують затримку й адміністративні накладні витрати, проте вони ризикують надто спростити довіру. Градація за атрибутами підвищує безпеку, але вимагає підтримки метаданих і буде збільшувати складність у PDP. У свою чергу рейтинговий підхід з оцінювання ризику дає максимум гнучкості й адаптивності, проте потребує тонкого калібрування ваг і безперервного оновлення списків довірених CA. У результаті для середовищ із підвищеними вимогами до безпеки рекомендується комбінувати ці рівні: базову перевірку дійсний/відкликаний, вторинну градацію атрибутів сертифіката й розширений risk-score, інтегрований з іншими сигналами Zero Trust.

У контексті ZTA ключовим викликом є не одноразова перевірка сертифіката під час встановлення TLS-з’єднання, а безперервне підтвердження його дійсності

на кожному запиті доступу. Постійні OCSP-запити або періодичні оновлення CRL створюють дві основні проблеми: по-перше, затримки (від сотень мілісекунд до кількох секунд) під час очікування відповіді CA, що може блокувати доступ до критичних сервісів, по-друге, надмірне кешування сертифікатів для зниження навантаження призводить до потенційного «вікна вразливості», коли відкликаний сертифікат ще вважається дійсним .

Аналітичний підхід до оптимізації цього процесу повинен включати моделювання двох параметрів:

1) Затримка CRL/OCSP – дослідження часу кругового обходу від PEP до OCSP-сервера та назад, з урахуванням змінної мережевої латентності.
2) Побудова графіків trade-off компромісу між затримкою та ризиком.

Приклад такого графіку (використані при побудові дані, були створені синтетично, для ілюстрації процесу) наведено на рисунку 1.4.

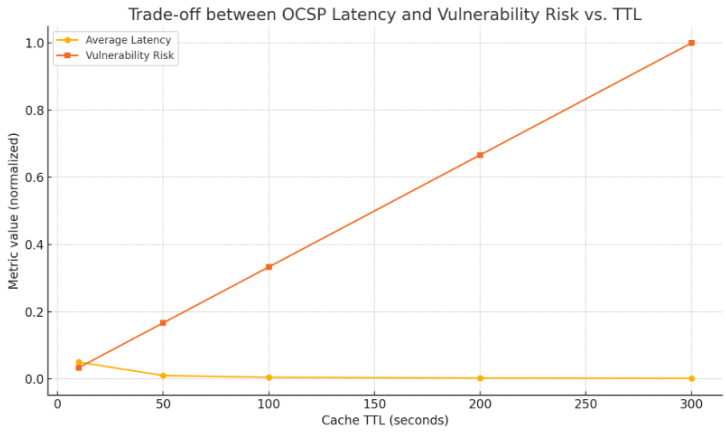

Рисунок 1.4 — Компроміс між між затримкою та ризиком для оптимального TTL

Точка перетину цих кривих, що випадає приблизно на 10 секундному інтервалі, є тією самою «золотою серединою», де ризик і затримка знаходяться у рівновазі. Саме в цьому діапазоні TTL мінімізує сумарні втрати: з одного боку, ризик пропустити відкликаний сертифікат, а з іншого — затримки на мережеві звернення. На основі такого графіка обґрунтовується вибір конкретного значення TTL — наприклад, 50 секунд — як оптимальний компроміс між безпекою й продуктивністю. При цьому важливо наголосити, що політика кешування не

повинна бути статичною. Під час підвищеної загрози рекомендується тимчасово зменшувати TTL для забезпечення максимальної свіжості даних, тоді як у години пікового навантаження можна збільшувати TTL, щоб знизити навантаження на сервера перевірки. На основі цих моделей можна розробити адаптивні стратегії: наприклад, для високопріоритетних ресурсів знижувати час кешування (TTL) та застосовувати приєднання відповіді OCSP на стороні сервера, а для менш критичних — використовувати Delta-CRL та локальні точки відмітки відкликаних сертифікатів. Такий підхід дозволяє зменшити загальну затримку перевірки і мінімізувати ризик використання скомпрометованих ключів без значного зростання навантаження на CA.

Взаємна автентифікація через mTLS піднімає довіру на новий рівень: при встановленні TLS-з'єднання і сервер, і клієнт обмінюються сертифікатами, що дозволяє не лише оберігати канал, а й гарантувати ідентичність обох сторін [10]. З однієї сторони, це значно знижує ризик "man-in-the-middle" атак, оскільки жодна із сторін не вірить «на перший запит», але з іншої — множить адміністративне навантаження: доведеться масштабувати зберігання приватних ключів на клієнтах і контролювати їхню безпеку, а також враховувати додатковий час на проведення двосторонньої перевірки при кожному повторному підключенні.

З погляду аналітики, сертифікат-орієнтовані методи мають найвищі показники криптографічної стійкості, проте потребують комплексного управління життєвим циклом сертифікатів та ретельного балансування між гнучкістю політик і продуктивністю мережі.

1.4.2 Токен-орієнтовані протоколи

У токен-орієнтованих протоколах автентифікація й авторизація відбуваються через передачу й верифікацію цифрових «талонів» (токенів), що дає змогу чітко розділити роль сервісів, які видають токени, й тих, які їх отримують.

OAuth 2.0 став де-факто стандартом для делегованого доступу: клієнт (наприклад, мобільний додаток) спочатку отримує від авторизованого серверу короткоживучий токен доступу із визначеним переліком дозволів (scopes) і потім

презентує його ресурсному серверу для доступу до API (див. рисунок 1.5). Такий підхід забезпечує гнучкість і масштабованість у розподілених системах, оскільки токен сам по собі містить підписані дані про користувача й права, дозволяючи будь-якому мікросервісу верифікувати його (публічним ключем) без звернень до центрального сховища сесій, — що зменшує зв'язність компонентів і дає змогу горизонтально масштабувати ресурси. Проте цей підхід ставить нові завдання: потрібно надійно зберігати токени, захищати канали їх видачі (PKCE, nonce) й мати механізми відкликання або автоматичного оновлення (refresh tokens) .

OpenID Connect (OIDC) накладає на OAuth 2.0 шар автентифікації: крім токену доступу видається ID - токен у форматі JWT, що містить набір стандартних і додаткових полів із інформацією про користувача та контекст його входу (час, спосіб автентифікації тощо) [11]. Завдяки кінцевій точці виявлення і динамічній реєстрації, OIDC спрощує інтеграцію «єдиного входу» (SSO) та полегшує централізоване управління сесіями й ролями в межах ZTA .

SAML (Security Assertion Markup Language) — це XML-базований стандарт федераційної автентифікації, де провайдер ідентичності формує SAML - твердження, підписану цифровим підписом, а провайдер сервісу на її основі вирішує, кому й до яких ресурсів відкривати доступ. Хоч SAML важчий за JSON-орієнтовані механізми OAuth/OIDC і вимагає складнішої обробки XML, він забезпечує потужні можливості: єдиний вихід, запит атрибутів та вирішення артефактів — корисні в класичних підприємницьких середовищах [12].

З погляду Zero Trust, токен-орієнтовані протоколи дозволяють централізувати всі запити на Policy Decision Point: кожен токен доступу чи SAML-твердження перевіряється на дійсність, термін важливості, набір атрибутів і ризикові політики перед тим, як Policy Enforcement Point відкриє канал доступу. Це дозволяє в реальному часі коригувати правила доступу (наприклад, вимагаючи повторної автентифікації, якщо токен було видано з високим рівнем ризику), а також зменшує залежність від мережевого периметра.

У підсумку, вибір між OAuth 2.0, OIDC і SAML у ZTA визначається вимогами середовища: там, де потрібна легковагова взаємодія мікросервісів і

мобільних клієнтів, зазвичай обирають OAuth 2.0 + OIDC, оскільки вся взаємодія будується на простих JSON/REST-запитах а токени JWT є компактними; у традиційних корпоративних інтеграціях із потужною інфраструктурою федерації та суворими вимогами до атрибутного обміну — SAML.

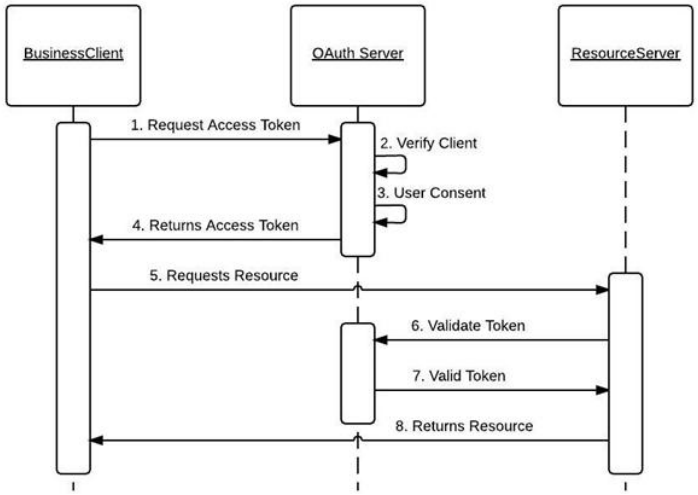

Рисунок 1.5 — Ілюстрація процесу автентифікації з токен-орієнтованим протоколом, на прикладі OAuth

1.4.3 Порівняння затримок сертифікат- і токен-орієнтованих протоколів автентифікації

Для дослідження було реалізовано експериментальне порівняння сертифікат-орієнтованої автентифікації (mTLS із X.509) та токен-орієнтованої автентифікації (OAuth 2.0 + JWT) у симульованому середовищі ZTA. Опис методології дослідження а також повний скрипт реалізації дослідження надано в Додатку А. Результатами імітації були занесені до таблиці 1.2.

Таблиця 1.2 — Порівняння затримок сертифікат- і токен-орієнтованих протоколів автентифікації

Метод автентифікації	Мінімальне значення затримки, с	Максимальне значення затримки, с	Середня затримка (μ), с	Стандартне відхилення (σ), с
OAuth 2.0 + JWT (RS256)	0.092	0.178	0.1278	0.0315
mTLS із X.509 сертифікатом	0.114	0.245	0.1883	0.0434

Оскільки у Zero Trust Architecture будь-який запит до ресурсу спочатку проходить через Policy Decision Point (PDP) і Policy Enforcement Point (PEP), затримка на етапі автентифікації безпосередньо впливає на загальний час відгуку системи. Отримані значення: час обробки запиту з використанням OAuth 2.0 + JWT становить приблизно 0,128 с із максимальною затримкою 0,178 с, тоді як mTLS із X.509 сертифікатом показав середню затримку близько 0,188 с із піковим значенням 0,245 с. Встановлення TLS-з'єднання з двосторонньою автентифікацією додає до загального часу виконання операції близько 0,06 с. Це означає, що OAuth обробляє запит на 47,4 % швидше, ніж mTLS.

У мікросервісній архітектурі, де один користувацький запит може породжувати десятки внутрішніх звернень, навіть невеликі додаткові затримки на кожному рівні автентифікації можуть накопичуватися до помітних величин. Якщо внутрішній API викликається 20 разів під час обробки одного запиту, надлишкові 0,05–0,1 с на кожному з'єднанні перетворюються на секунди загального очікування. У такій ситуації виправданим стає кешування результатів PDP-перевірки. Повний скрипт реалізації дослідження надано в Додатку А.

Втім, більші затримки обумовлюються більшими гарантіями безпеки. Двосторонність процесу автентифікації mTLS дає змогу не лише підтвердити, що клієнт має дійсний сертифікат, а й впевнитися в автентичності сервера, що створює взаємний довірчий контекст ще на етапі встановлення з'єднання. Будь-яка спроба підміни посередником або replay-атаки виявляється ще до початку передачі корисного навантаження, адже без приватного ключа, що відповідає сертифікату, зловмисник не зможе завершити TLS-handshake. У порівнянні з OAuth 2.0 + JWT, mTLS позбавлений вразливості «носія» — неможливо викрасти сертифікат із кешу або журналів без доступу до секретного ключа. Саме така архітектурна міцність виправдовує додаткові десяті долі секунди затримки: у критичних сценаріях, де будь-яка втрата конфіденційності або цілісності неприпустима, mTLS забезпечує рівень захисту, якого не може дати простий перенесений токен, а отже виправдовує витрати часу на рукостискання.

1.4.4 Моделі доступу й політик

У ZTA, вибір між моделями доступу й політик— це не просто технічне питання, а стратегічне рішення, яке визначає, наскільки гнучко та ефективно система зможе реагувати на мінливі умови безпеки. Всього виділяють 2 механізми: RBAC і ABAC

У традиційному RBAC (Role-Based Access Control) кожен користувач або пристрій отримує одну або кілька ролей, а права доступу до ресурсів визначаються на рівні ролей. Завдяки цьому модель RBAC добре масштабується в організаціях із чіткою ієрархією та стійкими функціональними підрозділами: наприклад, роль “менеджер з продажу” завжди матиме однаковий набір привілеїв незалежно від того, з якої точки мережі з’явився запит. Однак у Zero Trust, де довіра завжди має бути верифікована «на кожному кроці», така статичність перетворюється на вузьке місце. RBAC не враховує тих обставин, які роблять запит ризиковим: чи відрізняється IP-адреса від звичної для користувача, чи запит ініційовано з нової точки доступу, чи пристрій перебуває в належному стані безпеки. Унаслідок цього організації, що прагнуть відповідати принципам Zero Trust, часто стикаються з проблемою «role explosion» — необхідністю створювати безліч дрібних ролей, щоб частково врахувати контекст.

ABAC (Attribute-Based Access Control) пропонує інший підхід: рішення про доступ ухвалюється на основі атрибутів суб’єкта (роль, відділ, рівень довіри), атрибутів об’єкта (критичність даних, тип ресурсу) та атрибутів середовища (час дня, місцезнаходження, стан пристрою). Це відкриває можливість формулювати «тонкі» політики, наприклад: «дозволити доступ до бюджетних звітів тільки тим користувачам із роллю фінансового аналітика, якщо вони підключені через корпоративний VPN та пройшли багатофакторну автентифікацію протягом останніх 10 хвилин». Однак, із точки зору безпеки, набагато ефективніше, коли кожен атрибут користувача, пристрою чи середовища не просто бере участь у булевому виразі “дозволити/заборонити”, а оцінюється з точки зору впливу на загальну безпеку запиту. Замість традиційного підходу, де політика спрацьовує, коли всі умови виконані, тут кожному атрибуту присвоюється «вага» (коефіцієнт

важливості), а в кінці обчислення формується сукупний бал ризику. Якщо загальний ризик не перевищує заздалегідь визначений поріг, запит задовольняється; інакше – вимагається додаткова перевірка або відмовляється доступ .

З одного боку, ABAC набагато краще відповідає філософії Zero Trust: жоден запит не вважається надійним «за замовчуванням», і кожна сесія перевіряється з урахуванням найширшого контексту. З іншого боку, ця ж гнучкість породжує складнощі: необхідність централізованого каталогу атрибутів, потреба в надійному механізмі їхнього оновлення в реальному часі та ризик зниження продуктивності через складні політики з великою кількістю умов. Ще однією «точкою напруги» є управління конфліктами політик, коли атрибути суб’єкта і об’єкта створюють суперечливі рішення й потрібен механізм пріоритетів або медиатора політик .

Огляд практик [10] впровадження показує, що в реальних проєктах Zero Trust успішними є гібридні моделі, які поєднують RBAC та ABAC (див. рисунок 1.6), використовуючи першу для грубої фільтрації запитів та другу — для остаточного рішення із залученням контексту. Відтак, організація може спочатку відсортувати запити за ролями, а вже потім для кожного запиту виконати ABAC-аналіз із урахуванням геолокації, типу пристрою й рівня ризику. Такий «двоступеневий» підхід значно знижує навантаження на PDP (Policy Decision Point), оскільки RBAC-фільтр на PEP виконує дешеву O(1)-перевірку ролі і відсікає більшість запитів, а лише частка переходить до PDP для складного ABAC-аналізу O(n), що скорочує число важких обчислень на PDP і дозволяє зберегти керованість адміністративного процесу без втрати контекстності Zero Trust.

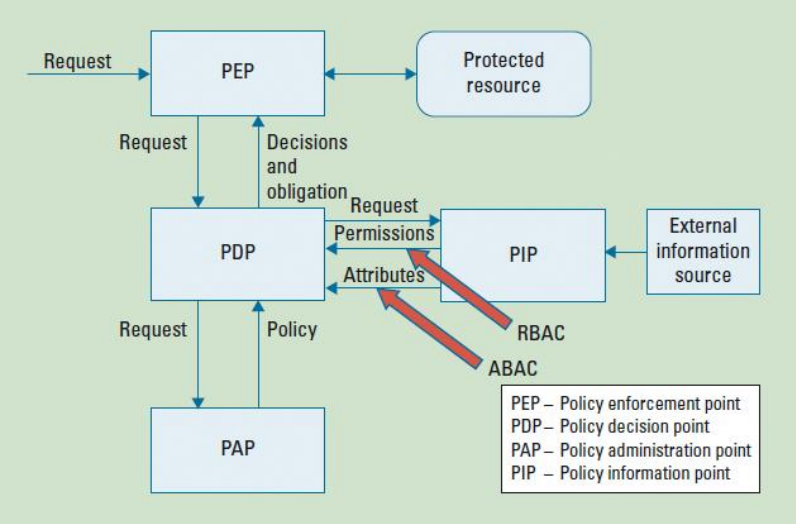

Рисунок 1.6 — Моделі доступу RBAC і ABAC у архітектурі нульової довіри

1.4.5 Адаптивна автентифікація

У Zero Trust архітектура адаптивної автентифікації на базі ризику перетворює процес перевірки ідентичності з одноразової події на безперервний цикл, у якому кожне нове середовище та зміна контексту можуть коригувати рівень довіри до суб'єкта. На відміну від традиційних моделей, де рішення «дозволити/заборонити» ґрунтується на статичному наборі атрибутів, тут система обчислює динамічний рівень довіри, який відображає поточний ризик запиту.

Перший крок — збір телеметрії: SIEM-дані, журнали мережевого трафіку, показники стану кінцевого приладу (онлайн-статус антивірусу, версія OS), розташування користувача, історія поведінкових шаблонів тощо. Кожен з цих сигналів зіставляється з внутрішніми моделями загроз і нормалізується в числові показники. Далі, на основі політик організації, кожному сигналу призначається вага, що відображає його відносну критичність: наприклад, запит від незнайомого IP може мати вагу 0,6, а відомий корпоративний VPN – 0,2. Підсумковий рівень довіри обчислюється як зважена сума цих показників (якщо він перевищує адаптивний поріг (який теж може змінюватися в залежності від часового інтервалу або загального рівня загроз), доступ дозволяється, інакше система ініціює повторну автентифікацію або автоматично продовжує збір даних для ретроспективного аналізу):

$$TS = \sum_{i=1}^{n} a_i w_i \quad (1.1)$$

де $a_i \in [0;1]$ — нормалізоване значення і-го сигналу (геолокація, час доби, стан пристрою тощо), а w_i — його вага в загальному рейтингу ($\sum w_i = 1$). Саме ця проста, але універсальна формула відображає гнучкість підходу: додавання нових сигналів чи коригування ваг дозволяє без масштабних змін у коді оновлювати політику під нові загрози. Важливим питанням є нормалізація: наприклад, геолокацію можна відобразити як

$$a_{geo} = \min\left(\frac{\text{dist(trusted,current)}}{D_{max}}, 1\right) \quad (1.2)$$

а стан пристрою — як частку встановлених патчів від загальної кількості.

Щоб зрозуміти як рахується рівень ризику, визначимо $D_{max} = 100$ км (може змінюватись в залежності від політики безпеки) — межа, після якої користувача вважається «поза зоною довіри», а відстань між поточними координатами користувача та довіреною геолокацією: (dist(trusted,current)) = 11.6 км.

Тоді a_{geo} буде дорівнювати: min(0.116 , 1) = 0.116. Нехай, крім геолокації, мається ще один сигнал — оцінка «стану пристрою» a_{dev} = 0.80, і політикою задано ваги: w_{geo}=0.2, w_{dev}=0.8. У підсумку, Загальний trust score обчислюється наступним чином: $TS = w_{geo}\, a_{geo} + w_{dev}\, a_{dev} = 0.2 \times 0.116 + 0.8 \times 0.80 = 0.0232 + 0.64 = 0.6632$. Якщо поріг довіри T_0 встановлено на рівні 0.6, то $TS = 0.6632 \geq T_0$, а отже доступ дозволено. Таким чином, навіть перебуваючи на відстані понад 10 км від офісу, користувач з «доброчесним» пристроєм отримує достатній trust score завдяки комбінуванню сигналів.

Точність калібрування ваг в таких системах є критичною: невиправдано висока чутливість до окремих сигналів призводить до «шумових» відмов легітимним користувачам, а низька – до неприйнятних ризиків. Також, пороги самі по собі повинні мати адаптивну природу: у період високої загрози (як-от після виявлення глобальної вразливості) організація повинна підвищити поріг, вимагаючи жорсткішої перевірки навіть для «базових» сценаріїв.

1.5 Автентифікація на основі поведінкових характеристик

1.5.1 Типи поведінкових біометричних даних

У базових схемах поведінкової біометрії виділяють три ключові модальності – клавіатурну динаміку, рухи мишки та взаємодію з сенсорним екраном. Усі вони спираються на видобуток часових і просторово-орієнтованих ознак, які потім подаються у вигляді векторів на вхід машинному класифікатору.

Головні ознаки у клавітурній динаміці – це час утримання клавіші (dwell time) і час між натисканнями (flight time). Якщо t_i^p і t_i^r – моменти натискання і відпускання і-тої клавіші, то:

$$d_i = t_i^r - t_i^p, \quad f_{i,i+1} = t_{i+1}^p - t_i^r [13] \tag{1.3}$$

Така пара $(d_i, f_{i,i+1})$ перетворюється на вектор ознак, часто доповнюваний статистиками (середнє, стандартне відхилення, percentiles). Система постійно аналізує час утримання клавіш і проміжки між їх натисканнями, коригуючи індекс довіри до користувача. При значущому відхиленні цих часових параметрів від еталонного профілю індекс довіри падає і ініціюється повторна автентифікація.

При таких умовах постає питання: Чи не заважають такі постійні обчислення користувачеві при роботі? Щоб дати відповідь, на основі формули (1.3) було реалізовано клієнтський агент (див. рисунок 1.7) на Python із GUI на основі tkinter. Агент вимірює ключові характеристики динаміки введення: час утримання та інтервал між відпусканням однієї і натисканням наступної клавіші.

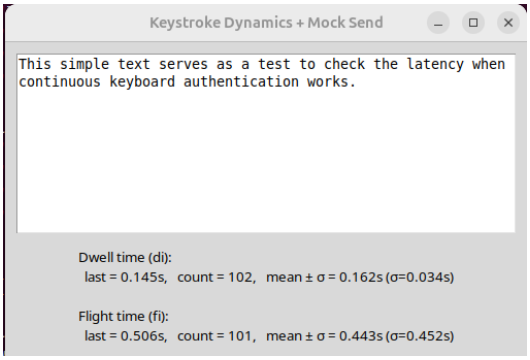

Рисунок 1.7 — Додаток для тестування затримок при обрахуванні клавіатурної динаміки

Як результат, у надрукованому тексті, який містить 102 символи, програма зафіксувала 102 випадки вимірювання часу утримання клавіш (dwell time) та 101 перехід між клавішами (flight time). Останнє значення dwell time становило 0,145 с, тоді як середнє по всій вибірці дорівнювало 0,162 с із відхиленням близько 0,034 с. Для flight time останній інтервал між відпусканням однієї клавіші і натисканням наступної склав 0,506 с; середній показник по 101 вимірюванню — 0,443 с із великою варіативністю ($\sigma \approx 0{,}452$ с). Така комбінація кількісних характеристик – кількість замірів, останнє значення, середнє та стандартне відхилення – дає повне уявлення про темп і стабільність набору тексту. При цьому затримки від обчислень не були помітними навіть для просунотого користувача ПК, який має швидкість друку близько 270 символів за хвилину (При середньому показнику по всьому світу близька 200 символів за хвилину [14]). Повний код та опис програми знаходиться в Додатку Б.

За аналогією з клавіатурою, трекінг курсора дозволяє збирати: траєкторію шляху (x(t), y(t)), миттєву швидкість v(t) = $\sqrt{\dot{x}^2 + \dot{y}^2}$, прискорення a(t)= $\dot{v}(t)$, час паузи перед кліком (click dwell). Від шуму сигнал нормалізують, виділяють фази руху та зупинки, а потім обчислюють ознаки типу середньої швидкості, ентропії траєкторії, геометричних характеристик (кривизна, довжина шляху). Ці ознаки зручно подавати у вигляді єдиного вектора для класифікатора.

У мобільних пристроях збирають набір сенсорних сигналів: координати дотику (x_i,y_i) і часова мітка t_i, тиск p_i та площа контакту A_i, орієнтацію пальця (кут = φ_i). Додатково можна отримати поведінкові ознаки кожного свайпу, серед яких: часові параметри (тривалість жесту, час між жестами), геометричні характеристики (координати початку й завершення, пряма відстань між кінцями, довжина траєкторії, середня та кутова спрямованість – mean resultant length, напрямок руху, прапор «вгору/вниз/ліворуч/праворуч»), статистики швидкості та прискорення, відхилення від прямолінійного руху (максимальне та відсоткове відхилення від прямолінійної лінії), а також середні значення напрямку й швидкості [15].

Усі три модальності мають спільний конвеєр обробки:

1) Збір сирих сигналів (натискання клавіш, координати мишки/тачпада, сенсорні події).
2) Нормалізація й фільтрація (видалення артефактів, вирівнювання частоти дискретизації).
3) Видобуток ознак (часові, просторові, статистичні).
4) Побудова векторів та класифікація (логістична регресія, SVM, нейронні мережі).

У голосових та жестових ознаках ключовим є перетворення безперервного сигналу в набір стабільних ознак, здатних однозначно ідентифікувати користувача та виявити спроби підробки.

У випадку голосових патернів (voice biometrics) система спочатку виконує цифрову обробку аудіосигналу: підсилення, видалення шуму та розбиття на фрейми тривалістю 20–40 мс із 50 % перекриттям [16]. З кожного фрейма обчислюють набір акустичних ознак — найчастіше мел-частотні кепстральні коефіцієнти (MFCC), енергетичні й спектральні характеристики (spectral centroid, bandwidth), а також параметри prosody (тонова висота, енергія в часі). Отримані вектори ознак агрегують статистично (середнє, дисперсія) або через моделі типу i-vector / x-vector, після чого в навчальній фазі будують класифікатор) для кожного користувача.

Рухові патерни охоплюють великий діапазон характеристик — від ходи (gait) до довільних рухів руками. У мобільних пристроях це найчастіше дані з акселерометра та гіроскопа, які записують у вигляді тривимірних часових рядів $\{x_t,y_t,z_t\}$. Зі сировини видобувають ознаки частотної області (FFT-коефіцієнти, вейвлет-коефіцієнти),енергетичні а також часові характеристики: середньої швидкості руху, прискорення, ентропії траєкторії або кривизни шляху. Важливою до огляду є енергетична ознака. Енергія руху прямо відображає наскільки сильні зміни відбуваються в положенні пристрою. Вона зростає при стрибках, бігу, різких поворотах — і залишається низькою при спокої або повільній ході. А це у свою чергу добре інтерпретується в механізм автентифікації, оскільки різні

користувачі мають унікальні шаблони фізичної активності — амплітуду, частоту та динаміку рухів. Енергетична ознака фіксує ці індивідуальні особливості без потреби знати точний напрям руху чи орієнтацію пристрою. Таким чином, вона дозволяє надійно розрізняти користувачів навіть за простими жестами або під час звичайної ходьби. Ядром для подальшого дослідження є формула енергії руху:

$$E(t) = a_x^2(t) + a_y^2(t) + a_x^2(t) \quad (1.4)$$

Щоб нівелювати вплив прискорення вільного падіння до формули (1.4) застовується проста фільтрація із виділеням лінійного прискорення

$$E(t) = l_x^2(t) + l_y^2(t) + l_x^2(t) \quad (1.5)$$

Для збору рухових даних було розроблено мобільний додаток на платформі Android з використанням мови Kotlin. Основною метою додатка є зчитування даних акселерометра в реальному часі та розрахунок енергетичної ознаки руху.

При досліджені, в якому брало участь дві людини, було отримано графіки енергії руху, показані на рисунку 1.8.

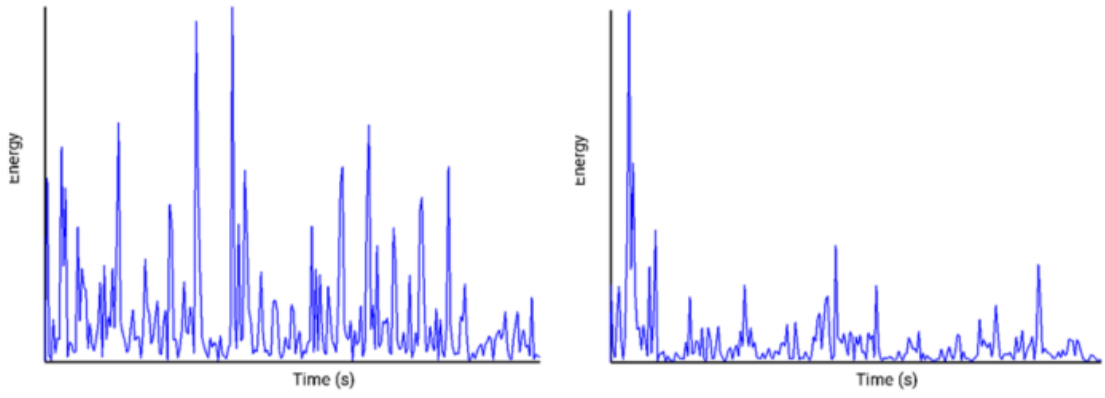

Рисунок 1.8 — Графіки залежності енергії рухів від часу у різних людей протягом 25 секунд.

Навіть візуального огляду двох графіків достатньо щоб зробити висновок, що це дві різні людини. У першому випадку спостерігаються численні високі піки

енергії з різкою зміною амплітуди: це свідчить про більш експресивний, фрагментарний стиль рухів, швидкі переходи між розділами активності та загалом вищий рівень моторної збудженості. У другому випадку один великий початковий сплеск (може детектуватись як розгін або шум), після якого енергетичні коливання значно спокійніші: амплітуда піків знижується, а інтервали між ними рівномірніші. Така картина характерна для людини з плавним, контрольованим темпом роботи, без різких ривків. Ці системні відмінності в частоті, амплітуді й профілі енергетичних коливань є однозначною ознакою різних біомоторних патернів двох осіб. Опис додатка, вибірка отриманих значень по кожній особі із дослідження, а також повний код реалізації програми надано в додатку В

Для класифікації жестів на кроці автентифікації застосовують схеми на основі HMM або DTW (для витримки варіацій у темпі руху) і сучасні нейромережі, які враховують часові залежності. Особливо вдалими виявилися підходи, що комбінують кілька моделей — кореляційний аналіз у часовій області разом із спектральними ознаками — для підвищення стійкості до спроб імітації та шуму сенсорів [17]. Такий технічний конвеєр обробки — від фреймування та нормалізації сигналів до побудови векторних ембедингів і моделей на їх основі — дозволяє досягати високої точності і водночас захищатися від replay- та mimic-атак, забезпечуючи потрібний рівень безпеки в Zero Trust середовищах.

1.5.2 Оцінка поведінкових методів

У поведінковій автентифікації одним із головних показників є баланс між рівнем безпеки (FAR – false acceptance rate) та зручністю (FRR – false rejection rate). Для клавіатурної динаміки на основі dwell/flight-time дослідження показують: Lin (1997)[13] : FAR ≈ 1.11 % при FRR = 0 %; Bergadano & Gunetti (2002) [18]: FAR = 0 % при FRR ≈ 2.3 %; Gunetti & Picardi (2005) [19]: FAR ≈ 0.005 % при FRR ≈ 5.0 %. При цьому Equal Error Rate (EER) для суто часового підходу варіюється від 1.15 % до 4.28 % [12]. Тоді як для комбінованих модальностей (клавіатура + сенсори IMU) можна досягти EER ≈ 2–3 % за правильно підібраних параметрів і класифікаторів. У голосових системах на базі MFCC та x-vector

типові показники FAR ≈ 5 % і FRR ≈ 2–3 % (залежно від якості мікрофону та наявності шумоподавлення). Touchalytics демонструє, що на основі сенсорних ознак можна досягти надзвичайно низьких рівнів помилок 0 % всередині сесії, 2–3 % між сесіями та < 4 % через тиждень після реєстрації [20][21][15] Для досягнення прийнятного рівня безпеки необхідно ретельно калібрувати пороги моделей і застосовувати мультимодальні підходи – інакше підвищення точності в одній модальності може збільшити ризик помилкових відмов у іншій — надто жорсткий поріг, встановлений для зниження FAR у клавіатурній динаміці, потенційно спроможний виключити частину, типових для сенсорних даних, варіацій жестів, що призведе до зростання FRR у touch- або IMU-модальностях через невідповідність критеріїв класифікації.

Збір і обробка поведінкових даних ставлять питання конфіденційності: сирі часові характеристики натискання клавіш, траєкторії курсора, фрагменти аудіо можуть містити непрямі персональні ознаки (наприклад, стиль письма, манеру говоріння). Щоб відповідати вимогам GDPR або місцевих законів про захист даних, системи повинні мінімізувати обсяг передаваних даних (наприклад, агрегувати та анонімізувати на клієнтському пристрої), шифрувати канали передачі та обмежувати зберігання лише до необхідного мінімуму.

Нарешті, UX-аспект у поведінковій автентифікації — це компроміс між зручністю та навантаженням системи. Часті запити повторної верифікації або висока чутливість моделей (низький поріг детектування) можуть дратувати користувачів і призводити до відмов у роботі, тоді як надто м’які налаштування знижують довіру до безпеки. Оптимальний UX досягається при комбінуванні поведінкових параметрів із фоновими перевірками (наприклад, запит додаткового фактора лише при низькому trust score) та застосуванні адаптивних порогів, що враховують історію успішних/невдалих сесій конкретного користувача

1.5.3 Безперервна автентифікація

У безперервній автентифікації ключовим є перетворення пасивного моніторингу на активний механізм оцінки довіри під час усього часу сесії. Замість

одноразової перевірки під час входу система збирає потік поведінкових даних $B=\{b_1,b_2,\ldots,b_t\}$, де кожен елемент може бути вектором ознак клавіатурної динаміки, миш-руху або інших поведінкових показників.

На кожному кроці t з останнього вікна розміру W формується матриця ознак

$$F_t = \begin{bmatrix} f_{t-W+1,1} & f_{t-W+1,2} & \cdots & f_{t-W+1,m} \\ \vdots & \vdots & \ddots & \vdots \\ f_{t,1} & f_{t,1} & \cdots & f_{t,m} \end{bmatrix} \quad (1.6)$$

де m — кількість вибраних параметрів (наприклад, dwell time, flight time, швидкість миші). За допомогою методу ковзного вікна (sliding window) або експоненційного згладжування (EWMA) обчислюється агрегований профіль

$$P_t=\alpha\, agg(F_t)+(1-\alpha)\, P_{t-1}, \quad 0<\alpha\leq 1, \quad (1.7)$$

який порівнюється з початковим референтним шаблоном P_0 (α – коеф. чутливості). Ступінь відповідності оцінюють метрикою схожості $d(P_t,P_0)$. Якщо $d(P_t,P_0) > \theta$, де θ — заздалегідь обраний поріг, детектується аномалія, і система повинна одразу ініціювати повторну автентифікацію або знижувати рівень довіри. Результати аналітичної роботи із формулою 1.7 наведено у додатку Г

З точки зору продуктивності, обробка кожного вікна потребує O(W×m) операцій для агрегації і O(m) для обчислення подібності — у сумі O(Wm). Щоб знизити навантаження, на рівні PEP можна виконувати попередню фільтрацію аномалій, а глибоку перевірку делегувати централізованому PDP.

У Zero Trust безперервна автентифікація повинна не лише виявляти аномалії, а й своєчасно ініціювати повторну перевірку (reauthentication), щойно якісь умови виходять за межі допустимого. Основні «тригери» можна розподілити на чотири категорії:

1) Перевищення порога поведінкової аномалії. Як тільки метрика схожості $d(P_t,P_0)$ падає нижче попередньо встановленого рівня θ, PDP одразу викликає повторну автентифікацію. Це найпростіша та найчутливіша умова запуску процедури повторної перевірки, яка захищає сесію від підміни протягом її активності.

2) Таймаут сесії та довжина активності. Навіть у відсутності явних аномалій, багато стандартів ZTA вимагають автоматичного завершення сессії за інтервалом T_{max}. Після його спливу або коли загальна тривалість роботи перевищує баланс безпеки/зручності, система ініціює повну повторну автентифікацію .[2]
3) Доступ до підвищено-ризикових ресурсів. Кожному ресурсу r присвоюють рівень чутливості S(r). Якщо користувач із поточним рівнем довіри (trust score, TS) намагається отримати доступ до ресурсу з $S(r)>S_{crit}$, навіть коли $TS \geq T_0$, система вимагає додаткового фактора (MFA) або повної реаутентифікації .
4) Зміни системного контексту. Зміна мережевого середовища (як-от з VPN до Internet або навпаки), оновлення статусу антивірусу чи патчів на клієнті, вихід із корпоративної LAN – усі ці події позначаються як різке зростання ризику й автоматично викликають процедуру reauth

1.6 Автентифікація пристроїв у IoT та критичній інфраструктурі

1.6.1 Виклики ресурсозалежних пристроїв

У середовищі IoT та критичної інфраструктури пристрої часто працюють на слабких 32- або навіть 8-бітних мікроконтролерах із лімітованим обсягом оперативної пам’яті (кілька десятків–сотень кілобайт), флеш-пам’яті (кілька мегабайт) і частотою CPU у кілька десятків мегагерц. Кожна криптографічна операція — асиметричний обмін ключами чи цифровий підпис — потребує від сотень до тисяч тактів, що на енергоефективних MCU може означати секунди очікування та суттєве зростання споживання струму під час алгоритмів рукостискань. В умовах обмеженого енергобюджету (батарейні вузли із інтервалами пробудження у хвилинах чи годинах) це може призвести до скорочення часу життя пристрою вдвічі й більше .

Паралельно із цим IoT-підмережі використовують низькошвидкісні протоколи: 6LoWPAN (6Lo adaptation over IEEE 802.15.4) та NB-IoT (Narrowband IoT) які обмежують MTU ≈ 128 байт та пропускну здатність у десятки–сотні

кілобіт за секунду, часто з суворими обмеженнями робочого циклу радіочастоти. У такому каналі фрагментація довгих TLS-фреймів чи обмін великими JWT-токенами стає неприпустимим накладним витратом, що збільшує ймовірність перевищення часу очікування та втрати пакетів. Додатково висока затримка (500 – 1000 мс на NB-IoT) й низька пропускна здатність ускладнюють реалізацію синхронних протоколів mutual TLS чи стандартного OAuth 2.0 flow без радикального спрощення handshake або застосування «легковагових» протоколів.

Ці обмеження диктують ряд стратегій: Використання симетричних схем (PSK-DTLS, легкі блочні шифри), перенесення складних обчислень у «хмару» чи шлюзи; Адаптивна компресія та фрагментація (6LoWPAN Header Compression, CoAP Observe) для зменшення кількості радіоповідомлень; Оптимізація duty-cycle — встановлення довших інтервалів між сесіями, але з передбачуваними «вікнами» автентифікації.

1.6.2 Легковагові протоколи

У умовах гранично обмежених ресурсів легковагові автентифікаційні схеми покликані мінімізувати число й складність криптографічних операцій, зберігаючи при цьому властивості цілісності та свіжості повідомлень. Однією з найпопулярніших є протокол RMAC (Lightweight Authentication Protocol for Highly Constrained IoT Devices) [22]. Опис роботи алгоритму знаходиться у додатку Д.

Переваги методу: Кількість “раундів” Even–Mansour = 2. Для одного MAC виконується 4 виклики RBOX і 4 XOR-операції, що на MCU з тактовою частотою ≈ 16 MHz дає приблизно 500 мс; Низьке споживання енергії завдяки мінімуму простих операцій (XOR і зсуви); Відсутність складних множень або великих таблиць S-боксів. [22]

Одним із перспективних напрямів у побудові нульової довіри між пристроями є використання характеристик фізичного середовища — зокрема, Channel State Information (CSI). Підхід LCDA (Lightweight Continuous Device-to-Device Authentication) показаний у [23] пропонує саме таку модель. Його

особливість полягає в тому, що ключі для автентифікації та шифрування формуються не на основі попередньо встановлених матеріалів чи PKI, а з фактичного стану бездротового каналу. Це дозволяє обійти потребу у зовнішньому довіреному центрі чи постійному збереженні секретів на пристрої. Повний опис алгоритму надано в додатку Е.

Підхід LCDA є не лише технічно обґрунтованим, а й ідеологічно близьким до концепції нульової довіри. Він відповідає кільком ключовим вимогам Zero Trust: відсутність постійної довіри до будь-якого вузла; періодична або безперервна перевірка справжності; незалежність від централізованого авторитету. Особливо цінним є те, що автентифікація базується на середовищі передачі — тобто сам факт спільного розташування в просторі та одночасного зчитування фізичних параметрів каналу стає основою довіри. Це створює новий рівень складності для зловмисника: для успішного зламу йому потрібно не лише перехопити ключі, а буквально фізично відтворити той самий бездротовий контекст, що в реальних умовах майже неможливо.

Інші легкі протоколи (PSK DTLS, або CoAP OSCORE) також прагнуть зменшити кількість RTT і асиметричних операцій. Стандартний DTLS 1.2/1.3 для забезпечення безпеки CoAP-з’єднань часто занадто «важкий» для NB-IoT / 6LoWPAN. Щоб зменшити затримку і обсяг повідомлень, у DTLS-мінімізованих реалізаціях (рисунок 1.9) застосовують: Алгоритми PSK-only замість X.509, що усуває потребу в асиметричних операціях і сертифікатах; 1-RTT handshake (або навіть 0-RTT у DTLS 1.3) із використанням сеансових квитків (session tickets) та коротких рукостискань, що скорочують кількість повідомлень, від клієнта до сервера і навпаки, до одного кругового обходу; Cookie exchange на рівні UDP, щоб усунути DoS-вектори й підтвердити справжність клієнта до початку важкої криптографії; Стиснення заголовків (TLS record header compression) та використання компактних PSK-ідентифікаторів замість повних SessionID.

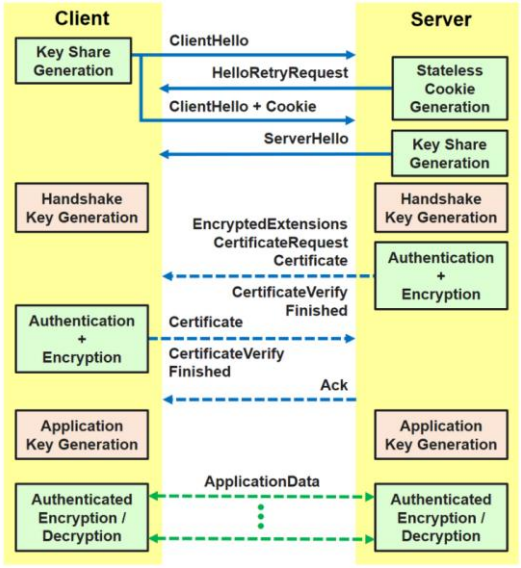

Рисунок 1.9 — Схема мінімізованого DTLS-handshake[24]

Окрім згаданих вище мінімізованих реалізацій DTLS, у [25] також розглядається протокол EDHOC (Ephemeral Diffie-Hellman Over COSE), який дозволяє значно скоротити обсяг handshake-повідомлень, з ≈ 505 байт (DTLS 1.3 з PSK) до 101 байту (EDHOC із статичними DH-ключами) — це 80%, або в 5 разів менше переданих даних. Затримка передачі handshake-повідомлень пропорційно знижується на той самий коефіцієнт (≈ 80 %), завдяки меншій кількості байт у «повітрі» та уникненню фрагментації .

Застосування EDHOC замість стандартного DTLS 1.3 із PSK дозволяє істотно скоротити обсяг трафіку при встановленні сесії та знизити затримки — критично важливо для NB-IoT / 6LoWPAN пристроїв із обмеженою пропускною здатністю та енергетичним ресурсом.

1.6.3 Апаратні корені довіри

Апаратні корені довіри (Hardware Roots of Trust, HRoT) закладають первинну «точку довіри» в Zero Trust Architecture: саме їм належить роль невразливого сховища для ключів, апаратного контролю цілісності та запуску захищеного коду. Без надійної HRoT будь-яке програмне рішення може бути скомпрометоване через фізичний або side-channel атак. У додатку Ж розглядається чотири основні реалізації—TPM, Secure Element, ARM TrustZone та

HSM—через призму їх переваг, обмежень і практичної придатності в IoT і критичній інфраструктурі. Порівняння реалізацій HRoT надано у таблиці 1.3.

Таблиця 1.3 — Порівняння реалізацій HRoT

Параметр	TPM	SE	TrustZone	HSM
Призначення	Захист платформи	Апаратна ідентичність	Безпечне виконання	Масова обробка криптоопераций
Продуктивність	Середня	Низька	Висока (TEE - залежна)	Дуже висока
Вартість	Низька - середня	низька	Включено в SoC	висока
Захист ключів	Високий	Дуже високий	Середній-високий	Дуже високий

1.6.4 Верифікація ідентичності на рівні чіпа (Chip-Rooted Trust)

Chip-rooted trust — це підхід, за якого довіра до пристрою формується безпосередньо з його апаратного компонента, без потреби у зовнішньому довіреному джерелі. Основна ідея полягає в тому, що ідентичність пристрою не призначається, а виводиться з фізичних характеристик чіпа, які не можуть бути скопійовані або відтворені — навіть виробником.

Найбільш характерний метод реалізації chip-rooted trust — це використання фізично неклонованих функцій (PUF), які на основі мікроскопічних варіацій у кремнієвій структурі мікросхеми генерують унікальну й повторювану відповідь (challenge-response pair) [10]. Наприклад, два однакові чіпи з тієї самої партії матимуть різні затримки в лініях передачі сигналу, які можна виміряти для генерації криптографічного ключа:

$$Key_{PUF}=PUF_{device}(challenge) \quad (1.8)$$

Оскільки цей ключ ніколи не зберігається в енергонезалежній пам’яті, він не може бути вилучений навіть при повному доступі до пристрою.

PUF відповіді використовують як корінь довіри при автентифікації, пристрій генерує підпис за ключем, виведеним із PUF, і сервер перевіряє його автентичність, або як джерело криптографічних ключів: у поєднанні з fuzzy extractors можна стабільно відновлювати ключі навіть за наявності варіацій температури або напруги.

Переваги chip-rooted trust: Неможливість клонування (ідентичність залежить від фізики конкретного пристрою; Захист ключа без сховища: приватний ключ не зберігається у флеш або EEPROM, і зникає при вимкненні живлення; Енергетична ефективність: обчислення PUF не потребують складної криптографії (переважно затримки сигналу або стан SRAM при запуску).

Висновки за 1 розділ

1) Аналіз основних принципів автентифікації в рамках архітектури нульової довіри дозволив визначити ключові особливості цього підходу та його відмінності від традиційних моделей захисту. Архітектура Zero Trust принципово змінює парадигму безпеки, заперечуючи автоматичну довіру до користувачів та пристроїв, навіть якщо вони знаходяться всередині корпоративної мережі. Основний принцип "ніколи не довіряй, завжди перевіряй" забезпечує постійний контроль доступу та динамічне прийняття рішень на основі контексту запитів.

2) Значну роль у реалізації Zero Trust відіграють механізми автентифікації користувачів, оскільки традиційна одноразова перевірка під час входу в систему не гарантує належного рівня безпеки. Багатофакторна, контекстуальна та безперервна автентифікація дозволяють ефективно ідентифікувати користувачів та аналізувати їхню поведінку для запобігання потенційним загрозам. Крім того, у контексті Zero Trust важливим є не лише контроль особи, а й забезпечення безпеки пристроїв, що підключаються до мережі. Використання криптографічних сертифікатів, модулів TPM та довірених середовищ виконання (TEE) значно ускладнює компрометацію інфраструктури та дозволяє мінімізувати ризики атак.

3) Впровадження Zero Trust вимагає модернізацію автентифікаційних механізмів, так і розширення моделей контролю доступу. Незважаючи на складність реалізації, цей підхід дозволяє суттєво підвищити рівень інформаційної безпеки організацій, зменшити ризики внутрішніх та зовнішніх атак і забезпечити адаптивний захист у сучасних динамічних кіберсередовищах.

4) Проведений аналіз сучасних методів автентифікації в архітектурі нульової довіри показує, що жоден окремий підхід не може задовольнити всіх вимог до безпеки та продуктивності в гібридних середовищах

5) Сертифікаторієнтовані протоколи гарантують найвищу криптографічну стійкість, проте потребують складного управління життєвим циклом сертифікатів і викликають надлишкові затримки через часті запити до CRL/OCSP, — токенорієнтовані механізми забезпечують простоту інтеграції й масштабованість завдяки компактним JWT-токенам і централізованій перевірці, але вразливі до відкликань і вимагають додаткових гарантій цілісності

6) Моделі доступу RBAC та ABAC у поєднанні найкраще поєднують простоту адміністрування з контекстною оцінкою запитів, оскільки RBAC дозволяє масштабувати базову фільтрацію, а ABAC—гнучко реагувати на мінливий ризик

7) Адаптивна й безперервна автентифікація формують беззупинний цикл оцінки довіри зваженим ризик-скорингом, що знижує кількість зайвих перевірок і підвищує рівень захисту, — поведінкові біометричні методи (клавіатурна динаміка, курсор, сенсори, голосові та жестові патерни) додають додаткові сигнали для підвищення точності моделі довіри без прямої участі користувача

8) Для ресурсозалежних IoT-пристроїв легковагові протоколи (RMAC, LCDA, PSK DTLS, EDHOC тощо) дозволяють зберегти властивості цілісності й свіжості даних, мінімізуючи обчислювальні та енергетичні витрати.

2 ПРОБЛЕМИ ТА ВИКЛИКИ РЕАЛІЗАЦІЇ АВТЕНТИФІКАЦІЇ У ZTA

2.1 Проблеми інтеграції існуючих систем

2.1.1 Несумісність з традиційними LDAP/AD-інфраструктурами

У класичних корпоративних середовищах довгий час ключовим елементом управління ідентичностями залишався (і по-сьогодні залишається) Active Directory (AD) або аналогічні рішення на базі LDAP-серверів. Саме через цю уніфіковану «приймальню» для облікових записів побудовані більшість політик доступу, групових налаштувань та навіть процесів видачі сертифікатів. Проте з підходом нульової довіри така архітектура виявляє кілька принципових обмежень. По-перше, LDAP/AD розраховані на модель «один раз і назавжди» – користувачі проходять автентифікацію при вході в домен і далі використовують отримані токени чи TGT (Ticket Granting Ticket) для доступу до ресурсів без повторних перевірок. AD за замовчуванням не надає механізмів для передачі контекстних атрибутів у процесі автентифікації, що унеможливлює побудову дійсно адаптивної політики доступу . Додатковою перепоною є жорстка прив’язка фіксованих схем LDAP до атрибутів користувача: групи, OU (організаційні підрозділи), властивості об’єктів тощо. У реальних впровадженнях ZTA політикам потрібен доступ до зовнішніх джерел телеметрії, щоб ухвалювати рішення в режимі реального часу . Спроби розширити AD шляхом додавання довільних атрибутів чи розробки надбудов закономірно призводять до ускладнення адміністрування, адже кожна зміна схеми вимагає синхронізації на всіх контролерах домену, суворого тестування сумісності, оновлення політик синхронізації та підвищує ризик виникнення неконсистентних «острівців довіри». Це значно збільшує трудовитрати, уповільнює випуск змін і суперечить ідеї єдиної, центричної системи перевірок. Ще одна важлива складова — це розрив між традиційними протоколами Kerberos/NTLM та новими механізмами mTLS і токенорієнтованими підходами. AD-інфраструктура спроєктована для внутрішніх мереж із довіреним периметром, тоді як ZTA передбачає мінімізацію такого

периметра та мікросегментування. Через це багато підприємств змушені реалізовувати складні гібридні архітектури, де AD синхронізується з окремим PKI-рішенням або зовнішнім Identity Provider, що призводить до затримок у процесі автентифікації та ризику розбалансування конфігурацій [26].

2.1.2 Різноманіття стандартів та протоколів

Різноманіття стандартів і протоколів вимагає від впровадника Zero Trust одночасно діяти у різних «площинах» автентифікації, що спричинює низку принципових проблем. Кожен протокол приносить власні формати повідомлень і моделі довіри, які не узгоджуються між собою самостійно. Так, SAML-твердження в XML-структурі з цифровим підписом вимагають складного обміну метаданими й масштабного «парсингу», що сповільнює «рантайм-обробку» в розподілених середовищах та створює додаткове навантаження на служби федерації. Навпаки, OAuth 2.0 + OIDC оперує легковаговими JSON-веб-токенами, але цей простір ускладнює підтримка механізмів захисту — PKCE, nonce, окреме відкликання та оновлення токенів, — які необхідно реалізувати й перевірити на всіх клієнтських і серверних компонентах .

Управління життєвим циклом сертифікатів та приватних ключів усіх кінцевих пристроїв у mTLS потребує надійної підтримки PKI та інтеграції з OCSP/CRL, крім того, двосторонній TLS-рукостискання збільшує затримки при кожному з’єднанні. Коли в середовищі співіснують і токен-орієнтовані, і сертифікаторієнтовані процесси, адміністратори змушені одночасно налаштовувати політику TTL для JWT-токенів та кеш-політику OCSP, що нерідко призводить до непередбачених «вікон вразливості» чи надмірного мережевого навантаження.

Через різницю в семантиці атрибутів від кожного протоколу ускладнюється централізований збір інформації для PDP. Наприклад, SAML-атрибути часто позначаються складними URN і можуть містити довільні XML-вузли, тоді як в OAuth/OIDC — це набір стандартних полів у JWT; mTLS-сценарії взагалі спираються на TLS-контекст, а не на передані користувацькі атрибути. Узгодити

ці різнорідні формати в єдиній моделі політик означає розробити ще один шар трансформації й нормалізації даних, що підвищує ризик помилок і створює «оточення» для непередбачених конфліктів при оновленні версій протоколів чи розширенні атрибутного каталогу.

Неможливість одразу уніфікувати ці моделі вимушує організації вкладати значні ресурси в розробку й підтримку «мостів» між ними, що контрастує з філософією мінімальної довіри та простоти розгортання.

2.2 Виклики для безперервної автентифікації

2.2.1 Контекстуальна перевірка та збереження приватності

Практична реалізація контекстуальної автентифікації стикається з низкою суттєвих викликів, насамперед із забезпеченням конфіденційності оброблюваних даних. По-перше, збір поведінкових та контекстних атрибутів (натискання клавіш, траєкторія руху миші, геолокація, метадані голосу тощо) для розрахунку поточного рівня ризику може розкривати непрямі персональні ознаки користувача. Як показують дослідження з анонімізації поведінкових даних, сирі дані часто містять унікальні біометричні патерни, здатні відновити ідентичність особи. По-друге, перенесення етапу екстракції ознак кінцевий пристрій мережі, як згадувалось у розділі 2, знижує ризик витоку чутливих даних, але створює нове навантаження на ресурси пристрою й вимагає захищеного середовища виконання (TEE, Secure Enclave). Без відповідної ізоляції коди аналізу можуть бути змінені або обійдені зловмисником, що спричинить маніпуляцію «trust score» і підрив довіри до всього процесу контекстуальної перевірки . По-третє, централізоване ведення журналів та моніторинг трафіку, необхідні для ретроспективного аналізу поведінки та виявлення аномалій, підпадають під дію політик приватності та потребують формалізованого підходу до згод користувачів і інформування про обсяг та мету збору даних. Згідно з рекомендаціями NIST Privacy Framework, організації повинні оцінити ризики перехоплення й аналізу мережевого трафіку, розробити процедури інформування користувачів і механізми отримання їхньої згоди [27].

2.2.2 Точність та адаптивність поведінкової біометрії

При реальному розгортанні поведінкової біометрії в архітектурі нульової довіри стикаються переважно з двома класами складнощів: дрейфом поведінкових патернів та гетерогенністю кінцевих точок Насамперед, поведінка користувача за своєю природою змінюється з часом: нові пристрої, оновлення ПЗ, стрес чи проблеми зі здоров'ям-зумовлені відхилення призводять до так званого «дрейфу концепції». Система, навчена на даних попереднього періоду, раптово фіксує зростання помилкових відмов (FRR) без реальних атак.

Друге ускладнення криється в гетерогенності пристроїв: мобільні телефони, ноутбуки й тонкі клієнти мають різні датчики та затримки введення. Навіть ідентичний користувач на різних пристроях генерує статистично відмінні вектори ознак. Без адаптивних методів доменної адаптації чи федеративного навчання модель або надто чутлива (зростають FRR), або обережна (зростають FAR). Додатково, відмінності в налаштуваннях ОС, драйверах і навіть розмірах клавіатури вносять «шуми», які без належної нормалізації перетворюються на хибні тривоги .

Для перевірки вищезгаданих двох суджень доцільно знову провести експеримент, пов'язаний із клавіатурною динамікою, який описаний у п.2.2.1 розділу 2 роботи, але із іншими умовами. Відтак для дослідження та сама людина буде друкувати той самий текст, але в інший день, опівдні (перший експеримент проводився ввечері, коли втома за день може корелювати результати) та із використанням іншого пристрою введення (більш швидкого у реагуванні, оскільки засноване на технології світлової передачі (Optical switch)). Результати дослідження показані на рисунку 2.1.

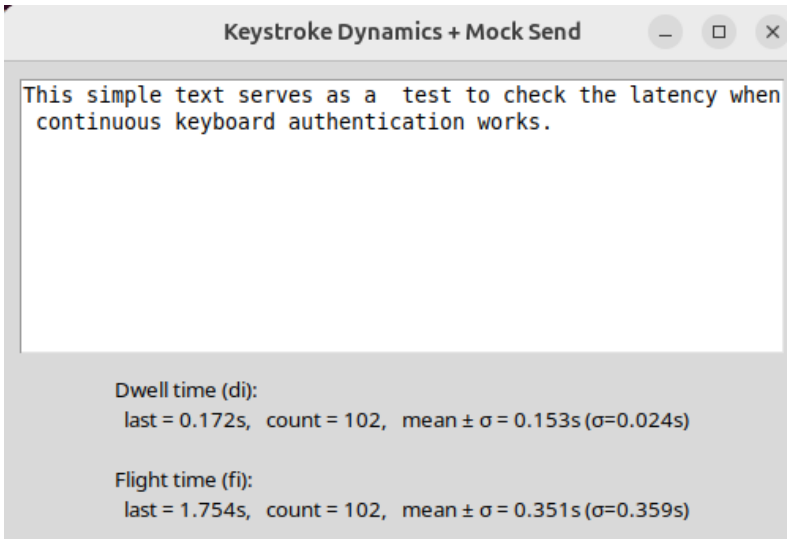

Рисунок 2.1 — Повторне дослідження затримок при обрахуванні клавіатурної динаміки

Середнє значення dwell time по всій вибірці дорівнювало 0,162 с у першому випадку, і 0.153 у другому що означає різницю у 5,6%. Середнє відхилення при цьому має різницю у 29,4%. Для часу між відпусканням попередньої і до натискання наступної клавіші відсоткові різниці такі: середні значення - 20,8%, відхилення - 20,6%. З отриманих даних видно, що перехід на більш швидкий пристрій і зміну часу доби призвів до невеликого, але помітного скорочення середнього dwell time і значного зростання розкиду індивідуальних натиснень. Аналогічно, середній інтервал між відпусканням однієї клавіші й натисканням наступної свідчить про суттєвий дрейф поведінкових характеристик.

2.2.3 Навантаження на мережу і затримки при повторних перевірках

При безперервній автентифікації кожен запит доступу породжує ланцюжок додаткових викликів до центральних сервісів (PDP, OCSP/CRL, авторизаційних API), через що мережеві витрати зростають пропорційно до кількості перевірок. Якщо в базовому сценарії запит до прикладного сервісу потребує в середньому 20 мс, то додавання одного OCSP-опитування збільшує цей час приблизно на 15 – 25 мс [28]. При паралельній обробці 10 000 сесій навантаження множиться: сукупна затримка лише від OCSP-запитів може складати сотні секунд у хвилину, що

ускладнює підтримання SLA для користувацьких додатків з інтерфейсом реального часу. Аналітичну оцінку впливу навантаження надано в додатку И

Щоб уникнути «вибухового» росту латентності, необхідно тримати ρ значно нижче пікових значень μ — за рахунок горизонтального масштабування сервісів, розподіленого кешування відповідей (OCSP-stapling, CRLite), буферизації черг і гнучкого балансування навантаження. Такий підхід дозволяє зменшити коливання λ і гарантувати стабільність затримок навіть у моменти пікових «бур» запитів.

2.3 Атаки на механізми автентифікації

2.3.1 Атаки на протоколи та моделі автентифікації

У протоколах SAML, OAuth 2.0 та mTLS виявляються специфічні вектори атак, які підривають довіру до самих механізмів автентифікації, описаних у розділі 2.

1) Replay-атаки й перехоплення токенів.У SAML-флоу зловмисник може повторно відправити раніше перехоплене SAML-твердження («assertion») до SP, якщо воно не містить надійного механізму захисту свіжості або повторного використання одноразових AssertionID. Аналогічно в OAuth/OIDC без коректного використання параметрів nonce та state атака replay дозволяє «оживити» вже відкликані чи вичерпані токени, особливо в реалізаціях без обов’язкової перевірки exp та iat у JWT.
2) Downgrade-атаки й маніпуляція метаданими.SAML-реалізації часто залежать від обміну XML-метаданими між IdP та SP. Атака XML Signature Wrapping полягає в тому, що зловмисник «обгортає» легітимне підписане твердження в інший XML-тег, підміняючи при цьому елемент, який перевіряє SP. В результаті перевірка цифрового підпису спрацьовує на оригінальному блоці, а SP обробляє вже змінений контент.
3) MITM-атаки на TLS/mTLS та зловживання OCSP/CRL. Навіть взаємна автентифікація через mTLS уразлива до MITM, якщо клієнт або сервер не перевіряють коректно ланцюжок сертифікатів чи не захищають канал OCSP/CRL-запитів.

2.3.2 Атаки на поведінкову автентифікацію

У поведінковій автентифікації головним ресурсом є модель користувача, збудована на історичних даних взаємодії з інтерфейсом. Саме ця модель стає мішенню для атак, які прагнуть змінити чи обійти поведінкові шаблони, знижуючи достовірність перевірок та збільшуючи ймовірність хибного прийняття. Однією з найнебезпечніших загроз є отруєння профілю (template poisoning). Атакувальник навмисне взаємодіє з системою, створюючи «шумові» або спотворені сесії (наприклад, вводячи певні символи з незвичайною затримкою), щоб ці нетипові дані були включені до зразків навчання. Якщо система не фільтрує аномалії, нова модель допускає ці патерни як легітимні і, навпаки, відхиляє справжні зміни в поведінці користувача.

Стрімкий розвиток методів машинного навчання ставить нові виклики для поведінкової автентифікації: Зловмисник,отримує, до прикладу, історію рухів миші легітимного користувача — траєкторії курсора, швидкість і прискорення між кроками, кутові відхилення і частоту дрібних корекцій. На основі цих даних він навчає генеративну нейронну мережу відтворювати сесії роботи з інтерфейсом, які за своєю статистикою практично не відрізняються від реальних. Після навчання мережа на льоту трансформує власні дії зловмисника в плавні, природні рухи курсора, підлаштовуючи середню швидкість, варіативність прискорень і характерні «зигзаги» так, щоб усі ключові метрики потрапляли в межі, яких очікує система. В результаті поведінкова біометрія реєструє таку сесію як легітимну, оскільки штучно згенеровані дані грубо збігаються з профілем користувача.

2.3.3 Атаки на механізми автентифікація пристроїв у IoT та критичній інфраструктурі

Недоліки енергоощадних MCU-пристроїв для IoT полягають не лише в обмеженості обчислювальних ресурсів, а й у центральній ролі шлюзу чи «хмари», на які перекинуті всі важкі криптографічні операції. Якщо атакуючий здобуде доступ до шлюзу або зможе перехопити його квитки PSK DTLS, він миттєво

отримає можливість «відтворювати» легітимні сесії вузлів, адже самі сенсори дешифрують дані лише симетричним ключем без жодного оригінального апаратного кореня довіри. Така атака в дійсності є Man-in-the-Middle проти облікового шлюзу: зламавши або підмінивши його PSK, зловмисник у прозорий спосіб відсилає йому запити від імені пристроїв, і навпаки — від імені «хмари» до сенсорів, ідентифікація при цьому успішно проходить, бо справжнє MCU не здатне відрізнити справжній шлюз від підробленого.

У протоколі RMAC критична точка — надмала кількість раундів Even–Mansour. Атака підвищеної складності дозволяє, зібравши кілька сотень повідомлень із різними IV чи підключами S_1/S_2, побудувати лінійну аппроксимацію роботи RBOX і вирішити ключі K_1/K_2 значно швидше, ніж повний перебір (наприклад, використовуючи диференційну криптоаналізу за двома раундами Even–Mansour). Після цього зловмисник генерує правильні MAC для будь-яких змінених даних, і пристрій RMAC-перевірка нічого не підозрює.

У випадку апаратних коренів довіри HRoT (TPM, Secure Element, TrustZone, HSM) основними векторами атаки стають side-channel та fault injection. Під час апаратної атаки через короткочасний збій напруги або синхронізації зловмисник підключає вузькопульсні збурення до лінії VDD, вчасно знижуючи або підвищуючи напругу у момент, коли Secure World виконує критичну перевірку автентичності або розкриває лічильник повторень. Якщо тривалість імпульсу та його затримка відносно фронту тактового сигналу підібрані з точністю декількох наносекунд, перевірка CRC коду або підпису прошивки не встигає завершитися коректно — і контролер переходить у стан помилки, який розробник часто недооцінює, дозволяючи виконати наступну інструкцію в неперевіреному блоці коду.

Нарешті, PUF-орієнтовані схеми chip-rooted trust страждають від модерних statistical modeling атак: зібравши кілька тисяч пар «challenge–response», атакуючий навчає ML-модель (наприклад, глибоку нейронну мережу чи градієнтне підвищення) апроксимувати фізичну нелінійність PUF. Після цього досить знати лише challenge, щоб передбачити відповідь із великою вірогідністю і

згенерувати легітимний підпис чи відкрити захищений канал. Якщо ж додати fault injection у саму схему PUF, можна погіршити вихідні характеристики витоку (навіть кілька бітів відповіді) так, щоб убудовані fuzzy-extractor алгоритми некоректно корегували помилки і розкривали внутрішні випадкові «пари» PUF.

2.4 Рекомендації, щодо реалізації механізмів автентифікації користувачів та пристроїв в рамках архітектури нульової довіри

В результаті проведених досліджень принципів Zero Trust (Розділ 1), методів автентифікації (Розділ 1) та виявлення ключових викликів (Розділ 2) можна сформулювати наступний перелік конкретних кроків для реалізації автентифікації користувачів і пристроїв в рамках архітектури нульової довіри. А саме:

1) Автентифікація користувачів:
   a) Мультифакторна автентифікація (MFA) за FIDO2.
      - Використовувати криптографічні ключі та біометрію (FIDO2/WebAuthn) як основний метод, оскільки це зводить до мінімуму ризик фішингу та викрадення паролів.
      - За відсутності FIDO-сумісних пристроїв — резервний рівень через TOTP/OATH-OTP із коротким терміном дії.

Мультифакторна автентифікація за FIDO2 дійсно є одним із найнадійніших підходів для захисту облікових записів, оскільки поєднання криптографічного ключа та біометричних даних зводить можливість фішингових атак практично до нуля. Ухвалення FIDO2/WebAuthn як першого рівня аутентифікації гарантує, що приватний ключ ніколи не залишає пристрій користувача, а будь-які спроби підставної форми входу просто не зможуть отримати необхідний підпис. За відсутності FIDO-сумісних пристроїв перехід на TOTP/OATH-OTP із коротким терміном дії забезпечує хоча б базовий рівень захисту, але водночас варто зауважити, що таке рішення має власні обмеження: користувач повинен мати змогу носити з собою телефон або апаратний токен, а у разі втрати пристрою процес відновлення доступу може виявитися складнішим.

b) Контекстна та безперервна автентифікація.

- Збирати мінімально потрібний набір контекстних сигналів (геолокація, тип пристрою, час запиту, поведінкові патерни) й оцінювати рівень ризику за зваженою сумою сигналів.
- Захищати екстракцію ознак на кінцевому пристрої в TEE (TrustZone, Secure Enclave) або із застосуванням HSM на граничних вузлах, щоб не порушувати приватність і не навантажувати мережу .

Контекстна та безперервна автентифікація відкриває новий рівень безпеки, бо враховує не лише те, ким є користувач, а й обставини його дій. Пропозиція збирати мінімально потрібні сигнали (геолокацію, тип пристрою, час запиту та поведінкові патерни) і обчислювати на їхній основі зважений trust score дозволяє відсіювати підозрілі входи ще на етапі прийняття рішення. Водночас захист самої екстракції ознак у TEE або за допомогою HSM гарантує, що чутливі дані не будуть вилучені в разі компрометації кінцевого пристрою. Щоб уникнути зайвих запитів, доцільно впроваджувати попередню фільтрацію даних безпосередньо на пристрої та передавати до центрального PDP тільки індексовані ознаки.

c) Адаптивні політики ризику

- Динамічно коригувати поріг trust score (T_0) залежно від загального рівня загроз («інцидентний режим» vs. «стандартний режим») .
- Забезпечити прозорість для користувача: у разі низького рівня — запитувати додатковий фактор.

Адаптивні політики ризику, які динамічно коригують поріг trust score залежно від загального рівня загроз, дозволяють оперативно підсилювати перевірки в моменти, коли інфраструктура зазнає підвищеного тиску. Такий підхід збільшує стійкість системи, адже змушує зловмисника долати значно вищий бар’єр, коли усі ресурси сфокусовані на протидії інциденту.

d) Моделі контролю доступу.

- Гібридна RBAC + ABAC**.** Перший шар — швидка перевірка ролі (RBAC) на PEP, щоб відсіяти більшість запитів за O(1). Другий шар — атрибутний аналіз (ABAC) у PDP для запитів, що пройшли RBAC-фільтр, з урахуванням контексту й ризик-показників .

Гібридна модель RBAC + ABAC дозволяє поєднати швидкість та ефективність перевірки ролей із гнучкістю атрибутного підходу. Ідея спершу відсіяти запити на PEP за O(1) завдяки ролям, а потім на PDP аналізувати вже відібрані запити з урахуванням контексту та довірчого балу є виправданою з точки зору продуктивності. Проте в реальному середовищі слід ретельно налаштувати черговість цих етапів: якщо роль узагалі не дає доступу, немає сенсу навіть звертатися до trust score, але якщо роль дозволяє, а trust score недостатній, потрібна миттєва передача запиту до ABAC, інакше користувачі знизять задоволеність системою. Також необхідно оцінити пропускну здатність PDP у пікові години, адже складні правила ABAC можуть уповільнювати прийняття рішення.

e) Поведінкова біометрія.

- Додавати поведінкову біометрію (клавіатурна динаміка, рух миші, голосові патерни, жестові патерни, сенсорні сигнали)**.** Для захисту від replay-атак застосувати лінійні challenge-response протоколи.

Поведінкова біометрія дає змогу здійснювати постійне підтвердження автентичності користувача без безперервного втручання з його боку: система може моніторити клавіатурну динаміку, рух миші, голосові патерни чи сенсорні дані і виявляти суттєві відхилення від навченої моделі. У разі застосування challenge–response-процедур під час початкового збору даних replay-атаки виявляються й відіграють мінімальну роль. Однак збір і обробка поведінкової інформації – дуже ресурсномісткий процес, а також порушує вимоги щодо конфіденційності: користувачі мають бути поінформовані про те, які саме дані збираються і як вони зберігатимуться. Крім того, хибнопозитивні спрацьовування можуть відбуватися у разі зміни обстановки (інша клавіатура, стрес тощо), тому

слід залишати «запасний шлях» – наприклад, при трьох поспіль відхиленнях переводити користувача на MFA.

2) Автентифікація пристроїв

a) PKI та X.509-сертифікати.

- Розгорнути корпоративну CA+федерацію CA для масштабованої видачі і ротації сертифікатів.
- Налаштувати OCSP-stapling на PEP і TTL кешу статусу сертифікатів адаптивно (наприклад, 10–60 с залежно від рівня навантаження) .

Централізована архітектура спрощує видання, ротацію й відкликання сертифікатів для служби захищених з'єднань. При цьому федерація CA дозволяє безпечно обмінюватися сертифікатами між підрозділами та партнерськими організаціями, але потребує напрацювання чітких CPS, щоб не допустити неконтрольованого випуску сертифікатів. Налаштування OCSP-stapling із адаптивним TTL прискорює підтвердження статусу сертифіката директором PEP, водночас під час пікового навантаження зменшена частота оновлень може створити тимчасову інертність, якщо сертифікат відкликано, а кеш ще не встиг оновитися.

b) Апаратні корені довіри.

- Використовувати TPM або мінімум TrustZone на кожному корпоративному пристрої для захисту приватних ключів та перевірки цілісності платформи.
- Для IoT/кінцевих пристроїв із критичними вимогами – інтегрувати PUF-чіпи з fuzzy extractor (KeyPUF) як джерело нестандартних ключів.

Використання апаратних коренів довіри (TPM, TrustZone) на кожному корпоративному пристрої — це базовий елемент сучасних практик кібербезпеки, який гарантує, що приватні ключі ніколи не залишають апаратне середовище, а платформа не завантажує невідоме програмне забезпечення. У корпоративному сегменті практично всі нові ноутбуки вже мають TPM 2.0, що дозволяє захистити

ключі та виконувати перевірку цілісності системи. Для смартфонів і планшетів завдання вирішують Secure Enclave чи Android Keystore.

с) Легковагові протоколи для обмежених пристроїв

- Для ультра-обмежених MCU (RAM ≤ 16 КБ, без PKI) Впровадити простий авторизаційний MAC протокол (наприклад, RMAC) через мінімальні обчислювальні витрати, відсутність складних крипто-бібліотек і управління ключами. Приклади пристроїв: датчики/актуатори без OS, із обміном раз на годину й жорсткими обмеженнями пам’яті.
- У мережах NB-IoT / 6LoWPAN (фрагментація, низька пропускна здатність) використовувати EDHOC для встановлення PSK-сесії, бо “рукостискання” займає ~100 байт (замість ~500 байт у DTLS), низька затримка. Приклади пристроїв: батарейні прилади з низькою частотою передачі, де кожен байт і мілісекунда важливі.
- У пристроях із частими запитами / переважно UDP (без TCP/IP стека) рекомендовано впровадити CoAP + OSCORE для захисту повідомлень на рівні додатку. Причина: не потребує TLS-каналу, шифрує й автентифікує заголовки та payload одночасно, накладні витрати ≈ 30 байт. Приклади пристроїв: класи IoT з невеликими повідомленнями (температура, статус), де TCP-протокол неможливо використовувати.
- У пристроях з базовим PSK-менеджментом та бажанням TLS-інтероперабельності використовувати DTLS 1.3 із PSK-only cipher suites + 0-RTT session resumption через швидке відновлення сесії, мінімальної кількості пакетів (1–2 пакети handshake), сумісність із існуючими TLS-стеками. Приклади пристроїв: девайси на Linux/FreeRTOS із підтримкою DTLS, де є простий PSK-сервіс.
- При потребі контекстно-орієнтованої безперервної автентифікації використовувати схему LCDA (Lightweight Continuous Device-to-Device Authentication) на основі CSI у разі, коли дві чи більше

кінцевих точок у спільному бездротовому середовищі потребують безперервної верифікації.

Легковагові протоколи для обмежених пристроїв необхідні в ситуаціях, де MCU має не більше ніж кілька кбайтів оперативної пам’яті та не може підтримувати асиметричні алгоритми. Пропозиція використовувати простий MAC-протокол (наприклад, RMAC) дає змогу вирішити завдання базової автентифікації та забезпечити цілісність даних при мінімальному обчислювальному навантаженні. Однак RMAC сам по собі не шифрує дані, і якщо потрібна конфіденційність, доведеться додавати легке шифрування. EDHOC для NB-IoT/6LoWPAN — це оптимальне рішення, яке скорочує „handshake” до приблизно 100 байт замість 500 у DTLS, проте через середовище NB-IoT із тривалими періодами deep sleep інколи доводиться повторювати спроби встановлення сесії, і це може затримувати обмін. Комбінація CoAP + OSCORE працює ідеально, коли пристрої обмінюються невеликими повідомленнями у великих кількостях (понад тисячу в секунду) та не потребують повноцінного TLS, але при цьому потрібно налаштувати централізований PSK-менеджмент і механізм оновлення ключів, щоб не доводилося перепрошивати датчики. DTLS 1.3 із PSK-only cipher suites і 0-RTT session resumption — відмінний варіант для Linux/FreeRTOS-пристроїв, які вже мають IP stack, але слід подбати про анти-реплей механізми, щоб 0-RTT не дозволяв повторювати старі пакети. Нарешті, LCDA на основі CSI демонструє перспективу безконтрактної автентифікації у локальному бездротовому середовищі, проте її ефективність залежить від стабільності каналу: у динамічних приміщеннях з великою кількістю руху доведеться застосовувати алгоритми фільтрації шуму, інакше система буде давати хибні спрацьовування.

Висновки за 2 розділ

1) Проведені в розділі 2 дослідження показують, що впровадження автентифікації в рамках архітектури нульової довіри стикається з низкою взаємопов’язаних перешкод:

а) існуючі LDAP/AD-інфраструктури виявляються надто негнучкими для постійного контекстного підтвердження запитів;

б) розмаїття стандартів вимагає складної нормалізації та синхронізації метаданих, що призводить до «вікон вразливості»;

в) безперервні контекстуальні перевірки порушують приватність користувачів. Окремо варто відзначити вразливість критичних компонентів — від шаблонів поведінкової біометрії до апаратних коренів довіри в IoT — перед витонченими replay-, downgrade- та side-channel-атаками.

2) Пропонується багаторівнева модель автентифікації за принципом «мінімальної довіри»: для користувачів — зробити FIDO2/WebAuthn ядром MFA з резервними TOTP і фоновою контекстною перевіркою з динамічним trust score; для пристроїв — використовувати корпоративну PKI з адаптивним OCSP-stapling і зберіганням ключів у TPM/TrustZone (у критичних IoT-моделях — PUF-чіпи з fuzzy extractor); а в сильно обмежених MCU-сценаріях застосувати легковагові MAC-протоколи (RMAC), EDHOC та CoAP + OSCORE для мінімізації RTT та метаданих. Такий гібридний підхід гарантує баланс між високою безпекою, продуктивністю та зручністю.

3 ПЕРСПЕКТИВИ РОЗВИТКУ ТА НОВІТНІ ПІДХОДИ У АВТЕНТИФІКАЦІЇ В РАМКАХ ZTA

3.1 Новітні тенденції в біометричній автентифікації

Сьогодні біометрична автентифікація представлена насамперед сканерами відбитків пальців, розпізнаванням обличчя та голосу, тощо. Ці методи вже довели свою ефективність у багатьох прикладних рішеннях: у смартфонах, ноутбуках та корпоративних системах їх використовують як швидкий і зручний спосіб підтвердження особи без необхідності запам’ятовувати складні паролі або використовувати сертифікати/токени. Проте сьогодні ми стоїмо на порозі нового етапу, коли біометрія може переступити межі зовнішніх сенсорів і перейти до безпосереднього виміру внутрішніх фізіологічних сигналів.

У найближчому майбутньому розвиток нейронних імплантів і пристроїв Brain-Computer Interface (BCI) відкриває перспективу використання унікальних електроенцефалографічних (ЕЕГ) відбитків мозку для автентифікації: кожен мозок генерує неповторні патерни активності, які можна було б поєднувати з традиційною зовнішньою біометрією. У 2024 році компанія Neuralink уперше у світі вживила в мозок людини інтерфейс Brain-Computer Interface (модель N1), що дозволило пацієнту керувати комп’ютерним курсором силою думки та стало першим кроком до комерційних нейроімплантів [29]. Потенційно це відкриває можливість використовувати унікальні патерни нейронної активності як новий біометричний фактор в автентифікації: кожен мозок генерує індивідуальні електричні сигнали, зокрема викликані потенціали (ERP), які можна реєструвати й аналізувати для підтвердження особи.

3.2 Використання машинного навчання для аналізу довіри

У нульовій довірі кожен запит підлягає аналізу не лише за його атрибутами, а й за його місцем у складній мережі взаємодій між користувачами, пристроями та сервісами, графові та глибинні моделі на основі штучного інтелекту обіцяють

стати наріжним каменем майбутньої аналітики довіри. Сьогодні лише поодинокі SIEM- та UEBA-платформи використовують елементи графових алгоритмів для виявлення аномалій у зв’язках між об’єктами, але вже в майбутньому можна буде побачити, як Graph Neural Networks (GNN) та Knowledge Graphs інтегруватимуться в архітектуру ZTA «на краю», безперервно оновлюючи мультифакторний рівень довіри.

Кожен клік, кожне звернення до API чи кожна зміна конфігурації пристрою будуть відображатися як вузли та ребра в динамічному графі, що моделює поточну картину довіри в корпоративній інфраструктурі. GNN-моделі зможуть «пропускати» через себе тисячі атрибутів — від історії успішних автентифікацій до оновлень безпеки на пристрої кінцевого користувача — і виявляти закономірності, недоступні класичним методам кластеризації чи лінійним скоринговим схемам [30].

3.3 Постквантові механізми автентифікації

3.3.1 Геш- та багатоваріантні підписи

У межах постквантової криптографії геш-базовані та багатоваріантні підписи сьогодні розглядаються як найперспективніші підходи для захисту від квантових атак. Наразі стандартизованим представником геш-базованих схем є SPHINCS+, що забезпечує високу стійкість завдяки кількісному використанню довгих послідовностей геш-функцій [31]. Багатоваріантні алгоритми на кшталт Rainbow та GeMSS базуються на складних системах поліноміальних рівнянь, що дають коротші ключі й швидші операції, але наразі страждають від не до кінця вивчених векторів атак та великих розмірів публічних ключів .

Проте в найближчому майбутньому очікується низка інновацій, які суттєво підвищать практичність цих схем у ZTA. По-перше, активна оптимізація геш-базованих протоколів із використанням Merkle-дерев нового типу та вбудованих у Secure Enclave апаратних прискорювачів дозволить зменшити габарити підписів і суттєво підвищити швидкість підписування й верифікації. По-друге, у галузі багатоваріантних схем розвиватимуться методи скорочення розміру публічних

ключів через застосування розріджених поліномів і багаторівневих компресійних структур. Крім того, перспективним є поява гібридних підходів «геш-базовані + багатоваріантні», коли короткі, але менш стійкі підписи Rainbow будуть комбінуватися з надміцними SPHINCS+ через техніку threshold-підписів. У ZTA це дозволить налаштовувати політики доступу залежно від необхідного рівня квантового захисту: для критичних запитів система вимагатиме повноцінної SPHINCS+, а для менш важливих — прискорений гібридний режим.

3.3.2 Ключі та підписи на базі решіток і кодів

Ключі та підписи на базі решіток і кодів сьогодні перебувають на етапі активного доведення до виробничих систем: серед lattice-схем лідирують CRYSTALS-Dilithium та Falcon [32], які вже потрапили до фінального кола стандартів NIST та демонструють прийнятну швидкість підписування й перевірки, хоча потребують більшого обсягу ключів порівняно з класичними Еліптичними Кривими. З іншого боку, кодобазована классична схема McEliece здавна відома своєю стійкістю, але зазнає критики через величезний розмір відкритого ключа (декілька мегабайт) та обмежену гнучкість при оновленні параметрів.

Найближчими роками ми побачимо кілька ключових вдосконалень, які дозволять зробити lattice- та code-based схеми справді практичними для Zero Trust-платформ. По-перше, на апаратному рівні з’являться вбудовані прискорювачі PQ-криптографії (як це вже реалізовано для AES і SHA-хешів), що знизять затримки підписування до мілісекунд і дадуть змогу виконувати операції «на краю» — в IoT-пристроях, мобільних терміналах та тонких клієнтах. По-друге, алгоритмісти працюють над оптимізацією параметрів Dilithium: вже зараз вивчаються варіанти зі скороченими ключами та спрощеною структурою мультирівневих матриць, що дозволить отримати компроміс «безпека проти розміру» без суттєвої втрати стійкості.

У паралельному потоці досліджень кодобазовані схеми, зокрема Classic McEliece та скорочений Niederreiter, удосконалюють за рахунок багаторівневих

технік стиснення ключів (sparse-matrix representations) та гібридних конструкцій, де невеликий primary ключ McEliece підсилюється додатковим шаром легшої lattice-схеми, наприклад Kyber-KEM, для одноразового захисту каналу обміну.

Для архітектури нульової довіри це означає можливість міграції всіх етапів автентифікації на постквантові основи: від генерації сертифікатів на CA до перевірки підписів у Policy Enforcement Point. Edge-модулі з апаратними прискорювачами зможуть миттєво верифікувати постквантовий підпис, а Policy Decision Point аналізуватиме якість ключа (розмір, версію параметрів) як частину рівня ризику. У результаті навіть появлення повноцінного квантового комп'ютера не зможе підірвати цілісності довіри — система вже опиниться на новому криптографічному рівні, готовому до будь-яких обчислювальних викликів.

Висновки за 3 розділ

1) Проведені в розділі 3 дослідження окреслюють майбутній вектор розвитку автентифікації в ZTA як поступовий перехід від зовнішніх до внутрішніх біометричних сигналів з одночасним поєднанням фізичних та поведінкових факторів, що дозволяє створювати гібридні, контекстозалежні профілі довіри; впровадження графових і глибинних моделей на базі машинного навчання для безперервного та конфіденційного оцінювання кожного запиту в режимі edge/federated learning; а також поступове заміщення класичних алгоритмів постквантовими геш- та багатоваріантними підписами й схемами на базі решіток і кодів із апаратними прискорювачами для забезпечення захищеності проти квантових обчислень.

2) Поєднуючи мультимодальну біометрію із адаптивним trust-score у динамічному графі довіри та постквантовими механізмами підписів, майбутня ZTA-архітектура зможе гарантувати не лише високий рівень безпеки, а й мінімальні затримки та збереження приватності користувачів, — саме такий комплексний підхід забезпечить стійкість корпоративної інфраструктури в умовах швидкозмінних загроз і появи квантових обчислень.

ВИСНОВКИ

1. У результаті проведеного дослідження було показано, що впровадження архітектури нульової довіри вимагає комплексного перегляду традиційних підходів до автентифікації: від одноразових перевірок під час входу до безперервного моніторингу кожного запиту.

2. Детальний розгляд механізмів автентифікації користувачів та пристроїв показав, що сертифікаторієнтовані протоколи, такі як mTLS на базі X.509, забезпечують найвищий рівень криптографічної стійкості, проте потребують складного управління життєвим циклом сертифікатів і можуть створювати затримки через часті перевірки CRL/OCSP. Водночас токенорієнтовані рішення відзначаються гнучкістю інтеграції та масштабованістю, але вимагають додаткових механізмів відкликання і гарантії цілісності токенів . Експериментальне порівняння показало, що обробка запиту з використанням OAuth 2.0 + JWT відбувається приблизно на 47,4 % швидше, ніж mTLS, що робить токенорієнтовані рішення доцільними для середовищ із численними внутрішніми зверненнями, тоді як mTLS доцільно застосовувати в критично чутливих сценаріях .

3. Обґрунтовано побудову моделей контролю доступу через поєднання RBAC та ABAC.

4. Важливою складовою Zero Trust є адаптивна та безперервна автентифікація, що поєднує традиційні фактори з поведінковими сигналами та контекстними параметрами. Застосування машинного навчання для аналізу шаблонів поведінки, клавіатурної динаміки, рухів миші, голосу, рухів дозволяє оперативно виявляти аномалії без прямої участі користувача й мінімізувати хибні спрацювання. У перспективі розвиток мультимодальної біометрії, глибинних графових моделей та постквантових криптографічних схем забезпечить ще вищий рівень захищеності й продуктивності системи .

5. Результати дослідження лягли в основу рекомендацій щодо побудови ефективних механізмів автентифікації в рамках Zero Trust Architecture:

A. Автентифікація користувачів:

a) Мультифакторна автентифікація (MFA) за FIDO2.

- Використовувати криптографічні ключі та біометрію (FIDO2/WebAuthn) як основний метод.
- За відсутності FIDO-сумісних пристроїв — резервний рівень через TOTP/OATH-OTP із коротким терміном дії.

b) Контекстна та безперервна автентифікація.

- Збирати мінімально потрібний набір контекстних сигналів й оцінювати рівень ризику за зваженою сумою сигналів.
- Захищати екстракцію ознак на кінцевому пристрої в TEE або із застосуванням HSM на граничних вузлах, щоб не порушувати приватність і не навантажувати мережу .

c) Адаптивні політики ризику

- Динамічно коригувати поріг trust score (T_0) залежно від загального рівня загроз.
- Забезпечити прозорість для користувача: у разі низького рівня — запитувати додатковий фактор.

d) Моделі контролю доступу.

- Гібридна RBAC + ABAC**.**

e) Поведінкова біометрія.

- Додавати поведінкову біометрію. Для захисту від replay-атак застосувати лінійні challenge-response протоколи.

B. Автентифікація пристроїв

a) PKI та X.509-сертифікати.

- Розгорнути корпоративну CA+федерацію CA для масштабованої видачі і ротації сертифікатів.

- Налаштувати OCSP-stapling на PEP і TTL кешу статусу сертифікатів адаптивно (наприклад, 10–60 с залежно від рівня навантаження) .

b) Апаратні корені довіри.

- Використовувати TPM або мінімум TrustZone на кожному корпоративному пристрої.
- Для IoT/кінцевих пристроїв із критичними вимогами – інтегрувати PUF-чіпи з fuzzy extractor (KeyPUF) як джерело нестандартних ключів.

c) Легковагові протоколи для обмежених пристроїв

- Для ультра-обмежених MCU (RAM ≤ 16 КБ, без PKI) Впровадити простий авторизаційний MAC протокол (наприклад, RMAC).
- У мережах NB-IoT / 6LoWPAN (фрагментація, низька пропускна здатність) використовувати EDHOC для встановлення PSK-сесії.
- У пристроях із частими запитами / переважно UDP (без TCP/IP стека) рекомендовано впровадити CoAP + OSCORE для захисту повідомлень на рівні додатку.
- У пристроях з базовим PSK-менеджментом та бажанням TLS-інтероперабельності використовувати DTLS 1.3 із PSK-only cipher suites + 0-RTT session resumption
- При потребі контекстно-орієнтованої безперервної автентифікації використовувати схему LCDA (Lightweight Continuous Device-to-Device Authentication)

Запропонований гібридний підхід дозволяє організаціям захистити свої середовища від внутрішніх і зовнішніх загроз, адаптуватися до мінливих контекстів доступу та забезпечити баланс між безпекою, продуктивністю і зручністю для кінцевого користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Forrester. Zero Trust Model John Kindervag et al. 2010. Build security into your networks dna: The zero trust network architecture. Forrester Research Inc 27 (2010)
2. Zero Trust Architecture. NIST Special Publication 800-207. National Institute of Standards and Technology, 2020.
3. How to protect your business from digital threats | Hideez. Passwordless Workforce Identity Solutions | Hideez. Режим доступу: https://hideez.com/uk/blogs/news/how-to-protect-your-business-from-digital-threats
4. Access Control Policy Enforcement for Zero-Trust-Networking / R. Vanickis et al. 2018 29th Irish Signals and Systems Conference (ISSC), Belfast, 21–22 June 2018. 2018.
5. John Kindervag. No More Chewy Centers: The Zero Trust Model Of Information Security
6. Zanasi C., Russo S., Colajanni M. Flexible zero trust architecture for the cybersecurity of industrial IoT infrastructures. Ad Hoc Networks. 2024. P. 103414.
7. A review of multi-factor authentication in the Internet of Healthcare Things / T. Suleski et al. DIGITAL HEALTH. 2023. Vol. 9. P. 205520762311771.
8. Zero Trust Architecture Draft (2nd 1) NIST Special Publication 800-207 2 3. National Institute of Standards and Technology, 2020.
9. Introduction to Trusted Execution Environment and ARM's TrustZone. sergioprado.blog. Режим доступу: https://sergioprado.blog/introduction-to-trusted-execution-environment-tee-arm-trustzone
10. Zero Trust Architecture (ZTA): A Comprehensive Survey / N. F. Syed et al. IEEE Access. 2022. P. 1.
11. The OAuth 2.1 Authorization Framework. IETF Datatracker. Режим доступу: https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-00

12. RFC 7522: Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. IETF Datatracker. Режим доступу: https://datatracker.ietf.org/doc/html/rfc7522
13. A Review on Feature Extraction in Keystroke Dynamics / M. N. Yaacob et al. Journal of Physics: Conference Series. 2020. Vol. 1529. P. 022088.
14. Typing Speed | Typing Pal. Typing Pal. Режим доступу: https://www.typingpal.com/en/documentation/school-edition/pedagogical-resources/typing-speed
15. Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication / M. Frank et al. IEEE Transactions on Information Forensics and Security. 2013. Vol. 8, no. 1. P. 136–148.
16. Fayek H. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. Haytham Fayek. Режим доступу: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
17. Joint Design of Multi-Dimensional Multiple Access and Lightweight Continuous Authentication in Zero-Trust Environments / H. Fang et al. GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 4–8 December 2023. 2023.
18. Bergadano F., Gunetti D., Picardi C. User authentication through keystroke dynamics. ACM Transactions on Information and System Security. 2002. Vol. 5, no. 4. P. 367–397.
19. Gunetti, D. & Picardi, C. "Keystroke analysis for user identification,"
20. BehaveFormer: A Framework with Spatio-Temporal Dual Attention Transformers for IMU-enhanced Keystroke Dynamics / D. Senarath et al. 2023 IEEE International Joint Conference on Biometrics (IJCB), Ljubljana, Slovenia, 25–28 September 2023. 2023.
21. COMPARISON OF DIGITAL SIGNAL PROCESSING METHODS AND DEEP LEARNING MODELS IN VOICE AUTHENTICATION / K. Ruda et al. Cybersecurity: Education, Science, Technique. 2024. Vol. 1, no. 25. P. 140–160.

22. Khoureich Ka A. RMAC - A Lightweight Authentication Protocol for Highly Constrained IoT Devices. International Journal on Cryptography and Information Security. 2018. Vol. 8, no. 3. P. 01–14.
23. LCDA: Lightweight Continuous Device-to-Device Authentication for a Zero Trust Architecture (ZTA) / S. W. Shah et al. Computers & Security. 2021. Vol. 108. P. 102351.
24. eeDTLS: Energy-Efficient Datagram Transport Layer Security for the Internet of Things / U. Banerjee et al. 2017 IEEE Global Communications Conference (GLOBECOM 2017), Singapore, 4–8 December 2017. 2017.
25. Secure Communication for the IoT: EDHOC and (Group) OSCORE Protocols / R. Höglund et al. IEEE Access. 2024. P. 1.
26. On-premises and hybrid authentication: Challenges and best practices. WorkOS. WorkOS Your app, Enterprise Ready. Режим доступу: https://workos.com/blog/on-premises-and-hybrid-authentication-challenges-and-best-practices
27. NIST PRIVACY FRAMEWORK:. Gaithersburg, MD : National Institute of Standards and Technology, 2020.
28. Zhu L., Amann J., Heidemann J. Measuring the Latency and Pervasiveness of TLS Certificate Revocation. Passive and Active Measurement. Cham, 2016. P. 16–29.
29. Neuralink implants brain chip in first human. Reuters. Режим доступу: https://www.reuters.com/technology/neuralink-implants-brain-chip-first-human-musk-says-2024-01-29
30. Revital Marbel, Yanir Cohen, Ran Dubin, Amit Dvir, Chen Hajaj. Dynamic Graph Neural Network for Early Detection of Cloud Services' User Anomalies
31. SPHINCS+ Stateless hash-based signatures. Режим доступу: https://sphincs.org/
32. NIST Releases First 3 Finalized Post-Quantum Encryption Standards. Режим доступу: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards.

ДОДАТОК А

Опис методології дослідження та код програми

Методологія дослідження:

1) Для кожного методу було реалізовано симуляцію:
   - OAuth 2.0 + JWT: генерація access-token з підписом RSA (RS256), перевірка параметрів токена (алгоритм, exp, iat, scope, aud).
   - mTLS (X.509): симуляція TLS-хендшейку з клієнтським сертифікатом, валідація його полів (CN, issuer, термін дії), імітація роботи з PDP.
2) Для кожної симуляції вимірювалась *затримка* (*latency*), що включала і мережеву затримку, і час перевірки атрибутів.
3) Було проведено N = 10 вимірювань для кожного типу автентифікації.

Код:

```
import time
import random
import jwt
from datetime import datetime, timedelta
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization, hashes

def gen_rsa_keys():
    key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
    priv_pem = key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    pub_pem = key.public_key().public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
    return priv_pem, pub_pem

PRIV_KEY_PEM, PUB_KEY_PEM = gen_rsa_keys()

def verify_jwt(token: str) -> bool:
    try:
        payload = jwt.decode(
            token,
            PUB_KEY_PEM,
            algorithms=["RS256"],
            audience="client-app"
        )
        assert "scope" in payload, "scope missing"
        assert payload["iat"] <= datetime.utcnow().timestamp(), "iat in future"
        assert payload["exp"] > datetime.utcnow().timestamp(), "exp expired"
        return True
```

```
    except Exception as e:
        print("PDP JWT reject:", e)
        return False

def simulate_oauth_request():
    delay = random.uniform(0.05, 0.2)
    time.sleep(delay)
    now = datetime.utcnow()
    token = jwt.encode({
        "iat": now.timestamp(),
        "exp": (now + timedelta(seconds=60)).timestamp(),
        "aud": "client-app",
        "scope": "read:data"
    }, PRIV_KEY_PEM, algorithm="RS256")
    return token, delay

def verify_cert(cert: dict) -> bool:
    try:
        if cert.get("subject", {}).get("commonName") != "Trusted Client":
            raise ValueError("Untrusted CN")
        if cert.get("issuer", {}).get("commonName") != "Trusted CA":
            raise ValueError("Untrusted CA")
        if cert.get("notAfter") < datetime.utcnow():
            raise ValueError("Certificate expired")
        return True
    except Exception as e:
        print("PDP Cert reject:", e)
        return False

def simulate_tls_handshake():
    delay = random.uniform(0.1, 0.3)
    time.sleep(delay)
    fake_cert = {
        "subject":       {"commonName": "Trusted Client"},
        "issuer":        {"commonName": "Trusted CA"},
        "notAfter":      datetime.utcnow() + timedelta(days=30)
    }
    return fake_cert, delay

def test_oauth():
    print("\nOAuth 2.0 + OIDC:")
    token, delay = simulate_oauth_request()
    valid = verify_jwt(token)
    print(f"  • Token valid: {valid}")
    print(f"  • Round-trip time (simulated): {delay:.3f}s")

def test_mtls():
    print("\n mTLS (X.509) PoC:")
    cert, delay = simulate_tls_handshake()
    valid = verify_cert(cert)
    print(f"  • Cert valid: {valid}")
    print(f"  • TLS handshake time (simulated): {delay:.3f}s")

if __name__ == "__main__":
    test_oauth()
    test_mtls()
```

ДОДАТОК Б

Опис та код програми

Кожні 0.5 с у графічному інтерфейсі оновлюється статистика: останнє значення d_{last} та f_{last}; кількість зібраних замірів; середнє μ та стандартне відхилення σ для кожного типу метрики. Первинна обробка і обчислення статистики відбуваються безпосередньо на клієнтському пристрої, що виключає передачу сирих часових міток і гарантує конфіденційність. Для інтеграції з серверним PDP-модулем у Zero Trust Architecture кожні 5 секунд агент формує JSON-навантаження щоб імітувати передання даних для аналізу ризиків.

Код:

```
import time
import statistics
import threading
import json
import tkinter as tk
import random

class KeystrokeDynamicsAppWithSend:
    def __init__(self, master):
        self.master = master
        master.title("Keystroke Dynamics + Mock Send")
        self.press_times = {}
        self.dwell_times = []
        self.flight_times = []
        self.last_dwell = None
        self.last_flight = None
        self.debounce_ids = {}
        self.repeat_interval = 0.05
        self.last_release_time = None


        self.text = tk.Text(master, width=60, height=10)
        self.text.pack(padx=10, pady=10)
        self.text.focus_set()

        self.stats_label = tk.Label(master, text="Почніть друкувати…", justify="left")
        self.stats_label.pack(padx=10, pady=(0,10))

        self.text.bind("<KeyPress>",   self.on_key_press)
        self.text.bind("<KeyRelease>", self.on_key_release)

        self.updater()
        self.sender()

    def on_key_press(self, event):
        key = event.keysym
        if key not in self.press_times:
            self.press_times[key] = time.time()
        if key in self.debounce_ids:
```

```
            self.master.after_cancel(self.debounce_ids.pop(key))

    def on_key_release(self, event):
        key = event.keysym

        def finalize_release():
            t_rel = time.time()
            t_press = self.press_times.pop(key, None)
            if t_press is not None:
                d = t_rel - t_press
                self.dwell_times.append(d)
                self.last_dwell = d
            if self.last_release_time is not None:
                f = t_rel - self.last_release_time
                self.flight_times.append(f)
                self.last_flight = f
            self.last_release_time = t_rel
            self.debounce_ids.pop(key, None)

        after_id = self.master.after(
            int(self.repeat_interval * 1000) + 1,
            finalize_release
        )
        self.debounce_ids[key] = after_id

    def updater(self):
        def fmt(lst):
            return "—" if not lst else f"{statistics.mean(lst):.3f}s
(σ={statistics.pstdev(lst):.3f}s)"

        last_d = f"{self.last_dwell:.3f}s" if self.last_dwell is not None else "—"
        last_f = f"{self.last_flight:.3f}s" if self.last_flight is not None else "—"

        text = (
            f"Dwell time (di):\n"
            f"  last = {last_d},   count = {len(self.dwell_times)},   mean ± σ =
{fmt(self.dwell_times)}\n\n"
            f"Flight time (fi):\n"
            f"  last = {last_f},   count = {len(self.flight_times)},   mean ± σ =
{fmt(self.flight_times)}"
        )
        self.stats_label.config(text=text)
        self.master.after(500, self.updater)

    def sender(self):
        """Раз на 5 с імітуємо відправку агрегованих даних на сервер."""
        def do_send():

            payload = {
                "timestamp": time.time(),
                "dwell": {
                    "last": self.last_dwell,
                    "count": len(self.dwell_times),
                    "mean": statistics.mean(self.dwell_times) if self.dwell_times else
None,
                    "std": statistics.pstdev(self.dwell_times) if self.dwell_times else
None
                },
                "flight": {
                    "last": self.last_flight,
                    "count": len(self.flight_times),
                    "mean": statistics.mean(self.flight_times) if self.flight_times else
None,
                    "std": statistics.pstdev(self.flight_times) if self.flight_times else
None
                }
            }
            time.sleep(random.uniform(0.05, 0.15))
            print(">>> [Mock SEND] POST /keystroke-data")
            print(json.dumps(payload, indent=2, ensure_ascii=False))
```

```
        threading.Thread(target=do_send, daemon=True).start()
        self.master.after(5000, self.sender)


if __name__ == "__main__":
    root = tk.Tk()
    app = KeystrokeDynamicsAppWithSend(root)
    root.mainloop()
```

ДОДАТОК В

Опис та код програми і вихідні дані рухових експериментів

Інтерфейс додатка максимально спрощений: передбачено кнопку "Старт", яка запускає процес вимірювання на фіксований період (25 секунд). Протягом цього часу додаток періодично (кожні 100 мс) зчитує вектор прискорення з трьох осей — x,y,z, фільтрує та обчислює миттєве значення енергії руху за формулою (1.5). Ці значення накопичуються у вигляді пари «час — енергія» та зберігаються у CSV-файл. По завершенні вимірювання також будується графік залежності енергії від часу, який зберігається у вигляді зображення.

Код програми:

```
package com.example.tsten
import android.content.Context
import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import java.io.FileOutputStream

class MainActivity : AppCompatActivity(), SensorEventListener {

    private lateinit var sensorManager: SensorManager
    private var accelerometer: Sensor? = null


    private val gravity = FloatArray(3)
    private val linearAcc = FloatArray(3)
    private val alpha = 0.8f

    private val dataPoints = mutableListOf<Pair<Float, Float>>()
    private var startTime = 0L

    private val interval = 100L
    private val duration = 25_000L
    private val handler = Handler(Looper.getMainLooper())
    private var runnable: Runnable? = null

    private lateinit var startButton: Button
    private lateinit var timerText: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        startButton = findViewById(R.id.startButton)
```

```
        timerText   = findViewById(R.id.timerText)

        sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)

        startButton.setOnClickListener {
            if (accelerometer == null) {
                Toast.makeText(this, "Акселерометр не знайдено!", Toast.LENGTH_LONG).show()
            } else {
                startMeasurement()
            }
        }
    }

    private fun startMeasurement() {
        Toast.makeText(this, "Вимірювання почалося", Toast.LENGTH_SHORT).show()
        dataPoints.clear()
        startTime = System.currentTimeMillis()
        startButton.isEnabled = false

        sensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_GAME)

        runnable = object : Runnable {
            override fun run() {
                val elapsed = System.currentTimeMillis() - startTime
                val remaining = ((duration - elapsed).coerceAtLeast(0L)) / 1000f
                timerText.text = String.format("%.1f s", remaining)

                val timeSec = elapsed / 1000f
                val energy = linearAcc.map { it * it }.sum()
                dataPoints.add(timeSec to energy)

                if (elapsed < duration) handler.postDelayed(this, interval)
                else finishMeasurement()
            }
        }
        handler.post(runnable!!)
    }

    private fun finishMeasurement() {
        sensorManager.unregisterListener(this)
        handler.removeCallbacks(runnable!!)
        saveDataToFile()
        drawGraph()
        Toast.makeText(this, "Завершено", Toast.LENGTH_LONG).show()
        startButton.isEnabled = true
        timerText.text = "25.0 s"
    }

    override fun onSensorChanged(event: SensorEvent) {
        gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0]
        gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1]
        gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2]
        linearAcc[0] = event.values[0] - gravity[0]
        linearAcc[1] = event.values[1] - gravity[1]
        linearAcc[2] = event.values[2] - gravity[2]
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}

    private fun saveDataToFile() {
        File(getExternalFilesDir(null), "data.csv").printWriter().use { out ->
            out.println("Time,Energy")
            dataPoints.forEach { (t, e) -> out.println("$t,$e") }
        }
    }

    private fun drawGraph() {
        val width  = 800
```

```
        val height = 600
        val margin = 50

        val bmp    = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)
        val canvas = Canvas(bmp).apply { drawColor(Color.WHITE) }

        val axis = Paint().apply { color = Color.BLACK; strokeWidth = 3f; textSize = 24f }
        canvas.drawLine(margin.toFloat(), (height - margin).toFloat(),
            (width - margin).toFloat(), (height - margin).toFloat(), axis)
        canvas.drawLine(margin.toFloat(), margin.toFloat(),
            margin.toFloat(), (height - margin).toFloat(), axis)
        canvas.drawText("Time (s)", width/2f, (height - margin/3f), axis)
        canvas.save()
        canvas.rotate(-90f, margin/4f, height/2f)
        canvas.drawText("Energy", margin/4f, height/2f, axis)
        canvas.restore()

        val paint = Paint().apply { color = Color.BLUE; strokeWidth = 2f }
        val maxT  = dataPoints.maxOfOrNull { it.first } ?: 1f
        val maxE  = dataPoints.maxOfOrNull { it.second } ?: 1f

        var px = margin.toFloat()
        var py = (height - margin).toFloat()
        dataPoints.forEach { (t, e) ->
            val x = margin + (t / maxT) * (width  - 2*margin)
            val y = height - margin - (e / maxE) * (height - 2*margin)
            canvas.drawLine(px, py, x, y, paint)
            px = x; py = y
        }

        FileOutputStream(File(getExternalFilesDir(null), "graph.png"))
            .use { bmp.compress(Bitmap.CompressFormat.PNG, 100, it) }
    }
}
```

Перший графік:

Time,Energy
0.021,0.100027815
0.123,8.026088
0.225,0.28435245
0.327,0.039666854
0.429,1.8720028
0.532,0.38208836
0.636,1.0446717
0.739,0.9669831
0.843,9.414629
0.946,4.1523314
1.048,7.6186686
1.151,0.17293897
1.253,0.8642248
1.356,0.7442484
1.459,0.41439438
1.561,0.42799103
1.665,5.9125156
1.769,1.7771379
1.872,4.0950656
1.975,3.252159
2.078,3.1270857
2.18,0.562724

2.283,1.6837981
2.386,0.7386379
2.49,0.47100025
2.594,0.9944869
2.697,1.5732565
2.801,3.4740179
2.904,0.18757373
3.008,4.213422
3.111,0.96400994
3.215,1.9801415
3.318,1.5350153
3.42,4.084865
3.523,1.4701711
3.625,4.7389135
3.728,10.471158
3.831,1.7539304
3.934,1.2657028
4.038,0.94384825
4.142,0.39896616
4.246,0.5475476
4.35,1.5168142
4.453,2.255848
4.556,0.9206775
4.659,1.0829586
4.761,1.0869874
4.864,0.14096802
4.969,1.2392008
5.071,4.515276
5.174,2.4234955
5.277,2.1087313
5.382,0.77020526
5.486,1.0993371
5.589,1.9206206
5.692,2.6706715
5.795,0.3733752
5.899,0.30069223
6.003,1.7106199
6.105,2.2501523
6.209,0.06394396
6.313,6.894244
6.417,5.942265
6.52,0.9298258
6.623,1.0072052
6.727,0.15266912
6.83,0.58157414
6.931,1.394339
7.042,3.5203688
7.144,1.3379918
7.245,1.2032112
7.347,2.3921075

7.449,1.8298824
7.55,0.356964
7.652,14.912053
7.754,7.7161045
7.856,3.708756
7.957,1.0097053
8.058,0.6372194
8.159,0.39182526
8.261,0.14553583
8.364,1.0429358
8.466,0.6243237
8.567,0.8455176
8.669,0.71463853
8.77,0.030900367
8.871,1.1420525
8.973,0.33856708
9.075,0.20946631
9.177,0.14280589
9.281,0.6250943
9.383,0.7612488
9.486,15.521868
9.59,2.0197892
9.695,0.57029086
9.799,6.007503
9.903,0.9143096
10.007,1.0333405
10.111,8.384829
10.214,5.56281
10.317,2.2572732
10.42,1.7841892
10.524,0.3468236
10.628,0.5255378
10.73,0.44392812
10.832,1.8985763
10.936,3.9564483
11.038,1.1558387
11.14,0.422354
11.241,0.5150933
11.343,0.26366007
11.445,0.3519652
11.546,2.638629
11.648,2.6978993
11.75,2.2223442
11.852,0.48470157
11.955,0.14076123
12.059,0.32984644
12.162,1.0773903
12.265,0.66226983
12.368,0.5359045
12.471,2.5120955

12.574,2.208052
12.678,0.3547237
12.78,0.95497924
12.884,1.3458836
12.988,0.11153023
13.091,0.49629903
13.195,0.34502828
13.298,0.83370626
13.399,0.9049009
13.501,5.94099
13.603,0.3962602
13.706,4.080567
13.809,0.47496277
13.912,3.825183
14.015,1.3512514
14.118,0.45814112
14.221,0.30144155
14.323,3.2991319
14.426,2.0551548
14.529,0.91764927
14.632,0.71361655
14.735,0.39580283
14.84,2.740999
14.943,7.6921725
15.046,8.540411
15.148,1.4855998
15.25,1.0955861
15.354,0.9618855
15.459,1.7948903
15.562,0.031412307
15.665,1.3624222
15.768,0.73095536
15.871,0.9075978
15.974,2.4575999
16.077,0.22113252
16.18,0.6043912
16.284,7.0665236
16.387,10.355143
16.49,1.4108722
16.593,2.4796169
16.696,0.5233445
16.798,5.0785522
16.9,0.63015497
17.004,0.85142237
17.106,1.7792554
17.21,1.6659191
17.314,1.9237509
17.417,0.41969514
17.521,0.321966
17.625,5.875937

17.728,4.4058385
17.83,0.8538768
17.932,0.52894956
18.035,0.6390232
18.139,1.2970095
18.243,0.36093864
18.346,0.6023241
18.448,3.7908072
18.551,0.15814912
18.653,0.48344424
18.756,1.2658153
18.86,0.6718413
18.963,6.6285706
19.066,7.1874723
19.17,2.8965921
19.274,0.4217937
19.376,0.5776804
19.479,1.5439409
19.582,0.776349
19.686,0.4617715
19.788,2.334858
19.891,0.06741635
19.994,1.8573875
20.097,0.48052645
20.2,0.18086722
20.302,1.4462363
20.406,8.547481
20.51,1.5840076
20.615,0.6641145
20.718,0.30127338
20.821,0.3771789
20.924,0.3521188
21.028,1.987337
21.131,1.7799547
21.235,3.400616
21.338,1.0438184
21.441,0.05892992
21.544,0.13560715
21.648,0.39947093
21.752,0.12351989
21.855,0.45713854
21.957,0.65164435
22.059,0.7726115
22.161,0.24674948
22.263,0.7471553
22.366,0.8830132
22.466,1.1263378
22.567,0.954215
22.668,0.07508487
22.769,0.99466527

22.87,1.2203462
22.972,1.3830322
23.074,0.9785701
23.177,1.2893388
23.281,2.1997628
23.384,0.62204576
23.487,0.14227301
23.59,0.09320636
23.694,1.0208869
23.798,1.646551
23.901,2.202891
24.005,0.40065372
24.108,1.1048646
24.21,1.3915917
24.311,0.6296986
24.415,0.73952836
24.518,0.24645923
24.62,2.8218966
24.723,0.0894583
24.827,0.33187476
24.93,0.22289144
25.034,0.21866114

Другий графік

Time,Energy
0.029,12.072823
0.13,1.9375606
0.232,0.7108447
0.334,7.4589844
0.437,11.9172325
0.54,3.6885636
0.644,0.3761597
0.746,4.1195703
0.847,19.374947
0.95,55.049545
1.053,14.467688
1.156,31.160316
1.26,10.629786
1.365,5.113558
1.469,5.4512215
1.573,1.929529
1.676,5.815624
1.78,3.4095476
1.883,0.9134847
1.987,14.990656
2.089,1.2831849
2.192,6.893437
2.295,20.648888

2.398,0.09640926
2.5,1.2667465
2.603,1.2988864
2.706,1.6905876
2.811,0.045120664
2.915,1.0083365
3.018,0.29922724
3.121,0.32202503
3.225,0.049152236
3.329,1.529432
3.433,0.9970991
3.535,0.54456174
3.639,0.51529366
3.742,0.48684818
3.846,3.1278038
3.95,0.92519534
4.054,10.288743
4.157,1.2841142
4.261,1.330199
4.364,3.3140433
4.466,3.0887969
4.569,0.92837876
4.672,5.359023
4.776,0.5024381
4.879,0.16624007
4.983,5.3968873
5.086,3.8569112
5.19,0.4961057
5.293,0.537106
5.396,2.852805
5.499,5.639982
5.603,1.423325
5.706,0.7869359
5.809,0.083655775
5.912,0.525452
6.015,1.8224299
6.118,2.3130188
6.22,0.36597875
6.324,1.7945855
6.427,2.6821785
6.531,0.9366424
6.634,4.808185
6.737,2.4723816
6.84,12.097561
6.944,5.559928
7.048,2.4282882
7.151,5.4185424
7.254,2.1931036
7.358,3.1072273
7.461,0.7547515

7.564,0.61721236
7.667,0.4983648
7.77,1.378789
7.873,0.49198988
7.976,1.2623394
8.079,2.1945894
8.182,0.8956989
8.285,0.42796758
8.388,1.6663051
8.495,1.6882911
8.599,1.1007671
8.702,2.8191197
8.805,3.0497217
8.908,2.4103513
9.011,5.917581
9.117,0.4890883
9.221,0.5677241
9.325,0.54646176
9.428,6.2538905
9.531,2.9470181
9.635,0.3617048
9.74,0.7924845
9.843,0.47037348
9.946,1.5925348
10.049,0.8756604
10.153,0.2570469
10.256,0.9372483
10.359,4.0005503
10.465,3.221686
10.568,6.7312007
10.669,7.7142406
10.773,1.984091
10.876,5.772896
10.979,9.772589
11.082,10.301018
11.186,3.4342642
11.289,0.36468393
11.392,0.93573356
11.495,18.293924
11.599,3.833713
11.702,3.5288694
11.805,2.8469663
11.908,4.4889812
12.013,2.5736954
12.117,1.0227532
12.221,4.7807407
12.325,4.0316105
12.428,0.2633045
12.531,3.9129446
12.636,2.6009588

12.738,4.2837963
12.843,1.4563751
12.946,1.6541097
13.048,0.3896231
13.152,4.6459985
13.256,3.5302076
13.359,4.1840963
13.462,0.68875355
13.565,11.997599
13.667,2.3898108
13.77,0.52031314
13.874,0.26482934
13.977,0.7097113
14.08,0.521688
14.183,0.680664
14.286,0.88114524
14.39,0.4691342
14.494,0.34341243
14.597,0.50454825
14.701,1.7498662
14.804,0.7698407
14.907,0.22909002
15.011,1.2524526
15.113,1.3024212
15.216,3.8791375
15.318,2.993278
15.421,2.0108702
15.525,2.126087
15.628,1.4455837
15.732,4.8150563
15.836,0.17501697
15.94,1.4543322
16.043,0.3026689
16.147,1.0220122
16.251,0.5413742
16.354,0.5308045
16.458,0.04133923
16.562,0.29913992
16.665,0.46518427
16.768,0.42580393
16.871,2.0288465
16.975,0.90574723
17.079,0.10542537
17.182,1.4433825
17.286,2.4202847
17.389,1.0015029
17.492,0.40143546
17.596,1.7449205
17.699,4.3070183
17.802,4.0297956

17.904,0.26363897
18.008,1.6696435
18.11,0.5737909
18.214,0.598893
18.317,0.42842716
18.42,0.062195852
18.523,0.5022392
18.626,0.6728624
18.729,1.2118483
18.832,6.678438
18.935,2.4006915
19.039,3.5162182
19.144,2.7146819
19.247,2.1831672
19.35,3.0407593
19.453,0.74707764
19.557,3.1575198
19.66,8.939238
19.764,4.373948
19.867,0.9020432
19.969,0.38385394
20.074,1.1598426
20.178,1.2803402
20.282,1.1330916
20.385,1.5228908
20.49,0.64461505
20.594,0.38375854
20.697,3.965385
20.801,1.5179355
20.904,2.1916173
21.009,4.9842606
21.112,0.29255018
21.216,1.4449505
21.318,2.347852
21.421,2.865471
21.523,4.010035
21.626,0.31236058
21.728,3.4819598
21.832,15.319105
21.935,9.203516
22.038,1.3029474
22.144,2.1434085
22.248,1.6917472
22.351,1.3602116
22.454,1.212436
22.557,0.6965372
22.66,0.5348452
22.764,0.90059364
22.867,1.593477
22.971,0.6619709

23.075,1.9088031
23.178,1.0371062
23.281,1.2723111
23.384,3.6239107
23.487,3.3115215
23.589,3.266818
23.693,0.9674554
23.796,3.526186
23.898,2.869524
24.002,1.562057
24.104,0.5789726
24.207,0.2263072
24.311,0.14981742
24.414,0.3752791
24.517,1.1504253
24.621,0.94968987
24.725,0.69508386
24.829,0.20345576
24.932,0.3032941
25.036,0.246432

ДОДАТОК Г

Аналітична робота із формулою 1.7

Якщо проводиться спостереження за масивом поведінкових даних і як пріорітет стоїть задача якомога швидше помітити аномалію, але при цьому не реагувати на кожну дрібну нетипову «спалахоподібну» зміну, то треба обережно налаштовувати параметри W, α і θ. По-перше, розмір вікна W. Якщо обирається невелике вікно, до прикладу, останні 5 вимірювань, то як тільки один із сигналів раптово зміниться — як-от, час утримання клавіші раптом зросте на 50 % — в наступному обчисленні P_t вже з'явиться ця великопомітна різниця, що швидко підніме метрику $d(P_t,P_0)$ вище порога θ. Як результат, отримане блискавичне виявлення аномалії, але ризик хибних спрацювань відлічень, які просто «виїхали» за межі норми, теж зросте. Якщо ж W збільшити до, скажімо, 50 точок, то один сплеск енергії від руху миші або незвично повільна реакція клавіші будуть «розпилені» серед 49 попередніх значень. У підсумковому P_t їхній вплив згладиться, і лише якщо подібних «сплесків» накопичиться декілька протягом вікна, $d(P_t,P_0)$ почне прямо натякати на проблему. Висновок: велике W дає менше хибнопозитивів, але збільшує час від моменту появи справжньої атаки до її детектування приблизно на W кроків збору даних.

По-друге, коефіцієнт згладжування α. У формулі (1.7) цей коефіцієнт задає, наскільки швидко профіль «підлаштовується» під нові дані. Якщо α=1, то ми ігноруємо історію взагалі, і Pt повністю будується з поточної матриці F_t — система надто чутлива навіть до мінімума змін. Якщо α=0.1, то кожен новий сигнал впливає лише на 10 % від підсумкового P_t, інші 90 % — накопичені раніше дані. Це означає: короткочасні «зриви» в поведінці (наприклад, користувач ненадовго відвів руки від клавіатури, щоб зробити перерву) майже не змінять P_t, тому сеанс не втратиться через дрібні коливання. Але справжня зміна поведінкового патерну, що триває довше, врешті-решт накопичить достатню вагу, щоб $d(P_t,P_0)$ перевищив θ.

Нарешті, поріг θ — це «рівень тривоги», який впровадник Zero Trust сам встановлює: скажімо, 0.6 дає середній баланс «швидка реакція – помірна кількість помилкових спрацювань». Якщо зменшити θ до 0.4, то навіть невелике відхилення в поведінці рівноцінне потенційній атаці, і система раніше детектує тривогу — але частіше помилково. Якщо ж θ підняти до 0.8, то лише суцільні, дуже помітні аномалії викличуть перевірку, і атаки з малим відхиленням можуть пройти непоміченими.

Нехай вимірюєтсья dwell time кожні 0.5 с. Якщо W=10, α=0.2 і θ=0.6, а на одинадцятому кроці dwell підскакує вдвічі відносно базової моделі, то EWMA за один крок зросте приблизно на 0.2×Δ, і тут же може досягти 0.6 лише якщо попередні P_t були вже близько до межі. Але якщо W було 50, то той самий сплеск «розчиниться» в 49 попередніх точках, і для досягнення θ знадобиться кілька таких же сплесків у короткому проміжку. Отже, W і α задають «інерцію» системи — чим більші W і менший α, тим «тяжчий на підйом» профіль, і тим повільніше, але стабільніше він реагує. θ визначає чутливий рівень сигналу, вище якого не розпитується користувач, а просто вживаються заходи. Зважене налаштування цих трьох параметрів дозволяє тонко балансувати між швидкою реакцією на справжню атаку і мінімізацією навантаження на користувача через хибні тривоги.

ДОДАТОК Д

Опис алгоритму RMAC

У схемі використовується малоресурсний блочний шифр (Small Cipher) з блоком 64 біт та двома 64-бітними ключами K_1,K_2, а також двома 64-бітними підключами S_1,S_2. Для обчислення MAC (Message Authentication Code, код автентифікації повідомлення) кожен 64-бітний блок даних X (аналог M_i) обробляється у двоетапній Xor-каскадній схемі Even–Mansour:

$$X_1 = K_1 \oplus X,$$

$$Y_1 = RBOX_{K_1,S_1}(X_1)$$

$$X_2 = (K_1 \oplus K_2) \oplus Y_1$$

$$Y_2 = RBOX_{S_1,K_1}(X_2)$$

$$MAC(X) = Y_2 \oplus K_2$$

Тут RBOX — це проста апаратно ефективна комбінація кількох раундів P-боксу, S-боксу (PRESENT) та циклічних зсувів.

ДОДАТОК Е

Опис алгоритму LCDA

Протокол складається з двох фаз. У першій відбувається взаємна автентифікація між пристроями, в якій використовуються попередні ідентифікатори та первинний ключ E_{init}, після чого CSI зчитується обома сторонами та використовується у HMAC- і XOR-операціях для отримання спільного сесійного ключа SN_{key}. У другій фазі — безперервній автентифікації — обидва пристрої обмінюються динамічними значеннями, сформованими за допомогою функції вигляду $f(t)=t^a+t^b$, де a і b є випадковими показниками, що щоразу змінюються. Перевірка проводиться під шифруванням, використовуючи наявний сеансовий ключ.

У фазі безперервної автентифікації LCDA кожне оновлення довіри містить 4 байти корисного навантаження f(t), 12 байтів IV, 4 байти лічильника та 32 байти HMAC-тегу, усього 52 байт. До цього додаються канальні накладні — 8 байт MAC-заголовка й 4 байти FCS — і в результаті у несучій хвилі міститься близько 64 байти, тобто 64×8=512 біт. Обчислення HMAC-SHA256 на MCU 80 MHz із пропускною спроможністю SHA-256 ≈15 MB/s для 52 B займає приблизно 52 B / 15 000 000 B/s ≈ 3.5 мс за один прохід, а HMAC потребує двох таких проходів, тобто ≈7 мс; якщо врахувати накладні витрати функції, консервативно візьмемо 50 мс (0,05 мс) на весь HMAC. Передача пакета 512 біт по LoRaWAN чи BLE зі швидкістю 250 kbps триває 512 біт / 250 000 біт/с ≈ 2,05 мс. Сумарна затримка на обчислення та передачу оновлення довіри становить близько 2,1 мс.

Для порівняння, повноцінний TLS 1.3 handshake із ECC (secp256r1) і AES-GCM вимагає передачі близько 2 300 B ≈ 18 400 біт, що при 250 kbps дає 18 400 / 250 000 ≈ 73,6 мс лише на передавання, а криптообчислення ECC-обміну на Cortex-M займають додаткові 200–300 мс. У сумі навіть оптимістична оцінка TLS-handshake перевищує 300 мс, тобто майже у два порядки більше, ніж мілісекунда-дві для одного LCDA-повідомлення. Така картина свідчить про те, що LCDA-фаза “оновлення довіри” приносить до 100–1000 оновлень на секунду без суттєвого

навантаження на обмежені ресурси IoT-пристроїв. Звичайно, це обумовлено технічно, всі операції — це лише XOR, HMAC та обробка виміряного CSI. У цьому полягає його перевага перед складнішими схемами, що покладаються на важкі криптографічні функції або машинне навчання.

ДОДАТОК Ж

Опис основних реалізацій HRoT—TPM, Secure Element, ARM TrustZone та HSM

TPM 2.0 пропонує стандартизоване апаратне середовище для зберігання Endorsement Key і генерації Attestation Identity Key, а також виконує гешування вимірювань завантажувача та прошивки в PCR-реєстрах. Перевага: відсутність можливості вивантаження приватних ключів із чіпа, підтримка асиметричної криптографії та апаратного Random Number Generator. Недолік: асиметричні операції (RSA/ECC) часто затримують процес автентифікації на сотні мілісекунд, що неприйнятно в низькошвидкісних середовищах IoT; додатковий чіп збільшує вартість і габарити пристрою .

Secure Element—це сертифікований (EAL5+) модуль із власним CPU та флеш-пам’яттю для ключів, який виконує всі криптографічні операції у «чорному ящику». Він має дуже високу стійкість до фізичного і логічного злому, повну апаратну ізоляція приватних ключів, мінімальну площа коду, що виконується в захищеному середовищі. Однак при цьому існує обмежений набір підтримуваних алгоритмів та фіксована прошивка, складність оновлення та інтеграції в недопустимо легкі MCU-проєкти.

TrustZone ділить SoC на Secure та Normal World, дозволяючи запускати Trusted Execution Environment (TEE) поряд з основною ОС без додаткового чіпа. Переваги: не потребує сторонніх модулів, гнучкість — Secure World може оновлюватися OTA, добре масштабується на різних платформах Cortex-A/M. Недолік: ізоляція базується на софтових механізмах MMU/MPU та SMC-викликах, що робить її потенційно вразливою до side-channel і speculative attacks; потребує ретельного розділення коду та довіреного монітора.

HSM—це серверне або gateway-рішення з високою продуктивністю й сертифікацією FIPS 140-2/3, призначене для агрегування криптозапитів. Включає підтримку тисячі транзакцій шифрування/підпису за секунду, захищене зберігання master-ключів, аудит подій і фізична оборона проти вторгнень. Серед

недоліків виділяють високу вартість, мережеві затримки при зверненні з IoT-пристроїв безпосередньо до віддаленого HSM; зазвичай використовується лише на edge-gateway, а не на «тлінних» сенсорах.

ДОДАТОК И

Аналітична оцінка впливу навантаження

Крім чистої латентності, зростає й обсяг трафіку. При відсотку кеш-хітів 60 % у середньому припадає 0,4 OCSP-запити на сесію, а якщо навантаження сягне пікових 50 000 паралельних сесій – загальна кількість додаткових повідомлень лише для перевірки сертифікатів перевищить 20 000 запитів у секунду . Це створює вузьке місце не лише на лініях зв’язку, а й у самому складі мережевих пристроїв, які обслуговують такі хвилі авторизаційних повідомлень.

Аналітичну оцінку впливу навантаження на затримки можна провести через просту модель черги M/M/1, де інтенсивність надходження λ відповідає сумарному числу перевірок, а μ – пропускній здатності сервісу авторизації. Середній час відповіді в такій системі розраховується як:

$$T = \frac{1/\mu}{1-\rho}, \quad \rho = \frac{\lambda}{\mu} \tag{3.1}$$

де μ — середня пропускна здатність сервісу (запитів за секунду), а λ — інтенсивність надходження запитів. Ключовим тут є знаменник $1-\rho$: він показує, наскільки залишкової потужності у системи залишається після обробки поточного навантаження. При малих завантаженнях ($\rho \ll 1$) значення $1-\rho$ близьке до одиниці, і середній час відповіді практично дорівнює чистому часу обробки $1/\mu$. Проте у міру того, як λ наближається до μ, знаменник стрімко зменшується і, відповідно, латентність росте за гіперболічним законом: під час 50 % завантаження ($\rho=0,5$) затримка буде вдвічі більшою, ніж базове $1/\mu$; при 80 % ($\rho=0,8$) вона вже в п’ять разів перевищує чистий час обробки, а при 90 % ($\rho=0,9$) — удесятеро більша.

Саме така експоненційна чутливість робить компоненти Zero Trust Architecture особливо вразливими до короткотривалих піків запитів: кілька десятків відсотків зайвого навантаження ведуть до кратного зростання затримок, що неприпустимо для інструментів реального часу. Водночас модель M/M/1 враховує

лише один потік обслуговування та експоненційні інтервали між запитами, тому в реальних системах з багатопоточністю, кешуванням та мережевими затримками цей ефект може бути ще гострішим.