Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ БЕЗПЕКОВИХ МЕХАНІЗМІВ АРХІТЕКТУРИ RISC-V ДЛЯ IOT-СЕРЕДОВИЩ У ПОРІВНЯННІ З ARM TRUSTZONE
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-1
Грек Є. С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні наук
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник ст. викл. Путятіна О. Є.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Греку Єгору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження безпекових механізмів архітектури RISC-V для IoT-середовищ у порівнянні з ARM TrustZone

затверджена наказом університету від 14 листопада 2025 року № 1045Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 28 листопада 2025 р.

3. Вихідні дані до роботи специфікації архітектур RISC-V та ARM TrustZone,літературні джерела щодо впровадження TEE в IoT-середовищах, програмні засоби для моделювання та аналізу, програмну основу дослідження у вигляді проєкту mTower, методичні підходи для оцінки та порівняння критеріїв безпеки, продуктивності та накладних витрат обох архітектур.

4. Перелік питань, що потрібно опрацювати в роботі
1.Аналіз сучасних механізмів апаратної безпеки та ізоляції (TEE) в IoT-системах архітектур ARM та RISC-V.
2.Аналіз літературних джерел щодо архітектурних особливостей та методів захисту пам'яті PMP.
3.Портування ядра mTower та налаштування PMP, обробників системних викликів ECALL.
4.Візуалізація схеми розподілу пам'яті та логіки перемикання контекстів між захищеним та незахищеним світами.
5.Розробка програмного забезпечення, що надасть змогу розгорнути TEE mTower на RISC-V та експериментально порівняти рівень безпеки з ARM TrustZone.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність теми, об'єкт та мета дослідження, постановка задачі, схеми моделей захисту та архітектурної ізоляції в ARM TrustZone та RISC-V, структура реалізованої системи mTower та схема розподілу пам'яті, результати тестування механізмів PMP та системних викликів, порівняльний аналіз ефективності та захищеності досліджуваних архітектур, матриця прийняття рішення щодо вибору архітектури, висновки, рекомендації та апробація роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Особливості методів безпеки RISC-V та ARM	15.10.25-20.10.25	
5	Дослідження методів безпеки RISC-V та ARM	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	19.11.25-10.12.25	
10	Перевірка на плагіат	20.11.25-10.12.25	
11	Рецензування	21.11.25-10.12.25	
12	Підготовка презентації та доповіді	21.11.25-22.12.25	
13	Занесення роботи в електронний архів	21.11.25-22.12.25	
14	Попередній захист кваліфікаційної роботи	01.12.25-22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Путятіна О. Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 94 с., 11 табл., 4 рис., 2 дод., 44 джерела.

БЕЗПЕКОВІ МЕХАНІЗМИ, ІЗОЛЯЦІЯ СЕРЕДОВИЩ, МИКРОКОНТРОЛЕРИ, ARM TRUSTZONE, IOT, KEYSTONE, MTOWER, PMP, RISC-V, TRUSTED EXECUTION ENVIRONMENT.

Об’єктом дослідження є TEE у вбудованих та IoT-системах.

Метою дослідження є розроблення та оцінка портування TEE рішення mTower з ARM на RISC-V для порівняння ефективності та захищеності.

Використано методи порівняльного аналізу архітектур ARM та RISC-V, моделювання ізольованих середовищ виконання у QEMU, експериментальне тестування механізмів PMP. Проведено аналіз сучасних підходів до побудови TEE.

Наукова новизна дослідження полягає у портуванні мінімалістичного TEE для вбудованих та IoT-систем на архітектуру RISC-V, що підтвердило можливість забезпечення рівня безпеки, співставного з ARM TrustZone.

Робота пов’язана з дослідженнями у сфері побудови TEE для відкритих архітектур та спирається на результати проєктів Keystone і mTower, що використовуються як базові фреймворки для аналізу та порівняння із ARM TrustZone.

Отримані результати можуть бути використані під час подальшої розробки та адаптації довірених середовищ виконання на базі архітектури RISC-V, а також у навчальних і науково-дослідних цілях для порівняльного аналізу механізмів безпеки вбудованих систем.

У результаті дослідження виконано портування ядра mTower на архітектуру RISC-V, проведено порівняльний аналіз із ARM TrustZone у середовищі QEMU.

ABSTRACT

Explanatory note to the qualification work: 94 pages, 11 table, 4 figures, 2 appendix, 44 sources.

ARM TRUSTZONE, ENVIRONMENT ISOLATION, IOT, KEYSTONE, MICROCONTROLLERS, MTOWER, PMP, RISC-V, SECURITY MECHANISMS, TRUSTED EXECUTION ENVIRONMENT.

The object of the research is TEE in embedded and IoT systems.

The aim of the research is the porting of the mTower TEE solution from ARM to RISC-V in order to compare the efficiency and security.

The methods of comparative analysis of ARM and RISC-V architectures, modeling of isolated execution environments in QEMU, and experimental testing of PMP. The analysis of modern approaches to TEE design and relevant scientific sources was carried out.

The scientific novelty of the research lies in porting a minimalist TEE designed for embedded and IoT systems to the RISC-V architecture, confirming the possibility of achieving a security level comparable to ARM TrustZone.

The interconnection with other works is determined by the research direction in the field of open-architecture TEE development and is based on the results of the Keystone and mTower projects, which serve as reference frameworks for comparison with ARM TrustZone.

The recommendations for using the results of the work include their application in further development and adaptation of TEE solutions based on RISC-V, as well as in educational and research purposes for the comparative study of embedded system security mechanisms.

As a result of the research, the mTower core was successfully ported to the RISC-V architecture, and a comparative analysis with ARM TrustZone was performed in the QEMU environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES – Advanced Encryption Standard (розширений стандарт шифрування)

API – Application Programming Interface (інтерфейс прикладного програмування)

ARM – Advanced RISC Machines (розширена RISC-машина)

ATF – ARM Trusted Firmware (довірена прошивка ARM)

CLI – Command Line Interface (інтерфейс командного рядка)

Clock Glitching (збій тактування)

CSR – Control and Status Registers (регістри керування та стану)

DMC – Direct Memory Access (прямий доступ до пам’яті)

EL – Exception Levels (рівні привілеїв)

FI – Fault Injection (ін’єкція несправностей)

IoT – Internet of Things (інтернет речей)

Load Access Fault (збій доступу при читанні)

MPU – Memory Protection Unit (модуль захисту пам’яті)

M-mode – Machine Mode (машинний режим)

NPU – Neural Processing Unit (нейронний процесор)

NSC – Non-Secure Callable (доступний для виклику з незахищеного режиму)

NW – Normal World (звичайний світ)

OP-TEE – Open Portable Trusted Execution Environment (відкрите портативне довірене середовище виконання)

OSH – Open-Source Hardware (відкрите апаратне забезпечення)

PMP – Physical Memory Protection (фізичний захист пам’яті)

REE – Rich Execution Environment (повнофункціональне середовище виконання)

RISC – Reduced Instruction Set Computer (обчислення зі скороченим набором команд)

SAU – Security Attribution Unit (модуль визначення безпеки)

SCA – Side Channel Attack (атака сторонніми каналами)

SG – Secure Gateway (захищений шлюз)

SHA – Secure Hash Algorithm (алгоритм безпечного хешування)

SM – Secure Monitor (захищений монітор)

SMC – Secure Monitor Call (виклик захищеного монітора)

S-mode – Supervisor Mode (режим супервізора)

SoC – System on a Chip (система на кристалі)

SW – Secure World (безпечний світ)

TA – Trusted Applications (довірені застосунки)

TCB – Trusted Computing Base (довірене обчислювальне ядро)

TEE – Trusted Execution Environment (довірене середовище виконання)

U-mode – User Mode (режим користувача)

Voltage Glitching (збій напруги)

ВСТУП

Бурхливий розвиток IoT, хмарних сервісів і мобільних обчислень зумовлює постійне зростання обсягу чутливої інформації, яку потрібно обробляти в небезпечних середовищах.

TEE став одним із ключових механізмів апаратно-програмної ізоляції, що дозволяє виконувати критичний код у захищеному контексті, відокремленому від решти операційної системи [1–3].

На ринку переважає реалізація ARM TrustZone, однак поява відкритої архітектури RISC-V і ініціатива OpenTitan створили запит на альтернативні, повністю репродуковані апаратні платформи [4, 5].

Проблема портування існуючих TEE на RISC-V є актуальною, оскільки забезпечує незалежність від пропрієтарних рішень, а також дозволяє уніфікувати підходи до безпеки в секторі OSH [6–8].

Крім технічного інтересу, тема має стратегічне значення для інформаційної безпеки державного та промислового рівнів; відкритість специфікацій RISC-V полегшує аудит та сертифікацію, що є критично важливим у галузях із підвищеними вимогами до довіри [9]. Тому дослідження порту TEE на RISC-V, а особливо порівняльний аналіз із ARM, є науково та практично актуальним.

Завданням роботи є розроблення та оцінка методики портування проєкту mTower [8] TEE на архітектуру RISC-V з подальшим порівнянням безпекових механізмів та ефективності з ARM TrustZone.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасний стан досліджень у галузі TEE, класифікувати загрози та вимоги;

– дослідити апаратні засоби безпеки RISC-V і ARM у контексті ізоляції та контролю доступу;

– спроєктувати ланцюг завантаження та модель пам’яті для mTower на RISC-V;

– реалізувати пілотний порт mTower;

– провести функціональне та безпекове тестування;

– сформулювати рекомендації щодо подальшого розвитку проєкту та узагальнити результати порівняльного аналізу.

Робота обмежується 64-розрядними ядрами RISC-V (RV64GC) та ARMv8-A (A-клас), а також використанням відкритого програмного стеку (mTower, Linux Kernel, U-Boot). Поза рамками розгляду залишаються апаратні акселератори криптографії, не передбачені базовим SoC, та питання комерційної ліцензованої сертифікації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Trusted Execution Environment: концепція, загрози та вимоги

TEE – це ізольоване середовище всередині основної системи, яке забезпечує захищене виконання коду та зберігання критично важливих даних. На відміну від REE, TEE має власні механізми доступу до пам’яті, апаратні ізоляційні механізми та контрольований інтерфейс взаємодії з зовнішнім світом [9].

TEE часто реалізується як апаратно-програмне розширення, наприклад TrustZone для ARM [1], чи Keystone для RISC-V [6]. Його ключові функції:

– ізоляція – код та дані в TEE відокремлені від решти системи;

– цілісність – контроль виконання для запобігання модифікаціям;

– конфіденційність – захист чутливих даних (ключів, біометрії, сертифікатів);

– довірене завантаження – можливість перевіряти підпис ПЗ під час старту.

Таким чином, TEE є ядром TCB, на якій будуються безпечні додатки, такі як мобільні платежі, DRM чи IoT-захист [7, 8].

Попри високий рівень захисту, TEE залишається вразливим до низки загроз. Одним з найбільш небезпечних напрямів атак є апаратні побічні канали: витоки через кеш, час виконання чи навіть електромагнітне випромінювання, що дозволяють обійти логічну ізоляцію [10].

Критичною вимогою є забезпечення безпечного завантаження, що гарантує виконання лише перевіреного та підписаного коду. Це дозволяє створювати ланцюг довіри від апаратури до прикладного ПЗ [11].

Програмні атаки можуть використовувати помилки у прошивці, драйверах чи гіпервізорі для отримання доступу до довіреного середовища [9].

Додатково небезпеку становлять неправильно спроєктовані інтерфейси взаємодії між REE і TEE, які можуть дати можливість атакуючому обійти механізми контролю доступу [12].

Окремим викликом є розробка TCB. Помилка у його коді може поставити під загрозу всі довірені застосунки, що на ньому виконуються [13]. Є дослідження, що показують як сучасні TEE часто не враховують атаки, пов’язані з відсутністю захисту від витоків інформації через планувальник, кеш-пам’ять та таймінги доступу [14].

Щоб протистояти зазначеним загрозам, TEE має будуватися на основі низки вимог, що охоплюють як апаратний, так і програмний рівні .

Першочерговим завданням є мінімізація TCB. Чим менший обсяг довіреного коду, тим легше його перевірити та формально верифікувати [9, 15]. Важливим принципом є наявність апаратної підтримки ізоляції. Прикладом є поділ системи на SW та NW у TrustZone для ARM [1, 3, 10] або використання фізичного захисту пам’яті завдяки PMP в RISC-V, який дозволяє задавати політики доступу на рівні регіонів пам’яті [4, 13].

У сучасних TEE також необхідно враховувати побічні канали. Окрім класичного захисту від атак по енергоспоживанню [16], все більшої уваги набуває необхідність захисту від атак за часом, яка вимагає нового контракту між апаратним та програмним забезпеченням.

Не менш значущим є використання криптографічних механізмів для автентифікації компонентів, шифрування даних та створення захищених каналів обміну ключовою інформацією [2]. Водночас сучасні системи потребують архітектурної гнучкості, TEE має підтримувати різні моделі, від апаратно орієнтованих, таких як ARM TrustZone, до легковагових рішень для IoT, таких як Keystone чи mTower [6, 8]. Лише поєднання цих підходів дозволяє створити довірене середовище, яке є не лише ізольованим, а й стійким до складних атак у довгостроковій перспективі.

Таблиця 1.1 – Загрози та контрзаходи для TEE

Загроза	**Опис**	**Контрзаходи**
1	2	3
Апаратні атаки через побічні канали [16, 17]	Витік інформації через кеш, енергоспоживання, час виконання або електромагнітне випромінювання	Використання методів маскування, рандомізація, захист від атак за часом, контроль доступу до кешів
Програмні атаки на TEE [9, 14]	Експлуатація вразливостей у прошивці, драйверах або гіпервізорі	Мінімізація TEE, регулярний аудит коду, верифікація
Атаки через інтерфейси між REE і TEE [12, 14]	Некоректно спроєктовані API можуть дозволити обхід ізоляції	Обмеження інтерфейсів, перевірка вхідних даних, формальні методи верифікації
Компрометація TCB [9, 14]	Навіть невелика помилка в базовому коді ставить під загрозу всі довірені застосунки	Зменшення TCB, модульна архітектура, використання верифікованих мікроядер
Відсутність захисту від атак за часом [18]	Витік інформації через планувальник, кеш-пам’ять та таймінги доступу	Новий апаратно-програмний контракт, планування з ізоляцією часу
Компрометація завантаження системи [11]	Завантаження модифікованого або шкідливого ПЗ у TEE	Secure Boot, криптографічна перевірка підписів, ланцюг довіри
Недостатня ізоляція пам’яті [2, 3]	Атакуючий може обійти бар’єри між REE та TEE	Використання апаратних механізмів (TrustZone у ARM, PMP у RISC-V)

Продовження таблиці 1.1

1	2	3
Компрометація даних і ключів [2]	Викрадення секретної інформації у процесі зберігання чи передачі	Використання криптографічних механізмів (AES, RSA, ECC), захищених каналів
Обмежена архітектурна гнучкість [6]	Традиційні TEE можуть бути занадто важкими для IoT чи MCU	Легковагові TEE (Keystone, mTower) з оптимізацією під ресурсні обмеження

1.2 Сучасний стан досліджень Trusted Execution Environment

Сучасні дослідження TEE зосереджені на поєднанні високого рівня безпеки з ефективністю та масштабованістю у різних обчислювальних середовищах – від мобільних пристроїв до IoT і периферійних систем.

Найбільш вивченою архітектурою в контексті TEE залишається ARM, яка широко використовується у мобільних і вбудованих пристроях. Її перевага полягає у поділі системи на SW та NW, що дозволяє розмежувати виконання довірених і звичайних застосунків.

Дослідження акцентують увагу на архітектурних перевагах ARM TrustZone та водночас відзначають низку обмежень. Середних можно виділити складність у формальній верифікації коду, ризики, пов’язані з неправильно спроєктованими API між REE та TEE, а також вразливість до побічних каналів [2, 3, 12]. Сучасні роботи спрямовані на вдосконалення механізмів безпеки ARM TrustZone і створення додаткових рівнів захисту, здатних зменшити ризики атак.

На архітектурі RISC-V активно розвивається відкритий фреймворк Keystone, який використовується як дослідницька платформа для створення легковагових TEE. Keystone застосовує PMP для ізоляції пам’яті та дозволяє

будувати модульні конфігурації з мінімальним TCB. Це робить його особливо привабливим для академічних і промислових експериментів, адже відкритість RISC-V сприяє перевірці безпеки та адаптації TEE під специфічні потреби [6, 13, 19].

У сфері IoT та мікроконтролерів спостерігається тенденція до створення максимально легких рішень TEE. Прикладом є проєкт mTower, розроблений як open-source TEE для мікроконтролерів. Він реалізує ізоляцію даних та середу виконання, та управління ключами у пристроях з обмеженими ресурсами. Такі підходи відкривають перспективи застосування TEE у великій кількості сценаріїв під IoT, де класичні архітектури, такі як TrustZone, є занадто важкими.

Окремим напрямом є дослідження балансу між продуктивністю та безпекою. Роботи [14] показують, що надмірне посилення ізоляції може суттєво впливати на швидкодію, тому сучасні TEE орієнтуються на гнучкі конфігурації, де рівень захисту підбирається відповідно до конкретного сценарію використання.

Ще однією актуальною темою є проблема захисту від атак за часом. Як зазначено в [20], існуючі TEE часто не враховують витоки інформації через часові характеристики виконання, що робить актуальним створення нового апаратно-програмного контракту для протидії таким атакам.

Таким чином, сучасний стан досліджень TEE можна описати як розвиток у трьох ключових напрямах

– удосконалення традиційних рішень (TrustZone);

– розробка відкритих і гнучких платформ (Keystone, mTower);

– оптимізація для ресурсно обмежених середовищ.

Усі ці напрями мають спільну мету – підвищення стійкості довірених середовищ до нових класів атак при збереженні продуктивності та доступності технології.

1.3 Проблематика портованості TEE між різними архітектурами

Проблема портованості TEE між різними апаратними архітектурами є однією з ключових у сучасних дослідженнях безпеки. Найбільш актуальною вона постає у контексті переходу від ARM TrustZone, яка є де-факто стандартом у мобільних і вбудованих системах, до відкритої архітектури RISC-V, де активно розвиваються рішення такі, як Keystone та mTower [2, 6, 8].

Основна складність полягає у тому, що ARM і RISC-V реалізують різні моделі ізоляції. TrustZone базується на поділі системи на два середовища – SW та NW, де апаратні механізми (включаючи MMU, SAU, інтерфейси периферії та контролери переривань) подвоюються і маркуються як захищені чи не захищені [1, 3, 12].

У той же час у RISC-V аналогічна ізоляція реалізується через PMP та розділення рівнів привілеїв, що вимагає іншої логіки управління пам’яттю та інтерфейсами [4, 13]. Це означає, що програмні моделі TrustZone не можна безпосередньо перенести на RISC-V без суттєвої адаптації.

Ще одним викликом є відмінність у завантаженні довірених компонентів. У екосистемі ARM використовується ATF, яка виконує роль початкового завантажувача для SW [17]. У RISC-V цю функцію може виконувати OpenSBI, що має зовсім іншу архітектуру та API [4].

Тому портованість рішень для безпечного завантаження вимагає переписування кодової бази, адаптації криптографічних бібліотек і забезпечення сумісності з новими механізмами довіри [11].

Суттєву проблему становить також різниця в екосистемах. ARM TrustZone підтримується широким спектром готового ПЗ, включаючи OP-TEE, який інтегрується з Linux і використовується у промислових рішеннях [7]. Для RISC-V подібна екосистема лише формується. Keystone [6] надає дослідницьку платформу, але вона має значно обмеженішу підтримку інструментів і потребує активної адаптації прикладних компонентів. Це робить

процес міграції трудомістким і вимагає залучення розробників як до низькорівневих (SBI, PMP), так і до високорівневих (TEE OS, API) шарів.

Окремим викликом є портованість IoT. ARM пропонує TrustZone-M для Cortex-M процесорів, що робить можливим застосування TEE у мікроконтролерах [18]. У RISC-V аналогічну роль виконують легковагові рішення наприклад mTower [8], але їх архітектура значно відрізняється від ARMv8-M, що унеможливлює пряме перенесення застосунків. Потрібна адаптація під інший набір інструкцій, модель доступу до пам’яті та механізми викликів API.

Таким чином, портованість TEE з ARM на RISC-V ускладнюється трьома групами факторів:

– архітектурні відмінності (TrustZone Worlds проти RISC-V PMP);

– різні моделі завантаження та ініціалізації довіреного середовища (ATF проти OpenSBI);

– нерівномірність зрілості екосистем (зріла OP-TEE на ARM проти ще експериментального Keystone/mTower на RISC-V).

Незважаючи на це, відкритість RISC-V і наявність таких проєктів, як Keystone та mTower, створюють передумови для формування уніфікованих стандартів TEE, які у перспективі можуть забезпечити кращу переносимість застосунків між різними архітектурами [6, 8, 15].

1.4 ARM TrustZone: модель безпеки та реалізація

ARM TrustZone є однією з найбільш поширених технологій побудови TEE у мобільних і вбудованих системах. Її модель безпеки базується на поділі всієї системи на два світи:

– SW – виконує довірене програмне забезпечення, наприклад TEE OS та TA;

– NW – виконує звичайні операційні системи та застосунки, наприклад Android чи Linux.

Ключова ідея TrustZone полягає в тому, що апаратні ресурси – ядра процесора, пам’ять, периферія та навіть контролер переривань – можуть динамічно маркуватися як належні до SW або NW. Це дозволяє розмежувати доступ і забезпечує, що код у NW не має змоги порушити цілісність SW [2, 3].

Модель безпеки TrustZone реалізується через такі механізми:

– поділ привілеїв – використовується спеціальний біт NS, який визначає, у якому світі виконується код;

– розширення інструкцій – доступ до SW здійснюється через інструкцію SMC, яка передає управління монітору безпеки;

– ізоляція пам’яті – контролери пам’яті та MMU підтримують розділення адресного простору для SW і NW;

– контроль периферії – доступ до пристроїв може обмежуватися апаратно, щоб запобігти витокам через небезпечні канали [12].

Для ініціалізації SW ARM пропонує ATF, яка реалізує монітор безпеки та забезпечує запуск довірених компонентів у SW [17]. На основі цього середовища зазвичай працює OP-TEE OS, яке реалізує стандартизовані API для TA та інтегрується з основними ОС у NW [7].

Варто зазначити, що TrustZone реалізований як для повнофункціональних процесорів з підтримкою багатоядерності і віртуалізації таких як ARMv8-A, так і для ARMv8-M, орієнтованих на мікроконтролери. У випадку ARMv8-M TrustZone дозволяє поділ пам’яті на безпечні й небезпечні регіони навіть у середовищах із дуже обмеженими ресурсами [18]. Це робить технологію універсальною для мобільних пристроїв, IoT та вбудованих систем.

Таким чином, модель безпеки ARM TrustZone поєднує апаратний поділ ресурсів, централізований монітор безпеки та стандартизовану програмну інфраструктуру. Завдяки зрілій екосистемі TrustZone стала основою для

більшості сучасних мобільних і вбудованих платформ, що вимагають апаратного TEE [2, 3, 7, 12, 17, 18].

1.5 Безпекові розширення RISC-V

Відкритість архітектури RISC-V дозволила розробникам створювати власні механізми безпеки, які забезпечують можливість побудови TEE без необхідності використання закритих рішень. Основу цієї моделі становлять розширення привілейованих інструкцій і механізми контролю пам’яті, описані у специфікації Privileged Architecture [4].

PMP є ключовим апаратним механізмом захисту у RISC-V. Він дозволяє ядру у M-mode визначати регіони фізичної пам’яті та задавати політики доступу (читання/запис/виконання). Завдяки цьому можна обмежити доступ програм у нижчих режимах привілеїв (S-mode та U-mode) до критичних ділянок пам’яті. Дослідження показують, що правильно сконфігурований PMP є основою для побудови TEE на RISC-V, забезпечуючи апаратну ізоляцію [13].

RISC-V підтримує класичну багаторівневу модель привілеїв:

– M-mode – найвищий рівень, контроль апаратних ресурсів, конфігурація PMP;

– S-mode – виконується ядро ОС, керує сторінками пам’яті, інтерфейсом з апаратурою;

– U-mode – звичайні застосунки, без прямого доступу до апаратних ресурсів [4].

Подібна ієрархія дозволяє реалізовувати принцип мінімальних привілеїв та обмежувати поверхню атак. Для TEE зазвичай використовують S-mode як рівень виконання довіреного ОС (аналог OP-TEE), у той час як M-mode виконує роль монітора безпеки.

У проєктах, орієнтованих на TEE, вводиться додаткове поняття SM. Це компонент, який працює в M-mode і виконує функції, подібні до ARM Secure Monitor:

– створює та ізолює довірені домени виконання;

– керує викликами з незахищеного середовища;

– конфігурує PMP для виділення безпечних зон [6, 13].

Таким чином, SM у контексті RISC-V – це не новий рівень привілеїв, а архітектурний патерн використання M-mode для ізоляції.

Ще одним важливим компонентом є SBI, що визначає стандартизований інтерфейс між M-mode та S-mode. Через нього ОС у S-mode виконує системні виклики до прошивки в M-mode (наприклад OpenSBI). Це дозволяє реалізувати функції безпечного завантаження та управління ресурсами для TEE [4].

Поєднання PMP, M/S/U-mode та SBI створює фундамент для реалізації довірених середовищ у RISC-V. Keystone [6] та інші проєкти демонструють, що ці розширення дозволяють формувати ізольовані середовища виконання, аналогічні ARM TrustZone, але з більшою архітектурною гнучкістю. Дослідження підтверджують, що конфігурація PMP безпосередньо впливає на продуктивність та безпеку систем, тому оптимальний баланс між ними залишається відкритою проблемою [13].

1.6 Огляд існуючих TEE-реалізацій

У практичній реалізації TEE сьогодні домінують три підходи: OP-TEE для ARM, Keystone для RISC-V та mTower для мікроконтролерів. Вони різняться за архітектурою, цілями і середовищем застосування, але мають спільну мету – створення ізольованого простору для виконання критично важливого коду.

OP-TEE є найпоширенішим TEE для ARM TrustZone [7]. Воно реалізує стандартизовані API (GlobalPlatform TEE Internal Core API та Client API), що дозволяє запускати TA незалежно від конкретного апаратного забезпечення. OP-TEE виконується у SW, тоді як основна ОС (наприклад, Linux або Android) працює у NW.

OP-TEE має модульну архітектуру, включає ядро TEE OS, інтерфейс взаємодії з REE та інструменти для розробників. Завдяки відкритому коду він використовується як у наукових дослідженнях, так і в промислових рішеннях. Його зрілість та широка підтримка роблять його стандартом де-факто для ARM-платформ [2, 3, 7].

Keystone – це відкрита дослідницька платформа TEE для архітектури RISC-V [6]. Вона побудована на використанні PMP для ізоляції пам’яті та SM, що виконується у M-mode та створює ізольовані середовища. Keystone має модульну структуру, користувацькі ізольовані програми, бібліотеки для взаємодії з REE та компонент SM, що конфігурує PMP.

Ключова перевага Keystone полягає у відкритості коду та можливості налаштування. Розробники можуть експериментувати з різними політиками безпеки, мінімізувати TCB та підібрати необхідний компроміс між продуктивністю й захищеністю [13]. Це робить Keystone платформою, орієнтованою на академічні дослідження і майбутні промислові впровадження у RISC-V-систем.

mTower – це open-source TEE, спеціально розроблений для мікроконтролерів. На відміну від OP-TEE та Keystone, які орієнтовані на повноцінні процесори з MMU, mTower працює у середовищах із сильними обмеженнями по ресурсах. Він забезпечує базові принципи TEE: ізоляцію виконання, захищене завантаження та управління ключами, але при цьому зберігає низьке споживання пам’яті й енергії.

mTower дозволяє переносити концепції TEE у сферу IoT, де потрібен захист комунікацій, автентифікація пристроїв та захищене зберігання даних у мікроконтролерах. Завдяки цьому mTower розглядається як практичне рішення

для інтеграції TEE у масові IoT-системи, де ARM TrustZone або Keystone можуть виявитися надмірно важкими.

Таким чином, OP-TEE представляє зріле промислове рішення для ARM, Keystone – дослідницьку платформу для RISC-V, а mTower – легковагове TEE для IoT і MCU. Разом вони охоплюють основні напрями розвитку сучасних TEE: від мобільних пристроїв до відкритих архітектур та ресурсно обмежених систем.

1.7 Висновки та постановка проблеми портування

Сучасні дослідження у сфері інтелектуальних систем обробки даних демонструють інтенсивний розвиток алгоритмів аналізу, класифікації та оптимізації обчислень. Роботи [21–28] присвячені зменшенню обчислювальних витрат, структурній компресії описів, сегментації зображень та вдосконаленню методів глибинного й структурного розпізнавання. Такі підходи забезпечують високу ефективність, але водночас створюють нові виклики щодо безпеки виконання обчислень і захисту даних у розподілених та граничних середовищах.

Особливу увагу дослідники приділяють реальним часовим обмеженням та стійкості до шумів у потокових даних [22, 29], а також роботі з ієрархічними або кластерними структурами ознак [23, 24, 30]. Ці завдання часто реалізуються безпосередньо на IoT-пристроях, де відсутність надійної апаратної ізоляції може призвести до витоків або модифікації критичних параметрів моделі.

Такі обчислення як виконання арифметичних операцій з використанням інтервального аналізу [31], моделювання невизначеності в технічних системах [29], а також стохастичні фінансові моделі з шумовими складовими [25–29] потребують гарантованої цілісності даних, що може бути забезпечено лише завдяки апаратним засобам контролю доступу й ізоляції.

У прикладних доменах, таких як від біомедичної аналітики [32], керування транспортом природного газу [33] і кібербезпеки мережевих систем [34] – безпека стає системоутворюючим чинником. Дослідження у сфері захисту інформації, включно з нечіткими моделями оцінювання ризиків для мобільних застосунків [35], підтверджують необхідність поєднання програмних і апаратних рівнів захисту.

Роботи [26–28, 36–38] підкреслюють, що навіть у задачах відеоаналітики, підрахунку відвідувачів або класифікації кулінарних зображень постає питання приватності користувацьких даних і довіри до середовища виконання. У цьому контексті апаратні механізми безпеки мають забезпечувати ізоляцію контекстів виконання, захист буферів пам’яті та контроль привілеїв.

Окрему групу становлять роботи, присвячені архітектурним аспектам апаратної безпеки в IoT-системах [23, 39]. У них показано, що традиційні підходи ARM TrustZone не можуть бути безпосередньо перенесені на відкриту архітектуру RISC-V через відмінності у моделі привілеїв, механізмах PMP/ePMP та організації пам’яті. Це формує наукову проблему портування TEE – створення еквівалентного середовища довіреного виконання на базі RISC-V з урахуванням її модульної структури та можливостей апаратного розмежування світів.

Таким чином, ураховуючи широкий спектр сучасних досліджень – від оптимізації обчислень [21–28], прикладних інтелектуальних систем [31–42] до безпосередніх досліджень архітектурної безпеки [34, 43], можна зробити висновок, що питання створення безпечного середовища виконання для відкритих архітектур є актуальним і потребує реалізації у вигляді портованого TEE для RISC-V.

Проблема портованості TEE між різними апаратними архітектурами є однією з ключових у сучасних дослідженнях безпеки. Найбільш актуальною вона постає у контексті переходу від ARM TrustZone, яка є де-факто

стандартом у мобільних і вбудованих системах, до відкритої архітектури RISC-V, де активно розвиваються рішення такі як Keystone [2, 6, 8].

Портованість TEE з ARM на RISC-V ускладнюється трьома групами факторів:

– архітектурні відмінності, у ARM використовується модель поділу системи на SW та NW, тоді як у RISC-V механізм ізоляції реалізується через PMP та багаторівневу систему привілеїв, це вимагає перепроєктування логіки безпеки замість прямого перенесення коду [2, 4, 13];

– відмінності у завантаженні та початковій ініціалізації, у ARM застосовує ATF для завантаження SW [17], тоді як у RISC-V цю роль виконує OpenSBI [4, 11], це вимагає адаптації процедури безпечного завантаження та створення ланцюга довіри;

– різна модель обробки переривань та викликів, у ARM доступ до SW здійснюється через інструкцію SMC, яка передає керування монітору безпеки, у RISC-V подібну роль виконує SM у M-mode, який конфігурує PMP і реалізує виклики з REE до ізольованих середовищ, це вимагає переробки викликів системних API та механізмів взаємодії між світами [2, 6, 13];

– відмінності в управлінні пам’яттю, ARM має інтегровану підтримку поділу пам’яті на захищені та незахищені регіони в MMU/MPU [2, 3, 12], тоді як у RISC-V це реалізується через PMP і сторінкову адресацію, що потребує глибокої адаптації менеджера пам’яті mTower під нову логіку;

– відсутність стандартизованих API для TEE у RISC-V, у ARM існують специфікації GlobalPlatform, які підтримуються OP-TEE і широко використовуються у промисловості [7], тоді як у RISC-V подібна стандартизація ще формується, це означає, що порт mTower потребуватиме не лише технічної адаптації, а й визначення сумісного API для TA.

Таким чином, головна складність портованості mTower на RISC-V полягає не у незрілості самого рішення, а у фундаментальних архітектурних розбіжностях між ARM та RISC-V, різних моделях завантаження, управління

пам’яттю та перериваннями, а також у відсутності зрілого стандарту API для TEE у RISC-V.

Незважаючи на це, відкритість RISC-V і наявність таких проєктів, як Keystone та mTower, створюють передумови для формування уніфікованих стандартів TEE, які у перспективі можуть забезпечити кращу переносимість застосунків між різними архітектурами [6, 8, 15].

1.8 Постановка задачі дослідження

Об’єктом дослідження є TEE у вбудованих та IoT-системах.

Метою дослідження є розроблення та оцінка портування TEE рішення mTower з ARM на RISC-V для порівняння ефективності та захищеності.

Основною задачею дослідження є експериментальне порівняння механізмів безпеки архітектур ARM і RISC-V у контексті побудови TEE для IoT-пристроїв на прикладі портування системи mTower з ARM Cortex-M33 (TrustZone) на SiFive FE310 (RISC-V) та аналізу отриманих результатів.

Для досягнення мети необхідно розв’язати такі підзадачі:

– виконати порт ядра mTower з ARM-архітектури на RISC-V з урахуванням особливостей рівнів привілеїв та системи фізичного захисту пам’яті PMP;

– реалізувати механізм системних викликів ECALL та контекстного перемикання між світами SW і NW;

– провести експериментальне тестування реалізованої системи на емуляторі QEMU, включно з перевіркою реакції на спроби порушення прав доступу;

– виконати порівняльний аналіз результатів роботи mTower на ARM TrustZone та RISC-V PMP, оцінивши точність, гнучкість і рівень апаратного захисту пам’яті;

– визначити обмеження поточної реалізації RISC-V та сформулювати рекомендації щодо досягнення апаратно забезпеченої ізоляції, аналогічної до ARM;

– сформулювати висновки та практичні рекомендації щодо доцільності використання ARM або RISC-V у залежності від вимог до безпеки IoT-систем.

Таким чином, дослідження спрямоване на експериментальне доведення можливості побудови TEE на базі відкритої архітектури RISC-V та визначення умов, за яких вона може забезпечити рівень безпеки, порівнянний з ARM TrustZone.

2 МОДЕЛІ ПІДХОДІВ ДО БЕЗБЕКИ ARM ТА RISC-V

2.1 Загрози та модель нападника

У контексті IoT-середовищ архітектури ARM та RISC-V стикаються з широким спектром загроз, які охоплюють як програмні, так і апаратні рівні.

До основних цінностей, які необхідно захищати, належать криптографічні ключі, прошивка й завантажувальний ланцюг, користувацькі дані та конфігурації, а також доступність сервісів [2, 11].

Типові вектори атак включають мережеві експлойти, локальний фізичний доступ, використання відладочних інтерфейсів, побічні канали, FI, компрометацію постачальницького ланцюга та атаки, пов’язані з компромісом продуктивності у системах реального часу [16, 20, 21, 22].

Нижче подано узагальнену таблицю ключових загроз для IoT-пристроїв із зазначенням активів, векторів атак та можливих заходів протидії.

Таблиця 2.1 – Загрози, вектори атак та можливі заходи захисту

Загроза	**Активи під ризиком**	**Основні вектори**	**Можливі заходи захисту**
1	2	3	4
Крадіжка криптографіч- них ключів [2, 5, 16]	Root-ключі, приватні ключі TLS/SSH	Побічні канали (cache, power), JTAG-дамп	Secure element/TPM, constant-time алгоритми, ізоляція пам’яті
Модифікація або rollback прошивки [11, 17]	Bootloader, ядро, образ TEE	Підміна OTA-оновлення, відкат образу	Secure boot, анти-rollback лічильники, перевірка підписів
Ескалація привілеїв [3, 4, 13]	Kernel/SW, supervisor	Експлуатація багів у драйверах, некоректні API	Мінімізація інтерфейсів, PMP, ретельна валідація

Продовження таблиці 2.1

1	2	3	4
Побічні канали [16, 20]	Криптографіч-ні операції, секрети в ОЗП	Timing, кеш-атаки, аналіз енергоспоживання	Тайм-паддинг, кеш-ізоляція, апаратний шум
FI [16]	Перевірка підписів, контроль цілісності	Voltage/clock glitching, EM, лазер	Дублювання обчислень, fault detectors, фізичний захист
Уразливості ПЗ [10]	Сервіси, драйвери, бібліотеки	RCE, переповнення буфера, уразливі парсери	Патч-менеджмент, песочниці, мінімізація attack surface
Відмова у наданні послуг (DoS)	Доступність сервісів, енергоресурси	Перевантаження CPU, пам’яті, живлення	QoS, rate-limiting, watchdog
Компрометація supply-chain [9, 14]	Silicon, прошивка, root-ключі	Бекдори під час виробництва або оновлення	Attestation, аудит, SBOM, контроль provisioning
Загроза	Активи під ризиком	Основні вектори	Можливі заходи захисту
Витік даних користувачів [2]	Конфіденційні дані, телеметрія	MITM, leakage через логи, компроміс застосунків	Шифрування даних, мінімізація прав доступу
Компроміс продуктивності/безпеки у системах реального часу [21, 22]	Алгоритми обробки зображень, сенсори	Вимкнення захистів задля latency	Баланс продуктивності й захисту, оптимізовані алгоритми

Для ARM TrustZone характерним є чіткий поділ на SW та NW, що спрощує проєктування TEE, але створює ризики, коли безпечне середовище покладається на ненадійні сервіси звичайного світу [2, 3, 12].

Для RISC-V загрози часто залежать від конкретної реалізації PMP та багаторівневої моделі привілеїв (M/S/U), що забезпечує гнучкість і модульність, проте відсутність єдиного стандарту призводить до різнорідності захистів [4, 6, 13].

Незалежно від архітектури, IoT-середовища особливо вразливі до побічних каналів і FI, а також до проблем довгого життєвого циклу пристроїв, рідкісних оновлень і фізичного доступу зловмисників [16, 20].

Окремою загрозою є компрометація supply-chain, яка може призвести до масової втрати довіри до всієї серії пристроїв [9, 14].

Для пристроїв реального часу актуальний компроміс між продуктивністю та безпекою, вимкнення захисних механізмів для зниження затримок може знизити рівень захисту систем, наприклад, при аналізі зображень у сенсорах [21, 22].

Модель нападника визначає потенційні ролі й рівні можливостей зловмисників у контексті IoT-середовищ, що працюють на ARM та RISC-V. Вона дозволяє структурувати аналіз загроз та чітко окреслити межі довіри системи.

Умовно можна виділити кілька типових класів нападників (табл. 2.2), які відрізняються своїми ресурсами, доступом і кінцевими цілями [9, 10, 16, 14].

У контексті IoT-застосувань реальним сценарієм є поєднання різних типів нападників. Наприклад віддалений зловмисник може скомпрометувати слабкий мережевий сервіс, отримати локальний доступ і перейти до рівня привілейованого програмного нападника. Додатково варто враховувати, що багато IoT-пристроїв працюють у публічних або неконтрольованих умовах, що значно підвищує ризик фізичного доступу [10, 16].

Таблиця 2.2 – Категорії нападників та їхні можливості

Клас нападника	**Можливості**	**Типові цілі**	**Приклади атак**
1	2	3	4
Віддалений мережевий нападник [9, 10]	Має доступ лише до відкритих інтерфейсів (API, мережеві порти), може здійснювати MITM, replay, brute-force	Виконання коду віддалено, викрадення даних, DoS	Експлуатація вразливостей у стеку TCP/IP, атаки на слабкі API
Локальний ненадійний користувач [10, 13]	Має фізичний доступ до інтерфейсів (UART, USB), може підключати дебагери, але не володіє привілеями ядра	Ескалація прав, модифікація локальних даних	Використання налагоджувальн-их портів, завантаження непідписаного коду
Привілейова-ний програмний нападник [3, 4, 12, 13]	Контролює ОС або гіпервізор у невірному (non-secure) середовищі; може генерувати системні виклики та винятки до привілейованих режимів	Доступ до секретів у SW на ARM або M/S-mode на RISC-V, обходи PMP	Компромета-ція ядра Linux, атаки на драйвери, маніпуляція SMC/ecall

Продовження таблиці 2.2

1	2	3	4
Апаратний нападник [16, 20]	Має фізичний доступ до пристрою та обладнання для побудови side-channel/FI	Отримання ключів, обходи перевірки підписів, модифікація пам’яті	Power analysis, cache-timing, glitching, лазерні ін’єкції
Supply-chain нападник/ виробник [9, 14]	Має контроль над процесом виробництва або постачання ПЗ/заліза	Масова компрометація, вбудовані бекдори, підробка сертифікатів	Внесення шкідливих модулів у прошивку, приховані апаратні закладки

Важливо також нагадати про компроміс між продуктивністю й безпекою. У середовищах реального часу, наприклад обробка зображень сенсорами, можуть навмисно вимикають частину захисних механізмів задля зменшення затримок, що створює додаткове поле для атак [21, 22].

Таким чином, модель нападника повинна враховувати не лише класичні сценарії атак, але й специфіку використання обмежених IoT-платформ у режимі реального часу.

2.2 Механізми апаратної ізоляції

Однією з ключових передумов безпеки вбудованих систем є забезпечення ізоляції середовищ виконання. Для IoT-пристроїв критично

важливо відокремлювати довірене програмне забезпечення від потенційно небезпечного коду, адже компроміс ядра операційної системи або прикладного рівня не повинен автоматично призводити до доступу до секретів чи зміни поведінки всього пристрою.

ARM TrustZone реалізує механізм апаратної ізоляції шляхом поділу системи на два світи – NW і SW. У NW виконується основна ОС разом із прикладними процесами, тоді як SW використовується для виконання довірених сервісі, каких як SecureOS, криптографічні бібліотеки, сховища ключів, тощо. Комунікація між світами здійснюється через спеціальні виклики SMC, які обробляються прошивкою рівня EL3.

На рисунку 2.1 показана типова схема взаємодії між різними рівнями винятків ARMv8-A. На ньому продемострована взаємодія між прикладнии рівнем (N-EL0), ядром ОС (N-EL1), безпечним ядром (S-EL1) і прошивкою монітора (EL3). У цій архітектурі межа між SW та NW реалізована апаратно і підтримується MMU, контролером пам’яті та системною шиною [2, 3, 12, 18].

На відміну від ARM, архітектура RISC-V не має єдиного стандартного рішення наприклад TrustZone, але забезпечує гнучкіші механізми ізоляції. Основою є рівні привілеїв та PMP, які дозволяють апаратно визначати межі доступу до пам’яті для різних режимів виконання.

На рисунку 2.2 показано узагальнену модель захисту. Прикладні програми виконуються в U-mode, ядро ОС у S-mode, а привілейовані функції прошивки та OpenSBI у M-mode.

Ізоляція довірених середовищ TEE, таких як Keystone чи mTower досягається через механізми PMP, які обмежують доступ до окремих областей пам’яті навіть для привілейованих компонентів.

Це дозволяє створювати захищені енклави, у яких виконуються чутливі обчислення, незалежно від компрометації основної ОС [4, 6, 8, 13, 15].

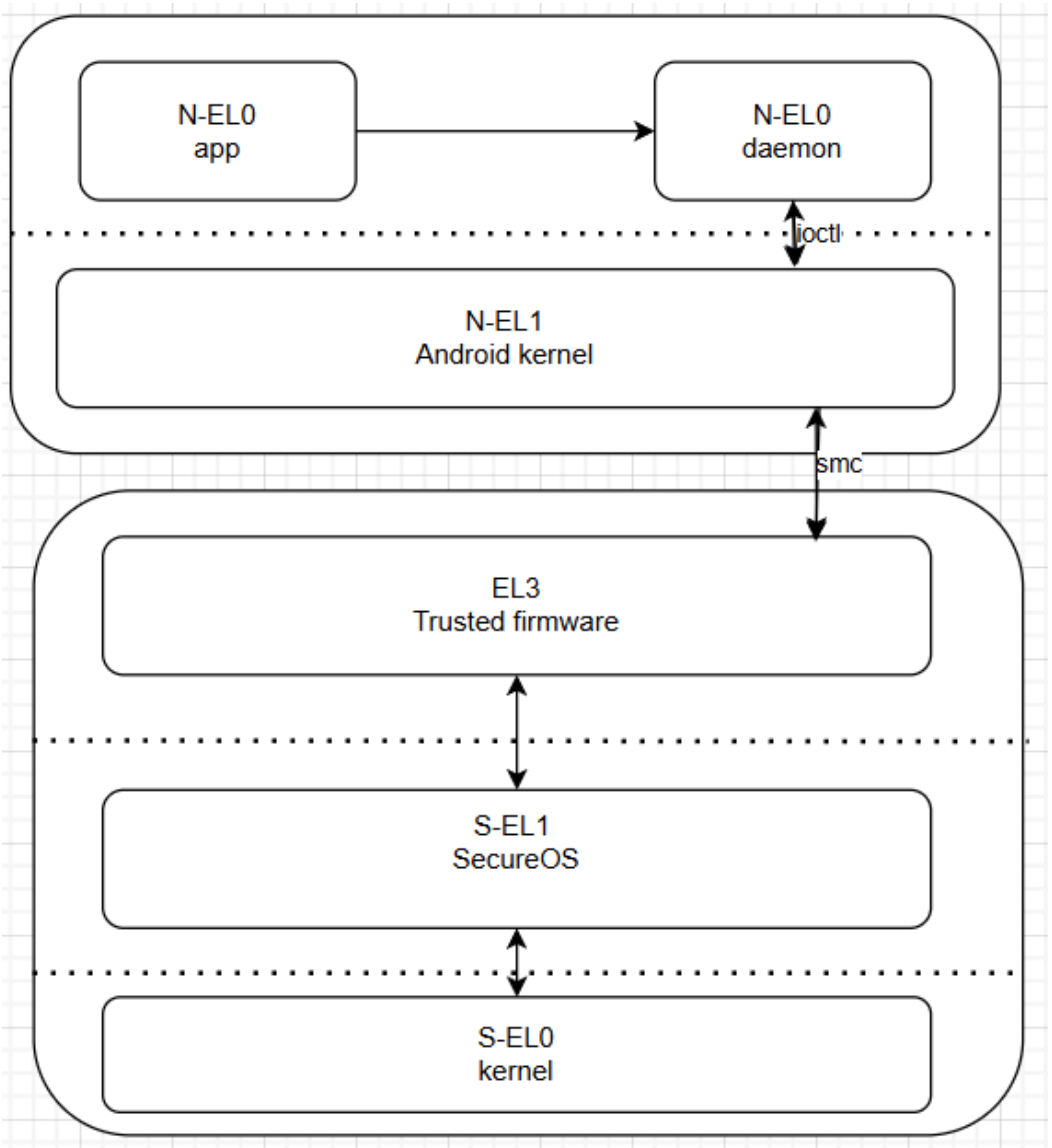

Рисунок 2.1 – Схема моделі захисту в ARM TrustZone

Таким чином, основна різниця між підходами полягає в тому, що TrustZone надає жорстко визначений механізм ізоляції, тоді як RISC-V покладається на набір модульних інструментів (табл. 2.3).

ARM підхід більш стандартизований і підтримується екосистемою виробників, тоді як RISC-V забезпечує більшу гнучкість, але потребує додаткових рішень на рівні TEE-фреймворків для досягнення еквівалентної безпеки.

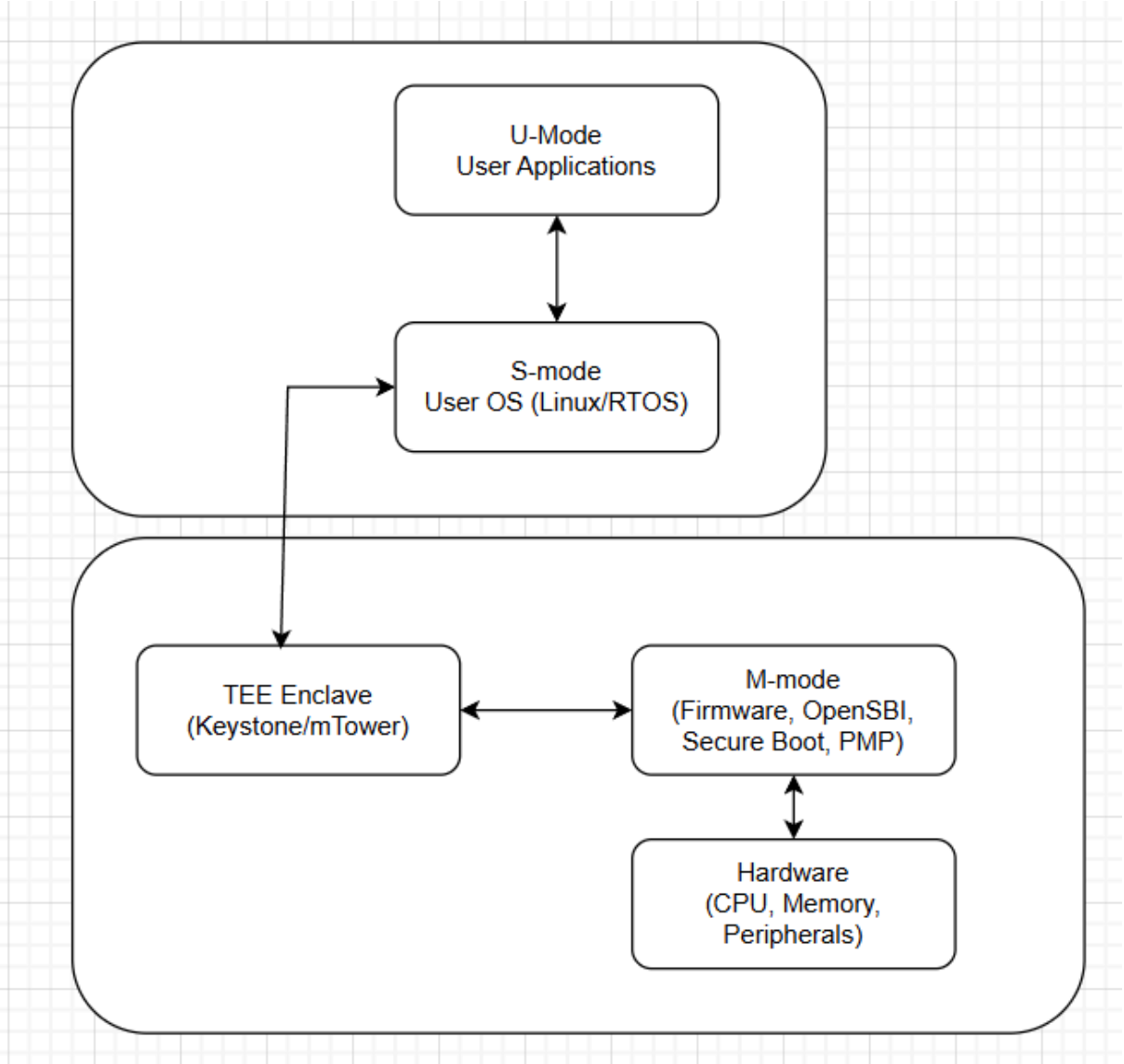

Рисунок 2.2 – Схема моделі захисту в RISC-V

Таблиця 2.3 – Порівняння механізмів апаратної ізоляції ARM TrustZone та RISC-V

Механізм	**ARM TrustZone**	**RISC-V (PMP, TEE)**
1	2	3
Модель ізоляції [2, 3, 4, 13]	Поділ на два світи (SW/NW)	Багаторівнева модель привілеїв та PMP для визначення доступу до пам’яті

Продовження таблиці 2.3

1	2	3
Контроль пам’яті [4, 13]	MMU та системний контролер апаратно відокремлюють регіони пам’яті	PMP дозволяє гнучко визначати межі доступу навіть для привілейованих режимів
Завантажувальний ланцюг [11, 15, 17]	Secure Boot із підтримкою Trust Firmware на рівні EL3, перевірка підписів ОС і TEE	OpenSBI та Secure Boot у M-mode, підтримка апаратного root-of-trust залежить від реалізації
TEE середовище [6–8]	SecureOS у S-EL1, виконання TA у S-EL0	Keystone, mTower, OP-TEE-подібні фреймворки на основі PMP
Інтерфейс взаємодії [4, 6, 13]	SMC для викликів між світами	ecall/виклики між S-mode та M-mode, API залежить від TEE-фреймворку
Стандартизація [2, 3, 6, 13]	Широко стандартизовано, підтримується ARM та вендорами	Відкрита екосистема, специфікація PMP стандартизована, але TEE-фреймворки різнорідні
Захист від атак [12, 14, 16, 20]	Рівень ізоляції високий, але можливі атаки на SW через вразливості або залежність від NW	Гнучкіший контроль доступу, але вразливість до різних реалізацій і нестача єдиного стандарту TEE
Проблеми продуктивності [14, 21, 22]	Невеликий оверхед при переході SMC, оптимізовано для мобільних і IoT	Накладні витрати залежать від реалізації PMP і TEE, важливий компроміс між безпекою і продуктивністю

2.3 Підтримка криптографії

Надійність криптографічних механізмів має вирішальне значення для IoT-пристроїв, оскільки саме вони забезпечують цілісність прошивки, конфіденційність даних і автентифікацію учасників комунікації.

Ефективність реалізації криптографії безпосередньо залежить від апаратної підтримки: спеціальних інструкцій процесора, доступності криптографічних співпроцесорів та апаратних прискорювачів.

У сучасних ARM-процесорах криптографічні можливості реалізовані як у вигляді інструкцій, так і через спеціальні модулі.

Криптографічне розширення для ARMv8 включає апаратне прискорення AES, SHA-1, SHA-256, а також інструкції для виконання множення/ділення для AES-GCM [2, 18]. Це дозволяє реалізовувати симетричне та асиметричне шифрування з мінімальним оверхедом для центрального процесора.

У контексті TrustZone криптографічні операції часто виконуються у SW, де ключі та контексти шифрування ізольовані від основної ОС [3, 12].

В архітектурі RISC-V криптографічна підтримка стандартизована окремим набором розширень, які охоплюють інструкції для AES, SHA-2, SHA-3, SM3/SM4, а також операції з великими числами, що необхідні для асиметричної криптографії [4, 15].

Основна перевага RISC-V в його модульній архітектурі що дозволяє розробникам реалізовувати лише ті інструкції, які потрібні для конкретного IoT-сценарію, що знижує вартість і енергоспоживання фінального виробу.

Додатково, у багатьох RISC-V SoC передбачені апаратні акселератори, які взаємодіють із процесором через PMP-захищені регістри або DMA. Це використовується у TEE-фреймворках, таких як Keystone та mTower [6, 8, 13].

ARM забезпечує зрілу екосистему з готовими інструкціями, драйверами та бібліотеками, оптимізованими під мобільні та IoT-пристрої. Проте набір криптографічних можливостей фіксований виробником і не завжди оптимально підходить для спеціалізованих сценаріїв.

RISC-V надає більшу гнучкість, його криптографічні розширення можна комбінувати з апаратними акселераторами та власними інструкціями, але зрілість екосистеми ще розвивається [13, 15].

Попри апаратну підтримку криптографії, існують виклики, пов’язані з побічними каналами. AES чи SHA можуть бути вразливими до аналізу споживання енергії чи атак на основі часу доступу до кешу, якщо відсутні контрзаходи [16, 20].

У реальному часі, наприклад у випадку сенсорів для аналізу зображень, необхідно враховувати компроміс між продуктивністю та рівнем захисту [21, 22]. Саме тому сучасні тенденції полягають у поєднанні апаратних інструкцій із додатковими акселераторами та оптимізованими бібліотеками, які зменшують як обчислювальні витрати, так і ризики витоку через побічні канали.

ARM надає стабільну та широко застосовувану платформу з апаратними інструкціями для основних криптографічних алгоритмів, інтегрованих у SW.

RISC-V завдяки відкритій специфікації дозволяє гнучко добирати набір криптографічних інструкцій і доповнювати їх власними акселераторами, що особливо важливо для спеціалізованих IoT-пристроїв. Такий підхід створює передумови для балансування між енергоспоживанням, вартістю і рівнем захисту.

2.4 Захист від побічних каналів і фізичних атак

IoT-пристрої часто працюють у неконтрольованих умовах, наприклад на вулиці, у виробничих цехах, у середовищах з відкритим фізичним доступом. Це робить їх особливо вразливими до фізичних атак, серед яких найпоширенішими є SCA та FI.

Побічні канали використовують непрямі характеристики апаратури для отримання секретної інформації: час виконання, споживання енергії,

електромагнітні випромінювання або поведінку кешу [16, 20]. Такі атаки дозволяють витягати криптографічні ключі навіть тоді, коли алгоритми реалізовані коректно на рівні ПЗ.

Система вважається вразливою до атак за часом, якщо час виконання криптографічної операції T має залежність від значення секретного ключа k [20]. Математично умову наявності вразливості можна виразити формулою 2.1.

$$\exists k_1, k_2 : T(op, k_1) \neq T(op, k_2) \quad . \tag{2.1}$$

Для атак, що базуються на аналізі енергоспоживання, міру витоку інформації оцінюють через статистичну залежність між споживанням пристрою та оброблюваними даними. Це описується коефіцієнтом кореляції Пірсона між гіпотетичною моделлю енергоспоживання H (наприклад, вага Геммінга) та реальним виміряним сигналом P, як показано у формулі 2.2 [16, 20].

$$Corr(H, P) = \frac{Cov(H, P)}{\sigma_H \sigma_P} \quad . \tag{2.2}$$

FI спрямовані на створення помилок у роботі пристрою (через voltage glitching, clock glitching чи лазерне опромінення), які обходять перевірки цілісності та автентичності коду [16]. У результаті можливе виконання непідписаного коду, обходи secure boot або витік ключів.

У випадку ARM TrustZone апаратна межа між SW і NW не захищає від атак побічними каналами, адже вони експлуатують сам процесор або спільні апаратні блоки (табл. 2.4). ARM пропонує набір загальних контрзаходів: constant-time реалізації криптографічних алгоритмів, блокування доступу до

відладочних портів, підтримку tamper-resistant модулів у деяких мікроконтролерах [2, 12, 16].

Для RISC-V стандарт ISA не визначає готових механізмів захисту від SCA чи FI. Однак завдяки відкритості архітектури виробники можуть інтегрувати власні засоби: апаратні генератори шуму, динамічне перемикання частоти, дублювання перевірок обчислень.

У наукових роботах досліджується використання PMP для обмеження доступу до регістрів, що зменшує ризик побічного аналізу пам’яті [13, 14, 20]. У рамках TEE-фреймворків (Keystone, mTower) передбачені бібліотеки з підтримкою контрзаходів, але їхня ефективність залежить від конкретного апаратного виконання [6, 8].

Таблиця 2.4 – Порівняння підходів ARM і RISC-V до захисту від побічних каналів та фізичних атак

Аспект	**ARM TrustZone**	**RISC-V (PMP, TEE)**
1	2	3
Типові загрози [16, 20]	Атаки за споживанням енергії/часом, атаки через кеш, FI	Ті самі загрози, залежність від конкретного SoC
Вбудовані механізми [13, 14]	Криптографічні розширення оптимізовані для виконання за сталий час, блокування JTAG, у деяких мікроконтролерах – захищені криптографічні модулі	Відсутні у базовому ISA, виробники додають власні модулі (наприклад, детектори збоїв, випадкові затримки)

Продовження таблиці 2.4

1	2	3
Контрзаходи на рівні TEE [6, 8, 12]	SecureOS у S-EL1 виконує криптографію у SW, використання сертифікованих бібліотек	Keystone/mTower забезпечують API для безпечної криптографії, але залежать від апаратної реалізації
Дослідницькі напрями [14, 20]	Протидія атакам на основі часу доступу до кешу у багатоядерних системах, апаратні контрзаходи у Cortex-M	Інтеграція контрзаходів проти атак через побічні канали у відкриті ядра, оптимізація PMP для захисту ізольованих регіонів
Обмеження [14, 16]	SW не захищений від SCA на рівні мікроархітектури, можливі атаки через спільні кеші	Немає стандарту для захисту від SCA/FI, різнорідність рішень у різних вендорів

У контексті IoT важливо враховувати, що навіть мінімальні побічні канали можуть бути використані для компрометації системи, оскільки пристрої зазвичай не мають можливості швидкого оновлення або заміни. Тому стратегія захисту має включати як апаратні контрзаходи, так і оптимізацію ПЗ, орієнтовану на зменшення інформаційного витоку [21, 22].

ARM пропонує більш зрілі апаратні рішення та перевірені практики, проте все одно лишається вразливим до SCA та FI.

RISC-V забезпечує більшу свободу для інтеграції індивідуальних контрзаходів, але відсутність стандартизації призводить до нерівномірного рівня захисту серед різних реалізацій.

2.5 Проблеми продуктивності та оптимізації для IoT

Одним із ключових викликів для IoT-пристроїв є баланс між рівнем безпеки та обмеженими ресурсами, такими як обчислювальна потужність, енергоспоживання і об’єм пам’яті. Додавання апаратних механізмів безпеки завжди супроводжується накладними витратами, які можуть суттєво впливати на продуктивність у сценаріях реального часу.

Для кількісної оцінки впливу механізмів безпеки на швидкодію системи реального часу необхідно визначити метрику накладних витрат. Повний час виконання захищеної задачі у середовищі TEE складається з часу безпосередніх обчислень та додаткових затримок на обслуговування ізоляції. Це можно вирахувати використовуючі формули 2.3 та 2.4.

$$T_{total} = T_{compute} + 2 \times T_{switch} + T_{cache\_miss} \quad , \tag{2.3}$$

$$\eta = \frac{T_{total} - T_{compute}}{T_{compute}} \times 100\% \quad , \tag{2.4}$$

де T_{switch} – час, необхідний для перемикання контексту між безпечним та звичайним світами (включаючи збереження регістрів та зміну стеку);

T_{cache_miss} – латентність, викликана промахами кеш-пам’яті та буфера асоціативної трансляції після зміни контексту;

η – коефіцієнт накладних витрат.

ARM TrustZone забезпечує низькі накладні витрати при перемиканні між NW і SW завдяки оптимізованим SMC та апаратній підтримці MMU. Проте навіть у цьому випадку затримки можуть бути критичними для IoT-систем із жорсткими вимогами до затримок (наприклад, сенсорів у промисловій автоматизації чи медичних пристроях). Дослідження показують, що складність

SW призводить до додаткових оверхедів у криптографічних операціях та верифікації коду [3, 12, 14].

RISC-V завдяки відкритій архітектурі дозволяє більш гнучко налаштовувати систему. Виробник може реалізувати лише потрібні інструкції та PMP-регіони, скоротивши енергоспоживання і затримки. Проте відсутність єдиного стандарту для TEE призводить до різної ефективності реалізацій. Keystone і mTower показують, що виконання криптографічних операцій у виділених регіонах зменшує ризик витоку таємниць, але збільшує накладні витрати на контекстні перемикання і захищений доступ до пам’яті [6, 8, 13].

Додатковою проблемою є системи реального часу. У таких системах пріоритетом є гарантована затримка обробки сигналу, а не абсолютна безпека. Тому інженери іноді відмовляються від частини захисних механізмів (наприклад, складних перевірок цілісності чи розширених контрзаходів проти побічних каналів) на користь стабільної продуктивності. Це створює додаткові вектори атак [20].

У сфері обробки зображень і сенсорних даних актуальною є оптимізація апаратних структур. Як показують сучасні дослідження, стискання структур опису та спеціалізовані алгоритми дають змогу зменшити обчислювальні витрати, одночасно підтримуючи прийнятний рівень захисту [21, 22]. Це особливо важливо для периферійних пристроїв, які працюють на батарейному живленні й мають обмежену пропускну здатність (табл. 2.5).

ARM TrustZone надає стабільну продуктивність із відносно низькими оверхедами, але обмежує можливості оптимізації під конкретні IoT-сценарії.

RISC-V завдяки відкритості дозволяє налаштовувати архітектуру під потреби (наприклад, додати чи прибрати криптоінструкції), що особливо корисно для периферійних пристроїв. Водночас відсутність стандартизації та зрілих фреймворків створює ризик неоднорідності рівня захисту.

Таблиця 2.5 – Основні проблеми продуктивності та оптимізації у IoT-середовищах

Проблема	**ARM TrustZone**	**RISC-V**
Накладні витрати ізоляції [3, 6, 8, 13]	Перемикання SMC: низькі, але значущі для RT-систем	PMP-захист: залежить від реалізації, у TEE (Keystone/mTower) вище
Криптографічні операції [4, 15]	AES/SHA інструкції вбудовані, висока оптимізація	Розширення K-інструкцій, можливість власних акселераторів
Захист від побічних каналів [16, 20]	Обмежений, частково апаратний; додає затримки	Реалізується вендором; можлива інтеграція додаткових модулів
Обмеження енергоспоживання [14]	Оптимізовані для мобільних/IoT, але обмежена конфігурація	Виробник може прибрати зайві модулі, зменшивши споживання
Оптимізація для обробки даних [21, 22]	Використання готових DSP/NEON блоків	Можливість інтеграції кастомних інструкцій/модулів

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЗАСОБІВ БЕЗПЕКИ В АРХІТЕКТУРАХ ARM І RISC-V

3.1 Вибір середовища реалізації

Вибір середовища реалізації має ключове значення для коректного проведення експерименту, оскільки саме воно визначає точність порівняння між архітектурами, можливість відтворення умов виконання, а також рівень контролю над процесом тестування.

Для реалізації практичної частини було обрано емуляційне середовище QEMU, що підтримує як архітектуру ARM, так і RISC-V, та дозволяє створити ізольовану і відтворювану платформу для аналізу безпекових механізмів.

Перевагою використання QEMU є його здатність точно відтворювати поведінку апаратних компонентів, включно з контролем доступу до пам’яті, обробкою переривань, таймерів і периферійних пристроїв. Що є важливо при дослідженні систем безпеки, через те що відмінності у поведінці апаратури можуть призвести до хибних висновків щодо ефективності захисту.

QEMU дозволяє виконувати покрокове налагодження, та аналіз виняткових ситуацій без потреби у фізичному обладнанні, що робить експерименти відтворюваними й прозорими.

Для порівняння було обрано дві платформи, які репрезентують архітектури V2M-MPS2-Qemu для ARM та SparkFun RED-V RedBoard на базі SiFive FE310 SoC для RISC-V.

Платформа V2M-MPS2 є стандартним варіантом, який часто використовується у дослідженнях технології ARM TrustZone.

SparkFun RED-V RedBoard базується на відкритій архітектурі RISC-V і підтримується емулятором QEMU.

Використання QEMU спрощує процес портування, оскільки надає єдине середовище для компіляції, запуску та налагодження як ARM, так і RISC-V

версій системи. Це дозволяє зосередитися на порівнянні саме архітектурних аспектів безпеки, таких як механізми привілеїв, управління пам’яттю, обробка винятків та доступ до ресурсів, а не на відмінностях у конкретних фізичних пристроях.

Таким чином, вибір середовища реалізації базується на трьох головних принципах – відтворюваність експерименту, архітектурна нейтральність та повний контроль над виконанням системи. Вони забезпечують об’єктивність порівняння між ARM і RISC-V та дозволяють отримати практичні висновки щодо придатності відкритої архітектури RISC-V для реалізації компонентів безпечного середовища виконання у пристроях IoT.

3.2 Постановка експериментальних завдань

Метою експериментальної частини роботи є практична оцінка можливостей архітектури RISC-V у забезпеченні рівня захищеності, еквівалентного реалізації на архітектурі ARM, на прикладі портованої системи mTower.

Для проведення дослідження було створено експериментальне середовище, у якому вихідна реалізація mTower запускається у емуляторі QEMU на платформі V2M-MPS2-Qemu для ARM, та на платформі SparkFun RED-V RedBoard FE310 для RISC.

Обидві системи побудовані в схожих умовах компіляції та конфігурації, що дозволяє виконати коректне порівняння поведінки та рівня безпеки.

У ході експерименту перевіряється робота ключових компонентів системи безпеки, серед яких:

– базова взаємодія між безпечним і небезпечним середовищами, це дозволяє переконатися, що механізми передачі управління та обміну даними між двома доменами працюють належним чином і забезпечують необхідний рівень ізоляції;

– робота прикладних модулів, що використовують криптографічні операції, такі тести демонструють, наскільки система здатна безпечно виконувати типові для TEE завдання, такі як шифрування, автентифікацію чи генерування одноразових кодів;

– коректність виконання завдань у середовищі реального часу, перевірка планувальника задач і міжзадачної взаємодії у FreeRTOS дозволяє оцінити, чи не порушується стабільність системи при взаємодії безпечного та звичайного контекстів;

– відповідність реалізації стандартам TEE, для цього проводяться тести, що перевіряють узгодженість інтерфейсів клієнта та внутрішніх API відповідно до специфікації GlobalPlatform;

– обробка винятків, пов’язаних із безпекою, аналізується, як система реагує на спроби доступу до заборонених ресурсів або порушення ізоляції між контекстами, і чи правильно відновлюється після таких ситуацій;

– спроба несанкціонованого доступу до захищених даних, імітація атаки з небезпечного середовища дозволяє оцінити, чи дійсно захищені області пам’яті та ресурси залишаються недоступними для сторонніх процесів.

Результати експериментів дозволять оцінити придатність архітектури RISC-V для побудови ізольованих середовищ виконання у пристроях IoT, а також визначити відмінності в підходах до забезпечення безпеки між ARM TrustZone та апаратно-програмними механізмами RISC-V.

3.3 Реалізація механізмів безпеки в ARM

Архітектура ARM Cortex-M33 з технологією TrustZone надає комплексний набір апаратних механізмів безпеки для створення TEE. У рамках даного дослідження реалізовано систему безпеки на базі платформи mTower для мікроконтролера ARM Cortex-M33 MPS2-AN505, яка використовує 4 ключові компоненти [2]:

– TrustZone Security States – апаратне розділення на захизщені та незахищені стани виконання на рівні процесора;

– SAU – конфігурація розподілу адресного простору між безпечним та небезпечним світами;

– MPU – окремі блоки захисту пам’яті для Secure та Non-Secure світів (по 8 регіонів кожен);

– SG – спеціалізовані інструкції для контрольованих переходів між світами.

Система реалізує двосвітову архітектуру TrustZone з апаратним розділенням на два ізольовані середовища (рис. 3.1).

Така архітектура забезпечує повноціну апаратну ізоляцію на рівні процесора через SAU, що працює незалежно від програмного забезпечення. SW має повний контроль над системою, а NW виконується в обмеженому контексті з можливістю звертання до довірених сервісів через контрольовані точки входу [2]. SAU апаратно блокує всі спроби NW отримати доступ до захищеної пам’яті, навіть якщо NW скомпрометовано зловмисником.

Архітектура ARM TrustZone для Cortex-M33 реалізує 4 комбінації станів виконання, які поєднують рівень безпеки (Secure / Non-Secure) та режим (Handler / Thread). Усі обробники та задачі у SW мають повний доступ до пам’яті обох світів, тоді як NW обмежений лише власним адресним простором.

Ключова особливість TrustZone – апаратно гарантована ізоляція: будь-яка спроба доступу NW до SW пам’яті викликає виняток. Це виключає можливість компрометації безпечного середовища навіть у разі повного захоплення NW. Таким чином, TrustZone забезпечує hardware-enforced модель безпеки, де захист реалізовано на рівні самої архітектури, без необхідності довіряти програмному коду.

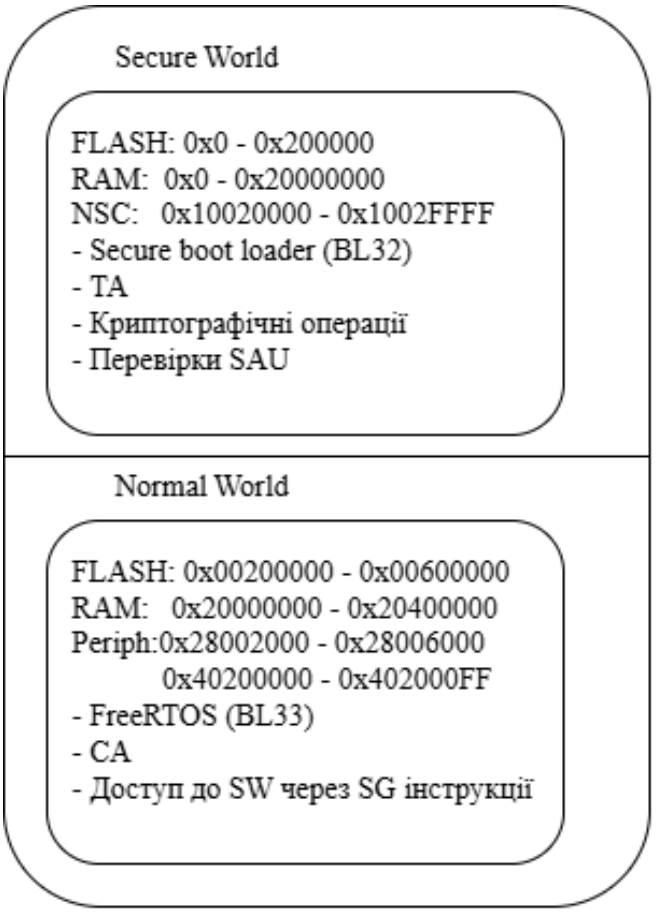

Рисунок 3.1 – Архітектура TrustZone з розділенням світів

Перемикання між світами виконується спеціальними інструкціями та апаратною логікою, що автоматично керує контекстом, стеком і регістрами, забезпечуючи передбачуваний і безпечний перехід між режимами виконання. Це дозволяє реалізувати двосвітову модель із повною апаратною ізоляцією й чітким розмежуванням привілеїв. Приклад функції переходу між світами представлено у лістингу 3.1.

Лістинг 3.1 Secure Gateway функція для переходу з NW до SW:

```
__attribute__((cmse_nonsecure_entry))
int32_t ioctl(uint32_t cmd, struct tee_ioctl_buf_data *buf_data)
{
   /* Автоматичне перемикання стеків виконано апаратно */
   /* Диспетчеризація запитів від NS світу */
```

```
    switch (cmd) {
    case TEE_IOC_OPEN_SESSION:
        return tee_ioctl_open_session(/*ctx,*/ buf_data);
    case TEE_IOC_INVOKE:
        return tee_ioctl_invoke(/*ctx,*/ buf_data);
    case TEE_IOC_CLOSE_SESSION:
        return tee_ioctl_close_session(/*ctx,*/ buf_data);
    default:
        return TEEC_ERROR_ITEM_NOT_FOUND;
    }

    return TEEC_SUCCESS;
}
```

Атрибут cmse_nonsecure_entry інформує компілятор про необхідність генерації SG інструкції, яка є єдиною легітимною точкою входу з NW.

У реалізованій системі SW виконується в Handler mode і містить:

– обробники довірених сервісів;

– криптографічні операції (AES, HOTP);

– менеджер переходів між світами.

NW виконується переважно в Thread mode і містить:

– FreeRTOS kernel;

– прикладні задачі (menuTask, ledBlinkTask);

– Client API для виклику захищених сервісів.

Лістинг 3.2 Виклик захищеного сервісу з NW:

```
extern int sec_sum(void);
extern int32_t Secure_func(void);

/* Використання в задачі FreeRTOS (menuTask) */
```

```
void menuTask(void *pvParameters)
{
   /* ... */
   switch (choice) {
      case '7':
         portDISABLE_INTERRUPTS();
         Secure_func();  /* Прямий виклик через SG - апаратне перемикання */
         portENABLE_INTERRUPTS();
         break;
   }
   /* ... */
}
```

При виклику через SG інструкцію процесор зберігає контекст NW, перемикає стек на стек SW, змінює біт NW в регістрі управління, після завершення відновлює NW контекст.

SAU є апаратним блоком, що визначає які регіони адресного простору належать до SW, а які до NW. Cortex-M33 підтримує до 8 SAU регіонів (SAU0-SAU7), кожен з яких описується трьома 32-бітними регістрами:

– SAU_RNR – номер поточного регіону;

– SAU_RBAR – базова адреса регіону;

– SAU_RLAR – кінцева адреса та атрибути.

SAU підтримує три типи регіонів, вони представлені у таблиці 3.1.

Реалізація виконує ініціалізацію SAU для захисту критичних регіонів.

Після ініціалізації система виводить діагностичну інформацію про конфігурацію, що підтверджує коректність розділення адресного простору між SW та NW.

Лістинг 3.3 Діагностична інформація про конфігурацію SAU:

```
[0000000000] +----------------------------------------------------------------+
```

[0000000000] | Type | Range addresses | Size | Remarks |
[0000000000] +----------+
[0000000000] | FLASH | 0x00200000 - 0x00600000 | 0x400000 | 4MB (Non-Secure) |
[0000000000] | RAM | 0x20000000 - 0x20400000 | 0x400000 | 4MB (Non-Secure) |
[0000000000] | NCS | 0x10020000 - 0x1002FFFF | 0x10000 | 64KB (SG region) |
[0000000000] | Periph| 0x28002000 - 0x28006000 | 0x4000 | 16KB (Non-Secure) |
[0000000000] | Periph| 0x40200000 - 0x402000FF | 0x100 | 256B (UART0, NS) |
[0000000000] +----------+

Таблиця 3.1 – Типи SAU регіонів

Тип регіону	**Біт NSC**	**Призначення**
Secure	0 (в RLAR=0)	Виключно для SW
Non-Secure	1 (RLAR.ENABLE=1, NSC=0)	Доступ тільки з NW
NSC	1 (NSC=1)	SG точка входу

ARM Cortex-M33 підтримує два незалежні блоки MPU. Secure MPU (MPU_S) і Non-Secure MPU (MPU_NS), по вісім регіонів у кожному. Кожен з них активний лише у своєму світі, забезпечуючи додаткову ізоляцію задач і даних на рівні ядра.

У поточній реалізації mTower для MPS2-AN505 MPU не використовується. Ізоляція пам’яті забезпечується виключно через SAU. Такий підхід обумовлений демонстраційною природою mTower і достатнім рівнем безпеки, який надає SAU у двосвітовій моделі.

MPU зазвичай застосовується для ізоляції стеків задач FreeRTOS, критичних структур ядра або буферів між задачами. Для цього CMSIS надає готові функції ARM_MPU_Enable() і ARM_MPU_SetRegion_NS(), але вони не задіяні в поточному коді.

ARM Cortex-M33 надає спеціалізовані інструкції для контрольованих переходів, вони показані у таблиці 3.2.

З NW виклик виконується через extern декларацію та прямий виклик захищеної функції

Таблиця 3.2 – Інструкції для переходів між світами

Інструкція	**Напрямок**	**Опис**	**Апаратна дія**
SG	Marker	Secure Gateway veneer	Позначає легітимну точку входу в Secure
BLXNS	З NW до SW	Branch with Link Non-Secure	Виклик Secure функції з NS світу
BXNS	З NW до SW	Branch Non-Secure	Перехід в Secure без збереження LR
RET	З SW до NW	Return	Повернення використовує стандартний RET з очищенням регістрів

При виклику через BLXNS інструкцію процесор:

– перевіряє що адреса вказує на NSC регіон;

– виконує SG інструкцію;

– перемикає SP на стек SW;

– очищає регістри для запобігання витоку даних;

– встановлює біт Security State у значення Secure.

На відміну від RISC-V, де перемикання стеків реалізовано програмно, ARM Cortex-M33 виконує це апаратно. Для цього потрібно коректно

проініціалізувати NW, встановивши адресу вектора переривань для несекюрного світу, вказівних на стек NW у цьому векторі, та здійснити перехід у NW як це показано у лістингу 3.4.

Лістинг 3.4 Ініціалізація NW:

```
typedef int __attribute__((cmse_nonsecure_call)) ns_func_void(void);
void nonsecure_init(void) {
 /* Встановити адресу Vector Table Non-Secure світу */
 uint32_t *vtor = (uint32_t *) TZ_VTOR_TABLE_ADDR;
 SCB_NS->VTOR = (uint32_t) 0x00200000;

 /* Встановити Non-Secure Stack Pointer (перший елемент Vector Table) */
 __TZ_set_MSP_NS(*vtor);
 __TZ_set_PRIMASK_NS(0x0);

 /* Отримати вказівник на Reset Handler (другий елемент) */
 ns_func_void *ns_reset = (ns_func_void*) (*(vtor + 1));

 /* Перехід в Non-Secure світ */
 ns_reset();
}
```

Під час переходу з NW до SW процесор:

– зберігає вміст регістрів r0–r3, r12, LR, PC та xPSR на стеку NW;

– перемикає покажчик стеку на стек SW;

– очищає регістри r0–r3 та r12, усуваючи можливість збереження залишкових даних;

– встановлює регістр LR у спеціальне значення FNC_RETURN, яке визначає точку повернення з SW до NW.

Механізм захисту від витоку даних через регістри в ARM реалізовано на апаратному рівні. Очищення робочих регістрів при кожному переході між світами гарантує, що NW не має доступу до залишкових значень, які могли містити конфіденційні дані з SW контексту.

У разі спроби порушення політики безпеки процесор ARM Cortex-M33 генерує апаратне виключення SecureFault, яке обробляється у SW.

У поточній реалізації платформи mTower для MPS2-AN505 використовується стандартний обробник, визначений у файлі startup_ARMCM33.S.

Реалізація механізмів безпеки на базі ARM Cortex-M33 з TrustZone забезпечує апаратне розмежування середовищ виконання, переходи між ними та апаратну детекцію порушень політик доступу.

SAU поділяє адресний простір на SW, NW і NSC регіони, у представленій конфігурації налаштовано п’ять регіонів, для FLASH, RAM, периферії та зон виклику.

Апаратне перемикання стеків між світами унеможливлює витік даних, а інструкції SG визначають єдині допустимі точки виклику функцій SW.

Інструкції BLXNS і BXNS забезпечують коректність переходів від NW до SW, а при поверненні із SW до NW процесор апаратно очищає регістри.

Механізм SecureFault забезпечує фіксацію спроб несанкціонованого доступу та некоректних переходів між світами, тоді як SAU визначає порушення атрибуції пам’яті.

Усі ключові компоненти середовища TEE, зокрема SAU, SG та базові захищені сервіси, реалізовані згідно стандартам ARM, криптографічні операції виконуються ізольовано в межах SW.

TrustZone забезпечує апаратну ізоляцію, при якій SAU блокує будь-які спроби доступу NW до пам’яті SW. Безпека не залежить від цілісності або компрометації NW.

Поточна реалізація mTower покладається лише на SAU без залучення Dual MPU, що зменшує гнучкість у внутрішньому розмежуванні пам’яті,

однак є достатньою для порівняльного аналізу з реалізацією на RISC-V, де роль ізоляційного механізму виконує PMP.

Використання Dual MPU в даній роботі не є необхідним, оскільки метою є саме демонстрація відмінностей між архітектурами, а не розширення внутрішньої сегментації.

Отримана реалізація TrustZone на Cortex-M33 є повноцінною експериментальною базою для порівняння з реалізацією mTower на RISC-V. Вона дозволяє провести коректну оцінку різниць у підходах до апаратної ізоляції, детекції порушень і управління доступом, що є ключовими параметрами при аналізі ефективності моделей безпеки ARM і RISC-V.

Повні тексти вихідних файлів додатку для ARM наведені у репозиторії проєкту [44].

3.4 Реалізація механізмів безпеки в RISC-V

Архітектура RISC-V формує багаторівневу модель захисту на основі апаратних механізмів привілеїв, контролю пам’яті та контрольованих переходів між режимами. У межах даного дослідження реалізовано систему безпеки на базі легковісного середовища mTower для мікроконтролера SiFive FE310.

Реалізована модель відповідає класичній концепції TEE з двома ізольованими середовищами SW та NW, між якими здійснюється передача управління через апаратно контрольовані переходи (рис. 3.4).

На відміну від ARM, ця ізоляція не є апаратною, оскільки обидва світи виконуються в M-mode.

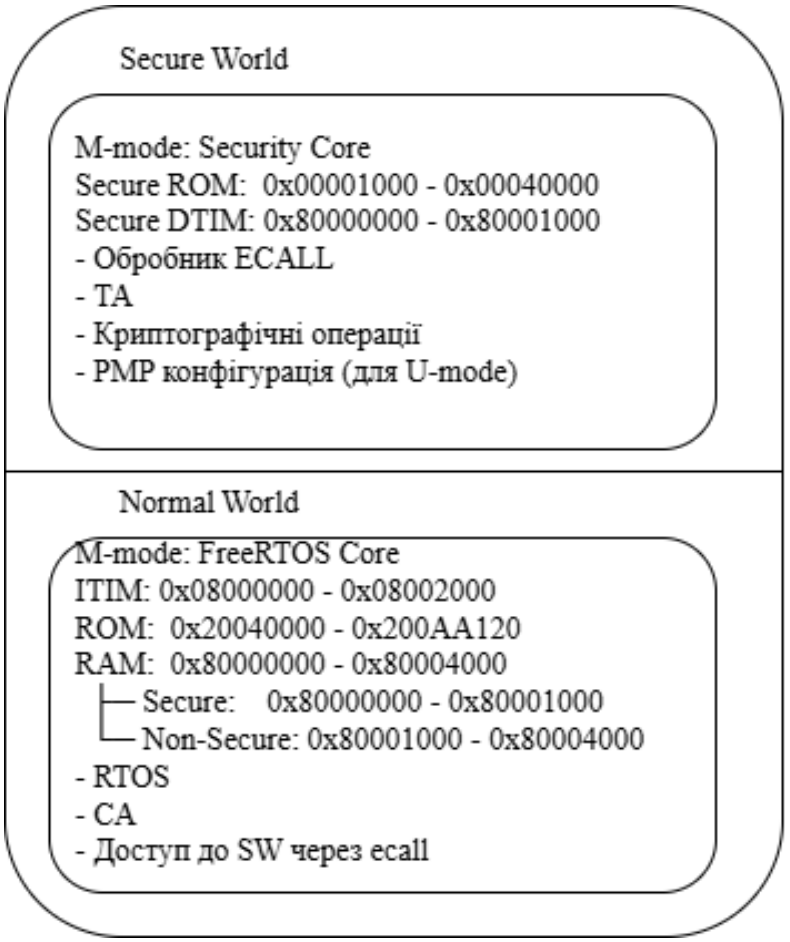

Рисунок 3.4 – Архітектура розділення SW та NW

Мікроконтролер FE310 підтримує лише M-mode та U-mode, що накладає обмеження на реалізацію TEE.

Поточний режим привілеїв визначається бітами MPP регістра mstatus. Перехід між режимами здійснюється через інструкції ecall (виклик більш привілейованого режиму) та mret (повернення з обробника пастки) [4].

Для визначення поточного режиму виконання реалізовано функцію, що аналізує стан системної змінної privilege_status, яка оновлюється ядром FreeRTOS під час перемикання контексту як представлено на лістингу 3.5.

Лістинг 3.5 Визначення поточного рівня привілеїв:

```
static inline char get_privilege_mode(void)
{
  /* Access FreeRTOS's privilege_status variable which tracks the current mode
```

```
    * This is set by vPortUpdatePrivilegeStatus() from the MPP field of mstatus
    * Values: 0 = U-mode, 3 = M-mode */
   extern volatile uint32_t privilege_status;

   if (privilege_status == 3) {  /* ePortMACHINE_MODE = 3 (MPP=11) */
      return 'M';  /* Machine mode */
   } else if (privilege_status == 0) {  /* ePortUSER_MODE = 0 (MPP=00) */
      return 'U';  /* User mode */
   } else {
      return 'S';  /* Supervisor mode (MPP=01, not used in this system) */
   }
}
```

Змінна privilege_status зберігає значення бітів MPP, витягнутих з регістра mstatus під час обробки переривань та перемикання задач.

У реалізованій системі всі задачі FreeRTOS виконуються в M-mode з наступних причин:

– архітектурні обмеження FreeRTOS – ядро потребує безпосереднього доступу до привілейованих CSR для управління перериваннями та таймерами;

– доступ до внутрішніх структур – глобальні змінні ядра (списки задач, черги, семафори) розміщені в захищених секціях пам'яті.

Важливо зазначити, що PMP конфігурація у поточній реалізації не захищає SW пам'ять від NW, оскільки обидва працюють в M-mode.

Проте реалізована інфраструктура системних викликів ecall дозволяє в майбутньому перевести користувацькі задачі в U-mode без зміни прикладного коду.

Створення задачі в M-mode виконується з встановленням біта portPRIVILEGE_BIT, приклад показаний у лістингу 3.6.

Лістинг 3.6 Створення привілейованої задачі:

```
xTaskCreate(menuTask,                    /* Функція задачі */
    "menu",                              /* Ім'я задачі */
    400,                                 /* Розмір стеку */
    (void *)NULL,                        /* Параметри */
  mainQUEUE_SEND_TASK_PRIORITY | portPRIVILEGE_BIT,
    NULL);                               /* Дескриптор задачі */
```

Такий підхід забезпечує стабільність системи при збереженні можливості майбутнього переходу на U-mode для прикладних задач.

Мікроконтролер FE310 підтримує 16 PMP регіонів, кожен з яких описується парою регістрів:

– pmpcfg[i] – конфігурація регіону (права доступу, режим адресації);

– pmpaddr[i] – адреса або межа регіону.

Конфігурація PMP регіону визначається 8-бітним значенням з наступною структурою (табл. 3.4).

Архітектура RISC-V підтримує 4 режими адресації PMP [4]:

– OFF (0) – регіон вимкнено, доступ дозволено для всіх режимів;

– TOR (Top of Range, 1) – регіон від pmpaddr[i-1] до pmpaddr[i];

– NA4 (Naturally Aligned 4-byte, 2) – вирівняний 4-байтний регіон;

– NAPOT (Naturally Aligned Power-Of-Two, 3) – вирівняний регіон розміром 2^n байт.

Таблиця 3.3 – Структура байта конфігурації PMP

Біт	**Назва**	**Опис**	**Значення**
1	2	3	4
7	L (Lock)	Блокування зміни конфігурації	0 = можна змінити, 1 = заблоковано

Продовження таблиці 3.3

1	2	3	4
6-5	-	Резерв	Завжди 0
4-3	A (Address)	Режим адресації	00=OFF, 01=TOR, 10=NA4, 11=NAPOT
2	X (Execute)	Право виконання	0 = заборонено, 1 = дозволено
1	W (Write)	Право запису	0 = заборонено, 1 = дозволено
0	R (Read)	Право читання	0 = заборонено, 1 = дозволено

Для програмного управління PMP створено набір структур даних, вони продемонстровані у лістингу 3.7.

Лістинг 3.7 Структура конфігурації PMP регіону:

```
typedef struct {
    /** @brief read right configuration */
    uint8_t R;
    /** @brief write right configuration */
    uint8_t W;
    /** @brief execute right configuration */
    uint8_t X;
    /** @brief address matching configuration */
    uint8_t A;
    /** @brief lock mode */
    uint8_t L;
} pmp_cfg_t ;

typedef struct {
    /** @brief Number of PMP available on the hart */
```

```
    uint32_t nb_pmp;
    /** @brief smallest granularity available on the hart */
    uint32_t granularity;
} pmp_info_t ;
```

Компоновщик системи визначає спеціальні секції для коду та даних з різними рівнями захисту. Привілейовані секції вирівнюються по межі 32 байти для ефективного використання PMP.

Лістинг 3.8 Визначення привілейованих секцій у скрипті компоновщика:

```
.privileged_functions : ALIGN (32) {
    __privileged_functions_start__ = .;
     KEEP(*(privileged_functions))        /* Секція для привілейованого коду
*/
    . = ALIGN(32);
    __privileged_functions_end__ = .;
} >rom

.privileged_data (NOLOAD) : ALIGN(32) {
    __privileged_data_start__ = .;
    *(privileged_data)                /* Секція для привілейованих даних */
    . = ALIGN(32);
    __privileged_data_end__ = .;
} >ram
```

Вирівнювання по 32 байти обумовлено мінімальною гранулярністю PMP на FE310 [4].

Процес ініціалізації PMP виконується під час завантаження SW.

Лістинг 3.9 Ініціалізація PMP:

```
void secure_pmp_init(void)
{
   /* Entry 0: Secure ROM, TOR mode, end address 0x00040000 */
   write_pmpaddr(0, g_secure_boot_regions[0].addr >> 2);
   /* Entry 1: Secure DTIM (0x80000000-0x80001000), NAPOT mode */
   write_pmpaddr(1, g_secure_boot_regions[2].addr);
   /* Entry 2: Rest of memory accessible to NS, NAPOT */
   write_pmpaddr(2, 0x3FFFFFFFu);  /* Cover all remaining address space */
   /* Configure permissions */
 uint32_t cfg = ((uint32_t)g_secure_boot_regions[0].cfg << 0) |   /* Entry 0: TOR, R+X */
      ((uint32_t)g_secure_boot_regions[2].cfg << 8) |   /* Entry 1: NAPOT, R+W (secure data) */
                ((uint32_t)(PMP_CFG_A_NAPOT | PMP_CFG_R | PMP_CFG_W | PMP_CFG_X) << 16);  /* Entry 2: NAPOT, R+W+X */
   write_pmpcfg0(cfg);
   uint32_t pmpcfg0 = read_pmpcfg0();
   secure_log("PMP configuration complete");
   secure_log_hex("pmpcfg0=0x", pmpcfg0);
}
```

Інструкція ecall забезпечує контрольований перехід від менш привілейованого режиму до більш привілейованого [4]. У контексті mTower, ecall виконує дві ключові функції:

– контроль доступу до сервісів – безпечний світ надає обмежений набір довірених сервісів через перевірений інтерфейс;

– абстракція між світами – небезпечний світ може викликати функції безпечного світу без знання про їх внутрішню реалізацію.

Обрабка ecall з ядра FreeRTOS, що працює в M-mode і виклик диспетчера продемонстровано у лістингу 3.10.

Лістинг 3.10 Виклик диспетчера з обробника ecall:

```
if (mcause == MCAUSE_ECALL_FROM_MACHINE_MODE)
{
   /* ECALL from M-mode (non-secure payload). */
   mailbox_log_u32(4u, 0x4543414Cu);
   mailbox_log_u32(5u, frame->a7);

   secure_service_dispatch(frame);

   mepc += 4u;

   __asm__ __volatile__("csrw mepc, %0" : : "r"(mepc));
   return;
}
```

Реалізація сервісу get_ticks демонструє стандартний підхід до безпечного доступу з NW до апаратних ресурсів через механізм ecall. У SW таймер зчитується безпосередньо через регістри модуля CLINT. Для уникнення помилки при переповненні молодших бітів використовується подвійне читання старших і молодших слів з перевіркою їх узгодженості, що гарантує коректність отриманого 64-бітного значення.

Лістинг 3.11 Функція читання 64-бітного таймера:

```
static inline uint64_t read_mtime(void)
{
   uint32_t lo, hi, hi2;
   do {
```

```
        __asm__ __volatile__("" ::: "memory");
        hi = *(volatile uint32_t *)(CLINT_MTIME + 4);
        lo = *(volatile uint32_t *)CLINT_MTIME;
        hi2 = *(volatile uint32_t *)(CLINT_MTIME + 4);
    } while (hi != hi2);
    return ((uint64_t)hi << 32) | lo;
}
```

Обмін даними між SW і NW здійснюється через уніфіковану функцію виклику mtower_secure_call, яка виконує ecall і повертає структуру з результатами у регістрах a0–a2. Ідентифікатор сервісу передається у регістрі a7.

Лістинг 3.12 Узагальнена функція виклику ecall:

```
typedef struct {
    uint32_t status;
    uint32_t value0;
    uint32_t value1;
} mtower_secure_result_t;

static inline mtower_secure_result_t mtower_secure_call(uint32_t service_id,
                                                        uint32_t arg0,
                                                        uint32_t arg1,
                                                        uint32_t arg2)
{
    register uint32_t a0 __asm__("a0") = arg0;
    register uint32_t a1 __asm__("a1") = arg1;
    register uint32_t a2 __asm__("a2") = arg2;
    register uint32_t a7 __asm__("a7") = service_id;
```

```
    __asm__ __volatile__("ecall"
                         : "+r"(a0), "+r"(a1), "+r"(a2)
                         : "r"(a7)
                         : "memory");

    mtower_secure_result_t result = {a0, a1, a2};
    return result;
}
```

Для доступу до сервісу з nw реалізовано функцію, що виконує ecall до SW і повертає значення таймера.

Лістинг 3.13 Функція читання таймера з NW:

```
uint64_t ns_secure_get_ticks(void)
{
    ns_secure_init_mtvec();
    bool did_secure = false;
                                        mtower_secure_result_t result =
ns_secure_call(MTOWER_SECURE_CALL_GET_TICKS,
                                    0u, 0u, 0u, &did_secure);
    if (did_secure && result.status == MTOWER_SECURE_CALL_STATUS_OK)
        return ((uint64_t)result.value1 << 32) | result.value0;

    return read_clint_mtime();
}
```

Кожен виклик супроводжується передачею ідентифікатора сервісу в регістрі a7 та аргументів у регістрах a0–a2. У SW запит обробляється через диспетчер, який ідентифікує тип сервісу і виконує відповідну функцію.

Лістинг 3.14 Виклик довіреного сервісу з NW:

```
mtower_secure_result_t result = mtower_secure_call(
    MTOWER_SECURE_CALL_GET_TICKS,
    0,
    0,
    0
);

if (result.status == MTOWER_SECURE_CALL_STATUS_OK) {
    uint64_t ticks = ((uint64_t)result.value1 << 32) | result.value0;
}

mtower_secure_call(
    MTOWER_SECURE_CALL_LOG_U32,
    0xDEADBEEF,
    0,
    0
);
```

SW здійснює перевірку джерела виклику за значеннями регістрів mcause і mepc, обробляючи лише ті пастки, які виникли в результаті ecall із M-mode. Це запобігає несанкціонованому доступу або спробам обійти механізм диспетчеризації.

Лістинг 3.15 Фрагмент обробника пасток з валідацією джерела:

```
void secure_trap_handler(struct trap_frame *frame)
{
    uint32_t mcause, mepc;
    __asm__ __volatile__("csrr %0, mcause" : "=r"(mcause));
```

```
    __asm__ __volatile__("csrr %0, mepc" : "=r"(mepc));

    mailbox_log_u32(2u, mcause);
    mailbox_log_u32(3u, mepc);
    if (mcause == MCAUSE_ECALL_FROM_MACHINE_MODE)
    {
        mailbox_log_u32(4u, 0x4543414Cu);
        mailbox_log_u32(5u, frame->a7);

        secure_service_dispatch(frame);
        mepc += 4u;
        __asm__ __volatile__("csrw mepc, %0" : : "r"(mepc));
        return;
    }

    secure_panic("Unexpected trap in secure world");
}
```

Така перевірка гарантує, що обробляються лише очікувані ecall із NW, а всі інші пастки викликають аварійне завершення роботи SW. Це забезпечує чітку межу між світами та мінімізує ризик виконання непередбачених або зловмисних викликів.

У реалізованій системі забезпечено повну ізоляцію стеків між SW та NW. Кожен світ використовує власну область пам’яті для стеку, визначену в компонувальнику. Стек захищеного середовища розміщується в області Secure DTIM за адресами 0x80000000–0x80001000, стек незахищеного середовищазнаходиться в NS RAM у межах 0x80001000–0x80004000.

Під час переходу між світами перемикання стеків виконується через механізм обробки пасток, який зберігає та відновлює контекст виконання в структурі trap_frame.

Повні тексти вихідних файлів додатку для RISC-V наведені у репозиторії проєкту [44].

3.5 Проведення експерименту та аналіз результатів

Даний розділ містить результати експериментального тестування механізмів безпеки на двох архітектурах: RISC-V та ARM. Метою експериментів є верифікація коректності реалізації механізмів ізоляції пам’яті, системних викликів та захисту привілейованих ресурсів.

3.5.1 Результати тестування платформи RISC-V

Конфігурацію тестового середовища на якому проводилось експериментальне дослідження приведено у таблиці 3.4.

Тестування виконувалось за допомогою автоматизованого скрипту, який має наступні кроки:

Крок 1. Запускає емулятор QEMU з двома консолями.

Крок 2. Виконує серію тестових сценаріїв через telnet-підключення.

Крок 3. Збирає вивід обох консолей для аналізу.

Крок 4. Перевіряє коректність виконання кожного тесту.

Усі тести виконуються в автоматичному режимі без втручання користувача, що забезпечує повторюваність результатів.

Таблиця 3.4 – Конфігурація тестового середовища RISC-V

Параметр	**Значення**
1	2
Архітектура	RISC-V RV32IMAC

Продовження таблиці 3.4

1	2
Мікроконтролер	SiFive FE310-G002
Платформа	HiFive1 Rev B
Емулятор	QEMU 8.2.2
Оперативна пам’ять	16 КБ SRAM
Флеш-пам’ять	512 КБ
Кількість PMP регіонів	16
ОС реального часу	FreeRTOS v10.4.6
Версія mTower	v0.6.0 (адаптована для RISC-V)

У тестовому скрипті запускаються 4 тестові сценарії. Перший тест демонструє виклик функції SW з NW використовуючи ecall, ця функція виконує операцію інкрементування та вимірювання часу. Його мета це перевірити коректність комунікації між NW та SW через механізм ecall.

Лістинг 3.16 Результати виконання тесту 1 на RISC-V:

[M][7940] TEE test: Hello_world
[M][7944] [demo] Hello World stub start
[M][7948] Invoking pseudo TA logic with 42
[M][7961] [N-ECALL] → Calling secure GET_TICKS (first call)...
[M][7973] [N-ECALL] ← Returned from secure world
[M][7981] [demo] Secure world timestamp (start): 0
[M][7992] [demo] TA log: Hello World!
[M][7999] TA incremented value to 43
[M][8004] [N-ECALL] → Calling secure GET_TICKS (second call)...
[M][8017] [N-ECALL] ← Returned from secure world

[M][8024] [demo] Secure world timestamp (end): 8156
[M][8031] [demo] Elapsed secure ticks: 8156
[M][8037] [demo] TA log: Goodbye!
[M][8042] [demo] Hello World stub done

Проаналізувавши результати можно побачити, що ecall успішно перемикає контекст з NW в SW. Значення 42 коректно передано в TA. TA успішно інкрементувало значення до 43. Результат коректно повернуто в NW.

Другий тест перевіряє криптографічні операції AES. Його мета це перевірити роботу криптографічних операцій у SW. Під час виконання тесту виконується шифрування та дешифрування блоку даних з використанням алгоритму AES-128.

Лістинг 3.17 Результати виконання тесту 2 на RISC-V:

[M][73433] TEE test: AES
[M][73437] [demo] AES stub start
[M][73441] Prepare encode operation
[M][73449] Key : A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5
[M][73463] IV : 00000000000000000000000000000000
[M][73477] Plaintext : 5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A
[M][73487] Ciphertext : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
[M][73497] Prepare decode operation
[M][73502] Decoded : 5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A5A
[M][73514] Clear text and decoded text match
[M][73519] [demo] AES stub done

Проаналізувавши результати можно побачити, що початковий текст успішно зашифровано. Зашифрований текст успішно дешифровано назад. Розшифований текст ідентичний початковому. Криптографічний ключ зберігається в безпечному світі.

Третій тест перевіряє ізоляцію через PMP з SW. Його мета це продемонструвати, що код SW має повний доступ до всієї пам’яті. В ньому з NW викликається функція в SW, яка намагається отримати доступ до захищеної пам’яті.

Лістинг 3.18 Результати виконання тесту 3 на RISC-V:

```
[M][138829] +----------------------------------------------------------------+
[M][138839] | H/W Security exception example (from secure)                   |
[M][138849] +----------------------------------------------------------------+
[M][138859] | Calling secure world function via ECALL to trigger exception...  |
[M][138869] +----------------------------------------------------------------+
[M][138881] | Secure function returned: 0x00000000
[M][138897] +----------------------------------------------------------------+
```

Проаналізувавши результати можно побачити, що SW у M-mode має доступ до всієї пам’яті та функція повернула результат без виключень до NW.

Четвертий тест перевіряє роботу PMP з U-mode у NW. Його мета це продемонструвати апаратний захист пам’яті через PMP при спробі доступу з непривілейованого режиму. Читання захищеної пам’яті з M-mode успішно виконується, а читання захищеної пам’яті з U-mode викликає виключення.

Лістинг 3.19 Результати виконання тесту 4 на RISC-V:

```
[M][236934] +----------------------------------------------------------------+
[M][236945] | H/W Security exception example (from non-secure)               |
[M][236955] +----------------------------------------------------------------+
[M][236971] | This demonstrates RISC-V PMP (Physical Memory Protection) in
action |
[M][237001] +----------------------------------------------------------------+
[M][237020] | TEST 1: Reading secure DTIM from M-mode...                     |
[M][237031] |   Result: Read succeeded! value=0x20046C7C                     |
```

[M][237041] | Status: Expected - M-mode bypasses PMP |
[M][237051] +----------+
[M][237064] | TEST 2: Reading secure DTIM from U-mode... |
[M][237073] | Temporarily dropping to U-mode to test PMP... |

[M][237089] !!! EXCEPTION CAUGHT !!!
[M][237093] Exception cause: 5 Load access fault
[M][237099] System halted.

Проаналізувавши результати можно побачити, що значення успішно прочитано з адреси 0x80003000. M-mode має повний доступ, PMP не застосовується. При переході у U-mode система успішно знизила привілеї та ми отримали виключення з кодом 5 (Load Access Fault) як очікувалось. PMP апаратно заблокував доступ до захищеної пам’яті.

Система перехопила виключення та зупинилась. PMP працює коректно, забезпечуючи апаратну ізоляцію пам’яті між привілейованим та непривілейованим кодом.

У підсумку маємо 4 виконаних функціональних тестів для верифікації механізмів безпеки платформи RISC-V FE310. Усі тести пройдено успішно, що підтверджує коректність реалізації:

– комунікації між світами, механізм ecall забезпечує коректні переходи з NW до SW з передачею параметрів та поверненням результатів, виміри часу демонструють працездатність системного таймера;

– криптографічних операцій, алгоритм AES-128 в SW виконує шифрування та дешифрування з коректним відновленням вихідних даних, підтверджуючи ізоляцію критичних операцій;

– PMP з доступом для SW, SW у M-mode має повний доступ до всієї пам’яті відповідно до специфікації RISC-V, що необхідно для функціонування ядра безпеки;

– апаратної ізоляції через PMP, програмне забезпечення у U-mode викликає п’яте виключення при спробі доступу до захищенної пам’яті.

Тести підтвердили, що PMP працює лише для U-mode. Для повноцінної апаратної ізоляції між SW та NW необхідно перевести весь код NW з M-mode в U-mode.

Командний інтерфейс для тестування RISC-V додатка представлено у Додатку А.

3.5.2 Результати тестування платформи ARM

Конфігурацію тестового середовища на якому проводилось експериментальне дослідження приведено у таблиці 3.5:

Таблиця 3.5 – Конфігурація тестового середовища ARM

Параметр	**Значення**
Архітектура	ARM Cortex-M33
Платформа	ARM MPS2+ AN505 FPGA
Емулятор	QEMU 8.2.2
Оперативна пам’ять	32 КБ SRAM
Флеш-пам’ять	252 КБ
Кількість SAU регіонів	8
Кількість MPU регіонів	не здійснюються
ОС реального часу	FreeRTOS v10.4.6
Версія mTower	v0.6.0
Дата тестування	24 жовтня 2025

Тестування виконувалось за допомогою автоматизованого скрипту, який має наступні кроки:

Крок 1. Запускає QEMU з двома послідовними портами.

Крок 2. Встановлює з’єднання з NW.

Крок 3. Встановлює з’єднання з SW.

Крок 4. Надсилає ініціюючий сигнал для запуску NW коду.

Крок 5. Збирає вивід обох світів для аналізу.

Крок 6. Виконує тести в трьох окремих сесіях з перезапуском QEMU.

Особливість ARM тестування полягає в тому, що деякі тести, зокрема опції 6 та 8, викликають фатальні виключення системи безпеки. Через це скрипт автоматично перезапускає QEMU між виконанням окремих тестів.

У першій сесії виконуються тести для перевірки комунікації між NW та SW, виклик криптографічних операцій, та перевірку доступу з SW до захищених даних. Ця сесія проходить без аварійної зупинки системи.

Друга сесія запускає тест, що викликає апаратне виключення безпеки – доступ до захищеної пам’яті з NW. ми повинні побачити зупинку системи через виключення.

Третя сесія виконує тест, який імітує поведінку шпигунського застосунку та також призводить до зупинки системи через виключення.

Лістинг 3.20 Результати виконання сесії 1 на ARM:

```
[0019633188] +- Menu ----------------------------------------------+
[0019633188] | [1] - Run Hello World (Ported from OP-TEE, pseudo TA)    |
[0019651220] | [2] - Run AES (Ported from OP-TEE, user TA)              |
[0019651220] | [3] - Run HOTP (Ported from OP-TEE, user TA)             |
[0019651220] | [4] - Get FreeRTOS Task List                             |
[0019651220] | [5] - Run Test Suite for GP TEE Client & Internal API    |
[0019651220] | [6] - Run H/W Security exception example (from non-secure)|
[0019651220] | [7] - Run H/W Security exception example (from secure)   |
[0019651220] | [8] - Run spy app that trying to get protected data      |
```

[0019651220] +--+

[0000000000] -=mTower v0.6.0=- Oct 24 2025 10:47:55
[0000000000] +--+
[0000000000] | Secure handler (BL32) is running ... |
[0000000000] +--+
[0000000000] | FLASH | 0x00000000 - 0x0003F000 | 0x3F000 | 252k |
[0000000000] | RAM | 0x20000000 - 0x20007FFF | 0x8000 | 32k |
[0019591945] Called secure function! #1
[0019633188] Called secure function! #2
[0032287278] Called secure function! #3

Аналіз результатів першої сесії показав, що меню коректно відображається на консолі NW середовища, а SW успішно ініціалізується та переходить у стан очікування викликів.

Комунікація між NW та SW працює стабільно. SW має доступ до всієї пам’яті застосунку, PMP її не обмежує. Криптографічні операції виконуються коректно.

Лістинг 3.21 Результати виконання сесії 2 на ARM:

[0020626679] -=mTower v0.6.0=- Oct 24 2025 10:47:57
[0020626679] +--+
[0020626679] | Nonsecure FreeRTOS (BL33) is running ... |
[0020626679] +--+

[0020328706] Secure Hard Fault Handler: invalid memory access or malware activity detected

Аналіз результатів другої сесії показав, що механізми безпеки SAU спрацювали коректно. Система своєчасно виявила спробу несанкціонованого доступу.

Обробник переривань був викликаний відповідно до очікуваної логіки, що підтверджує правильну роботу механізмів виключень у захищеному середовищі. Після цього система була зупинена, запобігши будь-якому витоку або пошкодженню даних.

Лістинг 3.22 Результати виконання сесії 3 на ARM:

```
[0019878628] +- Menu ----------------------------------------------------+
[0019878628] | [8] - Run spy app that trying to get protected data      |
[0019950876] +----------------------------------------------------------+

[0019856176] Called secure function! #1
[0019878628] Called secure function! #2
```

Аналіз результатів третьої сесії показав, що захисні механізми спрацьовували при спробі доступу до критичних даних. Це свідчить про ефективну роботу апаратного контролю доступу та правильну взаємодію між SW і NW.

Зафіксовані виклики захищених функцій підтверджують, що система активно моніторила діяльність і своєчасно реагувала на потенційно небезпечні дії.

Зведені результати тестів на платформі ARM Cortex-M33 демонструють стабільну роботу TEE у трьох сценаріях.

У першій сесії виконання відбулося успішно, без аварійних завершень.

Друга сесія очікувано завершилася спрацьовуванням апаратного виключення безпеки, що підтверджує правильну роботу захисних механізмів при атаках або шкідливій активності.

Третя сесія підтвердила ефективність моніторингу активності і контрольованість доступу до критичних ділянок пам’яті, демонструючи, що система не допускає витоку або пошкодження даних навіть під час моделювання шпигунського ПЗ.

Усі результати позначені як успішні, що свідчить про повну відповідність роботи TEE очікуваним моделям поведінки.

ARM TrustZone забезпечує апаратну ізоляцію двох середовищ виконання, використовуючи SAU для розподілу пам’яті, а також інструкції SG для безпечних переходів між ними.

Завдяки цим механізмам система забезпечує швидке виявлення спроб порушення безпеки, мінімізує наслідки помилок і гарантує цілісність даних навіть у разі появи потенційно небезпечних викликів.

Командний інтерфейс для тестування ARM додатка представлено у Додатку А.

3.5.3 Порівняльний аналіз результатів RISC-V та ARM

Експеримент показав, що RISC-V має значний потенціал для безпечних IoT-систем завдяки відкритій архітектурі, гнучкій кастомізації та відсутності ліцензійних обмежень. Вона дозволяє розробнику повністю контролювати політику безпеки та додавати власні інструкції.

Основне обмеження це відсутність апаратної ізоляції між SW і NW, оскільки обидва працюють у M-mode. Для повноцінної ізоляції потрібно перенести NW у U-mode, що вимагає суттєвої модифікації FreeRTOS.

ARM Cortex-M33 з TrustZone забезпечує апаратно гарантовану ізоляцію між SW і NW за допомогою SAU, яка блокує будь-який доступ NW коду до захищенної пам’яті.

Перемикання контекстів і стеків виконується апаратно, що виключає потребу у додатковій логіці.

Технологія зріла, має багаторічну промислову підтримку, сертифіковані рішення і розвинену інструментальну базу.

Для критичних IoT доцільно обирати ARM TrustZone, а для дослідницьких і кастомних проєктів RISC-V завдяки відкритості та гнучкості.

3.6 Практичні рекомендації та напрями подальших досліджень

Вибір між ARM TrustZone та RISC-V PMP для конкретного проєкту повинен базуватися на глибокому розумінні вимог до безпеки, обмежень бюджету та часових рамок розробки. Результати експериментального дослідження дозволяють сформулювати конкретні рекомендації для різних сценаріїв застосування.

Таблиця Б.1 у Додатку Б має матрицю прийняття рішення щодо вибору архітектури розроблену за результатами проведених есперементів. Її слід використовувати при виборі архітектури безпеки для IoT-систем.

Аналіз результатів експериментів дозволяє сформулювати рекомендації щодо вибору архітектури для різних класів IoT-пристроїв.

Для критичних систем у галузях медицини, фінансів та автомобільної промисловості рекомендовано використання ARM з TrustZone. Апаратна ізоляція є необхідною для відповідності регуляторним стандартам FDA, PCI-DSS та ISO 26262.

Експериментальні результати підтверджують, що SAU апаратно блокує спроби несанкціонованого доступу, забезпечуючи неможливість читання пам’яті SW з боку NW. TrustZone має сертифіковані реалізації, використовується у промисловості понад 15 років і відповідає вимогам безпеки критичних застосувань.

Поточна реалізація RISC-V, де обидва світи працюють у M-mode, не забезпечує апаратної ізоляції та не може використовуватись у таких системах, оскільки експерименти показали можливість читання захищеної пам’яті з NW.

Для пристроїв споживчого сегмента, таких як системи розумного дому, рекомендовано ARM TrustZone через необхідність захисту криптографічних ключів Wi-Fi, Matter і Thread, а також підвищену ймовірність компрометації NW.

Для пристроїв із низькими вимогами до безпеки, зокрема сенсорів або світлодіодних модулів, може бути застосований RISC-V за умови повного контролю довіреного коду NW і регулярного аудиту.

Для промислових IoT-сенсорів вибір архітектури залежить від критичності даних. TrustZone доцільно використовувати у випадках, коли порушення цілісності або доступ до даних може вплинути на безпеку технологічного процесу.

RISC-V із PMP може бути використаний для некритичних сенсорів, де потрібна гнучкість і масштабованість, за умови ізоляції NW у користувацькому режимі. Поточна реалізація з двома світами в M-mode забезпечує лише логічну ізоляцію й не відповідає вимогам безпечного виконання.

Для академічних досліджень та прототипування доцільно використовувати RISC-V, оскільки відкрита специфікація дозволяє аналіз апаратних механізмів безпеки, модифікацію інструкцій та експерименти з різними моделями ізоляції.

Відсутність ліцензійних витрат і можливість зміни симуляційного середовища роблять цю архітектуру придатною для навчальних і дослідницьких завдань. Результати показали, що PMP працює згідно специфікації, а система коректно генерує винятки при порушенні доступу у користувацькому режимі.

Для пристроїв граничних середовищ і ML-акселераторів рекомендовано ARM TrustZone у поєднанні з існуючими ARM NPU для захисту моделей і даних під час обчислень.

RISC-V може використовуватись для досліджень у галузі енергоефективних ML і створення кастомних акселераторів без вимог до комерційної конфіденційності.

Ключовим обмеженням поточної реалізації mTower на RISC-V FE310 є те, що механізм PMP не забезпечує апаратної ізоляції між SW і NW, оскільки обидва середовища виконуються в одному привілейованому режимі M-mode. У такій конфігурації захист PMP не застосовується до більшої частини NW, тому що FreeRTOS на базі якого розроблений mTower потребує M-mode. Це робить пам’ять SW потенційно доступною для NW.

Для досягнення апаратної ізоляції, еквівалентної TrustZone на ARM, необхідне архітектурне рефакторування системи з перенесенням NW у користувацький режим U-mode та відповідним оновленням механізмів керування доступом.Після міграції доступ NW до пам’яті SW буде апаратно блокуватись із генерацією винятку 5 (Load Access Fault).

План включає адаптацію FreeRTOS для розділення привілейованих і непривілейованих операцій, впровадження ecall як єдиного механізму виклику функцій M-mode та налаштування PMP для захисту пам’яті ядра. Завершальний етап передбачає тестування стабільності, контекстних перемикань і перевірку блокування несанкціонованих доступів.

Подальші роботи мають бути спрямовані на розширення і вдосконалення реалізації безпеки для RISC-V.

У найближчій перспективі основним завданням є завершення міграції NW у U-mode, тестування на фізичному обладнанні та публікація результатів як відкритого внеску до mTower. Також варто виконати порівняльний аналіз на реальних платформах ARM і RISC-V з вимірюванням продуктивності, енергоспоживання та затримок переходів між світами.

Практичні результати підтвердили, що ARM TrustZone забезпечує стабільну апаратну ізоляцію світів і придатна для промислового застосування, тоді як RISC-V є гнучким інструментом для експериментів і прототипування, але поки не гарантує апаратного рівня захисту.

ВИСНОВКИ

У межах кваліфікаційної роботи виконано портування mTower з архітектури ARM на RISC-V та здійснено комплексний порівняльний аналіз механізмів безпеки обох платформ на емуляторі QEMU.

Об’єктом дослідження є програмно-апаратні засоби забезпечення захищеного виконання коду у вбудованих та IoT системах.

Предметом дослідження виступає сукупність методів та засобів портування TEE на архітектуру RISC-V та оцінка їхньої ефективності порівняно з ARM TrustZone.

Наукова новизна роботи полягає у портуванні мінімалістичного TEE для вбудованих та IoT-систем на архітектуру RISC-V, що підтвердило можливість забезпечення рівня безпеки, співставного з ARM TrustZone.

Практичне значення полягає в тому, що результати портування та супровідні патч-серії можуть бути безпосередньо інтегровані в проєкти, сприяючи поширенню RISC-V у сфері захищених вбудованих рішень. Набір тестів та методика валідації можуть слугувати еталоном для майбутніх аудитів та академічних досліджень.

Під час портування реалізовано адаптацію ядра mTower для мікроконтролера SiFive FE310, включно з інтеграцією механізму PMP, системних викликів ECALL, а також налаштуванням інфраструктури рівнів привілеїв.

Експериментальна перевірка проводилась у QEMU та продемонструвала фундаментальне обмеження поточної реалізації RISC-V – PMP діє лише для S-mode та U-mode, але обидва світи SW та NW у реалізованій системі працюють у M-mode, через що PMP не забезпечує апаратної ізоляції між ними.

Тести у U-mode довели, що PMP працює коректно, з генеруванням виключення при порушенні прав доступу, що підтверджує потенціал RISC-V забезпечити апаратну ізоляцію після перенесення NW у U-mode.

Для порівняння проведено тестування оригінальної реалізації mTower на ARM Cortex-M33 із підтримкою TrustZone. Апаратна ізоляція між SW та NW забезпечувалася через SAU. Експерименти показали, що при спробі коду з NW отримати доступ до пам’яті SW система генерує виключення, що доводить апаратне блокування спроб порушення ізоляції незалежно від поведінки NW.

Технологія ARM TrustZone є зрілою пропрієтарною розробкою з понад п’ятнадцятирічною історією промислового використання, тоді як RISC-V залишається відкритим стандартом, який активно розвивається та набуває поширення у наукових і дослідницьких середовищах.

Усі функціональні тести пройдено успішно, часові вимірювання в QEMU виконувалися лише для верифікації коректності, а не для оцінки продуктивності.

Ключовий науковий результат полягає у експериментальному доведенні принципової різниці між підходами до безпеки в архітектурах ARM і RISC-V.

ARM TrustZone гарантує апаратну ізоляцію між SW та NW завдяки блоку SAU, який фізично блокує будь-які несанкціоновані звернення до захищеної пам’яті на рівні процесора.

Натомість RISC-V у поточній реалізації забезпечує лише довірену ізоляцію, оскільки обидва світи виконуються в M-mode, який за специфікацією обходить PMP. Водночас проведене тестування у U-mode довело, що RISC-V має потенціал досягти повноцінної апаратної ізоляції після перенесення NW у U-mode, забезпечуючи поведінку, аналогічну hardware-enforced моделі ARM.

Обидві архітектури займають своє місце в екосистемі IoT-безпеки. ARM TrustZone доцільно використовувати у критичних системах, де необхідна гарантована апаратна ізоляція, тоді як RISC-V є оптимальним рішенням для дослідницьких та некритичних застосувань за умови усвідомлення обмежень trust-based моделі.

Результати дослідження підтверджують можливість побудови повнофункціональних TEE на базі відкритої архітектури RISC-V та надають експериментально обґрунтовані рекомендації щодо вибору платформи безпеки

для IoT-пристроїв із перспективою досягнення апаратної ізоляції після повного переходу NW у U-mode.

Результати дослідження апробовано у вигляді 2 тез доповідей під час III Міжнародної науково-практичної конференції «Research on the development of science and the implementation of technology» [39] та VII Міжнародної науково-практичної конференції «Development of modern scientific technologies in the era of globalization» [43].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ARM, Architecure. "Security technology building a secure system using trustzone technology (white paper)." ARM Limited (2009).

2. Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). Handbook of applied cryptography. CRC press.

3. Ngabonziza, B., Martin, D., Bailey, A., Cho, H., & Martin, S. (2016, November). Trustzone explained: Architectural features and use cases. In 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC) (pp. 445-451). IEEE.

4. Waterman, A., Lee, Y., Avizienis, R., Patterson, D. A., & Asanovic, K. (2015). The risc-v instruction set manual volume 2: Privileged architecture version 1.7 (No. UCBEECS201549).

5. Costan, V., & Devadas, S. (2016). Intel SGX explained. Cryptology ePrint Archive.

6. Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K., & Song, D. (2020, April). Keystone: An open framework for architecting trusted execution environments. In Proceedings of the Fifteenth European Conference on Computer Systems (pp. 1-16).

7. OP-TEE Project. URL: https://op-tee.org (дата звернення 14.10.2025).

8. Drozdovskyi, T. A., & Moliavko, O. S. (2019). mTower: Trusted Execution Environment for MCU-based devices. Journal of Open Source Software, 4(40), 1494.

9. Sabt, M., Achemlal, M., & Bouabdallah, A. (2015, August). Trusted execution environment: What it is, and what it is not. In 2015 IEEE Trustcom/BigDataSE/Ispa (Vol. 1, pp. 57-64). IEEE.

10. Kocher, P., Lee, R., McGraw, G., & Raghunathan, A. (2004, June). Security as a new dimension in embedded system design. In Proceedings of the 41st annual design automation conference (pp. 753-760).

11. Rashmi, R. V., & Karthikeyan, A. (2018, March). Secure boot of embedded applications-a review. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 291-298). IEEE.

12. Pinto, S., & Santos, N. (2019). Demystifying arm trustzone: A comprehensive survey. ACM computing surveys (CSUR), 51(6), 1-36.

13. Shepherd, C., & Markantonakis, K. (2024). Trusted Execution Environments. Springer International Publishing AG.

14. Göttel, C., Pires, R., Rocha, I., Vaucher, S., Felber, P., Pasin, M., & Schiavoni, V. (2018, October). Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS) (pp. 133-142). IEEE.

15. Asanović, K., & Patterson, D. A. (2014). Instruction sets should be free: The case for risc-v. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146.

16. Da Silva, M. (2018). Securing a trusted hardware environment (Trusted Execution Environment) (Doctoral dissertation, Université Montpellier).

17. ARM Trusted Firmware. URL: https://www.trustedfirmware.org (дата звернення 17.10.2025).

18. Yiu, J. (2017). Software development in ARMv8-M architecture. Proceedings of Embedded World.

19. Levy, H. M. (2014). Capability-based computer systems. Digital Press.

20. Ge, Q., Yarom, Y., & Heiser, G. (2018, August). No security without time protection: We need a new hardware-software contract. In Proceedings of the 9th Asia-Pacific Workshop on Systems (pp. 1-9).

21. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

22. Kobylin, O., & Putiatina, O. (2025, April). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. In CEUR Workshop Proceedings (Vol. 3943, pp. 109-117).

23. Ibrahim Daradkeh, Y., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of Images Based on a System of Hierarchical Features. Computers, Materials & Continua, 72(1).

24. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. Advances in Electrical and Electronic Engineering, 21(1), 19.

25. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of effective methods for structural image recognition using the principles of data granulation and apparatus of fuzzy logic. IEEE Access, 9, 13417-13428.

26. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images.

27. Mashtalir, V., & Bogucharskiy, S. (2015). Covering image segmentation via matrix X-means and J-means clustering. Sensors & Transducers, 195(12), 56-61.

28. Bogucharskiy, S., & Mashtalir, V. (2014). On matrix modification of clarans clustering method in large video surveillance databases. Вісник Національного університету Львівська політехніка. Комп’ютерні науки та інформаційні технології, (800), 211-216.

29. Дубницкий, В. Ю., Кобылин, А. М., & Тевяшев, А. Д. (2015). Прямая и обратная задача проектирования технических систем с исходными данными, имеющими неопределённость типа B. Системи обробки інформації, (5), 85-92.

30. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. Telecommunications and Radio Engineering, 79(1).

31. Dubnitskiy, V., Kobylin, A., & Kobylin, O. (2021). Виконання на мобільних пристроях арифметичних операцій з використанням аксіом класичного та нестандартного інтервального аналізу. Advanced Information Systems, 5(3), 128-136.

32. Lyashenko, V., Kobylin, O., Ryazantsev, O., & Ryazantsev, I. (2019). Processing Technique for Biomedical Image Analysis.

33. Кобилін, А. М., Тевяшев, А. Д., & Кобилін, О. А. (2020). Спосіб та система визначення та регулювання параметрів транспорту природного газу на ділянці трубопроводу.

34. Lyashenko, V., Kobylin, O., & Minenko, M. (2018, October). Tools for investigating the phishing attacks dynamics. In 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T) (pp. 43-46). IEEE.

35. Yanholenko, O., Cherednichenko, O., Yakovleva, O., & Arkatov, D. (2020, June). A Model for Estimating the Security Level of Mobile Applications: a Fuzzy Logic Approach. In IntelITSIS (pp. 252-266).

36. Ведмедь, А. Г., Машталир, В. П., & Сакало, Е. С. (2010). Нейронная сеть и алгоритм ее обучения для анализа независимых компонент в задачах обработки изображений.

37. Yakovleva, O., Matúšová, S., Tvoroshenko, I., & Isaiev, Y. (2024). Visitor counting based on video stream analysis from surveillance cameras to solve various business problems. Verejná správa a regionálny rozvoj ekonómia, manažment a marketing, XX (1), 67-87.

38. Yakovleva, O., Kovač, M., Ardasov, V., & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Public Administration and Regional Development, 19(1), 158-184.

39. Грек, Є. (2025). Порівняльний аналіз механізмів апаратної ізоляції PMP/ePMP в RISC-V та ARM TrustZone для IoT-пристроїв.

40. Машталир, В. П., Путятина, А. Е., & Сакало, Е. С. (2008). Автоассоциативная многослойная нейронная сеть и алгоритм ее обучения при сжатии изображений.

41. Dubnitskiy, V., Kobylin, A., & Kobylin, O. (2021). Виконання на мобільних пристроях арифметичних операцій з використанням аксіом класичного та нестандартного інтервального аналізу. Advanced Information Systems, 5(3), 128-136.

42. Putyatina, O., & Sass, J. (2018). Approximation for portfolio optimization in a financial market with shot-noise jumps. Computational Management Science, 15(2), 161-186.

43. Грек, Є. (2025). Про апаратні механізми протидії атакам побічних каналів у RISC-V IoT-пристроях.

44. mTower: Microcontroller Trusted Execution Environment. URL: https://github.com/mTower (дата звернення 19.10.2025).