THALES
Building a future we can all trust

WasmStone

Enabling Transparent Execution of Standalone WebAssembly Applications in Keystone

Quentin MICHAUD, Lukas HERTEL, Louis CAILLIOT, Dhouha AYED, Olivier LEVILLAIN, Joaquin GARCIA-ALFARO
www.thalesgroup.com

ARES 2026

Motivation

> We largely protect data at-rest and in-transit, but not in use

> However, data is more and more processed outside trusted environments (e.g., clouds)

- Example : an AI app served through HTTPS (we want to protect the model and the inputs / outputs)

> Confidential Computing (CC) is a technology allowing to protect data in-use by leveraging hardware mechanisms to secure memory accesses

> Sadly, the complexity of this ecosystem is hindering adoption

> There is therefore an interest in democratizing the adoption of CC for all applications

RESEARCH QUESTION

How to transparently execute various applications in Confidential Computing-protected environments?

Overview of the Confidential Computing ecosystem

> Multiple Confidential Computing (CC) solutions exist with various approaches

> Most Confidential Computing environments are not open

> The application format for a CC solution is not standardized

Table 1. Overview of selected Trusted Execution Environments.

Technology	ISA	Open solution
AMD Secure Encrypted Virtualization (SEV) [1]	x86	✗
Arm TrustZone [3]	ARM	✗
Arm Confidential Compute Architecture (CCA) [2]	ARM	✗
Confidential VM Extensions (CoVE) [29]	RISC-V	✓
IBM Protected Execution Facility (PEF) [17]	PowerPC	✓
IBM Secure Execution (SE) [5]	z/Architecture	✗
Intel Software Guard Extensions (SGX) [18]	x86	✗
Intel Trust Domain Extensions (TDX) [19]	x86	✗
Keystone [22]	RISC-V	✓
Penglai [14]	RISC-V	✓

Introducing RISC-V

- An open and free Instruction Set Architecture (ISA), as opposed to x86 and ARM
- Popular in the research world, making its way towards industry
- Proposes security mechanisms, such as Physical Memory Protection (PMP), useful for designing CC solutions
- Allows to design open CC solutions and assess their security

REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007

Choosing a RISC-V CC solution

- Keystone : an open framework for architecting RISC-V-based Trusted Execution Environment (TEEs)
- One of the most researched TEEs
- Open source and flexible
- Does not require specific hardware

REF xxxxxxxxxxxx rev xxx – date Name of the company / Template: 87211168-COM-GRP-EN-007

GOAL

Design a solution allowing to transparently run standalone applications on top of Keystone

Introducing WebAssembly (Wasm)

> A portable ISA and binary format, initially designed for the Web

> Lightweight, with built-in sandboxing

> Can be used in a standalone mode on a server using WebAssembly System Interfaces (WASI)

> Support a large number of programming languages (C, C++, Rust, Go…)

WASI 0.2 interface overview

Interface	Provided resources and capabilities
CLI	Command-line arguments, environment variables, and standard input and output streams.
Clocks	Read the current time and to measure elapsed time.
Filesystem	Access existing filesystems with an abstraction on their implementation.
HTTP	A transparent abstraction over HTTP version and transport protocol choices.
I/O	Many different kinds of host streams, including files, sockets, pipes, character devices, and more.
Random	Obtain high-quality low-level random data suitable for cryptography.
Sockets	Create a basic functioning TCP/UDP application, with a POSIX compatible interface.

GOAL V2

Design a solution allowing to transparently run standalone Wasm applications (using WASI) on top of Keystone

Keystone inner workings

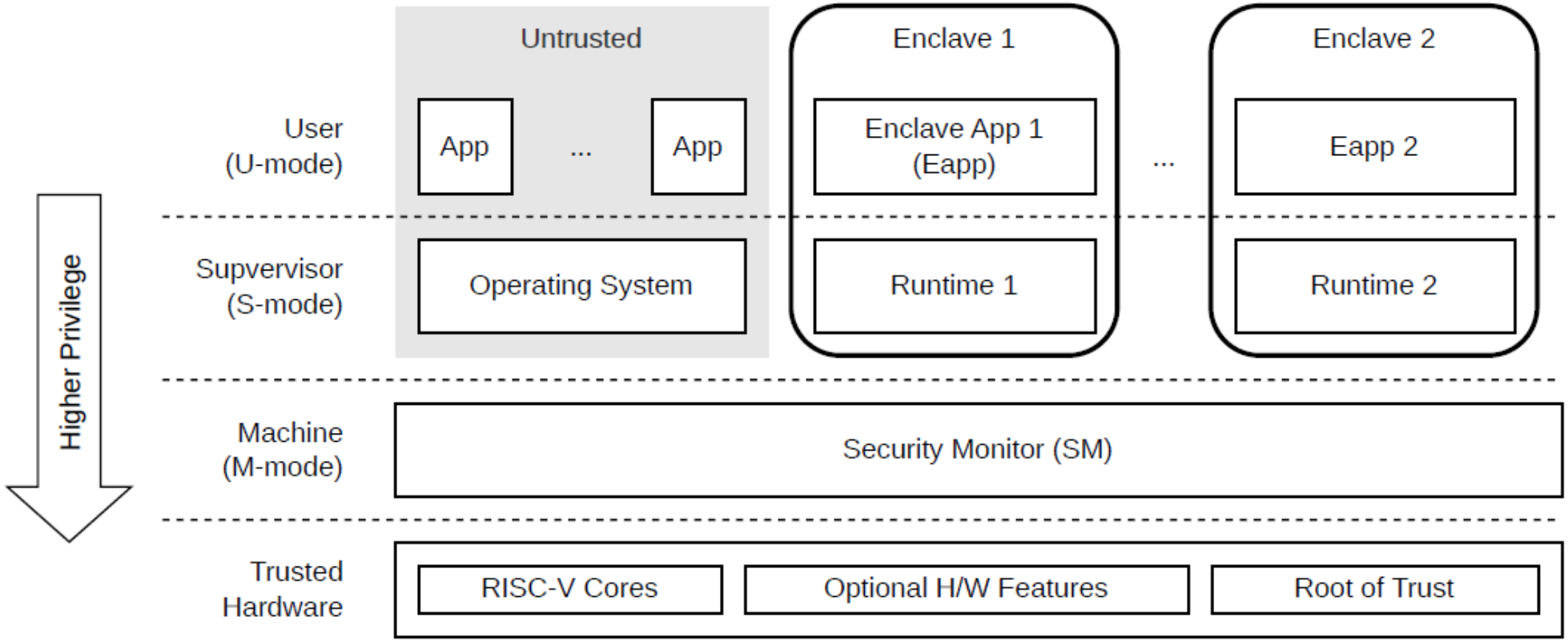


Fig. 1. Overview of the Keystone architecture

Keystone inner workings

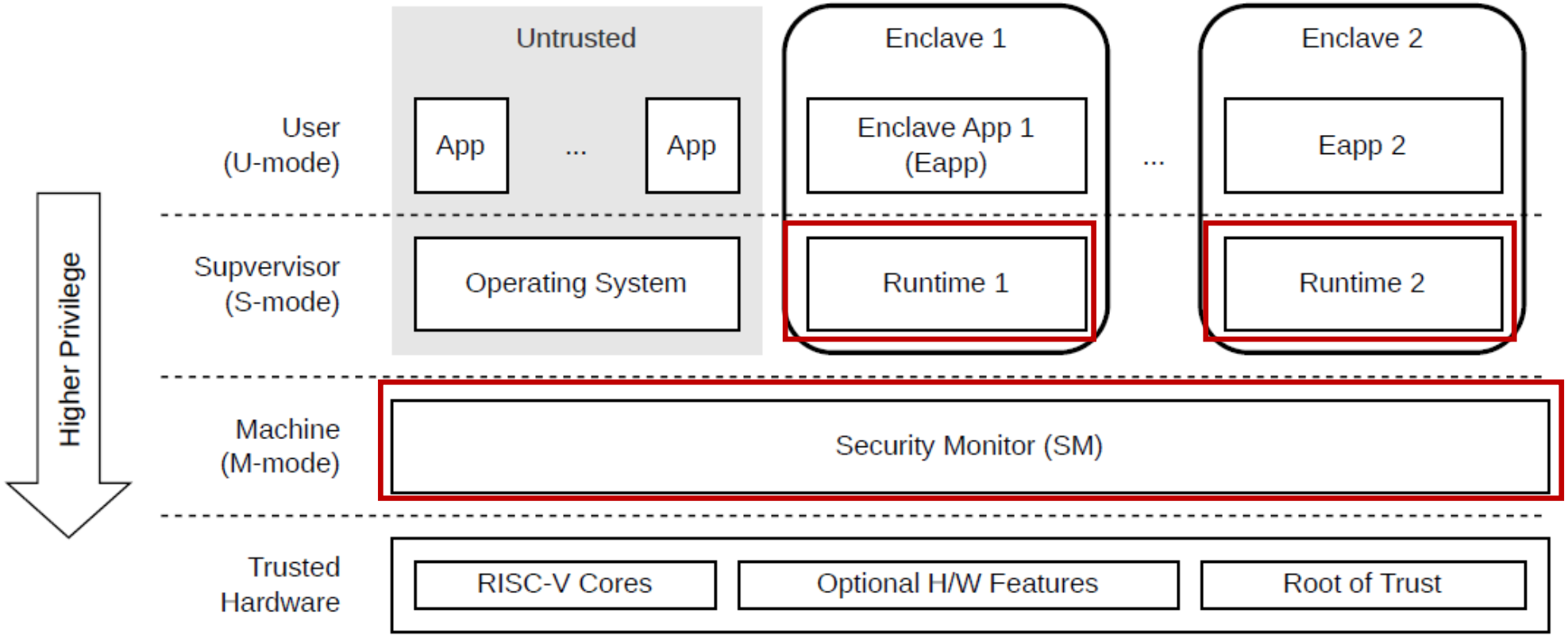


Fig. 1. Overview of the Keystone architecture

Keystone edge calls mechanism

- (1) The Enclave App (eapp) requests a specific action to the Keystone Runtime
- (2) The Keystone Runtime uses the shared memory to transfer the request to the host OS
- (3, 4, 5) The Host App (happ) retrieve the request and act as a proxy to execute the required actions on the host OS
- (6) The result of the operation, along with the potential data, is written back to the shared memory

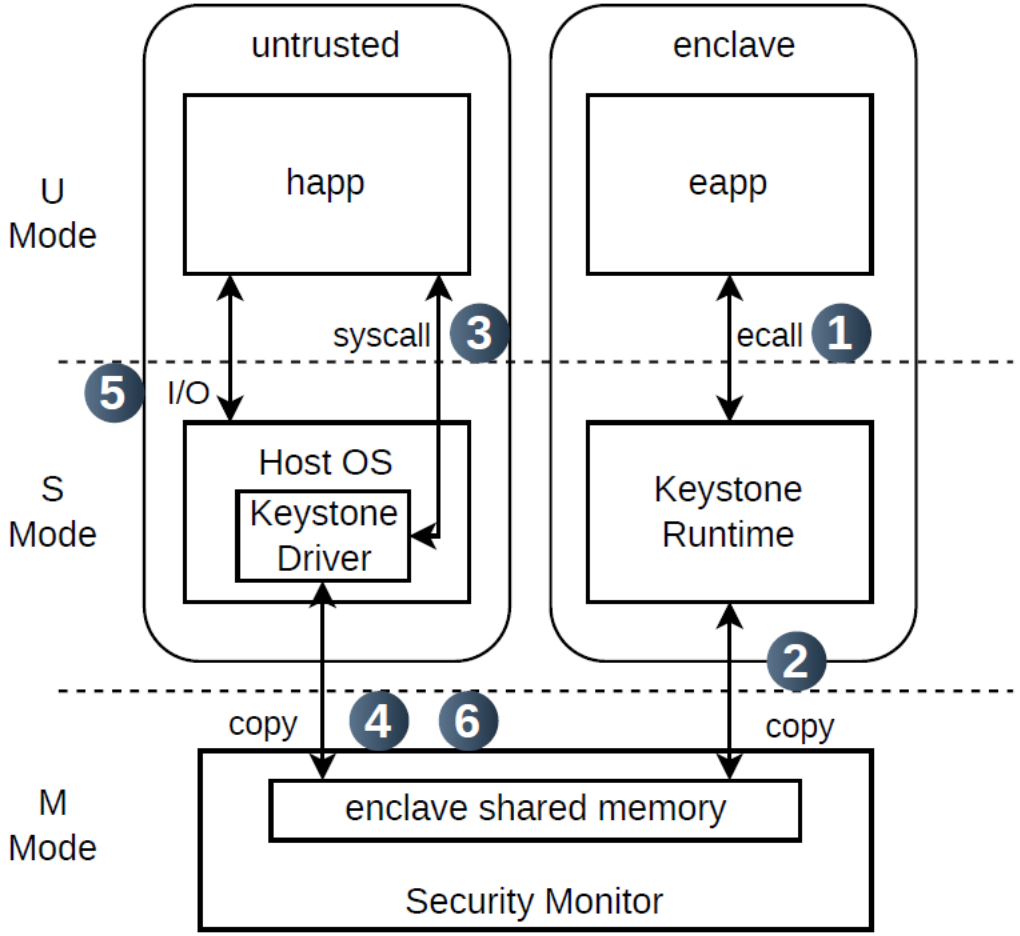


Fig. 2. Overview of the Keystone edge calls mechanism.

Designing a protocol to proxy WASI calls outside the enclave

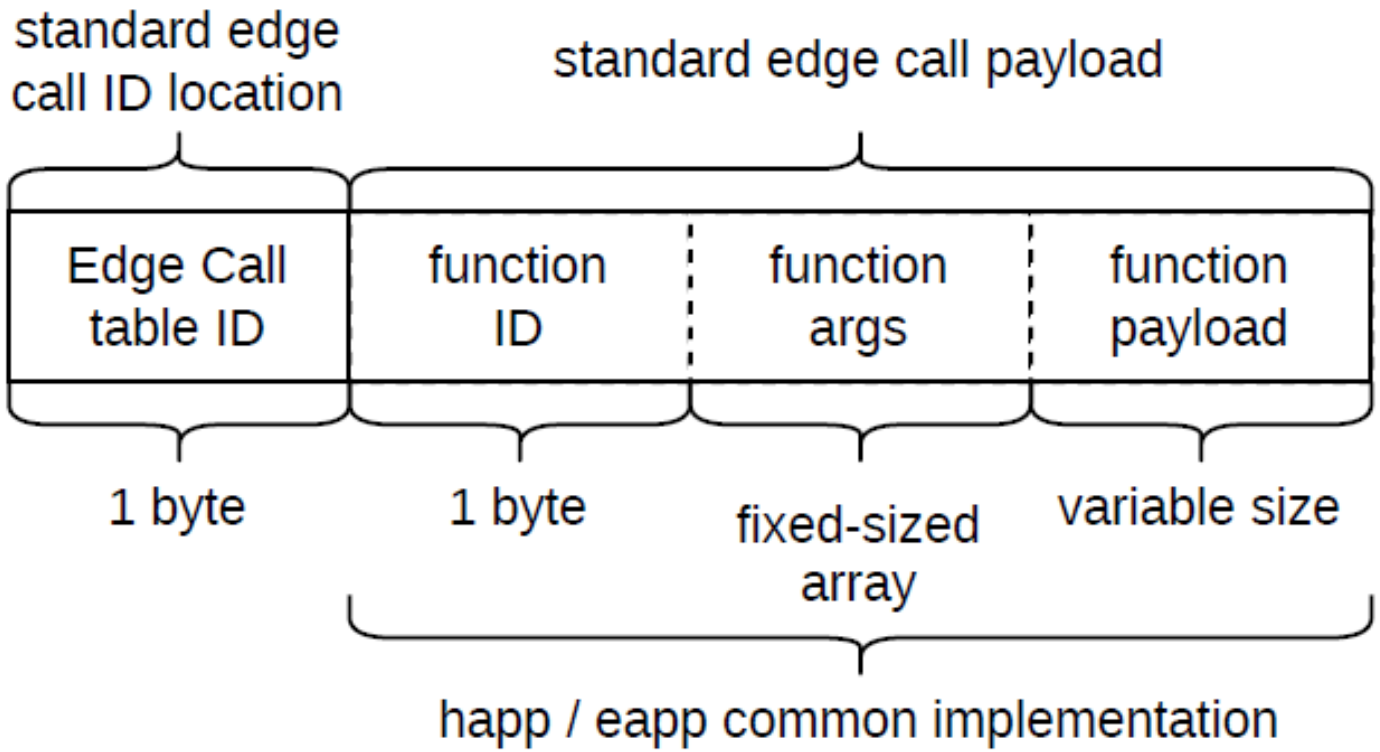


Fig. 4. Edge call structure following the multiplexing approach.

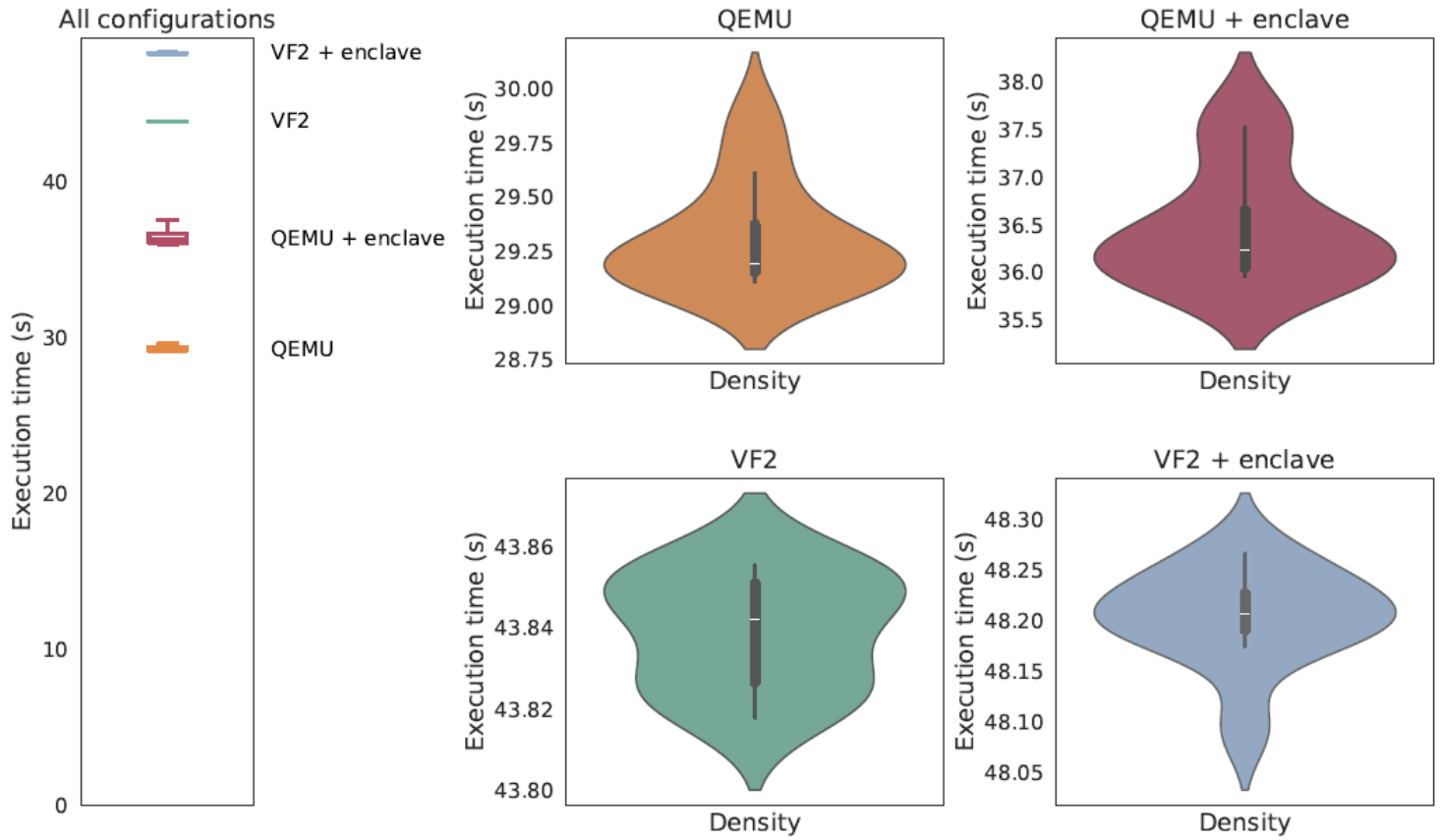
Implementing WasmStone

> 3100 LoC (Rust) for eapp, 900 for happ

> Required careful implementation of polling, time, async… Within the limits of Keystone

> Able to run a web application with an AI classifier

Evaluation



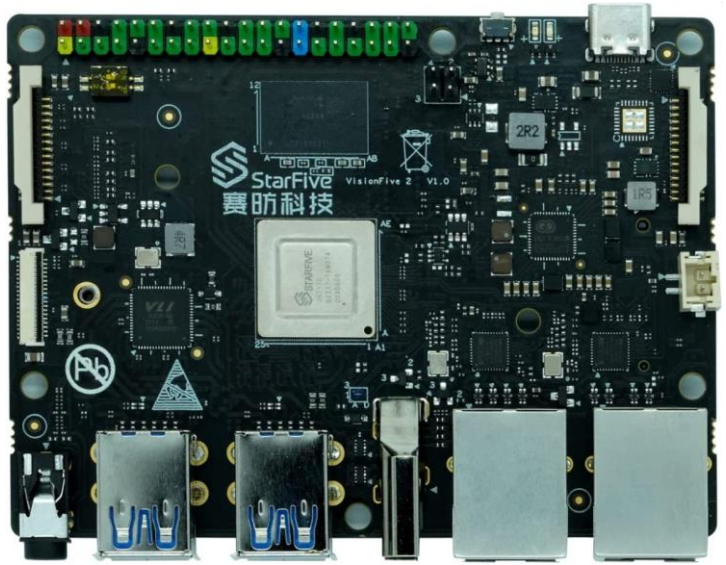


Fig. 5. Performance comparison of the WasmStone evaluation application in several situations.

Future work

- Propose implementations of security-related WASI APIs that fit the CC threat model (e.g., random)
- Add a transparent protection for both data at-rest and in-transit (e.g., filesystem and sockets)
- Extend WasmStone to other CC solutions and other ISAs
- Ongoing work

Conclusion

- Demonstrated possibility to protect unmodified Wasm applications with CC
- The first of its kind on RISC-V and using WASI 0.2
- WasmStone, the demo app and the evaluation code are all open-source
- Hoping to see large adoption of CC in the future

THALES
Building a future we can all trust

Thank you

www.thalesgroup.com