embeddedworld
Exhibition&Conference

A Novel Trusted Execution Environment for Next-Generation RISC-V MCUs

Sandro Pinto
Centro ALGORITMI, UMinho
Guimarães, Portugal
sandro.pinto@dei.uminho.pt

Matjaz Breskvar
Beyond Semiconductor
Ljubljana, Slovenia
matjaz.breskvar@beyondsemi.com

***Abstract*— Security is one of the major challenges within the realm of the Internet of Things (IoT). IoT devices are typically powered by microcontroller units (MCUs) that lack hardware security primitives to isolate security-critical functions from 3rd party less critical components. To make security practical at scale, ARM started to integrate TrustZone technology in the new Armv8-M MCUs (a.k.a. TrustZone-M); however, this technology is still limited to the old-fashioned dual-world architecture and has already been struggling with security problems. In turn, within RISC-V, MCUs still lack a Trusted Execution Environment (TEE) specification, leading to a set of fragmented approaches with restrictive power and performance costs. In this paper, we present a novel TEE solution for next-generation RISC-V MCUs. The proposed system relies on the concept of hardware-enforced, software-defined virtualization-based TEEs. At the hardware level, we describe the novel Beyond BA51-H, the first RISC-V MCU implementing (i) the RISC-V hypervisor extension, (ii) a unified (dual-stage) SPMP extension to protect memory, (iii) and other RISC-V extensions to optimize code size and interrupt latency. At the software level, we discuss the journey of porting the open-source Bao hypervisor to the BA5X-H while implementing the virtualization-based TEE architecture.**

Keywords—Virtualization; TEE; RISC-V; security; Bao, BA51.

I. INTRODUCTION

The world is becoming increasingly connected and thus easily accessible to remote, ill-intentioned hackers and attackers. Internet of Things (IoT) devices are being deployed on a massive scale (e.g., Arm estimates that it will produce a trillion IoT devices by 2035 [1]), generating and sharing large amounts of security- and privacy-sensitive data [2]. The problem is that designing secure IoT devices can be a quandary, with required features (e.g., connectivity stack, over-the-air firmware update, etc.) integrating multiple codebases, drivers, and libraries from different 3rd party entities with distinct assurance levels, on computing units that typically lack reliable mechanisms to enforce the separation among them [3]. Nevertheless, recent attacks on IoT devices have shown us that poorly designed connected devices can bring down critical infrastructures and/or even affect our safety and that the success of this new wave of the Internet is heavily dependent upon the trust built into these billions of connected devices [3-4].

Despite the proliferation of Trusted Execution Environment (TEE), e.g., Intel SGX [5] and Arm TrustZone [6-7], on application processor units (APUs) to secure servers and mobile/embedded devices for two decades, microcontroller units (MCUs) (i) are typically absent of most security hardware primitives or (ii) fail to take advantage of them when they exist [3]. As a step towards securing IoT devices at scale, Arm started to push for the integration of TEE TrustZone technology in Armv8-M MCUs (e.g., Cortex-M23, Cortex-M33); however, TrustZone-M is still limited to the old-fashioned dual-world architecture and has been already reveling security weaknesses [8-10]. In turn, within the RISC-V ecosystem, MCUs still lack a standard Trusted Execution Environment (TEE) architecture or specification, leading to fragmented approaches with prohibitive power and performance costs. For example, MultiZone TEE [11] supports RISC-V MCUs with two privilege levels (Machine- and User-mode), but most of the internals rely on trap and emulation facilitated by the classical virtualizable RISC-V architecture. This naturally has a significant performance impact (due to the frequent M-mode entries and exits), which reflects even further on the power consumption.

In this paper, we propose a novel TEE architecture to secure the next-generation RISC-V MCUs. A central novelty of our TEE design is that we leverage hardware virtualization without virtual memory support as the key underlying security primitive to provide hardware-enforced, software-defined virtualization-based TEEs. This is made possible by the current efforts to extend the RISC-V S-Mode Physical Memory Protection (SPMP) for Hypervisor [12-13]. We have implemented these extensions on the RISC-V golden model, i.e., Spike. We have also extended the Beyond Semiconductor's commercial RISC-V CPU BA51 to support the RISC-V hypervisor extension (without virtual memory), together with the unified (dual-stage) SPMP extension to enforce memory protection. This novel CPU has the codename BA51-H. At the software level, we ported the open-source Bao hypervisor to the BA51-H and implemented the virtualization-based TEE architecture.

II. SCOPE AND BACKGROUND

A. Microcontrollers & Memory Protection Unit

Microcontrollers (MCUs) are low-performance computing units often found in low-end embedded / IoT systems or as housekeeping or platform-management engines in high-performance / high-end SoCs. MCUs do not support virtual memory based on a Memory Management Unit (MMU) and cannot run general-purpose OSes such as Linux. Instead, they provide a Memory Protection Unit (MPU), enabling fine-grained access control over memory regions (with no memory translational functionality). MPUs provide a set of entries, each defining a region's base and bound and access permissions (read, write, and/or execute). The reason for opting for MPUs instead of MMUs on MCU implementations is twofold. First, these are much simpler hardware structures that consume fewer hardware resources, which is critical for the low-cost design point of these computers. Second, since MCUs are typically used for real-time applications, the timing characteristics of memory accesses mediated by an MPU are much more deterministic as it does not require expensive page-table walks with extra implicit memory accesses.

B. RISC-V Supervisor-mode PMP (SPMP)

In the RISC-V lingo, MPU is named Physical Memory Protection. Initially, only the highest privilege mode, i.e., Machine-mode (M-mode), featured a PMP, to enable isolation of firmware and OS-managed resources and can be used as a primitive for static resource partitioning and TEE creation. Due to the need to run real-time OS (RTOS) in Supervisor mode (S-mode) with hardware-enforced user-level task isolation, the RISC-V community proposed the addition of the Supervisor PMP, thus SPMP [12]. The SPMP follows essentially the same design as the original PMP, featuring an implementation-defined number of entries and a mode bit that differentiates between Supervisor and User access permissions. On each context switch, the OS must swap user entries. Depending on the number of regions, this might be an expensive operation; thus, the SPMP introduced a context-switch optimization mechanism unavailable on the original PMP. This mechanism defines a set of registers that enable/disable (bit-wise) SPMP entries. Thus, if enough entries are available, a context switch might resume to a write to these registers, effectively disabling entries related to suspended tasks and enabling entries associated with newly scheduled tasks. As of this writing, the SPMP extension is not yet ratified; however, it has already been submitted for prior consideration of the Architecture Review Committee (ARC). The next step is the public review and then ratification.

SPMP for Hypervisor. The SPMP is currently being extended to support the Hypervisor extension by defining a dual-stage PMP [13]. The first stage, dubbed the virtual SPMP or vSPMP, is controlled by the VMs running in VS-mode, while the second stage is controlled by the hypervisor, effectively enforcing isolation among VMs. The original proposal featured two separate PMPs controlled by HS mode, where the (i) first (the baseline SPMP) mediates accesses from HS/HU, and the (ii) second, dubbed hgPMP, mediates access from VMs, i.e., from VS/VU modes. However, a second proposal introduced a unified model where only a single SPMP is controlled by the hypervisor. In this model, execution under either VS- or VU-mode mimics the execution of a user-mode application on the baseline SPMP. Besides being a much cleaner extension, this approach prevents the a priori partition of SPMP entries between the hypervisor and the virtual modes, reducing the waste of possible unused entries if, for example, virtualization is not used. Finally, note that, besides the SPMP itself, the hypervisor must context-switch vSPMP entries on a VM switch, an operation for which no optimization is currently defined.

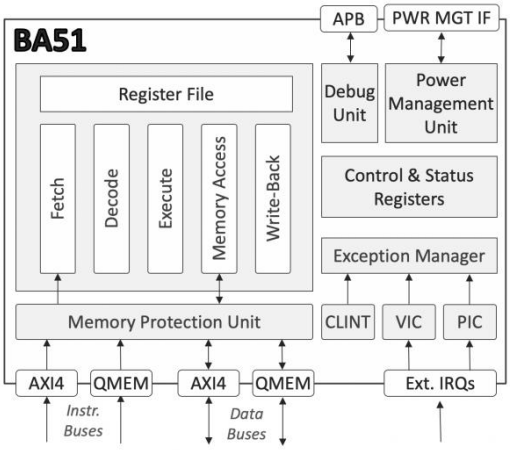


Fig. 1. High-level architecture of the RISC-V BA51 core.

C. Beyond BA51

The BA51 (Figure 1) is a configurable, low-power, deeply embedded RISC-V processor IP core that implements a single-issue, in-order, 2-stage execution pipeline and supports the RISC-V 32-bit base integer instruction set (RV32I) or the 32-bit base embedded instructions set (RV32E). The processor core can be configured to meet different application requirements. For instance, it can optionally support user and supervisor modes, as well as the ISA extensions for Compressed Instructions (C), Integer Multiplication and Division Instructions (M), Atomic Instructions (A), User-Level Interrupts (N), Control and Status Register (Zicsr), and Instruction-Fence (Zifencei); where support for the single-precision floating-point (F) ISA can also be added. Furthermore, the BA51 supports software and timer interrupts and up to 64 external interrupt lines; where the elapsed time from when an external interrupt is asserted until the first instruction in the resolved interrupt handler can be issued is just 4 clock cycles. The core uses AMBA®, AXI-4 and low-latency Quick-access Memory (QMEM) interfaces for fetching instructions and accessing data and peripherals, while the debug unit connects to an external JTAG/TAP controller via an APB port.

The BA51 is designed for low energy consumption. It is compact and enables advanced power management. Under its minimal configuration, the processor size is just 16k gates, and even when most of the optional features are enabled, it is about 50k gates. This small silicon footprint is critical for minimizing leakage currents during idle or standby modes and reducing dynamic power consumption. The BA51 can effectively replace existing 8-bit and 16-bit MCUs or be used as a secondary, housekeeping, or peripheral controller processor in complex

SoC designs. It is suitable for a wide range of deeply embedded applications such as mixed-signal embedded processing (e.g., SERDEScontrol), wireless communications ICs (e.g., Bluetooth, Zigbee, or GPS), and industrial MCUs.

D. Bao Hypervisor

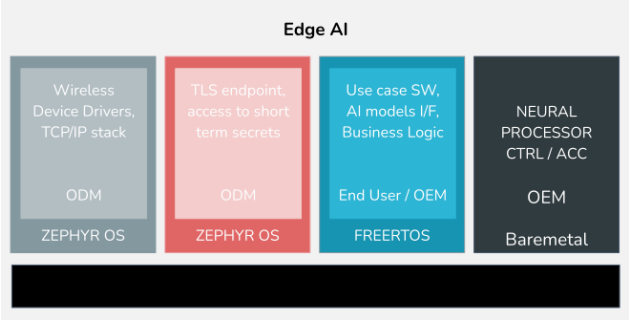
Bao [14] is an open-source static partitioning hypervisor targeting mixed-criticality, real-time systems. Driven by the design principles of minimality and least privilege, Bao was developed from scratch, following the static partitioning architecture [15]. It is statically linked, with no external library dependencies, thus achieving an extremely small code base on the order of 7K-9K Source Lines of Code (SLoC). Bao heavily relies on hardware virtualization support for privilege levels, memory, interrupts, and I/O. Unlike other embedded hypervisors, Bao has no dependence on privileged guest VMs to bootstrap the system and start/manage VMs (e.g., Linux). Given its focus on real-time and mixed-criticality, Bao has featured, from the get-go, a cache partitioning mechanism based on cache coloring that provide non-interference guarantees. Efforts to support memory bandwidth regulation are underway. Initially developed for Armv8-A 64-bit, Bao is one of the few hypervisors supporting the RISC-V Hypervisor extension. Recently, Bao has been extended to support 32-bit and state-of-the-art dual-stage MPU architectures, such as the ARMv8-R Cortex-R52 and the Infineon AURIX TC4x.

III. USE CASES MOTIVATION

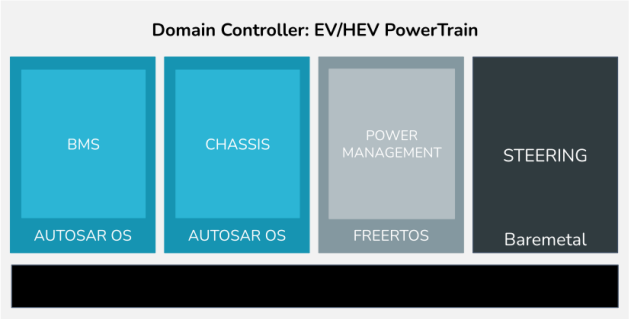
Virtualization enables the consolidation of multiple subsystems into a single-platform, while guaranteeing a strict isolation and fault encapsulation. The addition of virtualization capabilities on computing platforms without virtual memory is a new endeavor. This new profile of computers, sometimes referred to as real-time units, is gaining momentum in mainstream ISAs with the announcement and release of the Arm Cortex-R52 and the Infinineon TC4x. Due to deterministic and real-time guarantees, these computing units are finding applicability in next generation eMobility, ADAS, automotive E/E architectures, and affordable artificial intelligence (AI) applications. Bellow, we describe the high-level system view of two different use cases.

A. Secure AI at the Edge

Figure 2a) presents the high-level system view for a use case on secure AI at the edge. Securing IoT devices encompasses separating and segregating functionalities with different criticalities and/or provided via multiple non-trusted parties, i.e., Original Equipment Manufacturers (OEM), Original Design Manufacturers (ODM), and End-User(s). This microkernel-like philosophy, i.e., decomposing monolithic, fully linked software binaries into small, compartmentalized enclaves, also contains the lateral movement of any potential vulnerability, i.e., it enables consolidation while avoiding/minimizing security issues. In this particular example, there is a clear segregation of the functionalities related to connectivity (e.g., TCP/IP stack), secrets, proprietary AI models or business logic, and access to specific OEM accelerators. Typically, these functionalities are facilitated and/or provided by well-established RTOS, such as Zephyr, FreeRTOS, etc, and thus there is a need for an additional privilege component, i.e., hypervisor, for controlling and managing all guest OSes.



a) Secure AI at the Edge



b) Automotive Domain/Zonal Controllers

Fig. 2. Use cases motivation for virtualization without virtual memory.

B. Automotive Domain/Zonal Controllers

Figure 2b presents the high-level system view for an automotive domain/zonal controller, in particular, for an electric vehicle powertrain. EV/HEV powertrains typically embed a panoply of functionalities with different automotive safety integrity levels (ASIL), such as motor and chassis control, battery management systems (BMS), power management, and steering control. Traditionally, these functionalities would reside on dedicated MCUs found distributed throughout the vehicle. However, as the industry shifts towards domain/zonal-based vehicle architectures, these subsystems are consolidated into the same platform. Virtualization, in particular virtualization for "MMU-less" platforms, can, on one side, provide strict determinism for hard real-time workloads while guaranteeing the required freedom-of-interference required by functional safety (Fusa) standards (e.g., ISO26262).

IV. DESIGN AND IMPLEMENTATION

A. System Architecture

Figure 3 depicts the proposed TEE architecture. A central novelty of the TEE design is that we leverage hardware virtualization primitives to encapsulate subsystems (RTOS and Tasks/Apps) into dedicated VMs, which essentially stand as software-defined TEEs. These hardware virtualization primitives consist of the extra CPU execution modes as well as the extra stage for the SPMP. The RISC-V Hypervisor specification defines an orthogonal design where the S-mode is extended to a hypervisor-extended supervisor mode (HS-mode), and two new privileged modes are added, i.e., virtual supervisor mode (VS-mode) and virtual user mode (VU-mode). As described in section II.B, in the (dual-stage) unified PMP design, the first stage (vSPMP) is controlled by the VM running

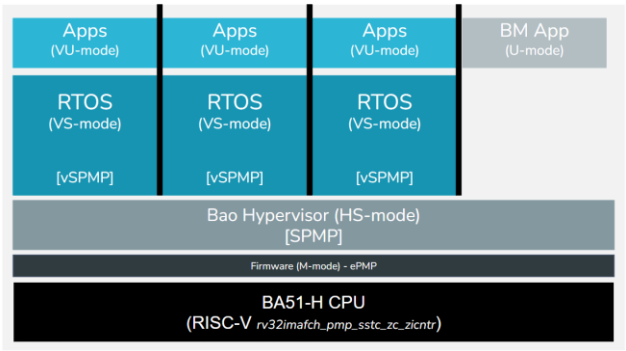


Fig. 3. Virtualization-based TEE architecture for RISC-V MCUs.

in VS-mode, while the second stage is controlled by the hypervisor in HS-mode.

B. *Spike SPMP Hypervisor*

Spike is a RISC-V ISA simulator [16] that provides a functional model for RISC-V cores. It allows one to run RISC-V programs with a variety of different RISC-V extensions, including the RA32I, PMP, and the Hypervisor extension. It is generally considered the golden model of the RISC-V specification.

Before we implemented the BA51-H core, we have first used the Spike simulator to set up a reference execution environment that is as close as possible to the BA51-H core. Having a reference execution environment is extremely useful as it allowed us to test the implementation BA51-H against the reference environment and, at the same time, allowed us to start porting Bao to the referenced execution environment before the BA51-H was implemented. Hopefully, the Spike simulator already supported most of the extensions needed for the referenced execution environment. For the initial execution environment, we have configured Spike to simulate a single 32-bit core with *rv32imafch_pmp_sstc_zc_zicntr* ISA with memory mapping disabled. We further extended the Spike simulator with the SPMP for hypervisor extension [17]. It encompassed adding the SPMP and vSPMP-related registers, copying and adjusting the logic of PMP extension, and taking into consideration when the core is in V-mode.

As a final note, we want to highlight that our reference execution environment does not support the simulation of the APLIC interrupt controller. We have considered adding APLIC to the Spike simulator, but because Bao is able to switch seamlessly between the PLIC and APLIC at compile time, we have decided to skip this due to time constraints.

C. *Beyond BA5X-H*

To the best of our knowledge, the BA51-H is the smallest silicon area RISC-V core with hardware virtualization support featuring unified SPMP, Sstc, and APLIC. The silicon area footprint comprises two components: the processor core logic size (gate count) and memory size. For best results, it is key to balance the added gate count of additional processor functionality with memory size reductions (smaller hypervisor code size overhead). Moving functionality from software to hardware (i.e., interrupts delegation) additionally minimizes the overheads associated with virtualization. The BA51-H is compliant with the base RV32IMAFC ISA. It supports several other standard RISC-V extensions, namely: (i) the Zc extension, to reduce the code size by adding to the 16-bit instruction set defined in extension C; (ii) the unified SPMP extension, which adds support for the supervisor mode physical memory protection; and (iii) the Sstc extension, which provides timer services in supervisor mode.

Hardware virtualization support into the BA51-H is mostly supported by supervisor-level architecture, documented by the Hypervisor ISA extension specification. This support required changes to the instruction decoding, the control and status register (CSR) array, and the trap handling mechanisms. Since the BA51-H, by design, does not provide support for memory virtualization, features related to address translation and memory protection were not incorporated into the core. Instead, the SPMP was extended with a dual-stage architecture to enforce memory protection and isolation between hypervisor domains.

We have also implemented Supervisor-mode timer facilities (Sstc extensions) and the Advanced Platform-Level Interrupt Controller (APLIC), enabling interrupt delegation. Implementing these specifications alleviates the hypervisor code size and reduces the virtualization impact on performance to little more than that of an inter-process context switch.

D. *Bao BA51-H support*

Despite being targeted initially at processors featuring a dual-stage MMU, the Bao Hypervisor has recently expanded towards lower-end MPU-based microcontrollers. MPU support first targeted the 32-bit Armv8-R family, which includes virtualization hardware support with a dual-stage MPU. In supporting the BA51-H, Bao has become the pioneering hypervisor running on virtualization-capable and SPMP-based RISC-V processors.

Bao SPMP Hypervisor support. To enable seamless porting between multiple MPU-based architectures, the Bao memory-protection design is architected into two layers. The top layer is an architecture-independent layer where each address space, i.e., the hypervisor and each VM, is assigned a virtual MPU spanning all processors on which that address space is hosted. Then, locally, on each core, the bottom layer multiplexes entries of the multiple virtual MPUs of the address spaces hosted on that core in the physical MPU. This enables both (i) agnostic memory management and checks and (ii) optimized MPU entry management according to the architecture's specific MPU capabilities. Specifically, in RISC-V, this design facilitates the hypervisor to use NAPOT entries, minimizing the use of SPMP entries. However, in the currently implemented unified model, the permission combinations provided by the SPMP do not enable all possible use cases of shared regions between the hypervisor and guest. For example, it is not possible to define a shared region with RW permissions for the hypervisor and RWX for the guest. This is desirable if some protocol requires sharing an (original) guest RWX region between them while limiting execution of such region by the hypervisor, as it might constitute an attack vector. To circumvent this, Bao does not make use of

shared permissions. It duplicates these regions with a flipped mode bit and keeps all guest regions deactivated during its execution, re-enabling them upon guest entry while disabling (most) hypervisor regions and vice-versa upon a guest exit.

Bao CPU sharing. The BA51-H is a single-core processor, and Bao features a purely static partitioning architecture with 1:1 vCPU to pCPU assignment. This specific combination requires CPU sharing by multiple vCPUs to enable the consolidation of various VMs. Currently, an experimental branch of the Bao hypervisor implements vCPU context switch and a simple round-robin preemptive scheduler. However, the available SPMP context-switch optimizations still need to be leveraged, and guest SPMP entries are fully swapped on each scheduling point. Thus, we are currently developing an algorithm to balance SPMP entry allocation to minimize swapped entries.

V. Preliminary Evaluation

Our preliminary validation and evaluation were performed on Spike and on a target FPGA platform, i.e., Arty 100-T. The software stack encompasses Bao v1.0 and baremetal VMs. Bao and the baremetal VMs were compiled using the RISC-V toolchain with Beyond Studio with -O2 optimizations. We focused on BA51-H hardware resources, and Bao TCB size.

BA5X-H Hardware Resources. Figure 4 shows the preliminary synthesis results of BA51 compact configuration (CC) and BA51-H feature-rich configuration (FRC). We have synthesized both designs using the ASIC synthesis tools on an advanced node process with a 100 MHz nominal timing constraint. As is usual and because we are interested in the relative size of a particular feature, we use gate count as a process-independent indication of area. Furthermore, we estimate that one bit of SRAM has the size equivalent to one gate as a process-independent measurement of SRAM area. The *"BA51 compact"* is a compact setup of BA51 with PIC, M-mode only, and C, E, Zicsr extensions, where *"BA51-H FRC"* is a feature-rich setup of BA51 with APLIC (8 interrupts), Debug module, Power management, PMP with 16 entries (16e), unified sPMP with 16 entries (16e), Memory debugger, CLINT, Triggers (8), Sstc, Trace, M-mode, S-mode, U-mode, and H (without virtual memory), A, C, F, H, I, M, N, Sspmp, Sstc, Zc, Zicntr, Zicsr, and Zifencei extensions.

From the preliminary synthesis results, we can see that adding virtualization support (H, PMP (16e) + unified SPMP (16e), APLIC, and SSTC extensions) does account for 5.1% of the area of the BA51-H FRC. But when we add 64KiB SRAM, the percentage of the area used by the added virtualization support drops to 1.4% of the whole area, which is an acceptable trade-off, especially if looking at the alternative of using multiple processing cores to obtain similar isolation guarantees. Furthermore, by using SPMP with 16 entries + unified SPMP with 16 entries instead of only SPMP with 32 entries, we only increased the design size by 490 gates.

Bao TCB Size. In security-critical systems, the size of the hypervisor code, measured in source lines of code (SLoC), is critical. It should be minimal as it is part of the trusted computing base (TCB) of all VMs. As well understood in the literature, a larger TCB typically has more bugs and a broader attack surface, resulting in a higher probability of vulnerabilities. We measured

Feature	Gates	% of (#1)	% of (#2)	% of (#3)	% of (#4)
(#1) BA51 CC	25239	100	4.6	12.2	3.5
(#2) BA51 CC + 64KiB SRAM	549527	2177.3	100.0	266.2	75.2
(#3) BA51-H FRC	206430	817.9	37.6	100.0	28.3
(#4) BA51-H FRC + 64KiB SRAM	730718	2895.2	133.0	354.0	100.0
Hypervisor extension	7685	30.4	1.4	3.7	1.1
PMP (16e) + unified SPMP (16e)	51223	203.0	9.3	24.8	7.0
PMP (32)	50733	201.0	9.2	24.6	6.9
APLIC	1403	5.6	0.3	0.7	0.2
Sstc	904	3.6	0.2	0.4	0.1
Zc	1287	5.1	0.2	0.6	0.2

Fig. 4. The size (in the number of gates) of the preliminary synthesis results of compact BA51 and feature rich BA51-H configuration with and without 64KB of SRAM.

SLoC for the target configurations using *cloc*. Bao for the BA51-H has a total of 6.5K SLoC, which translates into 31.2 KiB of size (.text).

VI. Roadmap and Next Steps

RISC-V SPMP Hypervisor spec. As already mentioned, the vanilla SPMP specification has not yet been ratified. We highlight thus that the SPMP for hypervisor extension is still at a very embryonic stage, with the current proposal shifting from a dual SPMP to a unified (dual-stage) SPMP. We plan to drive and lead the design of this specification among the RISC-V ecosystem and thus to update, expand, and adapt the Spike model, the BA51-H CPU, and the Bao hypervisor port as this specification goes from develop, ARC to public review, and then final ratification. We also plan to build the support for this specification on QEMU.

Multi-Domain Accelerators. Software running inside of the VMs often interface and interact with additional system bus masters that are not a part of the main processing core, e.g., cryptographic accelerators and/or peripheral devices with DMA capabilities. Sharing HW accelerator cores between different VMs can create unexpected information flows that break the isolation between VMs provided by the main CPU(s). This makes sharing of external HW modules problematic in terms of preserving isolation between domains. One way to address such problems is to better control how the information flows between the core and HW accelerator/bus masters. The design of such a mechanism is a topic for a separate article; but on a high-level, such a mechanism can be provided by propagating a part of the current execution context of a core over the interconnect to the external HW accelerators. For example, the BA51-H could expose an additional CSR to which Bao could write an identifier (ID) of the VM currently running on the core. The VM ID can then be propagated to HW modules with each transaction made over the bus interconnect and, based on this propagated information, the HW module could then decide how to handle the transfer based on which VM caused the transfer.

VII. Conclusion

This paper presented a novel TEE architecture to secure the next-generation RISC-V MCUs. The proposed system is centered on hardware-enforced, software-defined virtualization-based TEEs, facilitated by upcoming hardware virtualization primitives on RISC-V MCUs without virtual memory (but with SPMP). We have pioneered a reference architecture that includes (i) the open-source Spike model, (i) a commercial RISC-V CPU, i.e., BA51-H, and (iii) the open-source Bao hypervisor, which we expect to deploy on a commercial pilot use case for secure AI at the edge.

ACKNOWLEDGMENT

The authors would like to thank José Martins, Žiga Putrle, and Tilen Nedanovski for their contributions, support, and suggestions. This work is supported by (i) FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope UIDB/00319/2020 and (ii) European Union's Horizon Europe research and innovation program under grant agreement No 101070537, project CROSSCON (Cross-platform Open Security Stack for Connected Devices).

REFERENCES

[1] P. Sparks, "The route to a trillion devices - The outlook for IoT investment to 2035," Arm White Paper, 2017.

[2] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," IEEE Internet of Things Journal, 2014.

[3] X. Tan, Z. Ma, S. Pinto, L. Guan, N. Zhang, J. Xu, Z. Lin, H. Hu, Z. Zhao, "Where's the''up''?! A Comprehensive (bottom-up) Study on the Security of Arm Cortex-M Systems," arXiv preprint arXiv:2401.15289, 2024.

[4] O. Alrawi, C. Lever, M. Antonakakis and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments." IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019.

[5] V. Costan and S. Devadas, "Intel SGX Explained." IACR Cryptology ePrint Archive, 2016.

[6] S. Pinto and N. Santos, "Demystifying Arm TrustZone : A Comprehensive Survey," ACM Comput. Surv., vol. 51, no. 6, pp. 1–36, 2018.

[7] David Cerdeira, Nuno Santos, Pedro Fonseca, Sandro Pinto. "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems." IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2020.

[8] Z. Ma, X. Tan, L. Ziarek, N. Zhang, H. Hu and Z. Zhao, "Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense," ACM/IEEE Design Automation Conference (DAC), 2023.

[9] D. Oliveira, T. Gomes and S. Pinto, "uTango: An Open-Source TEE for IoT Devices," in IEEE Access, vol. 10, pp. 23913-23930, 2022.

[10] C. Rodrigues, D. Oliveira and S. Pinto, "BUSted!!! Microarchitectural Side-Channel Attacks on the MCU Bus Interconnect," in IEEE Symposium on Security and Privacy (SP), 2024.

[11] Cesare Garlati and Sandro Pinto,"Secure IoT Firmware For RISC-V Processors". In Proceedings of the Embedded World Conference, Nuremberg, Germany, 2021.

[12] Dong Du et al. "RISC-V S-mode Physical Memory Protection (SPMP)", Specification, v0.9.1., 2023. [Online:] https://github.com/riscv/riscv-spmp/releases/download/v0.9.1/rv-spmp-spec-v0.9.1.pdf

[13] Dong Du and Sandro Pinto. "RISC-V S-mode Physical Memory Protection for Hypervisor", Specification, v0.1, 2023. [Online:] https://github.com/riscv/riscv-spmp/blob/main/spmp-for-hyp/rv-spmp-for-hyp-spec.pdf

[14] J. Martins et al., "Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems," in Workshop on NG-RES, 2020.

[15] José Martins and Sandro Pinto, "Shedding Light on Static Partitioning Hypervisors for Arm-based Mixed-Criticality Systems," in Proc. Of RTAS, 2023.

[16] RISC-V. "RISC-V Spike Repository". [Online:] https://github.com/riscv-software-src/riscv-isa-sim.git.

[17] Žiga Putrle et al., "RISC-V Spike SPMP for Hypervisor Repository". [Online:] https://github.com/crosscon/riscv-isa-sim.