TEE Boot Procedure with Crypto-accelerators in RISC-V Processors

Ckristian Duran
University of Electro-Communications
Tokyo, Japan
duran@vlsilab.ee.uec.ac.jp

Trong-Thuc Hoang
University of Electro-Communications, National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
thuc@vlsilab.ee.uec.ac.jp

Akira Tsukamoto
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan
akira.tsukamoto@aist.go.jp

Kuniyasu Suzaki
National Institute of Advanced Industrial Science and Technology, Technology Research Association of Secure IoT Edge Application based on RISC-V Open Architecture
Tokyo, Japan
k.suzaki@aist.go.jp

Cong-Kha Pham
University of Electro-Communications
Tokyo, Japan
phamck@uec.ac.jp

ABSTRACT

In this paper, a Trusted Execution Environment (TEE) boot procedure with RISC-V processors and crypto-accelerators is presented. The RISC-V system consists of dual cores of Rocket Chip and an SHA-3 accelerator connected on the peripheral bus. Together with the Ed25519 computation on software, the TEE boot procedure, which based on the Keystone framework, is implemented. The Keystone framework provides TEE that can protect data by taking advantage of the Physical Memory Protection (PMP) of the RISC-V ISA. The completed system is built and tested on an Altera Field-Programmable Gate Array (FPGA). The experimental results show that the calculation process for any bootloader payload to authenticate can be reduced about 2.5 decades of milliseconds in comparison with pure software approaches.

ACM Reference Format:
Ckristian Duran, Trong-Thuc Hoang, Akira Tsukamoto, Kuniyasu Suzaki, and Cong-Kha Pham. 2020. TEE Boot Procedure with Crypto-accelerators in RISC-V Processors. In *CARRV '20: Workshop on Computer Architecture Research with RISC-V (with ISCA 2020), May 30, 2020, Valencia, Spain.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A Trusted Execution Environment (TEE) prevents unauthenticated code from running by using hashing, certificate signing, and cryptography. A clear example of this environment is the boot procedure, which authenticates a bootloader by using manufacturer-given keys to sign the payload from an untrusted device such as external flash memories. The security data is often protected over address regions by scaling the privilege level in a processor. In the RISC-V Instruction Set Architecture (ISA) specification, a machine mode runs first when the processor is in a reset state, then escalates to the user mode, protecting the access of previously configured memory regions using a Physical Memory Protection (PMP) scheme [1].

The TEE can be performed in several ways, and each one composes a framework that limits the communication in the system by trusted privileges. An enclave framework uses memory protection in user mode to store sensitive data in selected areas of the address space. TrustZone in ARM can be configured to protect memory address spaces through a barrier [4]. Sanctum is an enclave that isolates on software level over memory pages utilizing a memory translation modification in hardware [10]. Mi6 protects cache coherence attacks by using speculative out-of-order multicore processors and sanctums [9]. Keystone is a framework for RISC-V processors that takes advantage of the physical memory protection standard to authenticate the execution of programs in a safe environment [3]. TIMBER-V is also an isolation scheme for RISC-V processors that provides security with low overhead [8].

In this paper, we present a system for the TEE boot implemented in a high-end Altera Stratix IV Field-Programmable Gate Array (FPGA). The system consists of dual cores of Rocket Chip [7] with cryptography accelerators. The chosen RISC-V ISA is the RV64IMAFDC extensions. The RV64IMAFDC extensions stand for 64-bit RISC-V ISA with **I**nteger, **M**ultiplication, **A**tomic, **F**loating-point, **D**ouble, and **C**ompress instructions. The hardware SHA-3 [6] is utilized and connected to the system via the peripheral bus. The Ed25519 [2] calculations is implemented on software-level. The Keystone framework [3] is applied for the TEE boot procedure. The experimental results show that the calculation process for any

CARRV '20, May 30, 2020, Valencia, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

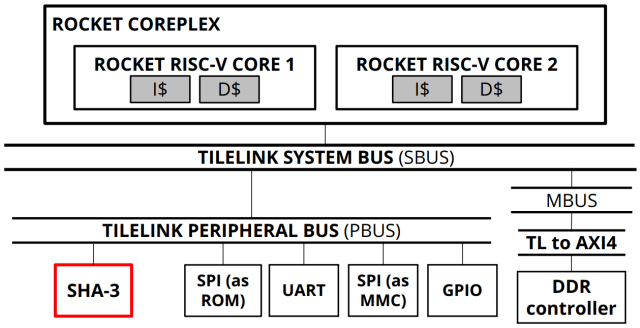

Figure 1: Hardware architecture of the SoC TEE.

bootloader payload to authenticate is reduced about 2.5 decades of milliseconds comparing to pure software computation.

The remainder of this paper is organized as follows. Section II describes the proposed system on FPGA. Section III explains the TEE boot procedure. Section IV gives the result. And Section V concludes the paper.

2 HARDWARE IMPLEMENTATION

The system for the TEE features RISC-V processors implementing the RV64IMAFDC instruction sets [1]. We use the Rocket chip generator as the hardware platform to integrate the security accelerators [5, 7]. Fig. 1 presents the computer architecture of the security system. We implemented two RISC-V cores supporting integer, multiplication, and floating-point operations. The system contains several peripherals attached to a limited TileLink peripheral bus such as GPIO, MMC controller, UART, and ROM. The security cores are located in the peripheral bus, whose configurations and data I/O are memory-mapped using register routers.

Fig. 2 shows the peripheral architecture of the SHA-3 accelerator. This accelerator contains a padding module and a Keccak-1600 round calculator [6]. The padding module retrieves 64-bit data from the register-router, then is pushed through the 576-bit buffer using a shifter. When the buffer is full, the accelerator performs a round calculation. A constant counter keeps track of the number of rounds and constant non-linearity of the iota phase of the Keccak round. The first round is calculated from the first 64-bit data push through the padding module. Every round state is stored in a 1600-bit status register. When the final data is pushed through the padding module, the round calculation performs the final rounds in the status registers. Then the first 512-bit word can be used for the hash output of the calculation.

3 TEE BOOT PROCEDURE

The TEE boot process is based on the Keystone platform [3]. The platform creates public and private keys from the manufacturer's keys. This platform uses these keys to generate a signature of the bootloader to authenticate. The Keystone procedure is stored in a boot ROM on a fixed memory range where the multiple processors point the reset vectors, often named the Zero-State BootLoader (ZSBL). After the ZSBL stage, another bootloader usually utilized called the First-Stage BootLoader (FSBL). Then after the FSBL stage,

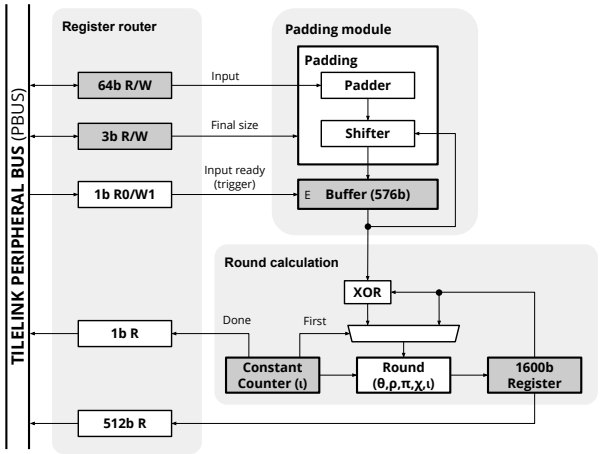

Figure 2: Hardware architecture of the SHA-3 accelerator.

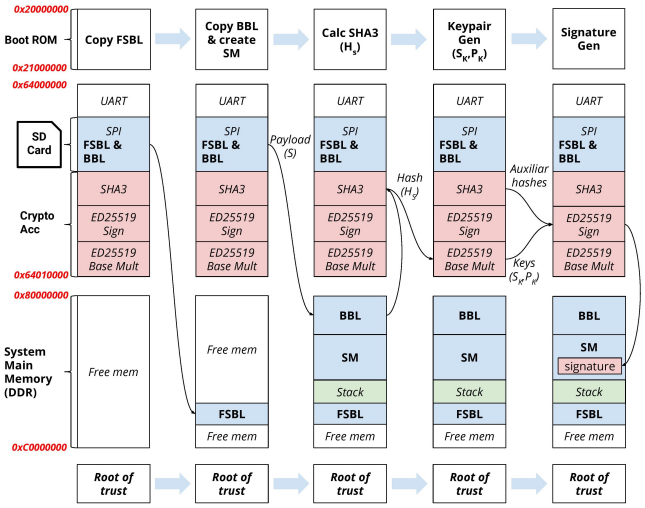

Figure 3: The procedure of TEE boot and root of trust.

the system runs a Linux bootloader like the Berkeley BootLoader (BBL) or OpenSBI, together with the Linux kernel and the initial ram file-system. The FSBL and BBL can be stored in any external media as a payload to authenticate. In this case, the payload is located in an external SD-card driven by an SPI controller.

Fig. 3 describes the TEE boot process using the previous hardware architecture. First, the ZSBL in boot ROM locates and copies the FSBL from the external media (SD-card) to the main memory. Then, it jumps to the FSBL and executes there. After that, the FSBL locates and copies the BBL from the SD-card to the main memory. It also creates the Secure Monitor (SM) in the memory. The SM then extracts the initial seed for the key pair by hashing the copied BBL payload. The hash calculation is performed via the memory-mapped SHA-3 previously described in Fig. 2 by pushing 64-bit chunks of data through the input register. The result of the hash is pushed to the Ed25519 base point multiplier for generating the

Table 1: Synthesis results of the Stratix-IV GX Altera FPGA.

	SHA-3	Rocket Tile
ALUTs	8108	24332
FFs	2790	15325
RAM Bits	0	17680
DSP	0	32
Total	10898	57369
Logic Util. (%)	3.4	12.4
RAM Util. (%)	0.0	1.0
DSP Util. (%)	0.0	2.4

public and secret pair of keys. With the help of the SHA-3 accelerator and the newly created keys, the SM performs the signature of the BBL payload then stores the result in a secure memory address for further use. At this point, the Linux kernel can be boot by executing the BBL. Finally, after boot, the BBL authentication can be done by the attestation function provided by the SM via signature verification.

4 RESULTS

The proposed system is implemented in the DE4 Altera FPGA with the Stratix IV GX EP4SGX230 FPGA chip. The built results are given in Table 1. Most of the logic utilization in the FPGA is occupied by the Rocket tiles with 12.4%, followed by the SHA-3 accelerator with 3.4%. The Rocket tiles contain floating-point logic, integer multipliers, and dividers that synthesizes 2.4% of the DSP resources. The caches for the Rocket tiles contain 4KB of RAM, utilizing 1% of the RAM resources.

The execution environment was tested on the system using both pure-software and hardware-accelerated implementations. We run this environment with several payload sizes to measure time on authenticating the bootloader and the Linux kernel. Fig. 4 presents the overall execution time for the Keystone bootloader to perform the root of trust in the TEE boot for several payload sizes. The payload size includes the Linux bootloader and the Linux kernel, along with the initial file system. This payload was increased on the initial file system by putting random data on a file in several sizes. For both software and hardware implementations, the time increments exponentially. For any stream size, the execution time on the hardware implementation of the Keystone framework decreases about 2.5 decades compared to pure-software.

The algorithm presented in Fig. 3 to authenticate the payload mostly uses the SHA-3 hashing, leaving the Ed25519 keypair and signing as an additional calculation process. We present the Table 2 to offer a broader perspective on the individual calculation execution over a 2MB of BBL. A software-based implementation of both SHA-3 and Ed25519 presents an exponential increase of time compared to a hardware-only SHA-3 in the signature procedure, as the payload needs to be hashed twice. A hardware solution for the SHA-3 does not impact the execution time heavily in the keypair generation, as the hash to calculate is performed over a 256-bit stream.

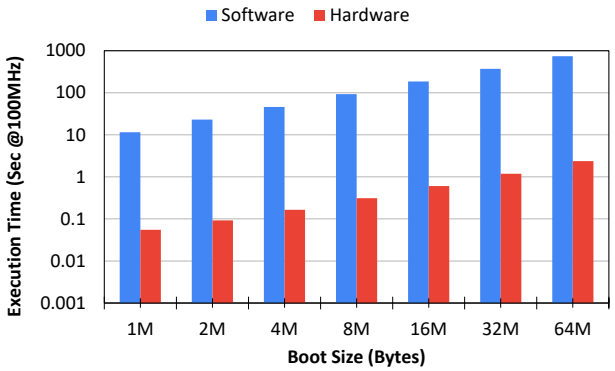

Figure 4: Comparison between software and hardware implementations of the TEE boot.

Table 2: Execution results for the Ed25519 key and signature processes.

2MB bootloader	Software	HW SHA-3 SW Ed25519
Ed25519 keypair (ms)	109.5	93.4
Ed25519 signature (ms)	231019	82.6

5 CONCLUSION

In this paper, a system platform for trusted execution environments (TEEs) featuring the SHA-3 accelerator is presented. The system integrates a RISC-V core with RV64IMAFDC ISA extensions using the Rocket chip generator, including memory protection for Keystone support. The SHA-3 accelerator hashes data using a 64-bit register as input. The software is composed of a root of trust to authenticate a Linux bootloader using Keystone for TEE boot. This software utilizes the accelerators by pushing the data over memory-mapped registers and triggers to obtain calculation results for the authentication signature.

ACKNOWLEDGEMENT

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović. May 2014. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0.* Technical Report UCB/EECS-2014-54. EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html

[2] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. Sep. 2012. High-speed High-security Signatures. *Journal of Cryptographic Engineering* 2, 2 (Sep. 2012), 77–89.

[3] D. Lee, D. Kohlbrenner, S. Shinde, D. Song, and K. Asanovic. 2019. Keystone: A Framework for Architecting TEEs. *CoRR* abs/1907.10119 (2019). http://arxiv.org/abs/1907.10119

[4] E. M. Benhani, L. Bossuet, A. Aubert. Aug. 2019. The Security of ARM TrustZone in a FPGA-Based SoC. *IEEE Trans. on Computers* 68, 8 (Aug. 2019), 1238–1248.

[5] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović. June 2012. Chisel: Constructing Hardware in a Scala Embedded Language. In *DAC Design Automation Conf.* 1212–1221.

[6] National Institute of Standards and Technology. Aug. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.
[7] RISC-V Foundation. 2019. Rocket Chip Generator. https://github.com/chipsalliance/rocket-chip
[8] S. Weiser, M. Werner, F. Brasser, M. Malenko, S. Mangard, and A.-R. Sadeghi. 2019. TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V. In *NDSS*.
[9] T. Bourgeat, I. Lebedev, A. Wright, S. Zhang, and S. Devadas. 2019. Mi6: Secure Enclaves in a Speculative Out-of-order Processor. In *Annual IEEE/ACM Int. Symp. on Microarchitecture*. 42–56.
[10] V. Costan, I. Lebedev, and S. Devadas. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *25th USENIX Security Symp.* 857–874.