Towards a firmware TPM on RISC-V

Marouene Boubakri[1,2], Fausto Chiatante[1] and Belhassen Zouari[2]
[1]NXP, Systems Engineering, Sophia-Antipolis, France
[2]Mediatron Lab, SupCom, University of Carthage, Tunis, Tunisia
boubakri.marouene@supcom.rnu.tn, fausto.chiatante@nxp.com, belhassen.zouari@supcom.tn

***Abstract*— To develop the next generation of Internet of Things, Edge devices and systems which leverage progress in enabling technologies such as 5G, distributed computing and artificial intelligence (AI), several requirements need to be developed and put in place to make the devices smarter. A major requirement for all the above applications is the long-term security and trust computing infrastructure. Trusted Computing requires the introduction inside of the platform of a Trusted Platform Module (TPM). Traditionally, a TPM was a discrete and dedicated module plugged into the platform to give TPM capabilities. Recently, processors manufacturers started integrating trusted computing features into their processors. A significant drawback of this approach is the need for a permanent modification of the processor microarchitecture. In this context, we suggest an analysis and a design of a software-only TPM for RISC-V processors based on seL4 microkernel and OP-TEE.**

Keywords*— *RISC-V, Security, fTPM, Trusted Platform Module, Security, Edge Processing, Edge Security, Processor, Security, IoT Security, Automotive Security.

I. Introduction

Embedded Systems solutions and applications are making a fundamental impact on all sectors of industrial, automotive and edge processing. Therefore, they constitute a crucial element of the Next Generation Internet. Processing speed has been always the first concern for embedded systems because of the resource-constrained nature of embedded devices. However, in order to make the system be used in critical applications, apart from the performance, the security on itself should be the main concern. The system needs to be protected from technical failures and external attacks in a way that prevents damages caused by any modification in its intended behavior.

A widespread method to maintain a platform in a trusted state, which saw a large growth in the last decade, is Trusted Computing, described in more detail in [1]. Under Trusted computing, a specific term introduced by Trusted Computing Group (TCG) [1], a system will always operate in a predetermined way, with hardware and software layers enforcing these behaviors.

Trusted Computing requires the introduction inside of the platform of an external tamper-proof stakeholder in charge of verifying and certifying that the state of the system is intact. This stakeholder is called Trusted Platform Module (TPM) and it was conceived by TCG expressly to support the concept of trusted computing. TPM is an international standard (described in ISO/IEC 11889) for standalone secure crypto-processor designed to protect a system using embedded cryptographic keys. The TPM generates, stores and limits the use of cryptographic keys. It adds resilient security capabilities such as software authentication and remote attestation. In addition, it includes multiple physicals security mechanisms to make it tamper resistant. Therefore, it is the cornerstone for building new security solutions and applications.

The adoption of TPMs has been driven by important commercial choices of key players in the IT market. To highlight a few: Microsoft in 2007 launched Bitlocker, the company's official disk encryption which operates in synergy with TPM to store the needed keys. Google bases the security of the Chromebook laptops on the TPM standard. Juniper Networks allows their routers to use TPM to validate proper boot files and system files. The United States of America's Department of Defense since 2007 requires that every newly purchased PC must include a TPM for security reasons.

These major events brought the TPM standard to become nearly ubiquitous in the personal computer market. However, this was not the case for embedded systems. Embedded systems are designed to fulfill a single very specific task in their entire life cycle. The predictability of the platform function allows the vendor to highly optimize the software and the hardware to limit the requirements and therefore the costs to a minimum. The vendor is, therefore, able to sell its hardware at a very low price to be competitive on a market that has incredibly large volumes compared to the consumer electronics one. The result is the proliferation of highly inexpensive systems with the bare minimum technical requirements needed to develop the target product. In the personal computer market, vendors have great flexibility in costs as they produce a complete product that is sold directly to the customer. The addition of a rather inexpensive TPM chip (around 1$) on a consumer product is in most cases a sustainable effort for the vendor. However, in the embedded system market, often a vendor sells a product that is only a module composing the final system sold to the customer. An example is the vendor of the central computing system of a vehicle, which will be sold to an automotive company in order to produce a consumer product: the car. In this market, the vendor of the embedded system has much less flexibility, as the price of the introduction of an additional chip is more impacting and often unsustainable to maintain competitiveness.

With the advent of the TPM 2.0 specification newer possibilities were introduced for implementing the module. Alongside the regular solution of a discrete chip soldered on to the machine's motherboard, vendors started integrating trusted computing features into their processors. Intel introduced the Platform Trust Technology (PTT) which implements TPM in system firmware. Ryzen Pro CPUs from AMD have built-in TPM functionalities. Another possible solution could be the use of firmware TPMs (fTPM), which are software-only TPMs running on a CPU's Trusted Execution Environment and, finally, virtual TPMs (vTPM).

RISC-V is a promising open source hardware instruction set architecture (ISA) that targets super computers, personal computer, low-power embedded devices and System-on-Chips (SoCs). It enables a new era of processor innovation

through open standard collaboration and its specifications is built from scratch with security and modern use cases in mind. RISC-V features an extremely modular and extensible design that provides enormous flexibility in building application-specific solutions that can leverage custom extensions and only include features that are really required. As an increasing number of vendors now plans to adopt the architecture in their products, implementing a TPM for RISC-V is becoming a significant concern.

The firmware TPM is the paradigm that could decisively introduce this technology inside the embedded system market, finally closing the gap on security standard with the personal computer counterpart. This solution solves the main problem that limited the TPM adoption in the embedded systems sector, namely the cost. A firmware TPM for RISC-V would eliminate the recurring cost of introducing an additional chip on each product sold running a RISC-V core. At the same time, the non-recurring cost of development would be reduced to a minimum due to the great quantities generated of this type of low-cost market.

The main goal of the work is to articulate the future design and the implementation of a RISC-V based fTPM which provides a trust architecture, resilient against software and hardware attacks.

Our main contributions are:

- Exploring the feasibility of implementing a software-only TPM for RISC-V, which performs the same functions as a hardware TPM.
- Analyzing the low-level primitives offered by RISC-V that can be used to build a software-only TPM and therefore build a system with high-level trusted computing semantics.
- Enumerating all challenges and improvements that need to be addressed to build an fTPM for RISC-V.

II. Related Work

With the emerging of the TPM 2.0 specification, the idea of a software implementation started to become a matter of common interest. The progress on this matter was also fueled by the fact that the TPM specification was released with an included codebase. The importance of this introduction resides in the historical lack of a reference implementation: under the TPM 1.2 standard, on the market there were modules with slightly different behaviors, depending on the manufacturer's implementation. The included codebase is part of an effort, made by TCG and Microsoft, to develop a TPM software simulator targeted to be the reference implementation for all manufacturers that aim to adhere to the standard. This introduction also sparked the development of firmware TPMs, a TPM solution completely designed in software, that inherits the security features from the fact that its execution and assets are protected inside a Trusted Execution Environments (TEE). The work carried out by Microsoft has been the main inspiration for this paper that aims at analyzing a firmware TPM 2.0 for the RISC-V architecture, something that has not been done before.

III. Analysis

The fTPM has a well-defined internal architecture, with several required internal components. These submodules are briefly described below.

A. Trusted Execution Environment

Trusted Execution Environment is a term coined by GlobalPlatform in its specification [2] and defined as a tamper resistant enclave in which software runs on a separation kernel. Enclave is a mechanism ensuring secure execution of a software by protecting it not only against other software in the stack (user-space application, operating system and hypervisor) but also against other malicious enclaves running in the same system. Furthermore, the enclave ensures the security even while using shared resources. Enclaves are running under the supervision of a trusted OS. Intel SGX and ARM TrustZone are examples of proprietary enclaves. In general, proprietary enclaves are difficult to experiment with [3]. RISC-V, as an open-source ISA, provides transparency and enables high assurance. Furthermore, it builds security-related task groups to help addressing specific issues such as defining an architecture specification for supporting TEEs. Finally, it provides necessary implementation guidelines and recommendations in order to assist developers to create hardware and software components that support and implement the TEE specification. In this context, efforts have been conducted by researchers and developers to implement an enclaving system for RISC-V. Sanctum [4] is an effort made for PCs. For embedded RISC-V there is Keystone [5] which is an open-source full-stack enclave, MultiZone Security [6] suggested by Hex Five Security and TIMBER-V: "Tag-Isolated Memory bringing fine-grained Enclaves to RISC-V" which delivers a variety of improvements to Keystone and MultiZone. In general, the software built to support TEE is rather simple and light-weight, limiting the attack surface and being able to provide the following four principles of separation: 1) Spatial separation: The inaccessibility of data between different partitions. 2) Time separation: The impossibility to exploit shared resources to leak information in another partition. 3) Control of information flow: The exchange of information is strictly monitored and occurs only when explicitly permitted. 4) Fault isolation: A security intrusion into one of the partitions cannot impact the integrity of the other ones.

This provides the minimal trust that should be enforced by hardware. Enclaved memory regions are mainly based on memory protection techniques provided by RISC-V.

B. Physical Memory Protection (PMP)

Physical Memory Protection (PMP) is a part of the RISC-V Privileged Architecture Specification which describes the interface for a standard RISC-V memory protection unit. The PMP defines a finite number of PMP regions which can be individually configured to enforce access permissions to a range of addresses in memory. PMP restricts the physical memory access of a privilege mode to certain regions defined via PMP entries. PMP is dynamically configured by machine mode, described in the following section.

C. Privilege levels

To enable Trusted Execution Environment, the hardware must support at least a user mode and privileged mode. RISC-V supports 4 privilege levels which are: Machine – M always present and it is the highest privilege mode. Supervisor (S) – Linux support MMU/Virtual memory. Hypervisor – Virtualization support – User/Application – unprivileged lowest level. TEE runs at the highest privilege.

D. Silicon Root of Trust

Silicon Root of Trust (RoT) is the first element of the trust chain and refers to a clearly defined security perimeter—a Trusted Execution Environment (TEE)—within which confidentiality, integrity and authenticity of sensitive or privileged computations and data is upheld. This is achieved through a combination of software, hardware, and cryptographic isolation techniques. The RoT is axiomatically trusted because it is known how it is built and the entity that has built it. Usually this comes in the form of a certificate provided by the manufacturer that guarantees the trustworthiness of the implementation The trustworthiness of TEE is not a condition that can be assured only in the secure enclave, on the contrary, it is a process that starts from a root of trust at the lower levels of the system. The building blocks of the trust of a Trusted Execution Environment are: 1) Secure Boot: the mechanism ensuring that during the boot chain only code that is certified to be intact is allowed to be executed. When a modification is detected the boot process is interrupted. Usually, the integrity of code is verified inside a multi-phase process in which every stage, allowed to be executed only if intact, will verify the following one. The trustworthiness is inherited from a root of trust that must be tamper-resistant by design. 2) Inter-Environment Communication: the mechanism mounted on the system to allow the exchange of information between the TEE and the other environments. This part is fundamental to make a TEE useful on the platform, however, it introduces new threats that need to be stemmed carefully. 3) Secure storage: the TEE main storage support. It must be able to guarantee the authenticity, integrity, confidentiality, and freshness (to protect against replay attacks) of the stored data. It is normally implemented using sealed storage, a mechanism involving a key only owned by TEE, authenticated cryptography over the data being stored and replay-protection mechanisms. 4) Trusted I/O path: the way to protect the authenticity and confidentiality (when needed) of all the performed communication between the TEE and the peripherals.

A successful way of supporting TEE in modern systems is to create already at the silicon level separate environments that pose the bases for the isolation concepts introduced above. This is the case for RISC-V [5].

E. Formally Verified Software

The fTPM solution is completely designed in software. A system is neither resilient nor secure if the underlying software platform is not verified. The first design challenge from software perspective is the introduction of a programmable trusted layer below the untrusted OS. This means that if the fTPM is correctly constructed, in terms of its functionality, then it will never be disrupted or fail due to a faulty kernel or other error prone applications. The kernel is defined as the software that executes in the privileged mode of the hardware, meaning that there can be no protection from faults occurring in the kernel, and every single bug can potentially cause arbitrary damage. seL4 [7] is a microkernel and hypervisor with provable security enforcement which relies on the underlying hardware not only for a separation mechanism but also for premade decision about the boundary between what is trusted and untrusted. For instance, it uses a single isolation domain exposed by the hardware and performs almost all resource management, isolation, and security enforcement in a verified software layer. seL4 implements isolation of components in terms of memory resource and security. Hence, this makes it suitable for our future fTPM design for RISC-V. The principle of seL4 is to isolate applications from each other and execute them on a safe and trusted platform.

F. Cryptographic subsystem

It is an isolated heterogeneous cryptographic engine which includes the following
functions: 1) Asymmetric encryption and decryption, signing and signature verification. supporting RSA and ECC using prime curves. 2) Symmetric encryption and decryption, supporting all the most used block cipher modes. 3) Hashing. 4) Symmetric signing using HMAC, and signature verification. 5) Symmetric and Asymmetric Key generation. 6) Random number generation used for nonces, key generation and signatures.

Several researches are being conducted to add hardware cryptography acceleration on RISC-V [8]. This would open the doors for a potential speed-up in term computational power of the fTPM to support a wide range of applications such as Vehicle-to-everything (V2X) communication.

IV. Design

The ecosystem consists mainly of two worlds, a non-trusted world and a trusted world. The non-trusted world runs a features-rich OS such as Linux. The trusted world runs a trusted OS or a safety-critical software library. Secure Monitor manages the context switching between the worlds. The fTPM runs in secure world as a Trusted Application (TA) in top of the Trusted OS. The fTPM communicates with the underlying OS using TEE APIs. The fTPM uses GlobalPlatform Internal Core API inside its cryptographic subsystem and could use an external cryptographic library, or offload cryptographic operations to RISC-V crypto-extensions. The following figure illustrates the overall architecture of the fTPM. It also shows, in a simplified way, how its components cooperate:

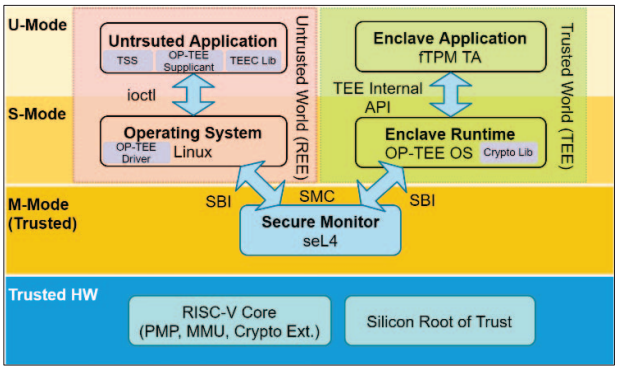

Fig. 1. Design of fTPM for RISC-V. **REE**: Rich Execution Environnement. **TEE**: Trsusted Execution Environeemnt. **TSS**: TPM Software Stack. **SMC**: Secure Monitor Call. **SBI**: Supervisor Binary Interface.

Silicon Roof of Trust: Stored in tamper-proof hardware. At this stage, it is used only to store a unique chip key which will be used for seL4 attestation and secure bootstrapping of the TEE stack. There is various way to implement RoT.

Keystone uses Sanctum's root of trust which uses ECDSA and SHA3 [5]. We are still studying the most convenient way to implement RoT for our design/implementation. OpenTitan [14] could be a good starting point.

seL4: Runs in M-Mode, it is responsible for managing the enclaves and PMP entries to establish a security boundary between worlds. The choice of seL4 is selected based on the strong need for a formal verification of the secure monitor. Keystone implements its own secure monitor on top of Berkeley Boot Loader [12]. Multizone uses a proprietary and closed-source M-Mode runtime. One of the concerns to be considered, is cache side-channel attacks on seL4 [13]. RISC-V could provide side-channel resistant implementations [10].

Enclave runtime: Based on a study [5] which compares existing TEEs and extension, we decided to use Open-source Trusted Execution Environment (OP-TEE) as enclave runtime [11]. OP-TEE is an open-source project which offers a trusted OS, designed to be versatile across different CPU architectures and could be customized to support various real use-case scenarios. OP-TEE like all others RISC-V TEEs, is resilient against software and hardware attacks. It provides flexible resource management including allowing an enclave to perform self-resource management. Unlike Sanctum and Timber-V, OP-TEE does not require microarchitecture modification. Our main motivation is to maintain the portability of existing Trusted Applications including the reference fTPM developed my Microsoft. OP-TEE shows low porting effort compared to Multizone and Timber-V. In addition, it provides the needed interface to accelerate cryptographic algorithms implementations on hardware which make efficient use of proposed RISC-V crypto-extensions. In contrast to Keystone, OP-TEE demonstrates full expressiveness adherence, including forking, multi-threading, system calls and shared memory. OP-TEE does not provide Side channel adversary protection and control channel adversary protection. Porting OPTEE to RISC-V could provide mitigation inherited from underlying core. Currently main activities are only for ARM. The next milestone towards an fTPM for RISC-V is porting OP-TEE relying on standard RISC-V Primitives.

Enclave Application (TA): It is where most of the fTPM will be implemented. When a client application (running in the rich OS) desires to communicate with the fTPM, the TA is loaded into the trusted world by the OP-TEE core. The TAs run at a lower level of privilege compared to the OP-TEE core. They are similar to user-space applications running in Linux, except that they are executed in the trusted world. To be run at the same privilege level of the OP-TEE core, the fTPM could be implemented as a Pseudo Trusted Application (PTA). In this case, the PTA is instead included inside the OP-TEE core tree, and it is built together with the OP-TEE core blob, to which it is statically linked. The drawback of this approach is that the fTPM implementation will not benefit from the GlobalPlatform Core internal APIs and have to rely only on the OP-TEE core internal APIs. In such case, we could consider using a sperate library which implements cryptographic primitives such as mbedTLS.

OS/ Host App: The OP-TEE driver runs on the rich OS. The host application could be a client application consuming the fTPM services, such as OpenSSL accelerated by fTPM or Linux TPM Software Stack (TSS).

To further our research, we plan: porting OP-TEE OS to RISC-V to establish a full-stack TEE solution, writing an OP-TEE dispatcher for seL4, and studying RoT solutions. We hope that further development will prove our design. Available RISC-V software (toolchain, simulator, debugger etc) and hardware VegaBoard with two RISC-V cores are used for development.

V. Conclusion

Due to the increasing demand for security features in modern technology and the growing complexity and connectivity of the produced devices, the challenges posed to protect these systems are bigger than ever. Because of these premises, in the latest years, the market saw a significant shift towards the introduction of trusted computing concepts, not only in regular computers but also reaching a larger sphere that includes also embedded systems and IoT devices. This paper tried to answer if the RISC-V specifications and existing implementations are mature enough to build software services for trusted computing. Although software-only TPM does not provide the same security level as a TPM chip, particularly with respect to physical attacks, executing the fTPM in a protected execution domain of RISC-V core, and using on-chip memory, provides resistance to software attacks (e.g., compromises of the OS), and a limited number of physical attacks. Research efforts are being carried out to provide defense against physical attacks [9], particularly, side-channel attacks [10]. In the other hand, several RISC-V extensions are proposed to leverage security features such as Control Flow Integrity (CFI), memory encryption of bus and addresses, Address Space Layout Randomization (ASLR). These are open problems that merit innovative future research.

References

[1] R. N. Akram and R. K. L. Ko, "Digital trust - trusted computing and beyond: A position paper", 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Sep. 2014, pp. 884–892, DOI 10.1109/TrustCom.2014.116

[2] GlobalPlatform website, https://globalplatform.org/

[3] (Hardware vulnerabilities: Intel SGX - ForeShadow (USENIX'18), AMD SEV - SEVered (EuroSec'18)

[4] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In USENIX Security, 2016.

[5] Keystone: An Open Framework for Architecting TEEs, https://arxiv.org/pdf/1907.10119.pdf

[6] MultiZone Security for RISC-V, https://hex-five.com/multizone-security-sdk/

[7] Gernot Heiser, Gerwin Klein, June Andronick: seL4 in Australia: from research to real-world trustworthy systems. Commun. ACM 63(4): 72-75 (2020)

[8] Ko Stoffelen: Efficient Cryptography on the RISC-V Architecture. LATINCRYPT 2019: 323-340

[9] Mario Werner, Robert Schilling, Thomas Unterluggauer, Stefan Mangard: Protecting RISC-V Processors against Physical Attacks. DATE 2019: 1136-1141

[10] Elke De Mulder, Samatha Gummalla, Michael Hutter: Protecting RISC-V against Side-Channel Attacks. DAC 2019: 45

[11] OPTEE website. https://www.op-tee.org/

[12] RISC-V Proxy Kernel and Boot Loader. https://github.com/riscv/riscvpk

[13] David Cock, Qian Ge, Toby C. Murray, Gernot Heiser: The Last Mile: An Empirical Study of Timing Channels on seL4. CCS 2014: 570-581

[14] OpenTitan wbesite, https://opentitan.org