Demystifying Arm TrustZone: A Comprehensive Survey

SANDRO PINTO, Centro Algoritmi, Universidade do Minho
NUNO SANTOS, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

The world is undergoing an unprecedented technological transformation, evolving into a state where ubiquitous Internet-enabled "things" will be able to generate and share large amounts of security- and privacy-sensitive data. To cope with the security threats that are thus foreseeable, system designers can find in Arm TrustZone hardware technology a most valuable resource. TrustZone is a System-on-Chip and CPU system-wide security solution, available on today's Arm application processors and present in the new generation Arm microcontrollers, which are expected to dominate the market of smart "things." Although this technology has remained relatively underground since its inception in 2004, over the past years, numerous initiatives have significantly advanced the state of the art involving Arm TrustZone. Motivated by this revival of interest, this paper presents an in-depth study of TrustZone technology. We provide a comprehensive survey of relevant work from academia and industry, presenting existing systems into two main areas, namely, Trusted Execution Environments and hardware-assisted virtualization. Furthermore, we analyze the most relevant weaknesses of existing systems and propose new research directions within the realm of tiniest devices and the Internet of Things, which we believe to have potential to yield high-impact contributions in the future.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Security and privacy → Systems security**; *Security in hardware*; Software and application security;

Additional Key Words and Phrases: TrustZone, security, virtualization, TEE, survey, Arm

ACM Reference format:
Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* 51, 6, Article 130 (January 2019), 36 pages.
https://doi.org/10.1145/3291047

1 INTRODUCTION

Arm TrustZone consists of hardware security extensions introduced into Arm application processors (Cortex-A) in 2004 [1, 63]. More recently, TrustZone has been adapted to cover the new generation of Arm microcontrollers (Cortex-M) [65, 113]. TrustZone follows a System-on-Chip (SoC) and CPU system-wide approach to security. This technology is centered around the concept of protection domains named *secure world* and *normal world*. The software executed by the

This work has been partially supported by COMPETE: POCI-01-0145-FEDER-007043, by COMPETE 2020/Portugal 2020/União Europeia within the project Mobile Security Ticketing (No. 11388), which is presented by Link Consulting Tecnologias de Informação SA, and by FCT—Fundação para a Ciência e Tecnologia—within the Project Scope: UID/CEC/00319/2013, UID/CEC/50021/2013, SFRH/BSAB/135236/2017, PTDC/EEI-SCR/1741/2014 (Abyss).
Authors' addresses: S. Pinto, Centro Algoritmi, Universidade do Minho, Campus de Azurém, Guimaraes, 4800-058, Portugal; email: sandro.pinto@dei.uminho.pt; N. Santos, Rua Alves Redol, Lisboa, 1000-029 Lisboa, Portugal; email: nuno.santos@inesc-id.pt.

0360-0300/2019/01-ART130 $15.00
https://doi.org/10.1145/3291047

processor runs either in the secure or non-secure states. On Cortex-A processors, the privileged software referred by the name of secure monitor implements mechanisms for secure context switching between worlds; on Cortex-M processors, there is no secure monitor software and the bridge by both worlds its handled by a set of mechanisms implemented into the core logic. Both worlds are completely hardware isolated and granted uneven privileges, with non-secure software prevented from directly accessing secure world resources. This strong hardware-enforced separation between worlds opens up new opportunities for securing applications and data. In particular, by constraining the operating system (OS) to operate within the boundaries of the normal world, critical applications can reside inside the secure world without the need to rely on the OS for protection.

For several years now, TrustZone has been widely available on commodity mobile devices. Unfortunately, in spite of all its potential for enhancing security, this technology has remained in a state of relative obscurity for quite some time [118, 119]. Manufacturers of TrustZone-enabled SoCs were somewhat reluctant to disclose technical details, oftentimes requiring developers to sign non-disclosure agreements (NDA) [119] before any architectural-related details could be disclosed to them. Research involving TrustZone has been further slowed down by the limited availability of development platforms in which all of TrustZone's capabilities were unlocked. For instance, certain boards were natively programmed to boot the processor directly into the normal world, thereby preventing system developers from deploying code inside the secure world [119]. As a result, for nearly ten years, TrustZone was mostly used by device manufacturers for monetizing proprietary secure services, the security of which was difficult to examine due to their closed nature.

Yet, over the past few years, we have witnessed a growing interest in TrustZone from both academia and industry. TrustZone has been leveraged in many academic research projects and commercial products alike, providing the security foundations for systems such as Samsung Knox [92], Android's Keystore [2], and OP-TEE [58]. Existing projects span across various application domains, notably mobile [52, 109], industry [31, 78], automotive [49], and aerospace [62, 84], and are released under different licensing policies (open-source and proprietary). A dynamic open source community has matured, contributing to the development of various projects based on TrustZone [31, 58, 102], and standardization bodies like the GlobalPlatform [35] have worked toward the definition of common API specifications to promote interoperability across TrustZone-based solutions. The research community, in particular, has been extremely active in exploring new ways to leverage TrustZone as a key-enabler technology for enforcing Trusted Execution Environments (TEEs) [70, 96, 109] and hardware-assisted virtualization [66, 82, 94]. Researchers have also been very keen on studying the security properties of existing TrustZone-based systems and have managed to uncover important vulnerabilities that demand further research to devise effective solutions [67, 99, 112].

An important factor to such a rising interest in TrustZone has been a shift of attitude by major hardware manufacturers, namely Xilinx, which has fostered R&D through the public disclosure of TrustZone technical details and the release of full-featured development boards [120]. Most importantly, this renewed interest can be explained by the potential impact of TrustZone given the widespread adoption of mobile devices and a near-future dissemination of ubiquitous Internet-enabled low-end devices—the so-called Internet of Things (IoT) [6]. Arm processors share the majority of mobile and embedded markets, powering over 60% of all embedded devices and 4.5 billion mobile phones. Arm has further extended TrustZone-support for the tiniest low-end devices, which Arm estimates to reach nearly 1 trillion by 2035 [104]. It is expected that such a plethora of interconnected devices will generate and exchange substantial amounts of data with security-critical and privacy-sensitive content, which tend to attract cybercrime [59, 91]. Similar to competing trusted hardware technologies such as Intel SGX [22], TrustZone can provide

fundamental primitives for securing sensitive data while benefiting from its wide deployment across a large number of mobile and low-end devices.

In this article, we provide a comprehensive study of TrustZone technology. We are driven, on the one hand, by a generalized interest in this technology that, for the reasons expressed above, is foreseeable to prevail for the upcoming years. On the other hand, we are motivated by the absence of systematization of knowledge surrounding this technology. Although there are some works in the literature that partially describe the TrustZone architecture and some of its applications [71, 90], to the best of our knowledge a complete state-of-the-art of TrustZone technology is absent at the time of this writing. We aim at closing this gap, not only by presenting a detailed picture of the most relevant work based on TrustZone but also by providing an insightful discussion on the current limitations and open issues in the use of TrustZone. Furthermore, based on our study and past research experience on TrustZone, we present our vision regarding what we believe to be promising future research directions. In summary, the key contributions of our work are as follows. First, we provide a systematic description of the TrustZone technology itself, covering the main processors and reviewing a few relevant development boards that are fully compatible with and amenable for TrustZone-based systems development. Second, we provide a detailed study of the existing literature, which we structure into two main bodies of work: TEE and hardware-assisted virtualization. Third, we analyze the most relevant reported weaknesses of existing systems and show that additional research is required before TrustZone-based systems can reach full maturity. Fourth, we propose new research directions that seem to us very exciting and potentially resulting in high impact contributions, particularly within the realm of the tiniest IoT devices.

The remainder of this article is organized as follows. Section 2 presents an architectural description of TrustZone technology for both application processors and the new generation of microcontrollers; additionally, some TrustZone-enabled (development) boards are also presented. Sections 3 and 4 provide an in-depth analysis of the state of the art in TrustZone-enabled systems for TEE and virtualization, respectively. Then, Section 5 provides a critical analysis of the most relevant security issues of existing TrustZone systems, and Section 6 discusses promising research directions targeting primarily IoT and cloud environments. Finally, Section 7 concludes this article.

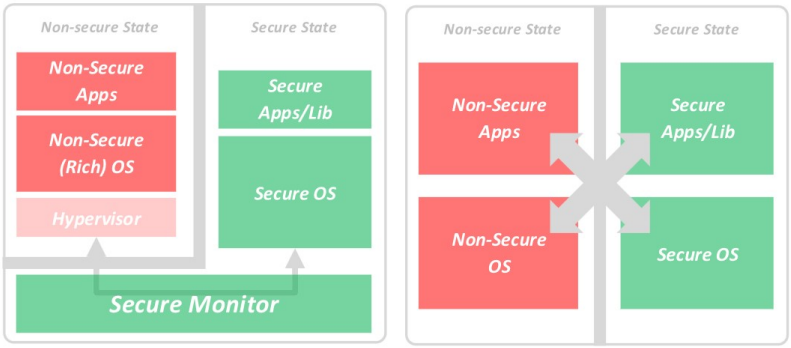
2 TRUSTZONE: HARDWARE AND PLATFORMS

In this section, we provide an overview of TrustZone technology. We start by describing its key architectural features for Arm Cortex-A processors, which can be currently found in a large number of mobile devices (Section 2.1). Then, we highlight the main architectural differences in the new generation Cortex-M processors regarding TrustZone, which was redesigned to accommodate the specific constraints of low-end devices (Section 2.2). Last, we introduce some testbed platforms that can be used for development and research on TrustZone (Section 2.3).

2.1 TrustZone for Application Processors

TrustZone for application processors refers to the hardware-based security built into SoCs to provide a foundation for improved system security for Cortex-A processors [63]. These extensions were added to the Armv6K architecture [1] and introduced significant architectural changes.

The most important architectural change at the processor level consists in the introduction of two protection domains designated by the name of worlds: the secure world and the normal world. Figure 1(a) illustrates these concepts. At a given point in time, the processor operates exclusively in one of these worlds. The world where the processor currently executes is determined by the value of a new 33rd processor bit, also known as the *Non-Secure* (*NS*) bit. The value of this bit can be read from the *Secure Configuration Register* (*SCR*) register, and it is propagated throughout the system down to the memory and peripheral buses. TrustZone introduces an extra processor mode



(a) TrustZone for Cortex-A (b) TrustZone for Cortex-M

Fig. 1. TrustZone technology.

that is responsible for preserving the processor state whenever world transitions occur. This processor mode takes the name of monitor mode, and acts as a bridge for placing the processor in the secure state, independently of the value of the NS bit. A new privileged instruction—*Secure Monitor Call* (*SMC*)—allows for the software stacks residing in both worlds to be bridged by the monitor software. Other than through this instruction, it is possible to enter monitor mode via proper configuration of exceptions, interrupts (IRQ), and fast interrupts (FIQ) handled in the secure world. To reinforce hardware isolation between worlds, the processor has banked versions of the special registers, as well as some system registers (accessed through coprocessor 15 on Armv7-A and using MSR and MRS instructions on Armv8-A). In the normal world, the security-critical system registers and processor core bits are either totally hidden or conditioned by a set of access permissions supervised by the secure world software.

The memory infrastructure has also been extended with TrustZone security features, in particular with the introduction of the TrustZone Address Space Controller (TZASC) and the TrustZone Memory Adapter (TZMA). The TZASC can be used to configure specific memory regions as secure or non-secure, such that applications running in the secure world can access memory regions associated with the normal world, but not the otherwise. Partitioning the DRAM into different memory regions and its respective association with a specific world is performed by the TZASC under the control of a programming interface restricted to the software running with secure world privileges. A similar memory partitioning functionality is implemented by the TZMA, but targeting off-chip ROM or SRAM. Note that the TZASC and the TZMA are optional components defined by the TrustZone specification, which may or may not exist on a specific SoC implementation. Also dependent on the SoC is the granularity at which the memory regions can be specified. Some SoCs with TrustZone extensions include memory controllers that provide limited access to specific memory regions; older SoC implementations do not incorporate such memory controllers at all. Such an example is the Versatile Express platform, which disallows any form of DRAM partitioning into secure and non-secure regions. Modern TrustZone-enabled SoCs, however, come equipped with fully functional TrustZone-enabled memory controllers. Xilinx Zynq and NXP i.MX6 are just a few examples of SoCs that provide full support for TrustZone technology. The TrustZone-aware Memory Management Unit (MMU) allows for each world to have its own virtual-to-physical memory address translation tables. To provide memory isolation at the cache-level, the cache line tags of the processor contain an extra bit that indicates under which world that cache line access has been performed.

TrustZone technology allows for system devices to be restricted to secure or normal worlds. This is achieved with the introduction of a TrustZone Protection Controller (TZPC), which is also an

optional component of the TrustZone specification. The fact that the TZPC is an implementation-specific component leads to diversity in the number and type of TrustZone-aware devices that can be found across hardware platforms. In Xilinx Zynq-based devices, for instance, the Triple Timer Counter 0 (TTC0) cannot be accessed from the normal world, since the TTC0 is permanently restricted to the secure world. The TrustZone architecture extends the Generic Interrupt Controller (GIC) with support for prioritized secure and non-secure sources. Interrupt prioritization is important to prevent denial-of-service (DoS) attacks by non-secure software, since it enables secure interrupts to be handled with higher priority than the non-secure interrupts. Depending on the way the GIC is configured, several interrupt models can be implemented with regard to IRQs and FIQs. Arm proposes to adopt IRQs as interrupt sources pertaining to the normal world and to associate FIQs with interrupt sources from the secure world.

2.2 TrustZone for Microcontrollers

TrustZone technology for Armv8-M [65, 113] has been designed for the new generation of Arm microcontrollers (Cortex-M). At a high level, this variant of TrustZone technology is similar to the variant in Arm Cortex-A processors. In both designs, the processor can execute either in secure or in non-secure state, with non-secure software blocked from accessing secure resources directly. There are, however, important differences between both processor families, namely that Cortex-M has been optimized for faster context switch and low-power applications. In fact, in microcontroller applications, low power consumption, real-time processing, deterministic behavior, and low interrupt latency are mainstream requirements, which lead TrustZone for Armv8-M to be designed from the ground up instead of being reused from Cortex-A processors. As a result, the underlying mechanisms of TrustZone technology for Cortex-M and Cortex-A processors are different.

More specifically, unlike TrustZone technology in Cortex-A processors, the division between worlds in Armv8-M is memory map-based and the transitions take place automatically in exception handling code (see Figure 1). This means that, when running code from the secure memory, the processor state is secure, and, when running code from non-secure memory, the processor state is non-secure. These security states are orthogonal to the existing processor modes, i.e., there are both a Thread and Handler mode in secure and non-secure states. TrustZone technology for Armv8-M excludes the monitor mode and the need for any secure monitor software. This considerably reduces the world switch latency, which translates to more efficient transitions. For bridging software between both worlds, TrustZone now supports multiple secure function entry points, whereas, in TrustZone for Cortex-A processors, the secure monitor handler was the sole entry point. For this purpose, three new instructions were included: secure gateway (SG), branch with exchange to non-secure state (BXNS), and branch with link and exchange to non-secure state (BLXNS). The SG instruction is used for switching from the non-secure to the secure state at the first instruction of a secure entry point; the BXNS instruction is used by secure software to branch or return to the non-secure program; finally, the BLXNS instruction is used by secure software to call non-secure functions. State transitions can also happen due to exceptions and interrupts.

Excepting for stack pointers, in the Armv8-M architecture most of the register file is shared between secure and non-secure states. To separate secure and non-secure stacks, TrustZone-enabled Armv8-M microcontrollers support four physical stack pointers; in this configuration, both security states implements the main stack and the process stack. The starting address of the vector table is determined by a memory-mapped register called the Vector Table Offset Register (VTOR) in the System Control Block (SCB). The VTOR register is banked, which means that one instance exists in each world. Some of the special registers are also banked: the Priority Mask, Control, as well as the Fault Mask Register and the Base Priority registers, just to name a few.

Regarding the memory infrastructure in the Armv8-M architecture, the memory space is also partitioned into secure and non-secure sections. Non-secure addresses are used for memory and peripherals accessible by all software that is running on the device. The secure memory space is further divided into two types: secure and non-secure callable (NSC). Secure addresses are used for memory and peripherals accessible only by secure software. NSC is a special type of secure memory location. This memory area is used to hold SG instructions that allow software to transition between non-secure and secure states. The reason for introducing NSC memory is to prevent other binary data, for example, a lookup table, which has a value the same as the opcode as the SG instruction, from being used as an entry function into the secure state. The security state attributed to each address is determined by the internal Secure Attribution Unit (SAU) or by an external Implementation Defined Attribution Unit (IDAU). The SAU is always present but the number of regions is implementation-specific, while the IDAU is optional and processor-specific. System designers can use an optional IDAU to define a fixed memory map and use an SAU to override the security attributes for some parts of the memory. The SAU can only be programmed in the secure state. The memory partitioning is also used to define peripherals as secure or non-secure. Each world can have a local set of memory access permissions for privileged and unprivileged software. This feature is enabled by the TrustZone-aware Memory Protection Unit (MPU), which provides two distinct MPU interfaces. As in earlier M-series processors, the MPU is an optional component; based on application requirements, designers can exclude the MPU to reduce area and power, or include either a secure or non-secure MPU, or both if necessary. The secure and non-secure MPU can be implemented with a different number of MPU regions.

The Nested Vectored Interrupt Controller (NVIC) was also extended for security. Each interrupt can be configured as secure or non-secure through the Interrupt Target Non-secure register (NVIC_ITNS). This register is only programmable in the secure world. There are no restrictions regarding whether a non-secure or secure interrupt can take place when the processing is running non-secure or secure code. If the arriving exception or interrupt has the same state as the current processor state, then the exception sequence is similar to the previous M-series processors. The main difference occurs when a non-secure interrupt takes place and is handled by the processor during the execution of secure code. In this case, the processor automatically pushes all secure information onto the secure stack and erases the contents from the register banks—this mechanism avoids any leakage of information. Notwithstanding, it is possible to deprioritize non-secure interrupts by setting the PRIS bit field of the Application Interrupt and Reset Control Register (AIRCR) or even avoid handling them while the secure software is running (through the PRIMASK_NS register).

2.3 TrustZone-enabled Hardware Platforms

As TrustZone becomes widespread across all Arm processor families and a key technology for securing small IoT devices, the number of available and cost-efficient TrustZone-enabled (development) platforms seems to follow this trend. Table 1 presents a set of available platforms by comparing them according to five dimensions: name of the platform, designation of the SoC, type of processor, number of cores, and the existence of publicly available TrustZone documentation.

As we can see, a considerable number of the mentioned platforms are endowed with a Xilinx Zynq-7000 SoC. Platforms based on this SoC family have been largely used in both academia and industry, mainly due to its heterogeneity, since it integrates the software programmability of Arm-based processors with the hardware programmability of a Field-Programmable Gate Array (FPGA) [77]. The number of existent Zynq-based development boards is growing: their prices range from less than a hundred (MiniZed) to thousands of US dollars (ZC702). The new generation of Zynq SoCs, the Zynq-based UltraScale+, brings the benefits of the Armv8 architecture and

Table 1. TrustZone-enabled Platforms

Platform	*SoC*	*Processor*	*Multicore*	*Publicly?*
CubieBoard4	Allwinner A80	Cortex-A15/A7	quad-core/quad-core	No
Musca-A1 Board	Arm Musca-A1	Cortex-M33	dual-core	Yes
V2M-Juno r2	Arm Juno (r2)	Cortex-A72/A53	dual-core/quad-core	Yes
SAML11 Xplained Pro	Microchip SAML11	Cortex-M23	single-core	Yes
SAMA5D2-XULT	Microchip SAMA5D2	Cortex-A5	single-core	Yes
MiniZed	Xilinx Zynq-7000	Cortex-A9	single-core	Yes
PYNQ-Z1	Xilinx Zynq-7000	Cortex-A9	dual-core	Yes
ZedBoard	Xilinx Zynq-7000	Cortex-A9	dual-core	Yes
ZYBO	Xilinx Zynq-7000	Cortex-A9	dual-core	Yes
NuMicro M2351	Nuvoton M2351	Cortex-M23	single-core	Yes
Jetson TK1 DevKit	Nvidia Tegra TK1	Cortex-A15	quad-core	No
Jetson TX2 DevKit	Nvidia Jetson TX2	Cortex-A57/Denver	quad-core/dual-core	No
IMX53QSB	NXP i.MX53	Cortex-A8	single-core	Yes
iMX6UL-EVK	NXP i.MX6 UL	Cortex-A7	single-core	Yes
RD-IMX6Q-SABRE	NXP i.MX6	Cortex-A9	quad-core	Yes
MCIMX7-SABRE	NXP i.MX7	Cortex-A7/M4	dual-core/single-core	Yes
Raspberry Pi 3	Broadcom BCM2837	Cortex-A53	quad-core	Yes
R-Car Starter Kit	Renesas R-Car H3	Cortex-A57/A53	quad-core/quad-core	No
ZC702 Eval. Kit	Xilinx Zynq-7000	Cortex-A9	dual-core	Yes
ZCU102 Eval. Kit	Xilinx Zynq UltraScale+	Cortex-A53/R5	quad-core/dual-core	Yes

the power of the 64-bit instruction set; however, at the time of writing of this article, the number of cost-efficient development boards is scarce. So, according to our experience, for those who are interested in building software for mid- to high-end TrustZone-based platforms (Cortex-A), Minized, ZYBO, PYNQ-Z1 or ZedBoard are seen as a good starting point due to the significant number of research papers, open-source projects and technical documentation available, as well as the reasonable selling price. Raspberry Pi 3 is also a very affordable alternative, but the number of available TrustZone-related resources, when compared to the Xilinx Zynq-7000 family, is scarce. NXP also offers interesting platforms, which are widely used by academics and researchers for a reasonable price. Nvidia and Renesas have also presented relevant hardware solutions (e.g., Jetson TX2 DevKit, R-Car Starter Kit Premier), which have been used in industrial settings by some companies [66, 76]. From an academic perspective, these boards are particularly unsuited for TrustZone exploration due to the reluctance of their manufacturers in openly disclosing the technical details regarding their implementation.

Regarding the low-end sector, the market of small IoT devices is still in its infancy. Although Cortex-M23 and Cortex-M33 were announced in Q4 2016, as of this writing, just a few platforms are available on the market. While STMicroelectronics, Renesas, and NXP have not disclosed the technical specifications of TrustZone(-M)-enabled platforms, other manufactureres have taken a different route: Nuvoton has already presented some prototypes of the NuMicro M2351 board, Arm has released the Arm Musca-A1 board, and Microchip has started selling the SAML11 Xplained Pro Evaluation Kit. The NuMicro M2351 features a single-core Cortex-M23 and the SAML11 Xplained Pro Evaluation Kit features a single-core Cortex-M23, while the Musca-A1 test chip features a dual-core Cortex-M33. At the time of the writing of this article, the SAML11 Xplained Pro Evaluation Kit can be purchased for 60 USD; however, the average price for SAML11 Arm Cortex-M23-based

microcontrollers stands around 2 USD, which enables secure small IoT devices (e.g, a smart light bulb or a smart plug) to be implemented at large scale.

3 TRUSTZONE-ASSISTED TEE

In this section, we cover one of the main application areas for TrustZone aimed at the creation of TEE on computer platforms. Next, we start by introducing the TEE concept and explaining how TrustZone is key to realize it. In Sections 3.2 and 3.3, we present, respectively, two design families of TrustZone-based TEE systems: one primarily focused in supporting multiple applications within the TEE, and other on leveraging the TEE to host a single specialized service. Section 3.4 focuses on present efforts do deploy such techniques for cloud clusters and Section 3.5 places TrustZone in perspective against a broader landscape of TEE-enabler hardware technologies. Last, we close this chapter by providing a brief discussion on the main outstanding challenges in the field of TrustZone-assisted TEE.

3.1 TrustZone: A key TEE-enabler Technology

Modern computer systems tend to depend on large trusted computing bases (TCBs). Typically, a TCB comprises complex software such as the OS kernel, privileged services, and libraries. Systems featuring a bloated TCB tend to be more vulnerable to attacks than small-sized TCB systems, because, on the one hand, the likelihood of undetected code vulnerabilities increases as a result of a larger number of lines of source code and more complex inter-component interactions. On the other hand, large TCBs tend to be more exposed to attackers thus opening more doors for effective exploitation of such vulnerabilities. Applications that rely on such complex software platforms inevitably inherit potential security deficiencies of the underlying TCB [111].

To address the TCB bloating problem, TrustZone has appeared as a fundamental hardware mechanism that enables to provide a TEE in which critical applications can execute securely featuring a TCB several orders of magnitude smaller than the rich OS. More specifically, a TEE consists of an isolated environment in which trusted applications can execute without the interference of the local (untrusted) OS. The security properties of a TEE guarantee the confidentiality and integrity of computations that take place inside it. In addition, to enforce isolated execution, a TEE abstraction defines mechanisms for secure provisioning of code and data (including cryptographic keys) into the TEE and trusted channels [116] for retrieving the results of computations and errors. It is also common for a TEE to access some private secure storage space [37] and to allow remote parties to check the integrity and authenticity of the TEE environment through a remote attestation protocol [51]; remote attestation constitutes the basic step for establishing trust between a TEE and a remote party, on top of which secure channels can be established for secure communication to proceed [45, 46].

TrustZone constitutes a natural enabler for building TEE support systems. Essentially, the secure world provides a restricted execution environment where the TEE can reside. Whenever the system boots, the processor enters the secure world to give to any privileged firmware the chance to set up its internal data structures, configure the interrupt controller of the entire system, and set up protections for secure memory regions and peripherals. Upon the completion of these operations, the processor switches worlds and yields control to the bootloader of the rich OS. Since the OS runs in the normal world, it enjoys no privileges to access memory or set up the interrupt table in a way that could gain access to the secure world. A common gateway used to access the secure world is to execute the SMC instruction, which forces the processor to enter into monitor mode. Return to the normal world is performed also through the SMC instruction. From these mechanisms, we can basically run a trusted program inside the secure world without the need to trust the integrity

of the rich OS; if the OS is compromised, attempts to access the secure world address space will result in violations, e.g., resulting in exceptions trapping to the secure monitor.

In addition to world isolation and context-switch capabilities, TrustZone provides building blocks to implement end-to-end security solutions, namely, trusted I/O paths, secure storage, and remote attestation [36, 54, 55]. For trusted I/O paths, TrustZone allows the reflection of the world state of the processor into the peripherals themselves, thereby allowing them to operate in different modes depending on whether the system operates in secure or non-secure states. This is achieved by routing the NS bit state of the processor (which identifies the current world) down to the respective peripheral [120]. In other words, one can configure a device so that data can be routed to/from a specific world. Secure storage can be implemented by installing a data storage component on the device and restrict it to secure world accesses [36]. Likewise, remote attestation is supported by the incorporation of a hardware component (e.g., a Trusted Platform Module) containing trusted code for measuring the integrity of the TEE kernel and unique cryptographic keys [125].

Depending on the trusted program that runs in the secure world, we distinguish two types of TEE architectures: *TEE kernel* or *TEE service*. In the first case, the trusted program implements a basic set of OS functions to manage multiple TEE instances each of them hosting a particular application [96]. The trusted kernel is responsible for: managing memory of the secure word, enforcing memory protection for each TEE, handling communication between TEE and the OS, and providing an API to TEE applications [31, 58, 109]. However, TEE services implement a specific function and do not require any low-level OS logic to manage their own memory and cross-world communication [54, 106]. To prevent mutual interference, only one TEE service can be deployed on the device. This is a disadvantage when compared to TEE kernels, which allow multiple applications to run in independent TEE instances. However, a downside of TEE kernels is that they normally depend on larger TCBs when compared to systems where a single TEE service is deployed. Next, we provide an overview of the state of the art about TEE systems, starting with the solutions based on a TEE kernel architecture, and then focusing on specific- and single-purpose TEE services. Throughout this discussion, we concentrate primarily on solutions targeting mobile platforms. The reason is that the majority of mobile devices are equipped with Arm (application) processors, hence featuring TrustZone technology. In Section 3.4, we cover existing systems and specific challenges when targeting cloud platforms.

3.2 Trusted Kernels for Trustzone-assisted TEE

The basic functionality offered by a TEE system consists of an execution environment where security-sensitive applications can execute in isolation from the rich OS. On TrustZone-enabled platforms, the runtime support for sustaining the lifecycle of such applications is typically provided by a privileged *trusted kernel*, which runs in the secure world. The communication between the rich OS and the trusted kernel requires context switch between worlds. To perform this operation, the rich OS needs to be enhanced with a user-space client API and a TEE device driver responsible for trapping into the trusted kernel.

TEE standardization efforts. Given that rich OS and trusted kernels are not necessarily developed by the same manufacturer and nevertheless need to interoperate with each other, a lot of TEE standardization efforts have been advanced. In 2009, the Open Mobile Terminal Platform (OMTP) took some first steps toward this end by specifying a TEE standard that defines a set of security requirements on the functionality a TEE should support [75]. The GlobalPlatform [35] organization went a step further by defining standard APIs: the internal APIs (e.g., TEE Internal API) that a trusted application can rely on and the communication interfaces that rich OS software can use to

interact with the TEE applications maintained by the trusted kernel. The GlobalPlatform has also defined device specifications (e.g., TEE Client API) that TEEs are requested to abide by. Included in these device specifications there is a trusted UI clause (Trusted User Interface API), which states that every GlobalPlatform-compliant device must provide support for a trusted UI. SierraTEE [102], T6 [110], and Open-TEE [70] comply with the GlobalPlatform standard and for this reason allow the development of trusted applications with secure user interfaces. Open-TEE's trusted UI feature is being developed by the community as it was not originally supported.

Rich TEE runtime systems. To gain competitive advantage, some companies build proprietary and closed-source trusted kernels. Samsung KNOX [92] features among the most representative of such systems. KNOX is a defense-grade mobile security platform that aims to provide enterprise data protection with strong guarantees. Security is achieved through several layers of data protection, which include secure boot, TrustZone-based integrity measurement architecture (TIMA) and Security Enhancements for Android (SEAndroid [103]). Samsung KNOX offers a product called KNOX Workspace, which is a container that provides specific mechanisms for isolating and encrypting work data from attackers. This secure container is available on commodity mobile devices and delivers a complete user-friendly environment, which comprises a specific home screen, applications, and widgets. By providing adequate management tools and utilities, this product has been specifically designed to serve the security needs of enterprises.

Small TEE runtime systems. An important drawback of closed systems like KNOX is that it is hard to evaluate whether or not the security properties claimed by its manufacturers are enforced in practice. Furthermore, since KNOX allows for regular rich applications to execute in the secure world, a large number of runtime functions need to be included into KNOX's trusted kernel. As a result, the system's TCB tends to be very large and thus more prone to be exploited than small TEE trusted kernels [5, 25]. To address this problem, the research community has investigated, for some time now, how to build trusted kernels featuring small code footprints. On-board Credentials (ObC) [50, 52] is one of such solutions, originally developed for Nokia mobile devices using the TI M-Shield technology and later ported to TrustZone. ObC supports the development of secure credential and authentication mechanisms. TLR [96] also shares similar goals while in addition providing simple programming abstractions that allow for certain pieces of application code (trustlets) to be instantiated inside the TEE and seamlessly invoked by the application components residing in the normal world. OP-TEE [58], TLK [114], Open-TEE [70], Genode [53], and AndixOS [31] are yet other TEE systems that reduce the TCB of privileged trusted kernel code. With the exception of TLK, these systems have the merit to be publicly available as open source projects that can be used by the research community.

Unconventional trusted kernels. Unlike the aforementioned solutions, such as Samsung KNOX, in which the isolated computing environments reside in the secure world, TrustICE [109] enables the creation of Isolated Computing Environments (ICEs) in the normal world. For this reason, TrustICE's architecture is slightly different from those described above. Figure 2 compares TrustICE's architecture with that of a traditional TrustZone TEE, where trusted applications run inside the secure world. TrustICE works by implementing a trusted domain controller (TDC), which runs in the secure world and is responsible for suspending the execution of the rich OS as well as other ICE's when another ICE is running. Thus, TrustICE supports CPU isolation for running ICEs. For memory isolation, a watermarking mechanism (the TZASC is accessed through Watermark technique on NXP's i.MX53 QSB) prevents the rich OS from accessing code running in the normal world memory belonging to ICE domains. To isolate I/O devices, the secure world blocks all unnecessary external interrupts from reaching the TDC. With the exception of a minimal set of interrupts that allow for trusted UI, this mechanism helps to protect the TDC