1 Trusted Computing Building Blocks for Embedded Linux-based ARM TrustZone Platforms

1.1 Introduction

The TCG specifications for the Trusted Platform Module (TPM) and the accompanying Trusted Software Stack (TSS) are focused on PC-style platforms. When attempting to implement TCG-compatible Trusted Computing systems on mobile and embedded devices a number of issues arise. Typical embedded and mobile platforms differ to PC-platforms with respect to their booting process and BIOS. For mobile phone platforms, a single TPM on the platform might be insufficient due to ownership issues. The TCG has published a set of specifications to address the different requirements of mobile and embedded devices, here two mobile versions to the TPM are defined. These requirements differ in terms of the supported command set and ownership rules:

- Remotely Owned (MRTM)
- Locally-owned (MLTM)

1.2 Prototype Mobile Trusted Platform Design

TCG's Mobile Reference Architecture decomposes the platform into isolated trusted engines owned by different entities. Those engines have an associated MTM and interfaces to communicate with other engines. The foundation for Access Control can be provided by SELinux and enforced by the TrustZone hardware features.

- Secure-world peripherals can not be accessed by any non-secure world software.
- Non-secure world software is not able to access secure-world memory without authorisation by secure-world.

Software Components The secure world partition and its operating system kernel are in ultimate control of the whole platform. For the implementation, an adapted version of the Linux kernel has been chosen as basis for the secure world operating system. This secure world Linux kernel contains a number of TrustZone specific extensions, most notably it provides a special user-space interface, allowing regular secure world user-space processes to act as "hypervisor" for the non-secure world partition.

Secure Boot Loader A secure boot process is of ultimate importance to mobile trusted computing because it provides the basis for establishing trust on mobile and embedded devices. The existence of MRTMs implies a requirement for some kind of secure boot process. On mobile devices where hardware MTMs are available, a secure boot process can be implemented by relying on their capabilities. Platforms which only contain software MTMs cannot take advantage of having MTMs as hardware roots of trust. The modified u-boot for instance, is capable of measuring the Linux kernel image, its initial ramdisk and the kernel command line. Before control is handed over to the operating system kernel, these measurement values are compared to a Reference Integrity Metric (RIM) certificiate and attached to the kernel image. The boot process only continues if the kernel's RIM certificate can be successfully validated. The measurement values obtained by u-boot are handed over to the secure-world Linux kernel and are available after the Linux kernel hands over control to the userspace init process.

Software-based Mobile Trusted Modules The platform contains a software Mobile Local-Owner Trusted Module (MLTM) and a software Mobile Remote-Owner Trusted Module. The MLTM is started as a child process of the VM supervisor process, taking advantage of shared memory communication mechanisms. Since the supervisor process is in full control of the initial non-secure world executable image, secure boot can be implemented easily for the non-secure world partition, using the software MTMs available in secure world. Similarly to the VM supervisor process, the Trusted Engine process can take advantage of any software isolation mechanism available inside the secure-world.

Trusted Engines The TCG Mobile Reference Architecture is based on "Trusted Engines". These are described as isolated computation environments which typically have their own private MRTMs or MLTMs. Inter-engine communication is only possible by means of well-defined interfaces exposed by the individual engines. Trusted Engines are organised hierarchically with respect to their inter-engine communication interfaces.

TrustZone VM Supervisor The TrustZone VM supervisor is the fundamental component needed to support a non-secure world partition. This application utilises the TrustZone user-space interface exposed by the secure-world Linux kernel to act as hypervisor for the non-secure world. Any secure monitor calls invoked by the non-secure world guest VM are routed to the user-space VM supervisor after minimal processing inside the secure-world Linux kernel. Minimising the amount of secure-world in-kernel processing, reduces the secure-world privileged mode attack surface visible to a potential adversary coming from the non-secure world partition. To guarantee the isolation between the VM supervisor main process and its worker subprocesses, any privilege separation and access control mechanisms found inside the secure-world Linux kernel can be leveraged. This especially includes, but is not limited to, mandatory access control enforced by SELinux domains or system call usage restrictions as supported by seccomp.

1.3 Virtualisation with ARM TrustZone

The framework features implementation of supervisors for non-secure world guests as ordinary secure-world user-space processes. Interaction with the guests is accomplished by using the user-space interface exposed by the framework. Only a small set of critical hypercalls has to be implemented within the secure-world host kernel.

Secure and non-secure world partitioning Depending on the implementation of the ARM TrustZone, the partitioning of peripherals and memory between secure and non-secure world can be hard-wired in silicon or can be reconfigurable by means of special platform dependent mechanisms. Platforms with dynamic secure/non-secure world repartitioning offer a high degree of flexibility with respect to handling non-secure world guests. Depending on the current software configuration of a non-secure world guest as reported by its associated MTM, hardware resources could be made accessible to that guest selectively. An interesting application of dynamic secure/non-secure world repartitioning could be the concurrent execution of multiple strongly isolated non-secure world virtual machines.

Interrupts two special TrustZone relevant properties of IRQ- and FIQ-type interrupts can be configured by secure-world privileged executives:

- Non-secure world access permissions to the global interrupt disable bit for FIQ-type interrupts can be configured to dissalow non-secure world modifications.
- The destination for handling IRQ- and FIQ-type interrupts can be either set to the currently executing worlds' regular vectors or to special Secure Monitor Mode vectors.

the virtualisation framework is not limited to a single non-secure world executive. As a consequence, IRQ-type interrupts can be targeted towards distinct, isolated non-secure world compartments running in parallel. In order to avoid unintended cross-compartment interference among the non-secure world compartments, interrupt handling must be under total control of the secure-world kernel. A non-secure world compartment must not be able to mask or disable any interrupt sources which are allocated to other non-secure world compartments running in parallel. The compartments are provided with a virtual interrupt controller managed by the secure-world kernel. From the non-secure world software point of view, this virtual interrupt controller consists of:

- global virtual interrupt controller status flags

- per-source virtual interrupt pending and mask flags
- secure monitor calls to interact with the secure-world part of the virtual interrupt controller

The lack of automatic notification of the secure world kernel is not a problem when disabling virtual interrupts. Actually the secure world part of the virtual interrupt controller has to check the “disabled” state of a virtual interrupt, when the corresponding hardware interrupt source actually fires. If the virtual interrupt target is disabled at this time, the secure world part masks the hardware interrupt source and records the virtual interrupt for deferred delivery.

Userspace supervisor interface A minimal set of operations required to implement a secure-world userspace supervisor process for the non-secure world guest compartment is the following:

- Opening the TrustZone VM interface
- Creating a guest VM
- Configuring guest VM resources
- Making guest VM memory accessible to the supervisor process
- Switch to non-secure world and handle requests from non-secure world
- Terminating the guest VM and release its resources