Security overview

Contents

1 Article Purpose

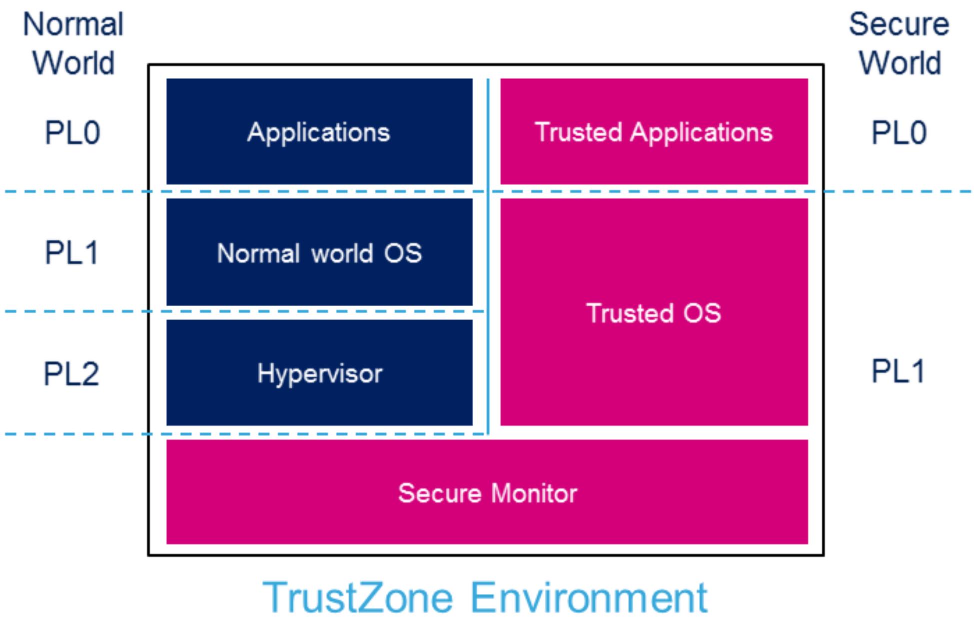
The purposes of this article is to explain how to secure an STM32 MPU-based platform thanks to several hardware mechanisms, and to briefly introduce the software components responsible for the secure configuration.

2 Introduction

The STM32 MPU is based on the Arm®Cortex-A® core, which is based on the Arm®TrustZone[1] architecture that enables context isolation: the **normal world** holds the applications whereas the **secure world** isolates all the trusted applications and core secure services so that they can safely manipulate platform secret data. The MPU includes Firewall mechanisms that allow the secure world to forbid read/write accesses from the normal world to given peripherals.

Armv7 defines PL0, PL1 and PL2 privilege levels:

- PL0 is the lowest software execution level (unprivileged calls allowed for applications).
- PL1 is the execution level for the OS.
- PL1 (secure) is also the privilege level for secure monitor execution, to switch from the secure to the normal world.
- PL2 is dedicated to the hypervisor (only non-secure).



The **normal world** is used to run rich OSs such as Linux Kernel and its applications framework.
The **secure world** runs a secure monitor with minimal services (i.e. TF-A) or a TEE as secure OS (i.e. OP-TEE OS).

The secure boot is a key feature of this multiple execution contexts environment. It allows the boot chain to be authenticated by the ROM code as well as the authentication of the components that are launched in the secure and normal worlds.

The TrustZone environment is a complete system solution that is not limited to the Cortex context. It provides a bus and peripheral infrastructure to the MPU in order to ensure that the secure world relies on a completely secured pipe when it controls a secure peripheral. The assignment of the peripherals to a given world is done thanks to a Firewall mechanism, which is set up during the secure world initialization.

Dedicated secure and normal contexts also impact the debugging facilities: depending on the targeted user, the debug can be opened to both worlds (e.g. for a secure aware developer), to normal world only (for a Linux® developer) or completely closed (for the end user). This is achieved by configuring the Debug control.

Some internal or external peripherals can be used by the secure world to support cryptographic operations. Refer to security peripherals for an introduction.

3 Secure boot

The secure boot is essential to ensure the integrity and security of the platform at runtime.

The STM32 MPU trusted boot chain was design to guarantee such a secure boot sequence.
It performs the following tasks:

- Configuration of the platform firewall, which is the foundation for a safe execution of the platform
- Configuration of the platform debug capabilities
- Verification of the integrity (thanks to a hash algorithm) and authentication (using asymmetric cryptography algorithms) of the started software components, including the Secure and non-secure worlds.

TF-A is the recommended open source bootloader. Its implementation supports the trusted boot and peripheral access control with firewall.

3.1 STM32MP1 secure boot

STM32MP1 secure boot implementation is described in the STM32MP15 secure boot article.

4 Firewall

MPU firewalls comprise access filters for MPU peripherals and subsystems memory mapped interfaces, internal RAMs/ROMs and external memory (DDR). Depending on the assignment, an MPU interface can be dedicated or shared between several hardware execution context(s).

4.1 STM32MP1 firewall

- ETZPC:
  - assigns access rights to MPU peripherals from Cortex-A7 contexts (secure or normal) and Cortex-M4 context.
  - assigns access rights to internal ROM/RAM from Cortex-A7 and Cortex-M4.
- TZC: assigns access rights to DDR regions.
- RCC: can restrict the access of some of its registers to the secure execution context.
- PWR: can restrict the access of some of its registers to the secure execution context.
- BSEC: The OTP memory can be fused to restrict the access to some of its content to the secure execution context.
- RTC: This MPU interface can restrict some of its interface registers to the secure execution context.
- GPIO: GPIO bank Z can be configured to restrict some GPIO configuration to the secure execution context.
- TAMP: can restrict the access of some of its registers to the secure execution context.
- EXTI: can restrict the access of some of its registers to the secure execution context.
- GIC: can restrict the access of some of its registers to the secure execution context.
- MDMA: can configure MDMA interrupt execution context.

5 Secure debug

The STM32 MPU offers the possibility to manage the platform debug configuration.
It is indeed possible to enable/disable independently secure and non-secure debug accesses.

5.1 STM32MP1 secure debug

Debug accesses are controlled through BSEC peripheral.
By default, the STM32 MPU is started by the ROM code with both secure and non-secure debug enabled. When the trusted boot is enabled, the ROM code disables debug accesses and relies on the FSBL to configure them.

6 Secure and non-secure worlds

Thanks to Arm®TrustZone, some portions of the executing code can be assigned to a non-secure world or to a secure world.

The secure world offers an isolated context that guarantee code and data integrity up to the hardware support. Secure world can be used to host a Trusted Execution Environment (TEE) that executes in parallel with the rich OS and provides secure services.

Firmware and more generally software executing in secure world implicitly use the TrustZone(r) NS signal/bit to be granted access to sensitive resources. These resources can be DDR locations, SoC peripheral interfaces or SoC internal resources such as clocks, debug facilities, and more.

6.1 TF-A runtime services: SP_MIN

The TF-A configuration supports the installation of a minimal runtime secure service provider and peripheral access control with firewall. This runtime firewall is built from TF-A BL32 SP_MIN image.

Run time services provided by TF-A includes not restricted to, PSCI controls, SCMI resources, some SiP & OEM SMCCC services for power states transition and other platform facilities.

6.2 OP-TEE OS runtime services

The OP-TEE is recommended by STMicroelectronics as it is an open source TEE solution. The package provides additional secure services to the platform since it can host core secure services and run trusted applications. OP-TEE provides the same level of services as TF-A listed above: PSCI, SCMI, SiP & OEM SMCCC. OP-TEE provides other high level services, such as running trusted applications and possibly generic services used by standard non-secure components such as random number generation.

7 Security frameworks

The platform non-secure world accesses to specific resources using generic frameworks. These operations are used to managed overall systems and must be implemented inside the TEE.

7.1 PSCI

PSCI[2] manages the power state of the CPU and the overall system. This framework is an Arm® generic implementation targeting CPU related power management services among which secondary CPUs boot up, dynamic addition and removal of CPUs, CPU idle management and system shutdown and reset.

7.1.1 STM32MP1 PSCI support

Both TF-A and OP-TEE implement PSCI v1.0 [3] and offers standard services for:

- Core idle management
- Boot up for secondary boot core
- Dynamic addition and removal of cores
- System shutdown and reset

7.2 SCMI

SCMI[4] specification defines a standard to access resources for power, performance and system management. This framework is an ARM® generic implementation.
SCMI protocols can be used to create device interfaces for external resource that are controlled by a SCMI server implementation. Targeted devices are of, not limited to:

- Power domain management
- Performance management
- Clock management
- Sensor management
- Reset controller management

7.2.1 STM32MP1 SCMI support

See the specific article on SCMI overview. It presents SCMI used by secure world (TF-A, OP-TEE) to expose system resources, as input source clocks and PLL output clocks, reset controllers to non-secure world (U-Boot bootloader, Linux kernel).

7.3 Platform SiP and OEM SMCCC services

7.3.1 STM32MP1

STM32MP1 embeds SiP and OEM SMCCC services for non-secure world to access low power transition facilities, clock calibration facilities, BSEC OTP access, possibly also some test facilities. Both TF-A and OP-TEE implement the same level of services aside possible test facilities.

8 Security peripherals

8.1 Cryptographic hardware acceleration

The STM32 MPU embeds multiple peripherals for cryptographic acceleration:

- CRYP
- HASH

8.2 Trusted platform module (TPM)

The STM32 MPU can be associated to an external trusted platform module (TPM).
It provides secret data storage capabilities as well as cryptographic capabilities allowing to use them.

8.3 Tamper event detection

The STM32 MPU embeds internal and external tampers detections mechanisms. They can be used to detect physical attack, out-of-spec voltage, etc. A tamper detection will result in the automatic clearing of sensitive data in the system : it is a key feature for security product !

8.3.1 STM32MP1

The tamper configuration is described in STM32MP15 Tamper configuration.

9 Secure Secret Provisioning

The STM32 MPU allows to provision secrets in BSEC fuses in a secure way. This is the Secure Secret Provisioning (SSP) feature.

This feature is a secure process which runs between the software programmer (including a HSM), the ROM Code and a customized TF-A BL2. The security relies on the chip attestation and a package encryption exchange between the programmer and the STM32 MPU.

This feature is fully describes in the AN5510: Overview of the secure secret provisioning (SSP) on STM32MP1 Series.

9.1 Fuses provisioning in STM32MP1

The build process for the TF-A SSP firmware is described here.

10 References

- https://www.arm.com/why-arm/technologies/trustzone-for-cortex-a
- https://developer.arm.com/architectures/system-architectures/software-standards/psci
- https://developer.arm.com/docs/den0022/latest
- https://developer.arm.com/architectures/system-architectures/software-standards/scmi

Microprocessor Unit

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

TrustZone®

Arm® and TrustZone® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Operating System

Trusted Execution Environment

Read Only Memory

Linux® is a registered trademark of Linus Torvalds.

Doubledata rate (memory domain)

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

One Time Programmed

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

First Stage Boot Loader

Trusted Firmware for Arm Cortex-A

Boot Loader stage 3-2

Power State Coordination Interface

System control and management interface

Silicon Provider

Original Equipment Manufacturer

secure monitor call (SMC) calling convention

Open Portable Trusted Execution Environment

Central processing unit

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Boot and Security and OTP control

Trusted Platform Module

Secure Secret Provisioning

Secure secrets provisioning

Hardware security module

Boot Loader stage 2