2023-04-09

Created: 2023-04-26 23:54:14

Questions

- Ch 2 - Introduction `(230409.FF-A.txt)`
  - `[09:00]` The below text are chapter 2 of “Arm Firmware Framework for Arm A-profile” `(FF-A.0409.md)`
- Ch 3 - Software Architecture `(230409.FF-A.txt)`
  - `[09:09]` The below text are chapter 3 of “Arm Firmware Framework for Arm A-profile” `(FF-A.0409.md)`
- Ch 3.1 - Isolation boundaries `(230409.FF-A.txt)`
  - `[09:17]` The below text are chapter 3.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`
- Ch 3.2 - Partitions `(230409.FF-A.txt)`
  - `[09:21]` The below text are chapter 3.2 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`
- Ch 3.3 - Partition manager `(230409.FF-A.txt)`
  - `[09:45]` The below text are chapter 3.3 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`
  - `[11:15]` So SPM and Hypervisor are partition managers. Right? `(FF-A.0409.md)`
  - `[11:15]` What is VMSA? `(FF-A.0409.md)`
  - `[11:16]` Is VMSA a software or a hardware? `(FF-A.0409.md)`
- Ch 3.4 Example configurations `(230409.FF-A.txt)`
  - `[09:52]` The below text are chapter 3.4 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`
  - `[10:14]` Can SPM see SP0 and SP1? `(FF-A.0409.md)`
  - `[10:15]` Can SP0 or SP1 see SPM? `(FF-A.0409.md)`
  - `[10:17]` Is TrustZone isolation boundary physical isolation or logical isolation? `(FF-A.0409.md)`
- Ch 4.1 - SPM architecture `(230409.FF-A.txt)`
  - `[10:26]` The below text are chapter 4.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`
  - `[10:48]` What is SPMC in the document? `(FF-A.0409.md)`
  - `[10:48]` Is SPMC of TrustZone? `(FF-A.0409.md)`
- Hypervisor & TrustZone `(230407.Arm.Hypervisor.txt)`
  - `[10:32]` Does TrustZone require hypervisor? `(OP-TEE.0327.md)`
- TCB `(230407.Arm.Hypervisor.txt)`
  - `[11:21]` What are components in ARM TCB? `(TrustZone.0329.md)`
- SM & SPM & SPD `(230329.TrustZone.txt)`
  - `[18:24]` Does TrustZone require SPM? `(TrustZone.0329.md)`
- PKCS#11 `(230327.OP-TEE.txt)`
  - `[20:40]` The following text are from https://optee.readthedocs.io/en/latest/architecture/porting_guidelines.html `(OP-TEE.0327.md)`
  - `[20:42]` The following continued text are from https://optee.readthedocs.io/en/latest/architecture/porting_guidelines.html `(OP-TEE.0327.md)`
  - `[20:44]` How does OP-TEE work with HUK? `(OP-TEE.0327.md)`
  - `[21:07]` What are common device that store HUK in ARM platform with OP-TEE? `(OP-TEE.0327.md)`

Q & A

Ch 2 - Introduction `(230409.FF-A.txt)`

Question: `[09:00]` The below text are chapter 2 of “Arm Firmware Framework for Arm A-profile” `(FF-A.0409.md)`

Answer: … …

Ch 3 - Software Architecture `(230409.FF-A.txt)`

Question: `[09:09]` The below text are chapter 3 of “Arm Firmware Framework for Arm A-profile” `(FF-A.0409.md)`

Answer: … …

Ch 3.1 - Isolation boundaries `(230409.FF-A.txt)`

Question: `[09:17]` The below text are chapter 3.1 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`

Answer: … …

Ch 3.2 - Partitions `(230409.FF-A.txt)`

Question: `[09:21]` The below text are chapter 3.2 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`

Answer: … …

Ch 3.3 - Partition manager `(230409.FF-A.txt)`

Question: `[09:45]` The below text are chapter 3.3 of “Arm Firmware Framework for Arm A-profile”. Please summarize it. `(FF-A.0409.md)`

Answer: … …

Question: `[11:15]` So SPM and Hypervisor are partition managers. Right? `(FF-A.0409.md)`

Answer:

Yes, that’s correct. In the context of the Arm Firmware Framework for Arm A-profile, the Secure Partition Manager (SPM) operates in the Secure world, while the Hypervisor operates in the Normal world. Both SPM and Hypervisor are partition managers responsible for creating and managing the physical isolation boundaries of partitions, assigning resources, and enabling inter-partition communication.

Question: `[11:15]` What is VMSA? `(FF-A.0409.md)`

Answer:

VMSA, or Virtual Memory System Architecture, is a part of the Arm architecture that defines the memory management and address translation mechanisms. It is responsible for managing the address spaces and memory protection features in Arm systems. VMSA allows for the creation of separate virtual address spaces for each process, providing isolation between processes and the operating system. In the context of the Arm A-profile, the Hypervisor uses the virtualization extension in VMSA to create physical isolation boundaries between different partitions or virtual machines.

Question: `[11:16]` Is VMSA a software or a hardware? `(FF-A.0409.md)`

Answer:

VMSA (Virtual Memory System Architecture) is not a specific piece of software or hardware, but rather a part of the Arm architecture specification that defines how memory management and address translation should be implemented in Arm-based systems. VMSA provides guidelines for designing hardware and software components that handle memory management, address translation, and access control, allowing for the creation of separate virtual address spaces and ensuring proper isolation between processes and the operating system.

Ch 3.4 Example configurations `(230409.FF-A.txt)`

Question: `[09:52]` The below text are chapter 3.4 of "Arm Firmware Framework for Arm A-profile". Please summarize it. `(FF-A.0409.md)`

Answer: … …

Question: `[10:14]` Can SPM see SP0 and SP1? `(FF-A.0409.md)`

Answer:

Yes, the Secure Partition Manager (SPM) can "see" and manage both SP0 and SP1. The SPM is responsible for creating and managing the physical isolation boundaries of Secure Partitions (SPs) like SP0 and SP1. It uses partition manifests to assign resources, initialize the partitions, and enable discovery of their services. Additionally, the SPM implements FF-A ABIs (Application Binary Interfaces) to enable inter-partition communication for access to partition services. So, while the SPM isolates SP0 and SP1 from each other, it still has visibility and control over both partitions.

Question: `[10:15]` Can SP0 or SP1 see SPM? `(FF-A.0409.md)`

Answer:

In the ARM Firmware Framework, SP0 and SP1 can communicate with the Secure Partition Manager (SPM) through well-defined interfaces, such as FF-A ABIs (Application Binary Interfaces). While the SPM is physically isolated from SP0 and SP1 in scenarios where they are physical partitions, they can still interact with the SPM using these interfaces for communication, service discovery, and memory management.

However, it is important to note that SP0 and SP1 cannot access the internal workings or the memory of the SPM directly. They can only communicate with the SPM through the defined interfaces, ensuring that the isolation between them is maintained.

Question: `[10:17]` Is TrustZone isolation boundary physical isolation or logical isolation? `(FF-A.0409.md)`

Answer:

TrustZone isolation boundary is a form of physical isolation. TrustZone is a security extension in ARM architecture that separates the processor's resources, such as memory and peripherals, into two distinct security domains: the Secure world and the Non-secure world. By utilizing hardware-based isolation, TrustZone ensures that sensitive data and functions in the Secure world are protected from unauthorized access by software running in the Non-secure world.

Ch 4.1 - SPM architecture `(230409.FF-A.txt)`

Question: `[10:26]` The below text are chapter 4.1 of "Arm Firmware Framework for Arm A-profile". Please summarize it. `(FF-A.0409.md)`

Answer: … …

Question: `[10:48]` What is SPMC in the document? `(FF-A.0409.md)`

Answer:

In the document “Arm Firmware Framework for Arm A-profile,” the SPMC (Secure Partition Manager Core) is a component of the Secure Partition Manager (SPM) responsible for:

1. Initializing and isolating Secure Partitions (SPs) at boot time.
2. Ensuring inter-partition isolation during runtime.
3. Managing inter-partition communication between Secure Endpoints (S-Endpoints) and Non-Secure Endpoints (NS-Endpoints).

The SPMC can co-reside with the SPM Dispatcher (SPMD) in EL3 or reside in adjacent exception levels, such as S-EL1 or S-EL2. It is a part of the Trusted Computing Base and has access to the entire physical address space.

Question: `[10:48]` Is SPMC of TrustZone? `(FF-A.0409.md)`

Answer:

The SPMC (Secure Partition Manager Core) is part of the Arm Firmware Framework for Arm A-profile, which is built on top of the Arm TrustZone technology. TrustZone provides the underlying hardware support for isolation and security, while the SPMC is responsible for managing partitions and their communication within the Secure world enabled by TrustZone. So, while the SPMC is not TrustZone itself, it is a component that utilizes and operates within the TrustZone technology.

Hypervisor & TrustZone `(230407.Arm.Hypervisor.txt)`

Question: `[10:32]` Does TrustZone require hypervisor? `(OP-TEE.0327.md)`

Answer:

No, TrustZone does not require a hypervisor. TrustZone is a hardware-based security technology implemented in Arm processors that provides a secure execution environment by creating two separate worlds: the Secure World and the Normal World. TrustZone itself is not dependent on a hypervisor.

A hypervisor is a software layer that manages and coordinates multiple virtual machines (VMs) running on a single hardware platform. Hypervisors can be used in conjunction with TrustZone to provide additional isolation and management capabilities for virtualized systems.

While TrustZone and hypervisors can work together, they serve different purposes and are not dependent on one another. TrustZone focuses on providing a secure environment for trusted applications, while hypervisors manage VMs and their resources.

TCB `(230407.Arm.Hypervisor.txt)`

Question: `[11:21]` What are components in ARM TCB? `(TrustZone.0329.md)`

Answer:

The Trusted Computing Base (TCB) of an ARM system consists of the essential components responsible for maintaining security properties and enforcing security policies. In the context of ARM TrustZone and related technologies, the TCB can include the following components:

1. Secure Boot: This is the mechanism that ensures the integrity and authenticity of the boot process, making sure that only trusted software components are executed during the system startup.

2. ARM TrustZone: A hardware-based security feature in ARM processors, which provides a secure execution environment (Secure World) and isolates it from the less secure environment (Normal World).

3. Secure Monitor: This is the lowest-level software component running in the Secure World. It manages transitions between the Secure and Normal Worlds and provides an interface for Secure World services.

4. Trusted Firmware-A (TF-A) or other trusted firmware: This firmware runs in the Secure World and provides various services, such as secure boot, power management, and platform-specific configurations.

5. OP-TEE or other Trusted Execution Environment (TEE) implementations: TEEs, like OP-TEE, provide a secure environment for running trusted applications (TAs) within the Secure World, ensuring that sensitive code and data are protected from unauthorized access or tampering.
6. Secure Partition Manager (SPM): This component manages Secure Partitions (SPs) within the TrustZone Secure World, including creating, configuring, and managing SPs, as well as inter-partition communication and resource management.
7. Trusted Applications (TAs): These are secure software components that run within the Secure World and provide security-sensitive services or perform security-critical operations.
8. Security-sensitive hardware components: These components, such as cryptographic accelerators, secure storage, and hardware random number generators, provide additional security features and services to the system.
9. Secure peripherals: These are peripherals that can be accessed only from the Secure World, providing additional isolation and protection for sensitive data and operations.

Please note that the specific components in an ARM TCB may vary depending on the system's architecture, SoC, and security requirements. The list above provides a general overview of the typical components in an ARM TrustZone-based TCB.

SM & SPM & SPD `(230329.TrustZone.txt)`

Question: `[18:24]` Does TrustZone require SPM? `(TrustZone.0329.md)`

Answer:

TrustZone itself does not inherently require a Secure Partition Manager (SPM). TrustZone is a hardware-based security feature in ARM processors that provides a secure execution environment (Secure World) and isolates it from the less secure environment (Normal World).

However, when using ARMv8.4-A architecture or later, which introduces the concept of Secure Partitions (SPs) to improve isolation and modularity of the Secure World components, an SPM becomes an important part of the system. The SPM is responsible for managing the lifecycle of Secure Partitions and handling inter-partition communication and resource management.

So, while TrustZone doesn't require an SPM in its basic form, utilizing SPM can enhance the security and modularity of the system when working with Secure Partitions in newer ARM architectures.

PKCS#11 `(230327.OP-TEE.txt)`

Question: `[20:40]` The following text are from https://optee.readthedocs.io/en/latest/architecture/porting_guidelines.html `(OP-TEE.0327.md)`

Answer: … …

Question: `[20:42]` The following continued text are from https://optee.readthedocs.io/en/latest/architecture/porting_guidelines.html `(OP-TEE.0327.md)`

Answer: … …

Question: `[20:44]` How does OP-TEE work with HUK? `(OP-TEE.0327.md)`

Answer:

OP-TEE (Open Portable Trusted Execution Environment) is designed to provide a secure environment for the execution of trusted applications on a device. One of the critical components for providing security within OP-TEE is the use of a Hardware Unique Key (HUK).

A Hardware Unique Key is a device-specific secret key that is usually derived from or securely stored in hardware. HUKs are essential for providing a secure root of trust, as they can be used to derive other cryptographic keys and establish secure communication channels between different components of a system.

Here's how OP-TEE works with HUK:

1. In OP-TEE, the HUK is used to derive other keys for various purposes, such as secure storage or cryptographic operations. The derived keys are typically used in combination with other keys or cryptographic primitives to ensure the confidentiality, integrity, and authenticity of the data being processed.

2. The function tee_otp_get_hw_unique_key(…) in core/include/kernel/tee_common_otp.h is responsible for retrieving the HUK from the hardware. In a real secure product, this function must be replaced with an implementation that retrieves the HUK from the device's hardware. This ensures that the HUK remains protected and cannot be accessed directly from software, even on the secure side.

3. If the device does not have hardware support for a HUK, it is crucial to change the implementation of the tee_otp_get_hw_unique_key(…) function to use something other than zeroes. Storing a key in software is not a secure practice, especially for a key that serves as the root for all other keys. As such, it is not recommended, but it may be necessary if hardware support is lacking.

4. Once the HUK is securely retrieved, it can be used by OP-TEE to derive other keys or perform cryptographic operations necessary for securing trusted applications and their data. The HUK helps establish a root of trust that can be extended to other parts of the system.

In summary, OP-TEE works with HUKs by leveraging them to derive other cryptographic keys and establish a root of trust for secure operations. The HUK is typically retrieved from the hardware using a secure function, and it is essential for providing a strong foundation for the security of the entire system.

Question: `[21:07]` What are common device that store HUK in ARM platform with OP-TEE?`(OP–TEE.0327.md)`

Answer:

Many devices in the ARM platform, especially those utilizing TrustZone technology, store Hardware Unique Keys (HUKs) for providing a secure root of trust. Some common devices that store HUKs and are compatible with OP-TEE include:

1. NXP i.MX family: Devices in the i.MX family, like i.MX6, i.MX7, and i.MX8, use ARM TrustZone and secure boot capabilities. They have integrated security features, such as cryptographic accelerators and secure storage, where the HUK can be stored.

2. Xilinx Zynq UltraScale+ MPSoC: The Zynq UltraScale+ MPSoC has ARM Cortex-A53 and Cortex-R5 cores with TrustZone technology. It provides secure boot and supports storage of HUK in the on-chip eFUSE memory.

3. STMicroelectronics STM32MP1: The STM32MP1 series, with its ARM Cortex-A7 and Cortex-M4 cores, supports OP-TEE and TrustZone. It provides secure storage for HUKs using its OTP (One-Time Programmable) memory or other secure storage mechanisms.

4. HiSilicon HiKey and HiKey 960: These development boards, based on HiSilicon's Kirin SoCs, feature ARM Cortex-A cores with TrustZone technology. They can be used with OP-TEE and support secure storage for HUKs.

5. Raspberry Pi 3 (unofficially): While not officially supported by OP-TEE, Raspberry Pi 3, featuring a Broadcom SoC with ARM Cortex-A53 cores, has been known to run OP-TEE. However, the secure storage of HUKs on Raspberry Pi 3 may not be as robust as in other devices designed with hardware security features in mind.

These are just a few examples of devices that store HUKs and can work with OP-TEE on the ARM platform. It's important to note that the secure storage and management of HUKs may vary depending on the specific device, its hardware features, and the implementation of OP-TEE. Always consult the device's documentation and follow the manufacturer's guidelines to ensure proper security measures are in place.
