FF-A compliant Secure User Mode partition

Secure User Mode Partition with Partition Manager at EL3.

Sayanta Pattanayak
Aditya Angadi
Objective

- Out-of-box secure software solution for ARM platforms that
  - do not support S-EL2 or do not deploy secure hypervisor at S-EL2.
  - do not deploy a secure OS at S-EL1.
- Defines a secure software solution following Firmware Framework Architecture for Armv8-A processors (FF-A).
● SPM-MM module in Trusted firmware is Secure partition manager. It manages the context of secure partition and communication between normal world and secure world.
● StandaloneMM is secure partition.
● Normal world firmware and secure partition communicate using SPM_MM SMC Ids.
Firmware Framework For ARMv8-A (FF-A)

- Standardize communication among normal world and secure world images.
- Partition manager, manages resources for all partitions and ensures inter partition isolation at run time.
- Partition manifest, captures configuration details of partitions.
- Extensive application binary interfaces for discovery of partitions, direct/indirect message communications, memory management operations, CPU cycle management operations.
Firmware Framework For ARMv8-A (FF-A)

- FF-A model

Secure world

- FF-A spec. splits the Secure Partition Manager (SPM) into two components Dispatcher (SPMD) and Core (SPMC).
- SPMC component initializes partitions, manages all partition resources and ensures isolation amongst them.
- FF-A spec. allows SPMC to be implemented at any of the exception level: EL3, S-EL2, S-EL1, and inclusion of SPMC to the appropriate exception level is done based on the platform architecture.
Updated following modules with FF-A support:

- SPMD at EL3 Trusted Firmware
- EDK2 MM_Communicate
- StandaloneMM

Introduced module:

- SPMC at EL3

For better code reuse reorganized existing spm-mm module and use some part of common code.
S-EL0 Partition with SPMC+SPMD at EL3

- MM_Communicate allows the normal world components to communicate with secure world via spec defined SMCs.
- StandaloneMM, loaded by EL3 firmware, executes as Secure Partition in S-EL0 on ARM AArch64 platforms.
- Changes, in StandaloneMM and MM_Communicate for FF-A support, are of similar type.
  - FF-A support for handling direct message communication, SPM version checking, error responses.
  - Build time option for user to choose either existing MM interface or FF-A interface for communication between normal world and secure world images.
S-EL0 Partition with SPMC+SPMD at EL3

- SPMD module is the gateway for handling any FF-A SMC from normal and secure world partitions and forwarding it to the SPMC.
- Since SPMC co-exist in EL3, SPMD doesn’t need to handle any exception level change for SPMC, but just a function call.
- SPMC initializes and manages secure partition.
- SPMC gathers memory region details, prepares memory mapping, translation table and execution context for S-EL0 partition.
- During handling of FF-A requests, SPMC does the job of save/restore execution context and jump to respective exception level.
SPMD, SPMC, & Secure Partition Setup

The diagram illustrates the setup process for Secure Partitioning in a multi-core system, focusing on SPMD, SPMC, and secure partition configurations. It outlines the steps and functions involved in initializing and setting up the secure partition environment, including boot arguments, memory management, and system initialization.

Key components of the diagram include:

- **SPMD**: A function that initializes SPMC context arrays for all secondary cores.
- **SPMC**: A function that sets up secure partition context.
- **EL3**: Represents the third execution level in a secure partitioning system.
- **SP**: A function that initializes secure partition context.
- **SPM**: A function that sets up secure partition memory.
- **Setup SP**: A function that sets up system registers for S-EL1 and S-EL0.
- **Secure partition init**: A function that initializes SP context to RESET.
- **Spmc_sp_synchronous_entry**: A function that takes Secure Partition CPU context pointer and performs a synchronous entry into the secure partition.
- **Spmc_secure_partition_enter**: A function that enters secure payload through EL3 exit.
- **EL3 exit**: The exit point for the secure partition.

The diagram also highlights the interaction between SHIM (Secure Hardware Interface Module) and the standalone MM (Secure Memory Management) secure partition.

This setup process is critical for ensuring secure and efficient execution in multi-core systems.
SPMD, SPMC, & Secure Partition Setup Continued...

From boot arguments prepare boot structure. As part of PeCoff image loading, there will be FF-A requests for change of memory region attributes.
UEFI Secure boot

**StandAloneMM (User secure partition)**
- Initialize Secure storage (NOR flash)
- Receives FF-A direct message request for secure key information.
- Access secure NOR Flash region for availing secure key.
- Copy secure key information into the FF-A payload buffer, which is shared between Secure Partition and Non-secure user.
- Responds successful return with FF-A direct response ABI.

**Trusted Firmware, SPMD + SPMC**
- SPMD module receives FFA direct message ABI from non-secure world, forwards the request to SPMC.
- SPMC module saves normal world context, prepares the context for secure partition, populate the FF-A direct request payload and does secure partition entry by changing exception level.
- SPMD receives FF-A direct response request from Secure partition, forwards to SPMC, which restores the normal world context and switches back to normal world.

**EDK2**
- Enable secure booting.
- Request for secure keys through MM_Communicate, using FF-A direct messaging ABIs.
Current status

- StandaloneMM FF-A interface support has been added by upstream community and merged into EDK2.
- FF-A support is added in MM_Communicate for interaction from Normal world.
- In TrustedFirmware changes are made in SPMD, SPM_MM to accommodate new FF-A module SPMC.
- At present only direct messaging ABIs are implemented for communication between normal world and secure world partitions.
- UEFI secure boot is one key user scenario, which is covered with current implementation.
- Next immediate objective is to get the solutions merged into respective codebases.
Thank you

Accelerating deployment in the Arm Ecosystem