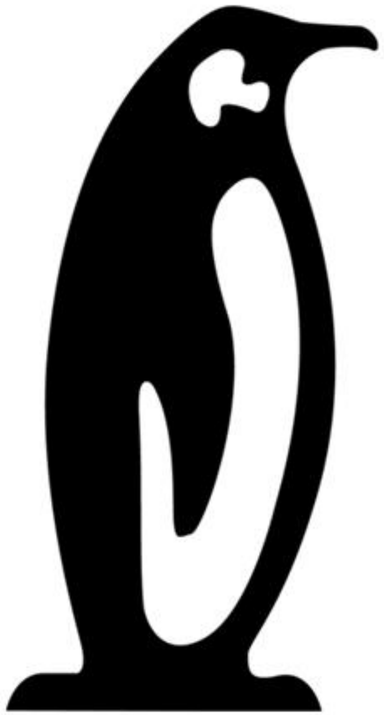


Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

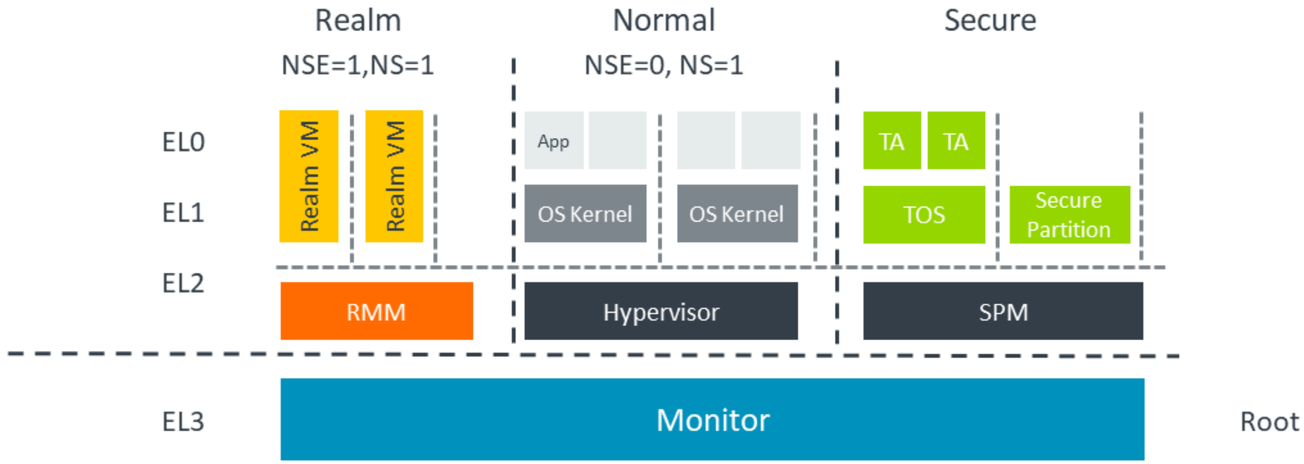
Interplane Communication on Arm CCA

Derek D. Miller 20 September 2024

LINUX PLUMBERS CONFERENCE | Vienna, Austria
Sept. 18-20, 2024

Arm CCA

- Full software stack Isolation from the host OS and hypervisor
- Hardware-based security mechanisms
- Blind hypervisor – Realm can contain a full OS/firmware stack
- Support for attestation
- Device Assignment can be used to extend the TCB to additional resources
  - PCIe devices
  - On SoC devices



CCA Planes

Multiple EL0+EL1 environments within a realm

P0 – privileged EL0+EL1 environment

- Control which plane can run next
- Read and modify register state of other planes in the realm
- Control memory access permissions of other planes in the realm
- Configure how interrupts are delivered to other planes in the realm

Pn – all of the others

N = number of planes

- 1 < N <= MAX_PLANES
- MAX_PLANES in an architectural constant

N is specified during realm creation and fixed for the lifetime of the realm

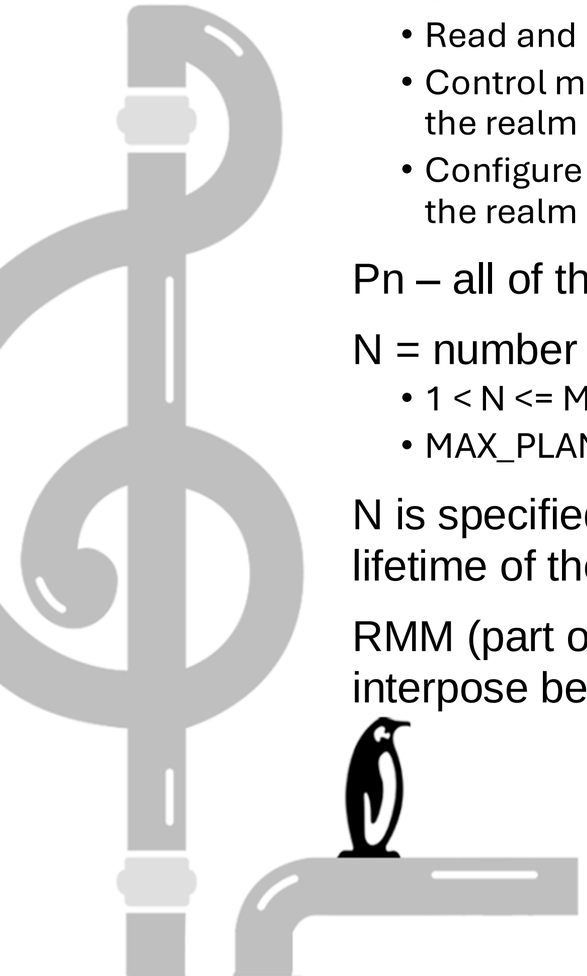
RMM (part of the TCB) is the only software that can interpose between planes

- **SMC exceptions in any Pn can be routed by the RMM to P0**

Intended Usage Model:

Provide a single “guest OS” plane with one or more “service” planes

Services can be anything, but the first use-case is likely to be virtual TPMs



Inter-VMPL Communication on AMD SEV-SNP: SVSM

- Defines a binary inter-VMPL transport mechanism
- Provides a Core Protocol
  - with many AMD-specific details
- Provides a discovery interface
- Provides an attestation protocol
- Provides mechanisms for defining new services

Allows workloads to be able to interface with any compliant implementation in VMPL0

Arm’s Firmware Framework for A-Profile(FF-A)

- Designed to standardize communication between the non-secure world and services running inside secure partitions
- Generalizes interaction between non-secure software and privileged firmware in the secure state
- But can also work for interplane communication
- Provides higher-level abstractions for communication
- Uses Arm’s Secure Monitor Call Calling Convention (SMCCC)
- Has a discovery mechanism

So why not SVSM for Arm?

- ~~Defines a binary inter-VMPL transport mechanism~~
- ~~Provides a Core Protocol~~
  - ~~with many AMD-specific details~~
- ~~Provides a discovery interface~~
- Provides an attestation protocol
- ~~Provides mechanisms for defining new services~~

We've already got most of what it offers, but there are still gaps

LIME

Layered Inter-plane Messaging Entrypoints

- This backronym is still being workshopped

Needs an attestation protocol

May need a discovery protocol

- In addition to the SMC/FF-A discovery protocols?
- So Pn users can be assured a service they are using is in-realm and not provided by the platform
  - But they **could** use attestation for this and discard the signature

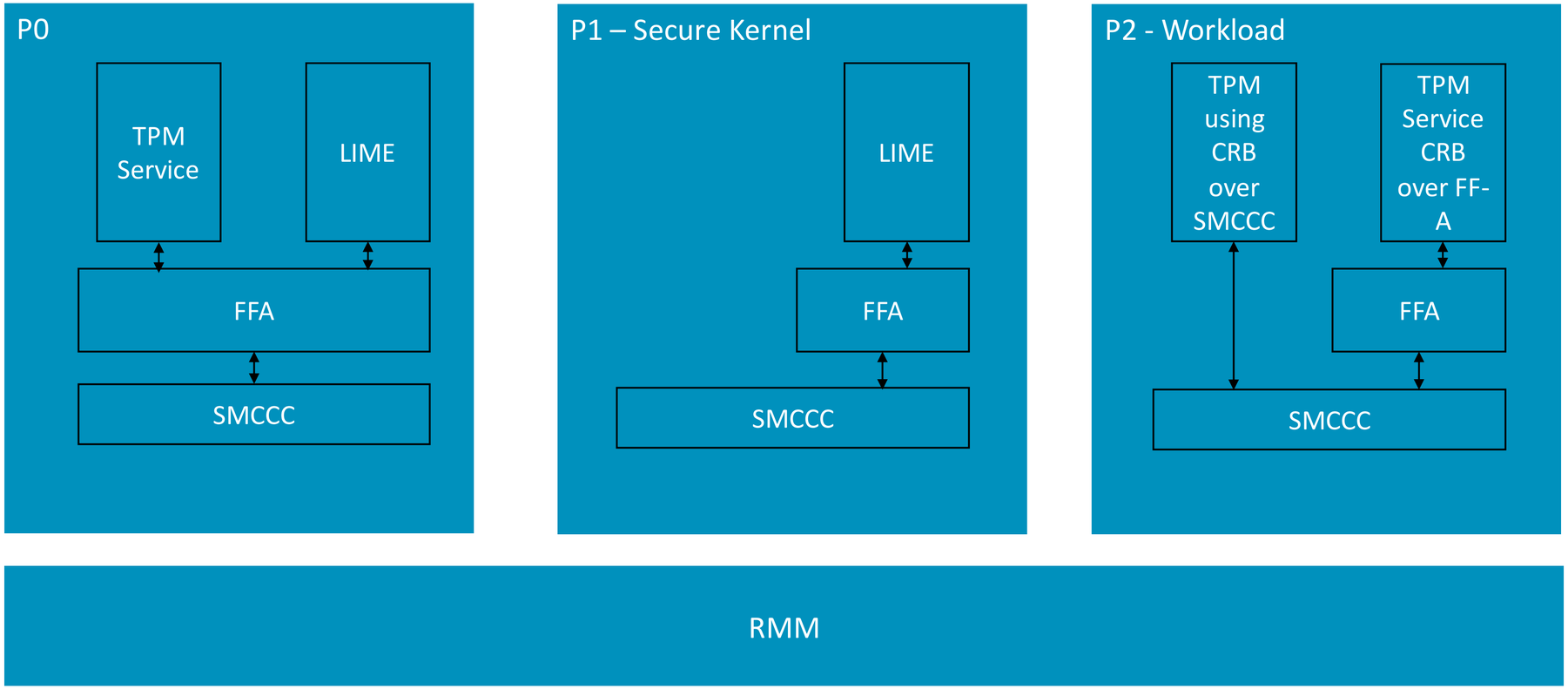
Utilizes FF-A

- Or should it use SMCCC?
- It really is just another SMCCC/FF-A Service

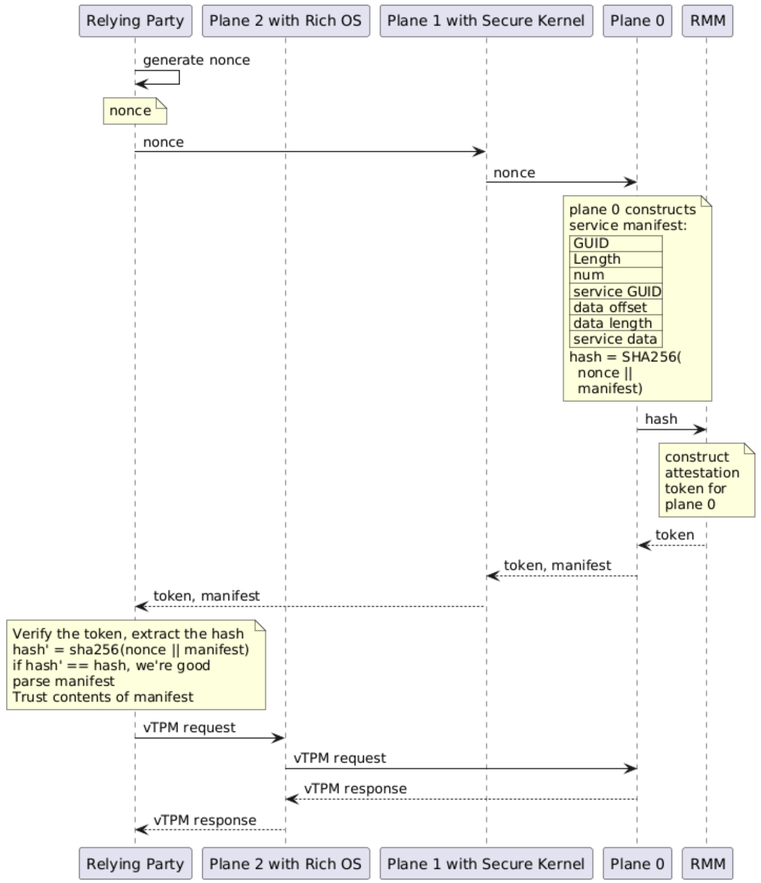
Can be used for services that use either FF-A or SMCCC

Backup

Protocol Stack



Attestation Protocol



- Provides a mechanism for a relying party (external to the platform) to establish trust in a p0 service
- Request for attestation can come from the Rich OS (plane 1 in the diagram) or a secure kernel in a different plane (plane 2 in the diagram)
  - Sequence diagram shows the latter
- Avoiding the eyechart: If you're familiar with SVSM's attestation – this looks the same

Lift-and-Shift?

- How does that work with the added attestation?
  - Obviously, existing workloads will not be doing this
- It depends on how things are set up, but if the Secure Kernel (plane 1 in the previous diagram) handles the attestation steps (on behalf of the relying party), few/no changes to the workload (Plane 2 in the sequence diagram) software stack are necessary

LIME_ATTEST_SERVICES fields

Input Fields

Field Use	Size(bytes)	Location
LIME FID	4	W0
Attestation Service GUID	16	X1, X2
LIME_ATTEST_SERVICES_OPCODE	8	X3
Attestation report buffer IPA	8	X4
Attestation report buffer size (in bytes)	8	X5
Nonce buffer IPA	8	X6
Nonce size (in bytes)	8	X7
Services manifest buffer IPA	8	X8
Services manifest buffer size (in bytes)	8	X9

Output Fields

Field Use	Size(bytes)	Location
Result Value	8	X0
Attestation report size (in bytes)	8	X1
Services manifest size (in bytes)	8	X2

- All guest IPAs provided must only be in the address space of the Pn making the call (P0 needs to check this)
- P0 will assemble the services manifest as input to its attestation request to the RMM

LIME_ATTEST_SERVICES – Services Manifest

Byte Offset	Size(bytes)	Meaning
0x00	16	Services manifest GUID
0x10	4	Services manifest length (in bytes)
0x14	4	Number of services described in the manifest
First service table entry, if any		
0x18	16	Service GUID
0x28	4	Service data offset
0x2c	4	Service data length
Next service table entry, if any. Additional service table entries follow, followed by the service data		
0x30		

- Each service will document its GUID value and the format of it service data
- The input `cca-realm-challenge` field of the `cca-realm-claim-map` supplied by P0 for the CCA Attestation request will be:
  - `SHA-512(nonce || services manifest)`

Example FF-A vTPM Service data

Byte Offset	Size(bytes)	Meaning
0x00	16	Partition ID
0x10	4	Version
0x14	4	Size of TPMT_PUBLIC structure (in bytes) K
0x18	K	TPMT_PUBLIC structure of the endorsement key

Example SMCCC vTPM Service data

Byte Offset	Size(bytes)	Meaning
0x00	4	Version
0x04	4	Number of Opcodes used by service (N)
0x08	4	Size of TPMT_PUBLIC structure (in bytes) K
First opcode entry		
0x08	4	Opcode 0
. . .		
0x08+4*(N-1)	4	Opcode N-1
0x08+ 4 *N	K	TPMT_PUBLIC structure of the endorsement key

Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024