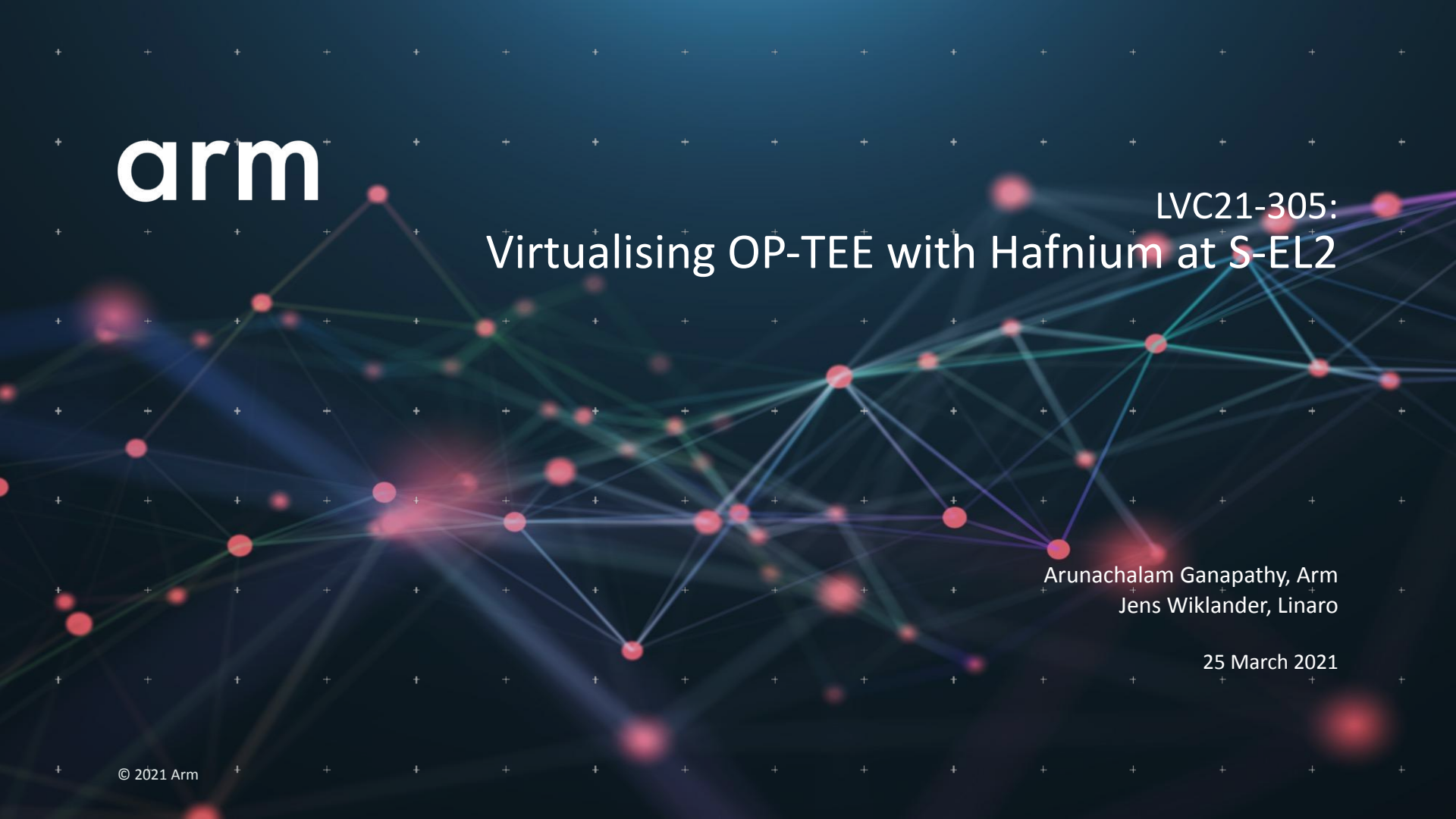


LVC21-305:
Virtualising OP-TEE with Hafnium at S-EL2

Arunachalam Ganapathy, Arm
Jens Wiklander, Linaro

25 March 2021

Agenda

- Arm SEL2 feature, SPM and FF-A
- OP-TEE SEL1 SP on Hafnium SEL2 SPMC
  - EL3 SPMD
  - Arm FF-A driver
  - OP-TEE driver FF-A support
  - OP-TEE OS: SEL2 SPMC and FF-A support
- Demo: FVP

Arm SEL2 feature, SPM and FF-A

OP-TEE SEL1 SP on Hafnium SPMC at SEL2

Demo: FVP

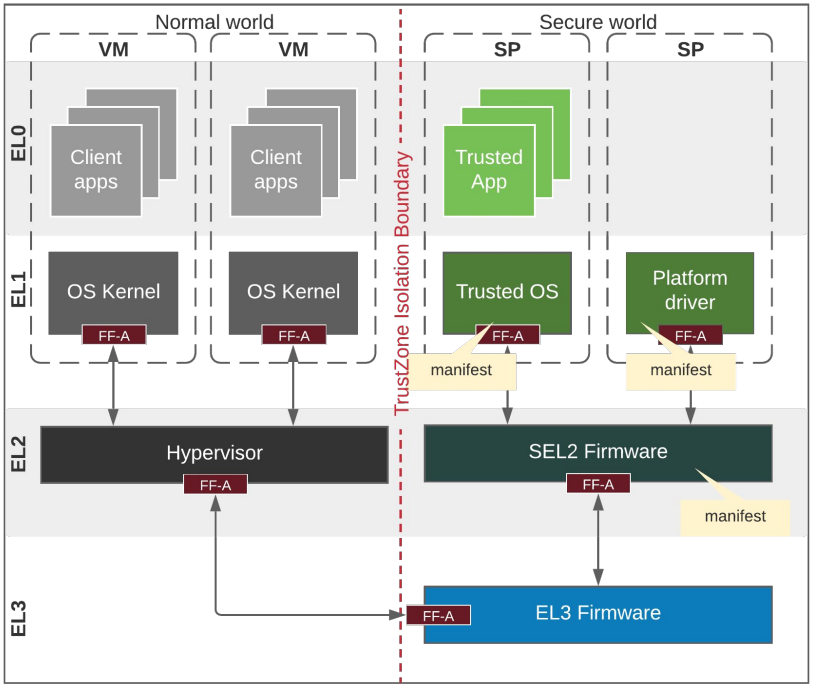
Virtualization in Secure World

Armv8.4 introduces virtualization in secure world

- New exception level S-EL2
- Key features: Stage-2 translation
  - VA → IPA (Guest VM)
  - Second translation: IPA → PA
- Isolates EL3 software from S-EL1
- Provides hardware isolation between multiple S-EL1 software components.
- Removes TOS dispatcher from EL3

SEL2 Firmware:

- Hafnium as SPM
- Meets most of the requirements for S-EL2 software architecture
- Many of the SPM requirements are common with Hafnium design goals S-EL2
- Minimal code size that reduces attack surface
- Uses FF-A



Firmware Framework – A profile (FF-A)

FF-A

- Developed by Arm, contributions from Google, SiPs and Trusted OS vendors
- Prior known as SPCI (Secure Partition Client Interface)
- Leverages new architecture (S-EL2) and system features (SMMU, vGIC)

Terminologies:

SPMD: Secure Partition Manager Dispatcher

SPMC: Secure Partition Manager Core

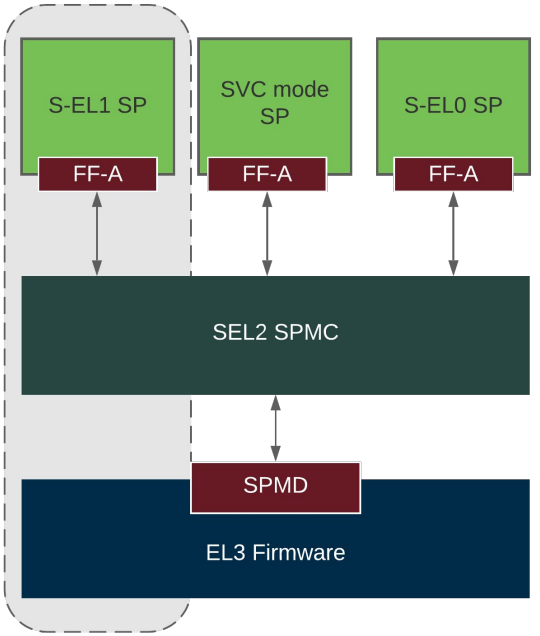
SP – Secure Partition (mostly encompasses S-EL0 and S-EL1)

S-Endpoint: SP (Spec supports various modes)

NS-Endpoint: VM in normal world or OS

Extends SMCCC to transport FFA messages between normal ↔ secure world

- FFA ABIs: Discovery, Message Passing, Memory Sharing



Arm SEL2 feature, SPM and FF-A

OP-TEE SEL1 SP as Hafnium SPMC at SEL2

Demo: FVP

EL3 SPD from opteed to spmd

SPD=spmd

- Replaces legacy TOS specific dispatcher
- Handles EL2 context save/restore for world switch
- Handles PSCI calls
- Forwards FF-A calls to normal and secure world using ‘eret’

TF-A initialization

- BL2 loads spmc and SP images (optee).
- Uses TBBR standards

SPMC manifest file with OP-TEE details

- Based on properties listed in FF-A spec
- Provides SPMC configuration like
  - SPMC load address, memory size, entry point
- Contains details on SP (OP-TEE) with
  - load address, vCPUs, SP memory size

```
compatible = "arm,ffa-core-manifest-1.0";
attribute {
    spmc_id = <0x8000>;
    maj_ver = <0x1>;
    min_ver = <0x0>;
    exec_state = <0x0>;
    load_address = <0x0 0x6000000>;
    entrypoint = <0x0 0x6000000>;
    binary_size = <0x80000>;
};

hypervisor {
    compatible = "hafnium,hafnium";
    vm1 {
        is_ffa_partition;
        debug_name = "op-tee";
        load_address = <0x6280000>;
        vcpu_count = <8>;
        mem_size = <30932992>;
    };
};
```

SPMC Manifest file entries

Arm FF-A driver in Linux

Arm FF-A driver

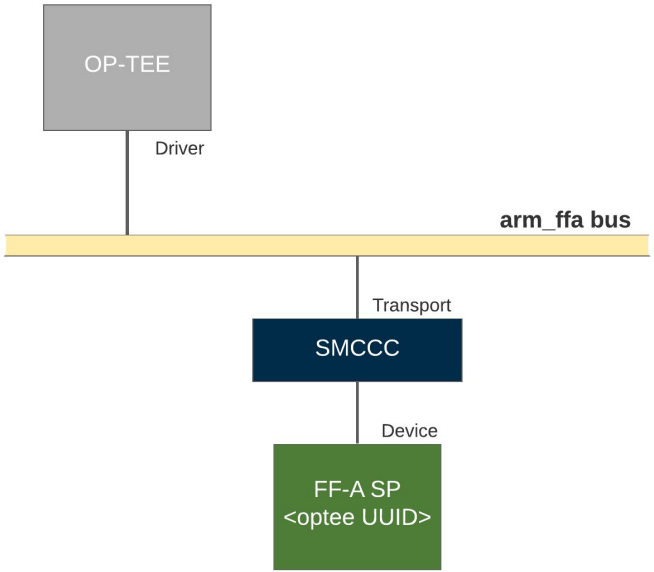
- Provides interface for all client drivers that uses FF-A
- Bus model: Creates FFA device to communicate with secure partitions.
- Uses SMCCC as transport (SMC/HVC)

Initialization

- Creates logical bus arm_ffa
- Probes devices (FF-A SPs) based on UUID in device tree
- Sets up RXTX mapping

FFA ops

- Direct message send / receive
- Memory share
- Memory reclaim



OP-TEE driver FF-A adoption

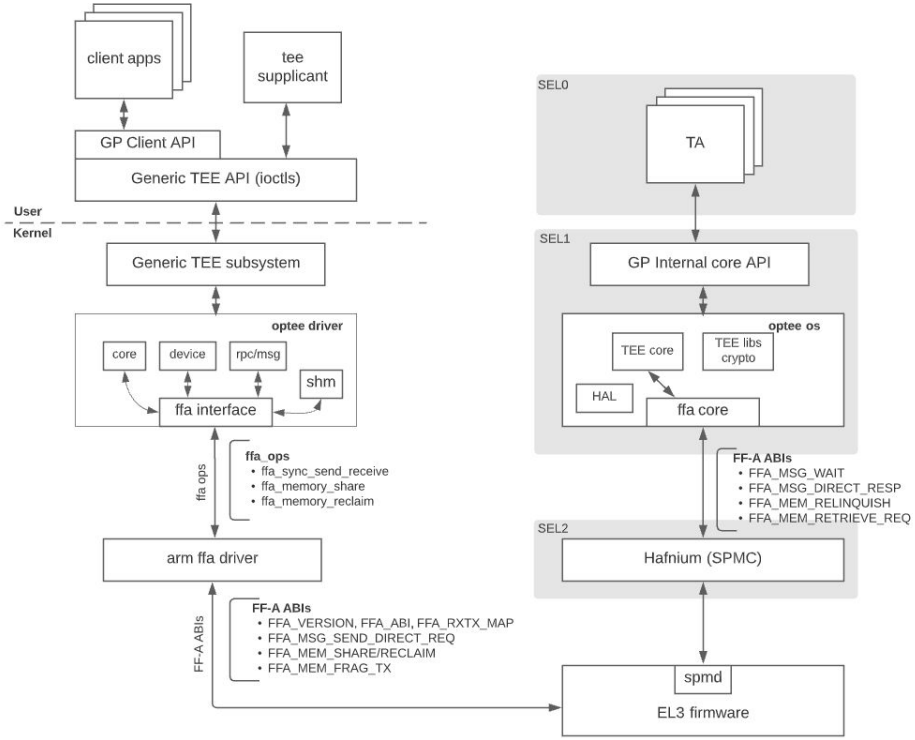
FF-A Transport adoption

- Replaces OP-TEE SMCs with FF-A calls
- OP-TEE client ffa ops, OP-TEE supplicant ffa ops
- OP-TEE core ffa ops(rpc/msg)

OP-TEE driver init

- Init optee_ffa_probe
  - called from ffa driver
  - fetches ffa_ops for ffa_device
  - FFA_VERSION compatible
  - OPTEE_FFA_EXCHANGE_CAPABILITIES
- DT bindings has details on OP-TEE UUID

```
ffa {
    compatible = "arm,ffa-1.0";
    conduit = "smc";
        optee {
        compatible = "arm,ffa-1.0-partition";
        uuid = "486178e0-e7f8-11e3-bc5e-0002a5d5c51b";
    };
};
```



OP-TEE driver FF-A calls

OP-TEE FF-A blocking calls

- Used in init code path

➢ `OPTEE_FFA_GET_API_VERSION`
➢ `OPTEE_FFA_GET_OS_VERSION`
➢ `OPTEE_FFA_EXCHANGE_CAPABILITIES`

OP-TEE FF-A yielding calls

- Invokes OP-TEE in secure world for TEEC commands
- Registers/Unregisters memory with TEE
- Resumes calls that needs retry (FFA_BUSY)

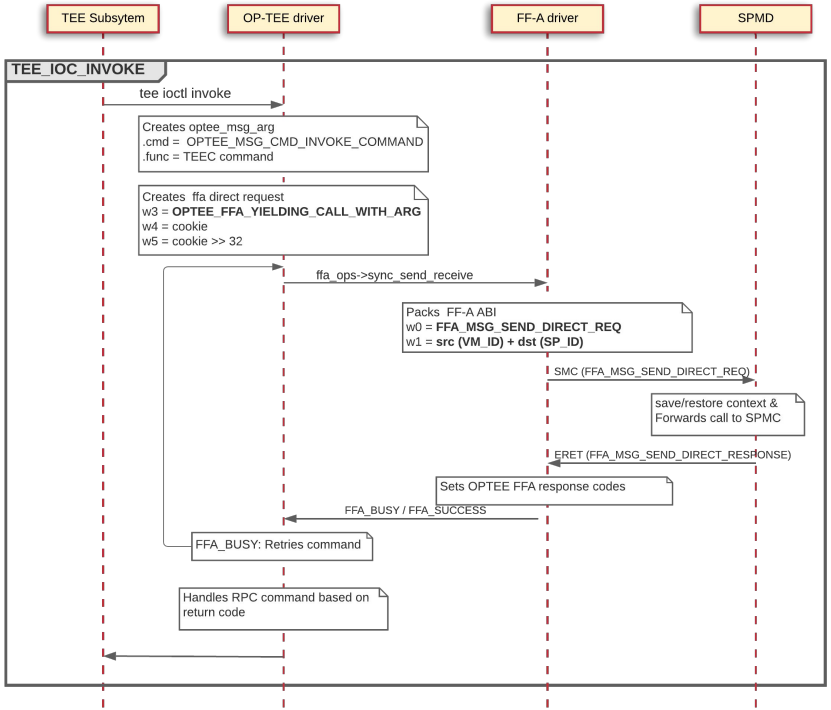
➢ `OPTEE_FFA_YIELDING_CALL_WITH_ARG`
➢ `OPTEE_FFA_YIELDING_CALL_UNREGISTER_SHM`
➢ `OPTEE_FFA_YIELDING_CALL_RESUME`

OP-TEE FF-A RPC commands

- Returned from secure world as part of response to direct message

➢ `OPTEE_FFA_YIELDING_CALL_RETURN_DONE`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_ALLOC_KERN_SHM`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_ALLOC_SUPPL_SHM`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_FREE_KERN_SHM`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_FREE_SUPPL_SHM`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_RPC_CMD`
➢ `OPTEE_FFA_YIELDING_CALL_RETURN_INTERRUPT`

OP-TEE FF-A calls to FF-A ABIs



OP-TEE OS FF-A support

FF-A Support

- Supports both S-EL1 SPMC or S-EL2 SPMC configs, that is, with or without a hypervisor at S-EL2
- Enabled by configuring OP-TEE with CFG_CORE_SEL1_SPMC=y or CFG_CORE_SEL2_SPMC=y

Boot

- Primary CPU boots as usual by starting from the entry point
- Primary CPU informs the SPM of the entry point of the secondary CPUs with PSCI_CPU_ON
  - This is an interim solution until fully covered by the specification
- Secondary CPUs starts from the indicated entrypoint

```
/* file entry.S */
/* Primary entry point */
FUNC _start , :
...
END_FUNC _start
...
/* Secondary entry point */
FUNC cpu_on_handler , :
...
END_FUNC cpu_on_handler
```

OP-TEE OS FF-A message wait loop

Exit messages

- Booting of a CPU is finished with a FFA_MSG_WAIT
- A Remote Procedure Call, RPC, is initiated with a FFA_MSG_SEND_DIRECT_RESP_32
- FFA_SUCCESS_32, FFA_ERROR etc as responses to other requests

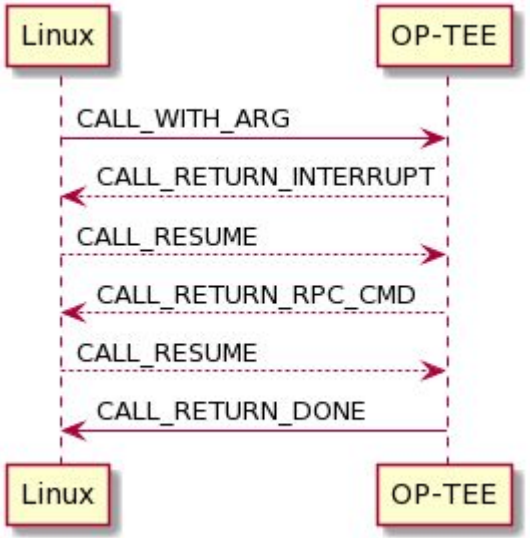
All these messages are sent from the main wait loop

- Execution is resumed at the next instruction as if the message returned
- All exit and re-entry is performed in this loop
  - Invoke return
  - RPC
  - Non-secure interrupt delivery, which also is a kind of RPC
- Re-entry may occur on a different CPU

```
.ffa_msg_loop:
hvc     #0
/* Store parameters on stack */
...
/* parse and handle message */
bl      thread_spmc_msg_recv
/* Load parameters from stack */
...
b       .ffa_msg_loop
```

OP-TEE OS - Messages

- Architected messages are run to completion
- A yielding behaviour is built on top of SEND_DIRECT_REQ and SEND_DIRECT_RESP
- A call can return early to request a RPC either a service or to deliver a non-secure interrupt
- The value in register w3 of SEND_DIRECT_RESP tells the kind of return
  - CALL_RETURN_DONE
  - CALL_RETURN_RPC_CMD
  - CALL_RETURN_INTERRUPT
- The value in register w3 of SEND_DIRECT_REQ tells the kind of yielding request
  - CALL_WITH_ARG
  - CALL_RESUME



OP-TEE OS S-EL2 SPMC - Memory sharing

Normal world -> Secure world memory sharing

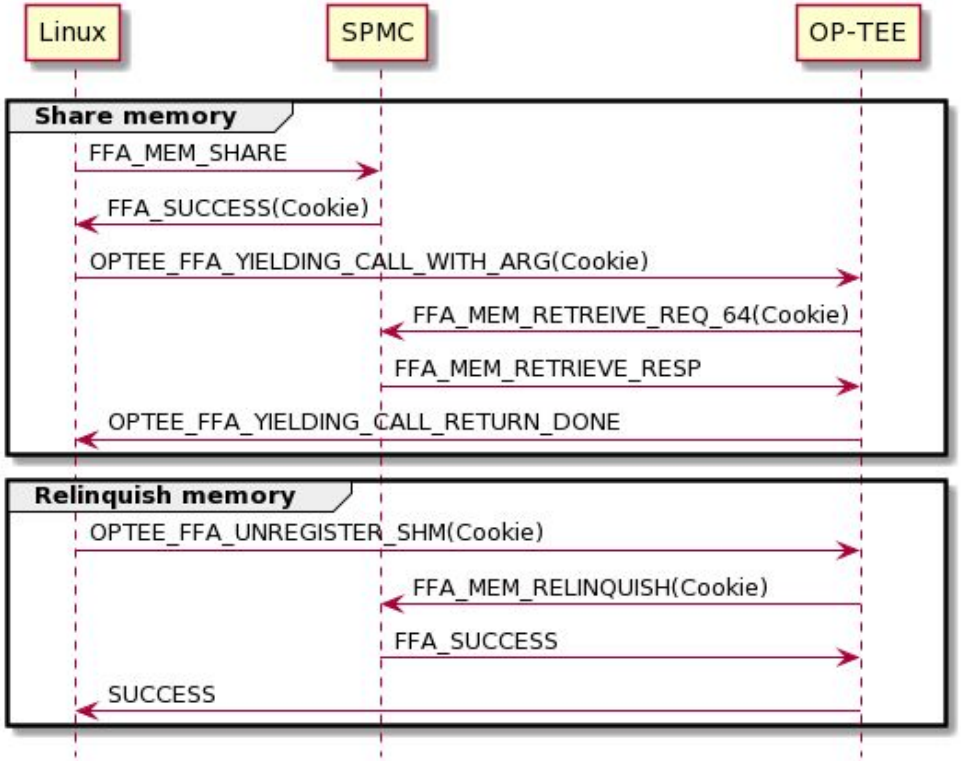
- Initiated with a FFA_MEM_SHARE from normal world
- Caller gets a global handle or a cookie

Memory retrieval

- OP-TEE is called via a normal CALL_WITH_ARG carrying the cookie of the memory to share
- OP-TEE completes the transaction with a FFA_MEM_RETRIEVE_REQ_64
- The memory can now be mapped in OP-TEE

Memory relinquish

- Normal world calls OPTEE_FFA_UNREGISTER_SHM
- If the memory is not actively used OP-TEE does a FFA_MEM_RELINQUISH
- Back in normal world a FFA_MEM_RECLAIM is performed



Arm SEL2 feature, SPM and FF-A

OP-TEE SEL1 SP with SPMC at SEL2

Demo: FVP

Demo on FVP

FVP with S-EL2 Feature

- OP-TEE as SEL1 SP, Hafnium as SEL2 SPMC
- Linux kernel 5.11 with FFA driver and optee driver with FFA_TRANSPORT support
- Hafnium and TF-A version 2.4
- OP-TEE (OS/client/tests) - version 3.11
- Many patches are Work in progress.

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה