OpenIoT Summit Europe 2018

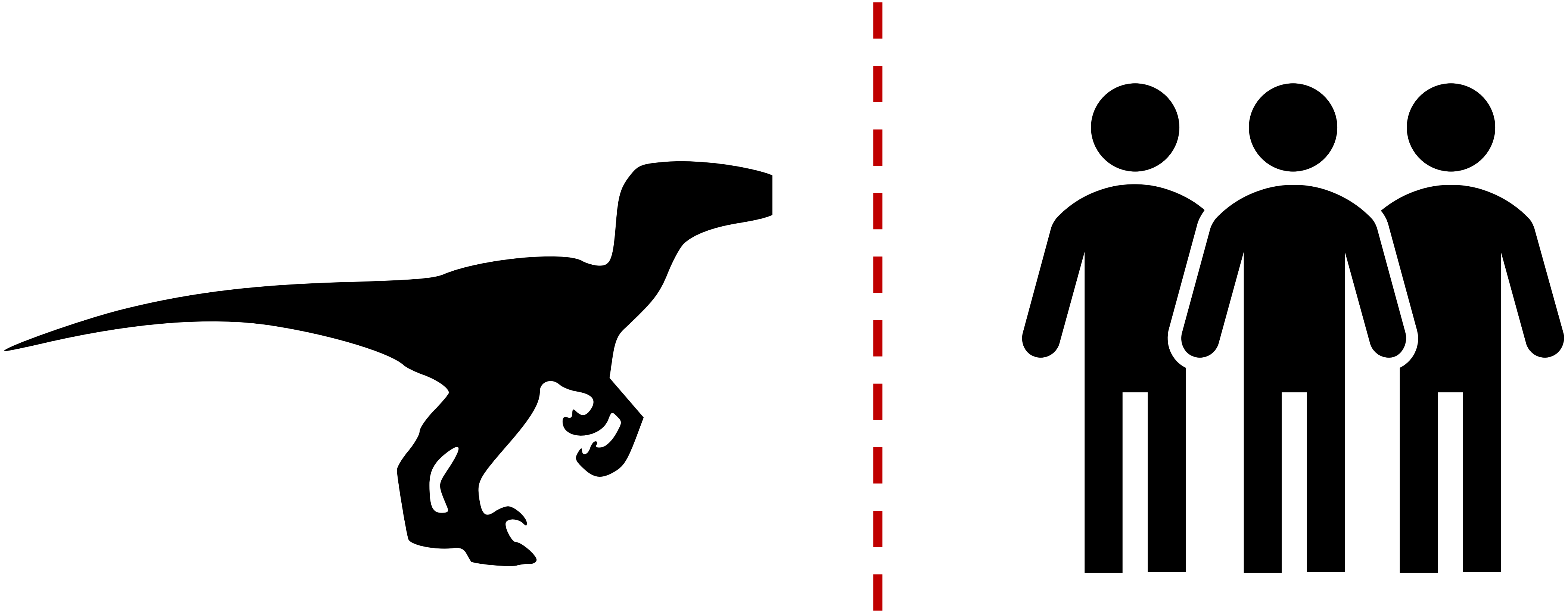
Compartmentalization in IoT

Trusted Firmware M Secure Partitioning

Miklos Balint
Ken Liu
Arm

Non-Confidential © Arm 2018

Compartmentalization is important...



Challenges in IoT

High volume, low cost, low power

- Microcontrollers
  - Small die
  - No MMU (single, physical address space)
  - XIP Flash code
  - Small SRAM

Wide spectrum of use-cases

- Different threat models
- Scalable solutions

Holistic approach to IoT security needed

Establishing the "right" level of security

Secure domain

Basic isolation – create a Secure Processing Environment

Protected TCB

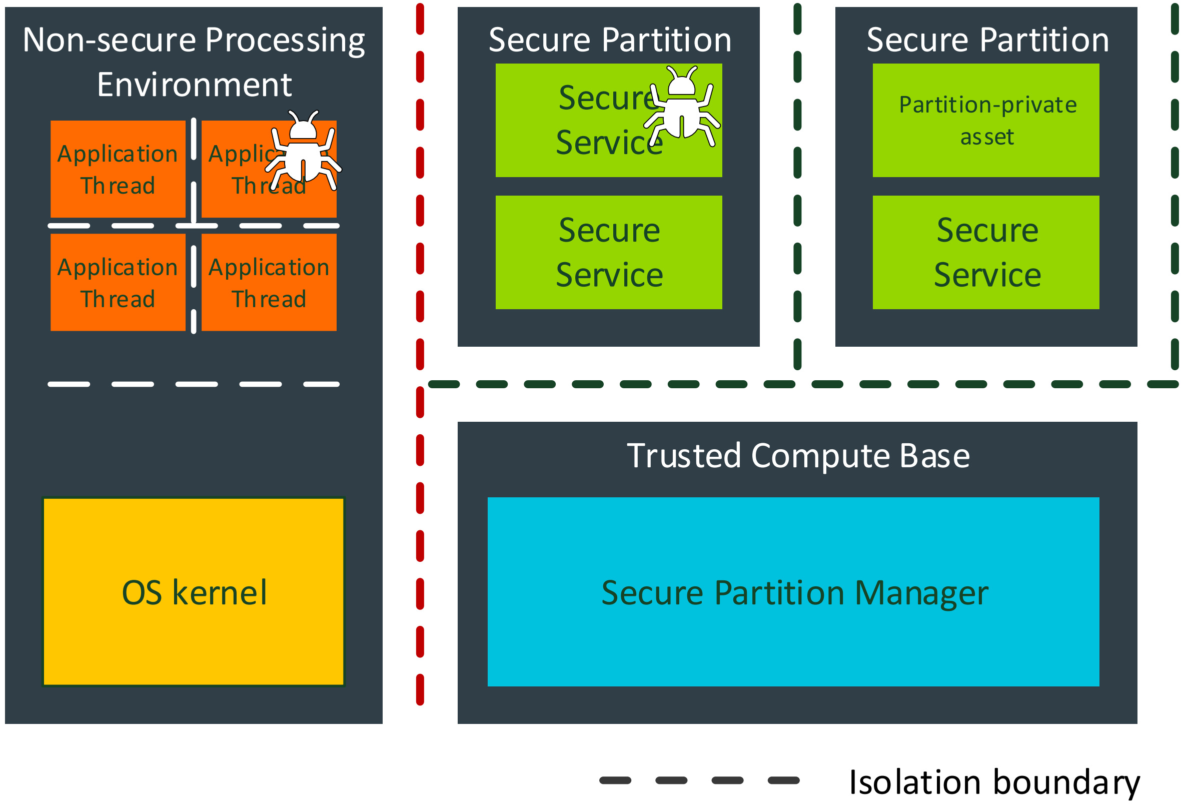
Separate Root of Trust from Secure Partitions within SPE

Multiple tenancy in secure PE

More robustness – isolate all partitions from each other

Non-Secure isolation

Access policies for NS threads
Concurrent contexts



Hardware isolation

… the foundation for software security

Physical isolation (e.g. dual-core system):

Dedicate cores/resources

Shared memory system or Mailbox

Concurrent execution

Temporal isolation (e.g. Arm-v8M):

Privilege control – using MPU

Secure/Non-secure states (Secure Attribution)

Shared Processing Element, resources

Interaction scenarios

Non-Confidential © Arm 2018

Execution flows

Crossing boundaries in single processing element

Crossing from Non-secure to secure state

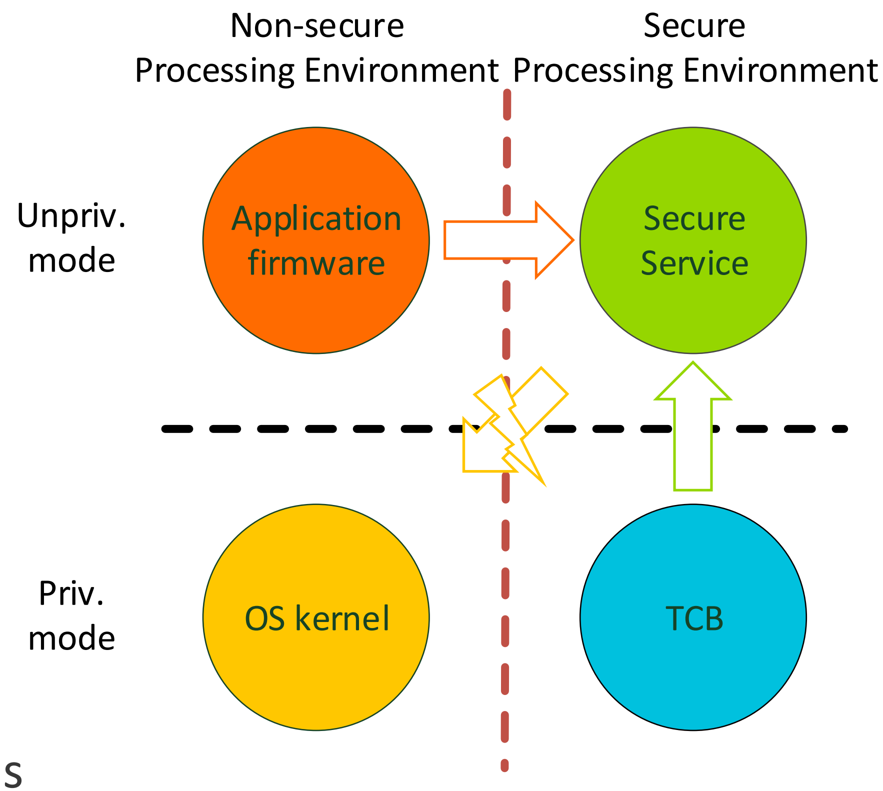
- Non-secure thread requests secure service

Isolated driver code

- ISR execution in unprivileged partition

Asynchronous events in non-secure PE

- Non-secure interrupt pre-empts secure operation
- Non-secure context awareness
- Concurrent secure service requests from non-secure threads



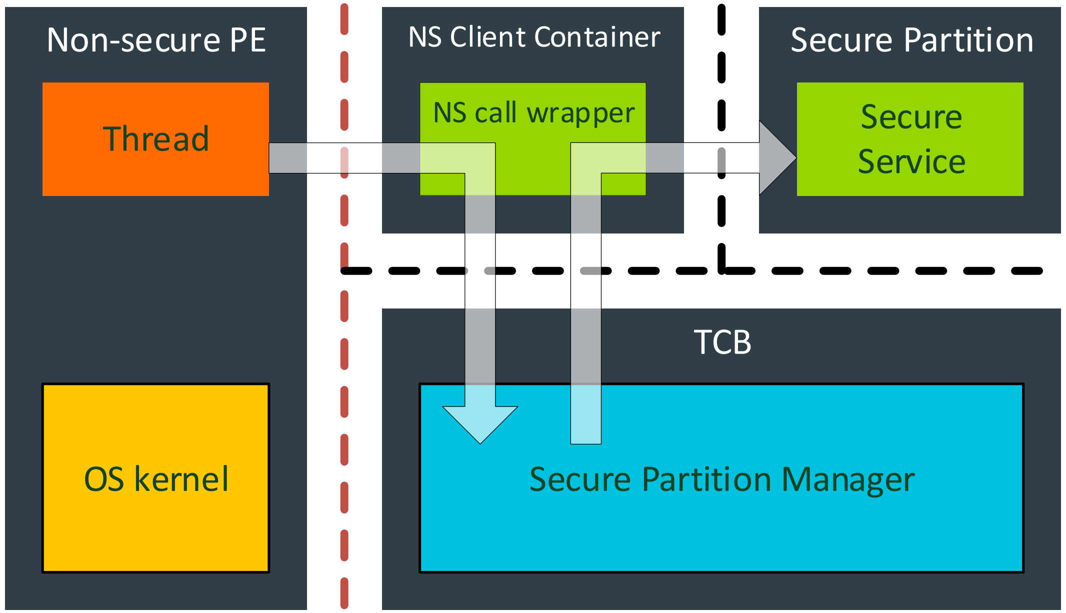
Non-secure call to secure service

Security state change only permitted using dedicated entry points

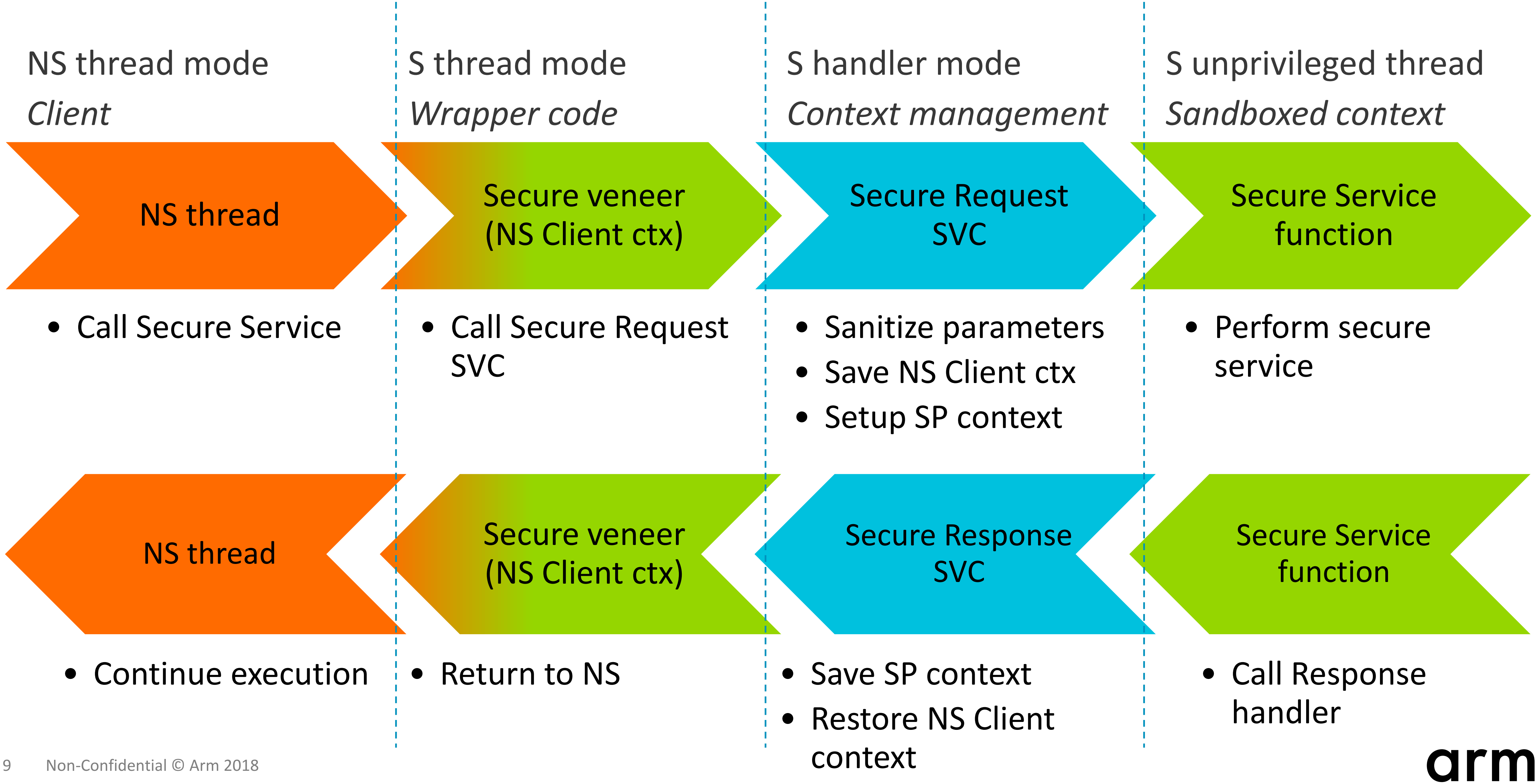
Wrapper function triggers privileged management code

Secure Partition Management code

- Access policy check
- Parameter sanitization
- Secure Partition (container) setup
- Invocation of partition code



Non-secure call to secure service



NS thread mode

Client

NS thread

- Call Secure Service

NS thread

- Continue execution

S thread mode

Wrapper code

Secure veneer (NS Client ctx)

- Call Secure Request SVC

Secure veneer (NS Client ctx)

- Return to NS

S handler mode

Context management

Secure Request SVC

- Sanitize parameters
- Save NS Client ctx
- Setup SP context

Secure Response SVC

- Save SP context
- Restore NS Client context

S unprivileged thread

Sandboxed context

Secure Service function

- Perform secure service

Secure Service function

- Call Response handler

Secure interrupt deprivileging

Device driver in Secure Partition

Privileged ISR is wrapper

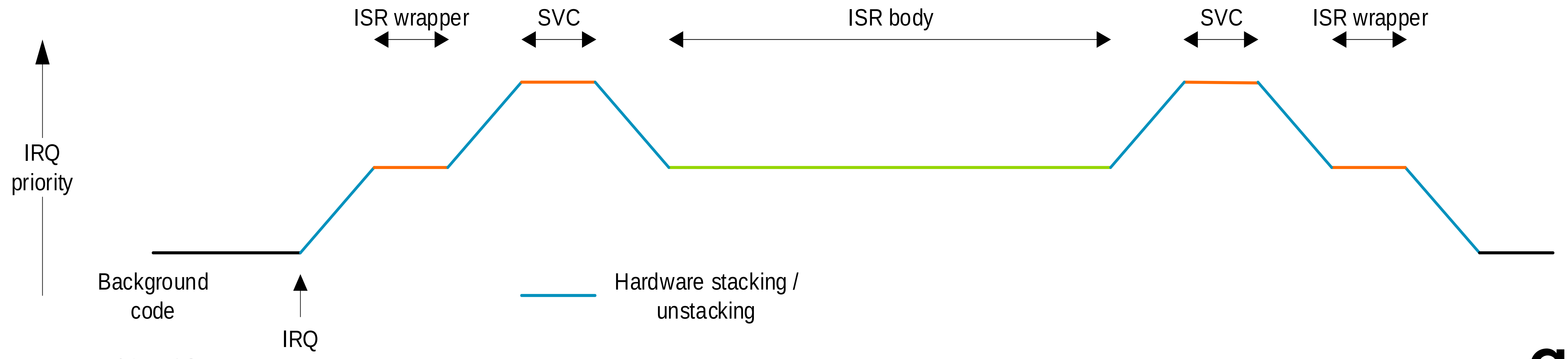
- Triggers Partition Manager

Sandbox created

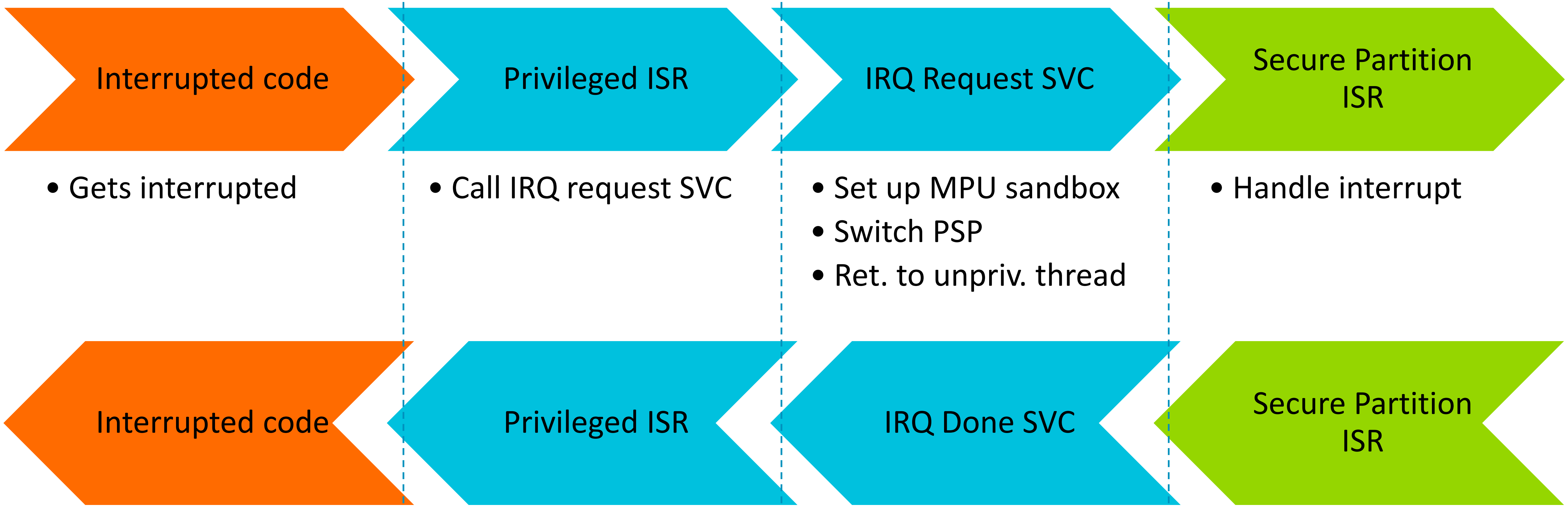
- Returns to thread mode

Secure Partition code

- Executes deprivileged ISR



Secure interrupt deprivileging



Original mode	S handler mode	S handler mode	S unprivileged thread
Original context	*Wrapper code*	*Context management*	*Sandboxed context*
Interrupted code	Privileged ISR	IRQ Request SVC	Secure Partition ISR
• Gets interrupted	• Call IRQ request SVC	• Set up MPU sandbox • Switch PSP • Ret. to unpriv. thread	• Handle interrupt
Interrupted code	Privileged ISR	IRQ Done SVC	Secure Partition ISR
• Continue execution	• Return to original state	• Restore MPU config, PSP • Return to priv. ISR	• Call IRQ Done SVC

Non-Secure interrupts

Pre-emption of secure execution

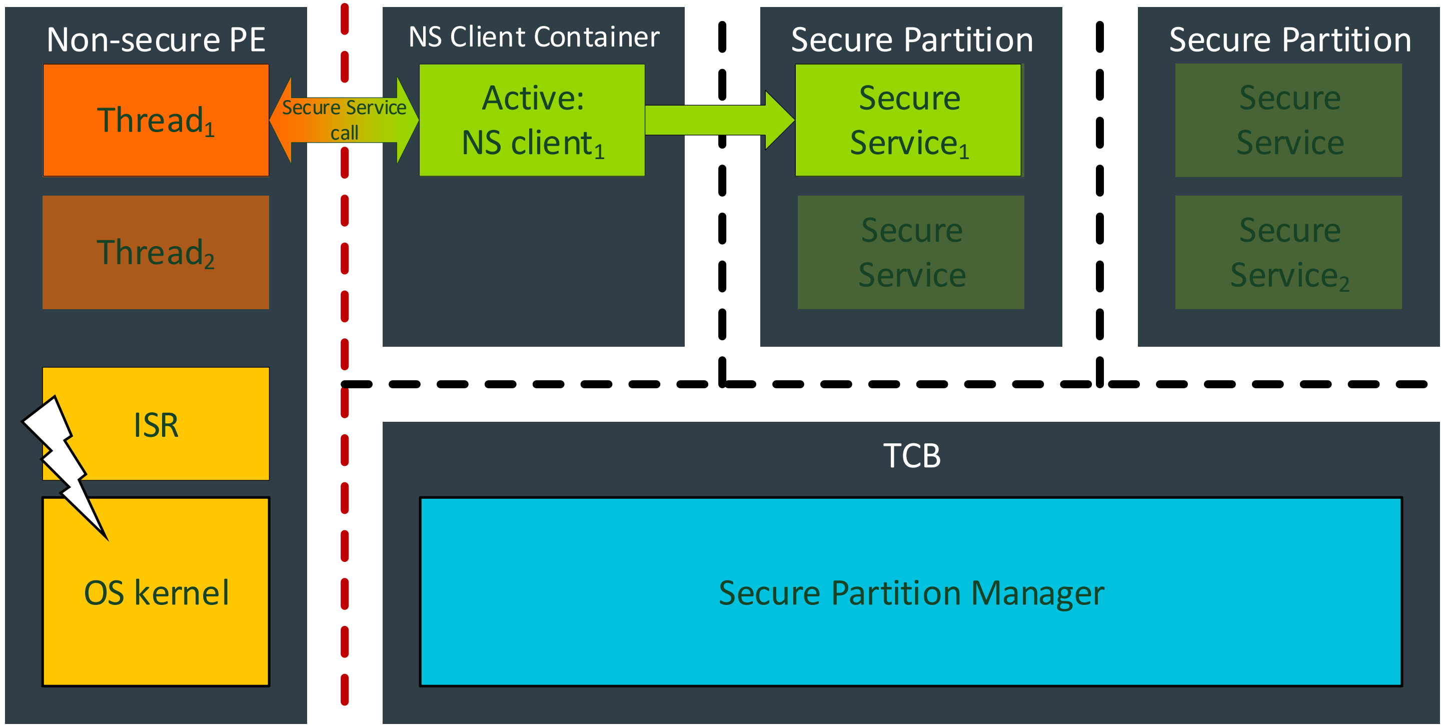
Non-secure IRQ pre-empts secure operation

Secure context is stacked

Non-secure ISR is executed

Return from ISR resumes secure execution

Secure context is unstacked



Context Management Functions

Non-secure context awareness in Arm-v8M

1. Non-secure threads created

2. Thread$_1$ calls Secure Service$_1$

3. Non-secure IRQ pre-empts operation -> context change

4. Thread$_2$ calls secure service$_2$

5. Secure service$_2$ returns

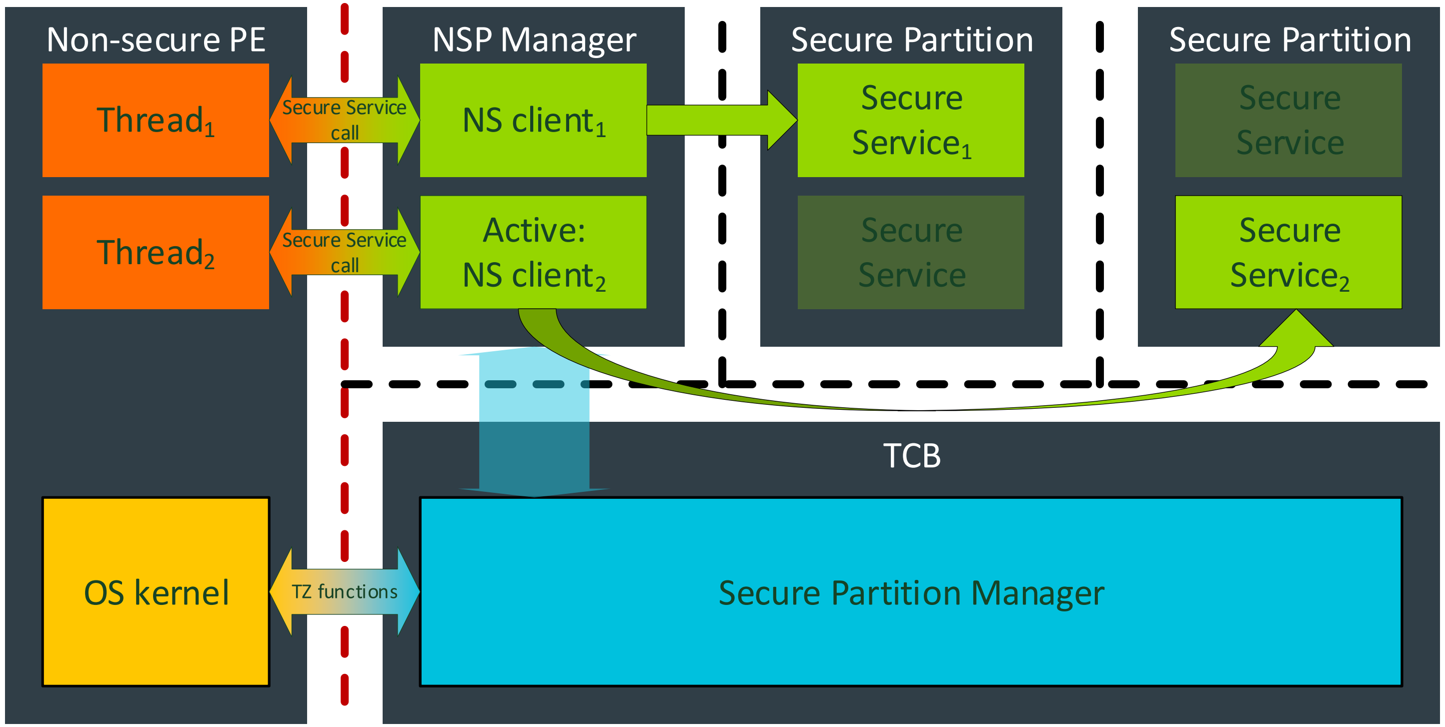
6. Thread$_2$ yields

7. Secure Service$_1$ returns

NS RTOS -> SPM notifications:

Thread creation, deletion, load or store

Enables NS context-dependent access to secure assets/services



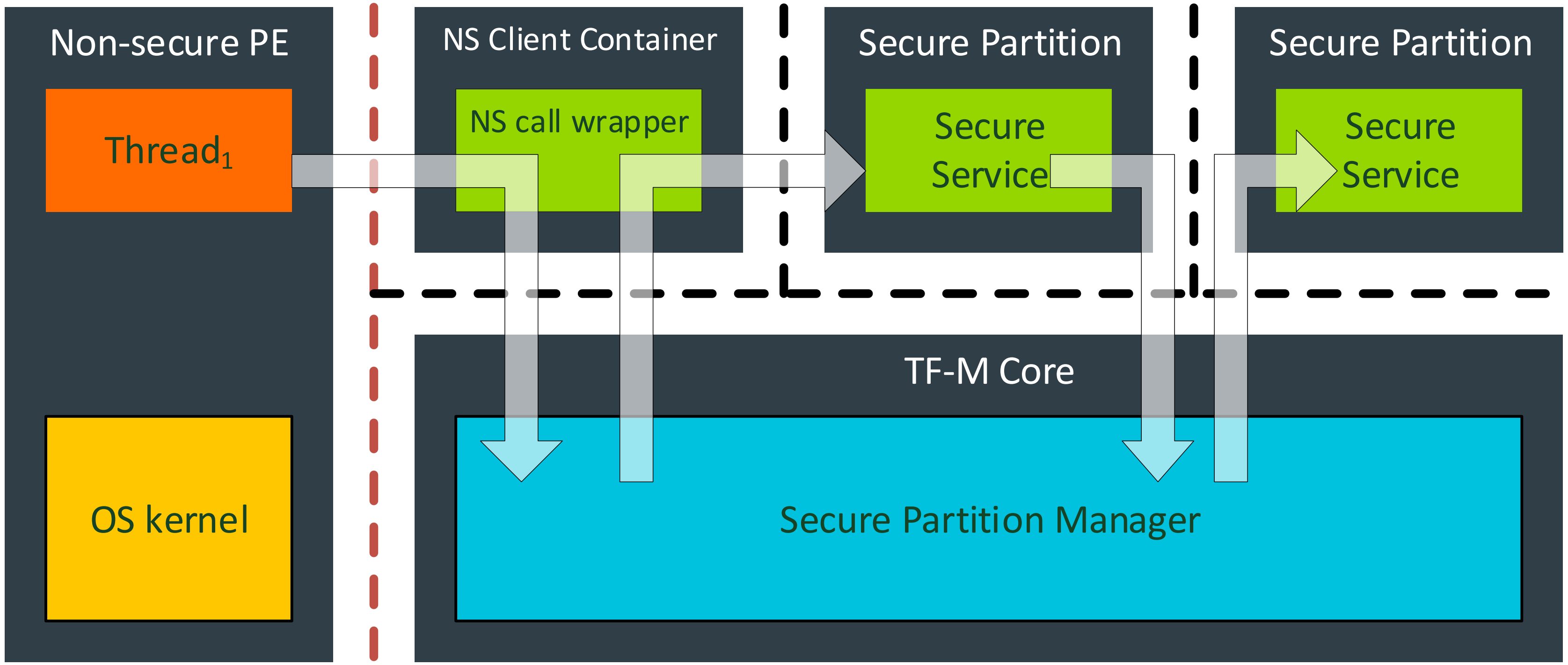
Implementations

Non-Confidential © Arm 2018

Trusted Firmware M library model

Secure Services implemented as functions

- ~ bare metal programming model
- Arm-v8M architecture support
- Secure Partition: library
- Synchronous execution
- Low footprint

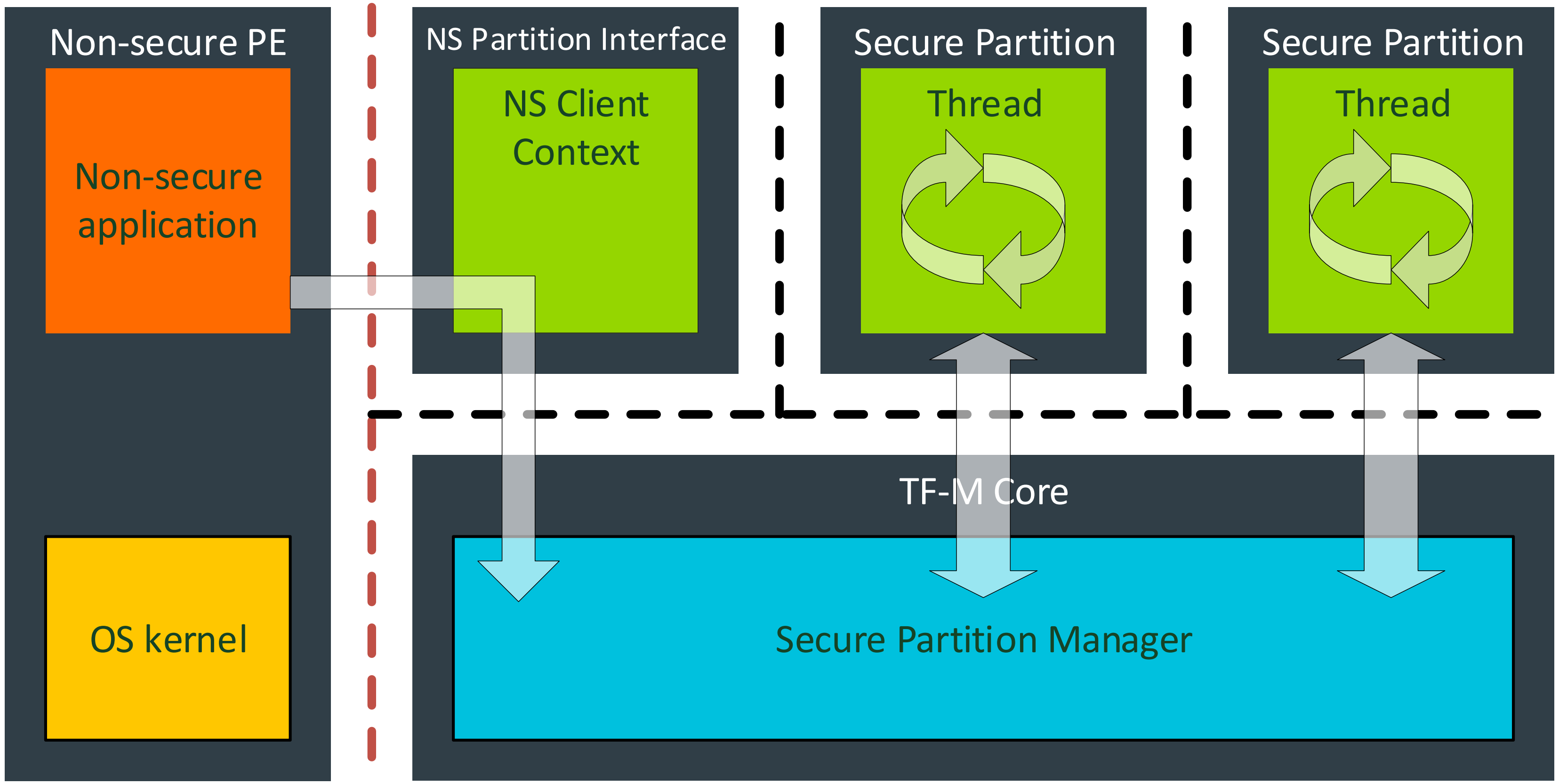


Non-Confidential © Arm 2018

Trusted Firmware M thread model

Secure Partitions implemented as threads

- Robust, more prescriptive framework
- Static allocation of secure resources
- Connection/message based interaction
- Asynchronous processing of service requests

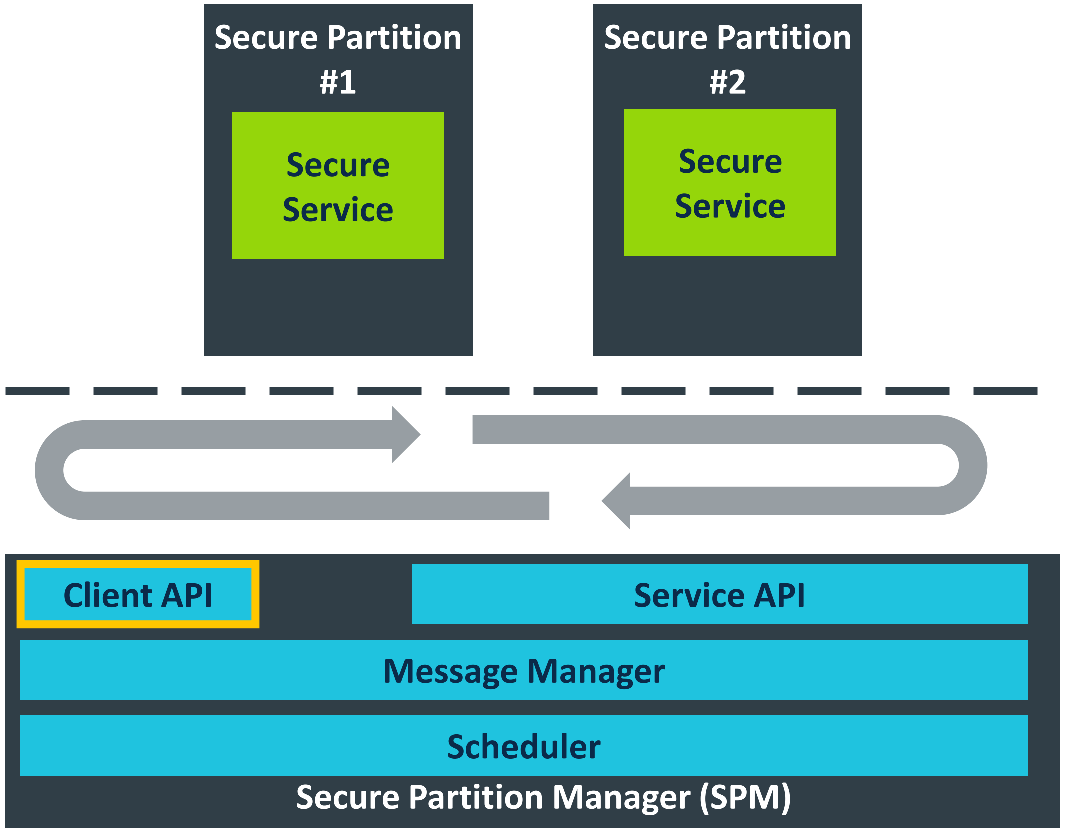


Interaction in thread model

Non-Confidential © Arm 2018

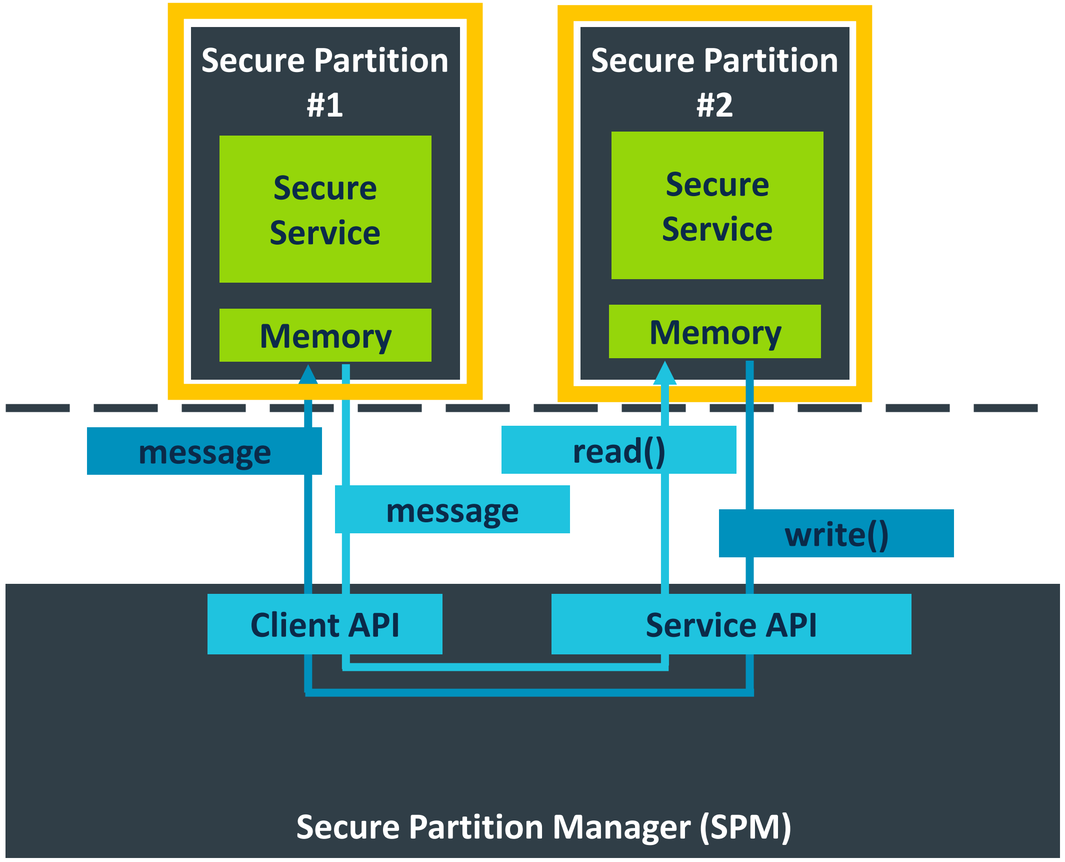
TF-M Inter-Process Communication (IPC)

- **For TF-M Thread model**
- **Secure Partitions provide secure services**
  - NSPE is reflected as one Non-Secure Partition
- **One thread in one Secure Partition**
- **While loop in thread waiting for messages**
- **Client call sent as messages**
  - Non-Secure Partition is a client
  - Secure Partition could be a client
- **Service Interrupt is handled asynchronously**

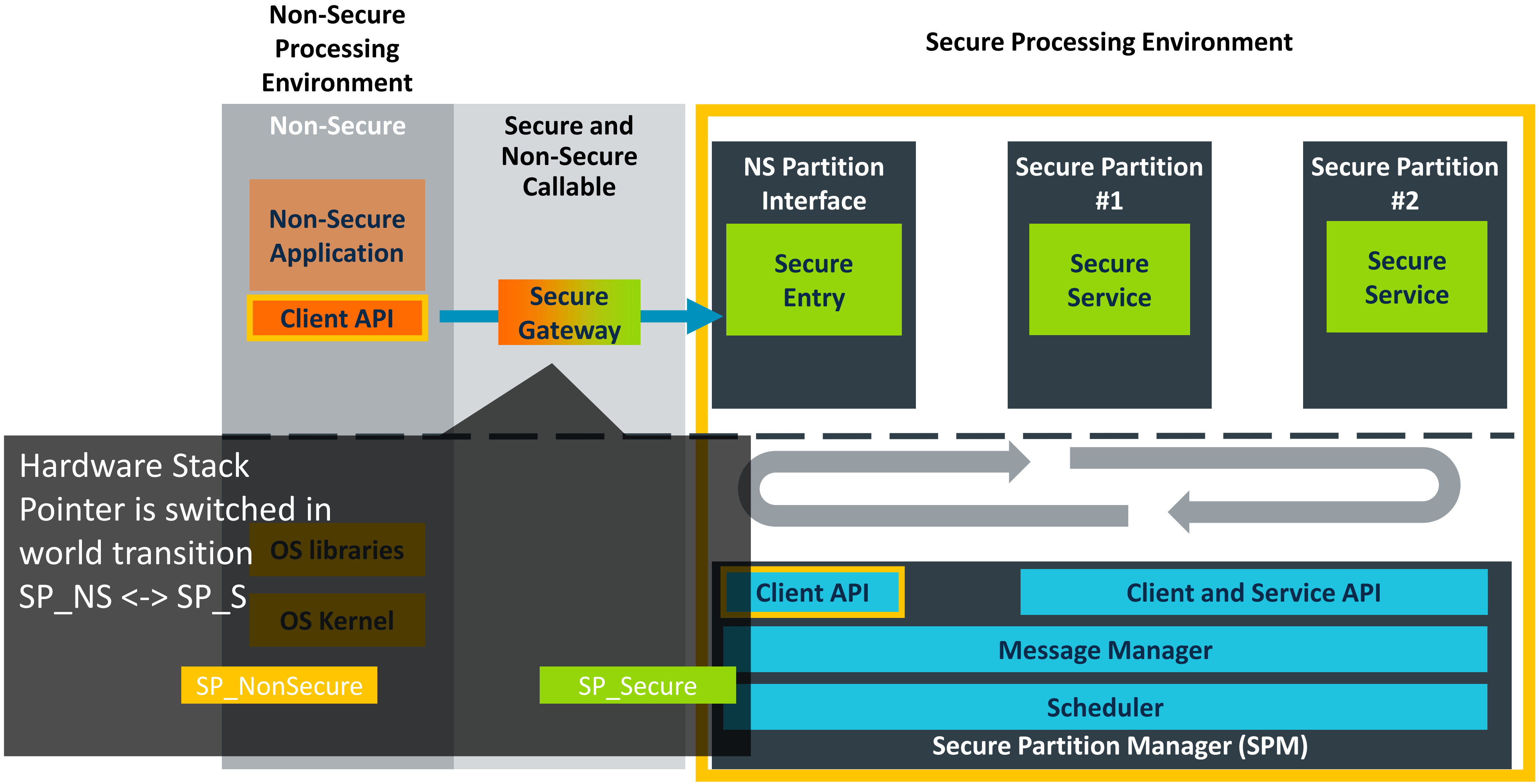


Security Consideration on Compartmentalization

- No shared memory between partitions
- Memory copy by streamed read/write API
- Memory integrity checking in SPM based on isolation level
- Peripheral usage is also Compartmentalized
- Runtime protection rule change

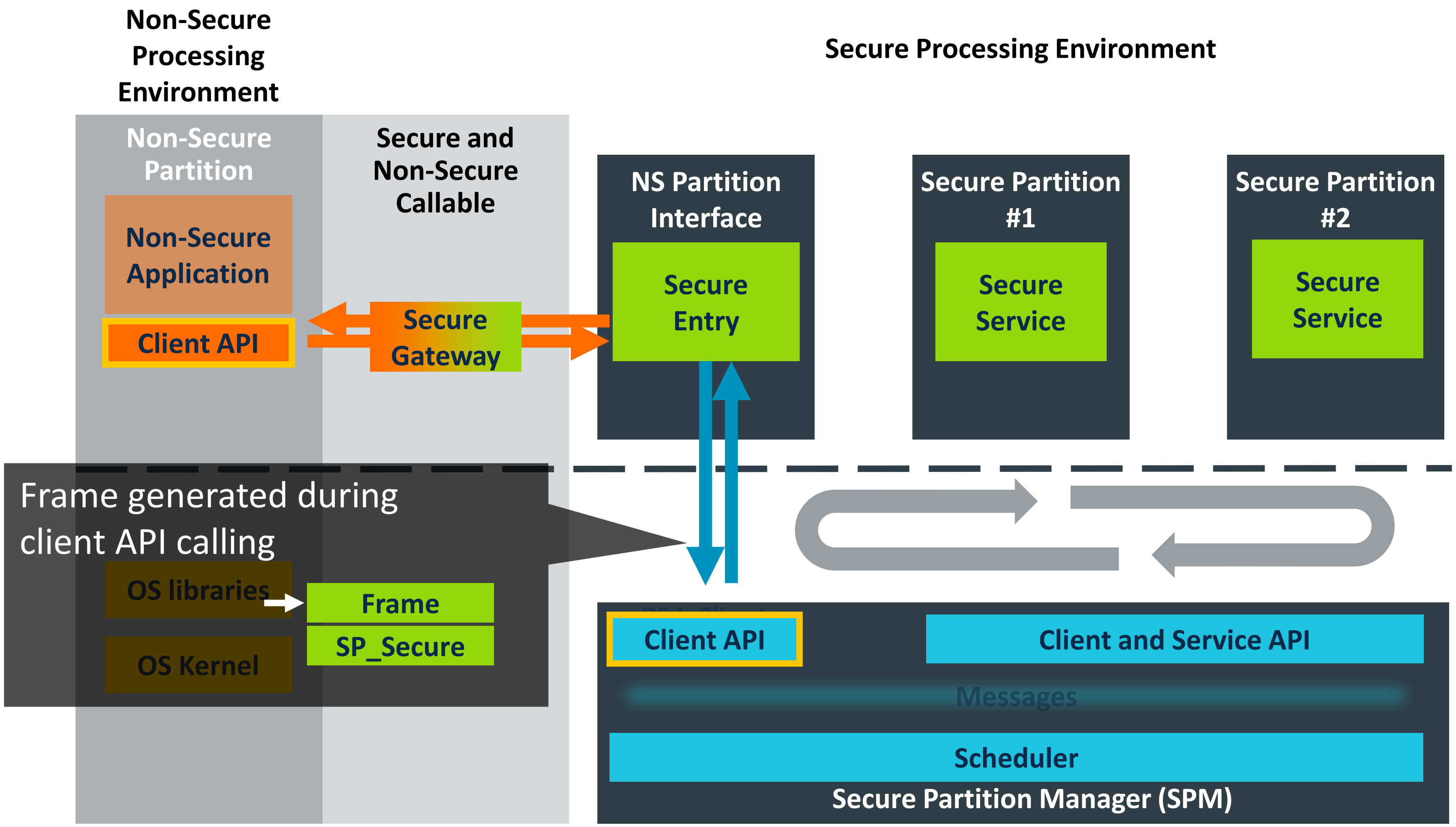


Expand NSP with Arm-v8M TrustZone



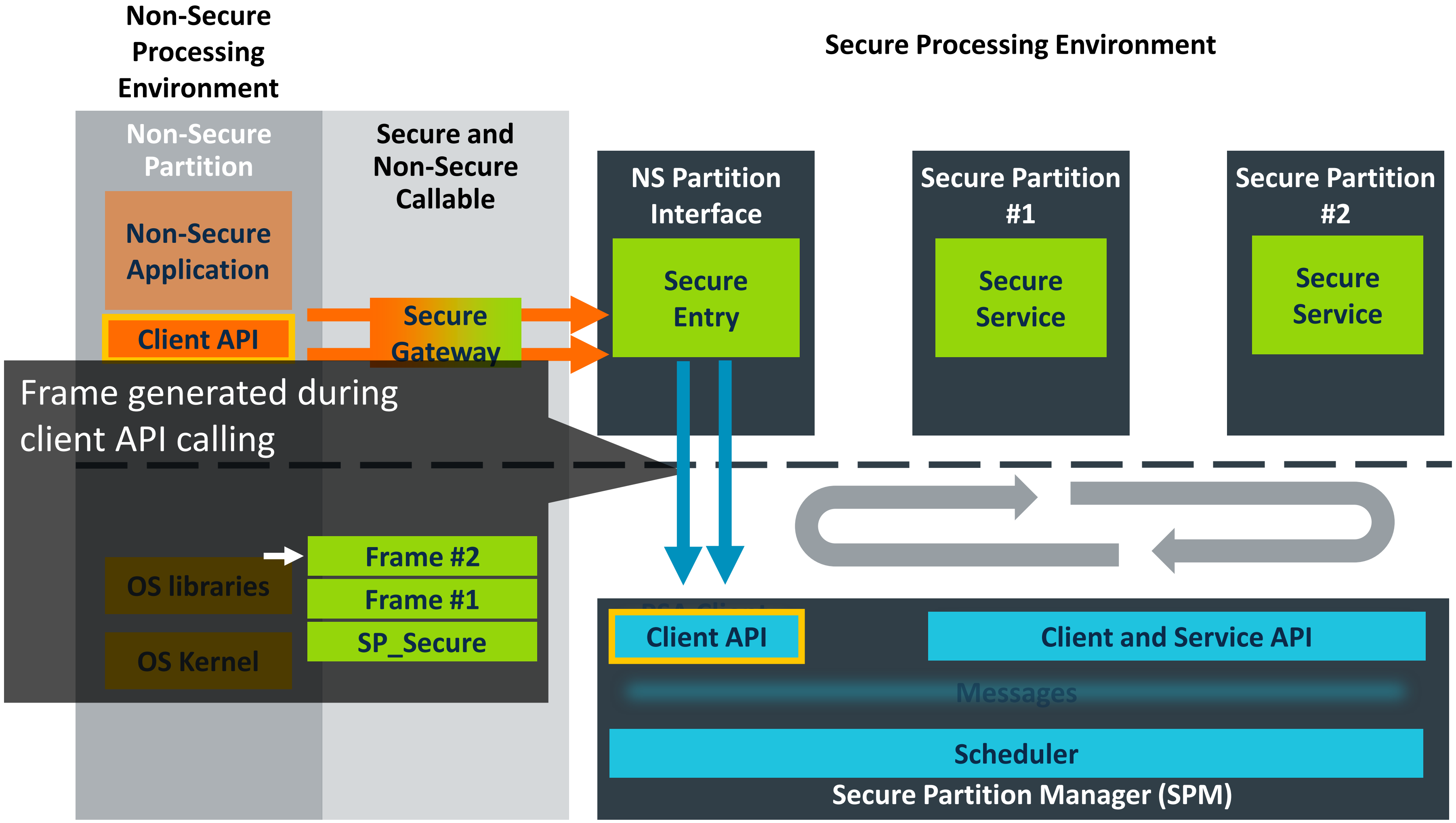
Hardware Stack Pointer is switched in world transition SP_NS <-> SP_S

Single NS Thread requests Secure Service



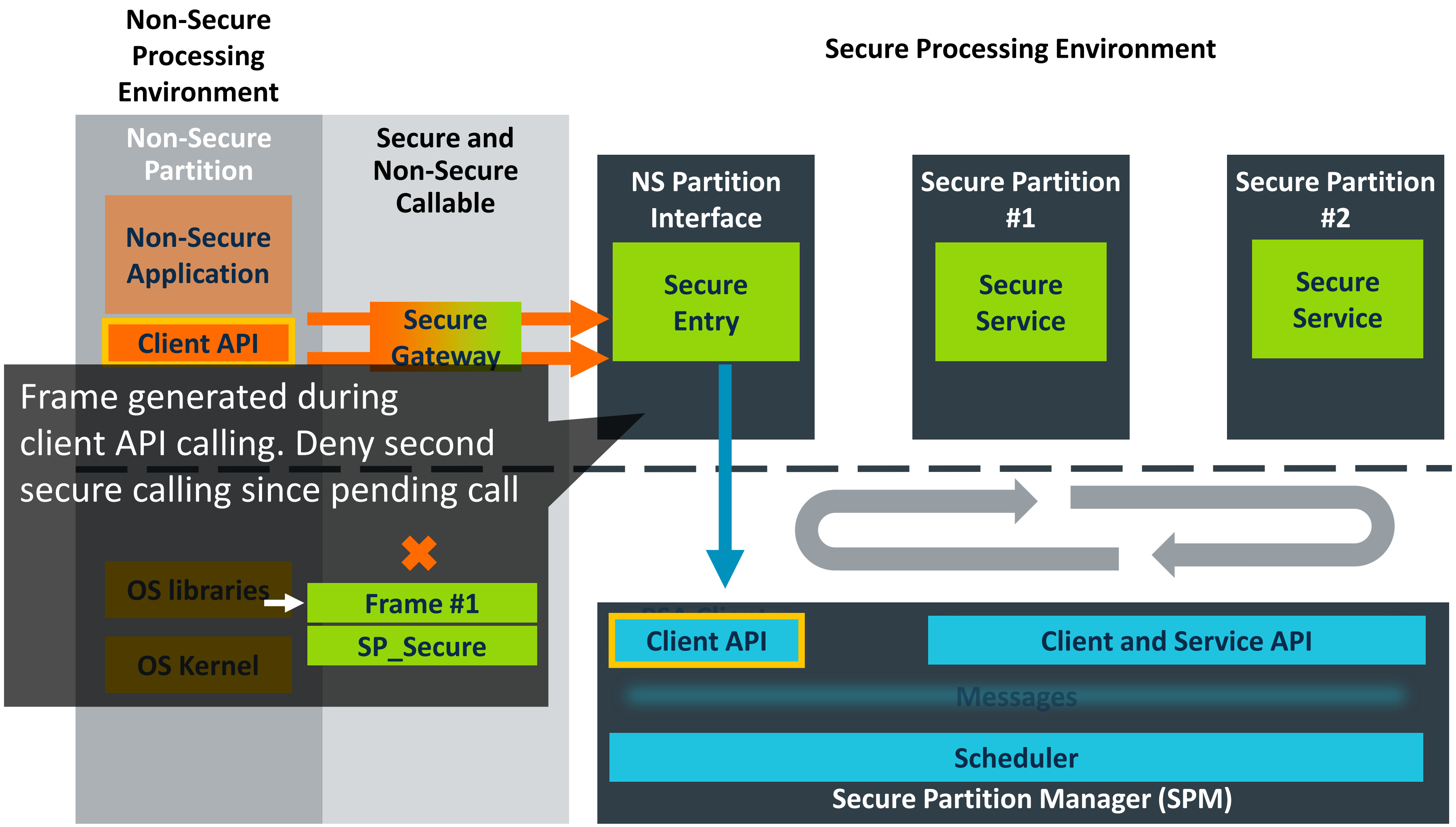
Frame generated during client API calling

Multiple NS Thread request Secure Service



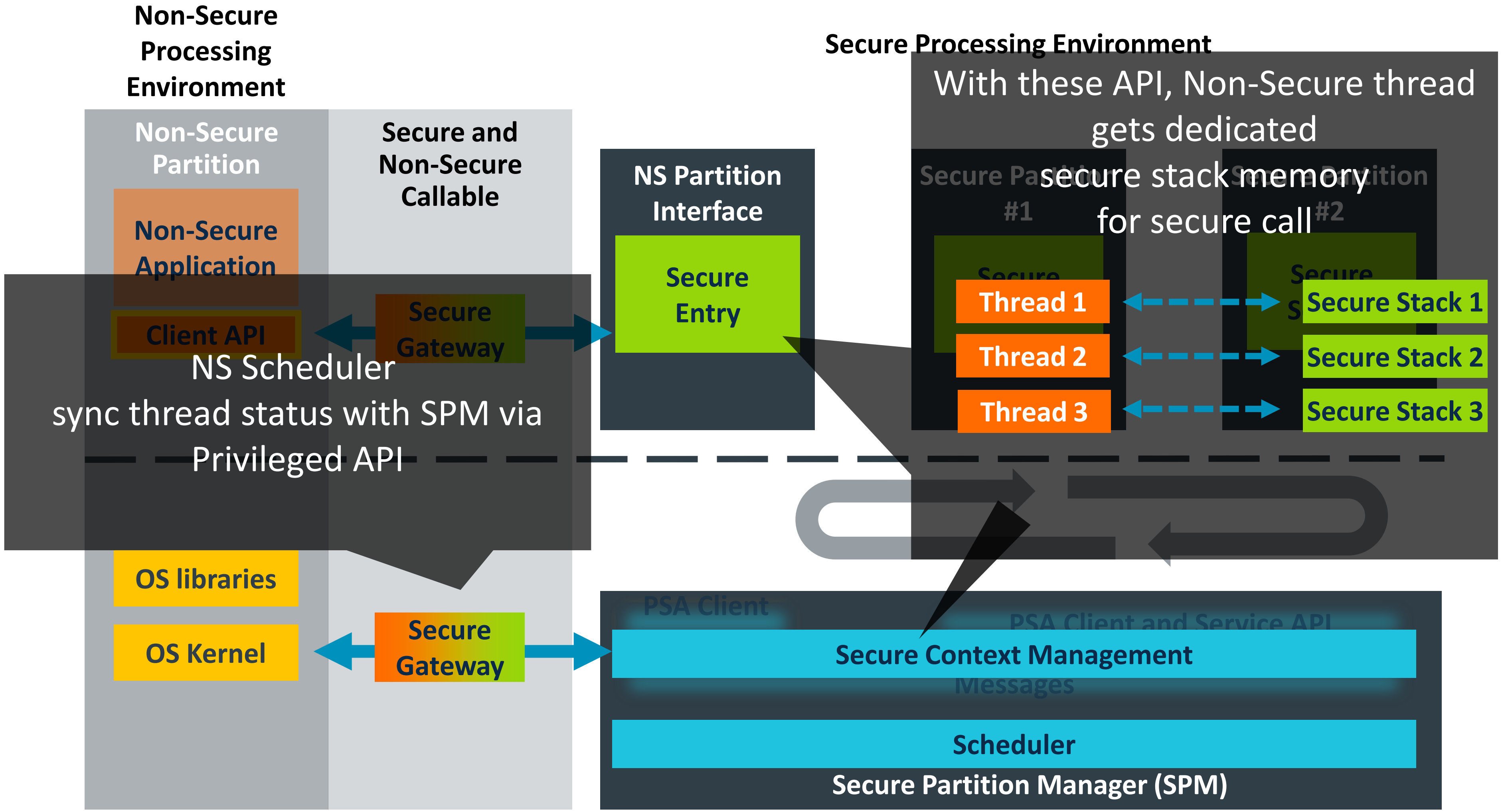
Frame generated during client API calling

Multi-Thread NSPE Secure Call Solution 1



Frame generated during client API calling. Deny second secure calling since pending call

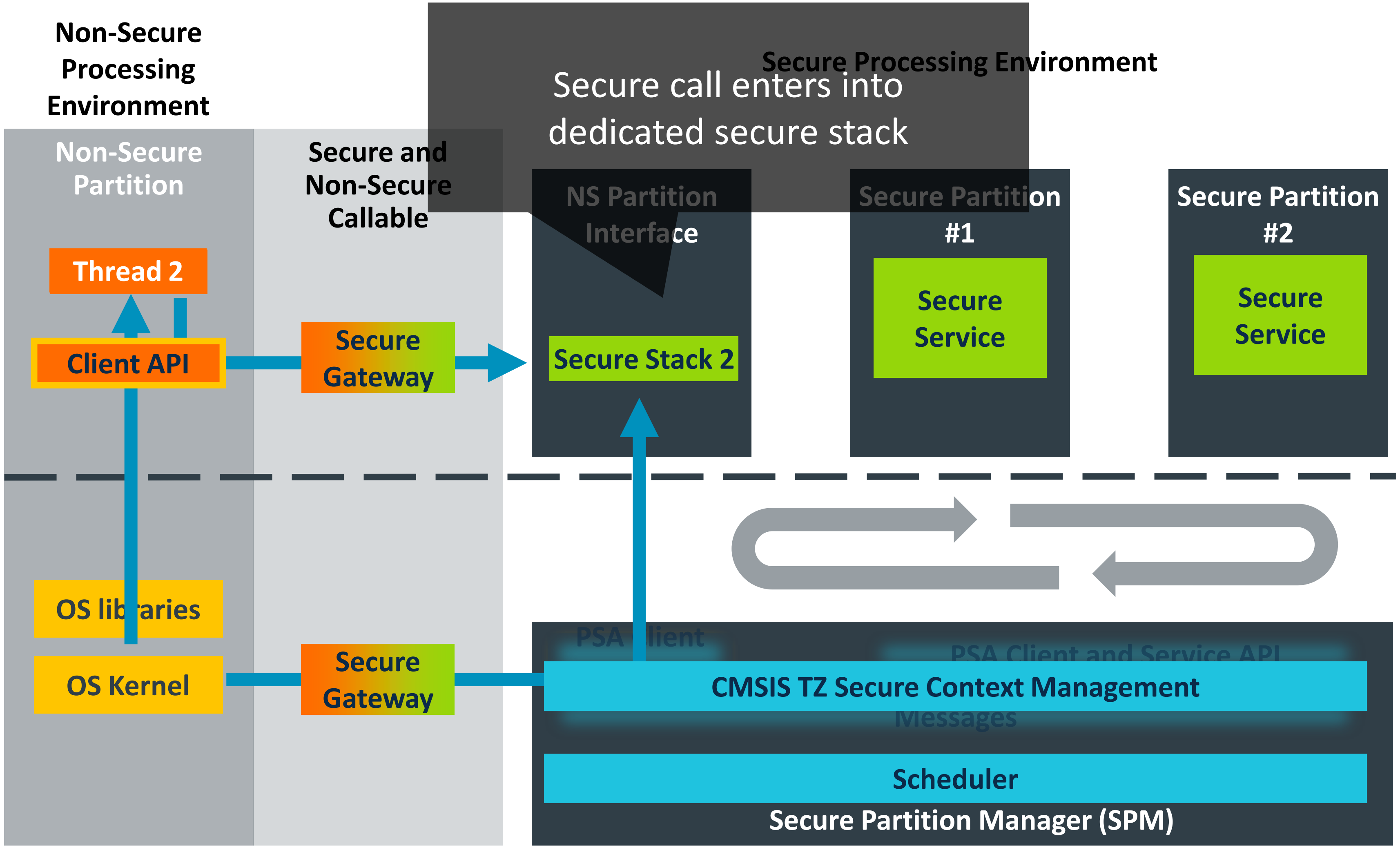
Multi-Thread NSPE Secure Call Solution 2



With these API, Non-Secure thread gets dedicated secure stack memory for secure call

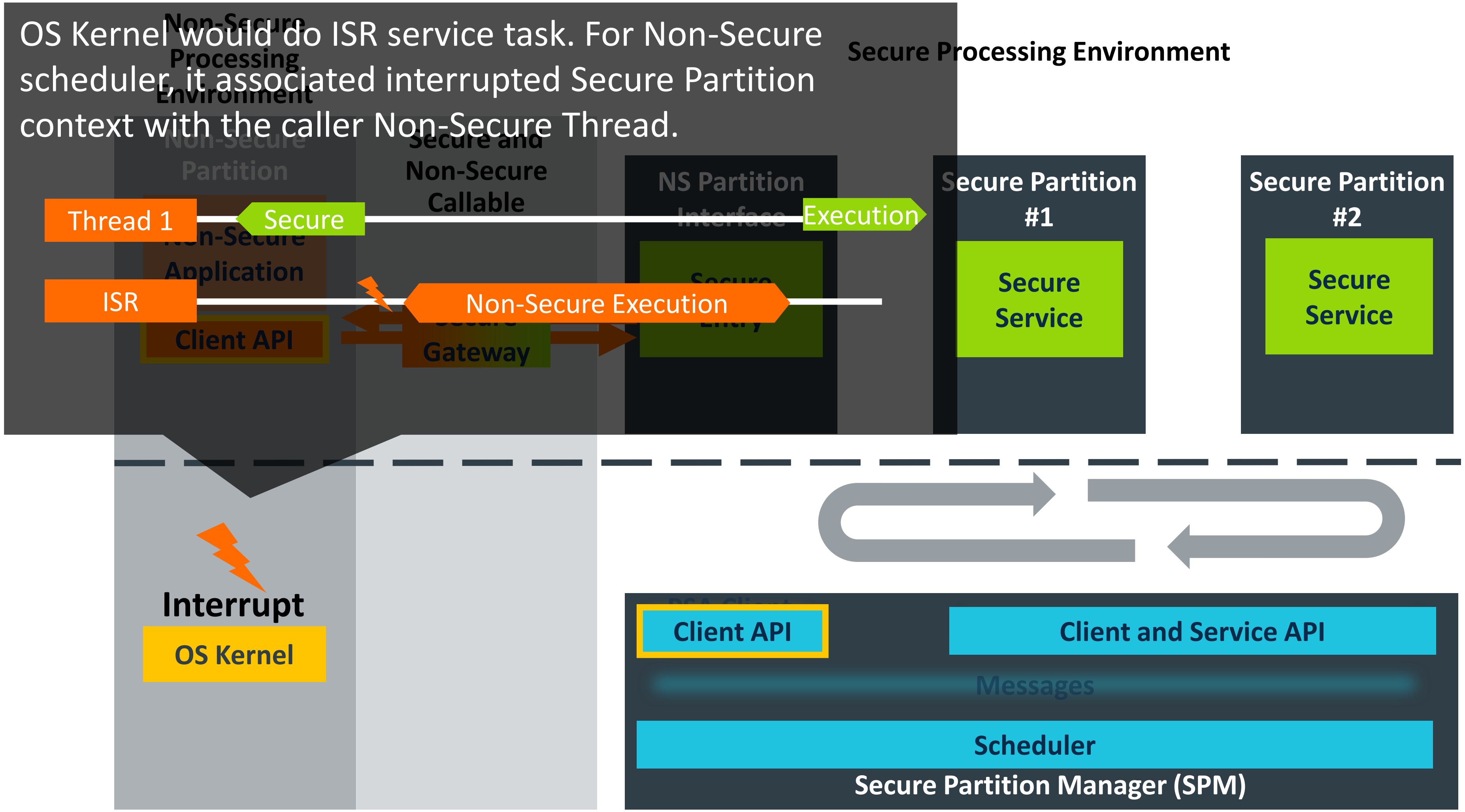
NS Scheduler sync thread status with SPM via Privileged API

Solution 2 Calling Process

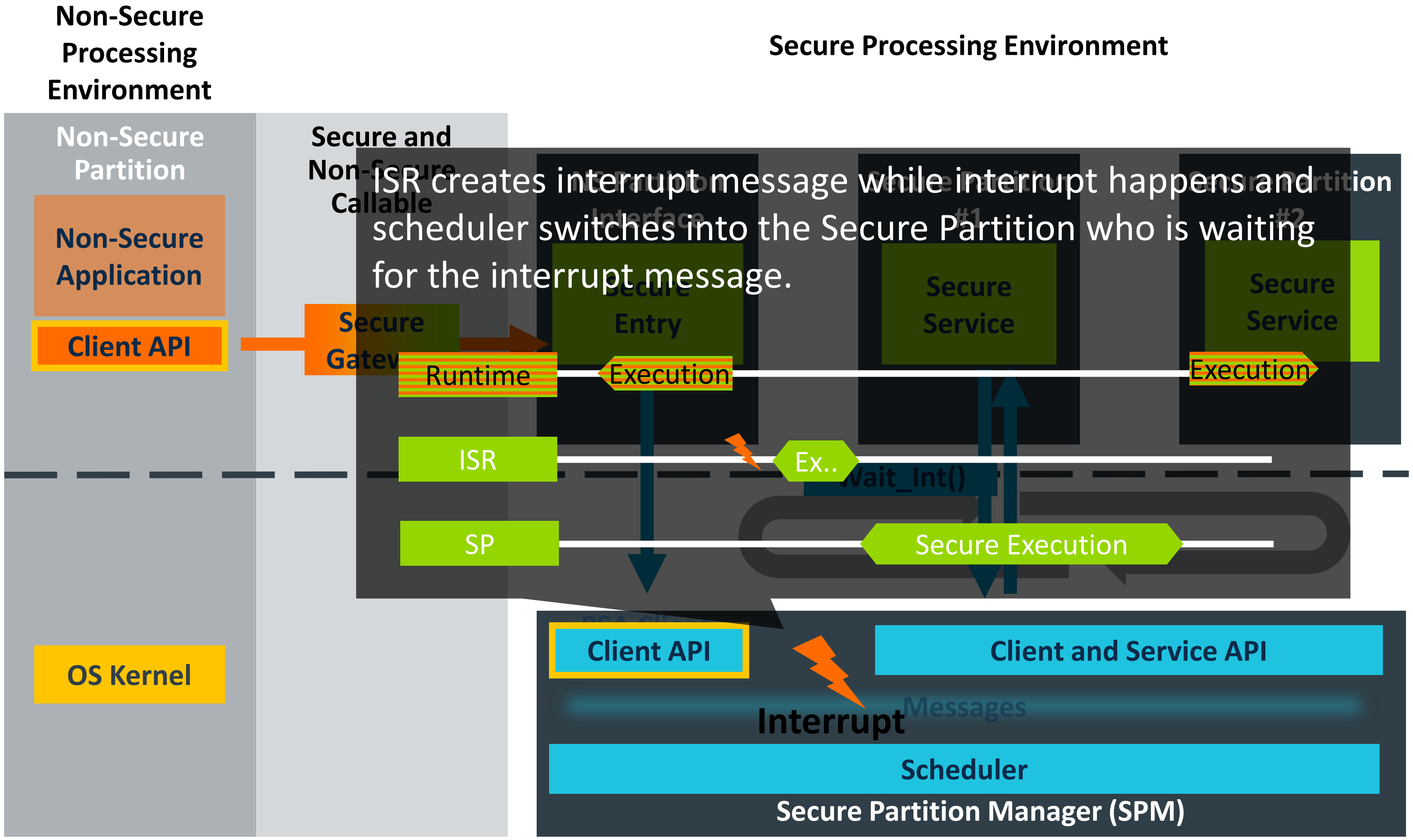

Non-Secure Processing Environment
Secure Processing Environment
Secure call enters into dedicated secure stack
Non-Secure Partition
Secure and Non-Secure Callable
Thread 2
Client API
Secure Gateway
NS Partition Interface
Secure Stack 2
Secure Partition #1
Secure Service
Secure Partition #2
Secure Service
OS libraries
OS Kernel
Secure Gateway
PSA Client
PSA Client and Service API
CMSIS TZ Secure Context Management
Messages
Scheduler
Secure Partition Manager (SPM)

Non-Secure Interrupt Preempts Secure Service

OS Kernel would do ISR service task. For Non-Secure scheduler, it associated interrupted Secure Partition context with the caller Non-Secure Thread.



Secure Interrupt Preempts Execution



ISR creates interrupt message while interrupt happens and scheduler switches into the Secure Partition who is waiting for the interrupt message.

Summary

Compartmentalization in IoT – No one-size-fits-all

Secure/non-secure isolation:
- physical
- temporal

Privilege control:
- none
- within secure domain
- within non-secure domain

Interaction:
- function calls
- IPC
- hardware mailbox

Trusted Firmware M – How to get involved

Part of Open Source/Open Governance trustedfirmware.org project

- Developer space: https://developer.trustedfirmware.org/
- Code base: https://git.trustedfirmware.org/

TF-M Team @ OpenIoT Summit Europe 2018

- Shebu Kuriakose
- Ashutosh Singh
- Ken Liu
- Miklos Balint

Get in touch

- Come round to the Arm booth during the summit
- Contact TF-M team at support-trustedfirmware@arm.com

More info on developer.arm.com and trustedfirmware.org

Non-Confidential © Arm 2018

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

감사합니다

धन्यवाद

arm

arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks