

Agenda

Introduction to PSA and Trusted Firmware M

High Level System Architecture

Overview of services

Use-case Scenarios

Getting Involved

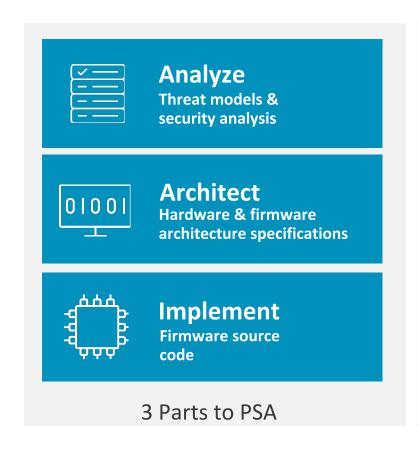
Please feel free to interrupt during the course of this presentation!

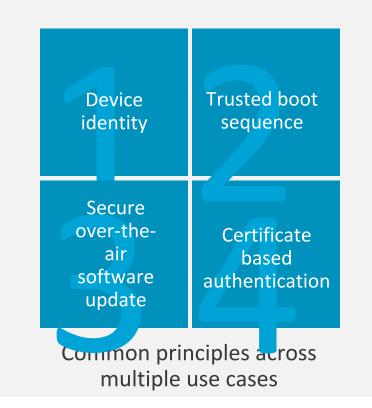


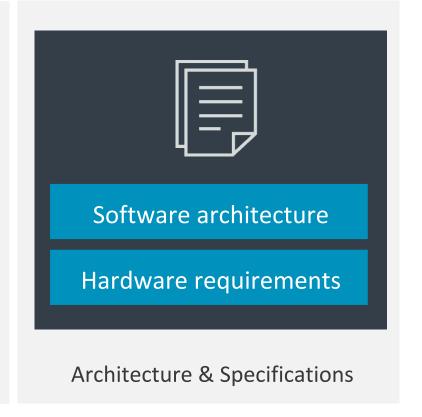
Overview –PSA and TFM



Platform Security Architecture



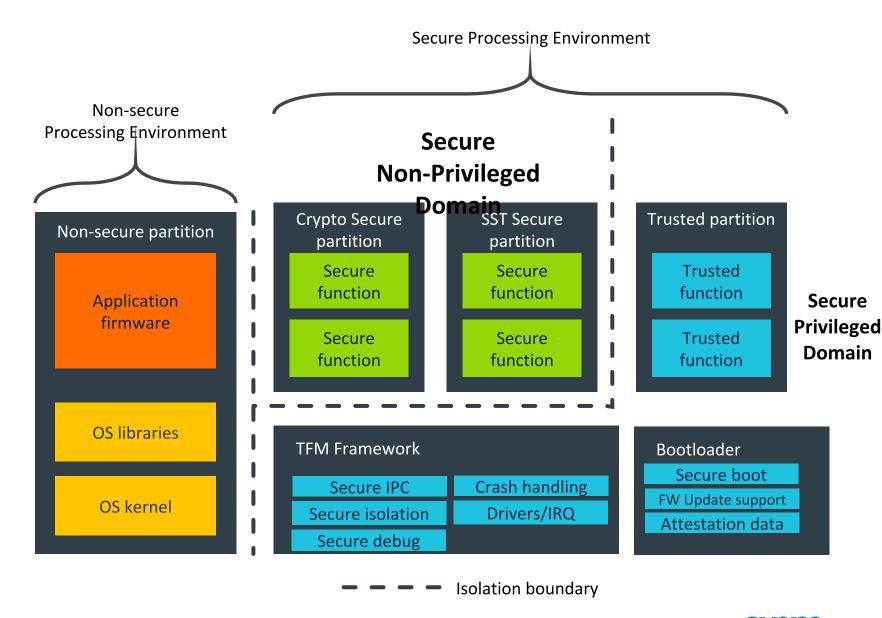






TF-M Overview

- Open source project
- PSA implementation for M class
- Secure Boot
- Secure Partition Management (SPM)
- Secure function call routing
- Isolation within SPE
- Secure Services
- NSPE API
- CMake based Build environment
- Test suite
- Documentation
- Infrastructure





TF-M - The Big Picture Secure Processing Environment Non-Secure Processing **∉**nvironment **Audit** Vendor Crypto Secure Attestat **Provisio** Secure applets **Service Storage** Log ion ning Non-Privile service service ged **Domain Application** firmware Bootl Framework **TFM** Platform Initial Secure oade and SPM **Service API** Crypto **APIs** attestation **Privileged** service **Domain OS** libraries **TBSA HAL API**

NV Counter

SAU/MPC/PPC



Hardware

OS kernel

HW

Keys

Crypto

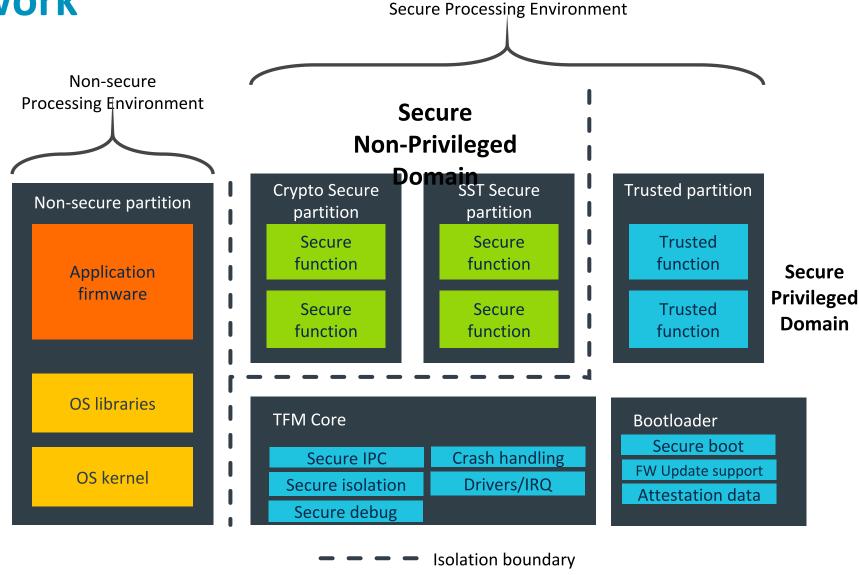
Accelerator

Secure Partition Manager and IPC



TF-M Core Framework

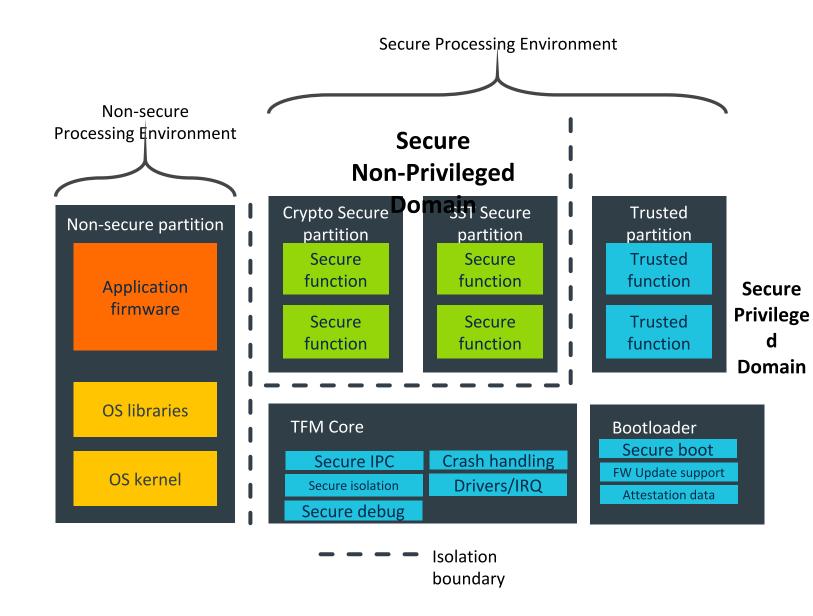
- Secure system init
- Secure Partition Management (SPM)
- Secure function call routing (IPC)
- Isolation within SPE
- NSPE API
- Build environment
- Test suite
- ...





Framework Updates

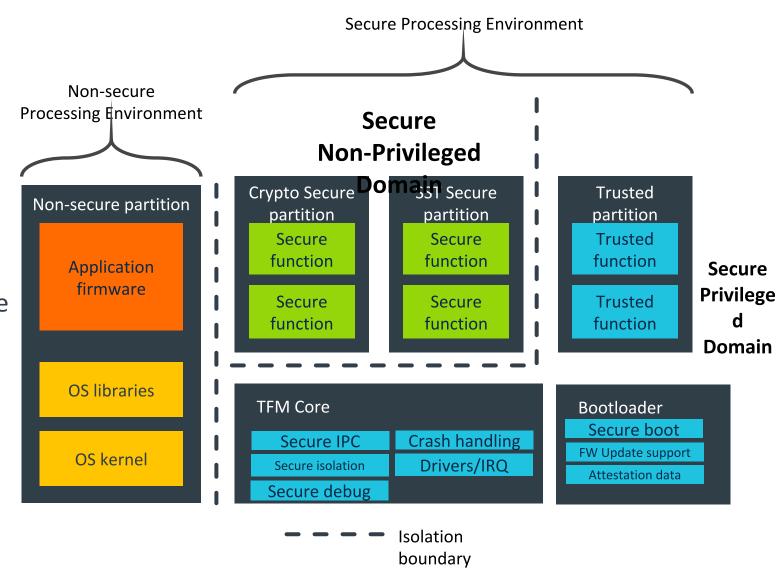
- NS to S transition in thread mode
- Manifest based service definition
- IPC support (on a feature branch currently)
- Support for Trusted and Secure partitions
- GCC support
- Platform abstraction improvements
- ARMv8-m mainline and baseline supported





IPC

- Facilitates secure communication between-
 - SPE Services
 - NSPE to SPE services
- Services are written as daemons running in a while loop
- Calls from clients are sent as a message to partition
- Synchronous execution of Service's interrupt handling
- Blocking API calls



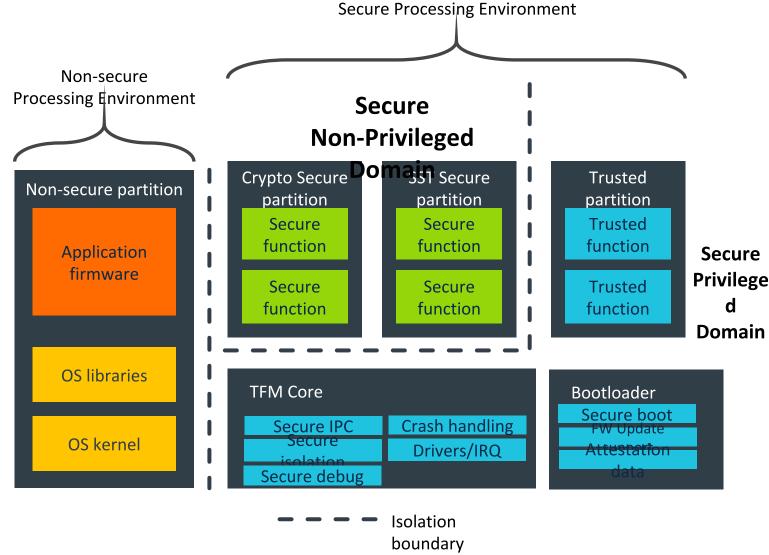


IPC Security Considerations

IOVEC based communication

```
typedef struct psa_iovec {
  const void *base;
  size_t len;
} psa_invec;
```

- Integrity protection of iovecs
- Streamed buffer read/write
- Framework level memory bound check
- API based access to client memory
- MMIO regions per partition for peripheral usage isolation





Secure Services



Crypto Secure Service

Secure
Processing Environment

Non-Secure
Processing Environment

Non-secure partition

Application firmware

Crypto Service API

OS libraries

OS kernel

Crypto service

Crypto Service API implementation

- Implements user facing APIs
- Key invisible to caller, caller knows only a reference
- Available to NSPE as well as SPE entities

Secure
Non-Privile
ged
Domain

TFM Crypto Glue API

TFM 'Crypto Service'

- Glue APIs, Available only to a limited set of entities
- Key passed in plaintext
- Abstracts usage of HW keys
- Either a service, or custom interface into SPM

Secure Partition
Manager

Secure Privileged Domain

TBSA HAL API

HW Keys

Crypto Accelerator

Hardware



Secure
Processing Environment

Secure Storage SST **Access Policy Policy** Management database Non-Secure Cryptographic HUK Processing **€**nvironment derived key binding Secure Non-Privile File System (or proxy) ged Hash Cert Key **Domain Application TFM Crypto Glue API** firmware **TFM 'Crypto Service'** Secure Partition Glue APIs, Available only to a limited set of entities Secure **SST Service API** Key passed in plaintext **Privileged** Abstracts usage of HW keys **Domain** Either a service, or custom interface into SPM OS libraries **TBSA HAL API** OS kernel Hardware **HW Keys Crypto Accelerator**

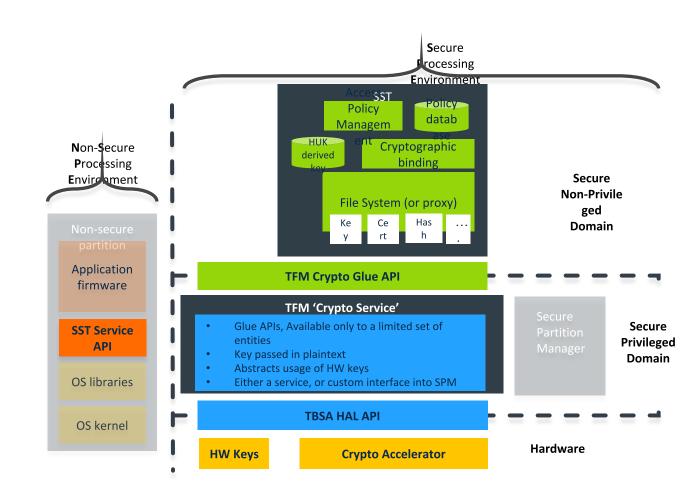


Secure Storage

- Protects Confidentiality, Integrity and availability of stored content
- Policy based access
- AES-GCM For AEAD
- Power failure safe operation
- Custom File System
- Rollback protection

New Stuff....

- PSA API, crypto property binding
- Rollback protection
- Key diversification (under review)
- File system abstraction (under review)





Audit Log

- Mitigation against repudiation
- Confidentiality/Integrity/Authenticity protested log of system's security critical events
- Use-case defined log entries
- Facilitates secure retrieval of logs by server



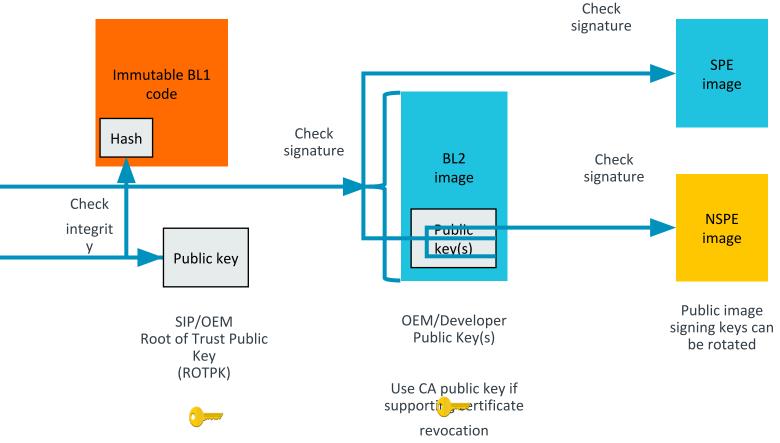
Bootloader



Bootloader

 SPE and NSPE image combined and signed

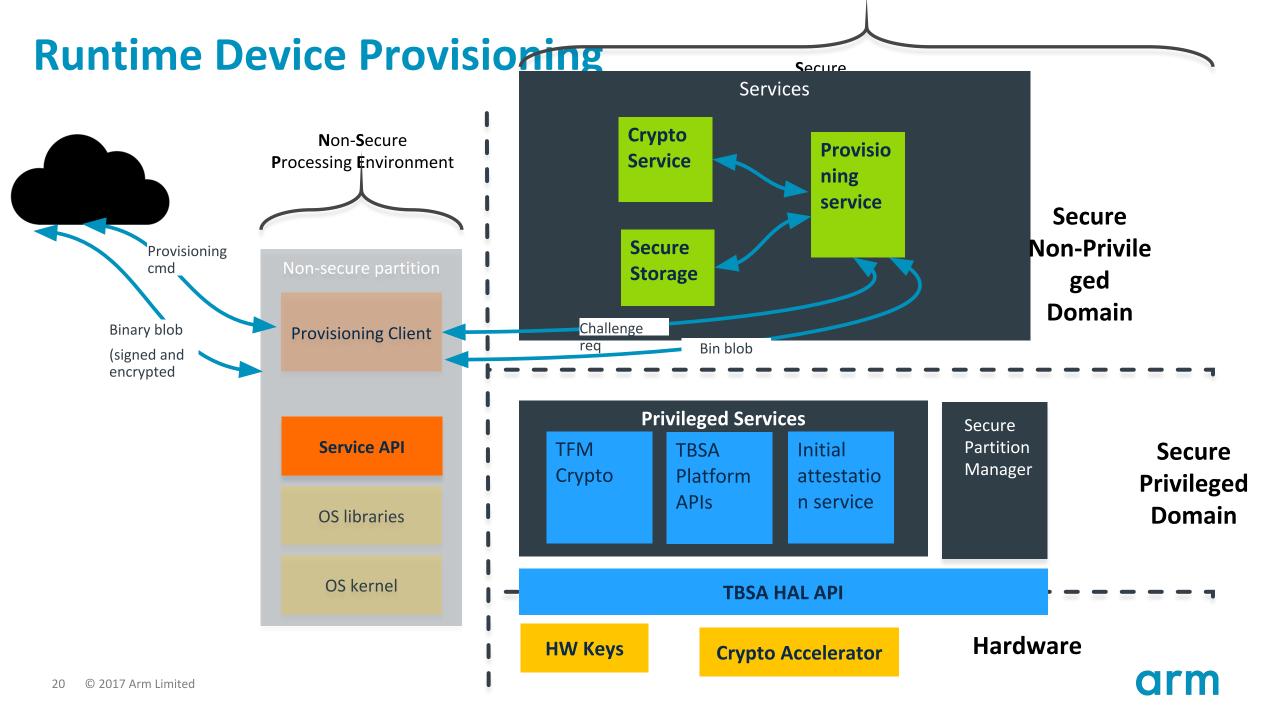
- SHA256, RSA-2048
- Firmware update support
- Attestation token collection
- Support for execution from RAM





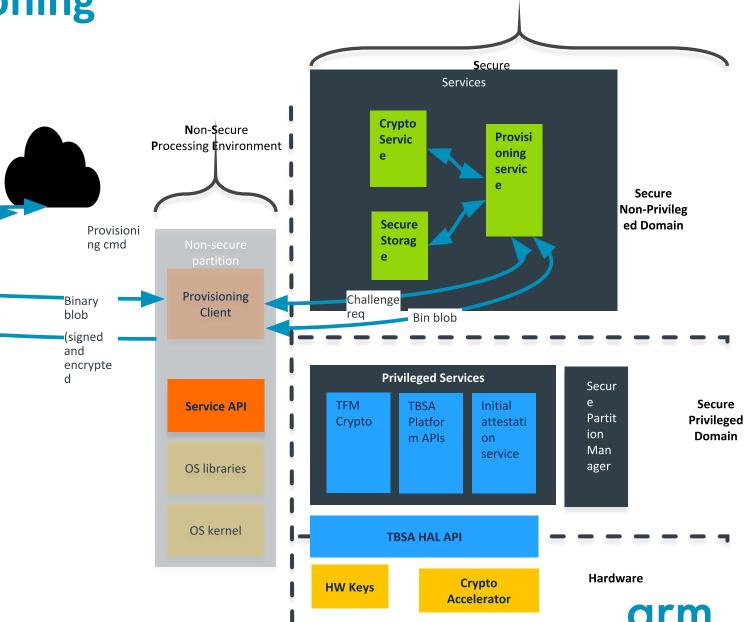
Usecase Scenarios

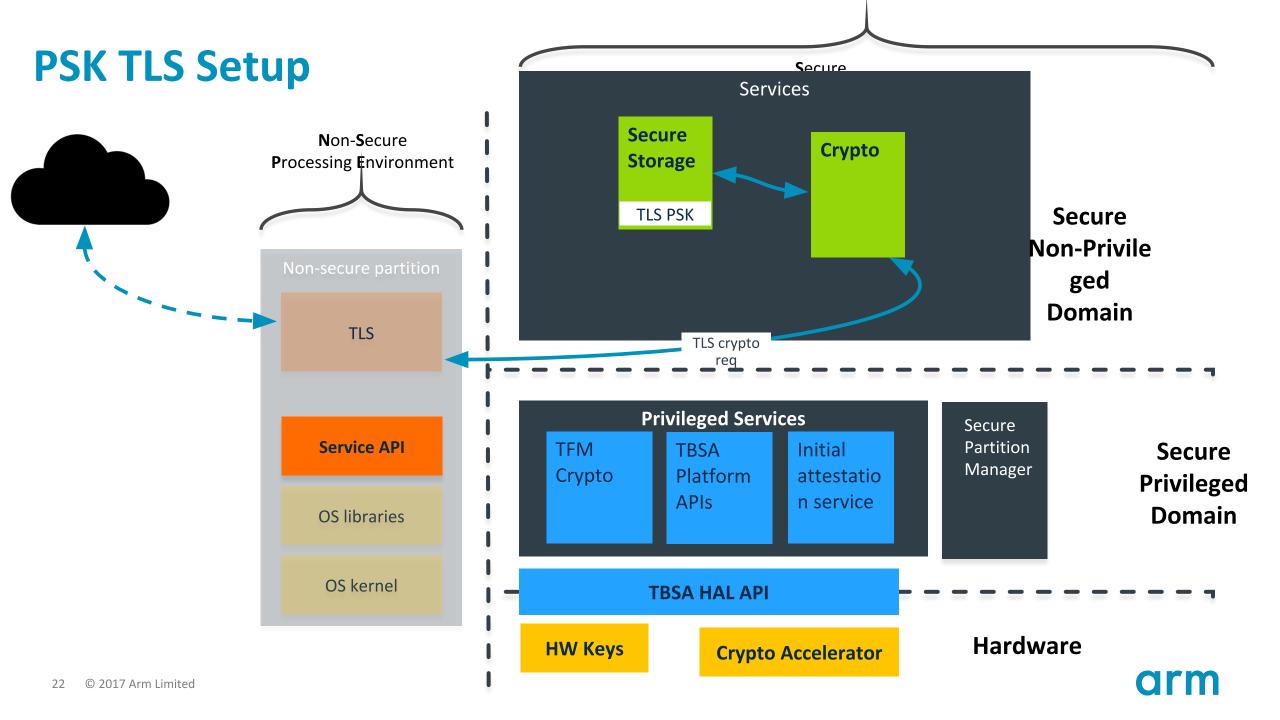




Runtime Device Provisioning

- Server sends provisioning cmd
- Provisioning client requests a challenge from provisioning service
- Server signs and encrypts the provisioning data(includes challenge)
- Provisioning service authenticates and decrypts the provisioning data
- Provisioning data is programmed in secure storage
- Flush crypto cache (key slots)



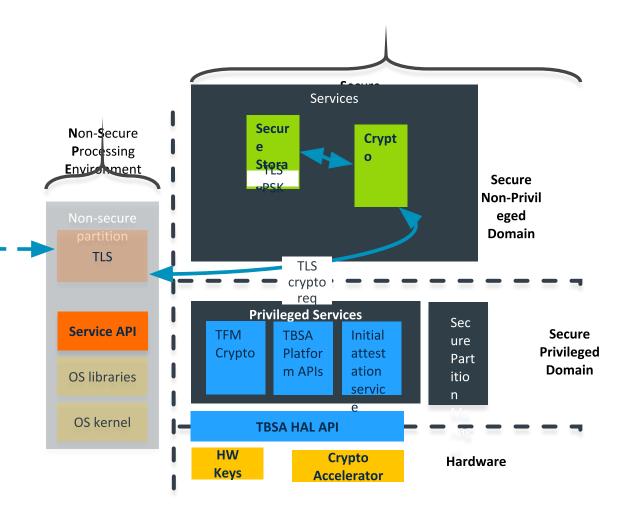


PSK TLS

 Server or device initiates the TLS protocol

 TLS lib on device requests Crypto for enc/dec/hash — — — — — —

- Crypto fetches the TLS key from secure storage
- Crypto performs the requested cryptographic operation
- Result is returned back to NSPE world
- Key never leaves secure world

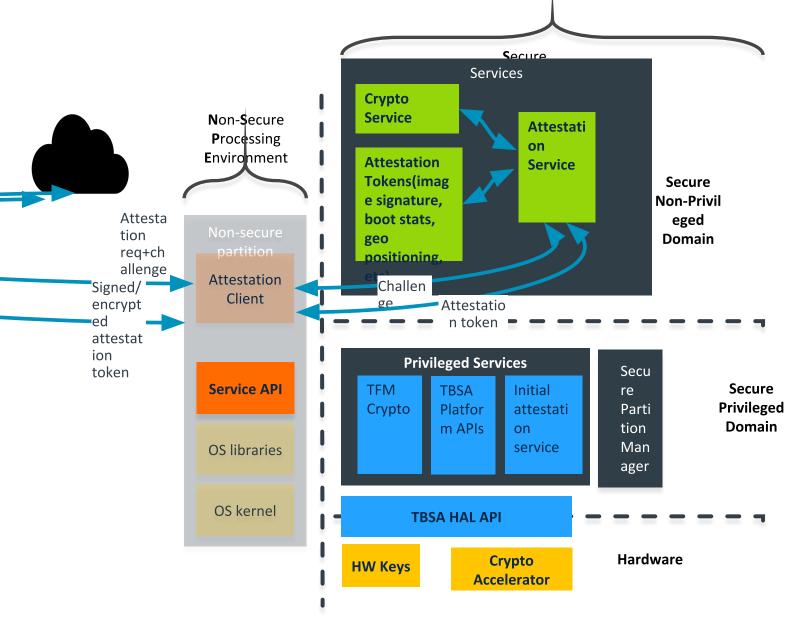




Attestation Secure Services **Crypto Service** Non-Secure **Attestation** Processing **E**nvironment **Service Attestation** Secure Tokens(image signature, boot Non-Privile Attestation stats, geo req+challen ged positioning, etc) **Domain** Challenge Signed/encry **Attestation Client** pted Attestation token attestation token **Privileged Services** Secure **Service API** Partition **TFM TBSA** Secure Manager Platform Crypto **Privileged APIs Domain** OS libraries OS kernel **TBSA HAL API** Hardware **HW Keys Crypto Accelerator**

Attestation

- Service to securely collect and provide the device measurements
 - Boot measurements
 - Image signature(s)
 - Device identity
 - Geographical location
 - Vendor data
- Server sends attestation request with a challenge
- Att service signs/encrypts the data alongwith challenge
- NSPE client returns the attestation blob to server





Getting Involved



How to get involved

Trusted Firmware Website

https://www.trustedfirmware.org/index.html

TF-M codebases

https://git.trustedfirmware.org/

TF-M Dev Team @ Connect HKG18

- Abhishek Pandit
- Ashutosh Singh

Get in touch

- Come round LITE hacking room
- Schedule a meeting via hkg18.pathable.com

More info on <u>developer.arm.com</u>



Looking Ahead

- Add support for IPC in all the services
- Secure interrupt handling
- TFM-M scheduler
- Crypto service enhancements to support asymmetric crypto
- Attestations Service
- Provisioning Service
- Platform HAL standardization
- Threat Modelling



Questions?

- BoF: IOT Security with Arm OSS 10:00 10:55, 18 September 2018
- Trusted Firmware Project Updates 16:00 16:55, 18 September 2018
- Open CI for Trusted Firmware 12:00 12:25, 18 September 2018



Thank You! Danke! Merci! 谢谢! ありがとう! **Gracias!** Kiitos! 감사합니다 धन्यवाद

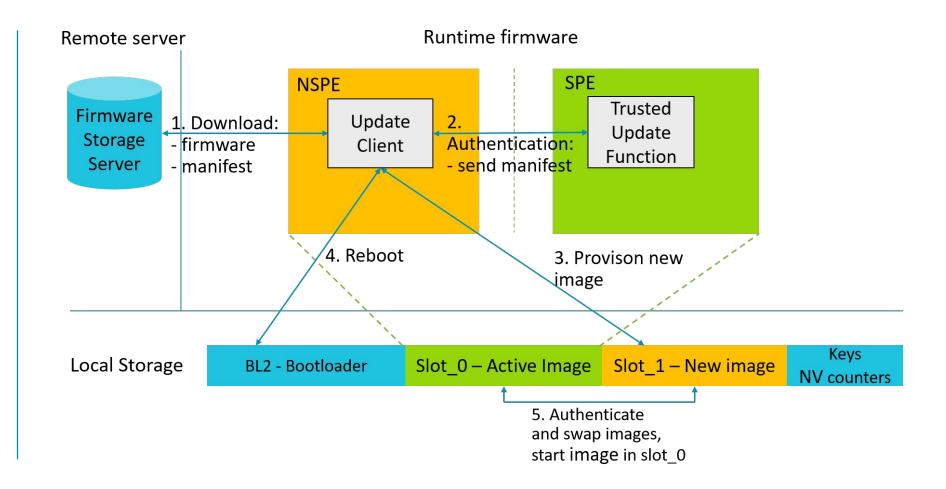


Backup Slides



Firmware upgrade

- Update client downloads the FW
- Bootloader validates the new image
- Swap the images
- Peform BIST in runtime and mark SPE and NSPE as 'safe'
- If not, revert

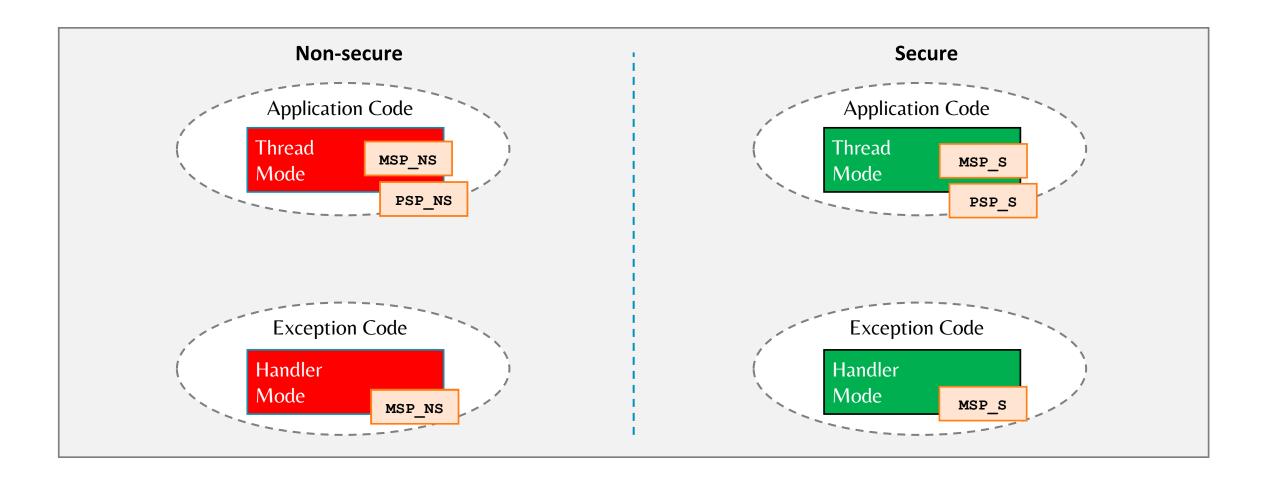




ARMv8-M TrustZone overview



ARMv8-M Secure and Non-secure states





ARMv8-M additional states

Existing Handler and Thread Modes mirrored with Secure and Non-secure States

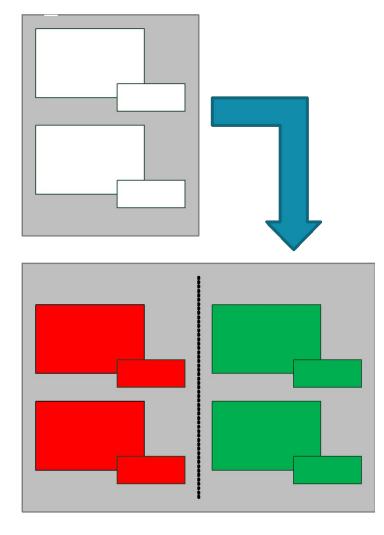
Secure and Non-Secure code run on a single CPU

Secure state for trusted code

- New Secure stack pointers for robust operation
 - MSP and PSP → MSP_NS, PSP_NS, MSP_S and PSP_S

Dedicated resources

- Separate memory protection units for S and NS
- Private SysTick timer for each state
- Secure side can configure target domain of interrupts





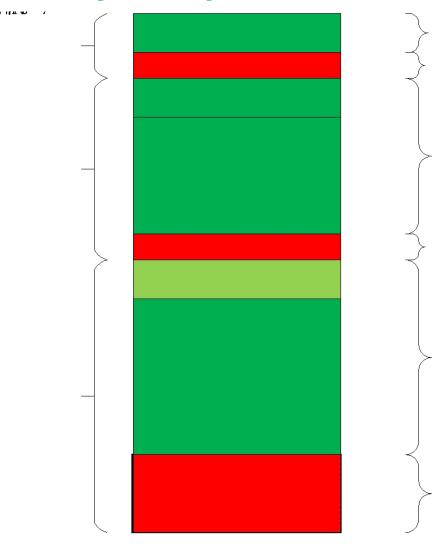
Secure Partitions: Address space layout/permissions

PSA level 1

SPE/NSPE isolation provided by v8M TrustZone (SAU, IDAU, MPC, PPC)

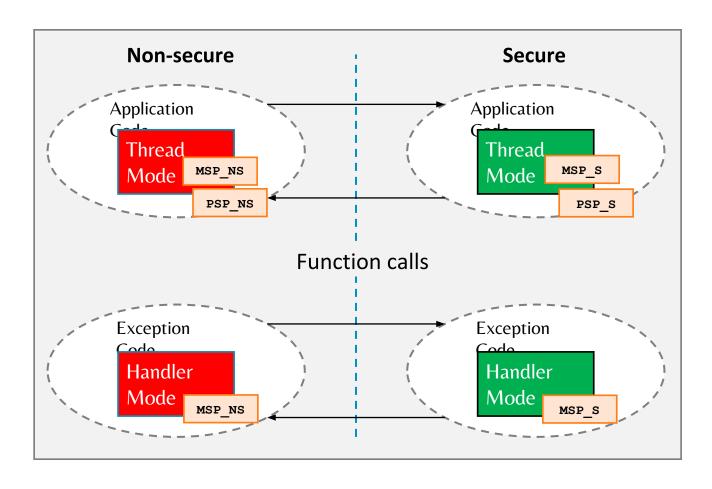
Partition Manager

- creates/maintains database of SPs
- sets up isolation boundaries
- prepares execution context for secure function
- keeps track of partition states





Calling between security states



Secure code can call Non-secure functions

Non-secure functions and data should not be trusted

Non-secure code can call into Secure libraries

- Only a sub-set of the Secure code is callable
- Secure entry points are limited
- Non-secure code does not need to know it is calling a Secure function

This is different from Armv8-A TrustZone

 Where changing security state can only occur on an exception boundary

Memory security

Physical memory is split into Secure and Non-secure regions (No MMU in M-class)

A Secure region can also be Non-Secure Callable (NSC)

