Trusted Firmware M
Core and Partition Manager

Miklos Balint
ARM
Agenda

Trusted Firmware M overview

TrustZone for ARMv8-M

TF-M Core and Partition Manager details

Summary
Trusted Firmware M overview
Introduction

Platform Security architecture (PSA) public launch in late 2017
TFM engineering work started early 2017, PSA – even earlier
PSA : set of specs & implementation – which is TF-M
PSA & Trusted Firmware M

A simplified comparison of PSA and TF-M

**PSA**
Wide architectural definition addressing security requirements on a range of platforms

- Platform-independent definition
- Focuses on key security requirements for both hardware and software
- No implementation details
- Multiple Isolation profiles

**Trusted Firmware M**
Reference implementation of PSA for M-Class CPUs.

- Covers PSA specification requirements
- Focusing on details to provide usable software implementation
- First version addresses PSA Level 1 on ARMv8-M mainline (Cortex M33) and baseline (Cortex M23)
- Incremental support for higher isolation profiles
- Long term plan to address other cores
TF-M Framework

- Secure bootloader
- Secure system init
- Secure Partition Management (SPM)
- Secure function call routing
- Isolation within SPE
- Trusted services, functions
- NSPE API
- Build environment
- Test suite
- ...

Non-secure Processing Environment (NS binary)

- Non-secure Partition
  - Application firmware
  - OS libraries
  - OS kernel

Secure Processing Environment (S binary)

- Secure Partition
  - Secure function
- Secure Partition
  - Partition-private code
- Secure Partition
  - Secure function
- Trusted Partition
  - Trusted Function

TF-M Core

- Secure call API and routing
- Partition Manager
  - Scheduling
  - Secure isolation
- Crash handling
- Secure Drivers
- Secure IRQ
- Secure Debug

Isolation boundary
TF-M v0.1 design choices

Primary goals, constraints set for first prototype:

- **SPE is non-reentrant**, i.e. one secure function is being serviced at a time
- Initially **NSPE (including IRQ) de-prioritized** during SPE execution, i.e. SPE cannot be preempted
- A secure partition is a library of secure functions, not a standalone execution thread
- **No isolation within SPE** (equivalent to PSA level 1)
- **NS OS agnostic**
- Focus on **v8M-based** solutions (SSE-200 sub-system)
- Provide working **solution**:
  - **Secure service**: Secure STorage (SST) with NSPE API
  - **BL2 bootloader** (based on MCUBoot)
ARMv8-M TrustZone overview
Introduction to TrustZone for Armv8-M

Armv8-M architecture includes optional Security Extension

- Branded as Arm TrustZone for Armv8-M

Similar in concept to TrustZone for Armv8-A

- Implementation is optimized for microcontrollers

System may be partitioned between secure and non-secure software

Secure software is highly trusted

- Has access to more system resources
- Protected from access by non-trusted code

To protect the secure software the security extensions provide:

- Isolated Secure memory for code and data
- Secure execution state to run Secure code
ARMv8-M Security Extension

Provides two security domains for code to run in

- Secure and Non-secure
- PEs without the Security Extension behave as though reset into Non-secure

Hardware assists in hiding Secure state from Non-secure code / debuggers

- Debugger can be blocked from accessing PE while Secure code is running
- Hardware pushes and clears registers if non-secure code interrupts secure code
- Stack limit registers added to assist in attack mitigation

Duplicate resources to enable software and hardware isolation

- For example, dedicated stack pointers, SysTick timers and MPUs for each domain
- Non-secure code only able to access non-secure controls

 Ability to expose PE’s security to system

- ARM’s AXI and AMBA5 AHB5 support propagation of NS attribute
ARMv7-M states

Application Code
- Thread Mode
  - MSP
  - PSP

Exception Code
- Handler Mode
  - MSP
ARMv8-M Secure and Non-secure states

**Non-secure**
- Application Code:
  - Thread Mode: MSP_NS
  - PSP_NS
- Exception Code:
  - Handler Mode: MSP_NS

**Secure**
- Application Code:
  - Thread Mode: MSP_S
  - PSP_S
- Exception Code:
  - Handler Mode: MSP_S
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
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)

---

Non-secure code

Secure, Non-secure callable (NSC)

Secure API

Secure Library

Branch

Branch

Return
Simple security attribution arrangement

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>TYPE</th>
<th>SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFF000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0xF0000000</td>
<td>SYSTEM</td>
<td>Secure</td>
</tr>
<tr>
<td>0xE0000000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0xD0000000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0xC0000000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0xB0000000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0xA0000000</td>
<td>DEVICE</td>
<td>Secure</td>
</tr>
<tr>
<td>0x90000000</td>
<td>RAM(WB)</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x80000000</td>
<td>RAM (WT)</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x70000000</td>
<td>DEVICE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x60000000</td>
<td>DEVICE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x50000000</td>
<td>SRAM</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x40000000</td>
<td>CODE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x30000000</td>
<td>CODE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x20000000</td>
<td>CODE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x10000000</td>
<td>CODE</td>
<td>Secure + NSC</td>
</tr>
<tr>
<td>0x00000000</td>
<td>CODE</td>
<td>Secure + NSC</td>
</tr>
</tbody>
</table>

Attribute options

- Secure
  - Non-secure Callable
- Non-secure

IDAU and SAU: Strictest applies

IDAU

NSC

Secure override

Secure + NSC

Secure + NSC

Secure + NSC

Secure + NSC
Security extends to the whole system

- **IDAU**
  - ARMv8-M Processor
  - TrustZone aware bus master

- **Secure Boot loader**
  - Secure access only

- **Secure Boot loader**
  - Secure access only

- **Memory Protection Controller**
  - Memory Protection Controller
  - Memory Protection Controller

- **Flash**
  - (Page based partitioning)

- **SRAM**
  - (Watermark level based partitioning)

- **System Security Controller**
  - Security wrapper

- **Legacy bus master**
  - (Non-Secure)
  - Legacy bus master

- **Legacy bus master**
  - (Secure)
  - Legacy bus master

- **Peripheral Protection Controller**
  - Security wrapper

- **AHB5 to APB bridge**
  - AHB5 to APB bridge

- **AHB5 Peripheral Protection Controller**
  - APB Peripheral Protection Controller

- **AHB Peripheral Protection Controller**
  - APB Peripheral Protection Controller

- **AMBA 5 AHB5 interconnect**

**Legend:**
- **Secure regions**
- **Non-secure regions**
- **System IP**
High Level System View – Ref Hardware

CoreLink SSE-200 subsystem

- Secure Debug
- Cortex-M33
- Instruction Cache
- IDAU
- Cortex-M33
- Instruction Cache
- IDAU
- Local SRAM
- Always-on domain
- Power Control
- CoreLink SIE-200 IP
- Subsystem IP
- Other ARM IP
- Non-ARM IP

AHB5 expansion ports
Master/Slave

- DMA
- HW acceleration
- Other radios
- Peripherals
- ADC/DACs
- Interfaces (SPI, I²C, SDIO,...)
- ...

Options

Secure Debug
Cortex-M33
Instruction Cache
IDAU
Cortex-M33
Instruction Cache
IDAU
Local SRAM
Always-on domain
Power Control
CoreLink SIE-200 IP
Subsystem IP
Other ARM IP
Non-ARM IP

AHB5 code interface
TrustZone Filters
Flash Controller
Embedded Flash or External Flash
SRAM Ctrl
System SRAM
TrustZone Cryptocell
Cordio Radio (digital part)
APB Bridge
APB Peripherals
Cordio RF

Options
TF-M Core and Partition Manager
TF-M Core features

- Secure system init
- Secure function call routing
- Secure Partition Management (SPM)
- Enforcing isolation
- Scheduling
- Crash handling
- Secure IRQ and drivers
- Secure Debug
- ...

Non-secure Processing Environment (NS binary)
- Non-secure Partition
  - Application firmware
  - OS libraries
  - OS kernel

Secure Processing Environment (S binary)
- Secure Partition
  - Secure function
- Secure Partition
  - Secure function
  - Partition-private code
- Secure Partition
  - Secure function
- Trusted Partition
  - Trusted Function
  - Trusted Function

TF-M Core
- Secure call API and routing
- Partition Manager
  - Scheduling
  - Secure isolation
- Crash handling
- Secure Drivers
- Secure IRQ
- Secure Debug
TF-M call routing: An Example Use-case
TF-M call routing: An Example Use-case

NS unprivileged thread
- NS thread
  - Crypto service request

NS privileged handler
- NS OS / TF-M lib
  - blocking NS lock

S privileged handler
- Secure Request handler
  - Non-blocking lock
  - Save NSPE context
  - Setup SP context
  - Disable NSPE

S (un)privileged thread
- Crypto Secure Partition
  - Perform crypto operation

NS thread
- NS privileged handler
  - NS OS / TF-M lib
  - unlock NS lock

S privileged handler
- Secure Request handler
  - Save SP context
  - Restore NSPE context
  - Unlock S lock
  - Enable NSPE

S thread
- Crypto Secure Partition
  - Fill return buffer

NS unprivileged thread
- NS thread
  - Continue execution

NS thread
- NS OS / TF-M lib
  - unlock NS lock
Simple security attribution arrangement

Attribute options
- Secure
  - Non-secure Callable
- Non-secure

IDAU and SAU: Strictest applies

Secure + NSC
Secure + NSC
Secure + NSC
Secure + NSC

IDAU

NSC

SAU override
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
Partition Manager: PSA isolation levels

Level 1
Lower cost hardware – only isolate the SPE

Level 2
Separate Root of Trust from Secure Partitions within SPE

Level 3
More robustness – isolate all partitions from each other
Partitions: Address space layout/permissions

PSA level 1 vs. level 3

SPE/NSPE (PSA level 1) isolation provided by v8M TrustZone (SAU, IDAU, MPC, PPC)

PSA level 2+ isolation being investigated

- Utilize TrustZone/MPU combination
- Higher resource demand
- Permission types
  - Privileged: TF-M internals, trusted
  - Private unprivileged: Secure Partition resources, not to be shared (in level 3)
  - Shared unprivileged: accessible to all secure code
TF-M Core outlook

Ongoing improvements, investigations

- SPE **preemption** by NSPE
- Secure IRQ handling
- Data buffer sharing / client-SP **API improvements**
- Closer **PSA alignment**
  - **Isolation** within SPE (PSA level 2+)
  - Structure **Secure Partitions as threads**
Summary
TF-M First Release

• Level 1 isolation
• framework for integrating secure services
• Secondary stage bootloader
• Secure storage service
• cmake build system, cross environment compilation (Ubuntu, Windows)
• Documentation – includes quick starter guide, Rich OS integration guidelines
• (Internal) RTX/MbedOS/FreeRTOS integrated
• Various demos to showcase real use-cases
• Partner engagement
• Available to the public
Initial target support
Cortex M33/M23 based systems

FPGA on MPS2 board

FVP (Fast Model) of MPS2 system
**Trusted Firmware M - Plans**

1. **Align with the PSA specifications by the time they make public release**
   1. Standardised APIs for crypto, attestation, secure storage, IPC, hardware RoT, audit logging, debug control, etc
   2. Full SPM and IPC to isolate many secure functions and their interactions
   3. Device initialisation, Trusted boot and firmware update
   4. Many build configurations to support from most constrained to most secure

2. **Support Arm development platforms and IP**
   1. Musca test-chip, SDK-20x FPGA on MPS2/3, AEMv8-M/IoT-kit FVP
   2. V8-M system IP, Arm Cryptocell, Arm Cryptolisland

3. **Support and enable contributions**
   1. Partner SoCs and systems
   2. Software integrations – secure services, RTOSes, secure-Oses

All in public open source project
How to get involved

TF-A and TF-M master codebases

• [https://git.trustedfirmware.org/](https://git.trustedfirmware.org/)

TF-M Team @ Connect HKG18

• Abhishek Pandit
• Ashutosh Singh
• Tamas Ban
• Miklos Balint

Get in touch

• Come round LITE hacking room between 3-4 pm Wednesday
• Schedule a meeting via [hkg18.pathable.com](http://hkg18.pathable.com)

More info on [developer.arm.com](http://developer.arm.com)
Thank You!
Danke!
Merci!
谢谢!
ありがとうございます!
Gracias!
Kiitos!
감사합니다
धन्यवाद
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