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
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

S thread mode

* Wrapper code

Secure veneer (NS Client ctx)

- Call Secure Request SVC

S handler mode

* Context management

Secure Request SVC

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

S unprivileged thread

* Sandboxed context

Secure Service function

- Perform secure service

NS thread

- Continue execution

Secure veneer (NS Client ctx)

- Return to NS

Secure Service function

- Call Response handler

Secure Response SVC

- Save SP context
- Restore NS Client context
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

Diagram:
- IRQ priority
- Background code
- Hardware stacking / unstacking
- ISR wrapper
- SVC
- ISR body
- SVC
- ISR wrapper
Secure interrupt deprivileging

Original mode
Original context

Interrupted code
• Gets interrupted

S handler mode
Wrapper code

Privileged ISR
• Call IRQ request SVC

S handler mode
Context management

IRQ Request SVC
• Set up MPU sandbox
• Switch PSP
• Ret. to unpriv. thread

S unprivileged thread
Sandboxed context

Secure Partition ISR
• Handle interrupt

Interrupted code

Privileged ISR
• Return to original state

IRQ Done SVC
• Restore MPU config, PSP
• Return to priv. ISR

Secure Partition ISR
• Call IRQ Done SVC

Original mode
Original context

Interrupted code
• Gets interrupted

S handler mode
Wrapper code

Privileged ISR
• Call IRQ request SVC

S handler mode
Context management

IRQ Request SVC
• Set up MPU sandbox
• Switch PSP
• Ret. to unpriv. thread

S unprivileged thread
Sandboxed context

Secure Partition ISR
• Handle interrupt

Interrupted code

Privileged ISR
• Return to original state

IRQ Done SVC
• Restore MPU config, PSP
• Return to priv. ISR

Secure Partition ISR
• Call IRQ Done SVC

Original mode
Original context

Interrupted code
• Gets interrupted

S handler mode
Wrapper code

Privileged ISR
• Call IRQ request SVC

S handler mode
Context management

IRQ Request SVC
• Set up MPU sandbox
• Switch PSP
• Ret. to unpriv. thread

S unprivileged thread
Sandboxed context

Secure Partition ISR
• Handle interrupt

Interrupted code

Privileged ISR
• Return to original state

IRQ Done SVC
• Restore MPU config, PSP
• Return to priv. ISR

Secure Partition ISR
• Call IRQ Done SVC

Original mode
Original context

Interrupted code
• Gets interrupted

S handler mode
Wrapper code

Privileged ISR
• Call IRQ request SVC

S handler mode
Context management

IRQ Request SVC
• Set up MPU sandbox
• Switch PSP
• Ret. to unpriv. thread

S unprivileged thread
Sandboxed context

Secure Partition ISR
• Handle interrupt

Interrupted code

Privileged ISR
• Return to original state

IRQ Done SVC
• Restore MPU config, PSP
• Return to priv. ISR

Secure Partition 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\textsubscript{1} calls Secure Service\textsubscript{1}
3. Non-secure IRQ pre-empts operation -> context change
4. Thread\textsubscript{2} calls secure service\textsubscript{2}
5. Secure service\textsubscript{2} returns
6. Thread\textsubscript{2} yields
7. Secure Service\textsubscript{1} returns

NS RTOS -> SPM notifications:
Thread creation, deletion, load or store
Enables NS context-dependent access to secure assets/services
Implementations
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
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

Diagram:
- Non-secure PE
- Non-secure application
- OS kernel
- NS Partition Interface
- NS Client Context
- Secure Partition Thread
- Secure Partition Manager
- TF-M Core
Interaction in thread model
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

Non-Secure Processing Environment

Non-Secure Application

Client API

Secure Gateway

Secure and Non-Secure Callable

NS Partition Interface

Secure Entry

Secure Service

Secure Partition #1

Secure Service

Secure Partition #2

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

Non-Secure Processing Environment

Secure Processing Environment

OS libraries

OS Kernel

SP_NonSecure

SP_Secure

Client API

Client and Service API

Message Manager

Scheduler

Secure Partition Manager (SPM)
Single NS Thread requests Secure Service

Non-Secure Processing Environment

- Non-Secure Partition
  - Non-Secure Application
    - Client API
  - Secure and Non-Secure Callable
    - Secure Gateway

Secure Processing Environment

- Secure Partition Interface
  - Secure Entry
  - Secure Service
- Secure Partition #1
  - Secure Service
- Secure Partition #2
  - Secure Service

Frame generated during client API calling

- OS libraries
  - Frame
  - SP_Secure
- OS Kernel
  - Client API
  - Client and Service API
  - Messages
  - Scheduler
  - Secure Partition Manager (SPM)
Multiple NS Thread request Secure Service

Non-Secure Processing Environment
- Non-Secure Partition
  - Non-Secure Application
    - Client API
  - Non-Secure Callable
- Secure and Non-Secure Callable

Frame generated during client API calling
- Frame #1
- Frame #2
- SP_Secure

Secure Processing Environment
- Secure Partition Interface
  - Secure Entry
  - Secure Gateway
  - Client API
- Secure Partition #1
  - Secure Service
- Secure Partition #2
  - Secure Service

OS libraries
- Frame #1
- Frame #2
- SP_Secure

Client and Service API
- Messages
- Scheduler
- Secure Partition Manager (SPM)
Multi-Thread NSPE Secure Call Solution 1

Non-Secure Processing Environment

- Non-Secure Partition
  - Non-Secure Application
    - Client API
  - Secure and Non-Secure Callable
    - Secure Gateway
- Secure Processing Environment
  - Secure Partition Interface
    - Secure Entry
  - Secure Partition #1
    - Secure Service
  - Secure Partition #2
    - Secure Service

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

- OS libraries
  - Frame #1
- OS Kernel
  - SP_Secure

Client API

- Client and Service API
  - Messages
  - Scheduler
  - Secure Partition Manager (SPM)
Multi-Thread NSPE Secure Call Solution 2

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

Non-Secure Processing Environment

- Non-Secure Application
- Non-Secure Callable
- Client API

Secure Processing Environment

- Secure and Non-Secure Callable
- Secure Entry
- Secure Context Management
- Secure Partition Manager (SPM)

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

- Secure Stack 1
- Secure Stack 2
- Secure Stack 3

Non-Secure Stack 1
Non-Secure Stack 2
Non-Secure Stack 3

NS Scheduler

sync thread status with SPM via Privileged API

OS libraries
OS Kernel

Secure Gateway

Secure Gateway
Solution 2 Calling Process

Non-Secure Processing Environment

OS libraries
OS Kernel

Non-Secure Partition
Thread 2
Client API

Secure and Non-Secure Callable

Secure Gateway

NS Partition Interface

Secure Processing Environment

Secure call enters into dedicated secure stack

Secure Stack 2
Secure Service

Secure Partition 
#1

Secure Partition 
#2

Secure Service

Secure Gateway

CMSIS TZ Secure Context Management

PSA Client

PSA Client and Service API

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.

Non-Secure Processing Environment

Non-Secure Partition

Non-Secure Application

Client API

OS Kernel

Secure Processing Environment

Secure and Non-Secure Callable

Secure Gateway

Runtime

Execution

Ex.

ISR

SP

Secure Execution

Client API

Client and Service API

Scheduler

Secure Partition Manager (SPM)
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
Thank You!
Danke!
Merci!
谢谢!
ありがとうございます!
Gracias!
Kiitos!
감사합니다
धन्यवाद