IoT Security and Privacy

IoT System Security including TrustZone

YIER JIN

UNIVERSITY OF FLORIDA

EMAIL: YIER.JIN@ECE.UFL.EDU

SLIDES ARE ADAPTED FROM PROF. XINWEN FU @ UCF/UMASS

Learning Outcomes

Upon completion of this unit:

- Students will be able to explain the system security
- Students will be able to explain TrustZone hardware architecture
- Students will be able to explain TrustZone software architectures

Prerequisites and Module Time

Prerequisites

- Students should have taken classes on operating system and computer architecture.
- Students must have taken crypto and know how public key crypto and symmetric key crypto work.
- Students should have mastered programming Raspberry Pi.
- Students should know basic concepts of networking.

Module time

- Two-hour lecture
- Two-hour homework

Main References

[1] ARM Security Technology Building a Secure System using TrustZone® Technology, 2009

[2] Secure the Windows 10 boot process, 06/23/2017

Outline

Introduction

System security

TrustZone Hardware Architecture

TrustZone Software Architecture

TrustZone System Design

What Is Security?

Asset – A worth protecting resource of value

- Tangible object, e.g. a user password
- Intangible asset, e.g. network availability

Attack

- *Intentional* act of acquisition, damage or disruption of an asset without permission
- Attack tools: software, hardware monitoring and hardware tampering

Defense

- Design of a system with hardware or software to counter attacks.

Limitations of Security Solutions

We can only counter a subset of possible attacks

- Impossible to counter all possible attacks

A security design must identify

- what assets to protect
- what attacks to counter

If committing an attacks needs too much money and time

- The defense is a success
- The attacker most possibly will move on to next target

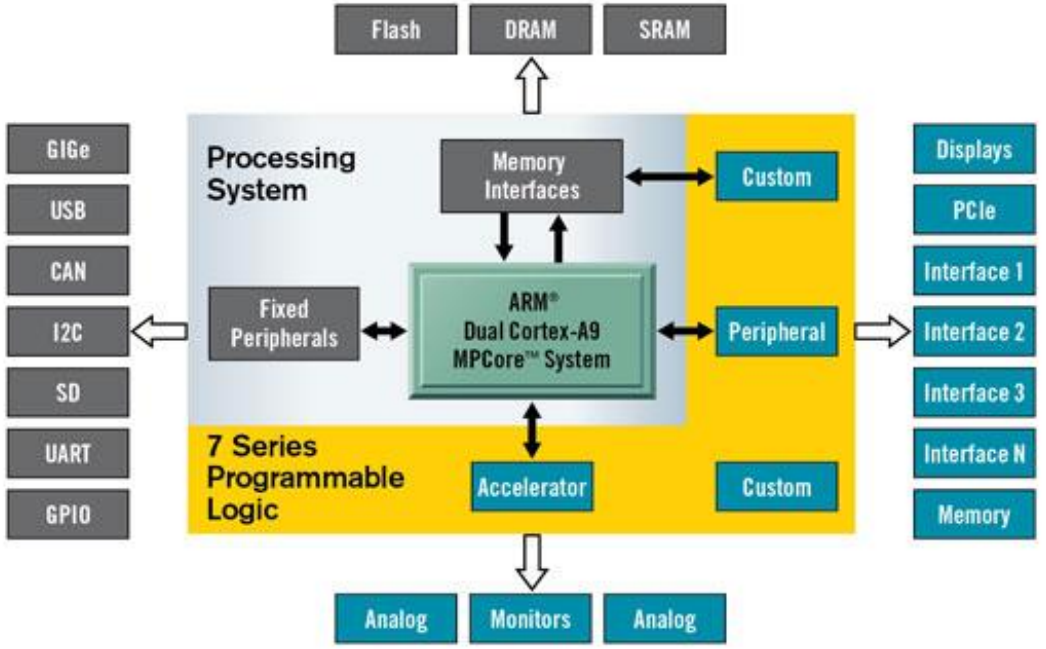
Hardware Enforced Security

Trusted Computing Group recommends

- Dedicated hardware from the start of system design
- Processor-level, SoC (System on a chip) infrastructure based

ARM TrustZone

- Uses ARM processor, bus fabric, and system peripheral IP
- Provides a framework and supports flexible secure system architectures with low cost



System on a chip

SoC Components

A microcontroller, microprocessor or digital signal processor (DSP) core – multiprocessor SoCs (MPSoC)

Memory blocks including a selection of ROM, RAM, EEPROM and flash memory

Timing sources including oscillators and phase-locked loops

Peripherals including counter-timers, real-time timers and power-on reset generators

External interfaces, including industry standards such as USB, FireWire, Ethernet, USART, SPI analog interfaces including ADCs and DACs

Voltage regulators and power management circuits

A bus that connects these blocks above

DMA controllers route data directly between external interfaces and memory

Threats against Market Sector - Mobile Sector

International Mobile Equipment Identity (IMEI) code

- Identify a handset
- Block a stolen one using the cellular network, when reported

Low level SIMLock protocol

- Bind a *handset* to a *SIM* from a service operator during a mobile phone contract
- Can be bypassed through a USB cable and a reprogramming tool running on a desktop. Industry losing big money because of this

Data on mobile devices needs security

- Digital Rights Management (DRM)
- Confidential user data like synchronized email accounts?

Threats against Market Sector - Consumer Electronics and Embedded Sector

Consumer electronics

Embedded sector

Odometer fraud – rollback of odometer of a second-hand vehicle

Security Features for Embedded Systems

Secure firmware updates

Secure debug mechanisms

Can be implemented with TrustZone

Economic Value in Security Issues

Risk analysis

- Probability of a successful attack
- Cost to the business for a successful attack
- Cost of defense

Example risk analysis outcomes

- *Probability of an attack too low to be worth defending*
- Cost of a defense too high for the asset of interest
- Justified asset defense

TrustZone is cost efficient for various asset defense

“Professional hackers are predominantly motivated by financial gain.”

- Really?

Class-breaking Attacks

Who are the attackers?

- Professional criminals (for financial gain)
- Academic security researchers (for academic prestige)
- Enthusiasts working at home (for fun)

Most sought achievement - class-break: attacks that break a whole generation, or class, of devices.

- E.g. , breaking software restrictions on games consoles and the content protection schemes on DVD movies

Investment for class break attacks

- Initial investment could be big
- E.g. electron microscopes and transistor stimulating lasers for silicon-level analysis

Positive Economics

A good payment system stimulates spending by 20%

- Easy to use
- Secure
- Increased consumer confidence
- New revenue streams
- Different business models.

Security features differentiate manufacturers

How Are Devices Attacked?

Hack attack – software attack by hackers

- Viruses and malware
- Why possible? Human factors --- "Given a choice between dancing pigs and security, users will pick dancing pigs every time." [1] ☺

Shack attack - low-budget hardware attack from cheap off-the-shelf tools bough from for example Radio Shack

- *Hardware interface attack*, not attack against integrated circuit packages.
- Examples: JTAG debug, boundary scan I/O, and built-in self test facilities.
- Attacks: forcing pins and bus lines to be at a high or low voltage, reprogramming memory devices, and replacing hardware components with malicious alternatives.

How Are Devices Attacked? (Cont'd)

Lab attack – unlimited reverse engineering with sophisticated equipment such as electron microscopes

- Most comprehensive and invasive
- Rule of thumb: every device can be broken
- The defense is to make the lab attack uneconomic – for example, use of per-device unique secrets, so that the attack is limited to one compromised device

Who Attacks Devices?

Lab attacks are outside of the scope of the protection provided by TrustZone technology

Remote attacker

- Increased complex software on embedded devices
- Installation of code from the Internet

Security specialist

- Criminal gangs, security experts, and enthusiasts attacking devices for fun

“Trusted” developer – insider attack

- Can be mitigated by defensive measures in business process

Device owner

- Gaining free access to services and content.
- Often script kiddies using approaches from the Internet
- Exposed to embedded malware since they download tools from the Internet

Outline

Introduction

System security

TrustZone Hardware Architecture

TrustZone Software Architecture

TrustZone System Design

Embedded Device Design

Functional components (for the target application)

- Multiple independent processor cores
- Secondary bus masters such as DMA engines
- large numbers of memory
- Peripheral bus slaves

Non-function components

- Invasive (via control) and non-invasive debug (via observation) capabilities
- Component boundary scan: a method for testing interconnects of a board
- Built-In-Self-Test (BIST) facilities

Security solution has to address all above

System Security Strategy - External Hardware Security Module

A dedicated hardware security module, or trusted element, outside of the main SoC

- A SIM card in a mobile handset
- A conditional access smartcard in a set-top box.

Advantages

- Work well for the assets it protects

Disadvantage of external hardware security module (e.g. smartcard)

- Cannot protect assets outside of the external hardware, e.g. pin/password

System Security Strategy - Internal Hardware Security Module

Two main forms

- *Hardware block* that manages cryptographic operations and key storage
- *General purpose processing engine* placed alongside the main processor

Advantages

- Cost effective compared to the use of many smartcards for different assets

Disadvantages

- Cryptographic hardware block has similar issues like smartcard
- A second security processor introduces complexities such as communication between the main processor and itself

Software Virtualization

A hypervisor

- Uses the Memory Management Unit (MMU)
- Runs independent software platforms inside a virtual machine

Advantages

- Flexible

Disadvantages

- The isolation is restricted to the processor implementing the hypervisor.
- Other bus masters such as DMA and GPUs can bypass the protection

TrustZone Hardware Security

Protection strategies discussed so far are dedicated to specific assets

TrustZone is for system-wide security

- Protects any part of the system
- Can be used with other security strategies such as secure boot, authenticated debug mode and unique secret for each device

Outline

Introduction

System security

TrustZone Hardware Architecture

TrustZone Software Architecture

TrustZone System Design

TrustZone Hardware Architecture

It constructs a programmable environment that protects almost any asset

Any SoC hardware and software resources exist in two worlds

- Secure world
- Normal (Non-secure) world

Hardware logic in the TrustZone-enabled AMBA3 AXI™ bus fabric ensures

- Normal world components cannot access secure world resources

A single ARM core of some version can execute code from both normal world and secure world in a time sliced fashion

- No need of dedicated secure processor

It has a security-aware debug infrastructure that limits secure world debug

System Architecture

AMBA3 AXI system bus

- An extra control signal, Non-Secure, or NS bits, for each of the read and write channels on the main system bus.

AMBA3 APB peripheral bus

- Secure peripherals, such as interrupt controllers, timers, and user I/O devices (e.g., a securable keyboard peripheral that protects a user password)
- Use of Advanced Peripheral Bus (APB) attached to the system bus through an AXI-to-APB bridge

System Architecture (Cont'd)

Memory aliasing

- The NS bit is like a 33rd address bit, a 32-bit physical address space for Secure transactions and a 32-bit physical address space for Non-secure transactions
- The same memory location appears as two distinct locations in Secure and Non-secure worlds – needs to be taken care

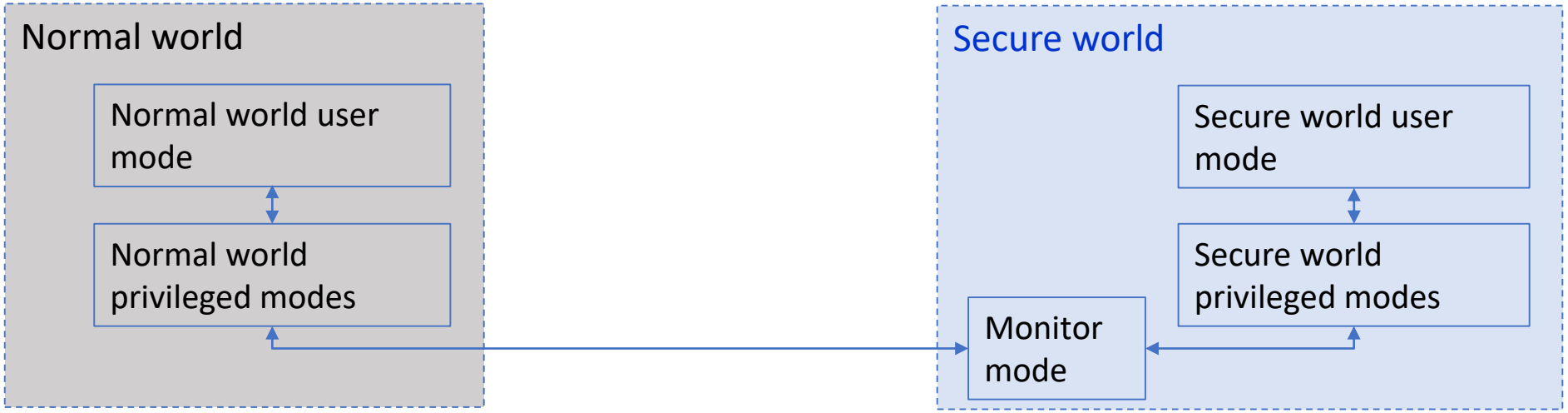
Processor Architecture

ARM processors with security extensions, including

- ARM1176JZ(F)-S™ processor
- Cortex™-A8 processor
- Cortex-A9 processor
- Cortex-A9 MPCore™ processor

Each physical processor core has two virtual cores

- Secure
- Non-secure
- **Monitor mode** as “context switch”



[1]

Switching Worlds

Monitor mode – a new core mode

Tightly controlled switching mechanisms

- Viewed as exceptions to the monitor mode software
- Triggered by the Secure Monitor Call (SMC) instruction by some hardware exception mechanisms

The monitor mode software is *implementation defined*

- Saves the state of the current world
- Restores the state of the world being switched to
- Performs a *return-from-exception* to restart processing in the restored world

NS-bit in the *Secure Configuration Register* (SCR) in CP15 (the system control coprocessor) indicates the world of the processor

- In monitor mode, the processor executes in the Secure world

Outline

Introduction

System security

TrustZone Hardware Architecture

TrustZone Software Architecture

TrustZone System Design

TrustZone Software Architecture

Security Extensions are an *open* component of the ARM architecture

Any developer can create a custom Secure world software environment to meet their requirements.

Software Architecture

The most complex is a dedicated Secure world operating system

The simplest is a synchronous library of code placed in the Secure world

Many options between the most complex and the simplest

Software Architecture (Cont'd)

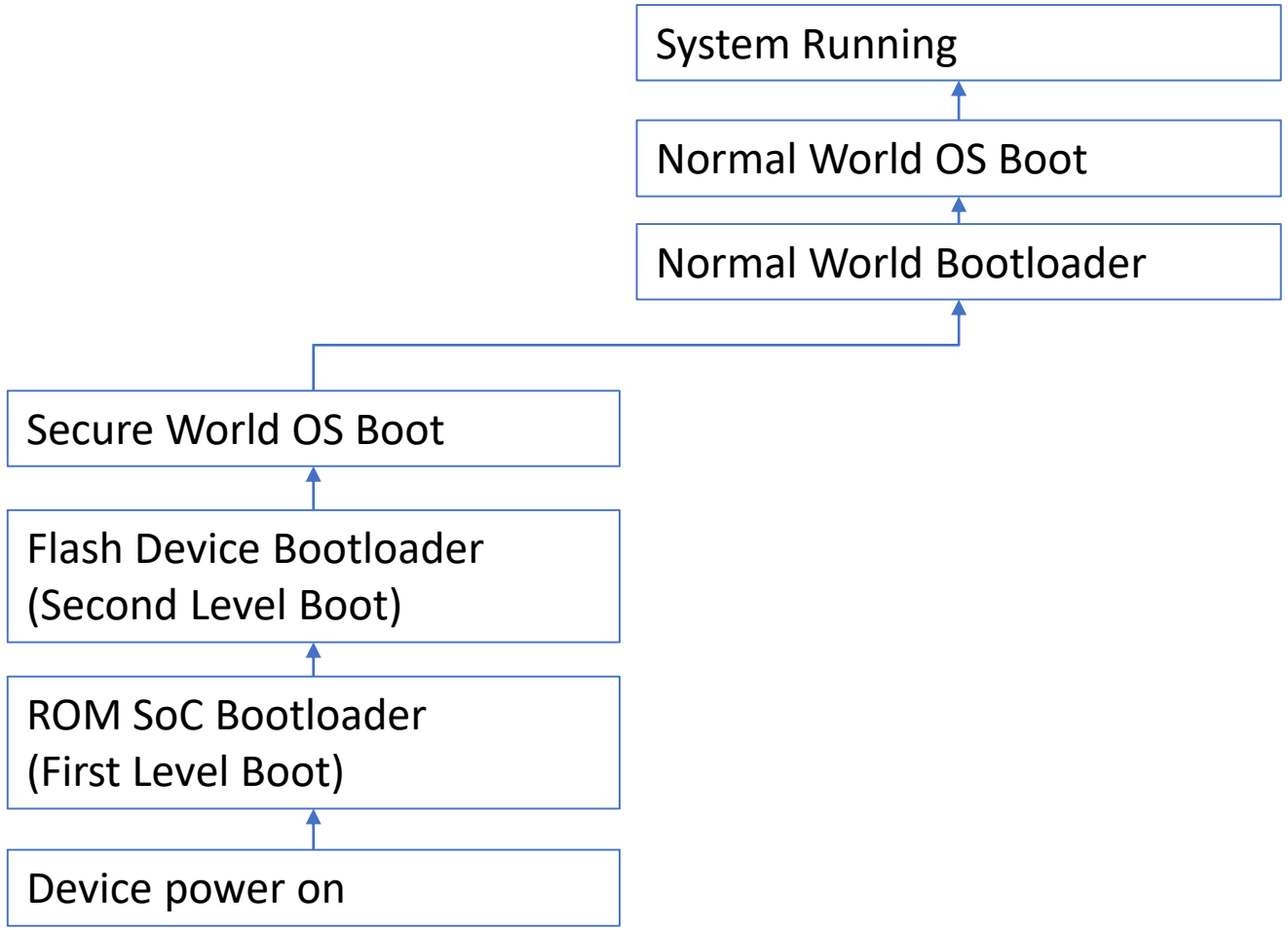
Secure operating system

- Can simulate concurrent execution of multiple independent Secure world applications
- run-time download of new security applications
- Secure world tasks completely independent of the Normal world environment.

Synchronous library

- Handle *one task at a time* is sufficient for many applications

Booting a Secure System [1]



Secure Boot

A trusted vendor

- Uses their Private Key (PrK) to sign the code of interest
- Pushes this to the device alongside the software binary.

The device contains the Public Key (PuK) of the vendor,

- Verify that the binary has not been modified and
- It was provided by the trusted vendor in question.

The PuK does not need to be kept confidential

- Should be stored within the device in such a manner that it cannot be replaced by a PuK of an attacker.

Chain of Trust

Chain of trust starts with the root of trust so that later applications can be authenticated before being executed.

A public key (PuK) belonging to the device OEM might be used to authenticate the first bootloader

- but the Secure world OS binary might include a secondary PuK that is used to authenticate the applications that it loads.

Storage of the PuK for the root of trust is a challenge

- Embedding it in the *on-SoC ROM?* The SoC ROM is the only component that cannot be trivially attacked.
- **On-SoC One-Time-Programmable (OTP) hardware**, such as poly-silicon fuses, can be used to store unique values in each SoC during **device manufacture**.

On-SoC Secure World or Off-SoC Secure World

The simplest defense against shack attacks is to keep any Secure world resource execution located in on-SoC memory locations.

- A physical attack on the SoC package is hard

The secure boot code is generally responsible for loading code into the on-SoC memory, how to authentication the code?

- Code or PuK to authenticate should be performed in secure world

Monitor Mode Software

A gatekeeper that manages the switches between the Secure World and Non-secure World.

Like a traditional operating system context switch

- Saves state of the world that the processor is leaving
- Restores the state of the world the processor is switching to
- Critical component to ensure the TrustZone mechanism

Best practice: disable interrupts in the monitor mode

- Complex with enabled interrupts
- Benefit is not big

TrustZone API

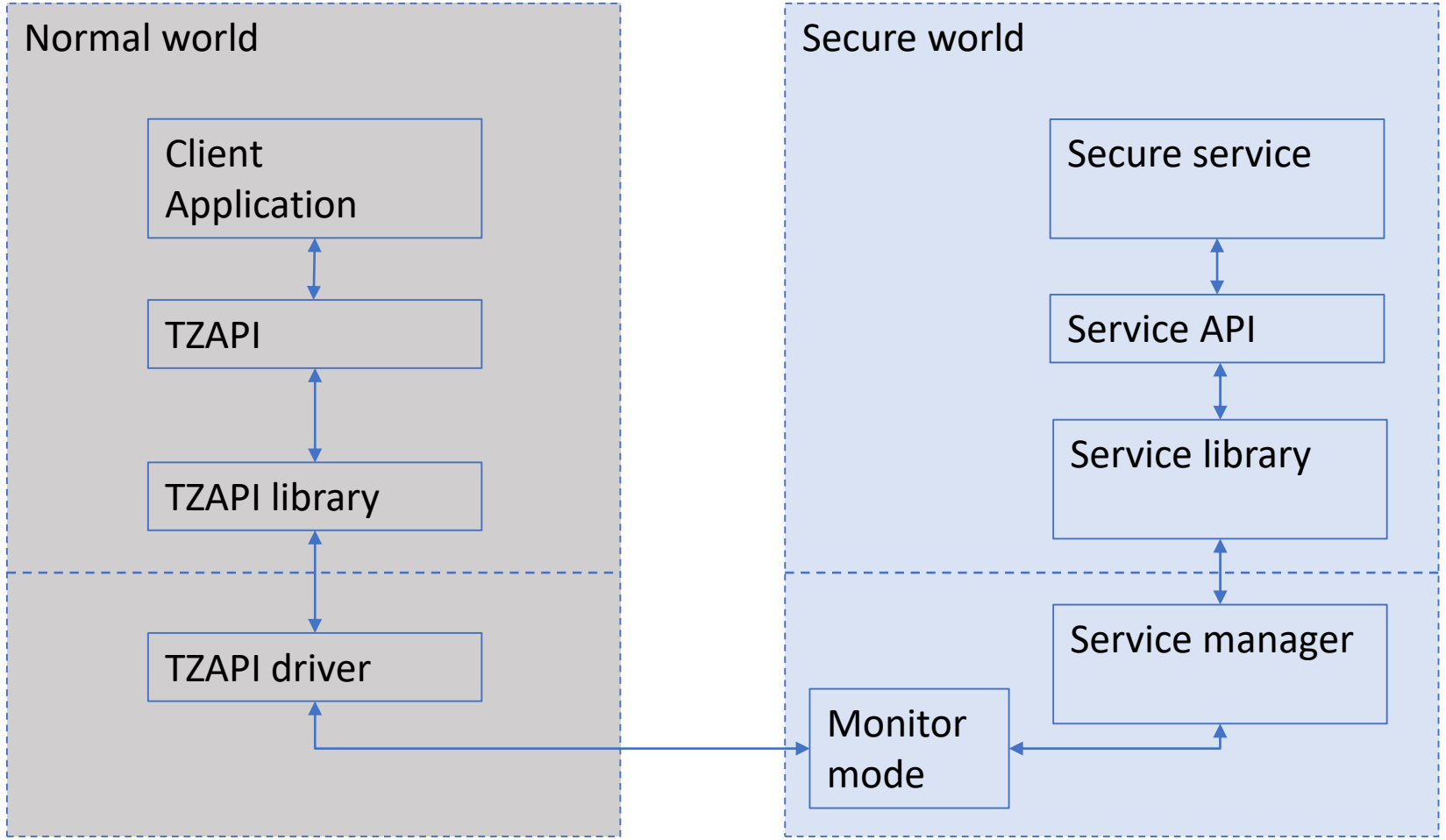
ARM provides a standardized software API, called the TrustZone API (TZAPI)

- A software interface for client applications in the non-secure (rich) operating environment interact with the secure world.

TZAPI is mainly a communications API

- A client sends command requests to a security service
- The client exchanges data with the security services
- The interface support World-shared memory shared by both normal world and secure world, for high performance bulk data transfer.

A System Using TrustZone API [1]



Outline

Introduction

System security

TrustZone Hardware Architecture

TrustZone Software Architecture

TrustZone System Design

Gadget2008 Product Design Brief

Below we use an example to show how TrustZone is used to design a cellular handset that meets security requirements.

- Read [1] for details.

A portable cellular handset

Feature-rich operating system capable executing user downloaded applications.

Audio and video playback with DRM content protection

GadgetStore will allow the user to pay for downloaded content using standard banking facilities

Content Management

A Digital Rights Management (DRM) agent enforces the access to media content based on the purchased rights by the user

- expiry date, number of plays, number of minutes of playback, and the playing device

For a DRM protected file for playback,

- The agent must ensure that the user's rights are still valid and,
- if they are, decrypt the file and pass it to the media player.

Integration in a System Using TrustZone Technology

DRM agent component in the Secure world

- rights storage, rights validation and content decryption

CODEC and media player in the Normal world.

- What can be the problem?

CODEC stack in the Secure world?

- Too complicated to avoid bugs!

Assets

Assets	Security	Description
Rights Data	Authenticity	the information about the user's rights
Rights Checking Code	Authenticity	checks the rights object must be authentic and executed from a secure location
Device Key	Confidentiality, Integrity	the root secret that is used to decrypt the rights data when transmitted from the content store.
Content Key	Confidentiality	the secret key which is stored in a rights object, and is used to decrypt a specific piece of content before playback.
Content Data	Confidentiality	decrypted content that is passed to media player

Attackers

Device owner who wants to enjoy more …

Attacks

- software hack attacks
- simple hardware attacks

General Specification - Secure Boot

Secure boot code located in on-SoC ROM

256-bits of OTP fuse for a SHA256 hash of the RSA public key owned by the Secure world software developer

- Authenticate the public key when needed
- Then validate certificates of code on the device

A statistically unique secret key located in an on-SoC cryptographic accelerator.

- Available only to Secure software
- Not trivial to recover the secret key from SoC, so it is safe
- Confidential data can be encrypted and bound to the device.
- Used as the device key?

General Specification (Cont'd)

Multi-tasking Secure world software

- The Gadget2008 design – multiple pieces of software in the Secure world: a *DRM agent*, *a secure GadgetStore shopping backend*, and *a payment application*.
- A Secure world operating system using the MMU to separate user tasks.

Securing the debug channels

- Two batches of production devices.
- The initial batch with full debug access, including Secure world access, for use by only trusted parties, with the OTP fuse set to a known invalid key.
- The second batch of devices with enabled Normal world debug and permanently disabled all Secure world debug using a fuse.

Content Management Specification

Software video decoding

Soft real-time performance

Non-volatile counter to defend against a replay attack

- An attacker replaces a file system image containing an expired rights object with an older version containing the rights object before it had expired

Secure real-time clock source

Memory requirements

References

[1] ARM Security Technology Building a Secure System using TrustZone® Technology, 2009

[2] Secure the Windows 10 boot process, 06/23/2017