Trusted Execution Environments

Chester Rebeiro
IIT Madras

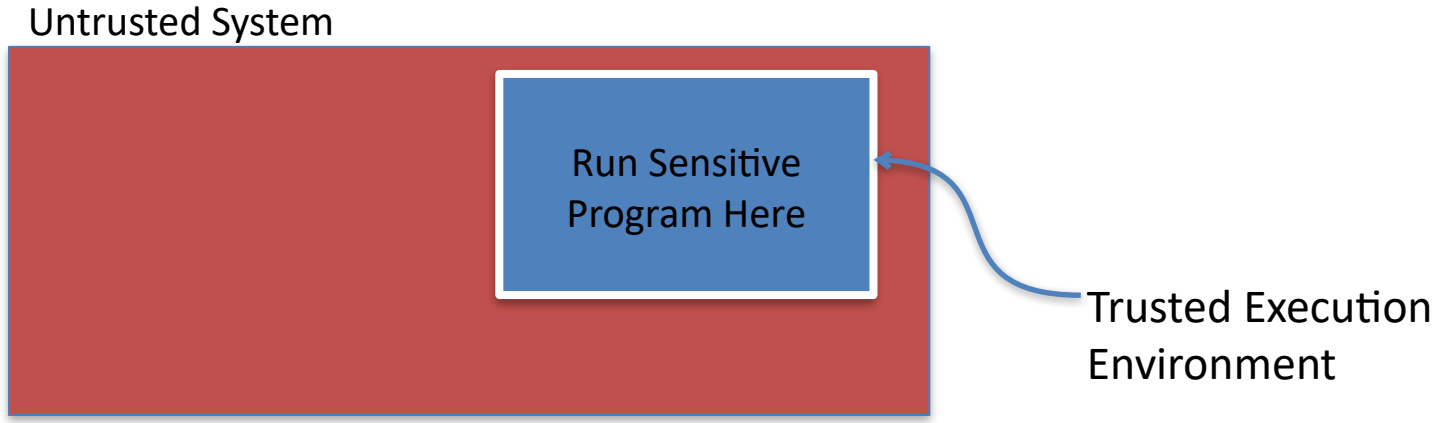
Some of the slides borrowed from Intel; CDACH; ARM

Previously in SSE…

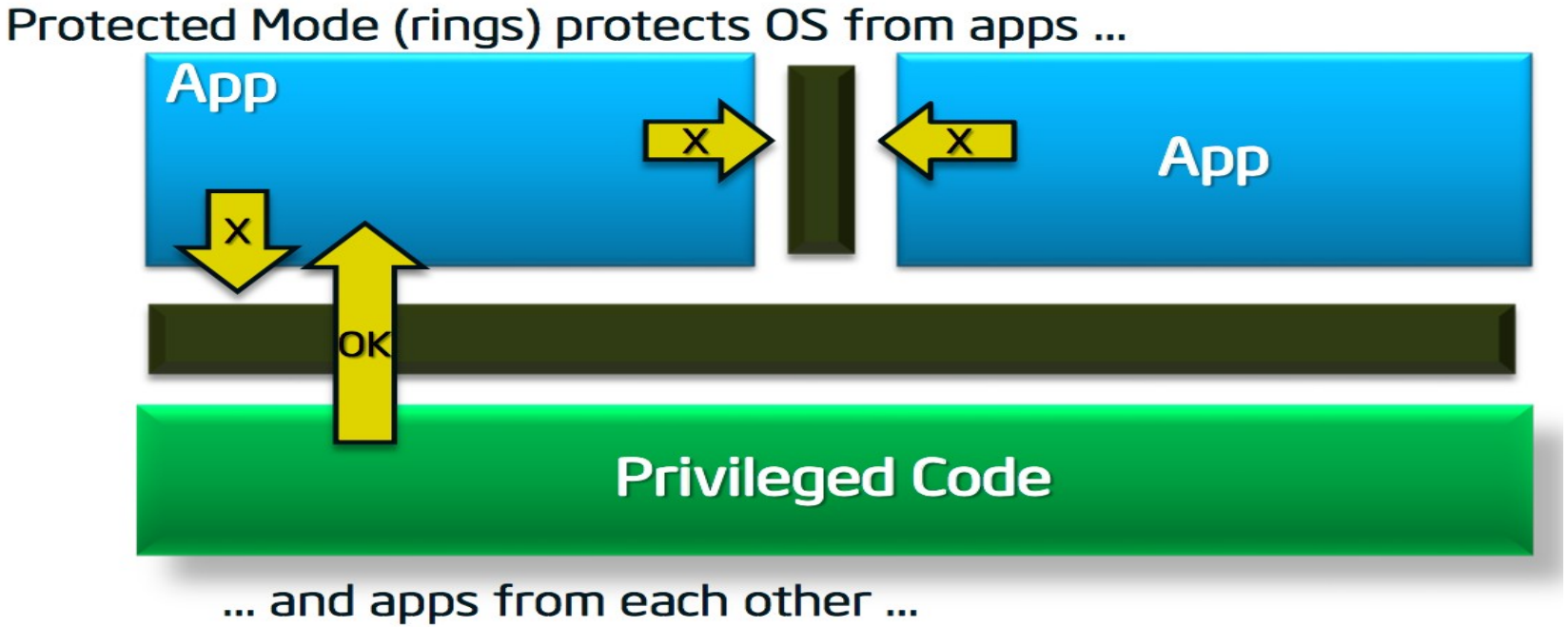
- We looked at techniques to run an untrusted code safely

Today in SSE…

- We now look at how to run sensitive code in an untrusted environment
  - Besides other applications, the OS can also be untrusted.
  - Attackers can probe hardware
- What to worry about:
  - Code / Data of the sensitive app gets read / modified by the system

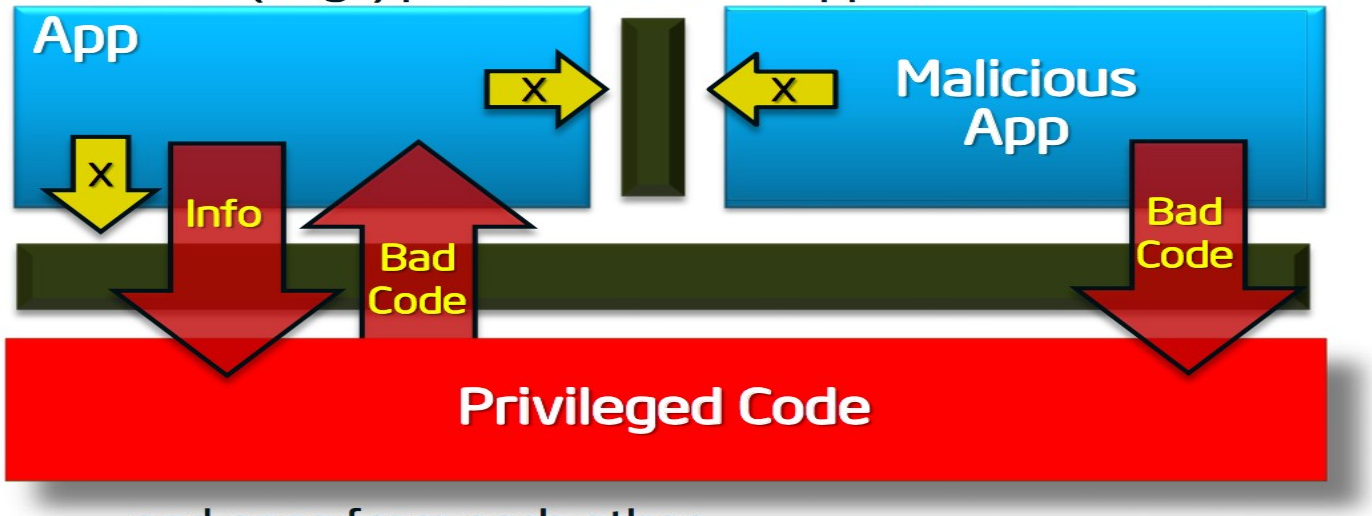


Basic Problem
(Ring Architecture)



Basic Problem
(Ring Architecture)

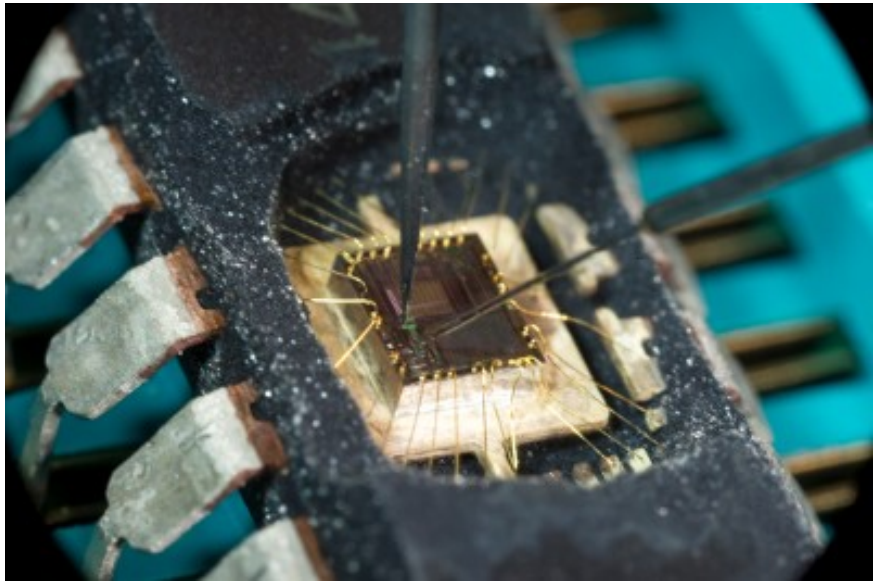
Protected Mode (rings) protects OS from apps …

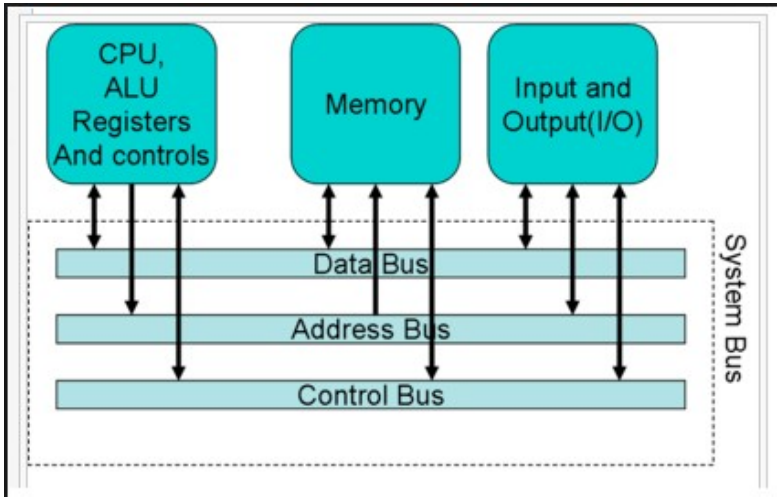


… and apps from each other …

… UNTIL a malicious app exploits a flaw to gain full privileges and then tampers with the OS or other apps

Invasive Attacks





Trusted Execution Environments

Achieve confidentiality and integrity even when the OS is compromised!

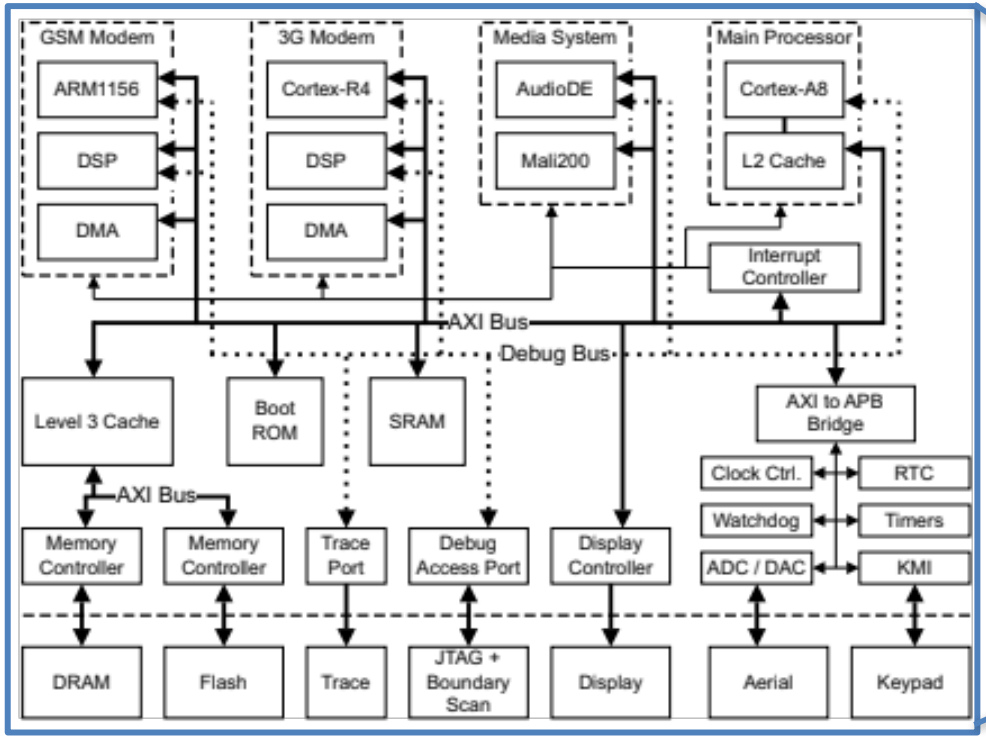
- ARM : Trustzone (trusted execution environments)
- Intel : SGX (enclaves)

ARM Trustzone

Trustzone Security Whitepaper, ARM
http://infocenter.arm.com
/help/topic/com.arm.doc.prd29-genc-009492c/PRD29GENC-009492C_trustzone_security_whitepaper.pdf

Some of the slides borrowed from CDACH; ARM

ARM System on Chips



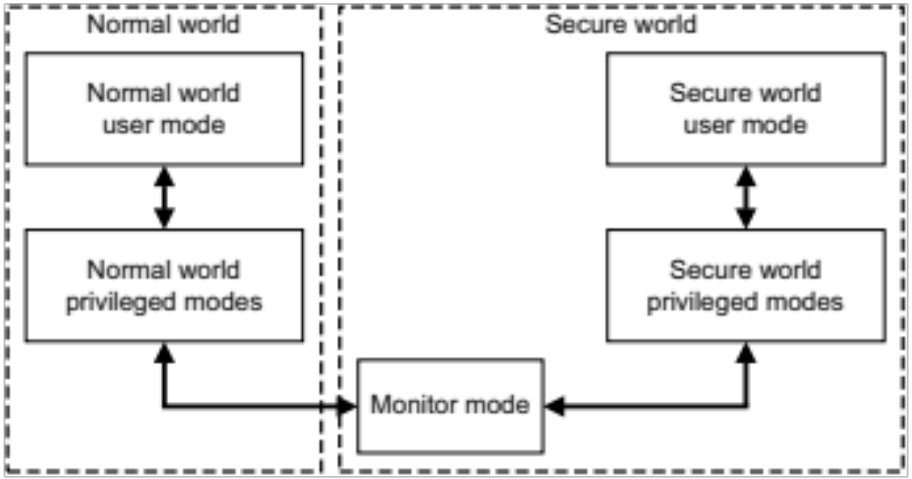
ARM Trustzone (Main Idea)

Hardware and Software partitioned into two: Normal and Secure worlds

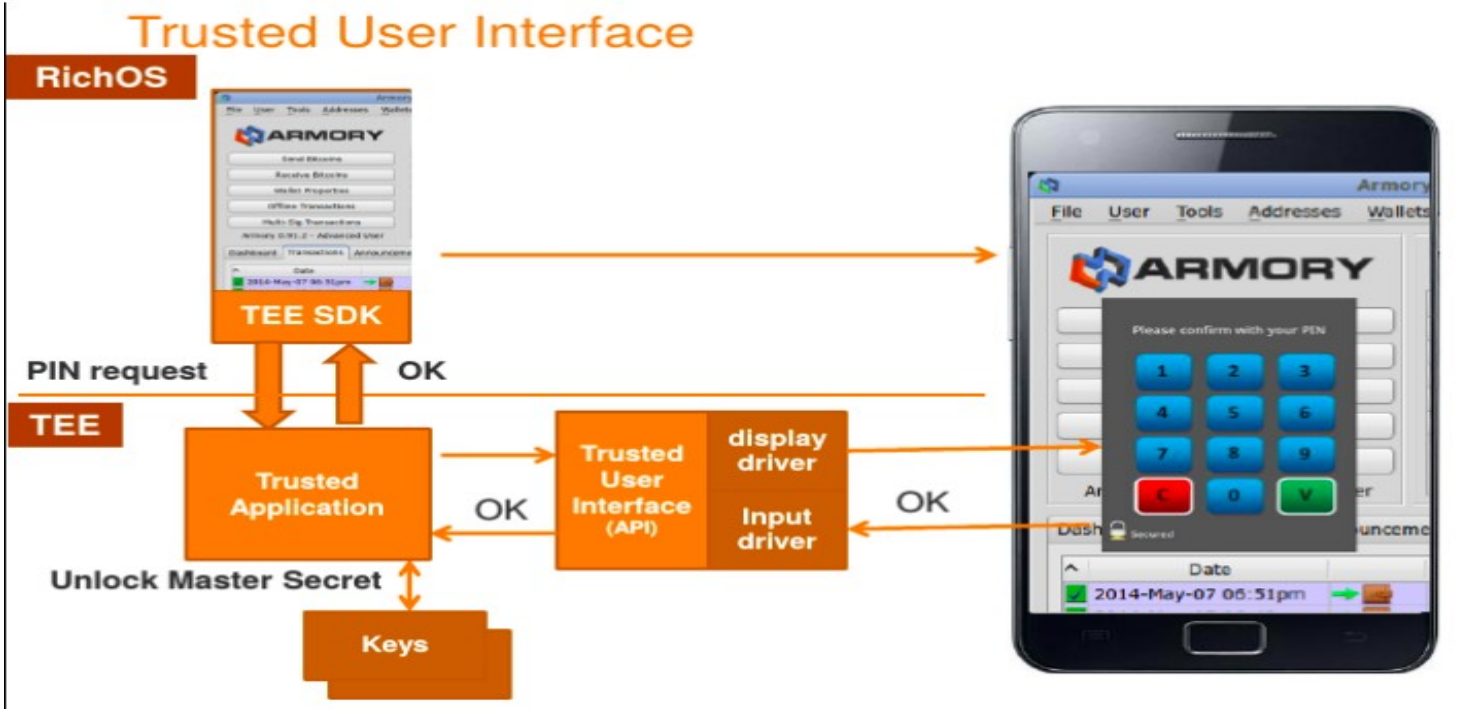
A single hardware processor timesliced between secure and normal worlds

Secure world provides an environment that supports confidentiality and integrity.

- Can prevent software attacks
- Cannot prevent invasive attacks



A Typical Trustzone Application



Switching Worlds

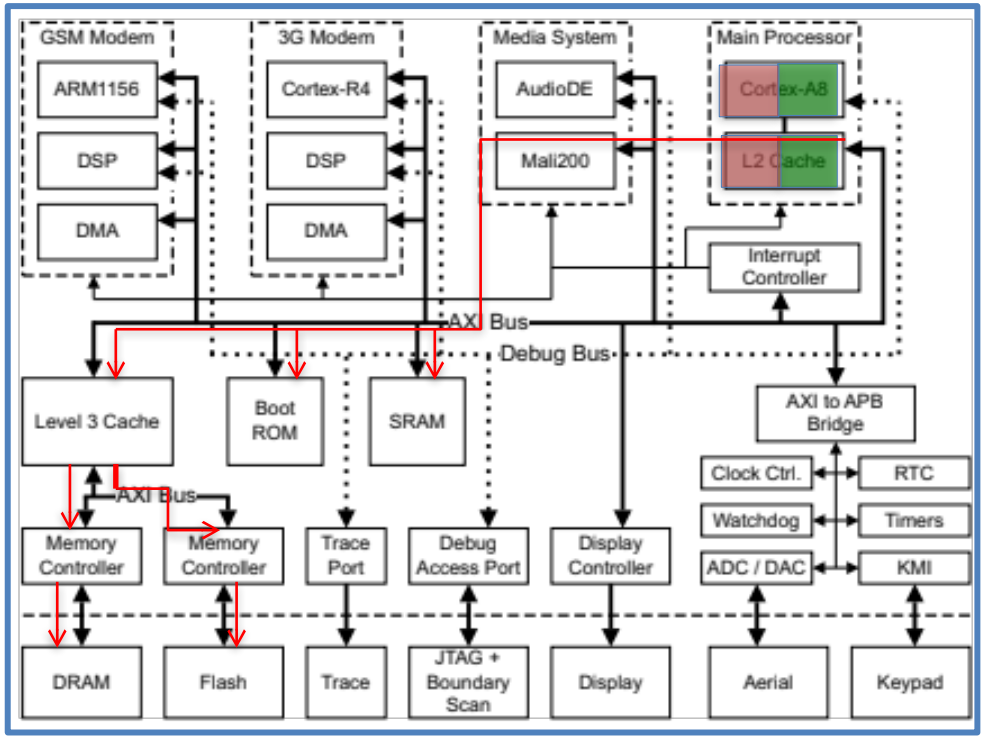
- Execution in time sliced manner (Secure <-> Normal)
- New mode (monitor mode) that is invoked during switching modes
- Mode switching
  - triggered by *secure monitoring call* (SMC) instruction
  - certain hardware exceptions (interrupts, aborts)

- Monitor Mode: saves state of the current world and restores the state of the world being switched to. Restoration by return-from-exception.

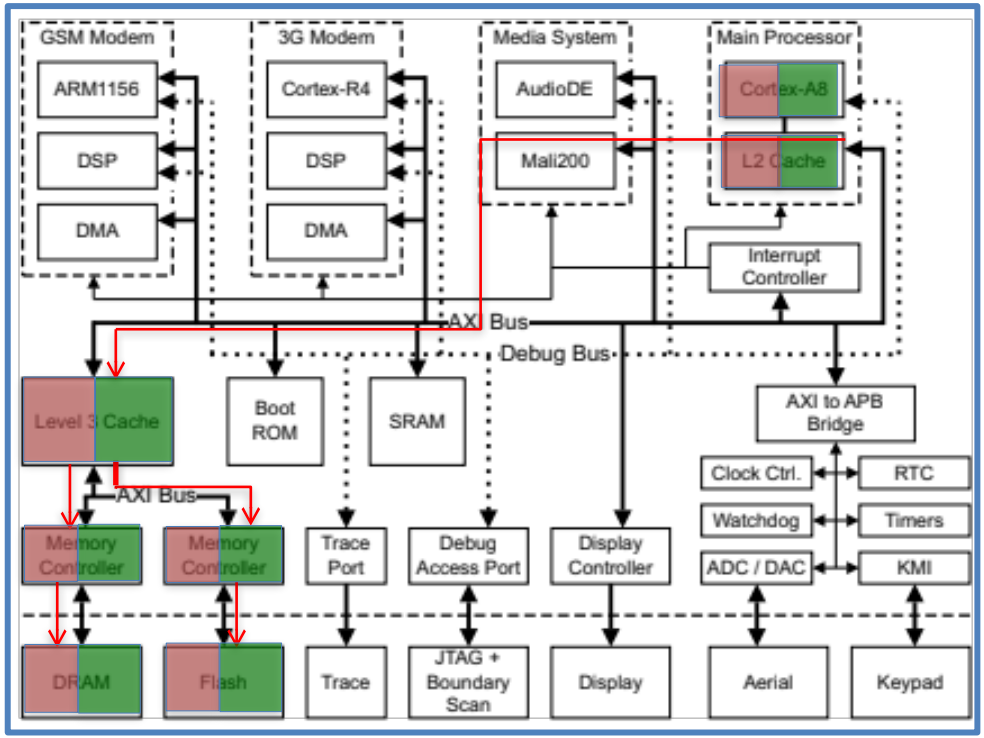
- NS Bit: in configuration register indicates secure / normal operating mode.

  NS = 1 -> indicates non-secure (normal) mode

NS Bit extends beyond the chip

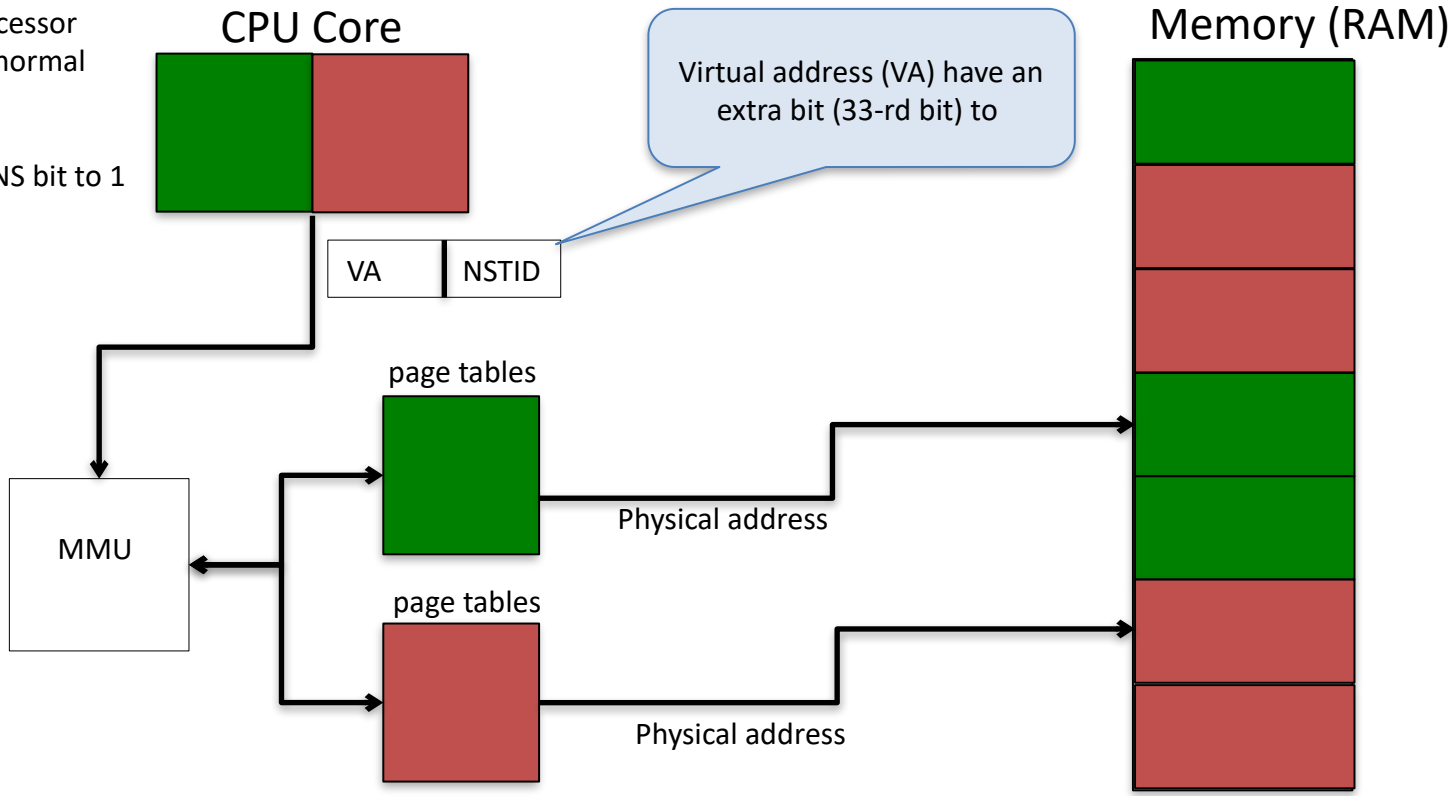


NS Bit extends beyond the chip

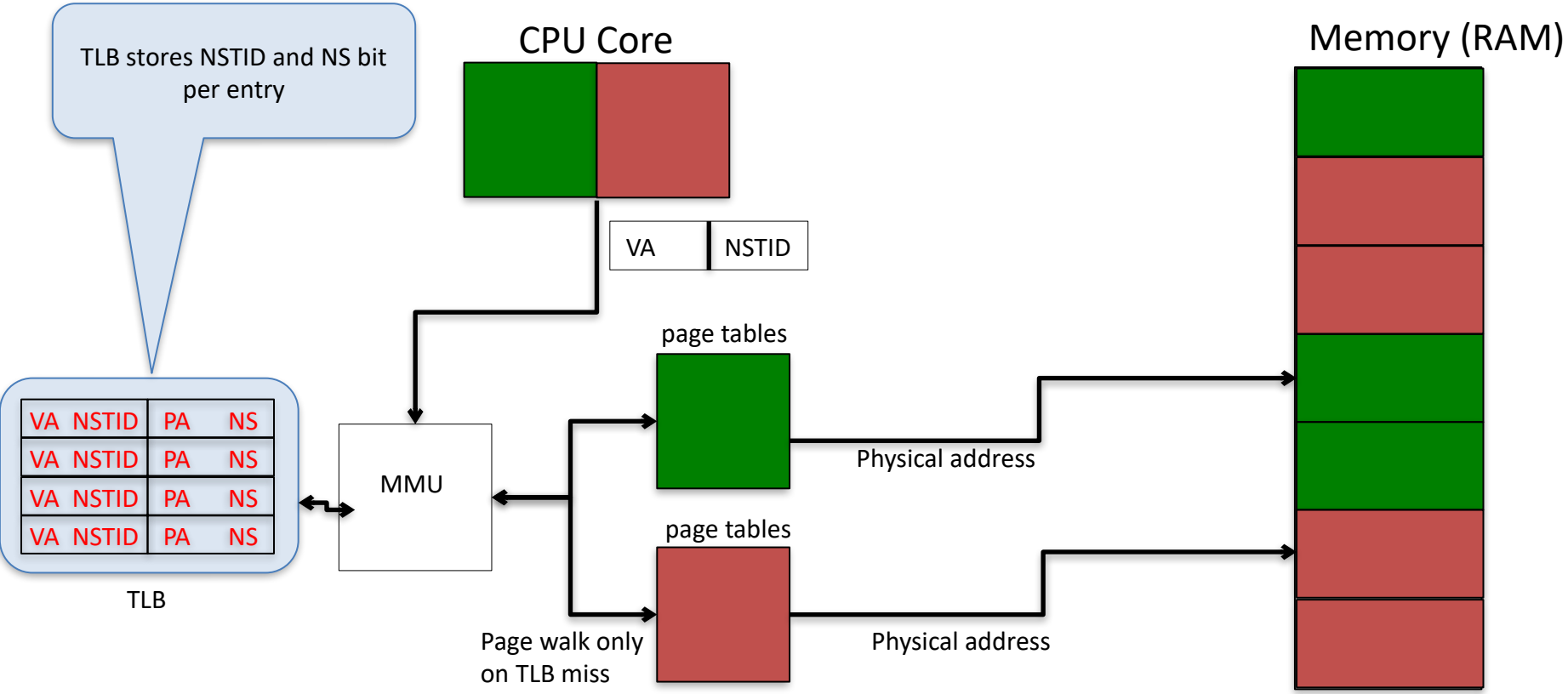


Memory Management

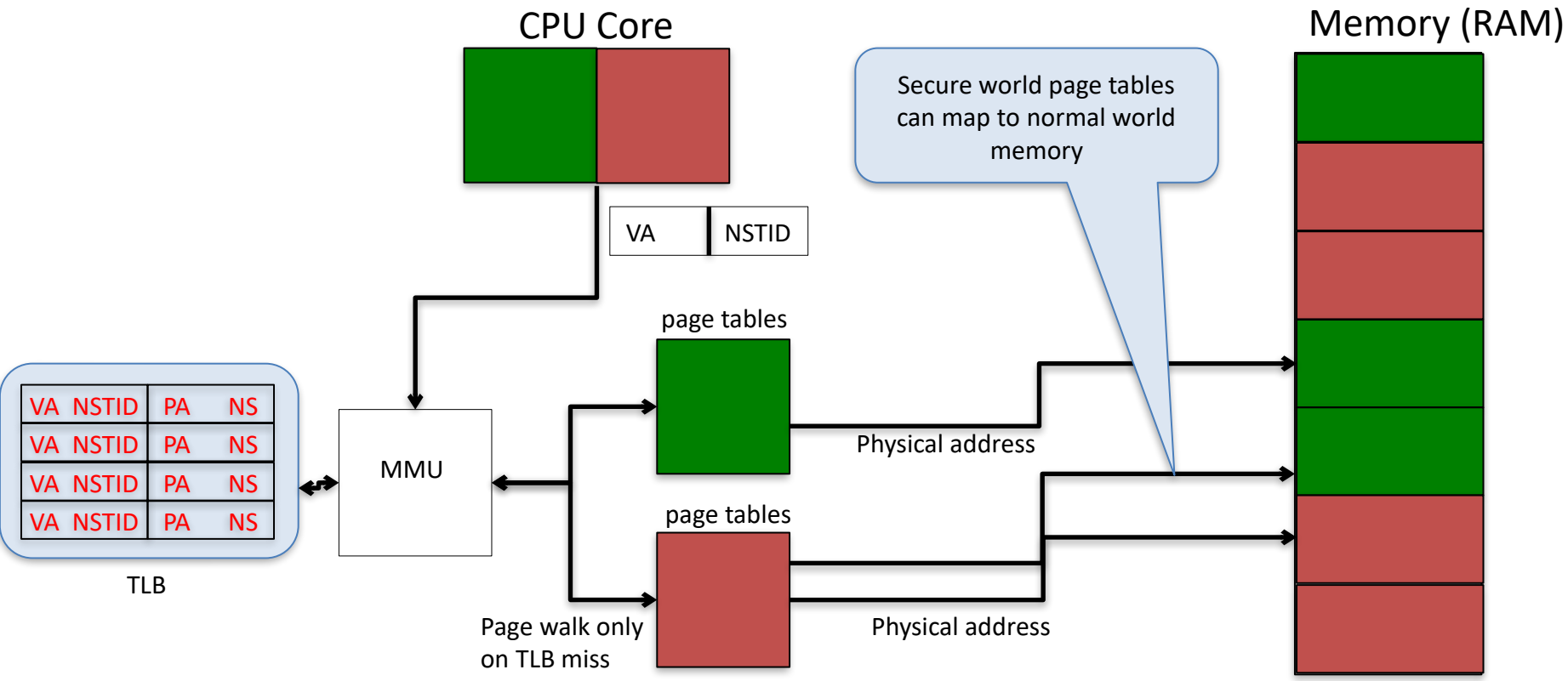
- Non Secure Table Identifier current state of the processor (0 if secure world / 1 if normal world
- If NSTID = 1 then force NS bit to 1



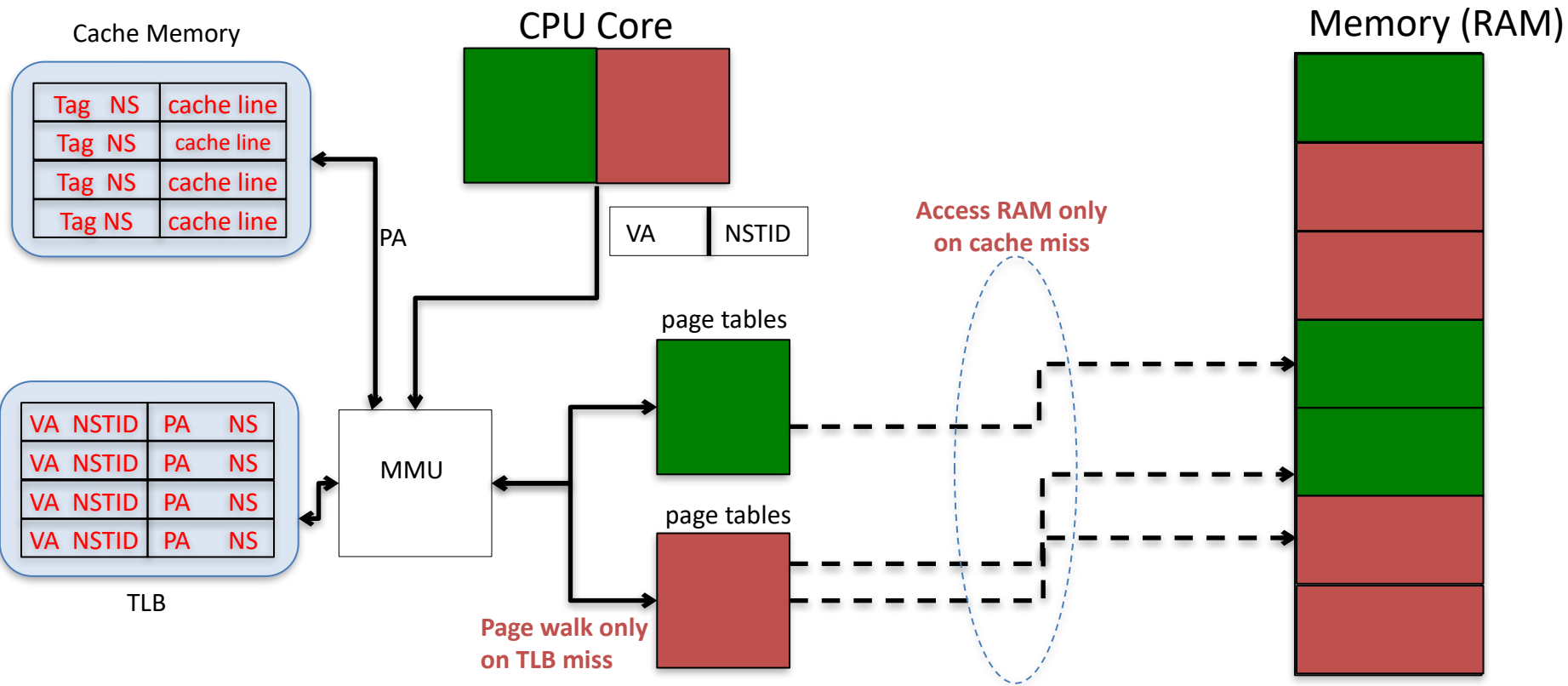
Memory Management



Memory Management



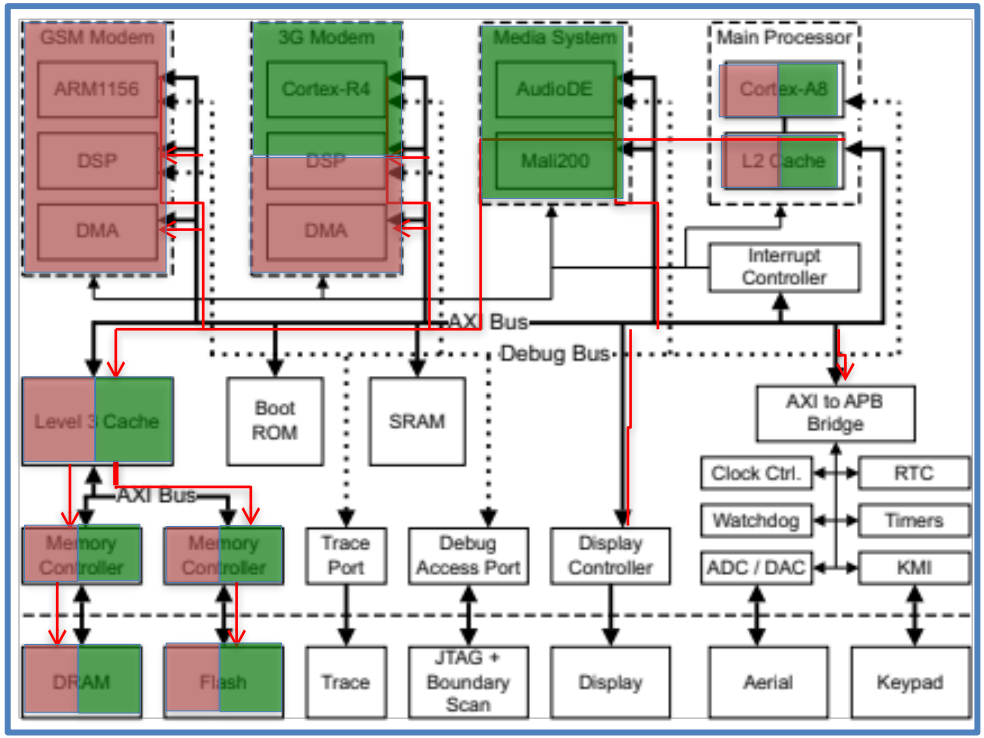
Memory Management



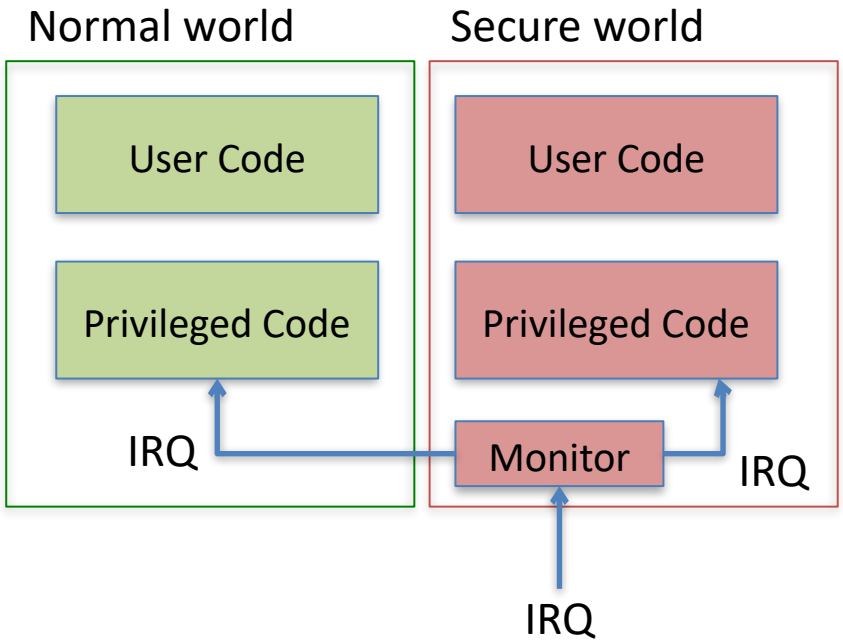
Memory Management Units

- Two virtual MMUs (one for each mode)
  - Two page-tables active simultaneously
- A single TLB present
  - A tag in each TLB entry determines the mode
    (Normal and Secure TLB entries may co-exist; this allows for quicker switching of modes)
  - alternatively the monitor may flush the TLB whenever switching mode
- A single cache is present
  - Tags (again) in each line used to store state
  - Any non-locked down cache line can be evicted to make space for new data
  - A secure line load can evict a non-secure line load (and vice-versa)

Secure and Normal Devices

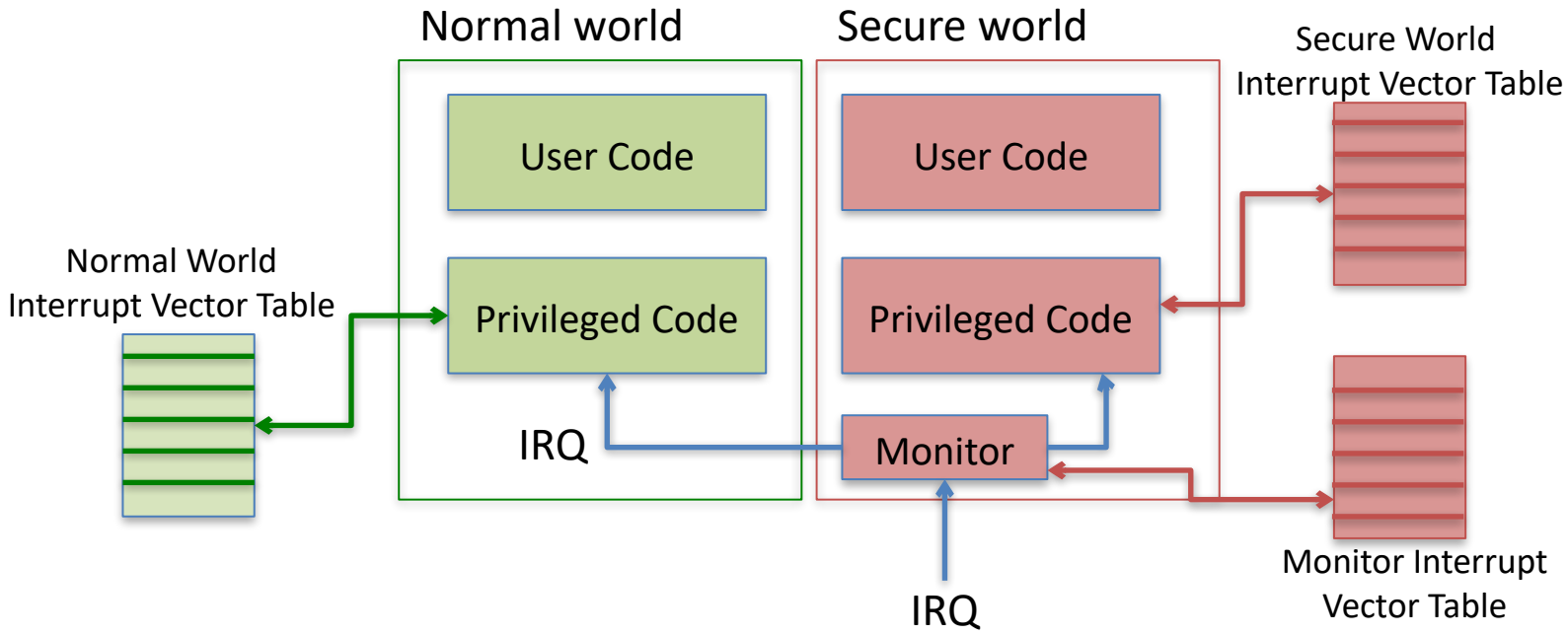


Interrupts



All interrupts routed to monitor first.
Interrupts can be configured to go either to the normal world or secure world.

Interrupts



All interrupts routed to monitor first.

Interrupts can be configured to go either to the normal world or secure world.

Software Architecture

- The minimal secure world can just have implementations of synchronous code libraries
- Typically has an entire operating system
  - Qualcomm's QSEE; Trustonics Kinibi; Samsung Knox; Genode
  - The secure OS could be tightly couples to the rich OS so that a priority of a task in the rich OS gets mapped accordingly in the secure OS
  - Advantage of having a full OS is that we will have complete MMU support
- Intermediate Options

Secure Boot

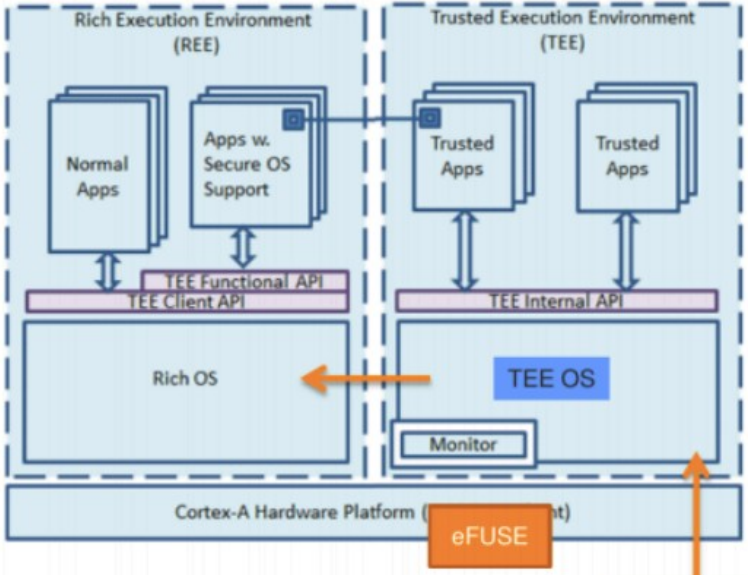
Why?

Attackers may replace the flash software with a malicious version, compromising the entire system.

How?

Secure chain of trust.
Starting from a root device (root of trust) that cannot be easily tampered

Secure Boot Sequence



On chip ROM based Bootloader

May be internal to the CPU;
Initializes critical peripherals, memory controllers

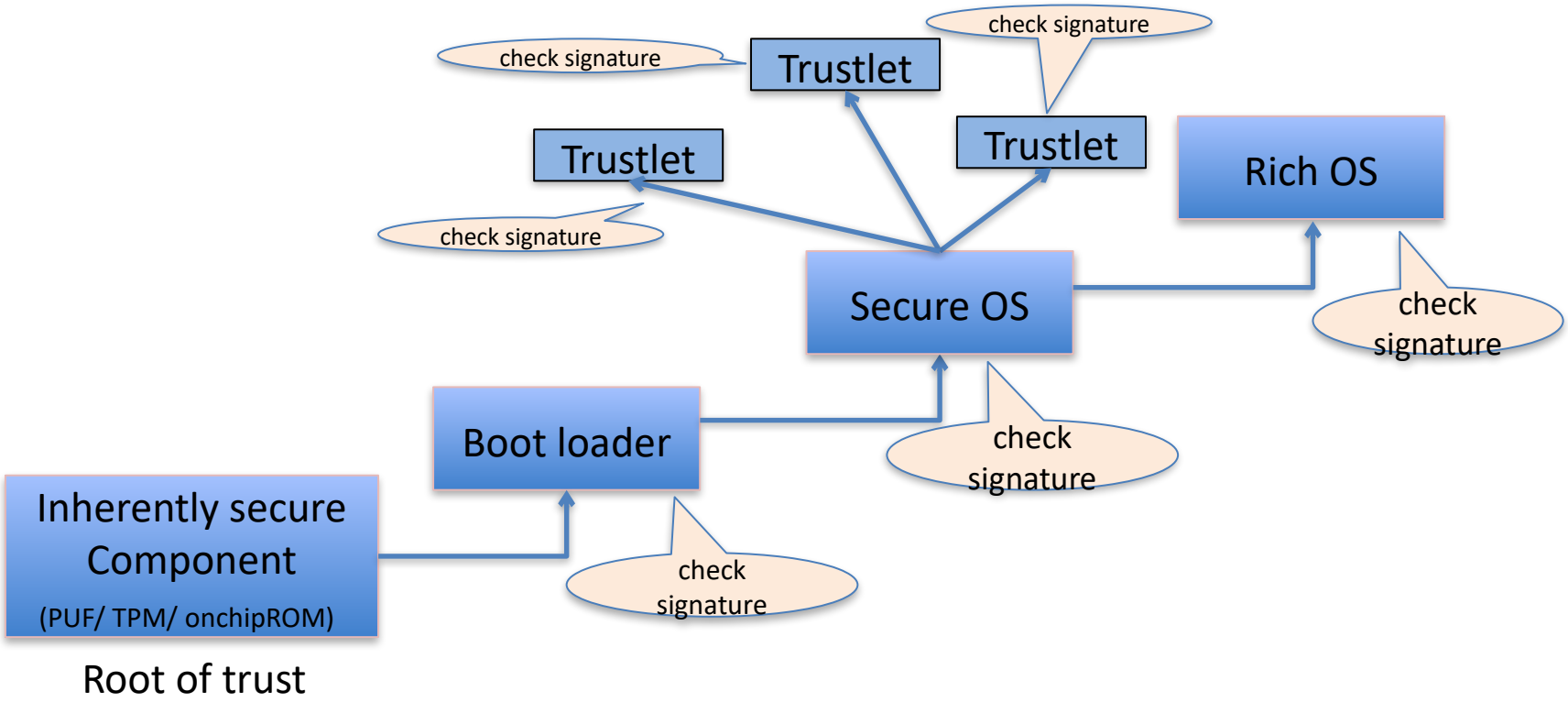
Device bootloader

Stored on Flash device (typically)
Initialize critical peripherals

Secure operating system

Rich operating system

Chain of Trust



Points to Ponder

Describe how ARM trustzone can handle invasive attacks?

What can it handle?

What are the limitations?