*presented by*


# ARM Server's Firmware Security

Spring 2017 UEFI Seminar and Plugfest
March 27 - 31, 2017

Presented by Zhixiong (Jonathan) Zhang
(Cavium, Inc.)

# Agenda

- Challenges
- Hardware Design Differentiations
- Firmware Solutions
- Conclusion
- Q&A

# Challenges

# Market for ARMv8 volume servers

HPC

Cloud Compute

Telco

Storage

OCP

Web Hosting

# Future opportunities

- Tomorrow's edge devices face similar security challenges as today's server do.

- Tomorrow's edge devices are:
  - Always on-line
  - Open hardware design
  - Open software design

# Managing firmware images in a data center

- Data centers have many hosts and appliances with different architectures.
- One network storage appliance may have dozens of hosts.
- One host has multiple FW images:
  - Processor FW images: ARM TF, UEFI
  - Microcodes for inter-processor link, PCIe, etc.

# Security related ARM standards

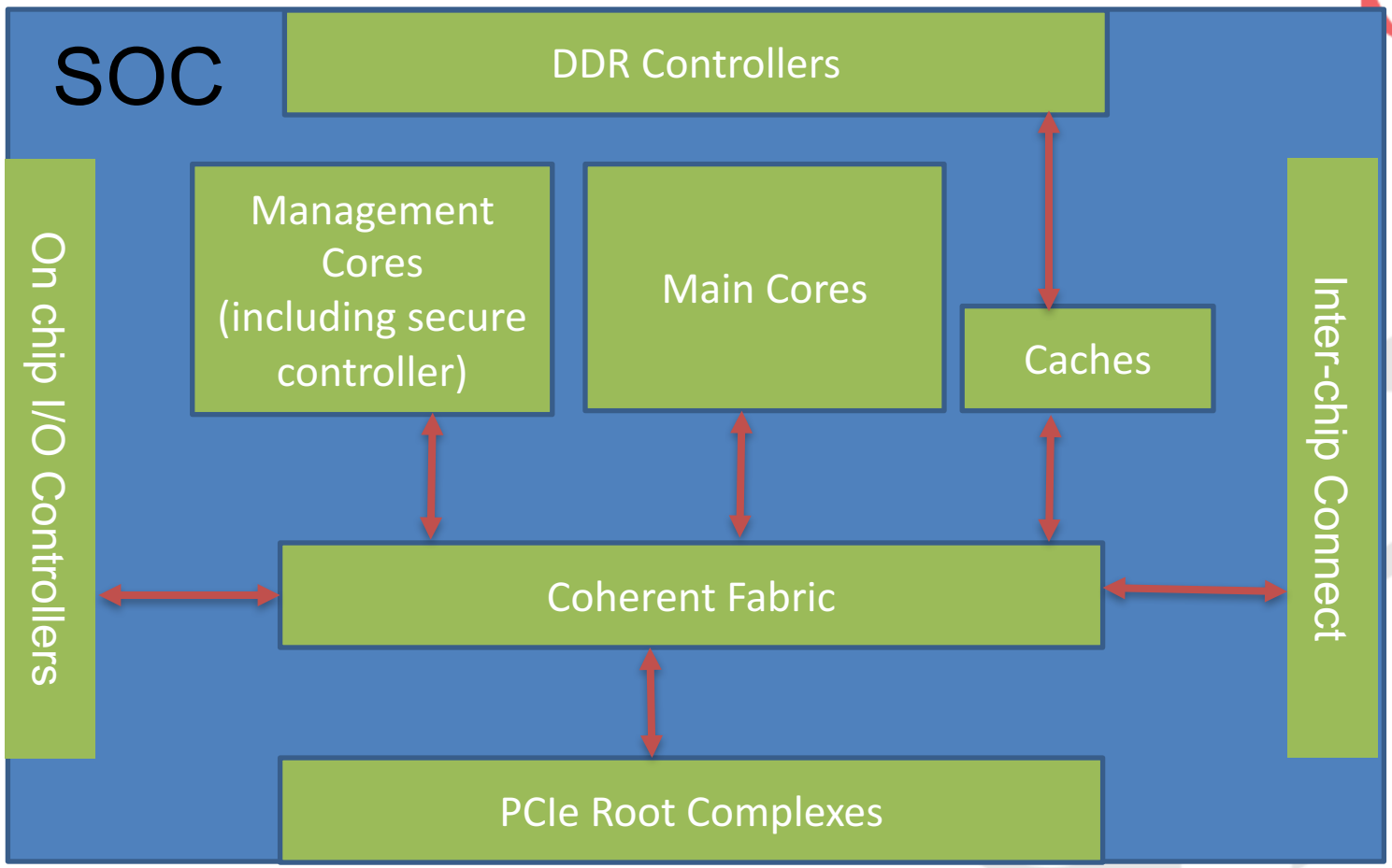
- **Trusted Base System Architecture**
  - Presents a System-on-Chip (SoC) architecture that incorporates a trusted hardware base suitable for the implementation of systems compliant with key industry security standards and specifications, in particular those dealing with third party content protection, personal data, and second factor authentication.

- **Trusted Board Boot Requirements**
  - Describes and defines a Trusted Boot Process for application processors based around the ARMv8-A architecture.

Note: Both standards are ARM partners only. They are for client devices, but useful as reference for server.

# Hardware Design Differentiations

# On-chip secure controller

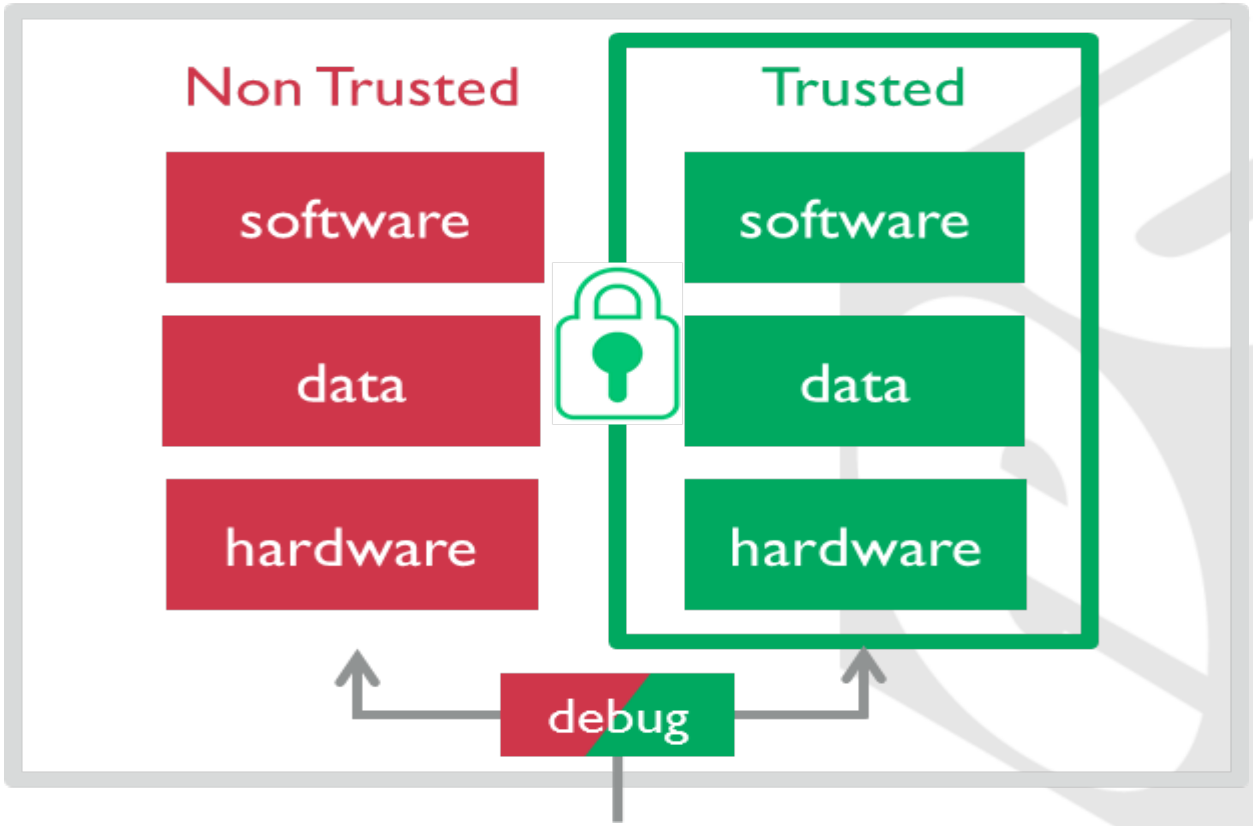


# Security related co-processors

- Common storage area in SoC for all cores.
- Random number generator/memory.

# ARM TrustZone technology

- https://www.arm.com/products/security-on-arm/trustzone

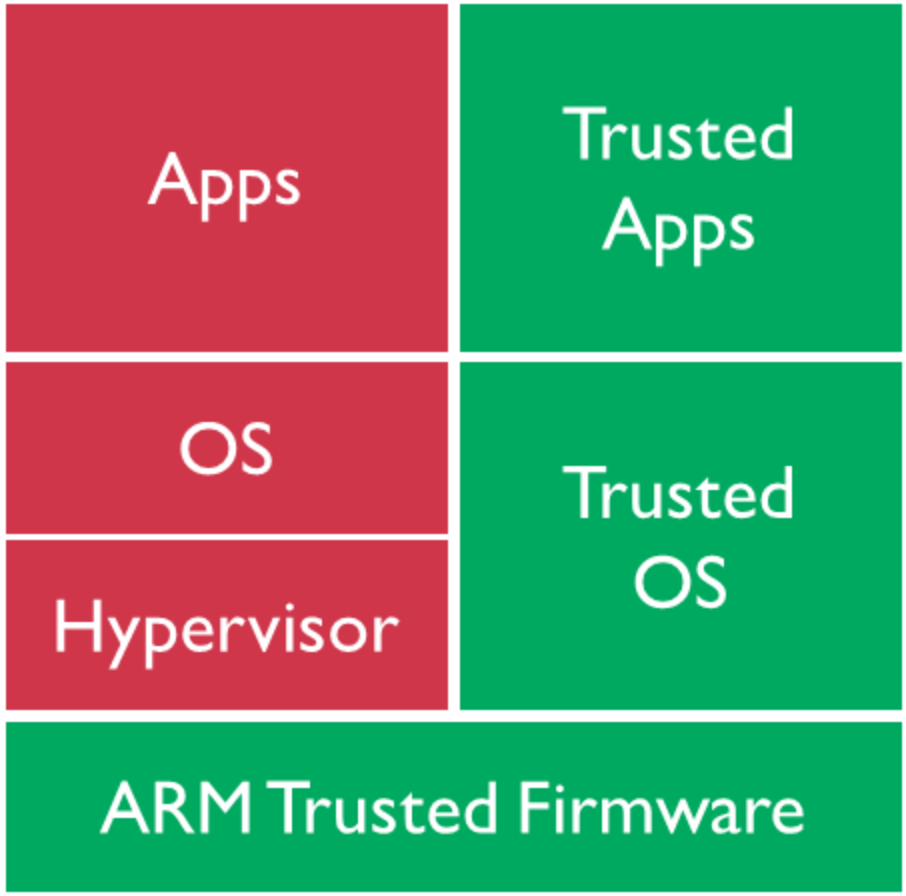


# ARM TrustZone technology

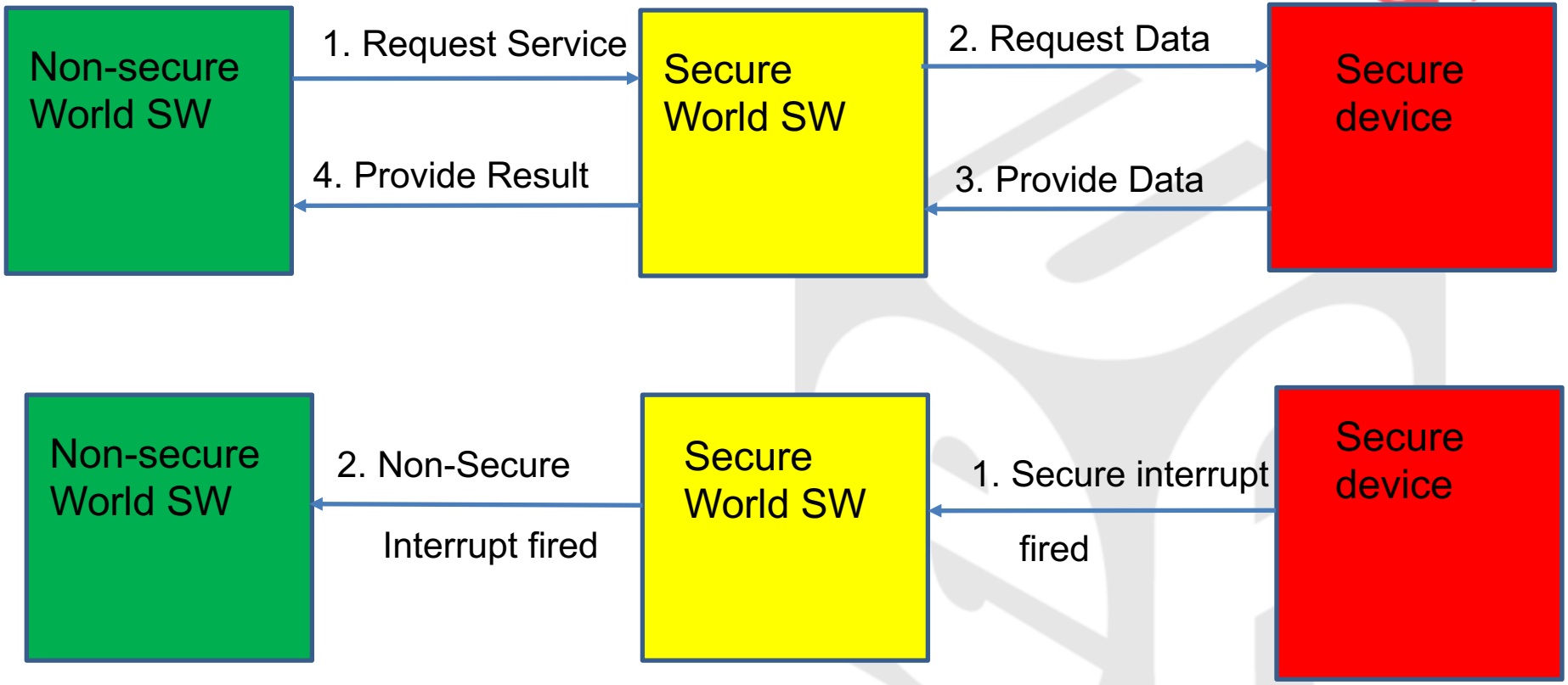
- Secure memory
  - Operation fails when a non-secure bus master attempts to access secure memory.
- Secure devices
  - A secure interrupt controller and timer allows a non-interruptible secure task to monitor the system.
  - A securable keyboard peripheral enables secure entry of a user password.
- Secure world
  - ARMV8 architecture defines secure world vs. non-secure world. A CPU can context switches between secure world and non-world and among different exception levels.

# Firmware Solutions

# ARM Trusted Firmware



# Secure world service



# Secure memory

- Any data not needed by non-secure SW must reside in secure memory! Non-secure memory is inherently not safe.
- Don’t leak secrets! When copying data from secure memory to non-secure memory.
- Don’t trust input! When copying data from non-secure memory to secure memory.

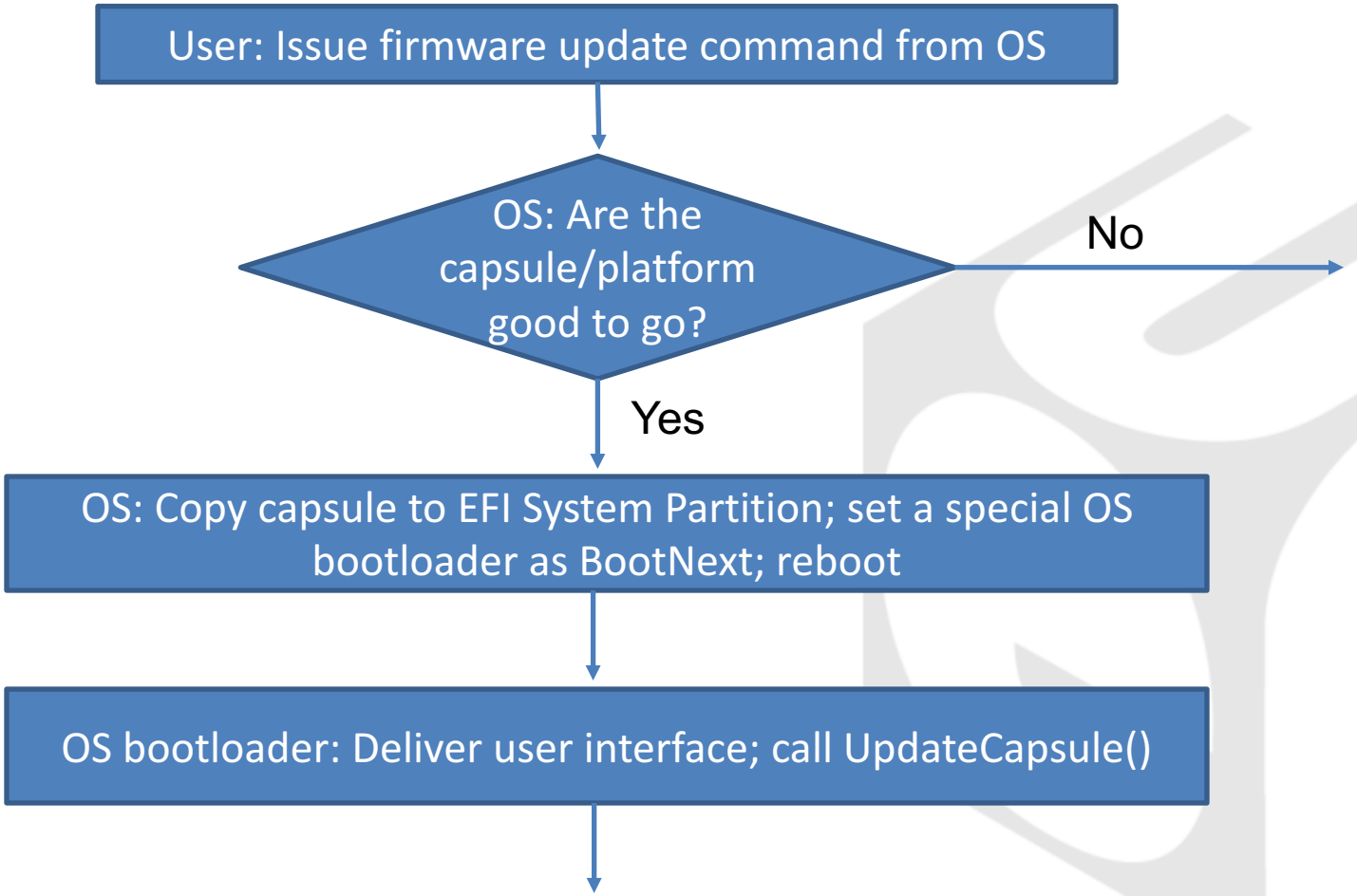
# Secure device

- Which devices must be made secure devices?
  - If direct device access from NS world is not necessary. Example: flash.
  - If device is not needed by main cores. Example: management controller controlled devices.
  - If device is only needed by secure world. Example: security related co-processors, secure interrupt/timer.
- Devices unsecure but not exposed to OS:
  - This can be done through either ECAM enumeration disable, ACPI (Device Tree), or both.
  - Device used by UEFI run time service. Example: RTC device.

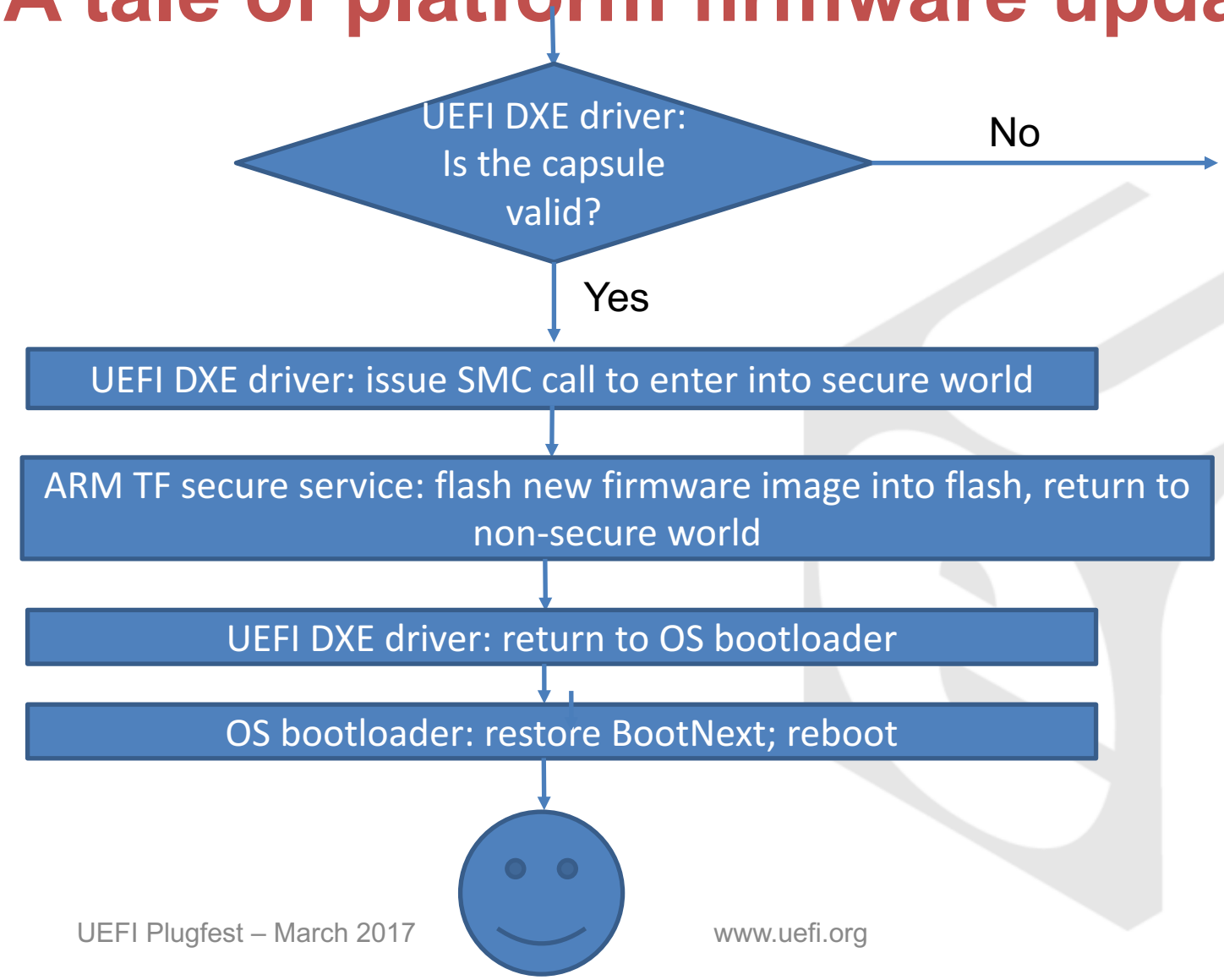
# Capsule update

- Intel's recent works:
  - Patchset: https://lists.01.org/pipermail/edk2-devel/2016-November/004244.html
  - Whitepaper: https://github.com/tianocore-docs/Docs/raw/master/White_Papers/A_Tour_Beyond_BIOS_Capsule_Update_and_Recovery_in_EDK_II.pdf
- Windows:
  - Whitepaper: https://msdn.microsoft.com/en-us/windows/hardware/drivers/bringup/windows-uefi-firmware-update-platform
- Redhat:
  - Fwupd project: http://fwupd.org.s3-website-eu-west-1.amazonaws.com/
  - Blog: https://blog.uncooperative.org/blog/2015/09/16/an-update-on-firmware-updates/

# A tale of platform firmware update



# A tale of platform firmware update

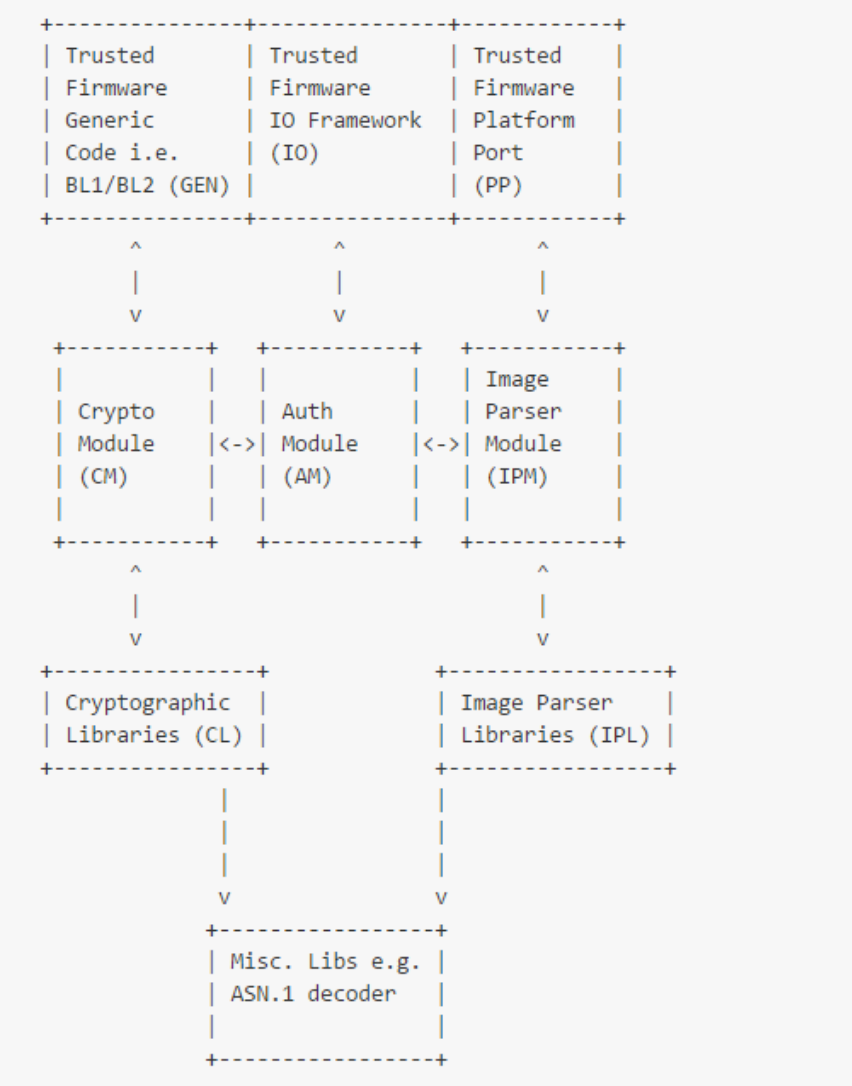


# Secure boot

- Secure boot vs. managed boot.
- Chain of trust following PKCS (Public Key Cryptography Standards)
- Starting from bootRom, each boot loader loads, decrypts, authenticates, passes controls to, the next boot loader, all the way to OS.
- FW of other devices such as microcodes, need to be securely loaded as well, if applicable.

# Secure boot – ARM TF

- https://github.com/ARM-software/arm-trusted-firmware/blob/master/docs/auth-framework.md
- https://github.com/ARM-software/arm-trusted-firmware/blob/master/docs/trusted-board-boot.md

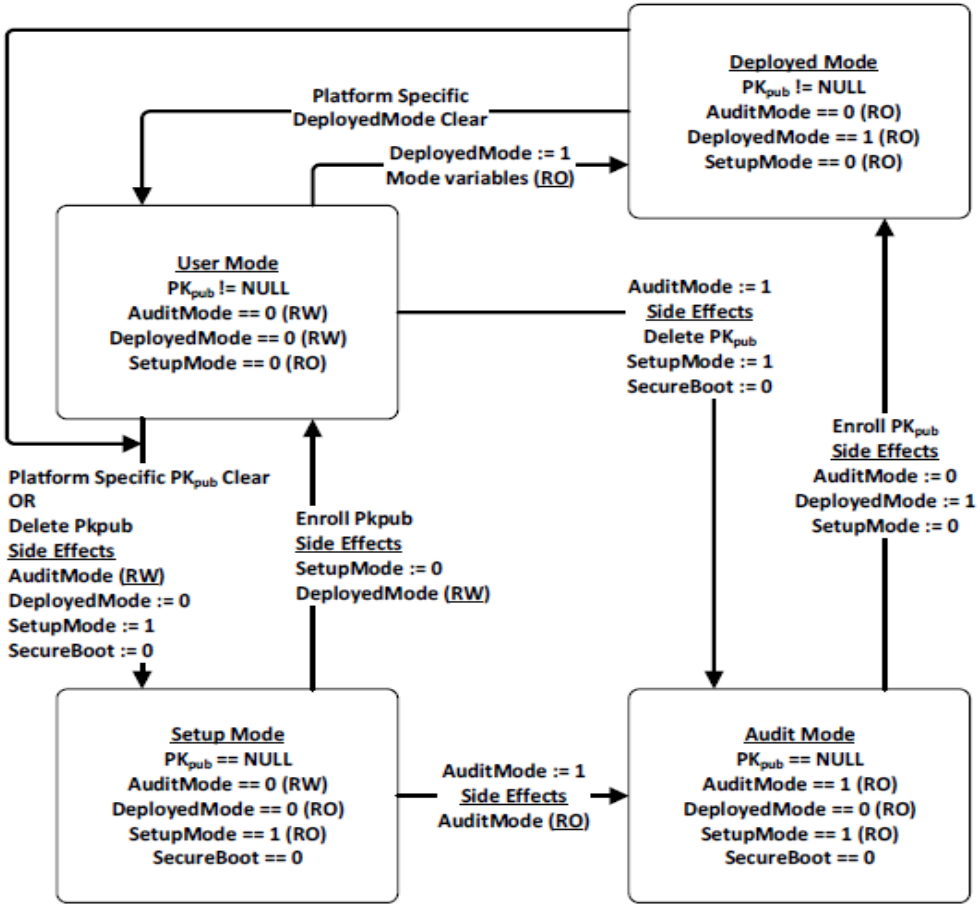


# Secure boot – UEFI 2.3.1

- Intel whitepaper: https://github.com/tianocore-docs/Docs/raw/master/White_Papers/A_Tour_Beyond_BIOS_into_UEFI_Secure_Boot_White_Paper.pdf
- Platform Keys:
  - Establishes a trust relationship between the platform owner and the platform firmware.
  - Must be stored in non-volatile storage which is tamper and delete resistant. Example: EEPROM.
- Key Exchange Keys:
  - Establish a trust relationship between the operating system and the platform firmware.
  - Must be stored in non-volatile storage which is tamper resistant. Example: flash that is secure, eg. non-accessible from non-secure world.

# Secure boot – UEFI 2.6

- Customized UEFI secure boot: http://www.uefi.org/sites/default/files/resources/UEFI_Plugfest_VZimmer_Fall_2016.pdf
- https boot: https://github.com/tianocore-docs/Docs/raw/master/White_Papers/EDKIIHttpsBootGettingStartedGuide_1.2.pdf



# Conclusion

# Challenges mean opportunities

- As UEFI becomes centerpiece of modern day devices, from server to embedded devices, their security faces new challenges
- ARM SoC and UEFI FW ecosystem provide necessary building blocks for security solutions

# Go ahead of the curve

- We will bankrupt ourselves in the vain search for absolute security – Dwight D. Eisenhower
- But in the mean time, we need to plan ahead…
- A chain is no stronger than its weakest link.


Q&A

Thanks for attending the Spring 2017 UEFI Seminar and Plugfest

For more information on the UEFI Forum and UEFI Specifications, visit http://www.uefi.org

*presented by*