tianocore

UEFI & EDK II TRAINING

Introduction to Platform Firmware Security w/ UEFI

tianocore.org

tianocore

LESSON OBJECTIVE

- Why is platform firmware Security important
- UEFI boot flow with the threat model
- Security technologies overview
- Tools and resources on how to test firmware for security

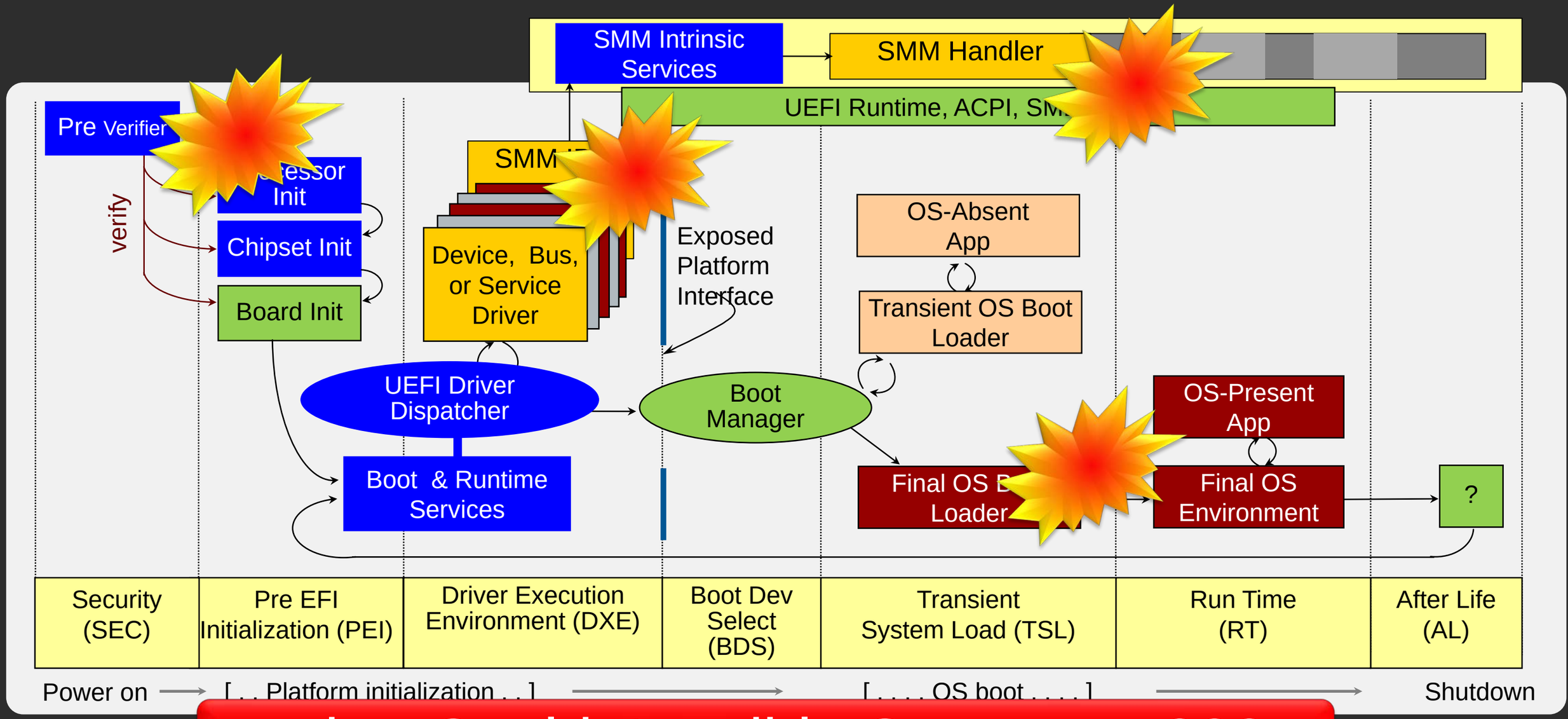
tianocore

UEFI BOOT FLOW

UEFI boot flow with the threat model

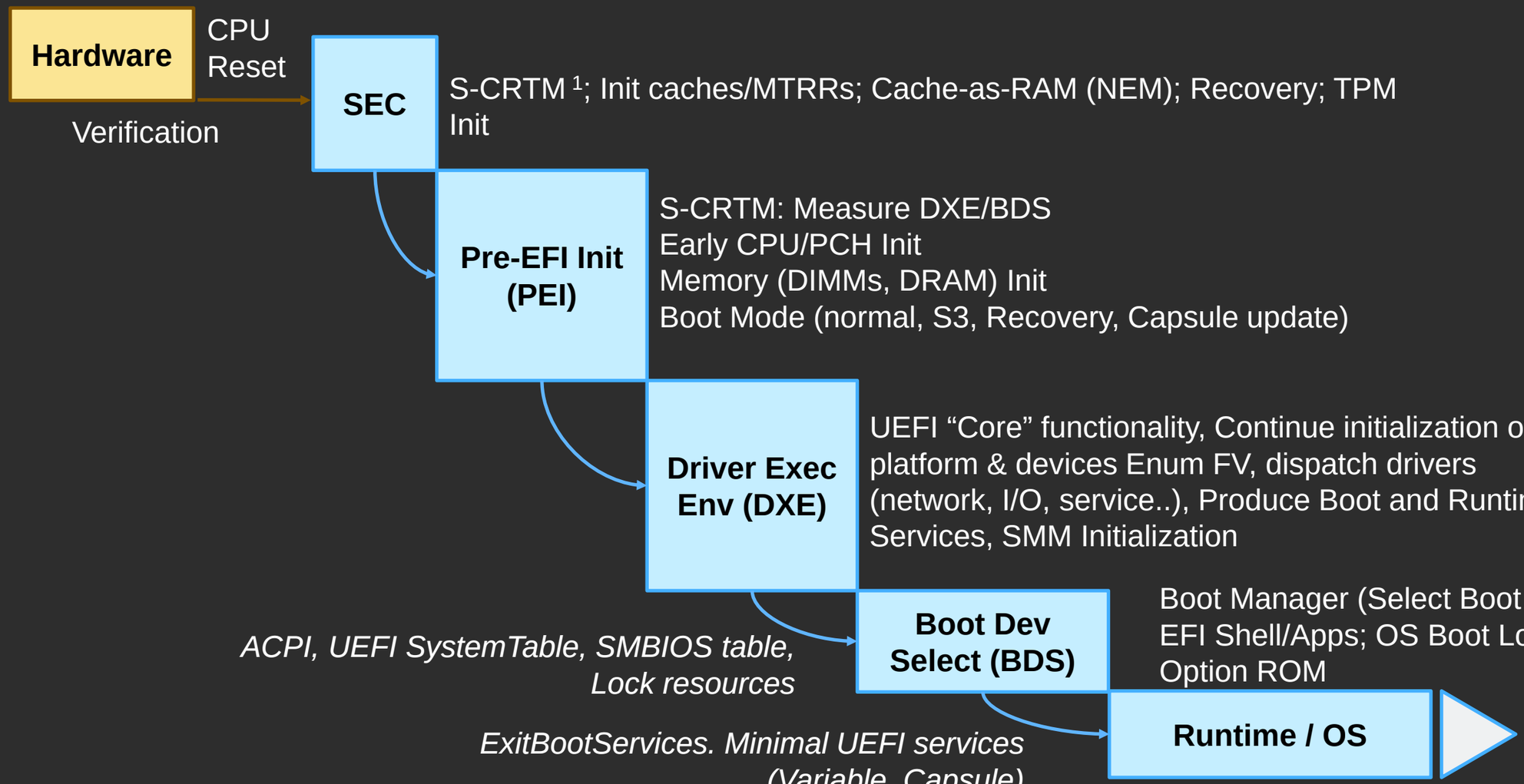
tianocore

UEFI Boot Execution Flow



What Could Possibly Go Wrong ???

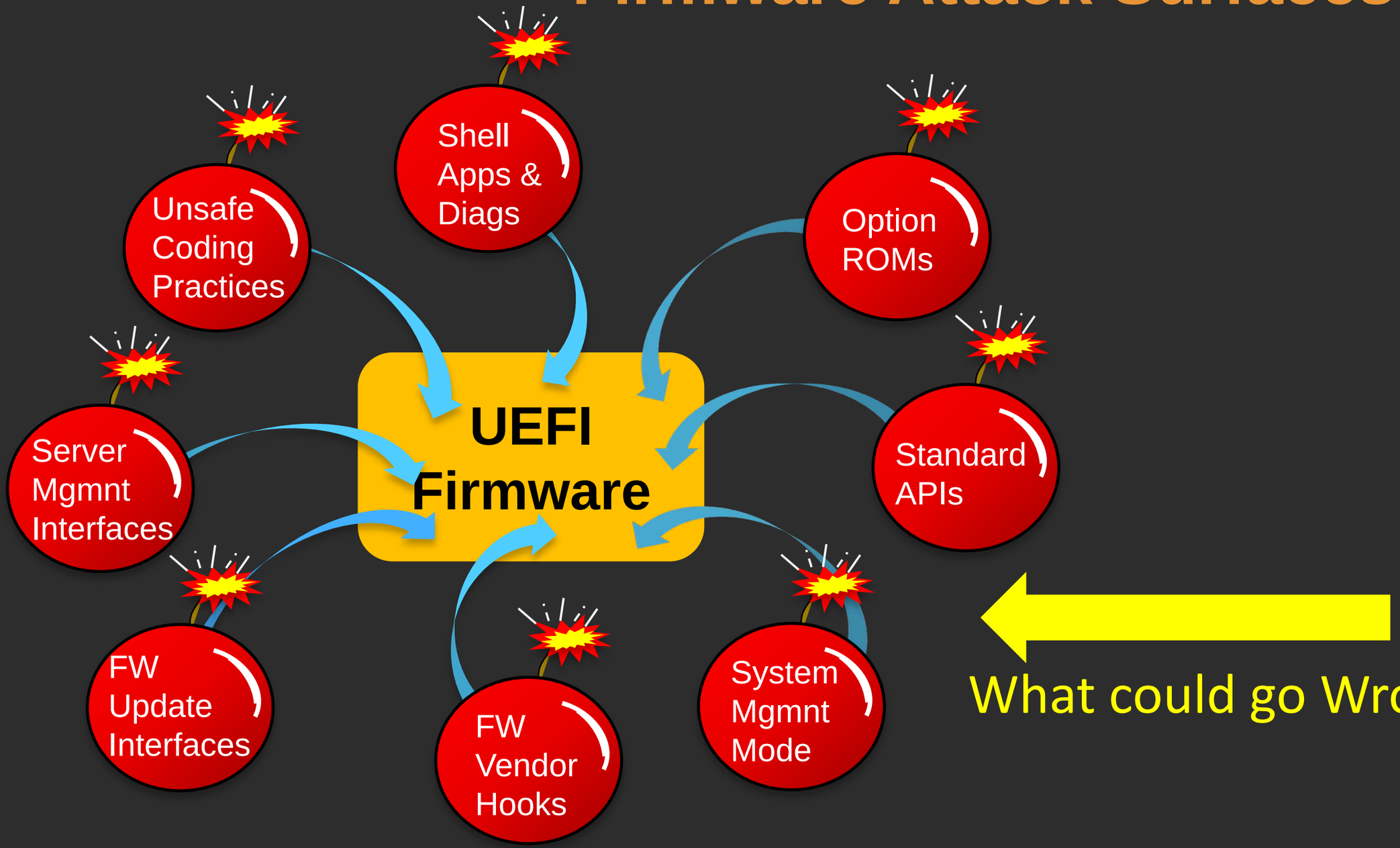
UEFI & Platform Initialization Task Flow



[1] S-CRTM - static core root of trust for measurement

tianocore

Firmware Attack Surfaces



tianocore

Goals of security architecture and assets that are protected

NIST SP 800-33 – IT Security Objectives

Availability, Integrity, Confidentiality, Accountability, Assurance

The goal of information technology security:

Enable an organization to meet all of its mission/business objectives by implementing systems with due care consideration of IT-related risks to the organization, its partners and customers.

tianocore

Goals of security architecture and assets that are protected

NIST SP 800-33 – IT Security Objectives

Availability, Integrity, Confidentiality, Accountability, Assurance

Confidentiality	• Protect against unauthorized access
Integrity	• Protection of Content & Quality
Availability	• Ensure access
Accountability	• Also Authenticity - traced uniquely to the source entity
Assurance	• Guarantee on the correctness, Non-repudiation

All of these objectives are interdependent with Assurance

What to build & defend – Rationale for a threat model

"*My house is secure*" is almost meaningless

- Against a burglar? Against a meteor strike? A thermonuclear device?

"*My system is secure*" is almost meaningless

- Against what? To what extent?

Threat modeling is a process to define the goals and constraints of a (software) security solution

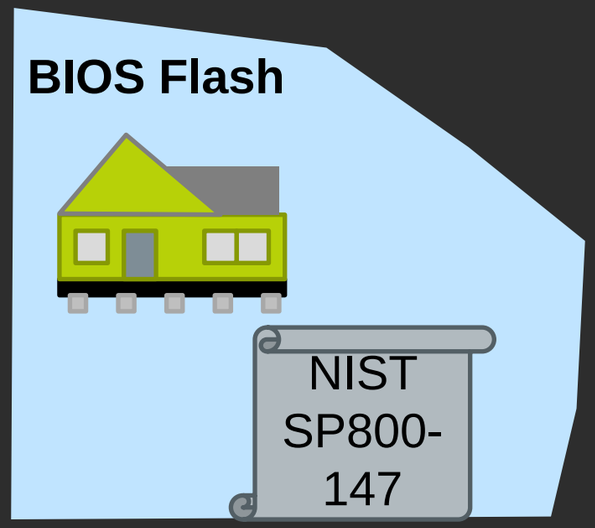
- Translate user requirements to security requirements

We use threat modeling for our UEFI / PI codebase

- We believe the process and findings are applicable to driver implementations as well as UEFI implementations in general

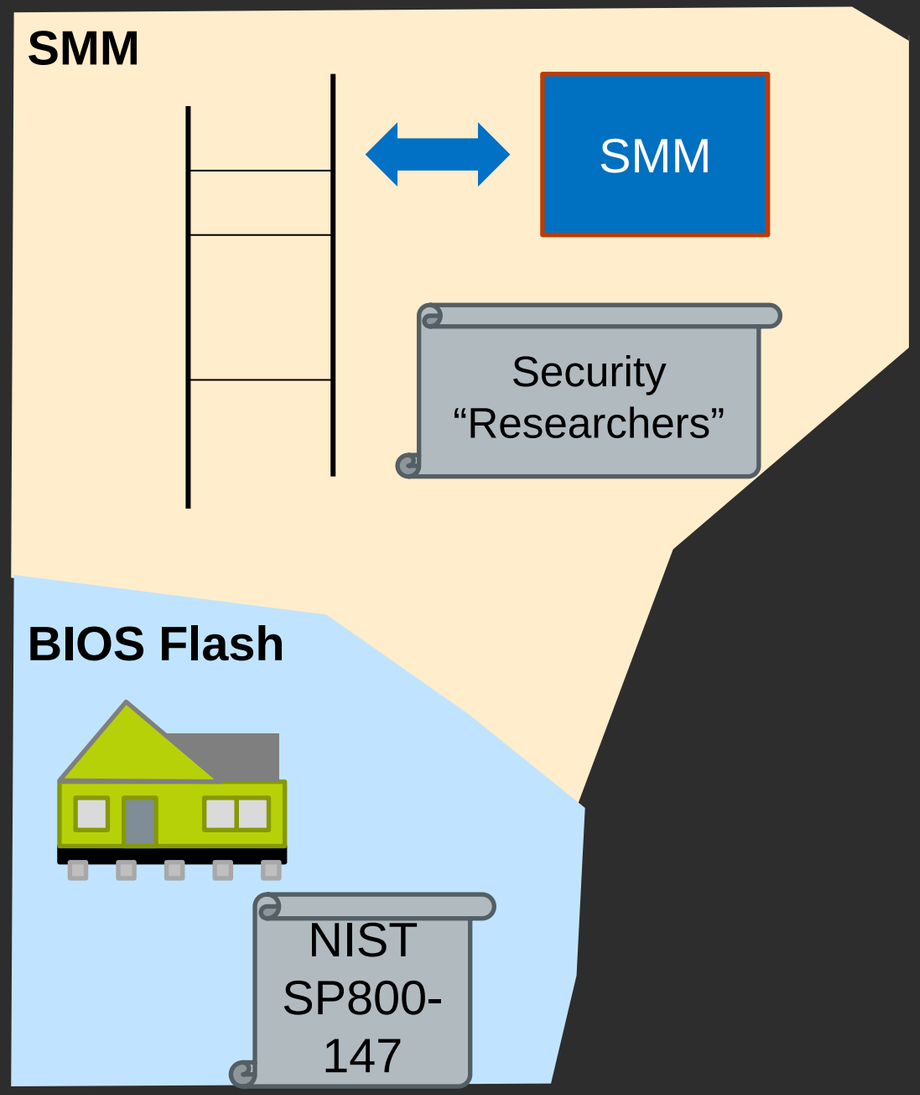
We Need to protect our Assets from Threats

We don’t always get to choose our Assets



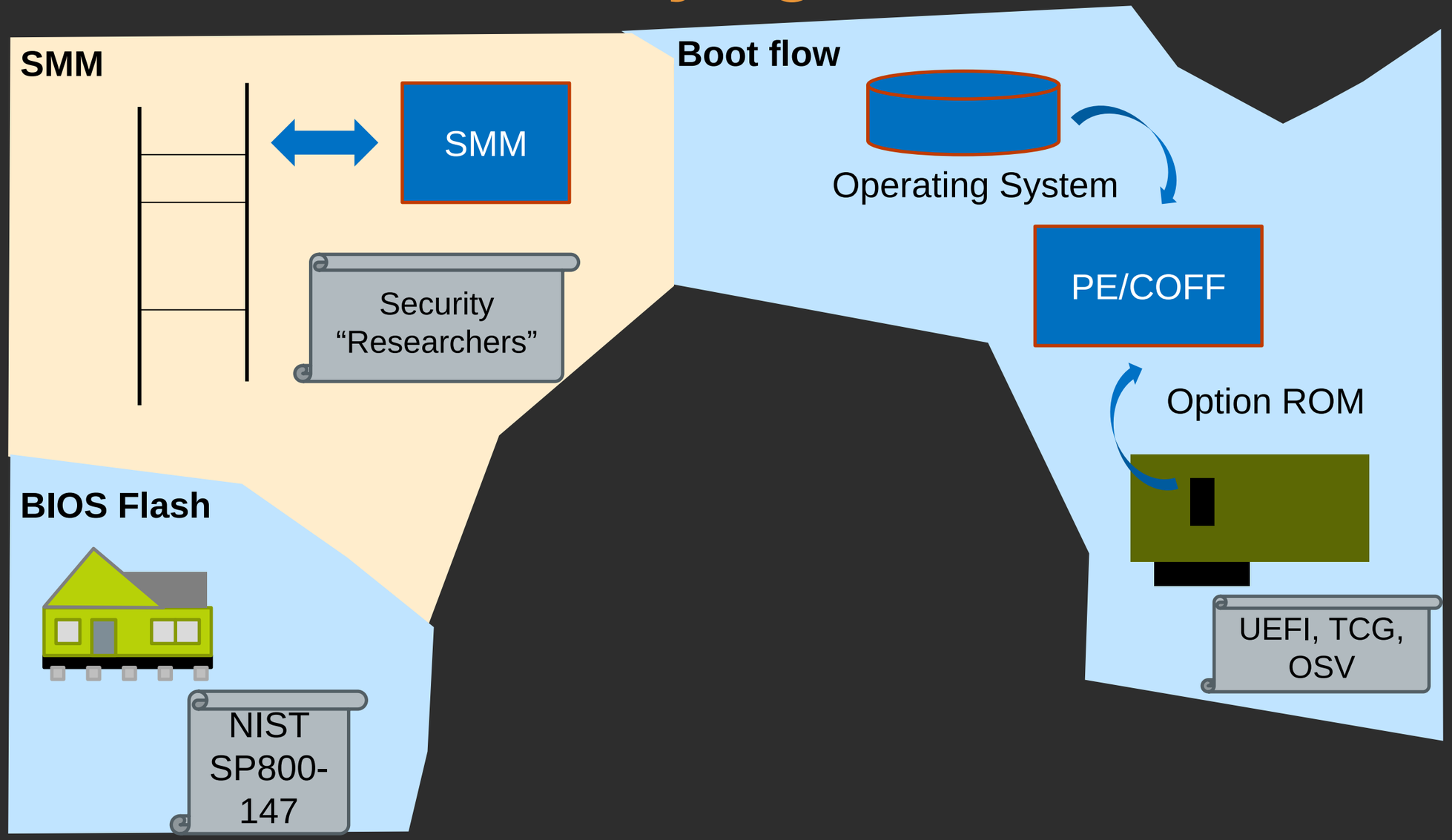
Source: UEFI open Platforms_Vincent.ppt

We don’t always get to choose our Assets



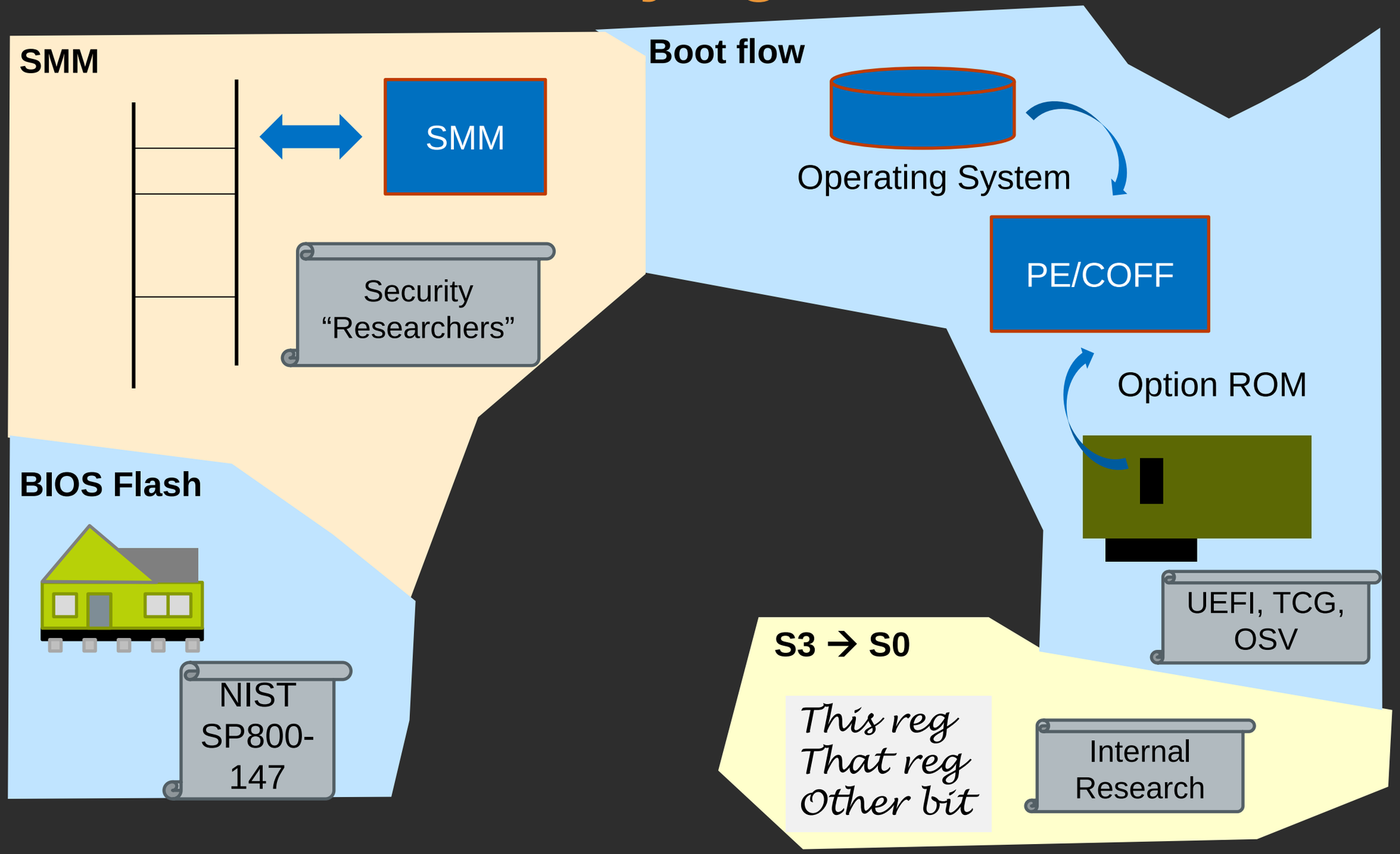
Source: UEFI open Platforms_Vincent.ppt

We don’t always get to choose our Assets



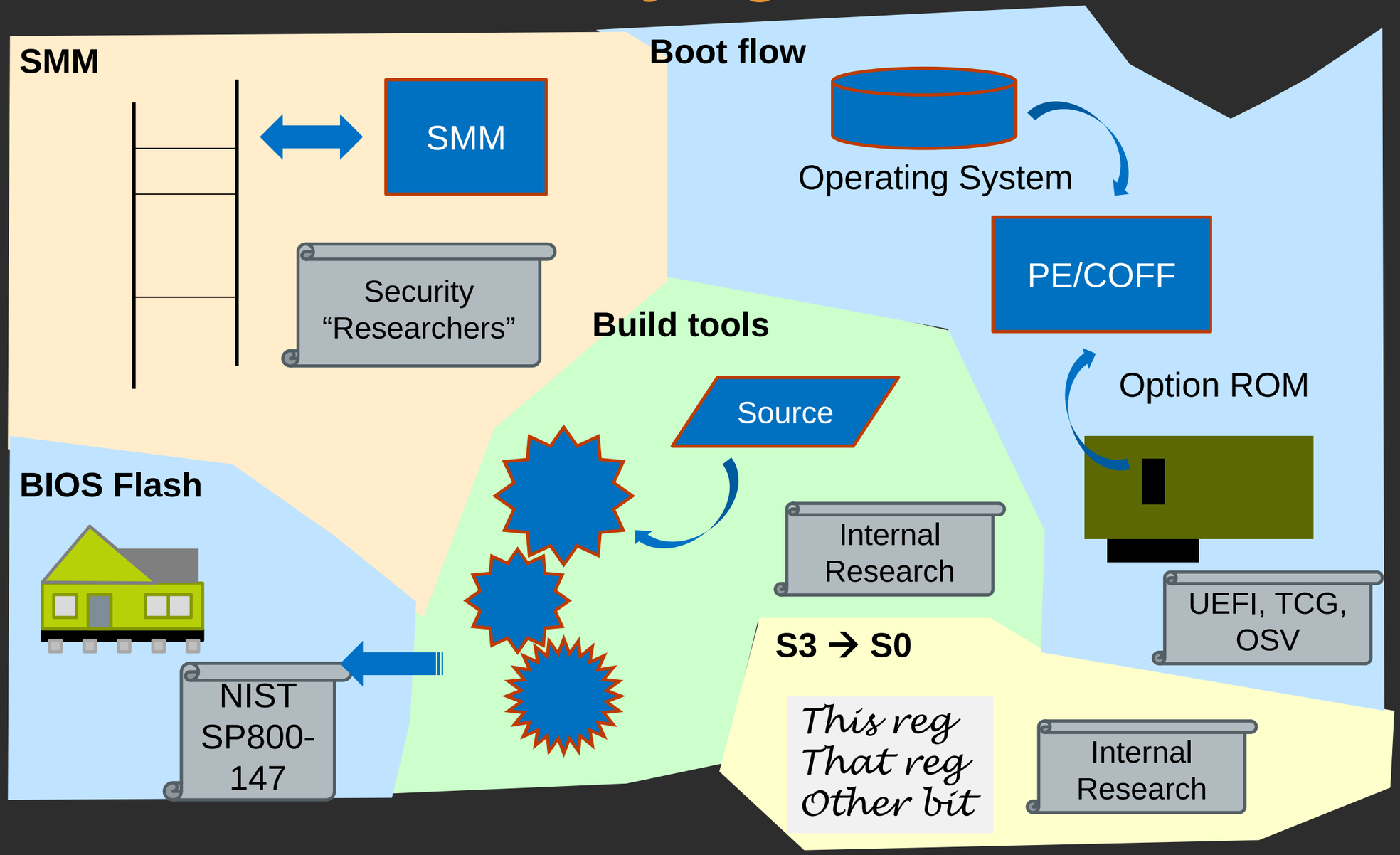
Source: UEFI open Platforms_Vincent.ppt

We don’t always get to choose our Assets



Source: UEFI open Platforms_Vincent.ppt

We don’t always get to choose our Assets

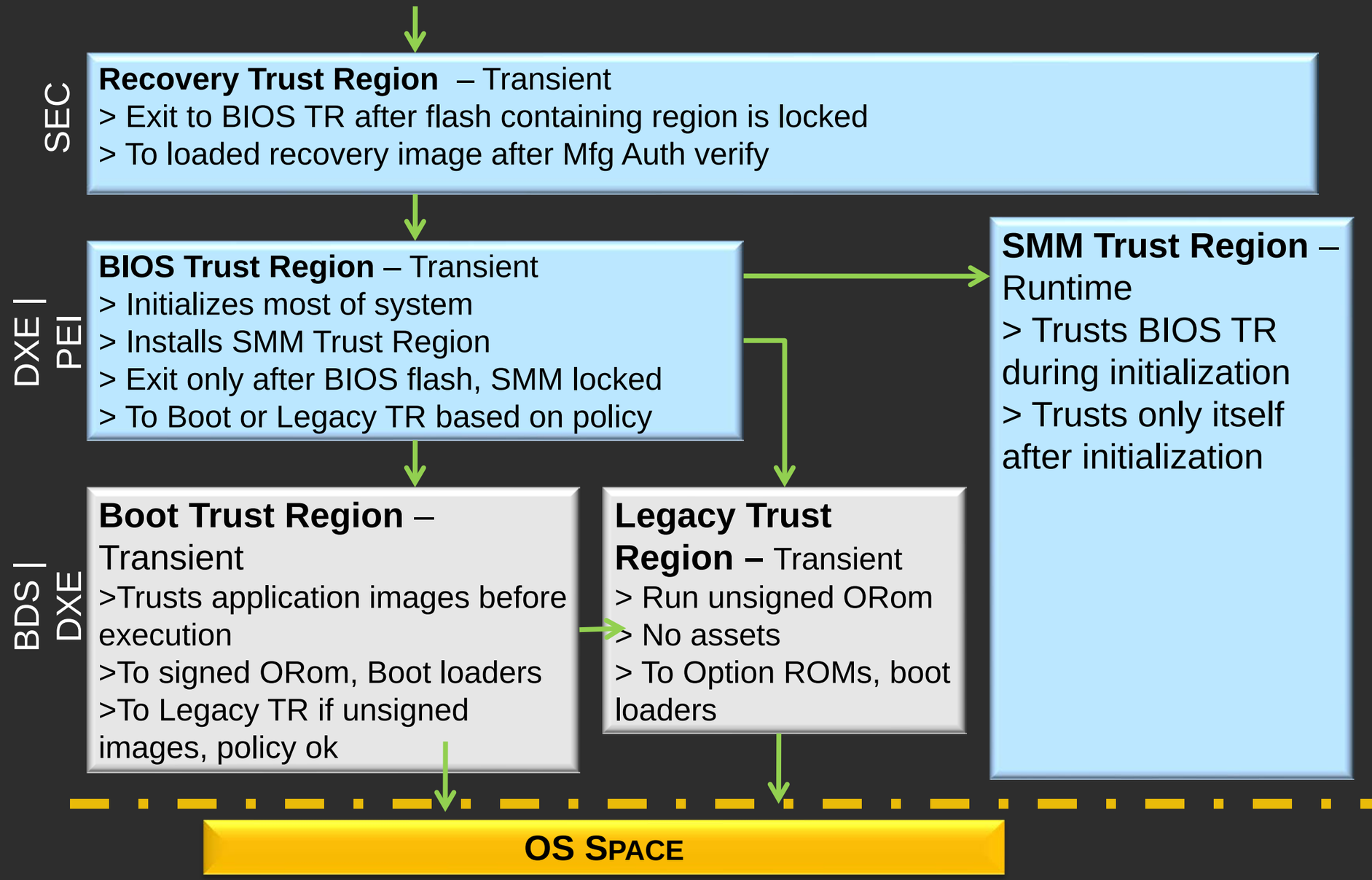


Source: UEFI open Platforms_Vincent.ppt

Baseline: Boot trust regions

A Threat Model (TM) defines the security assertions and constraints for a product

- Assets:
- Threats:
- Mitigations:



Threat Model with Examples

Boot Trust Regions and Assets

Asset	Example Threats	Mitigations	Checks
Firmware/BIOS Flash Contents	CIH attack: erase boot block	SPI locks, descriptor	CHIPSEC
SMM	Callouts; Access to SMM	TSEG, SMRR, SMM_CODE_CHK	CHIPSEC
Execution During Boot Flow	Run malware in Op ROM	Secure Boot, DMA protection	Manual testing (eg CHIPSEC)
S3 Boot Script & S3 Resume Boot Flow	Resume reconfiguration losing locks	SMM Lock Box	Manual testing (eg CHIPSEC)
UEFI Variables (includes Authenticated & non-Authenticated)	Variable store full; Content change	Attributes, Lock Protocol	Manual testing (eg CHIPSEC)
Etc . . .			

Summary

Platform Firmware Security – Why is it important?

Why is platform firmware Security important	Prevent low level attacks that could “brick” the system
UEFI boot flow with the threat model	Identify where UEFI firmware is vulnerable and define a Threat Model

tianocore

SECURITY TECHNOLOGIES

Security Technologies Overview

tianocore

BOOT SECURITY

BOOT SECURITY TECHNOLOGIES

Hardware Root of Trust	Boot Guard, Intel® TXT
Measured Boot	Using TPM[1] to store hash values
Verified Boot	Boot Guard + UEFI Secure Boot

[1]TPM – Trusted Platform Module

Resources: https://firmwaresecurity.com/2015/07/29/survey-of-boot-security-technologies/

HARDWARE ROOT OF TRUST

Boot Guard

CPU verifies signature

Verification occurs before BIOS starts

Hash of signature is fused in CPU

Verification

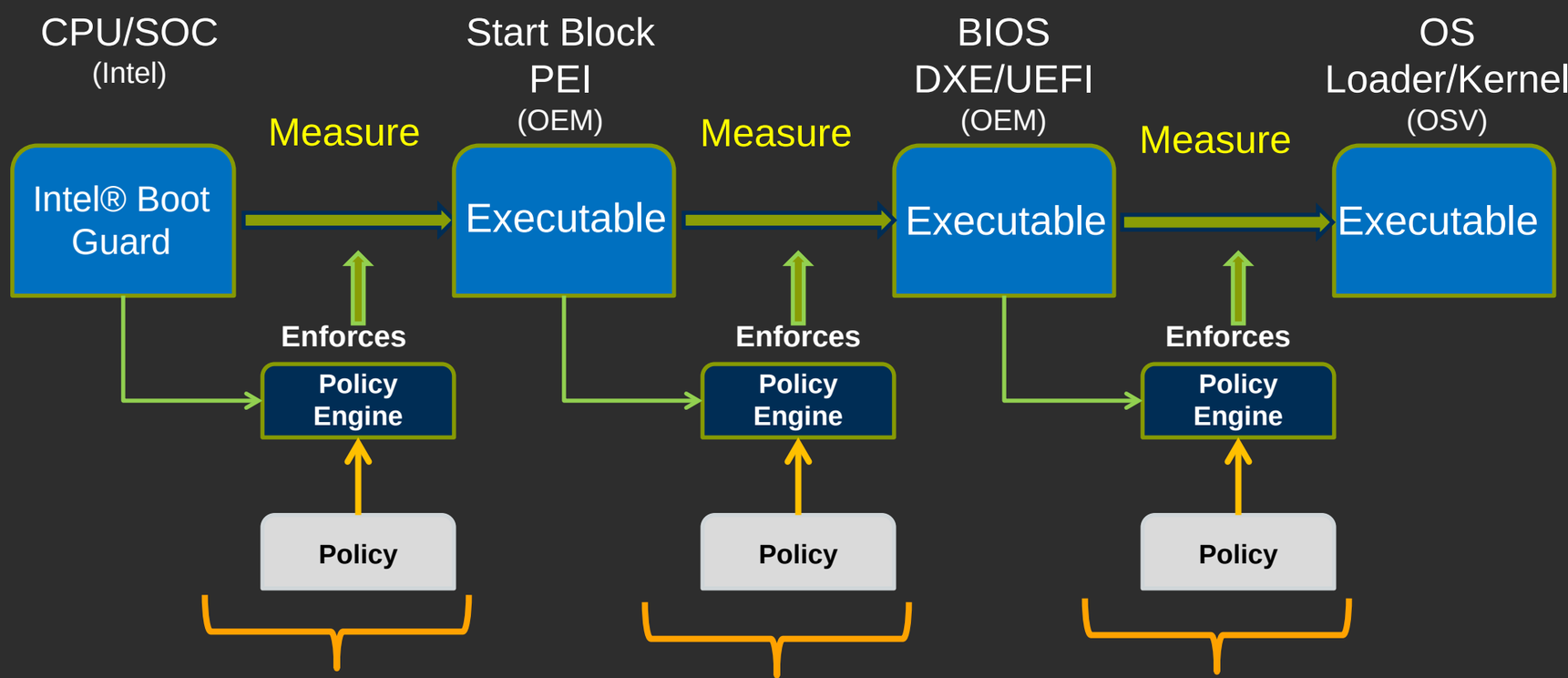
Intel® TXT

Uses a Trusted Platform Module (TPM) & cryptographic

Provides Measurements

Measurements

Full Verified Boot Sequence



Intel® Device Protection Technology with Boot Guard

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

OEM PI Verification Using PI Signed Firmware Volumes

Vol 3, section 3.2.1.1
of PI 1.3 Specification

OEM UEFI 2.4 Secure Boot

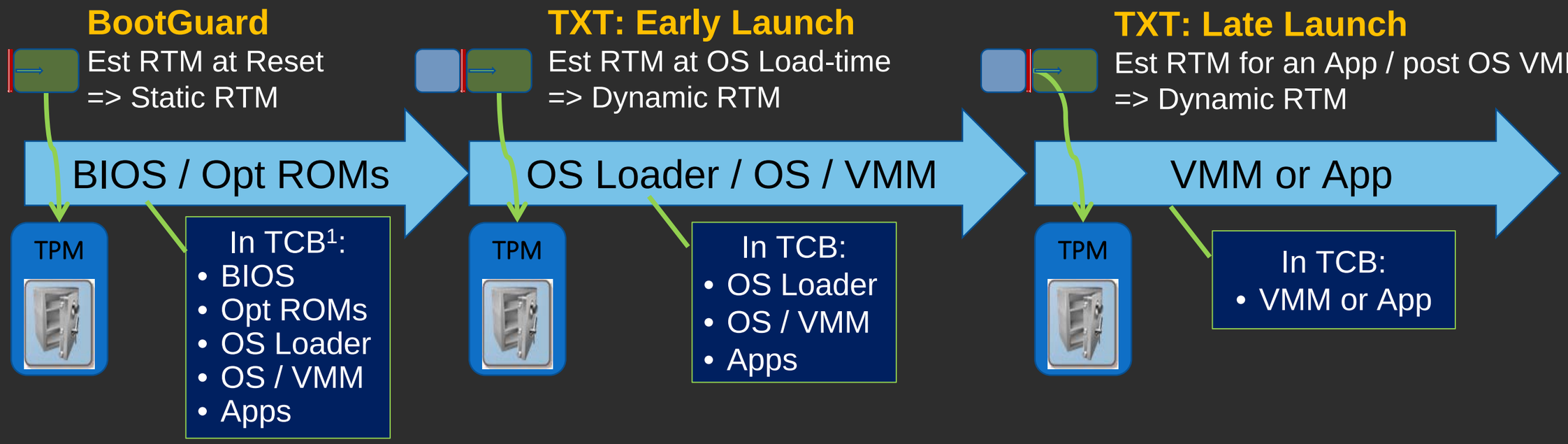
Chapter 27.2 of
The UEFI 2.4
Specification

https://firmware.intel.com/sites/default/files/resources/Platform_Security_Review_Intel_Cisco_White_Paper.pdf

Boot Guard and TXT

Both features rooted in Trusted Computing.

Establishing a Trusted Environment (Intel provides 3 modes)
Starts with Root of Trust for Measurement (RTM)

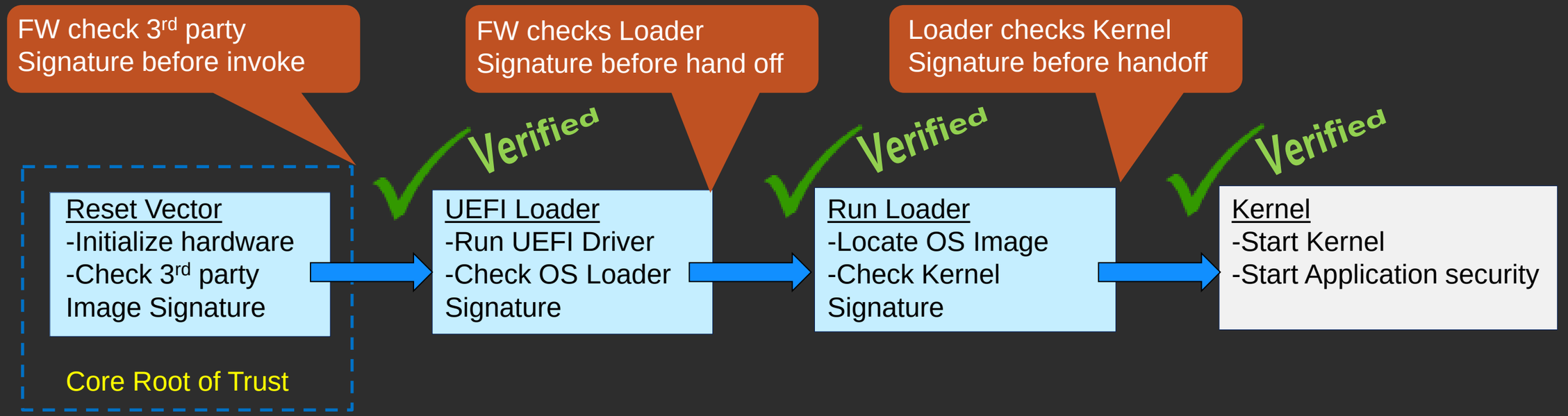


1. Trusted computing base (TCB)

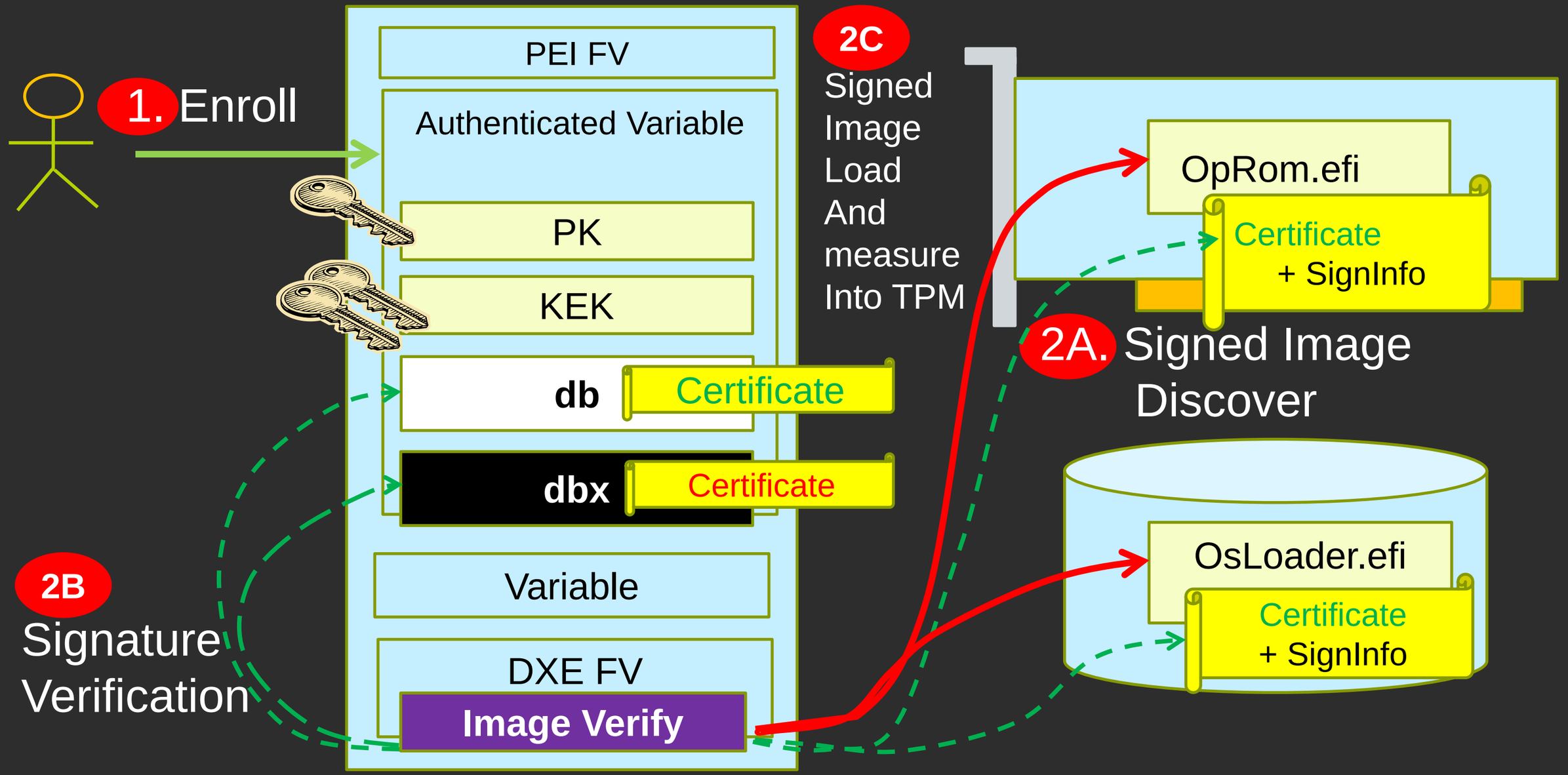
UEFI SECURE BOOT

Software ID checking during every step of the boot flow:

1. UEFI System FW (updated via secure process)
2. Add-In Cards (signed UEFI Option ROMs)
3. OS Boot Loader (checks for “secure mode” at boot)



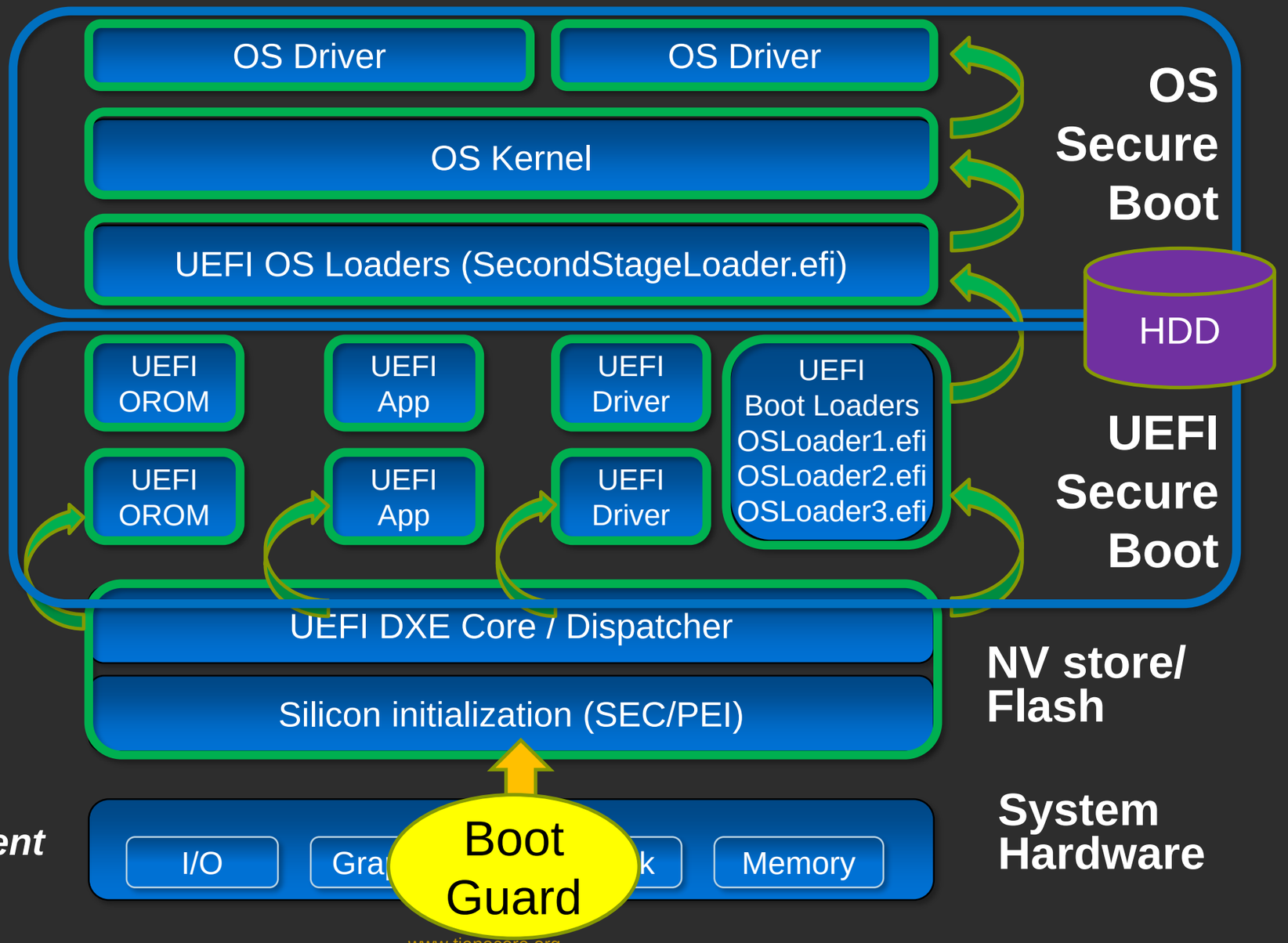
UEFI Secure Boot Flow



tianocore

End to End Platform Integrity

Reference:
Where do I 'sign' up?



Intel® Device Protection Technology with Boot Guard – Secure *Boot Policy Enforcement*

tianocore

FLASH DEVICE

tianocore

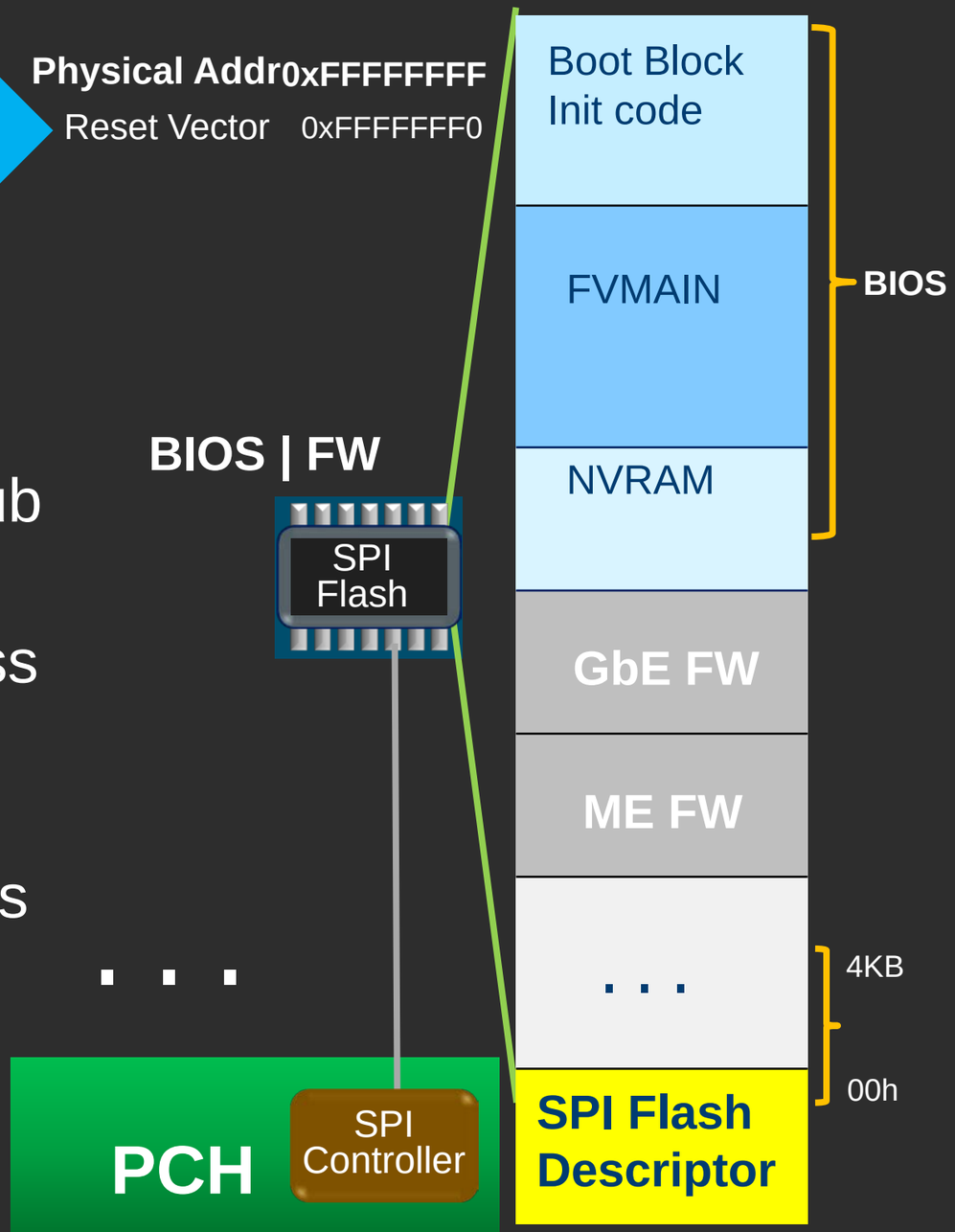
BIOS Firmware uses SPI Flash

Serial Peripheral Interface (SPI)

- Access controlled by the Peripheral Controller Hub (PCH)
- Flash Memory - Direct access to physical address space is programed through SPI MMIO registers
- SPI Flash Descriptor - Access Control table defines which device (CPU, ME, GbE) can access which regions

Flash Descriptor has to be write-protected

ME – Manageability Engine
GbE – Gigabit Ethernet



tianocore

System Flash Security

Chipset (SPI controller) based protections

- SMM based BIOS Write Protection: write-protects entire BIOS/Firmware region from software other than SMI handler firmware executing in SMM
- SPI Protected Range registers(PR0-PR4) : read/write protection of SPI flash regions based on Flash Linear Address for program register access
- Flash Descriptor based access control: defines read/write access to each flash region by each Host Device

Firmware may use SPI flash chips write protection(WP#)

PR0-PR4 defined in SPI MMIO

Lock SPI - BIOS Range is not protected - Threats

- BIOS Write Protections often still not properly enabled on many systems
- SMM based write protection of entire BIOS region is often not used: BIOS_CONTROL[SMM_BWP]
- If SPI Protected Ranges (mode agnostic) are used (defined by **PR0-PR4** in SPI MMIO), they often don't cover entire BIOS & NVRAM
- Some platforms use SPI device specific write protection but only for boot block/startup code or SPI Flash descriptor region

Mitigations:

- Set `BIOS_CONTROL[SMM_BWP] <- 1`
- Program SPI flash protected ranges (**PRx**) to cover BIOS range

References: Persistent BIOS Infection (used flashrom on legacy BIOS), Evil Maid Just Got Angrier, BIOS Chronomancy, A Tale Of One Software Bypass Of Windows 8 Secure

tianocore

FIRMWARE SECURE UPDATE

Solving Firmware Update

Reliable update story

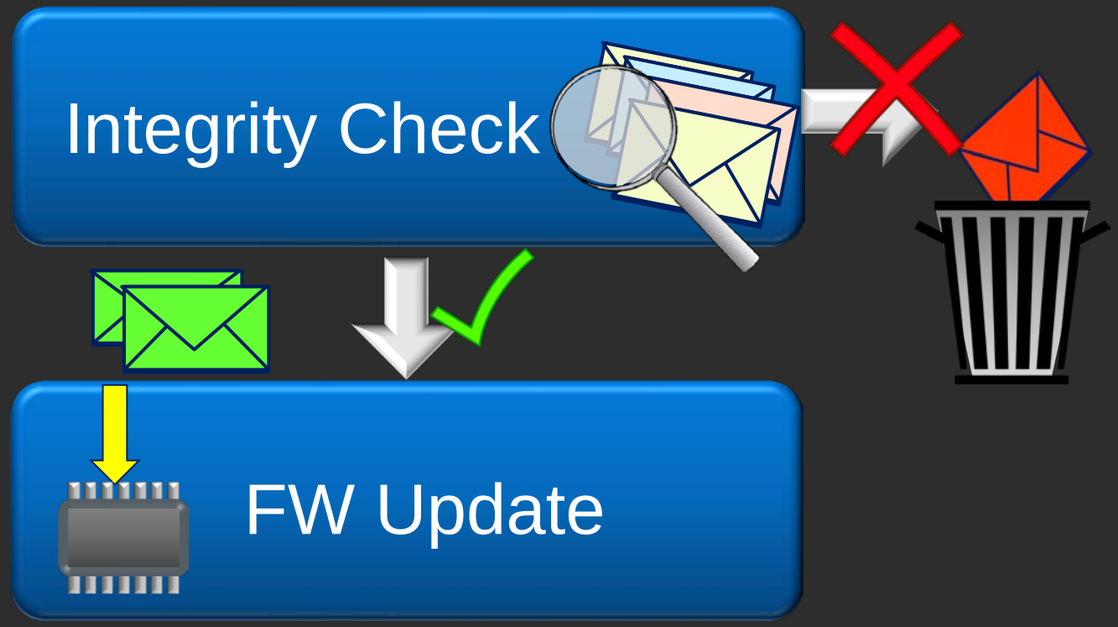
- Fault tolerant
- Scalable & repeatable

Refrences [6] at: UEFI, Open Platforms and ppt

Solving Firmware Update

Reliable update story

- Fault tolerant
- Scalable & repeatable



Refrences [6] at: UEFI, Open Platforms and ppt

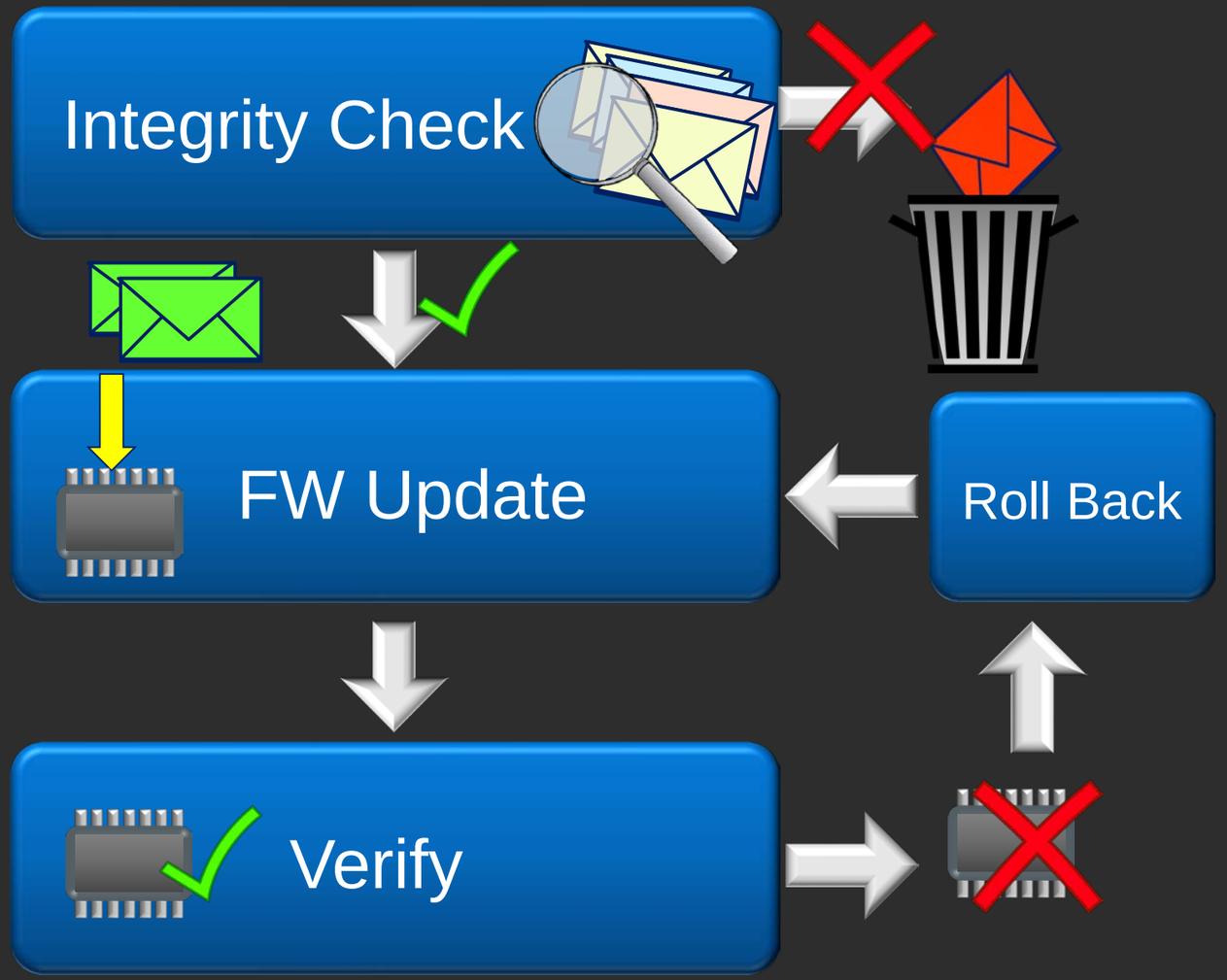
Solving Firmware Update

Reliable update story

- Fault tolerant
- Scalable & repeatable

How can UEFI Help?

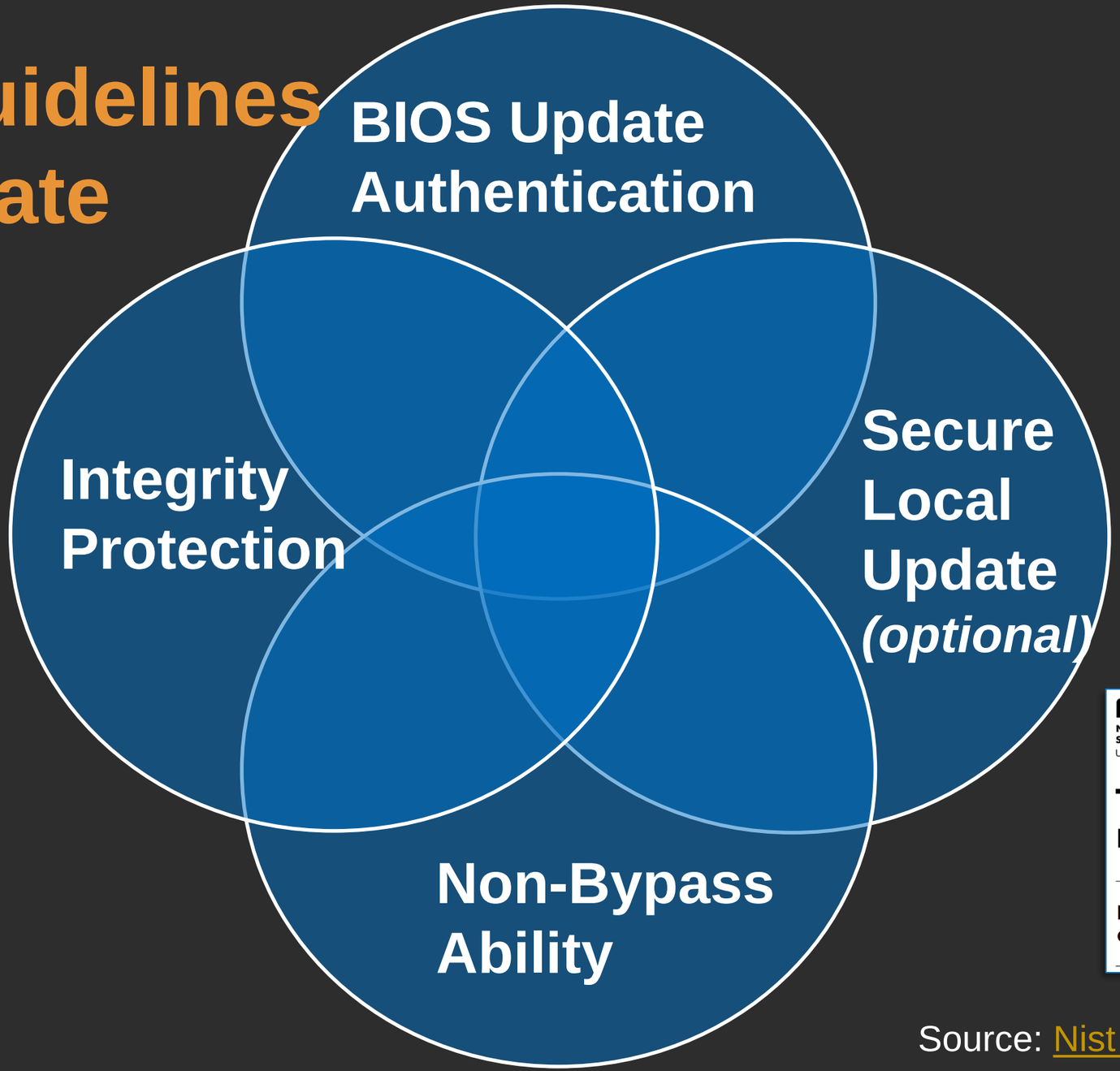
- Capsule model for binary delivery
- Bus / Device Enumeration
- Managing updates via
  - EFI System Resource Table
  - Firmware Management Protocol
  - Capsule Signing

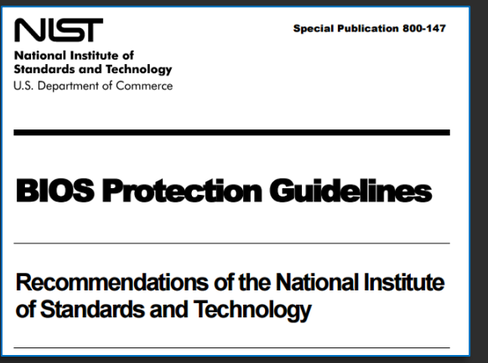


Refrences [6] at: UEFI, Open Platforms and ppt

tianocore

Security Guidelines - BIOS Update




NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-147

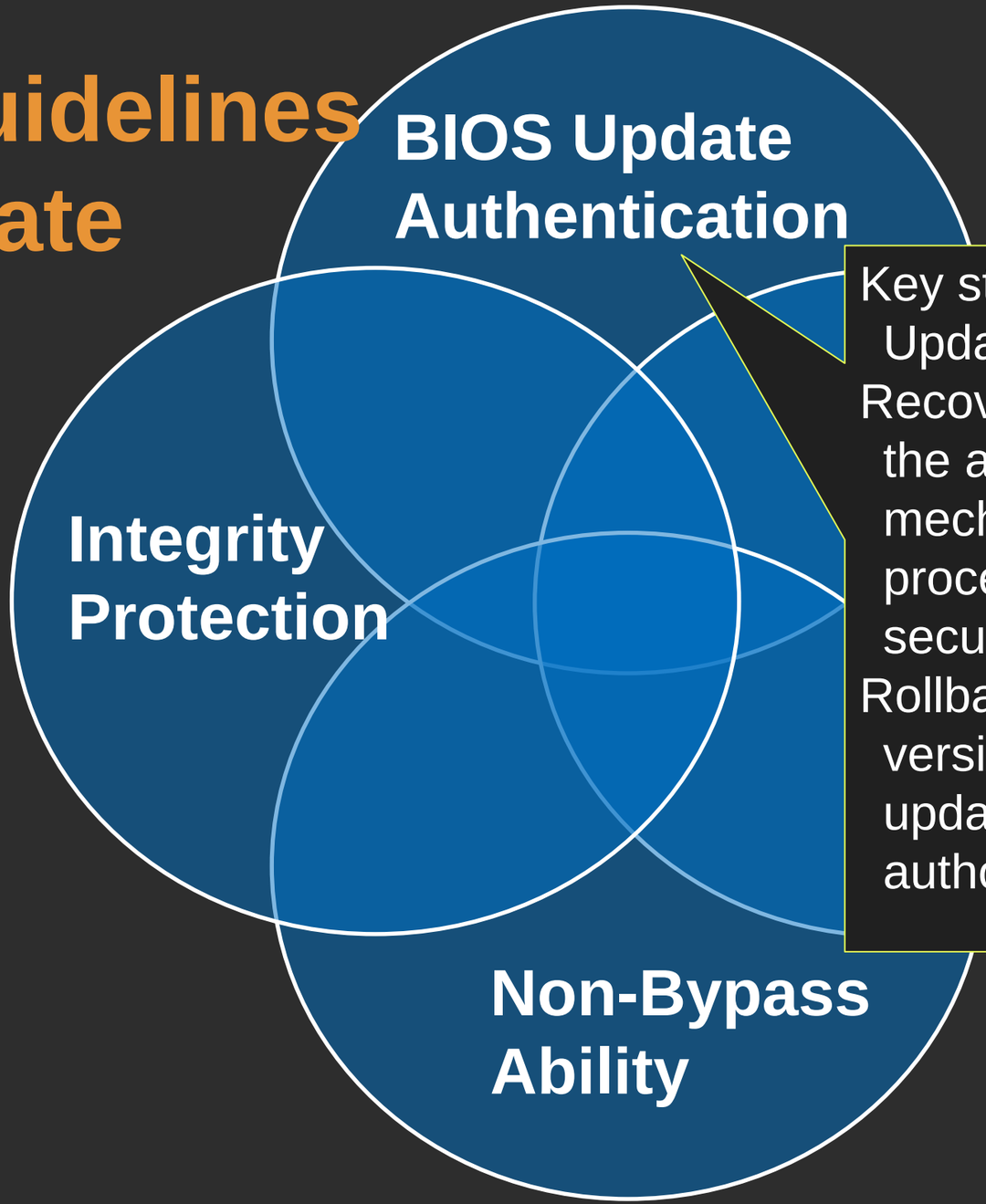
BIOS Protection Guidelines

Recommendations of the National Institute of Standards and Technology

Source: Nist SP 800-147 .pdf

tianocore

Security Guidelines - BIOS Update



Key storage in Root of Trust for Update (RTU)

Recovery mechanisms shall also use the authenticated update mechanism unless the recovery process meets the guidelines for a secure local update.

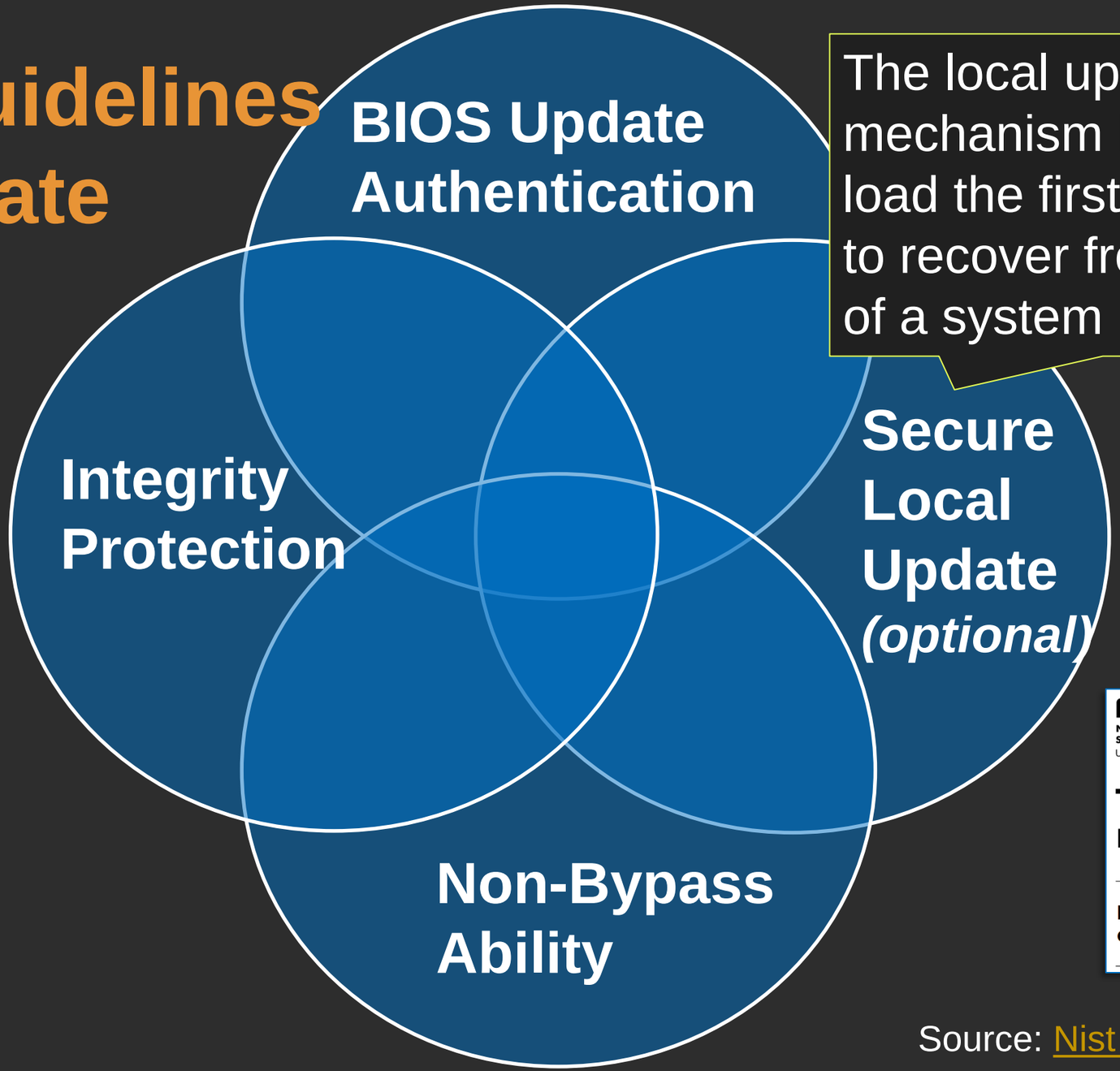
Rollbacks of the BIOS to an earlier version are permitted only if the update or rollback has been authorized by the organization.

Recommendations of the National Institute of Standards and Technology

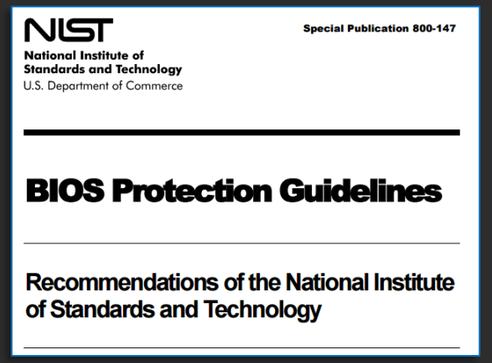
Source: Nist SP 800-147 .pdf

tianocore

Security Guidelines - BIOS Update



The local update mechanism be used only to load the first BIOS image or to recover from a corruption of a system BIOS.


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-147

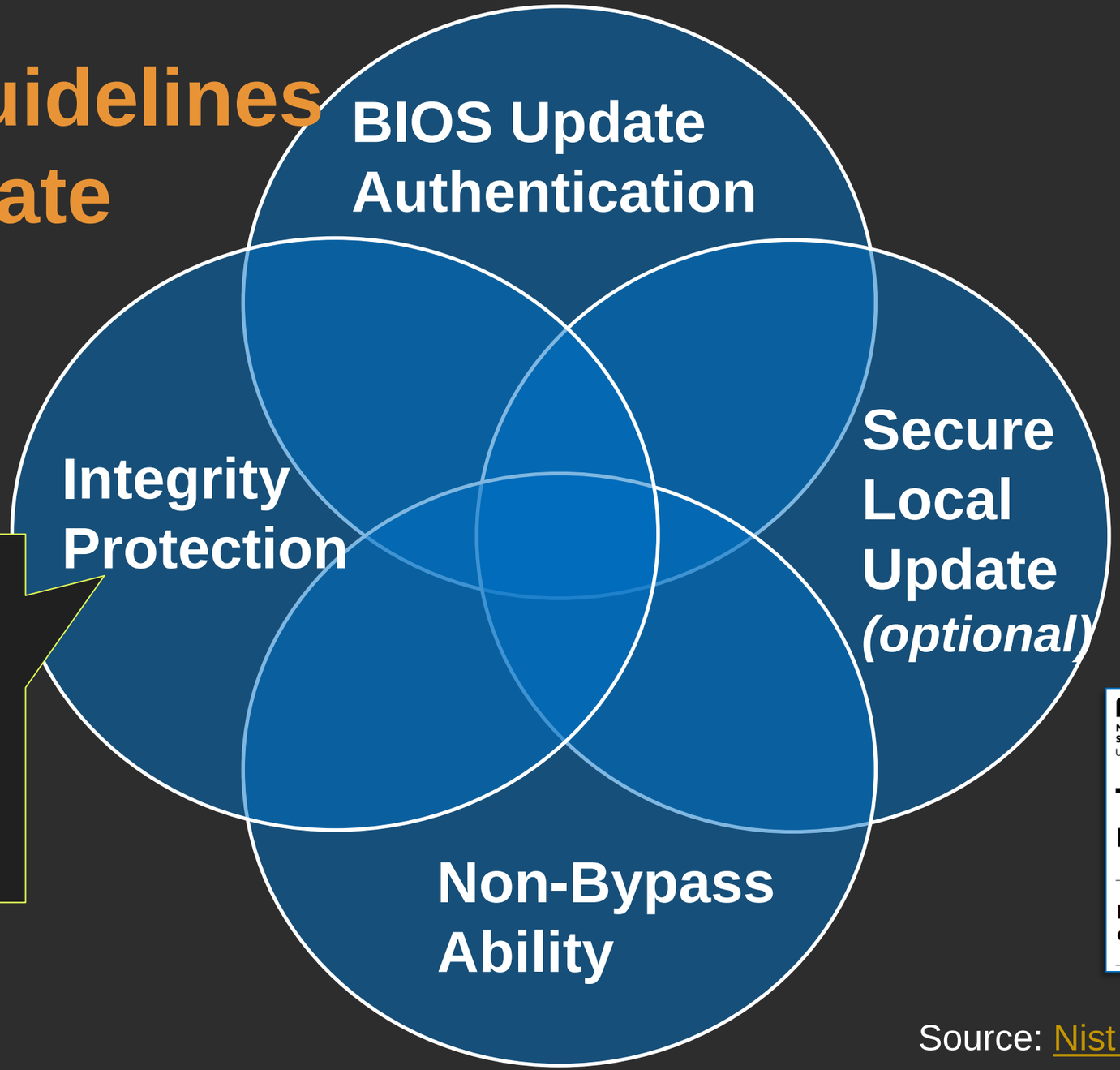
BIOS Protection Guidelines

Recommendations of the National Institute of Standards and Technology

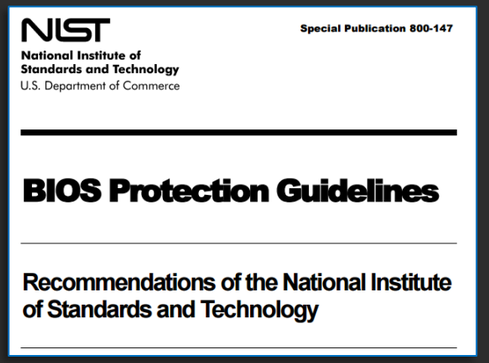
Source: Nist SP 800-147 .pdf

tianocore

Security Guidelines - BIOS Update



The RTU and the system BIOS shall be protected from unintended modification.


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-147

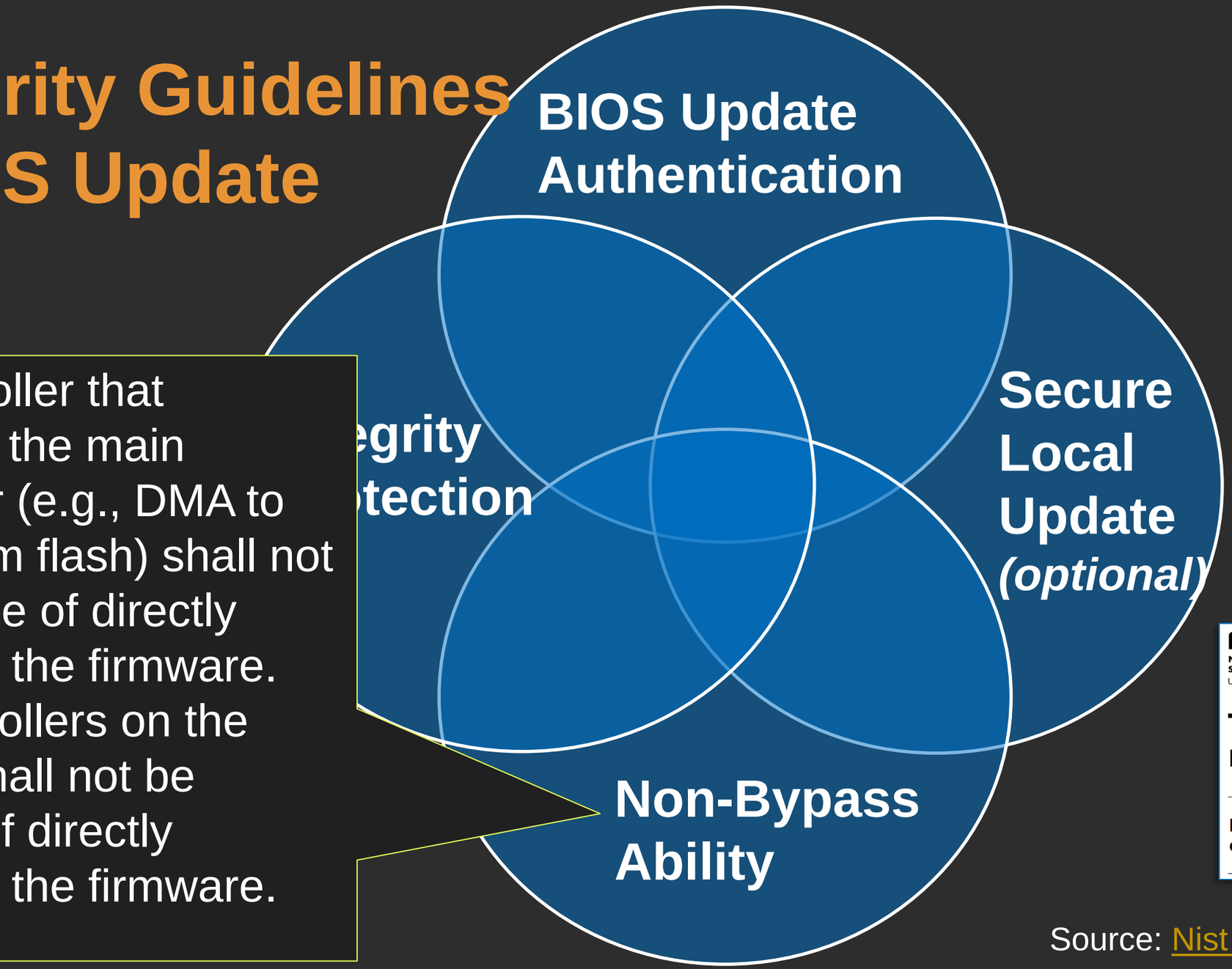
BIOS Protection Guidelines

Recommendations of the National Institute of Standards and Technology

Source: Nist SP 800-147 .pdf

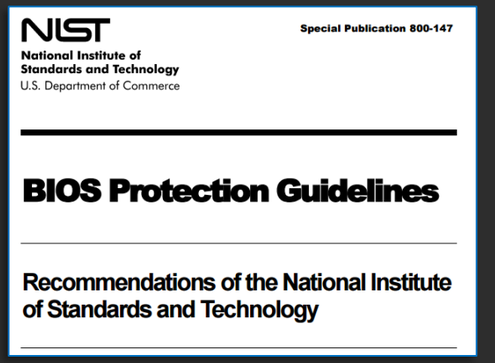
tianocore

Security Guidelines - BIOS Update



Bus Controller that bypasses the main processor (e.g., DMA to the system flash) shall not be capable of directly modifying the firmware.

Microcontrollers on the system shall not be capable of directly modifying the firmware.


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-147

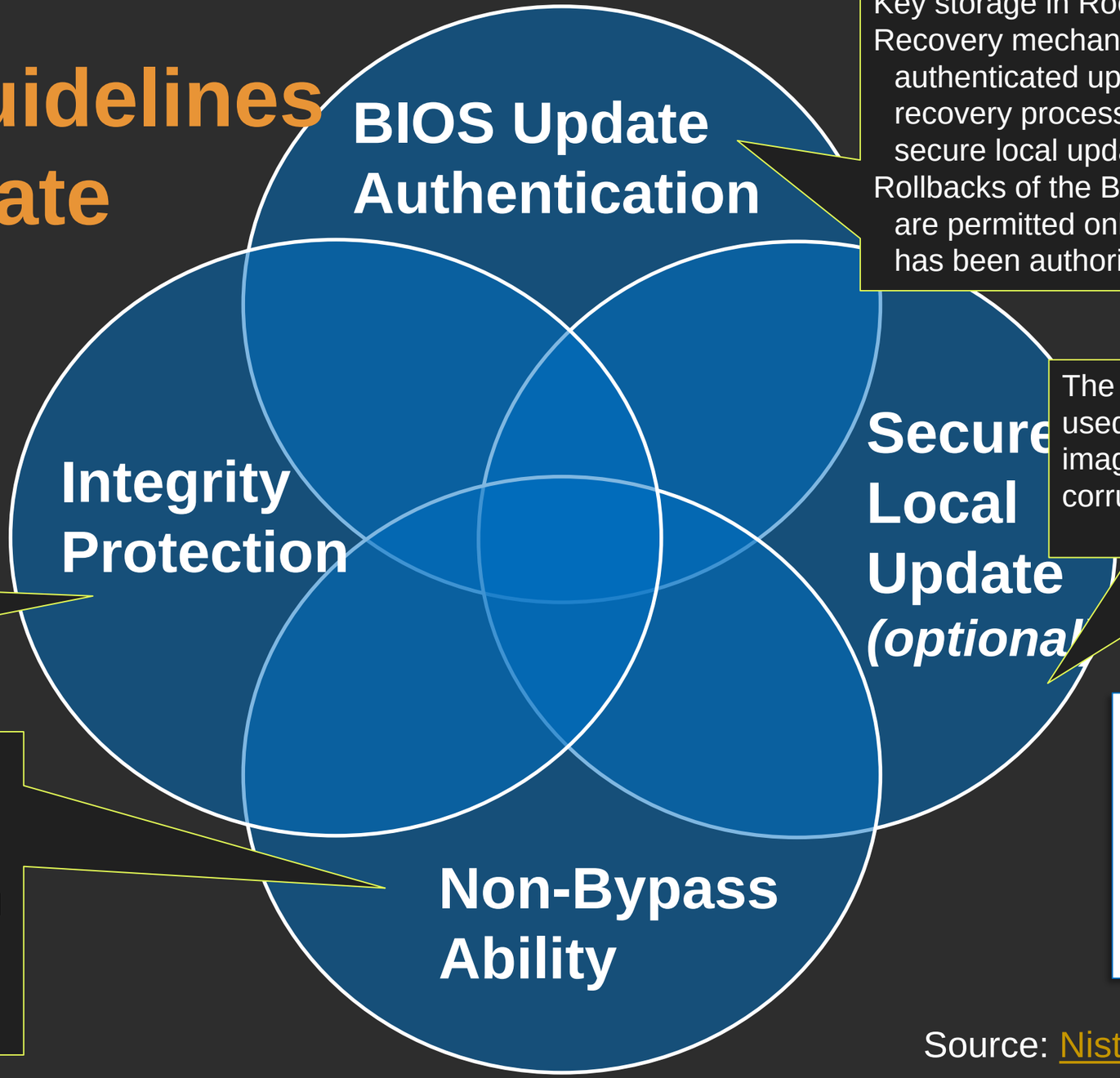
BIOS Protection Guidelines

Recommendations of the National Institute of Standards and Technology

Source: Nist SP 800-147 .pdf

tianocore

Security Guidelines - BIOS Update

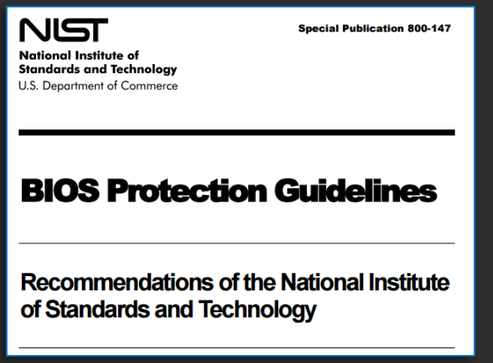


Key storage in Root of Trust for Update (RTU)
Recovery mechanisms shall also use the authenticated update mechanism unless the recovery process meets the guidelines for a secure local update.
Rollbacks of the BIOS to an earlier version are permitted only if the update or rollback has been authorized by the organization.

The local update mechanism be used only to load the first BIOS image or to recover from a corruption of a system BIOS.

The RTU and the system BIOS shall be protected from unintended modification.

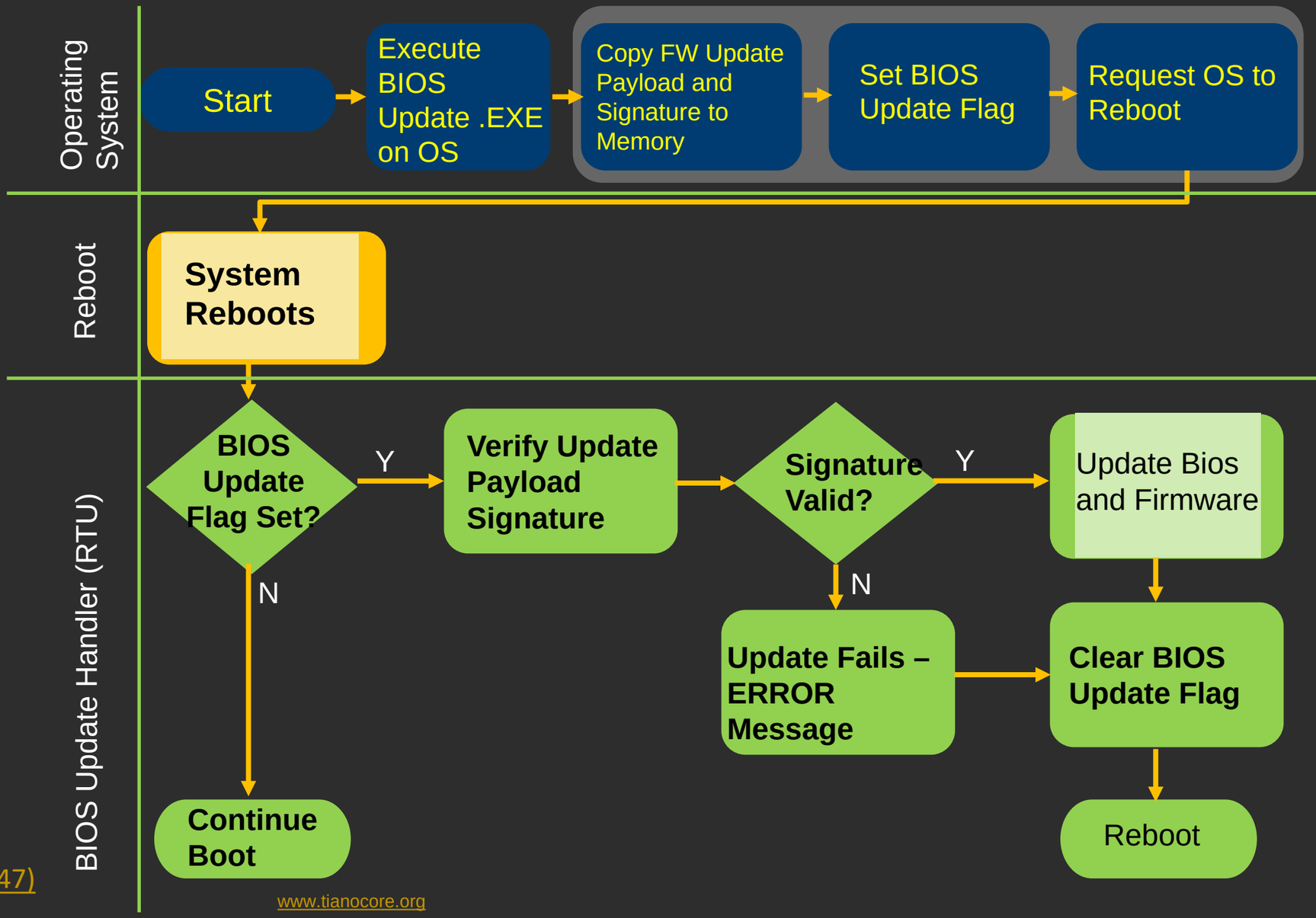
Bus controller that bypasses the main processor (e.g., DMA to the system flash) shall not be capable of directly modifying the firmware.
Microcontrollers on the system shall not be capable of directly modifying the firmware.


NIST
National Institute of Standards and Technology
U.S. Department of Commerce
Special Publication 800-147
BIOS Protection Guidelines
Recommendations of the National Institute of Standards and Technology

Source: Nist SP 800-147 .pdf

Signed Firmware Update

- RTU protected by flash locking mechanisms at the hardware level
- BIOS key store includes the full public key used to verify the signature of all BIOS and firmware updates
- Capsule Update with UEFI FMP



RTU - BIOS Root of Trust for Update
FMP- Firmware Management Protocol

Source: Dell Signed Firmware Update (NIST 800-147)

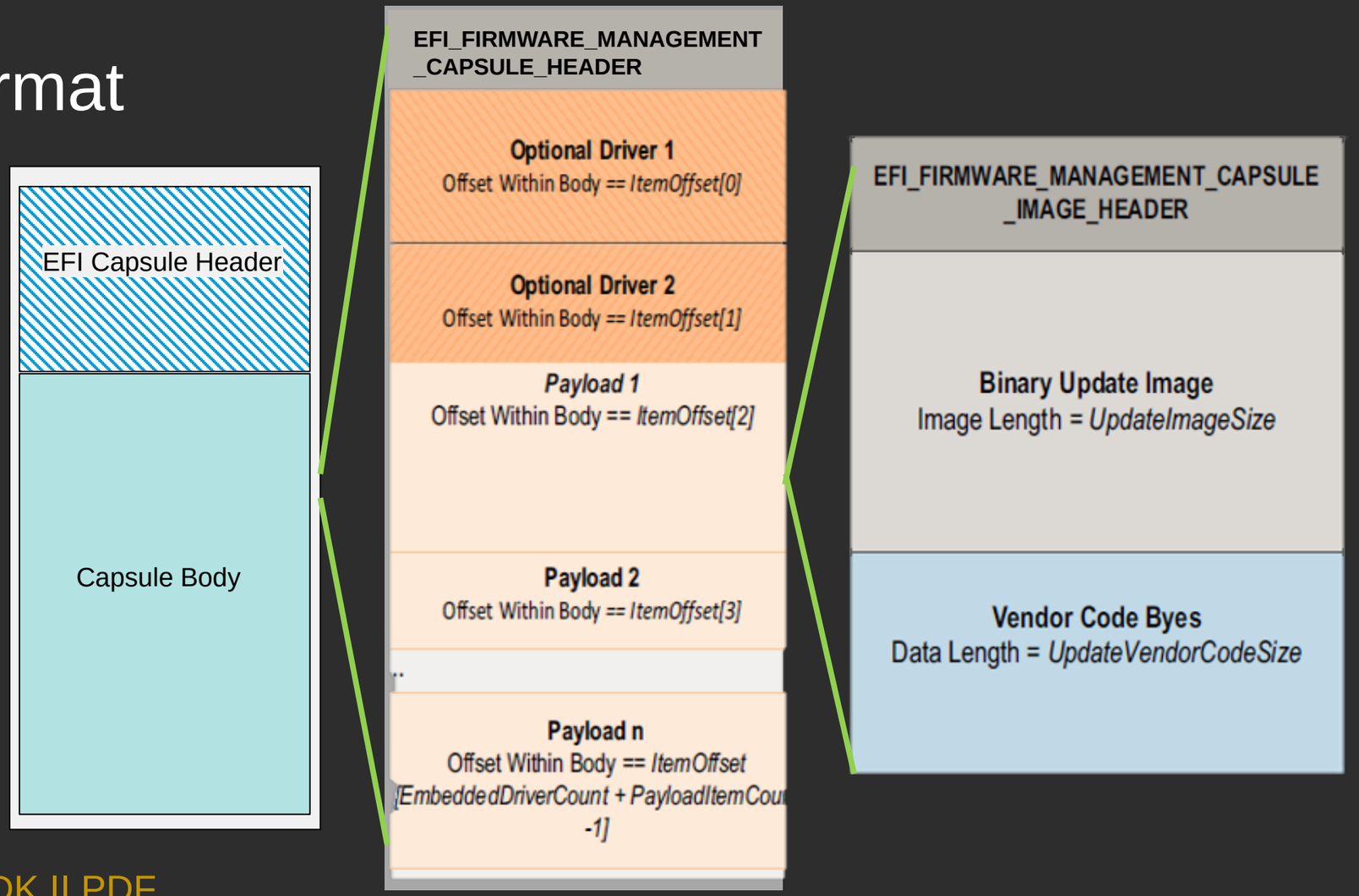
UEFI Capsule Update – Firmware Management Protocol (FMP)

FMP capsule image format

- Update FMP drivers
- FMP payloads

  binary update image and optional vendor code

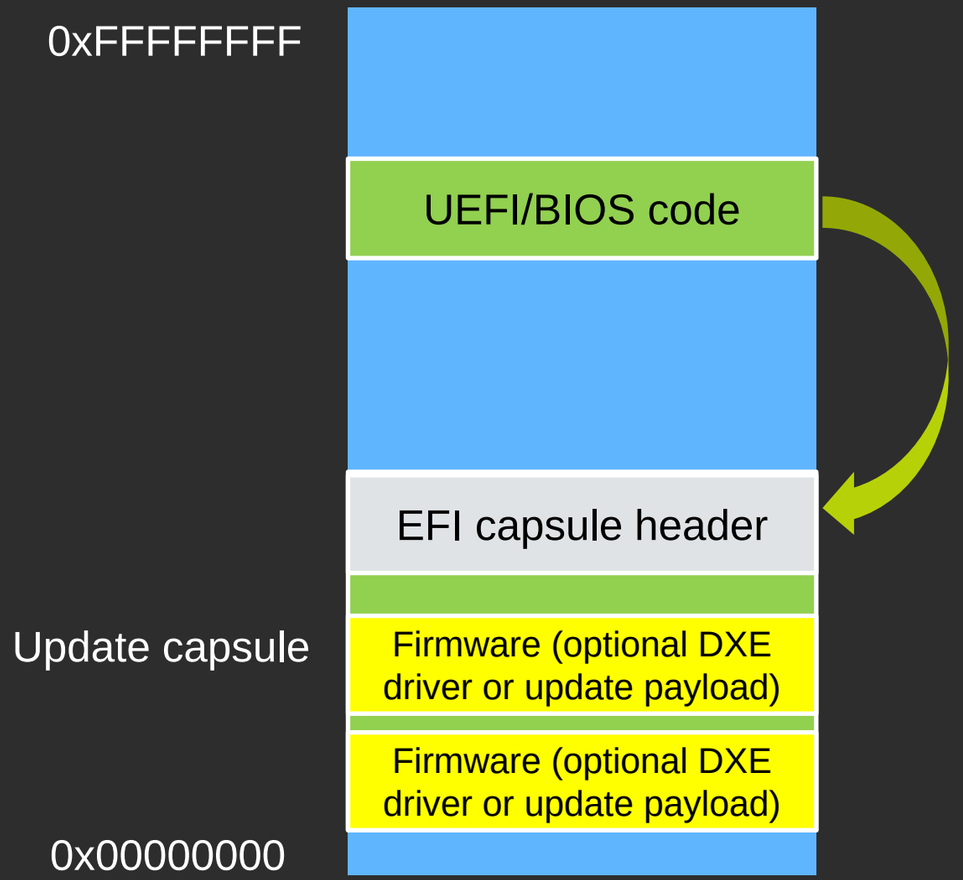
The platform may consume a FMP protocol to update the firmware image



Source: Capsule Update & Recovery EDK II PDF

UEFI Firmware Secure “Capsule” Update

Capsule update is a runtime service used to update UEFI FW

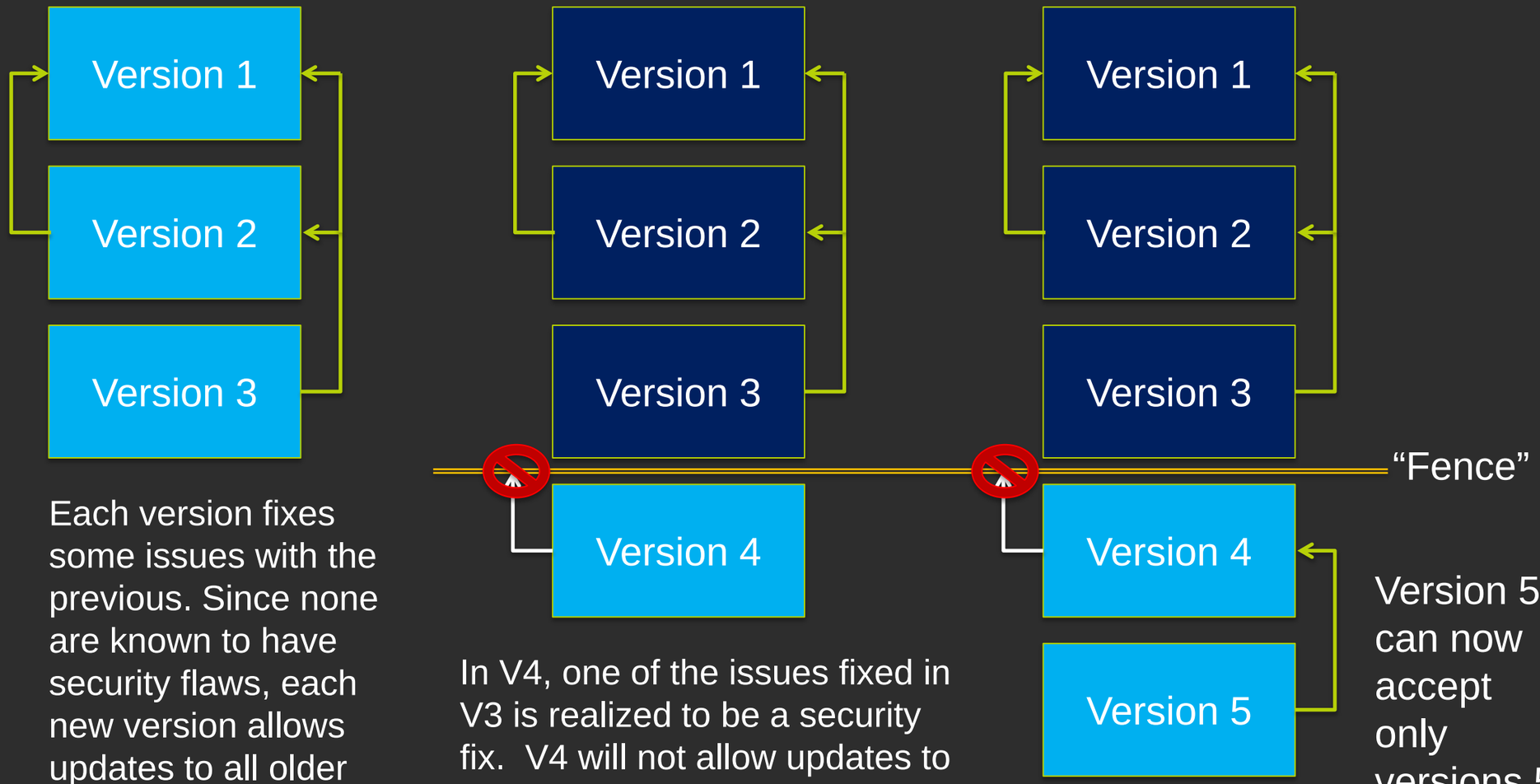


1. Update is initiated by update application/OS run-time
2. Update application stores update “capsule” in DRAM or HDD on ESP (e.g. **`\EFI\CapsuleUpdate`**)
3. Upon reboot or S3 resume, FW finds and parses update capsule
4. After FW verifies digital signature of the capsule, FW writes new BIOS FV(s) to SPI flash memory

Source: UEFI Spec Version 2.4 Facilitates Secure Update UEFI Summerfest – July 15-19, 2013

ESP = EFI System Partition

Firmware Update Rollback Protection



Each version fixes some issues with the previous. Since none are known to have security flaws, each new version allows updates to all older versions.

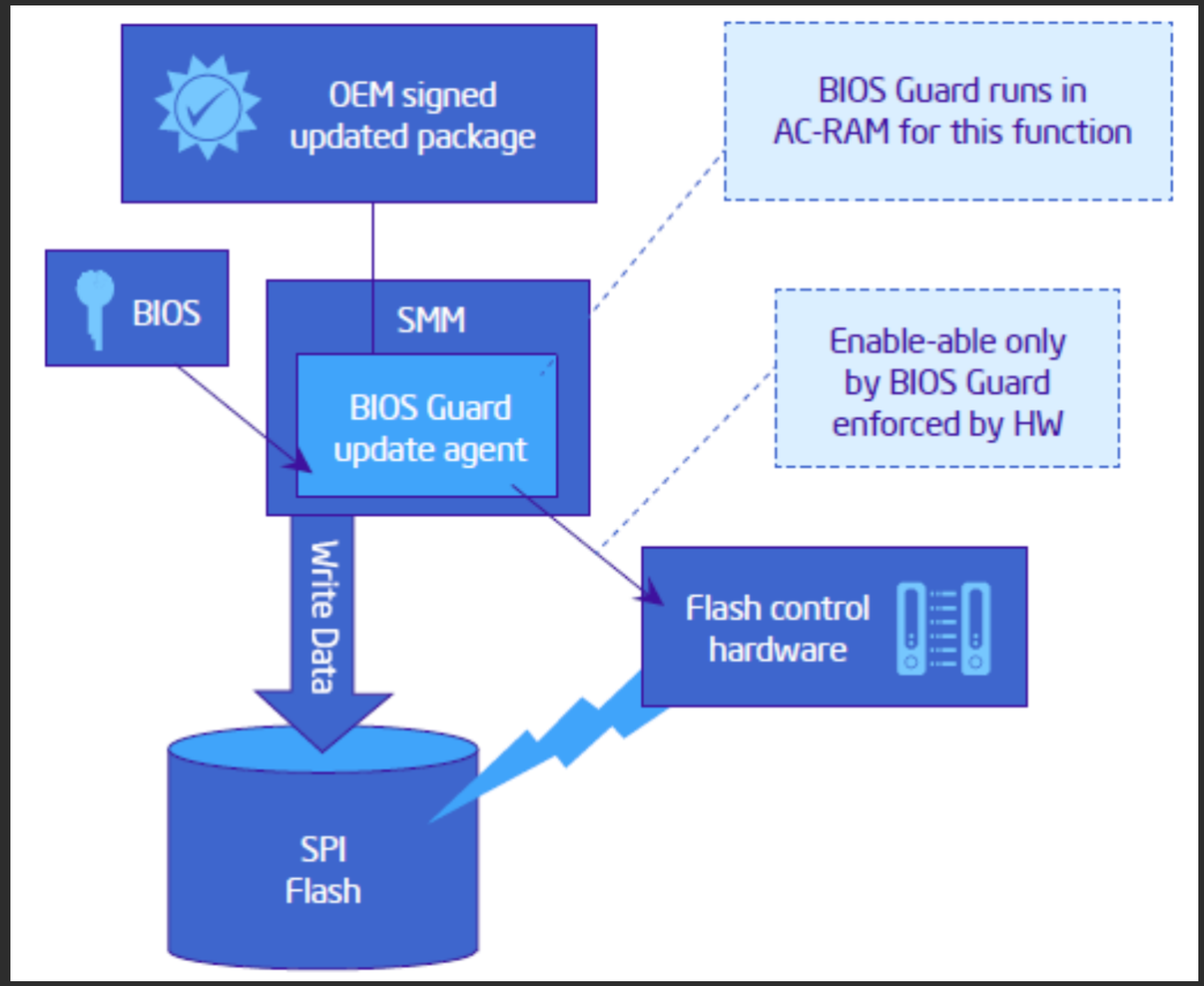
In V4, one of the issues fixed in V3 is realized to be a security fix. V4 will not allow updates to earlier versions, even V3 since it allows update to V2.

Version 5 can now accept only versions 5 and 4.

Hardware based System Firmware Update

BIOS Guard address SMM vulnerabilities by strengthening the update trust boundary

Example: Intel® Platform Protection Technology w/ BIOS Guard



AC-RAM- authenticated code module (ACM) RAM

Source: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/security-technologies-4th-gen-core-retail-paper.pdf

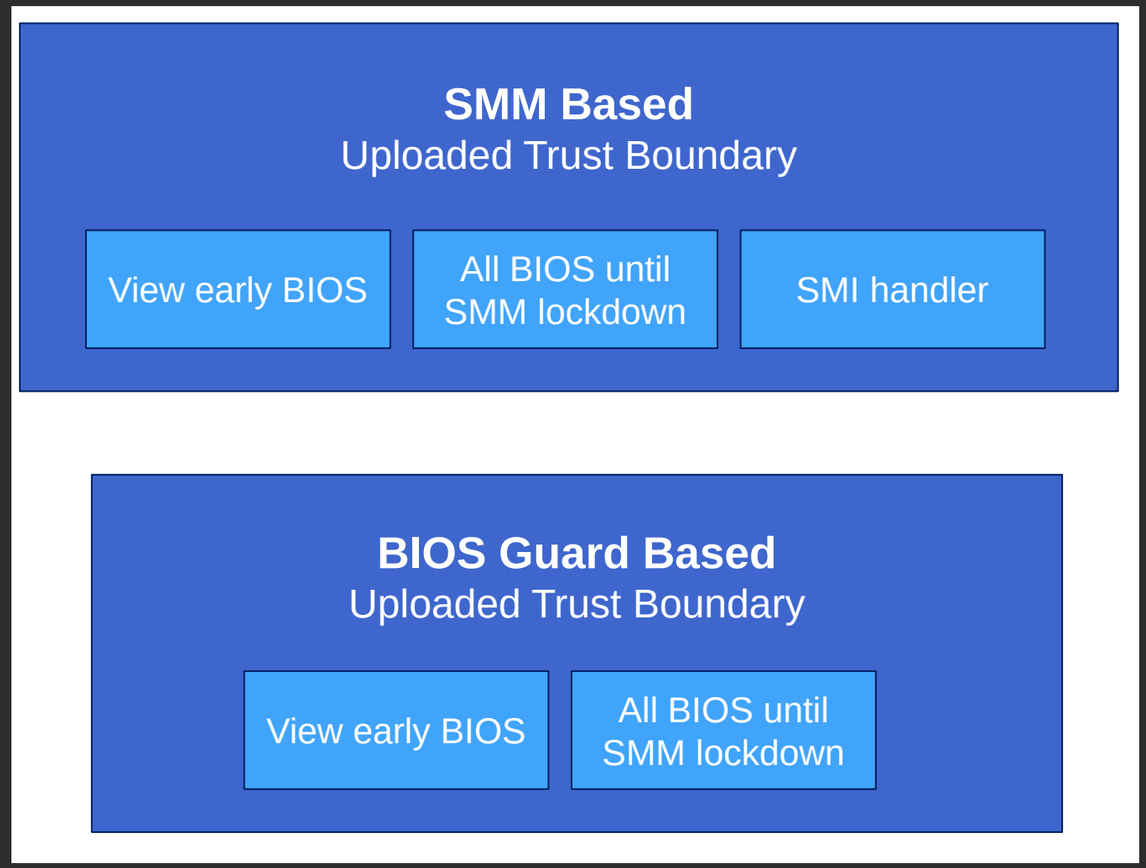
SMM BIOS Update Trust Boundary

- For runtime BIOS Update (e.g. on server platforms), all complex SMI handlers code is in the trust boundary of the firmware update
- Different systems have different SMI handlers which makes it difficult to ensure consistent security level of SMI code across all system and security level of firmware update
- BIOS Guard reduces SMI handler attack surface, using **one** signed BIOS Guard Authenticated Code Module (ACM)
- Platforms enabling BIOS Guard only need to use **one** module for a given processor generation

BIOS Guard Based Firmware Update

- BIOS Guard can update contents of the BIOS region in system SPI flash and EC firmware on EC flash memory
- BIOS Guard module is Authenticated Code Module (ACM) executing in internal processor AC RAM
- When BIOS Guard is enabled, only BIOS Guard module is able to write to system SPI flash memory
- BIOS Guard verifies the signature of a firmware update package signed by a platform manufacturer prior to writing to system SPI flash memory

Trust Boundary with BIOS Guard



EC – Embedded Controller

tianocore

When Is Secure Boot Actually Secure?

When all platform manufacturers…

- protect the UEFI BIOS from programmable SPI writes by malware,
- allow only signed UEFI BIOS updates,
- protect authorized update software,
- correctly program and protect SPI Flash descriptor,
- protect Secure Boot persistent configuration variables in NVRAM,
- implement authenticated variable updates,
- protect variable update API,
- disable Compatibility Support Module (Legacy BIOS),
- don’t allow unsigned legacy Option ROMs,
- configure secure image verification policies,
-

and don’t’ introduce a single bug in all of this, of course.

tianocore

Summary

Platform Firmware Security – Why is it important?

Why is platform firmware Security important	Prevent low level attacks that could “brick” the system
UEFI boot flow with the threat model	Identify where UEFI firmware is vulnerable and define a Threat Model
Security technologies overview	Boot Guard, Secure Boot and NIST Secure Updates provide mitigations to some hacking methods

tianocore

SUMMARY

- Why is platform firmware Security important
- UEFI boot flow with the threat model
- Security technologies overview
- Tools and resources on how to test firmware for security

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

ACKNOWLEDGEMENTS