Secure Processors in Industry

Mengjia Yan

Fall 2020

Based on slides from Christopher W. Fletcher and Jakub Szefer

Reminder

- Fill the google form
    - https://forms.gle/G6gh6sEYJ4UY24ePA

- First review will be due @ 09/27 (2.5 weeks from now)

Recommended Reading

- *Intel SGX Explained; Victor Costan, Srini Devadas*
  - Great refresh on computer architecture
  - Background on cryptographic
  - Basic SGX programing model and architecture support (next lecture)

Outline

- IBM secure coprocessor 3848 and follow-ons
- Trusted Platform Module (TPM)
- Intel TXT, AMD

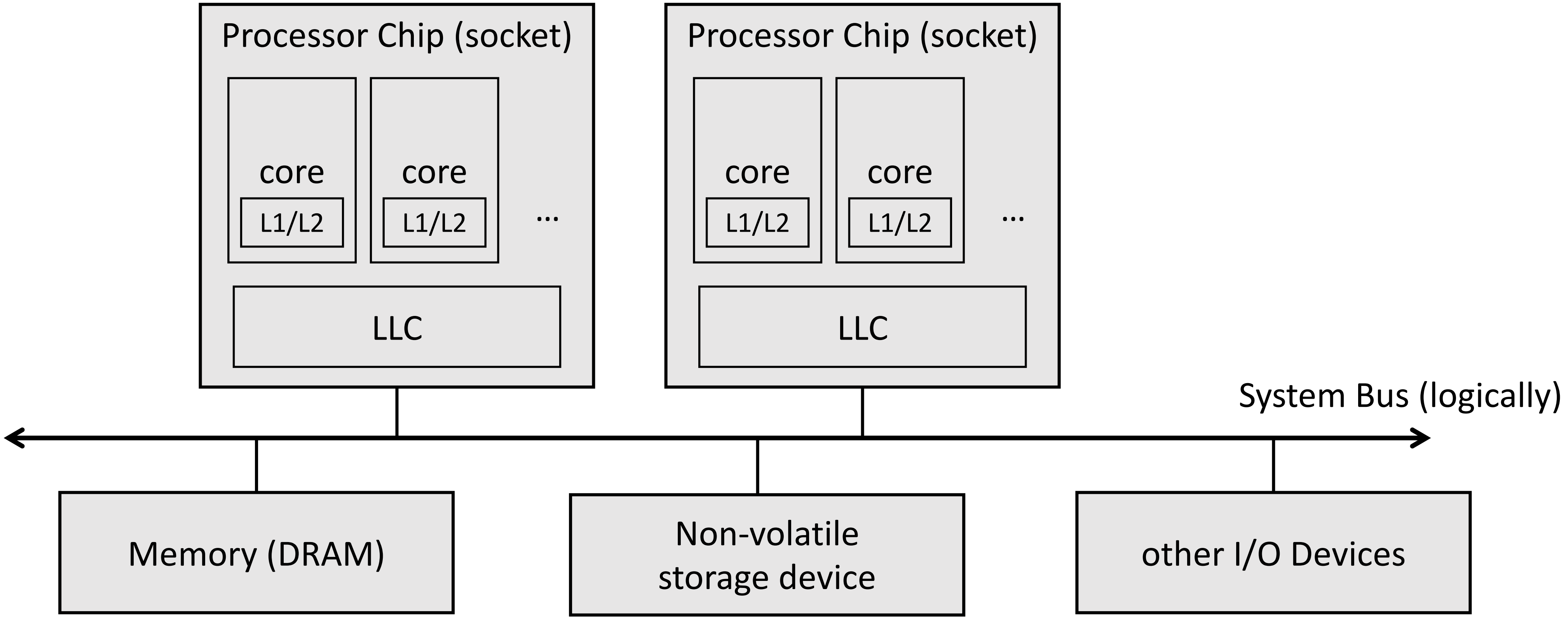
- Arm TrustZone
- Intel SGX
- AMD SEV

Threat model

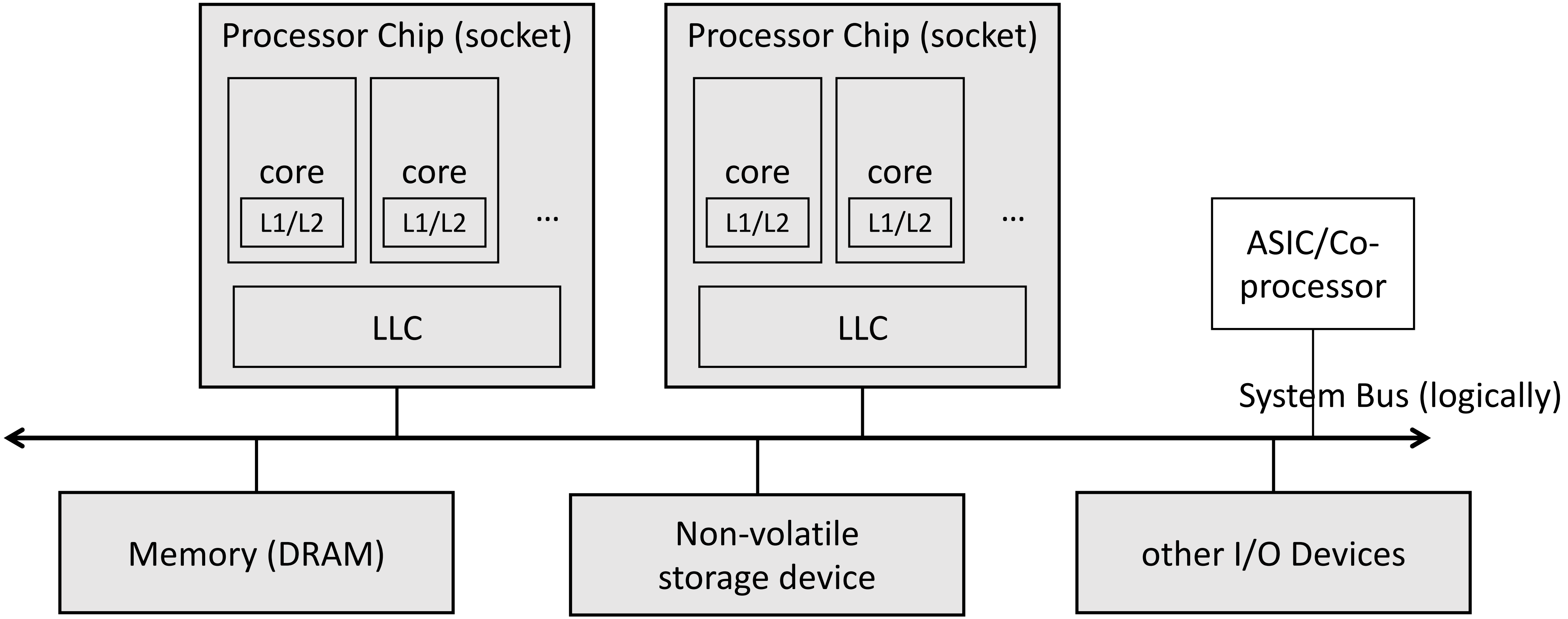
Trusted Computing Base (TCB)

Physical Attacks

Computing Model

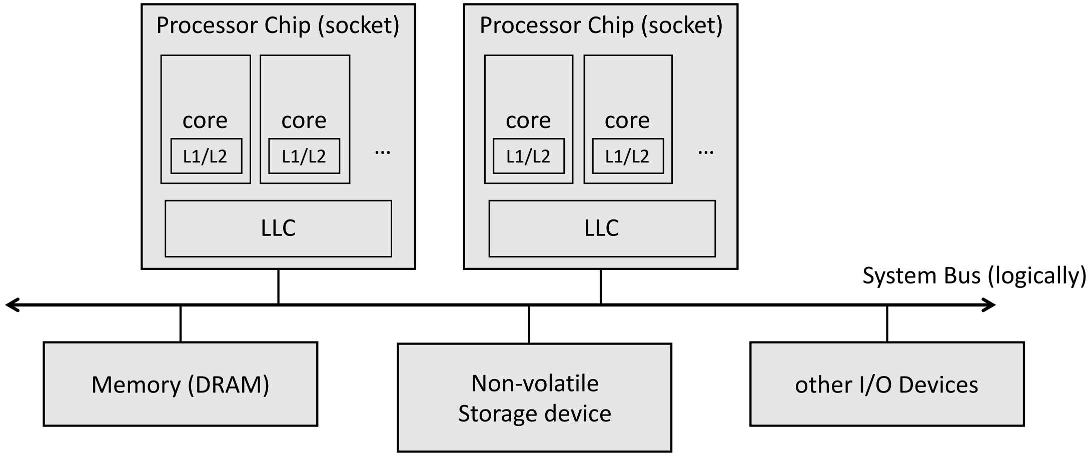


Computing Model



Hardware Adversary

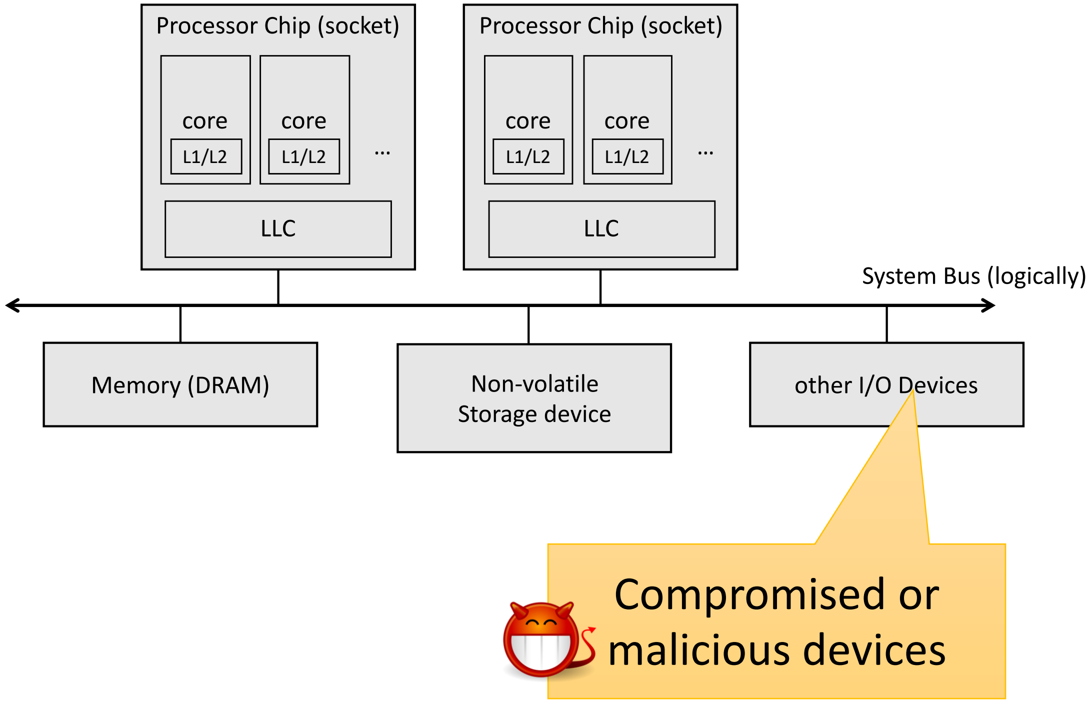
- Pre-fab adversary (HW trojans)
- Physical attacks
  - Generally require physical access
  - Classified according to cost
  - A cold boot attack example



Advanced Hardware Hacking Techniques; Joe Grand; DEFCON'12

Hardware Adversary

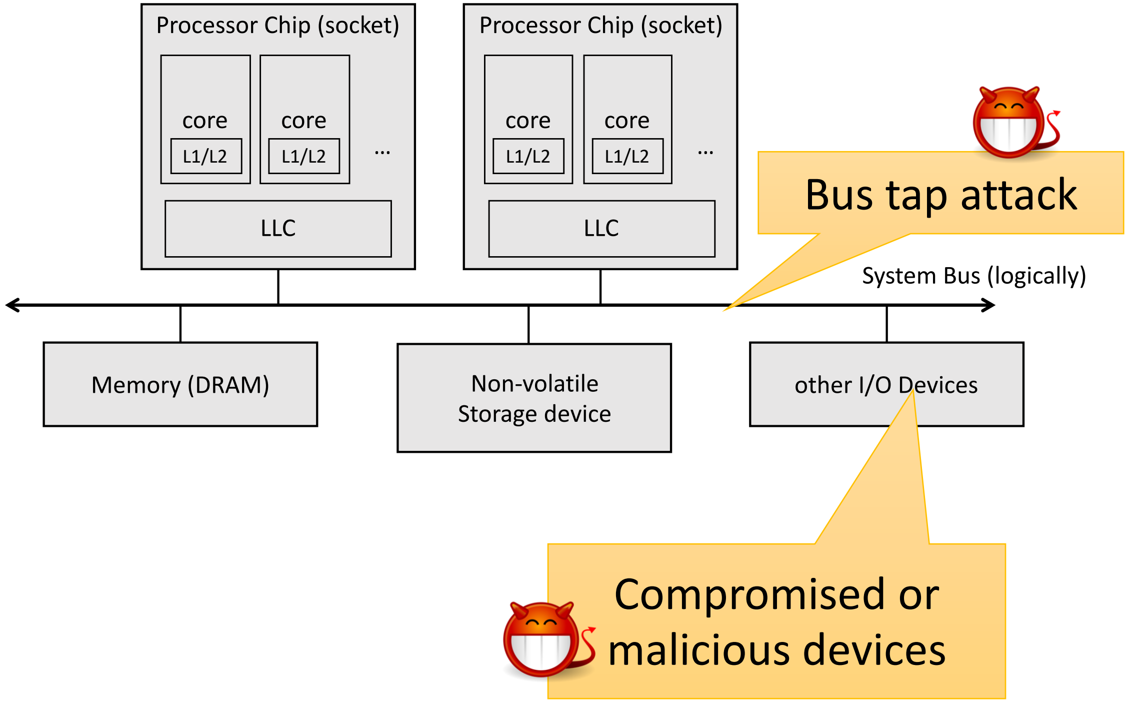
- Pre-fab adversary (HW trojans)
- Physical attacks
  - Generally require physical access
  - Classified according to cost
  - A cold boot attack example



Advanced Hardware Hacking Techniques; Joe Grand; DEFCON’12

Hardware Adversary

- Pre-fab adversary (HW trojans)
- Physical attacks
  - Generally require physical access
  - Classified according to cost
  - A cold boot attack example

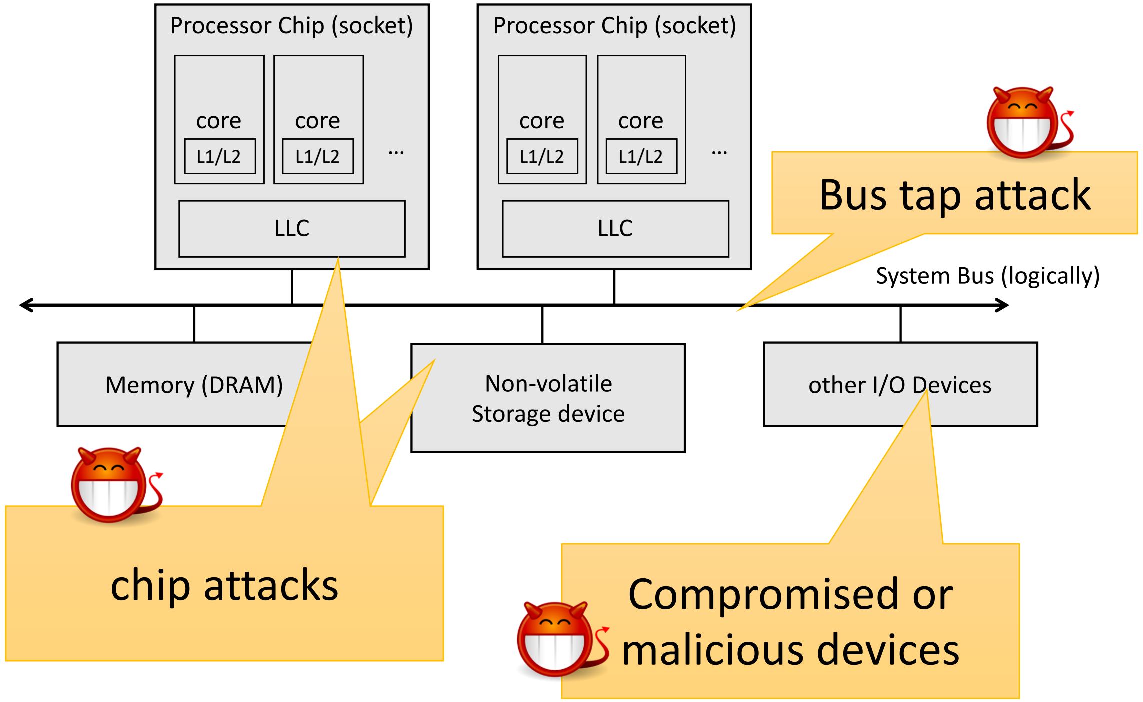


Advanced Hardware Hacking Techniques; Joe Grand; DEFCON'12

Hardware Adversary

- Pre-fab adversary (HW trojans)

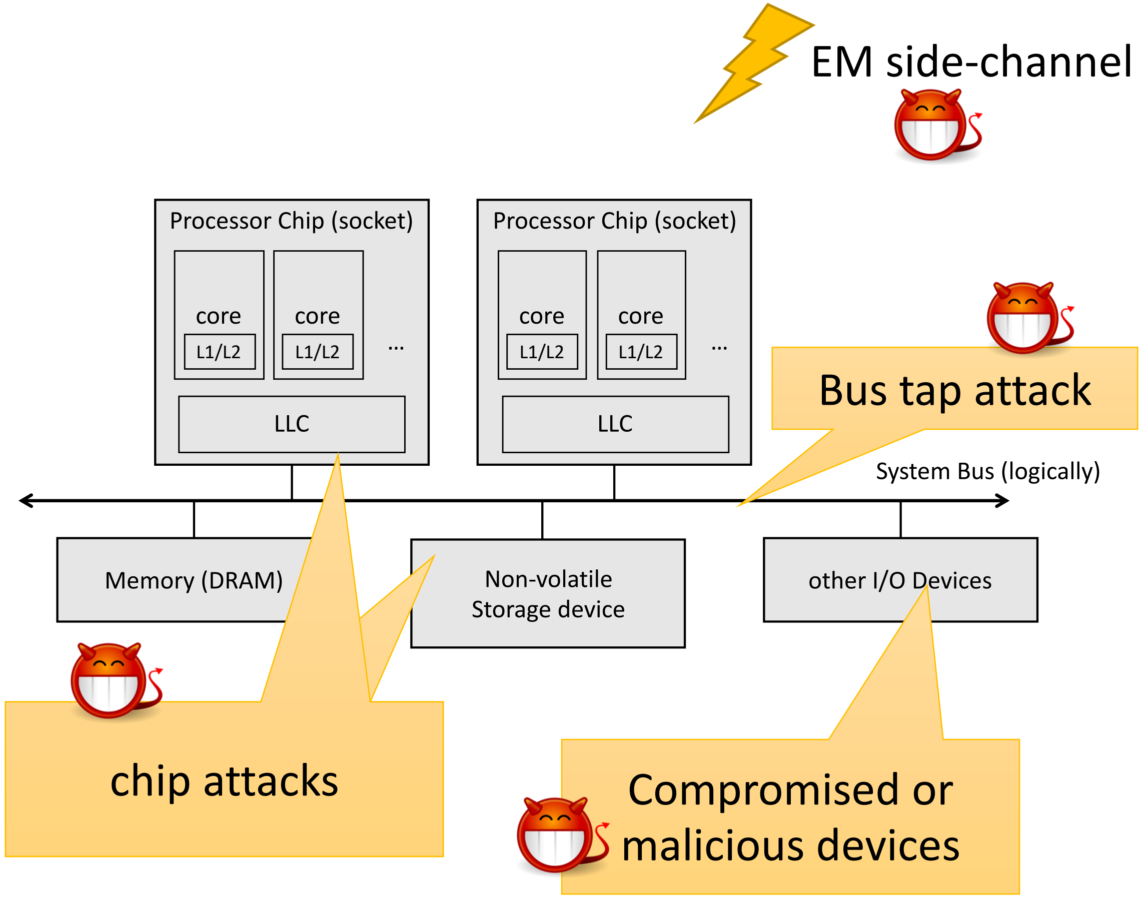
- Physical attacks
  - Generally require physical access
  - Classified according to cost
  - A cold boot attack example



Advanced Hardware Hacking Techniques; Joe Grand; DEFCON'12

Hardware Adversary

- Pre-fab adversary (HW trojans)
- Physical attacks
  - Generally require physical access
  - Classified according to cost
  - A cold boot attack example

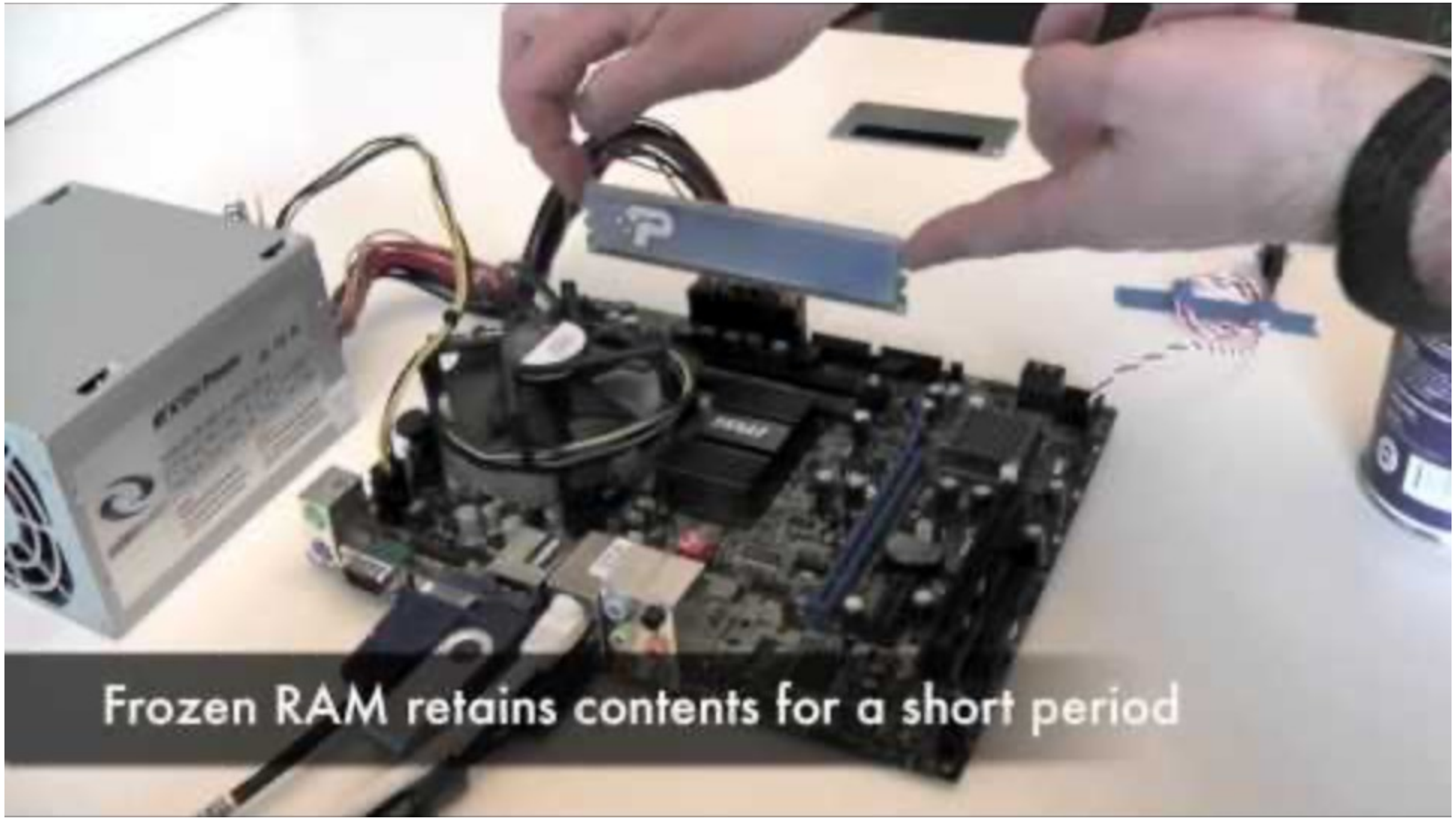


Advanced Hardware Hacking Techniques; Joe Grand; DEFCON'12

A Cold Boot Attack Example

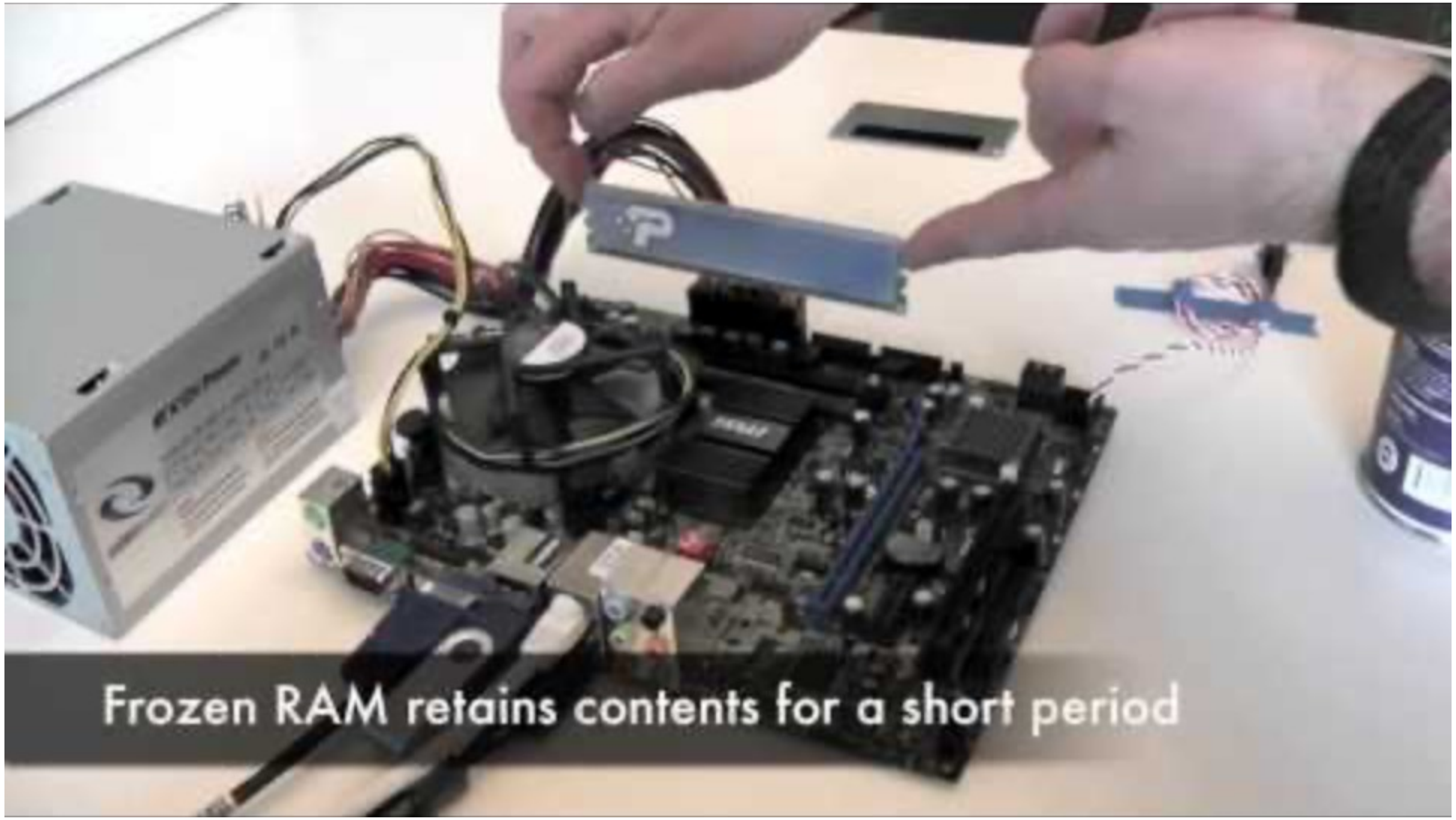
https://www.youtube.com/watch?v=vWHDqBV9yGc

A Cold Boot Attack Example



https://www.youtube.com/watch?v=vWHDqBV9yGc

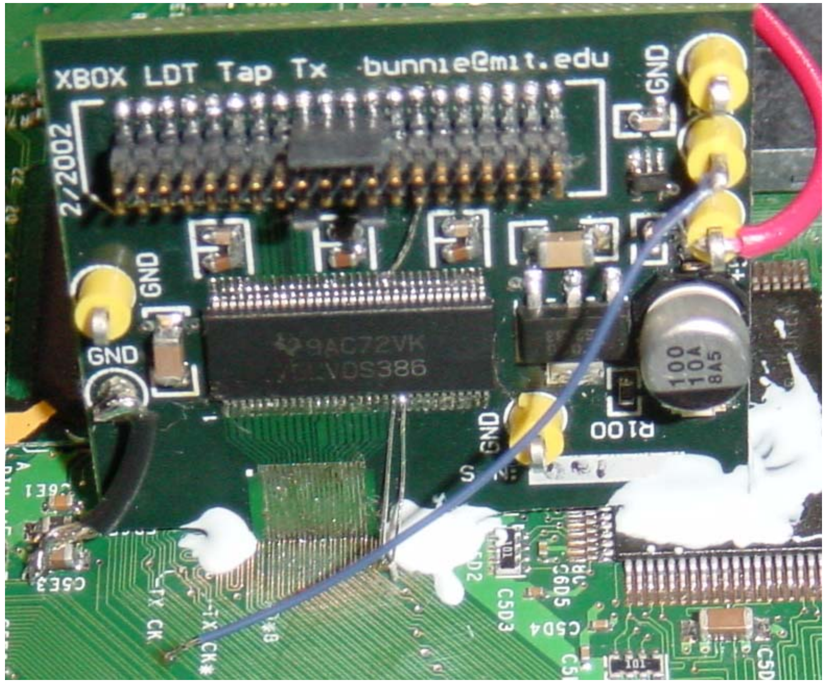
A Cold Boot Attack Example



https://www.youtube.com/watch?v=vWHDqBV9yGc

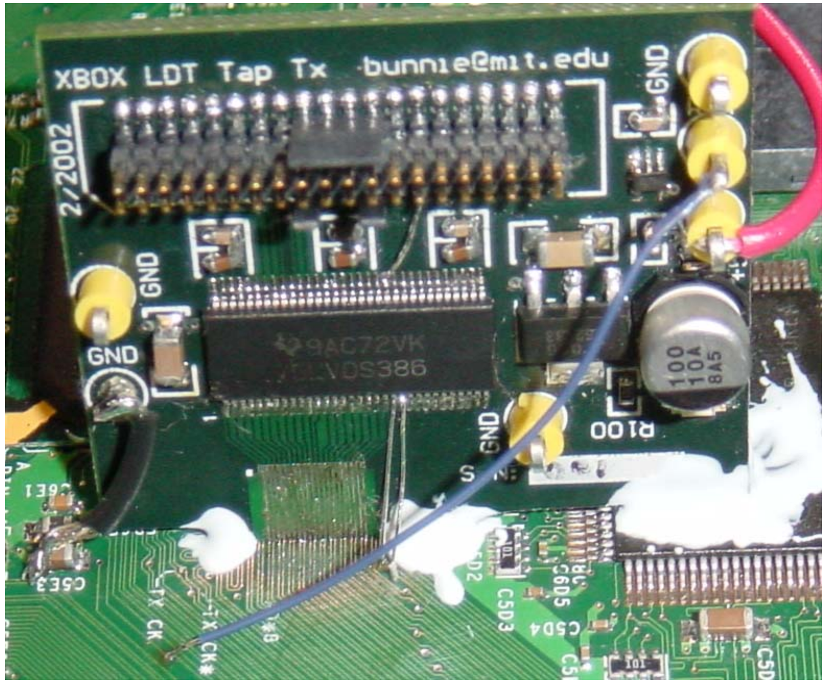
Gutmann et al. "Data Remanence in Semiconductor Devices"

More Physical Attack Examples

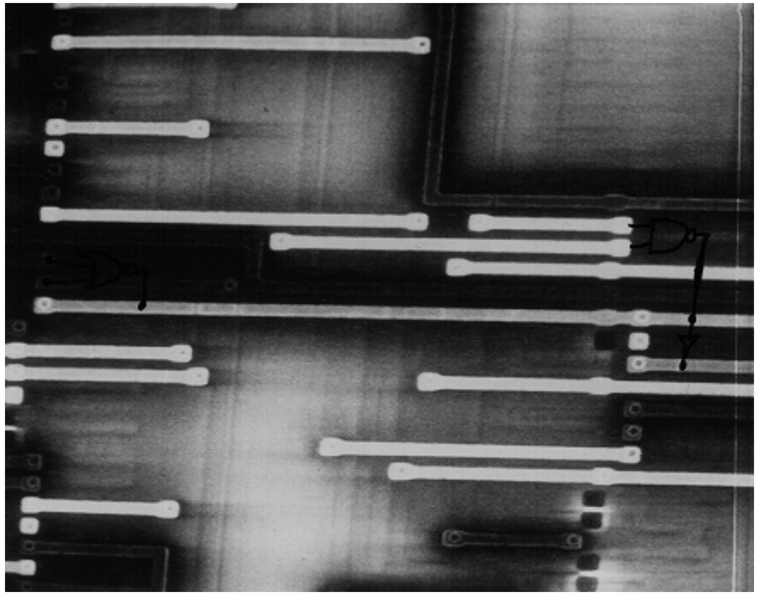


Tap board used to intercept data transfer over Xbox's HyperTransport bus
from *http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html*

More Physical Attack Examples



Tap board used to intercept data transfer over Xbox's HyperTransport bus
from *http://www.xenatera.com/bunnie/proj/anatak/xboxmod.html*



IC analysis. Extract information from a Flash ROM storage cell
from http://testequipmentcanada.com/VoltageContrastPaper.html

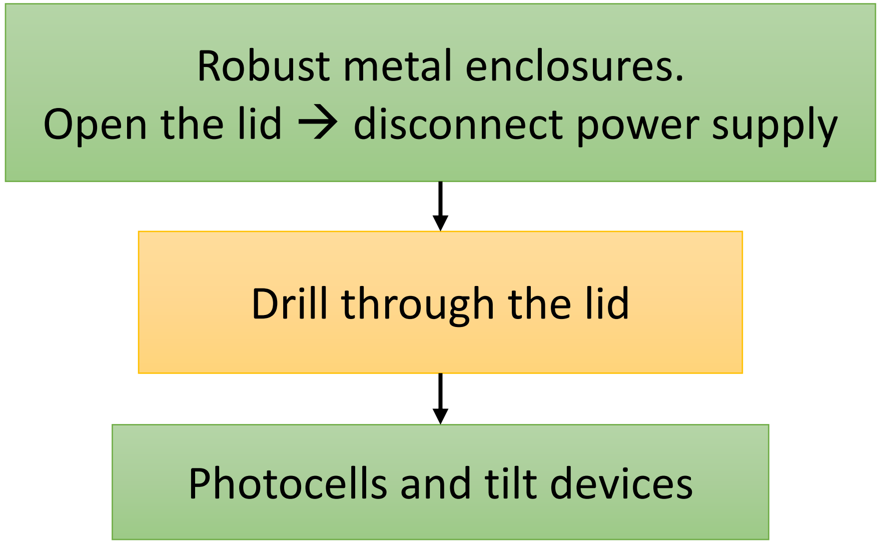
Physical Tamper Resistance

- Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
- A history lesson of physical security by IBM 4758

Physical Tamper Resistance

- Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
- A history lesson of physical security by IBM 4758

Tampering Detection



Robust metal enclosures.
Open the lid → disconnect power supply

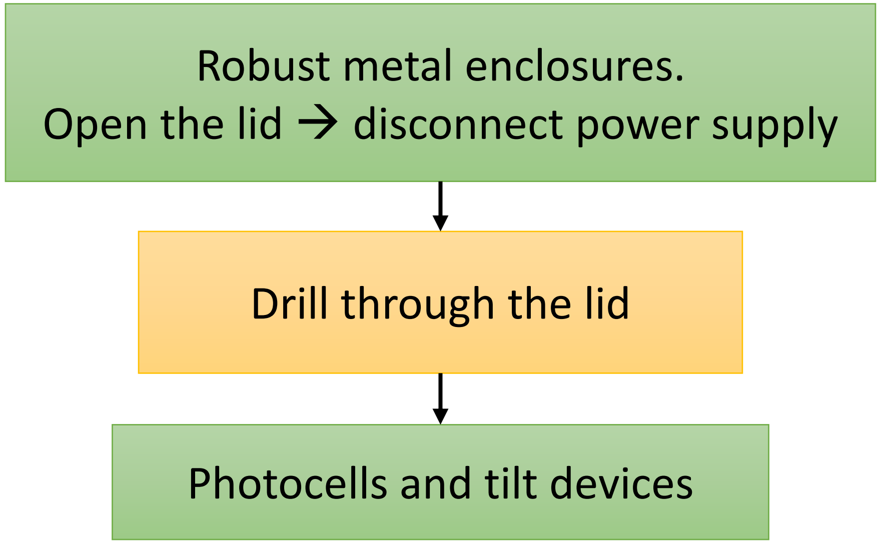
Drill through the lid

Photocells and tilt devices

Physical Tamper Resistance

- Standalone security modules to protect cryptographic keys and personal identification numbers (PINs)
- A history lesson of physical security by IBM 4758

Tampering Detection

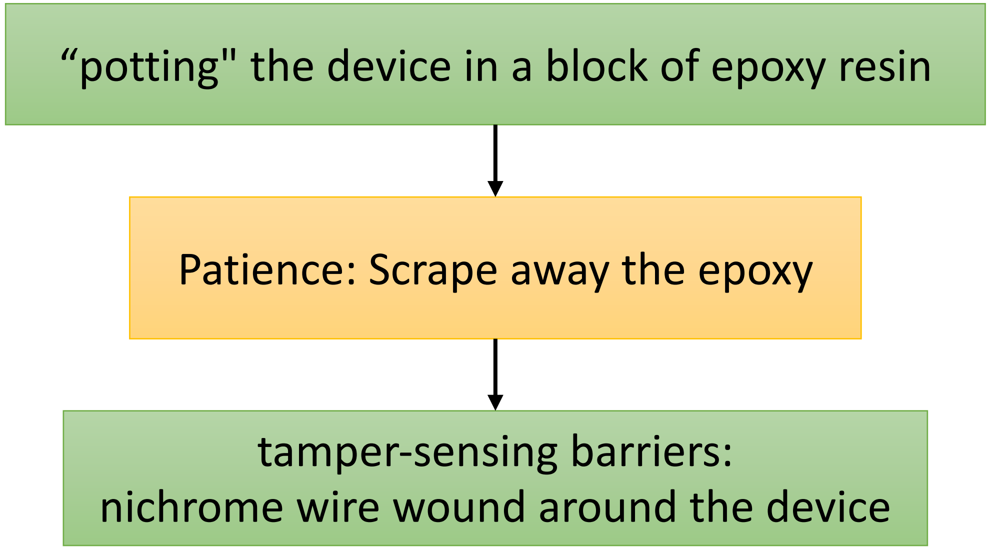


Robust metal enclosures.
Open the lid → disconnect power supply

Drill through the lid

Photocells and tilt devices

Tampering Evident



“potting" the device in a block of epoxy resin

Patience: Scrape away the epoxy

tamper-sensing barriers:
nichrome wire wound around the device

IBM 4758 Secure Co-Processor

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from *https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html*

IBM 4758 Secure Co-Processor

- Memory remanence
  - constant movement of values from place to place
- Cold boot
  - detects changes of temperature
- X-ray
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from *https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html*

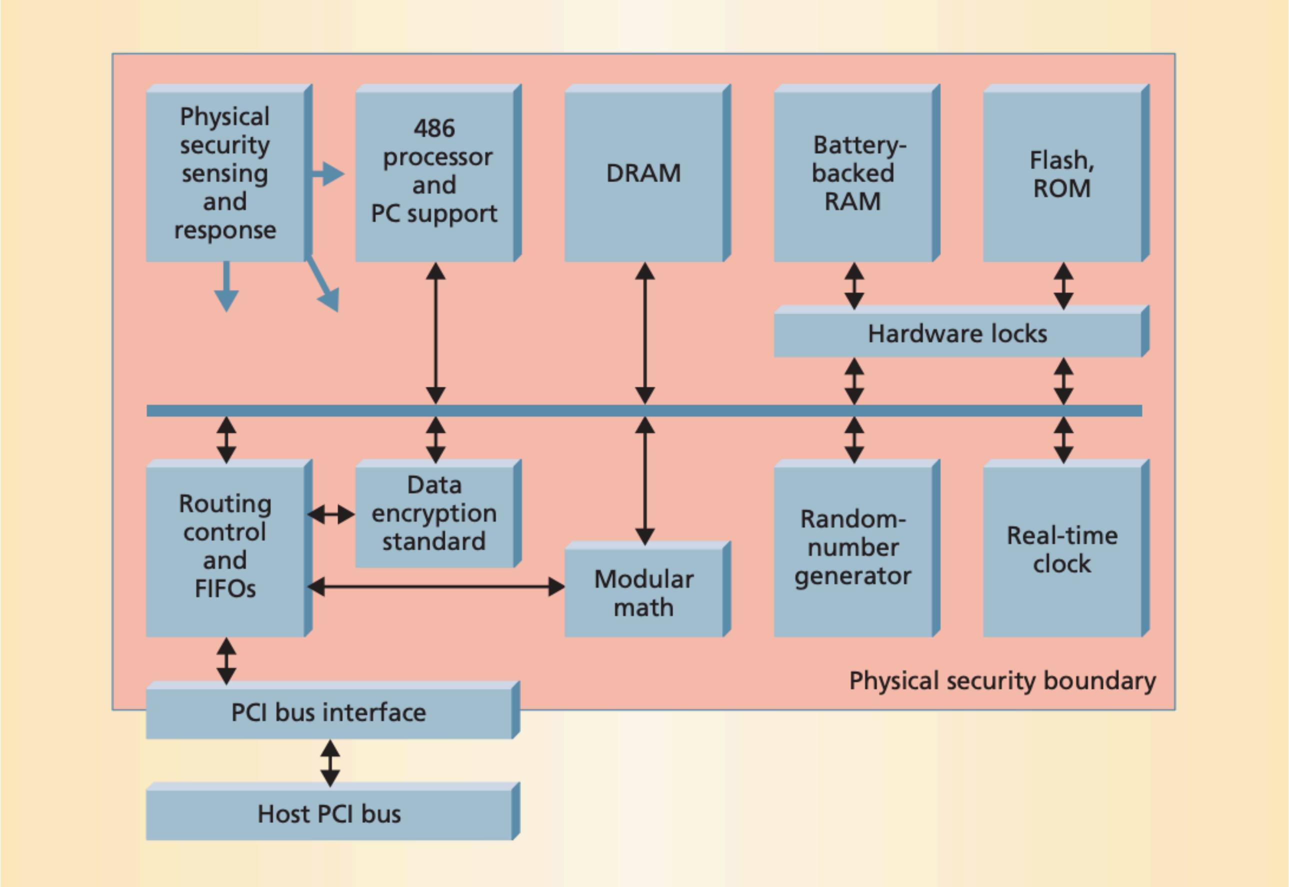
IBM 4758 Secure Co-Processor

- Memory remanence
  - constant movement of values from place to place
- Cold boot
  - detects changes of temperature
- X-ray
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from *https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html*

Expensive. Other secure processors only focus on a limited set of physical attacks.

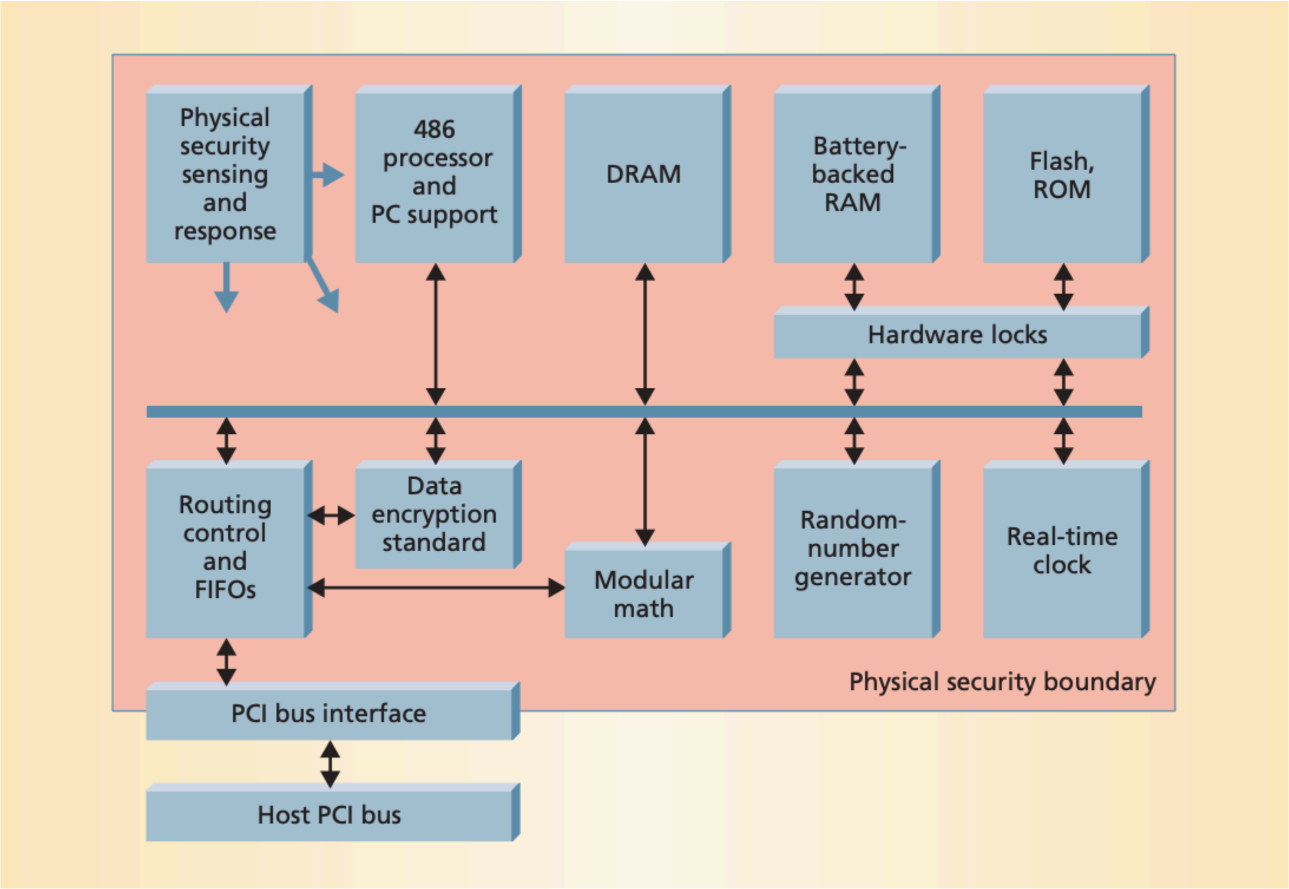
IBM 4758 and Follow-ons



From Dyer et al. "Building the IBM 4758 Secure Coprocessor"

- The first FIPS 140-1 Level 4 validation, arguably the only general-purpose computational platform validated at this level by 2001

IBM 4758 and Follow-ons



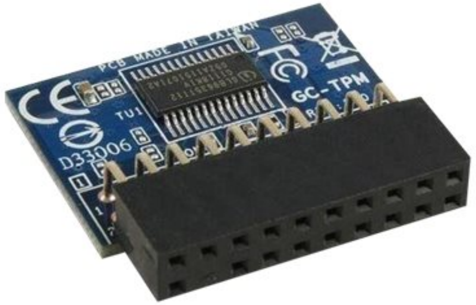
From Dyer et al. "Building the IBM 4758 Secure Coprocessor"

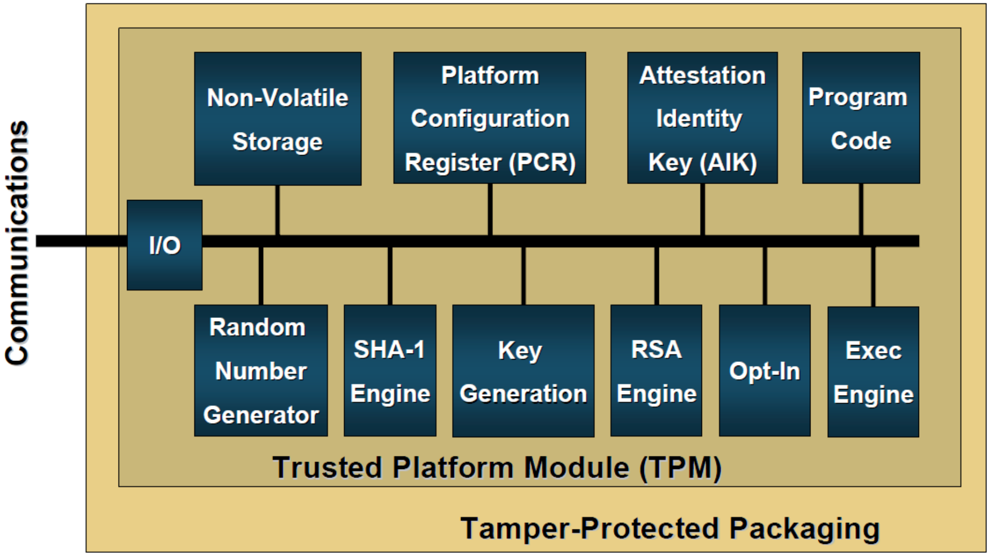
- The first FIPS 140-1 Level 4 validation, arguably the only general-purpose computational platform validated at this level by 2001
- A multipurpose programmable device
- Secure Boot and SW attacks (discussed later)

Bond et al. "API-Level Attacks on Embedded Systems."

Trusted Platform Module (TPM)

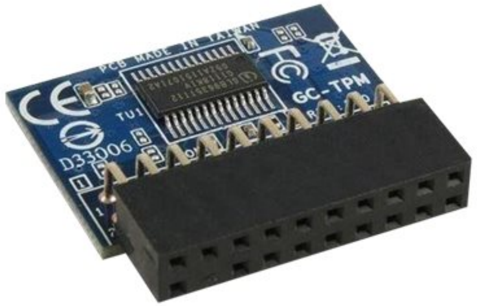
- “Commoditized IBM 4758”
- Standard LPC interface – attaches to commodity motherboards
- Weaker computation capability



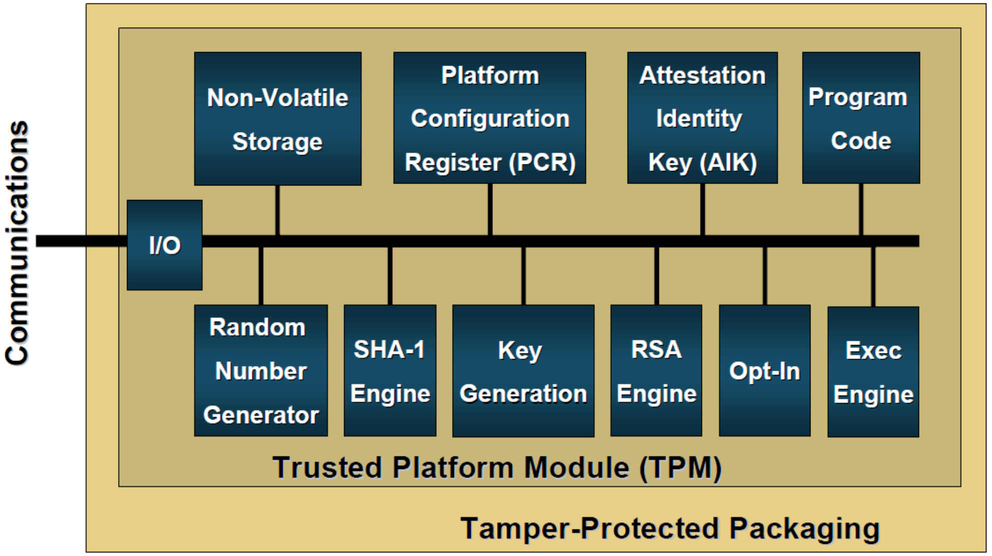


Trusted Platform Module (TPM)

- “Commoditized IBM 4758”
- Standard LPC interface – attaches to commodity motherboards
- Weaker computation capability

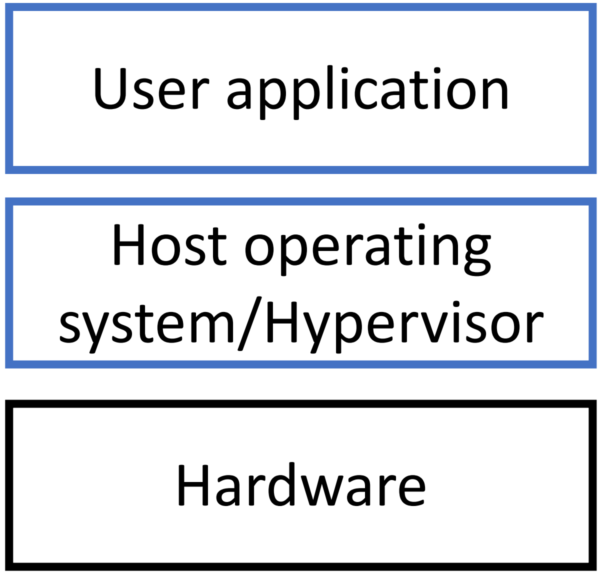


- Uses:
  - Verify platform integrity (firmware+OS)
  - Disk encryption and password protection



Software Attacks

Software Stack

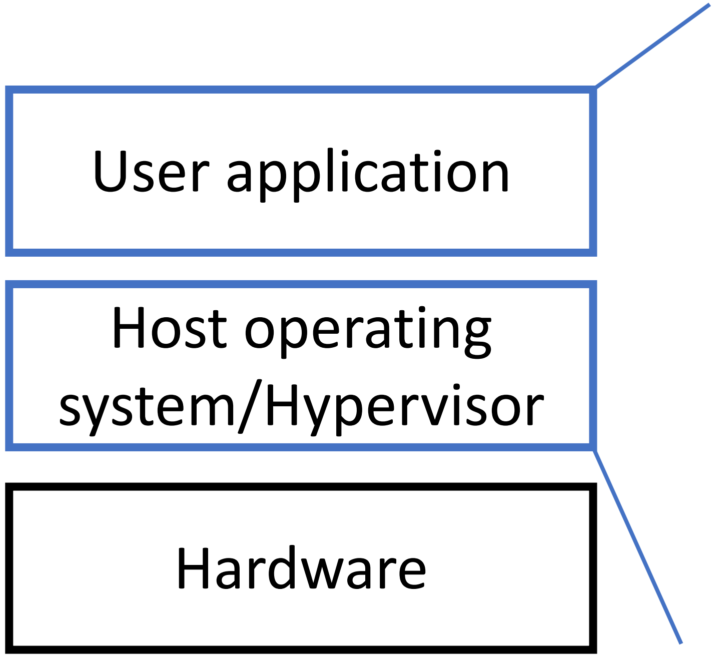


User application

Host operating system/Hypervisor

Hardware

Software Stack



Intel’s Privilege Level

Less Privilege

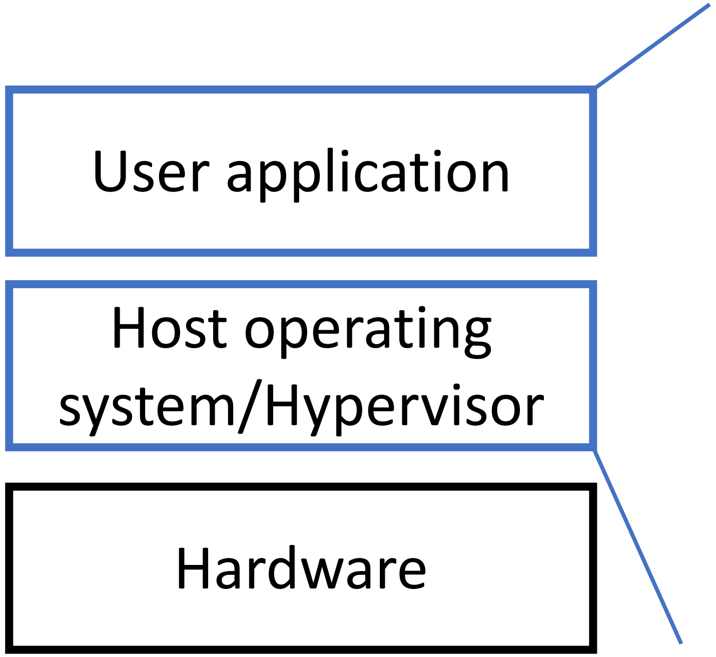
Ring 3	Application Enclave application
Ring 2	
Ring 1	
Ring 0	OS kernel

SMM	BIOS/firmware

More Privilege

SMM: system management mode

Software Stack

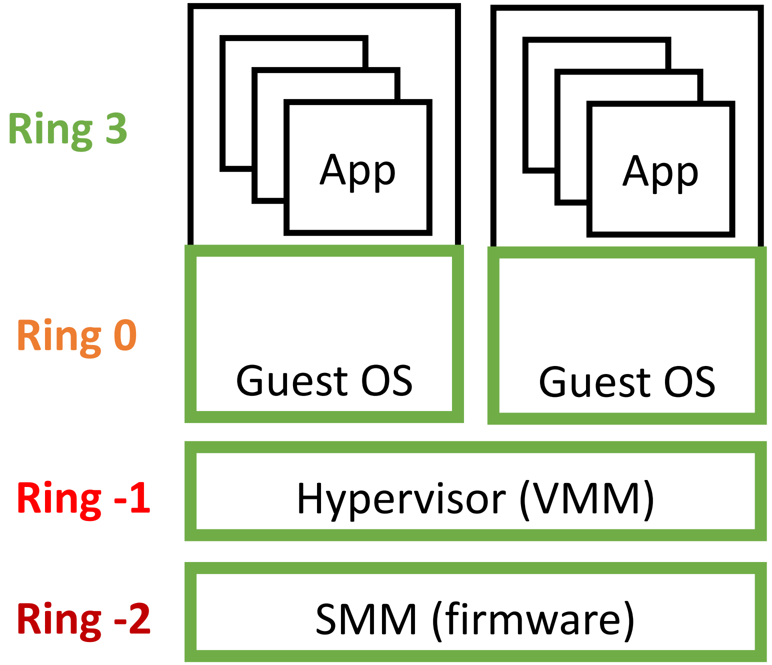


Intel's Privilege Level

Less Privilege

Ring 3	Application Enclave application
Ring 2	
Ring 1	
Ring 0	OS kernel
SMM	BIOS/firmware

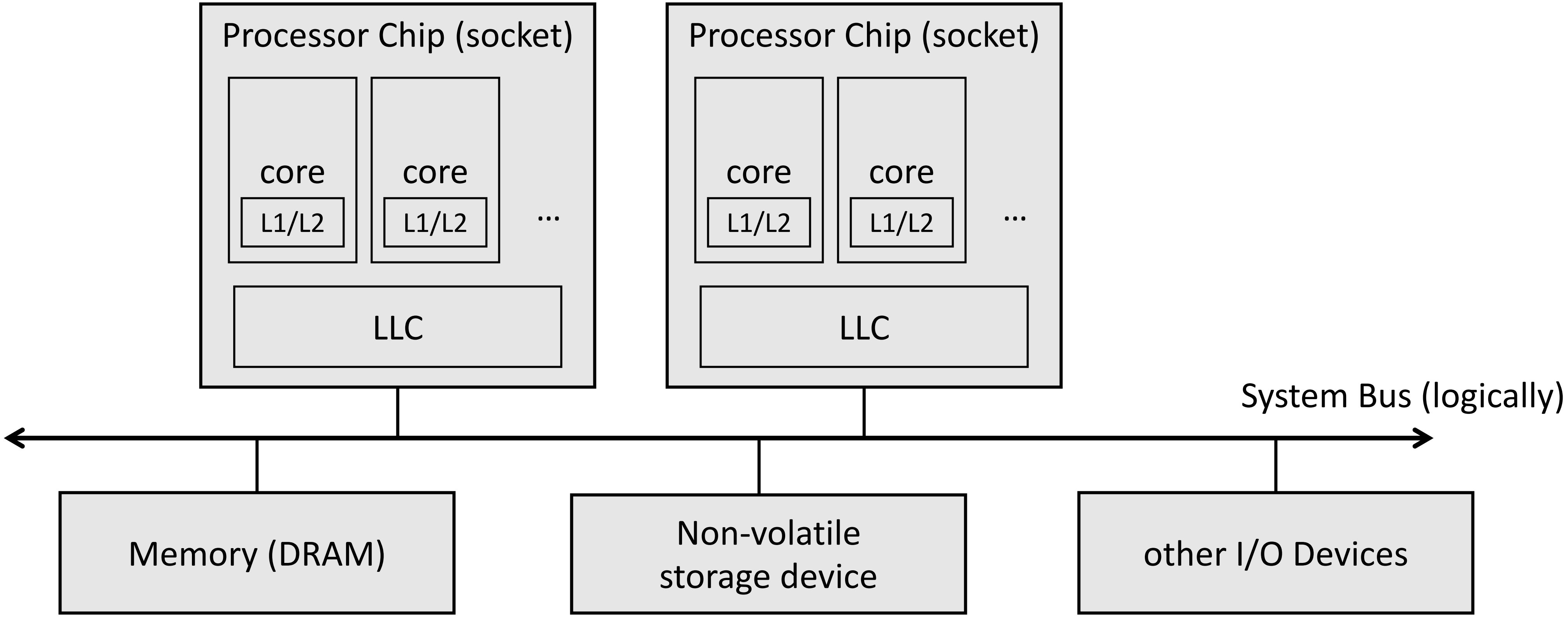
More Privilege



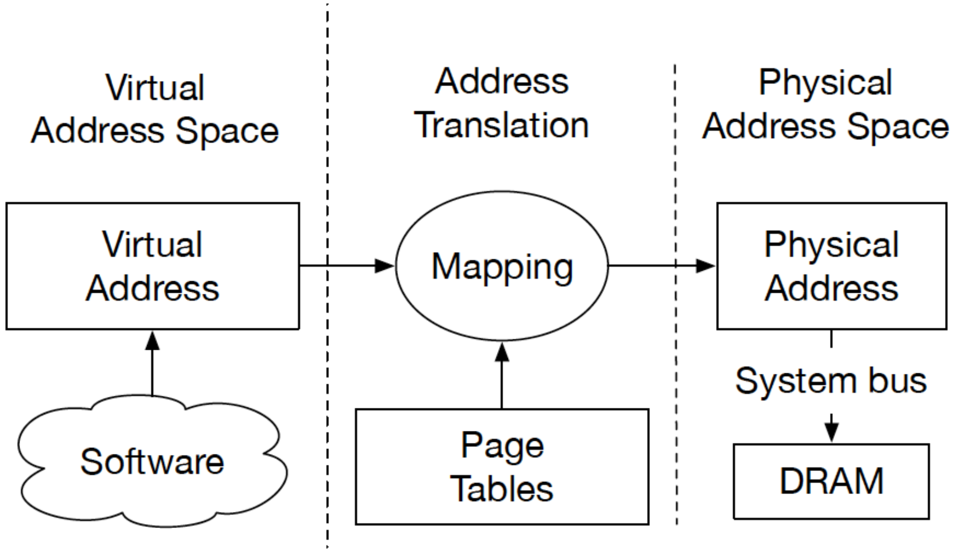
SMM: system management mode

Process Isolation When Sharing Hardware

- Share HW resources in SMT contexts, same processor chips, across sockets.



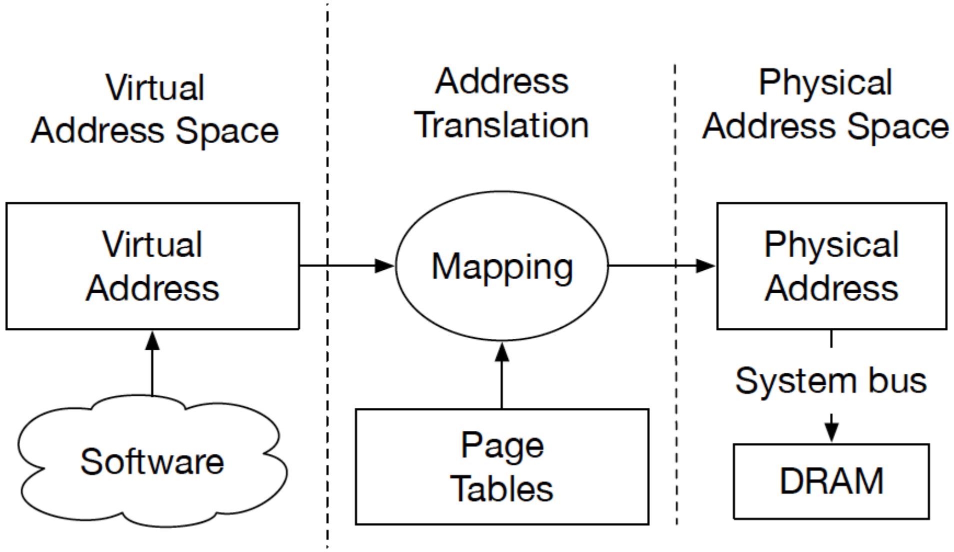
Virtual Address Abstraction



Virtual Address Abstraction

Benefits of virtual memory abstraction:

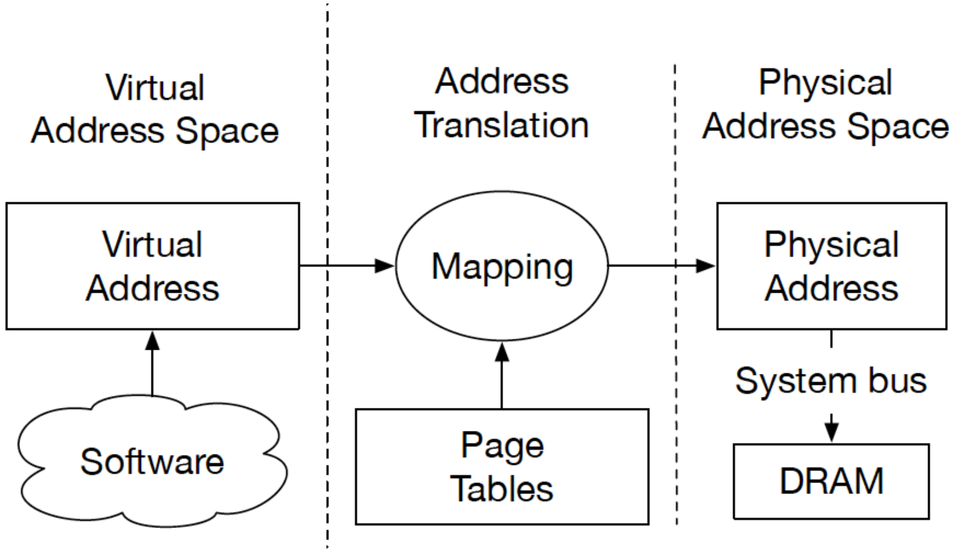
- Over-commit memory: the illusion that they own all resources
- Security: process isolation
- Programmability: software independent of DRAM size

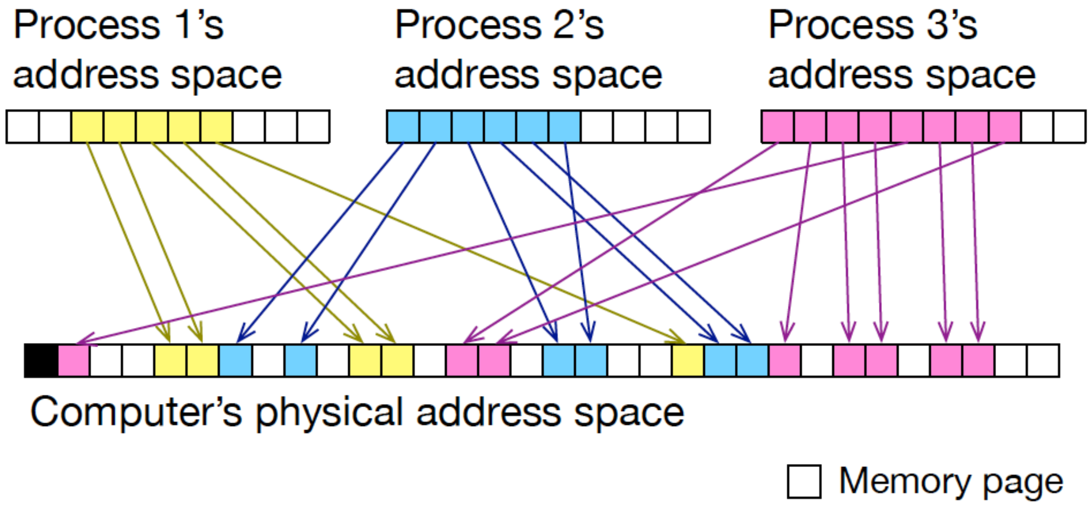


Virtual Address Abstraction

Benefits of virtual memory abstraction:

- Over-commit memory: the illusion that they own all resources
- Security: process isolation
- Programmability: software independent of DRAM size





Process 1's address space
Process 2's address space
Process 3's address space
Computer's physical address space
□ Memory page

Page Table

- Page table:
  - A data structure to store address translation entries
  - Multi-level trees

Page Table

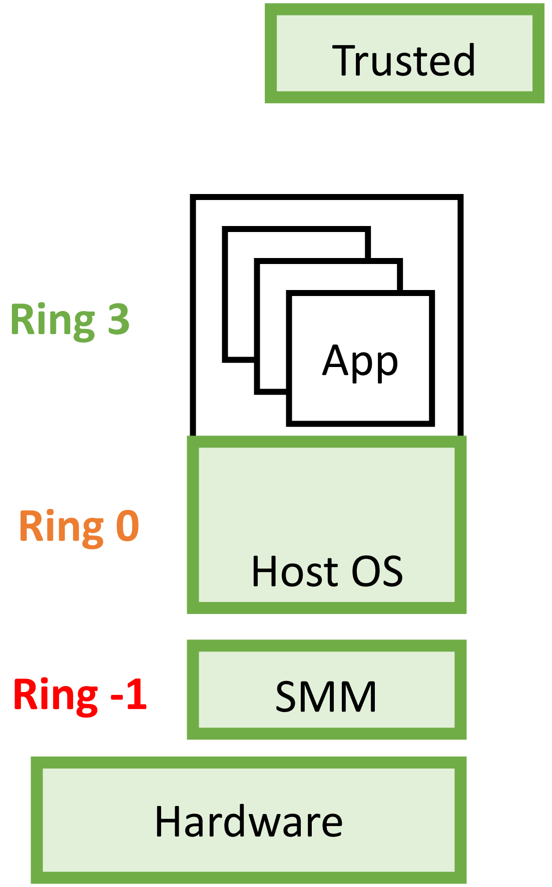
- Page table:
  - A data structure to store address translation entries
  - Multi-level trees
- Page table entry attributes:
  - Writable (W), Executable (X), Supervisor (S), etc.
  - E.g., data execution prevention (DEP)

Page Table

- Page table:
    - A data structure to store address translation entries
    - Multi-level trees
- Page table entry attributes:
    - Writable (W), Executable (X), Supervisor (S), etc.
    - E.g., data execution prevention (DEP)
- MMU (memory management unit)
    - A hardware unit performs address translation
- TLB:
    - Caches for page tables

Trusted computing base (TCB)

- Trusted computing base (TCB)
  - TCB is trusted to be correctly implemented
  - Vulnerabilities or attacks on TCB nullify TEE protections
  - TCB may not be trustworthy

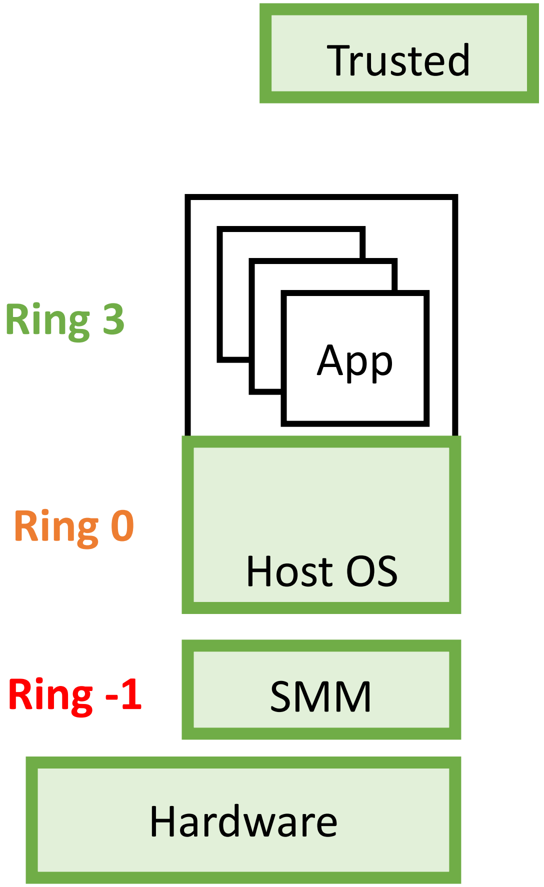


Trusted computing base (TCB)

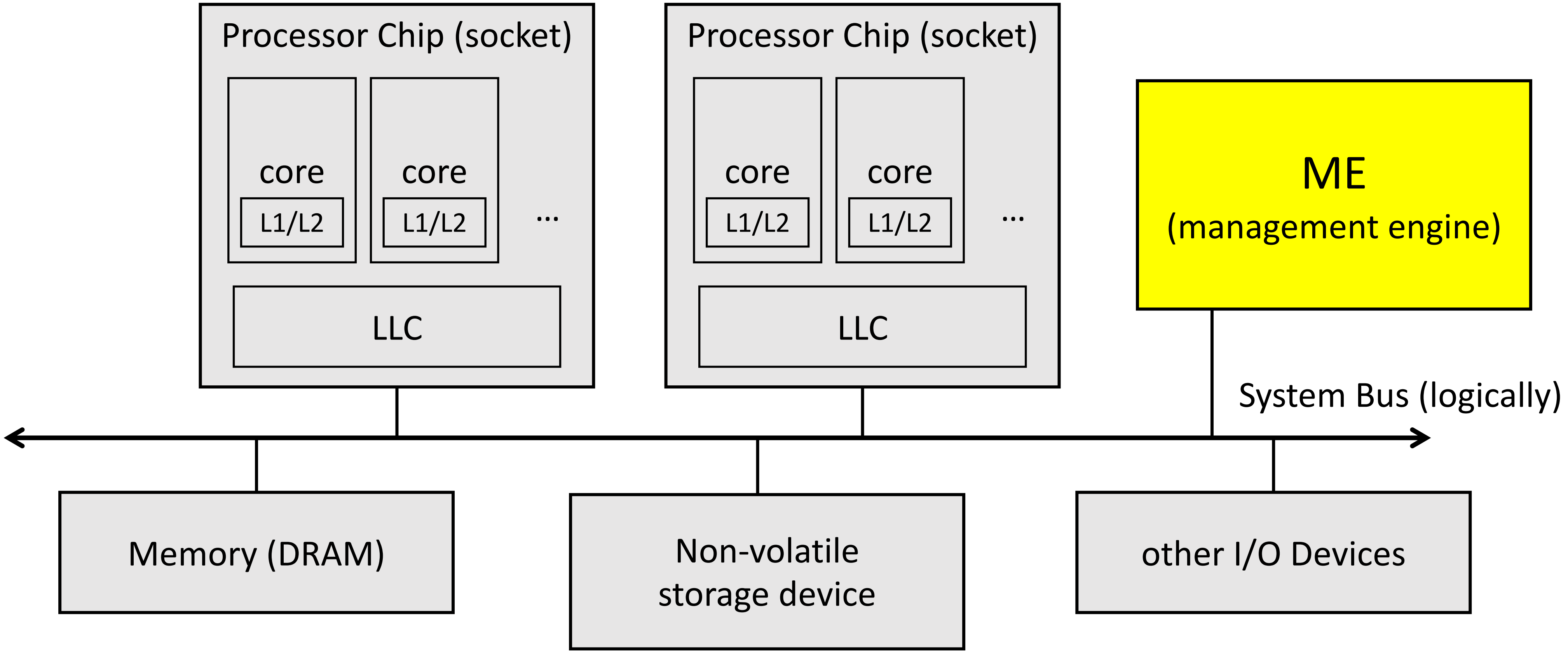
- Trusted computing base (TCB)
  - TCB is trusted to be correctly implemented
  - Vulnerabilities or attacks on TCB nullify TEE protections
  - TCB may not be trustworthy

- Attacks, e.g., Rootkit, may change the **integrity** of TCB

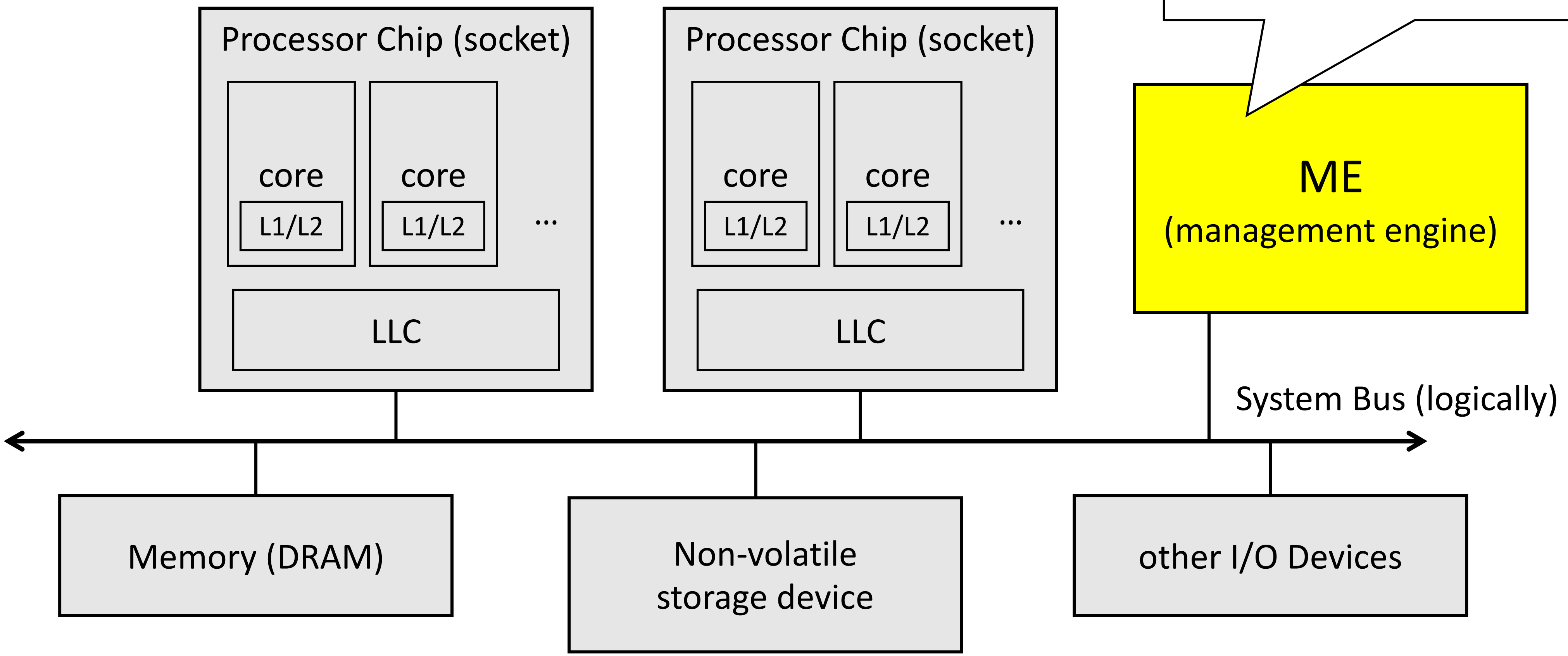
- How to verify platform (HW + low-level SW) integrity



Platform Initialization (Booting)

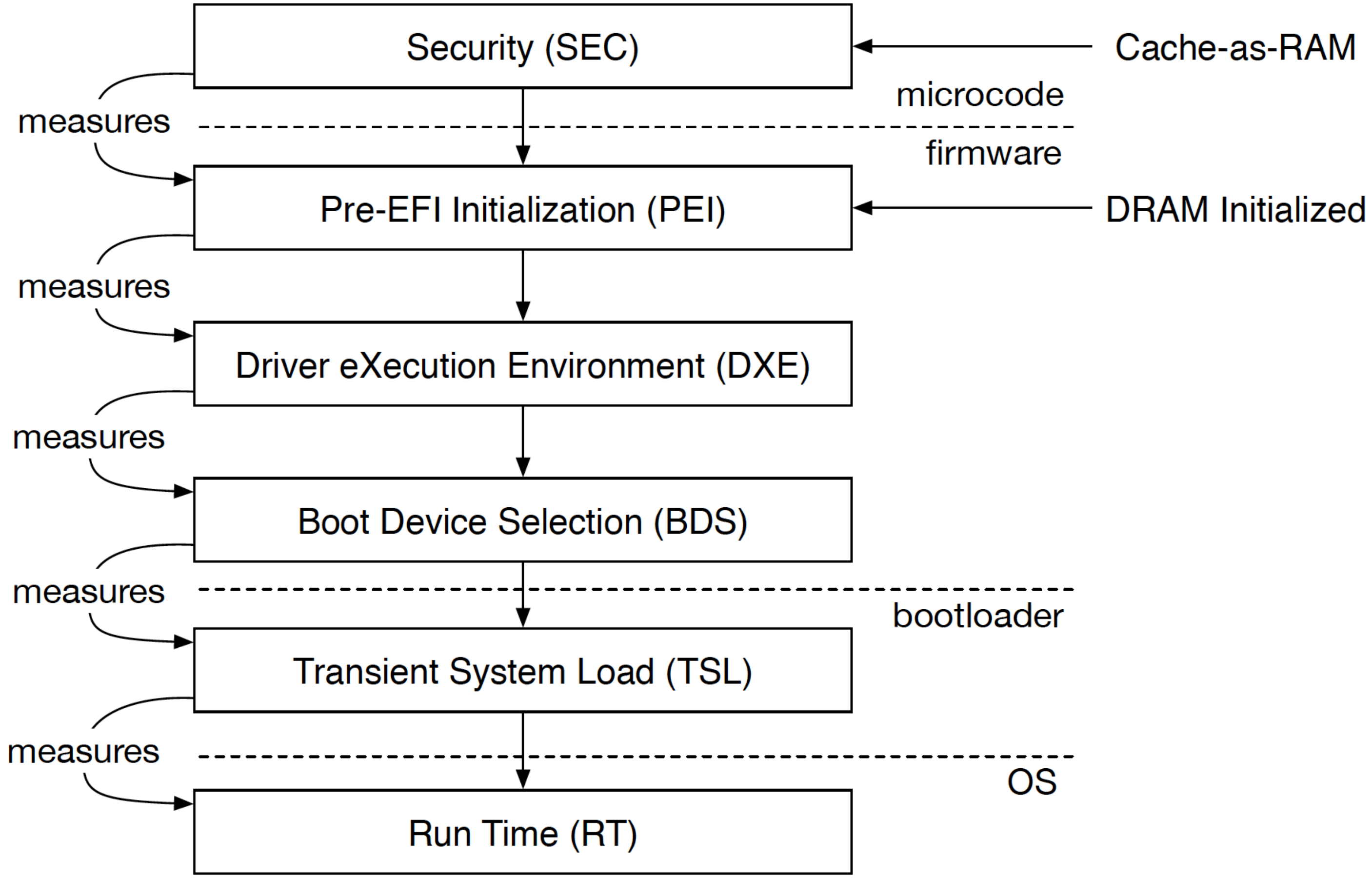


Platform Initialization (Booting)

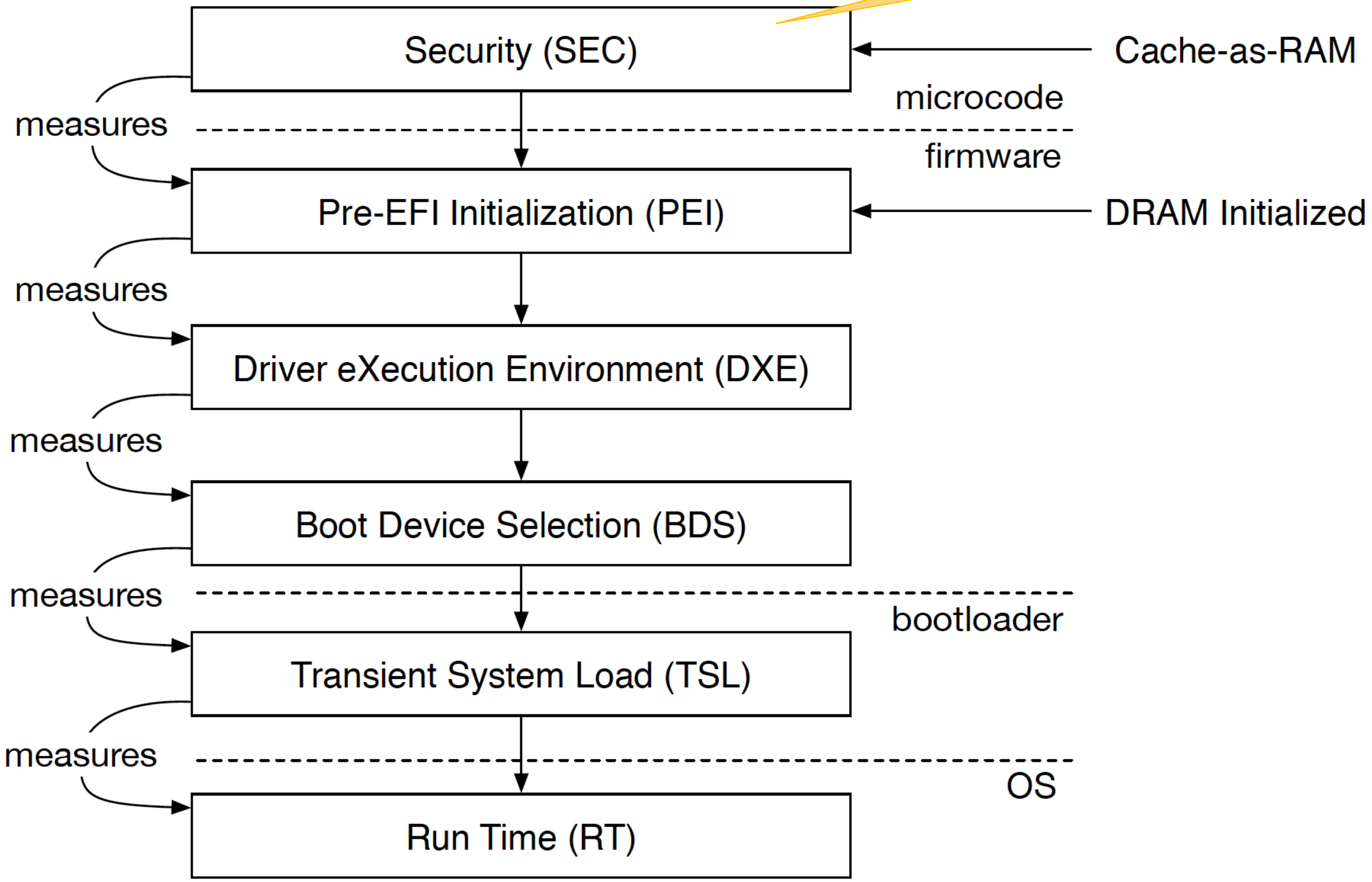


For remote system management.
Also manage booting.

Boot Process (UEFI)

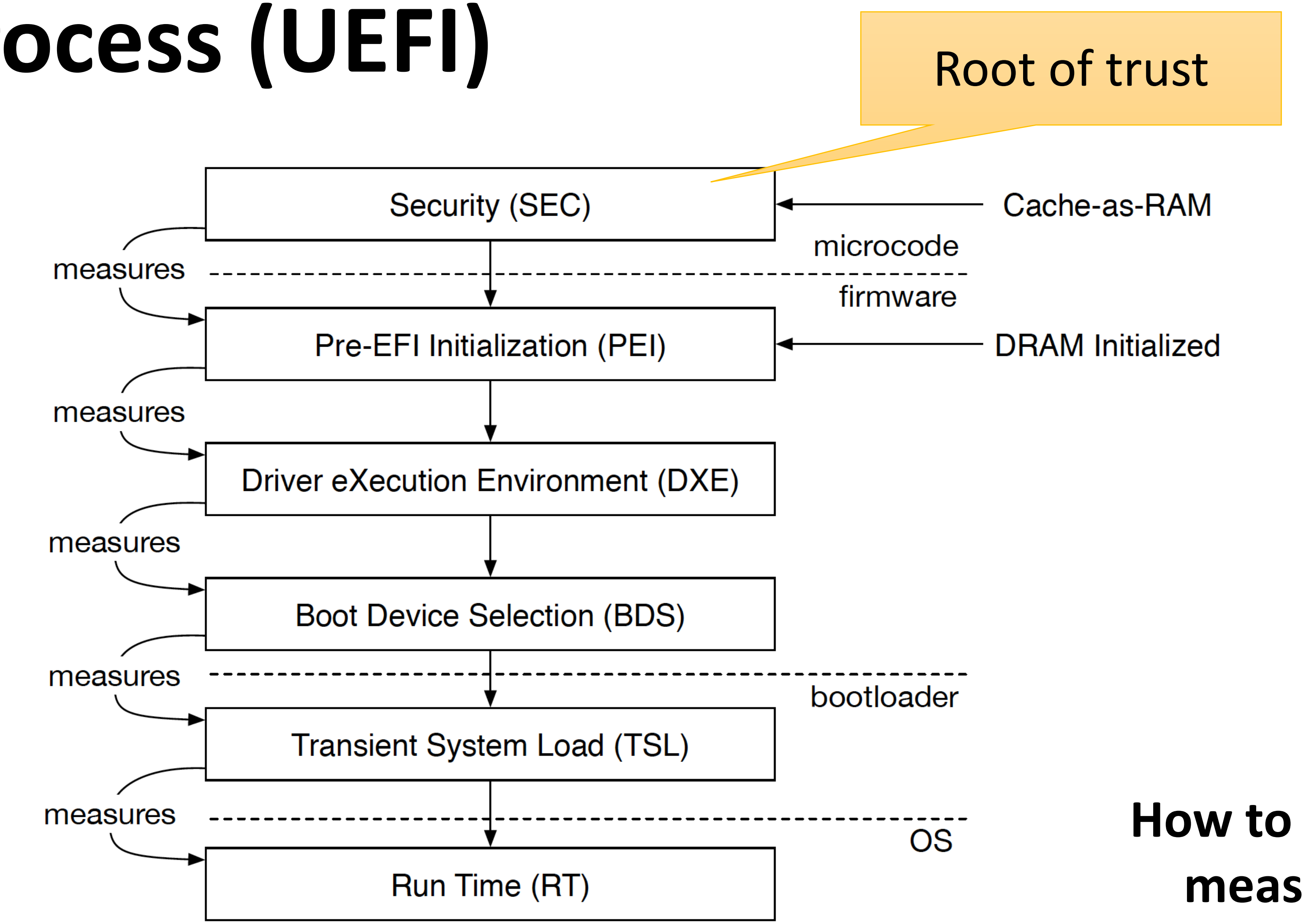


Boot Process (UEFI)



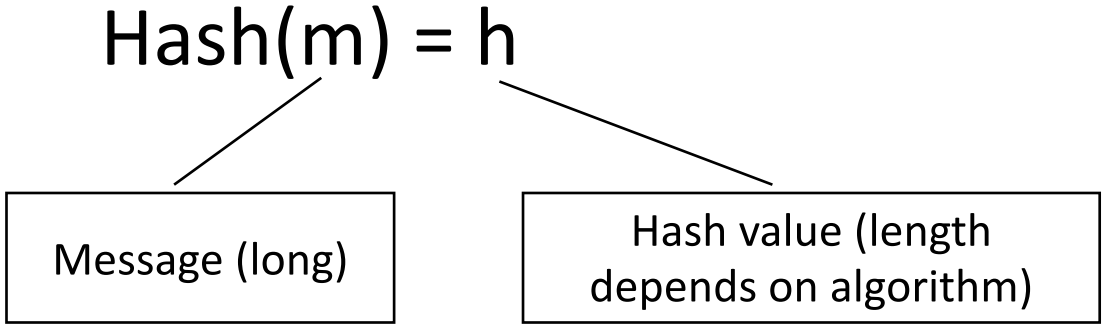
Root of trust

Boot Process (UEFI)



How to perform the measurement?

Cryptographic Hashing (e.g., SHA 1-3)



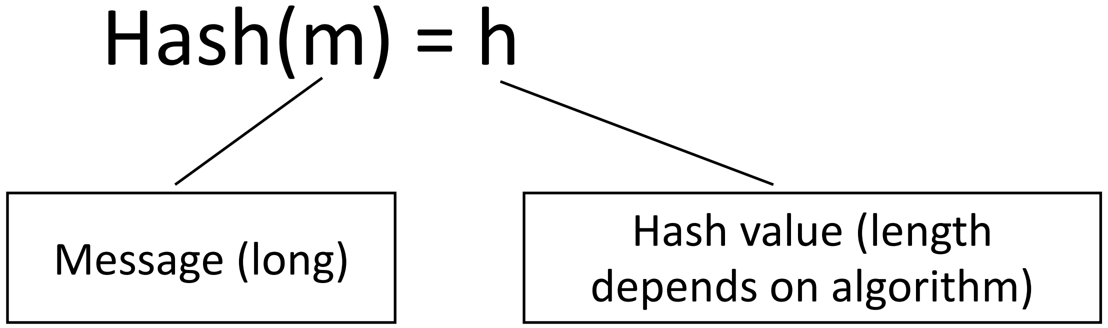
Hash(m) = h

Message (long)

Hash value (length depends on algorithm)

Use as fingerprints

Cryptographic Hashing (e.g., SHA 1-3)



Hash(m) = h

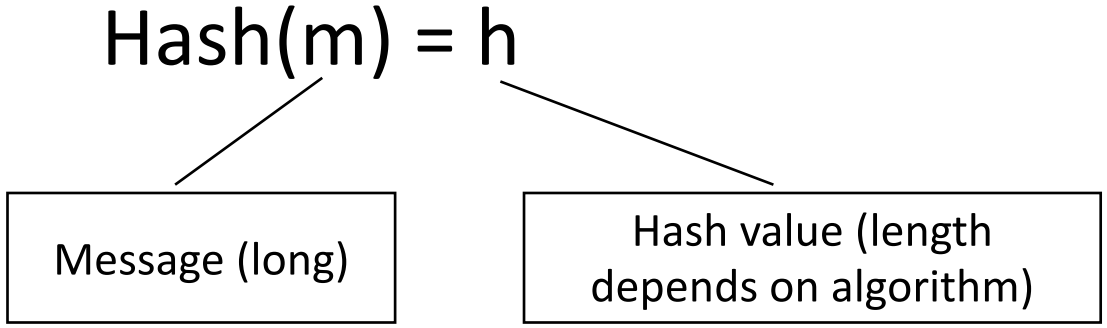
Message (long)

Hash value (length depends on algorithm)

Use as fingerprints

- One-way hash
  - Practically infeasible to invert, Difficult to find collision

Cryptographic Hashing (e.g., SHA 1-3)



Hash(m) = h

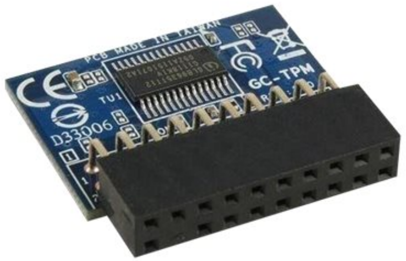
Message (long)

Hash value (length depends on algorithm)

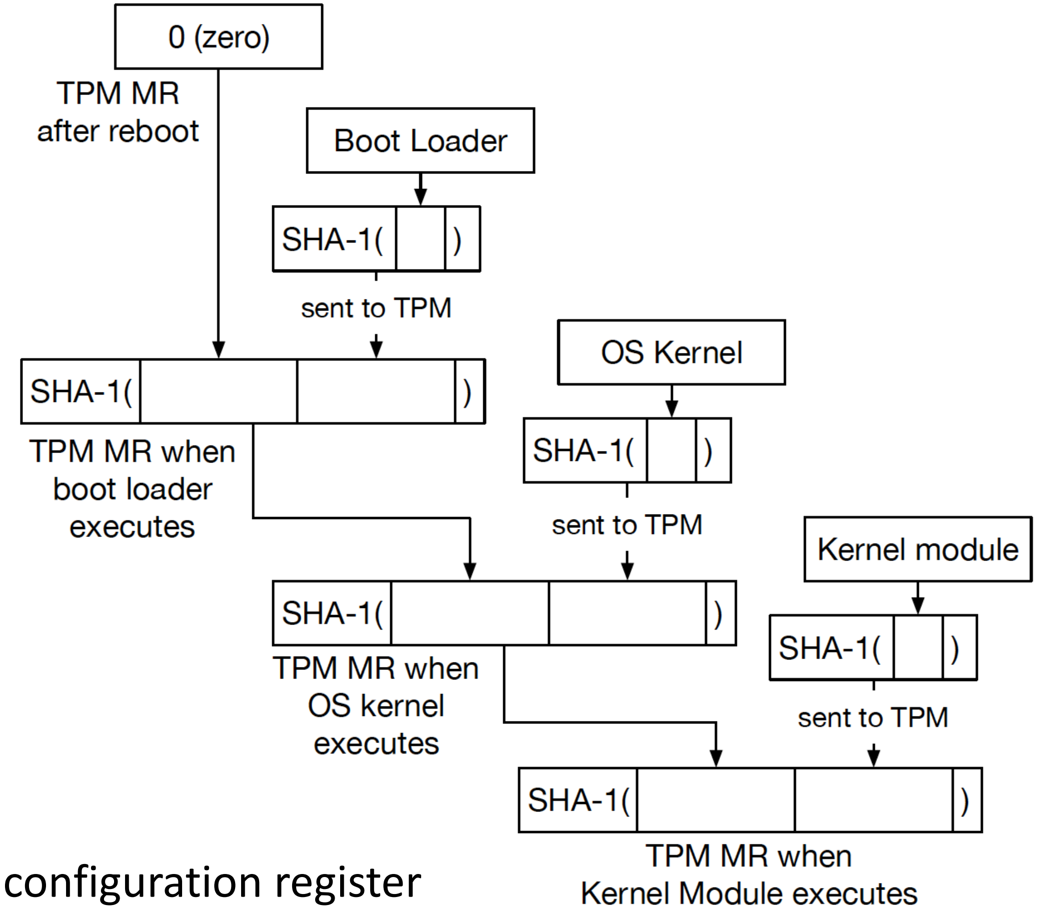
Use as fingerprints

- One-way hash
  - Practically infeasible to invert, Difficult to find collision
- Avalanche effect
  - “Bob Smith got an A+ in ELE386 in Spring 2005”→01eace851b72386c46
  - “Bob Smith got an B+ in ELE386 in Spring 2005”→936f8991c111f2cefaw

Secure Boot using TPM



- Static root of trust for measurement (SRTM)



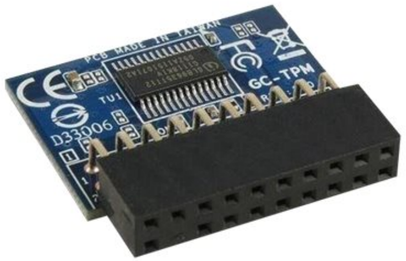
TPM + firmware

Boot Loader

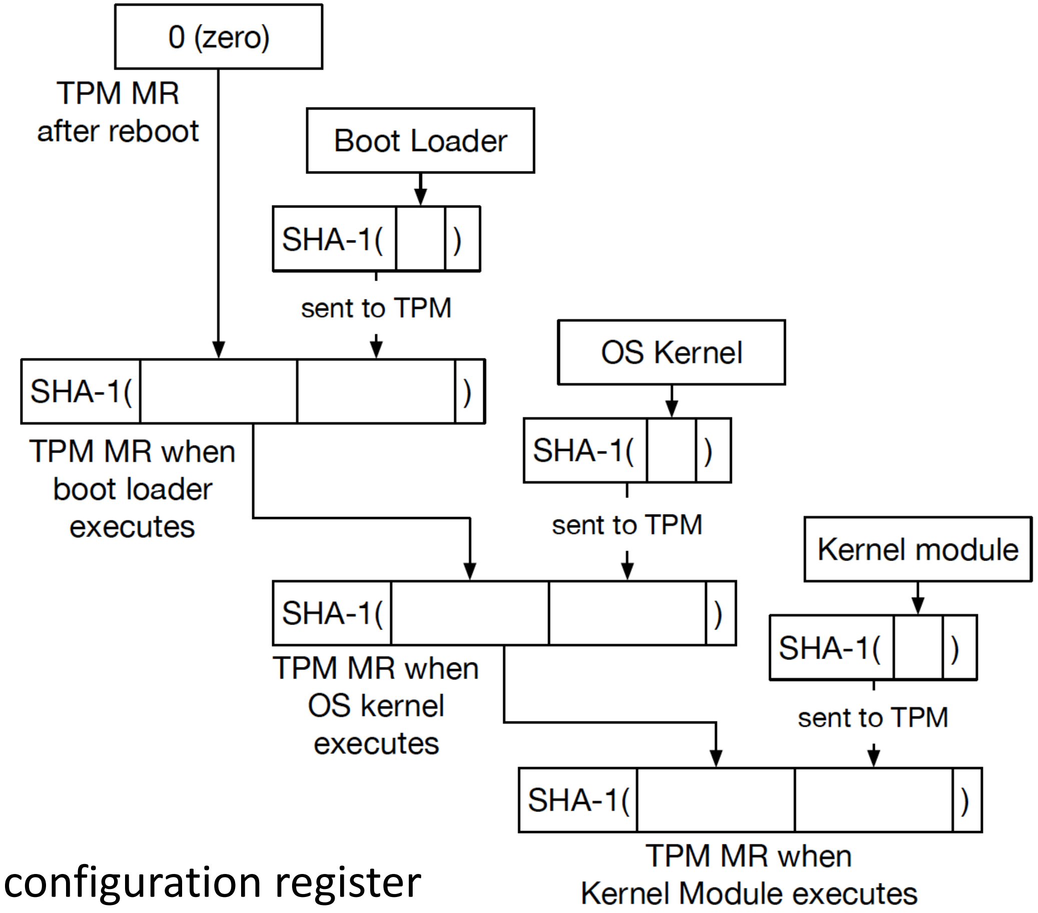
OS kernel

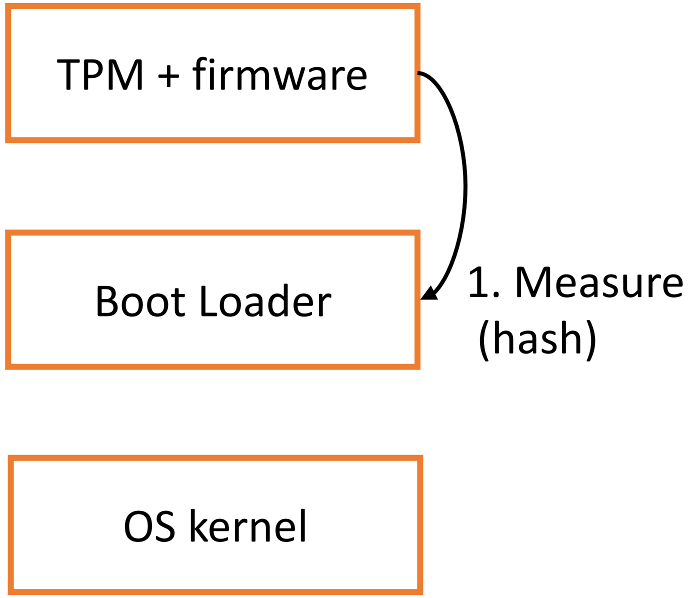
PCR: platform configuration register

Secure Boot using TPM



- Static root of trust for measurement (SRTM)





TPM + firmware

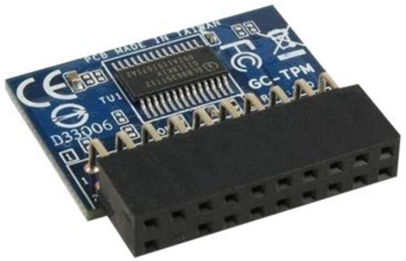
Boot Loader

OS kernel

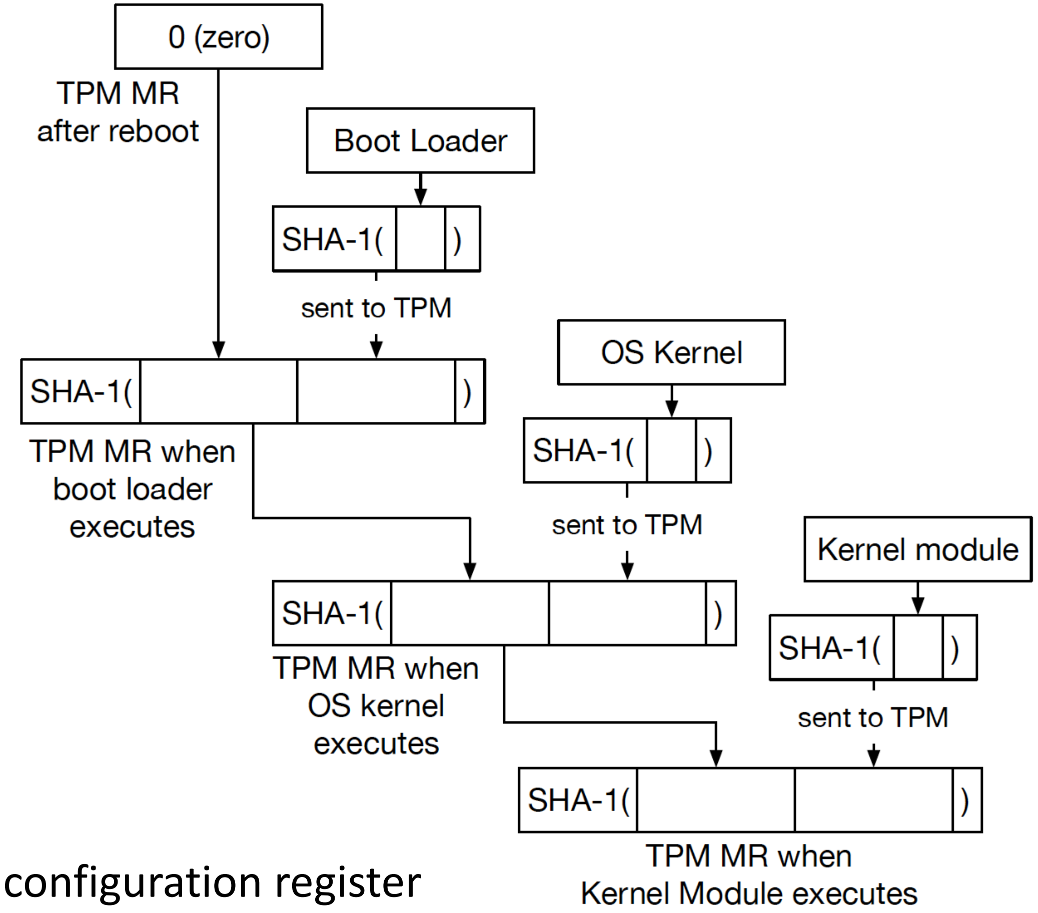
1. Measure
(hash)

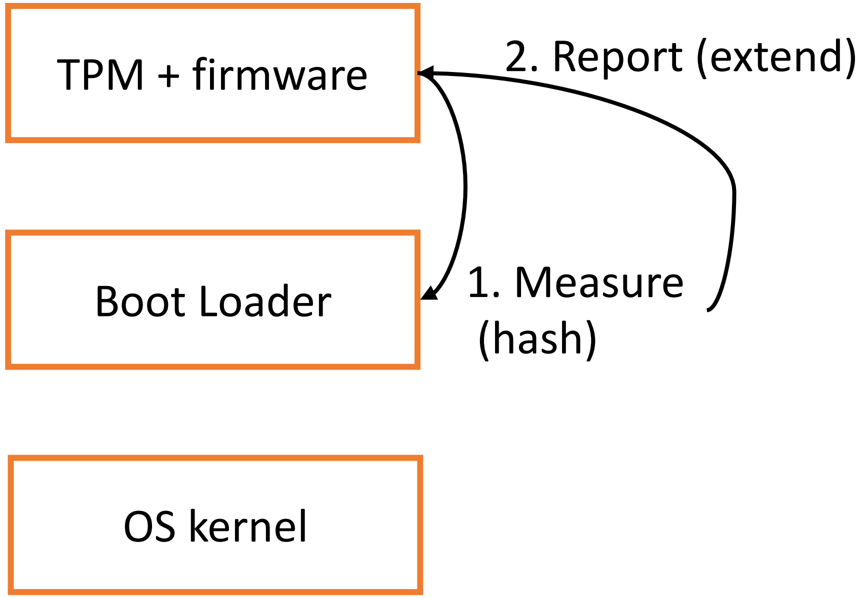
PCR: platform configuration register

Secure Boot using TPM



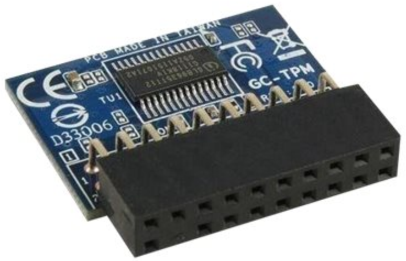
- Static root of trust for measurement (SRTM)



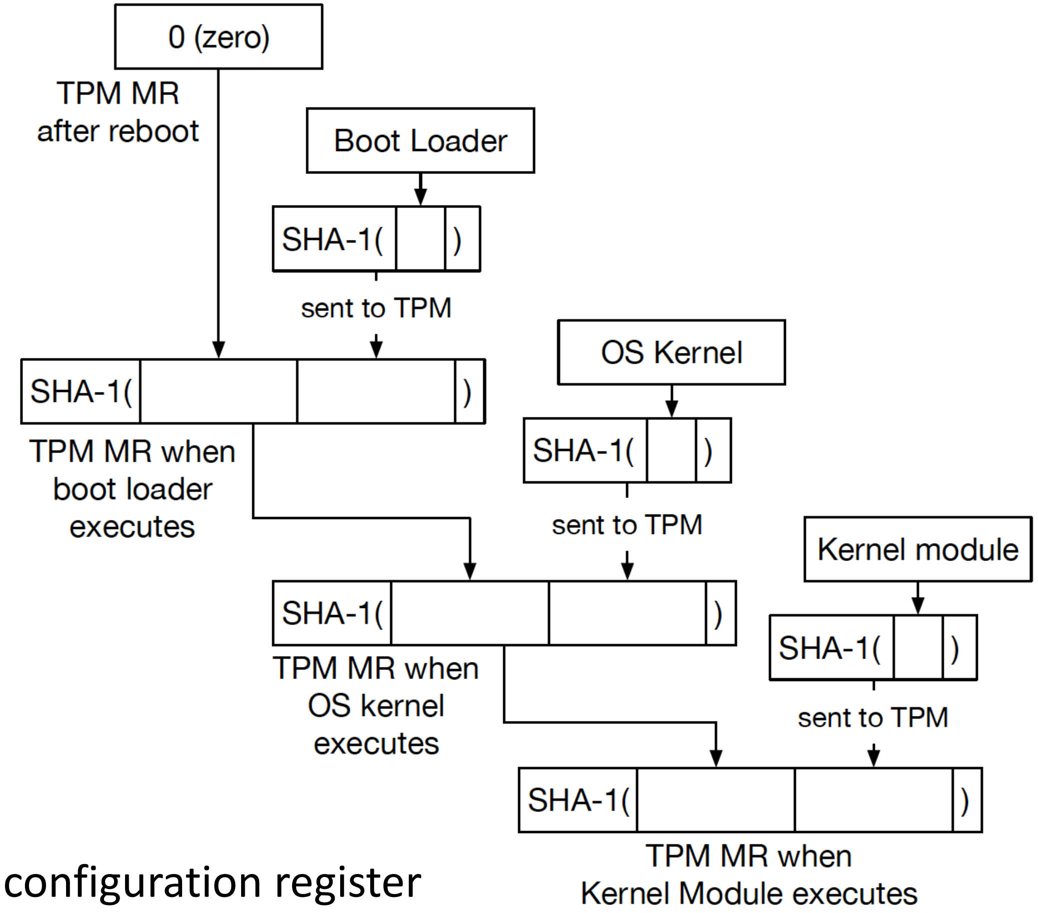


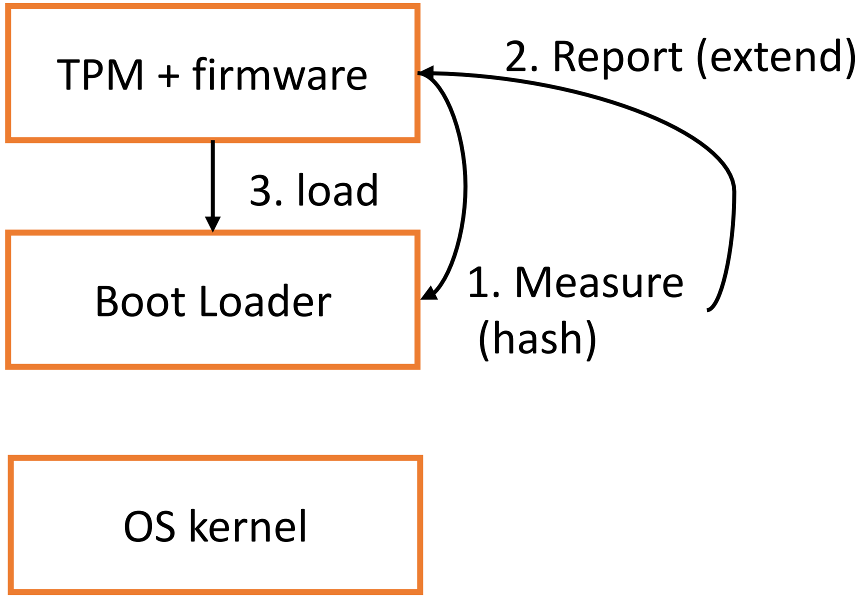
PCR: platform configuration register

Secure Boot using TPM



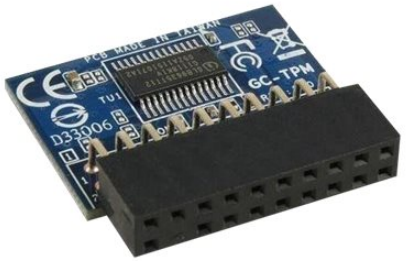
- Static root of trust for measurement (SRTM)



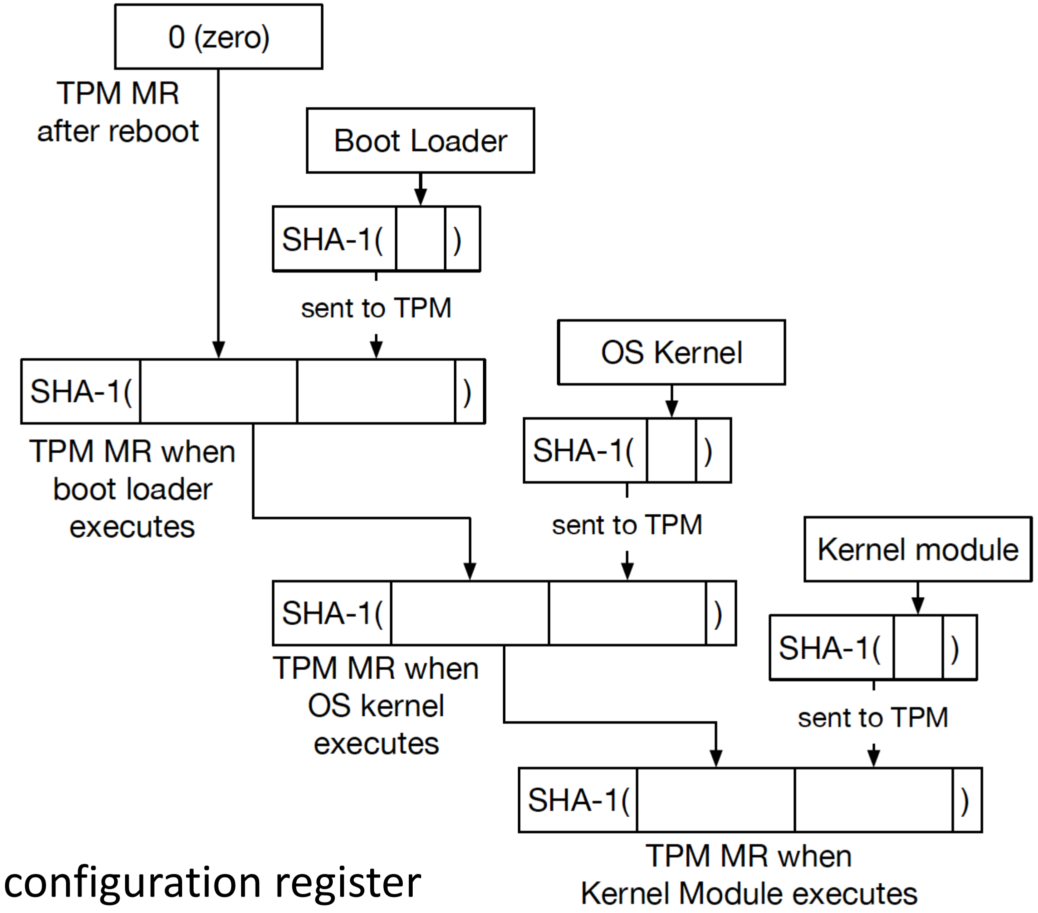


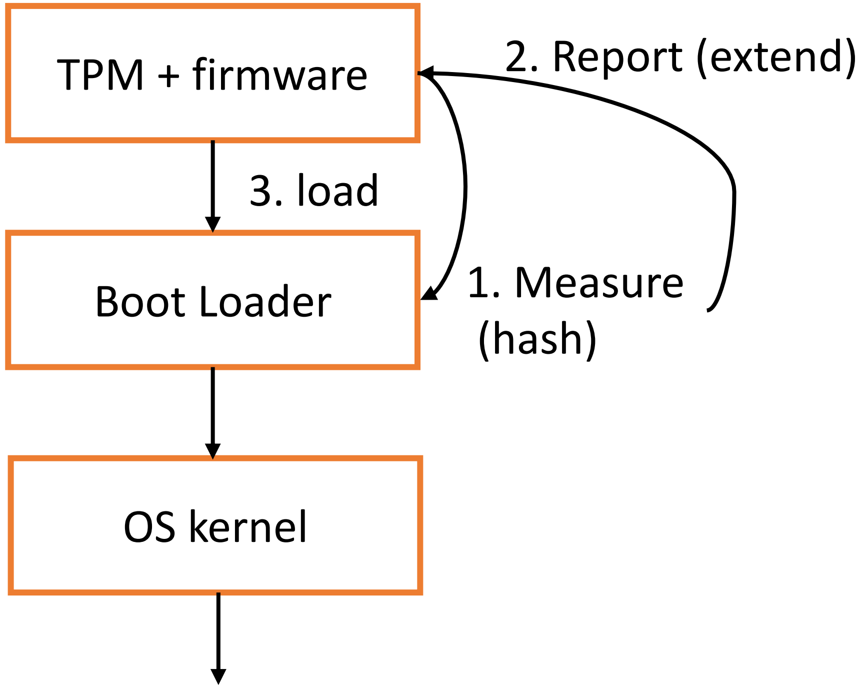
PCR: platform configuration register

Secure Boot using TPM



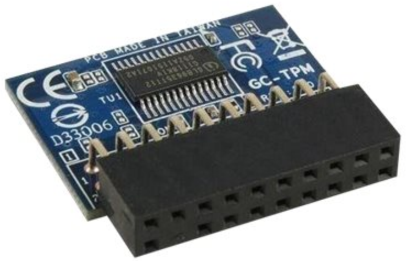
- Static root of trust for measurement (SRTM)



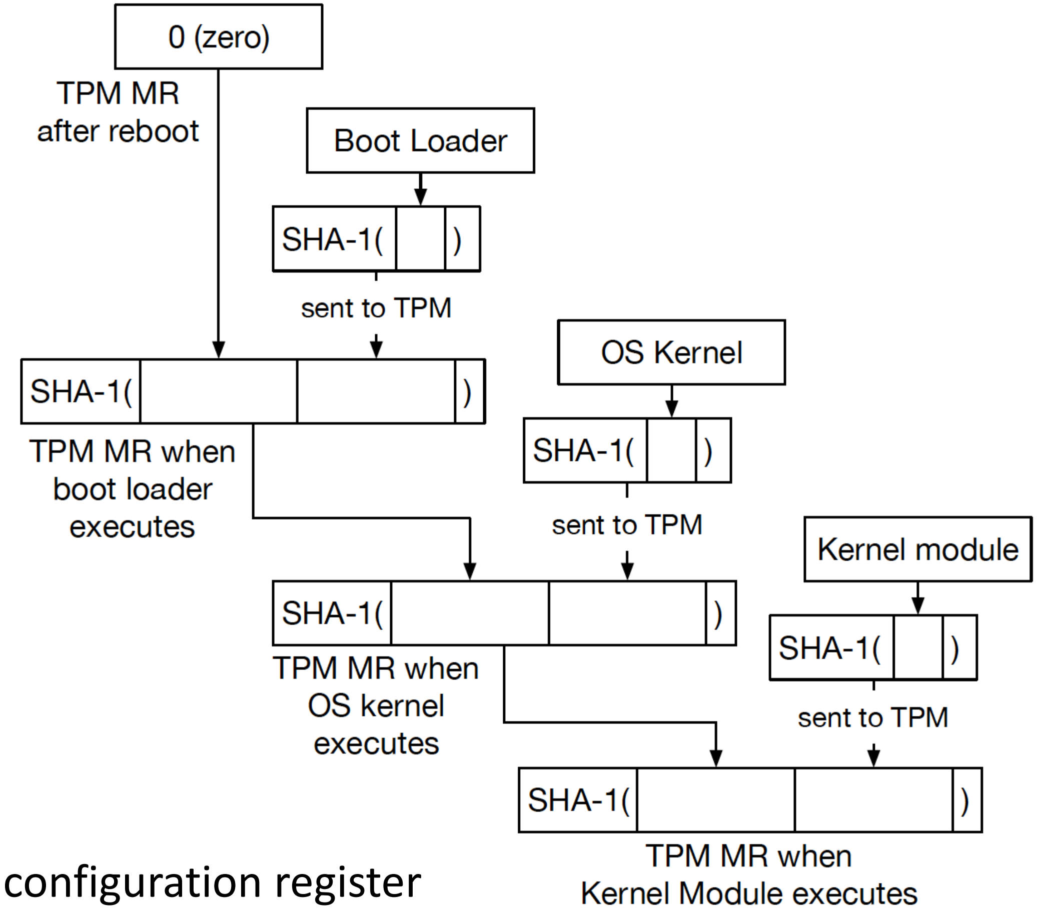


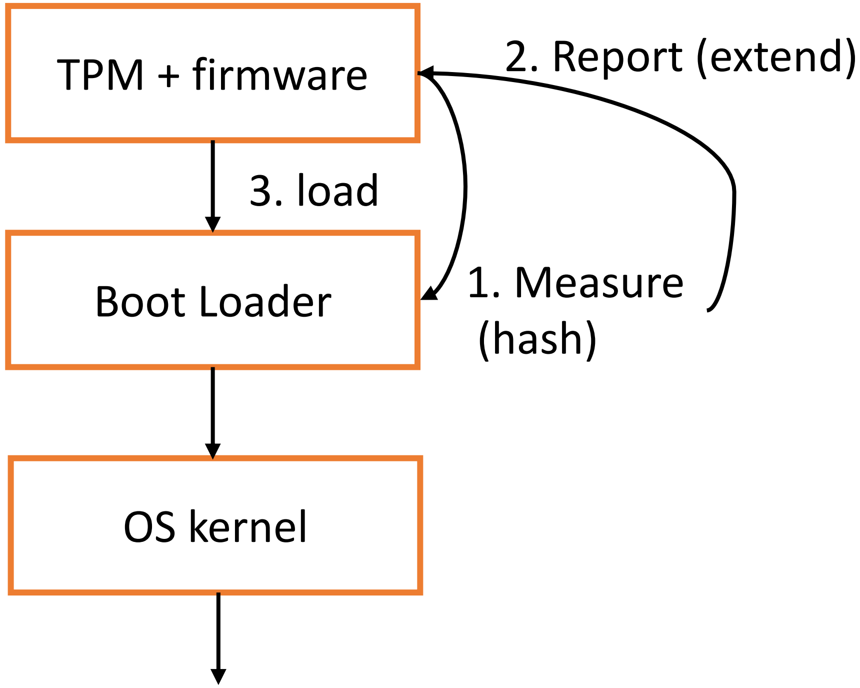
PCR: platform configuration register

Secure Boot using TPM



- Static root of trust for measurement (SRTM)





Compared to expected values locally or submitted to a remote attestor.

PCR: platform configuration register

Software Attestation

- Report a measurement list to a remote verifier

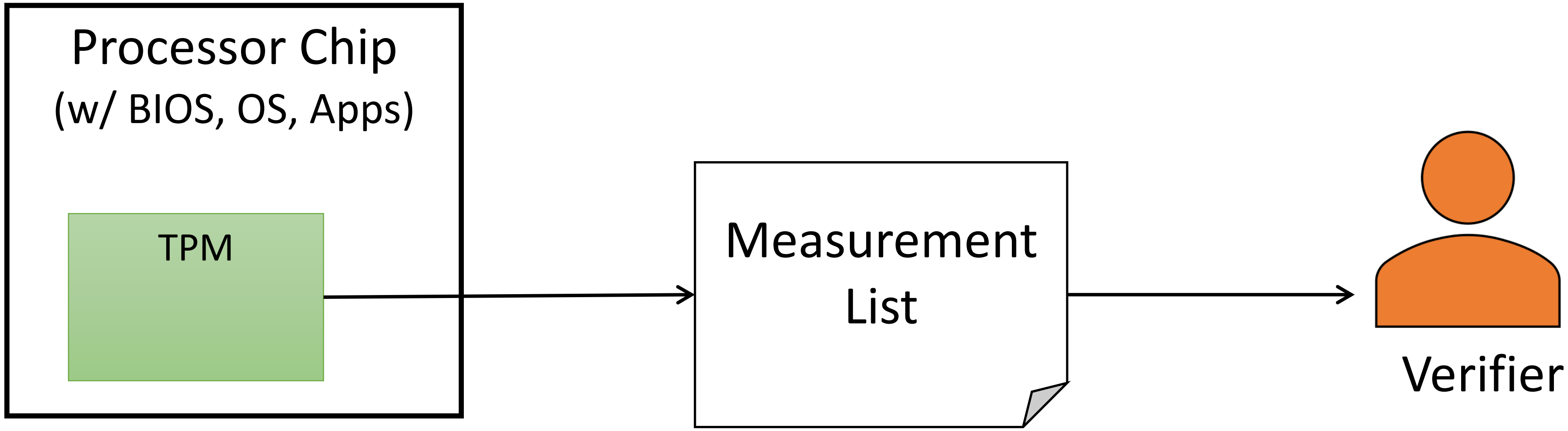
Processor Chip
(w/ BIOS, OS, Apps)

TPM

Verifier

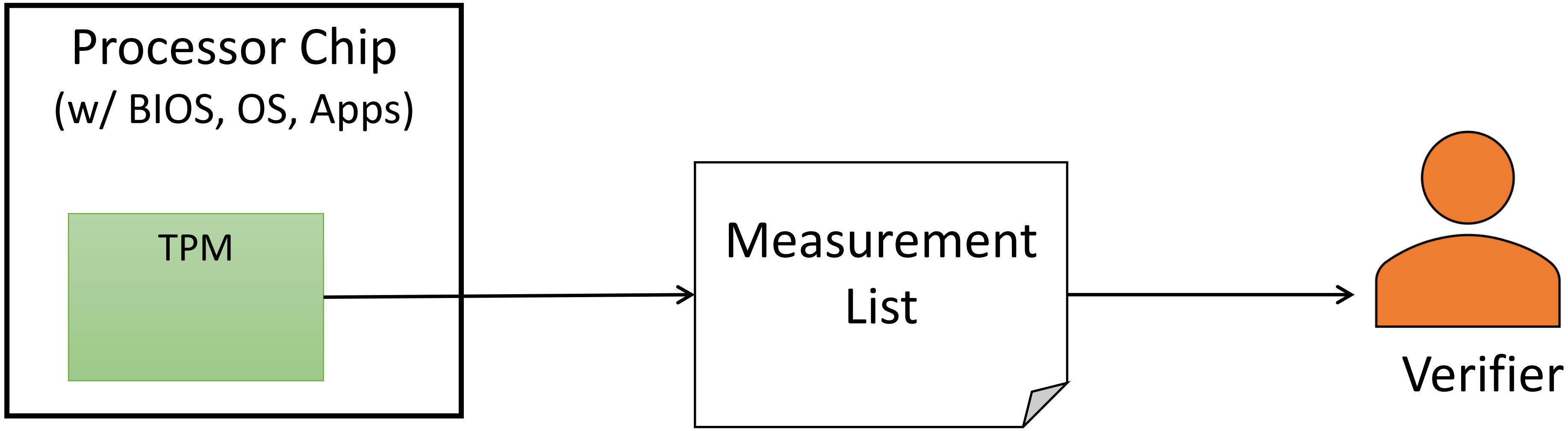
Software Attestation

- Report a measurement list to a remote verifier



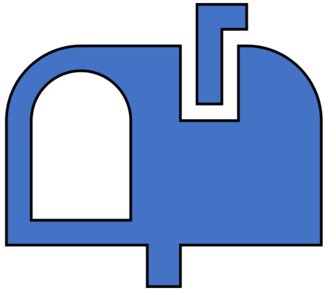
Software Attestation

- Report a measurement list to a remote verifier
- Problem: How can the verifier know the list is not faked?



Public Key Cryptography (e.g., RSA, EC)

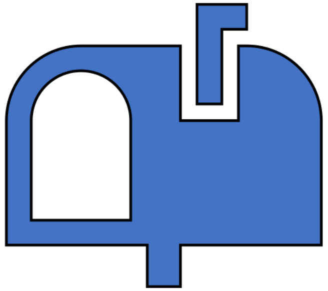
- A pair of keys:
  - Private key (**K_{pri}** – kept as secret); Public key (**K_{pub}** – safe to release publicly)



Mail box is public;
Box key is private

Public Key Cryptography (e.g., RSA, EC)

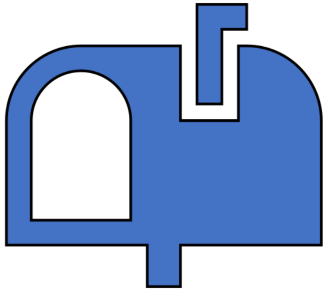
- A pair of keys:
  - Private key (K_{pri} – kept as secret); Public key (K_{pub} – safe to release publicly)
- Encryption:
  - Encrypt(plaintext, K_{pub}) = ciphertext
  - Decrypt(ciphertext, K_{pri}) = plaintext



Mail box is public;
Box key is private

Public Key Cryptography (e.g., RSA, EC)

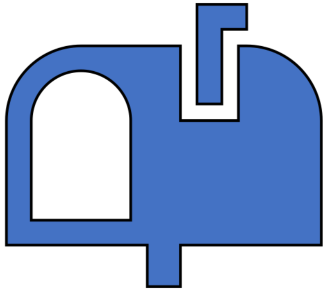
- A pair of keys:
  - Private key (**K_{pri}** – kept as secret); Public key (**K_{pub}** – safe to release publicly)
- Encryption:
  - Encrypt(plaintext, **K_{pub}**) = ciphertext
  - Decrypt(ciphertext, **K_{pri}**) = plaintext
- Digital signatures:
  - Proof that msg comes from *whoever owns private key corresponding to* ***K_{pub}***



Mail box is public;
Box key is private

Public Key Cryptography (e.g., RSA, EC)

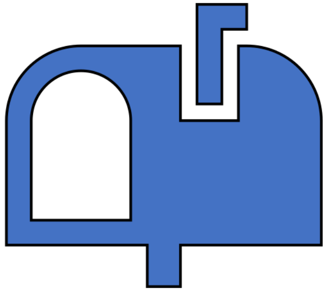
- A pair of keys:
  - Private key (**K_{pri}** – kept as secret); Public key (**K_{pub}** – safe to release publicly)
- Encryption:
  - Encrypt(plaintext, **K_{pub}**) = ciphertext
  - Decrypt(ciphertext, **K_{pri}**) = plaintext
- Digital signatures:
  - Proof that msg comes from *whoever owns private key corresponding to **K_{pub}***
  - Sign(msg):
    - h = Hash(msg); signature = Encrypt(h, **K_{pri}**)
    - Return {signature, msg}



Mail box is public;
Box key is private

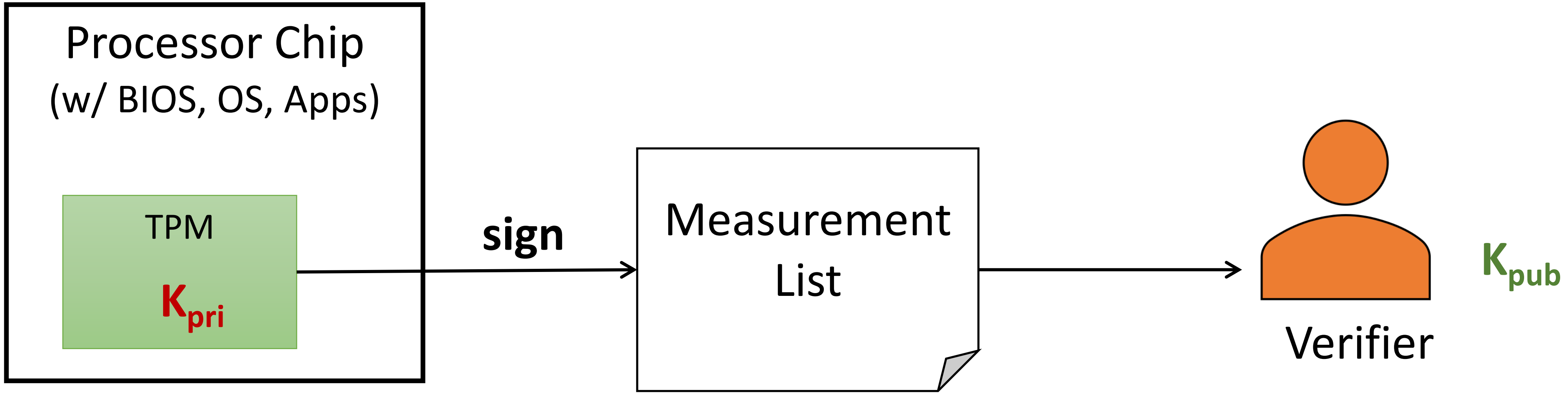
Public Key Cryptography (e.g., RSA, EC)

- A pair of keys:
  - Private key (**K_{pri}** – kept as secret); Public key (**K_{pub}** – safe to release publicly)
- Encryption:
  - Encrypt(plaintext, **K_{pub}**) = ciphertext
  - Decrypt(ciphertext, **K_{pri}**) = plaintext
- Digital signatures:
  - Proof that msg comes from *whoever owns private key corresponding to* ***K_{pub}***
  - Sign(msg):
    - h = Hash(msg); signature = Encrypt(h, **K_{pri}**)
    - Return {signature, msg}
  - Verify:
    - Decrypt(signature, **K_{pub}**) ?= Hash(msg)

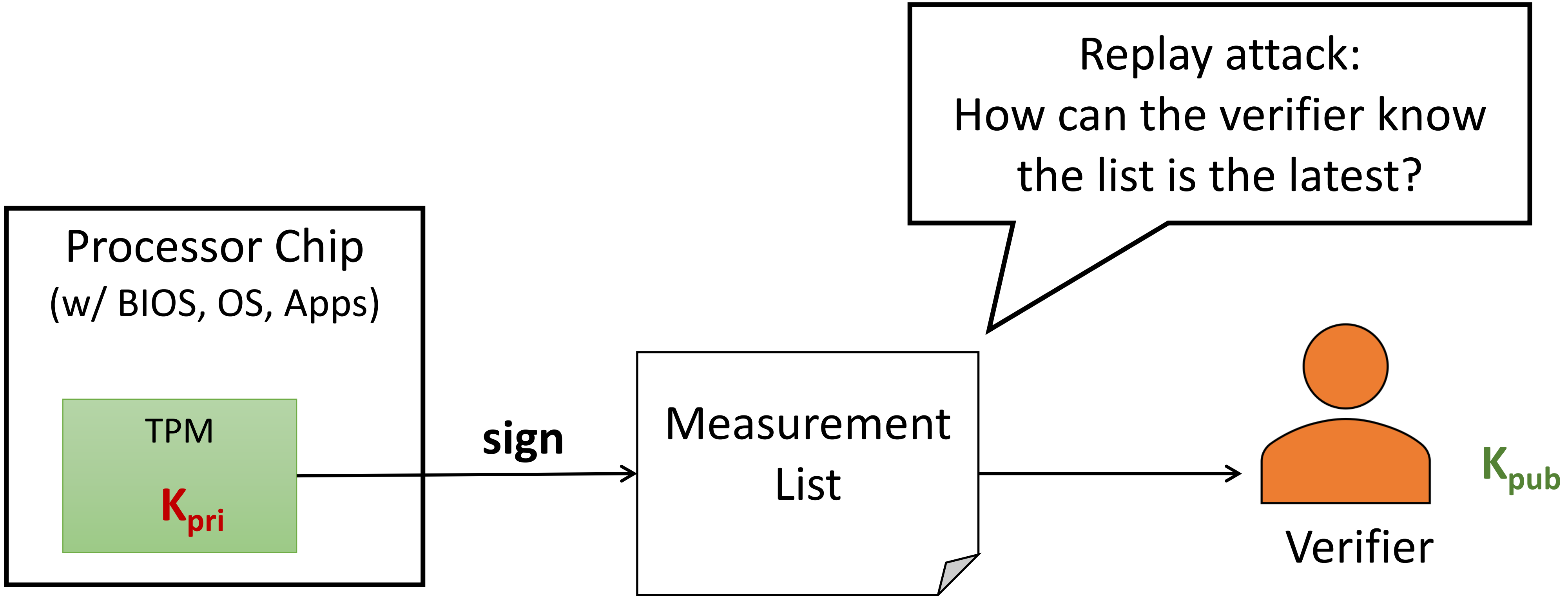


Mail box is public;
Box key is private

Software Attestation



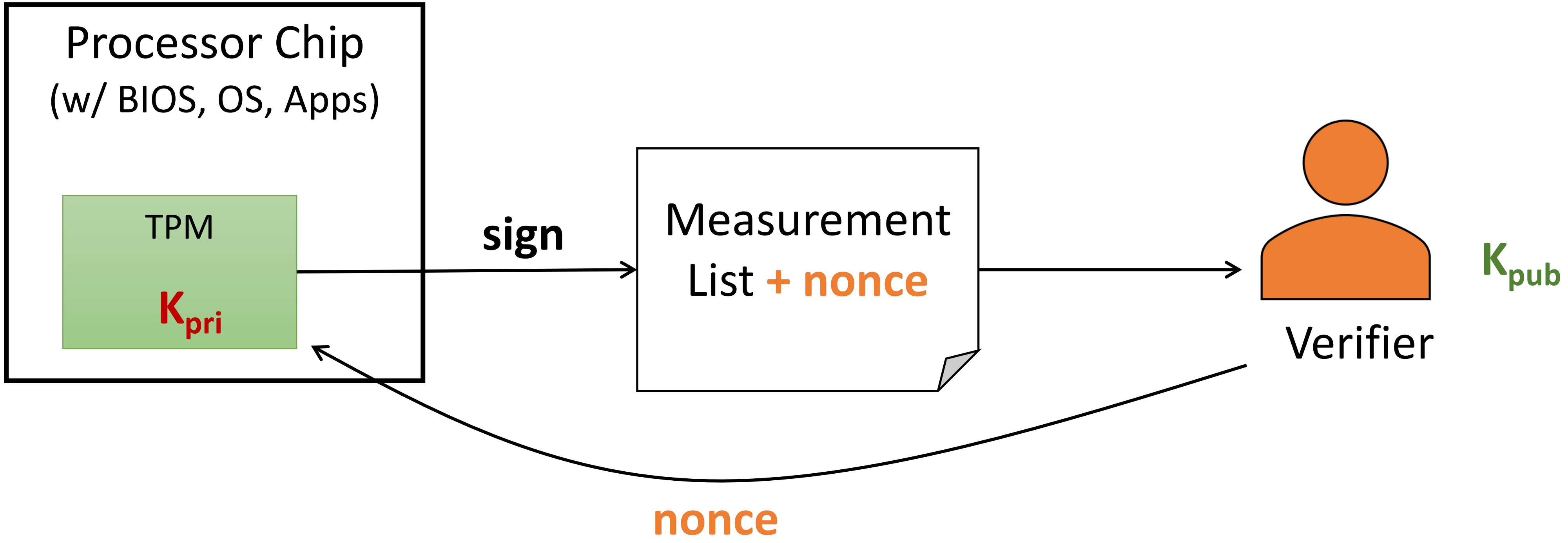
Software Attestation



Replay attack:
How can the verifier know the list is the latest?

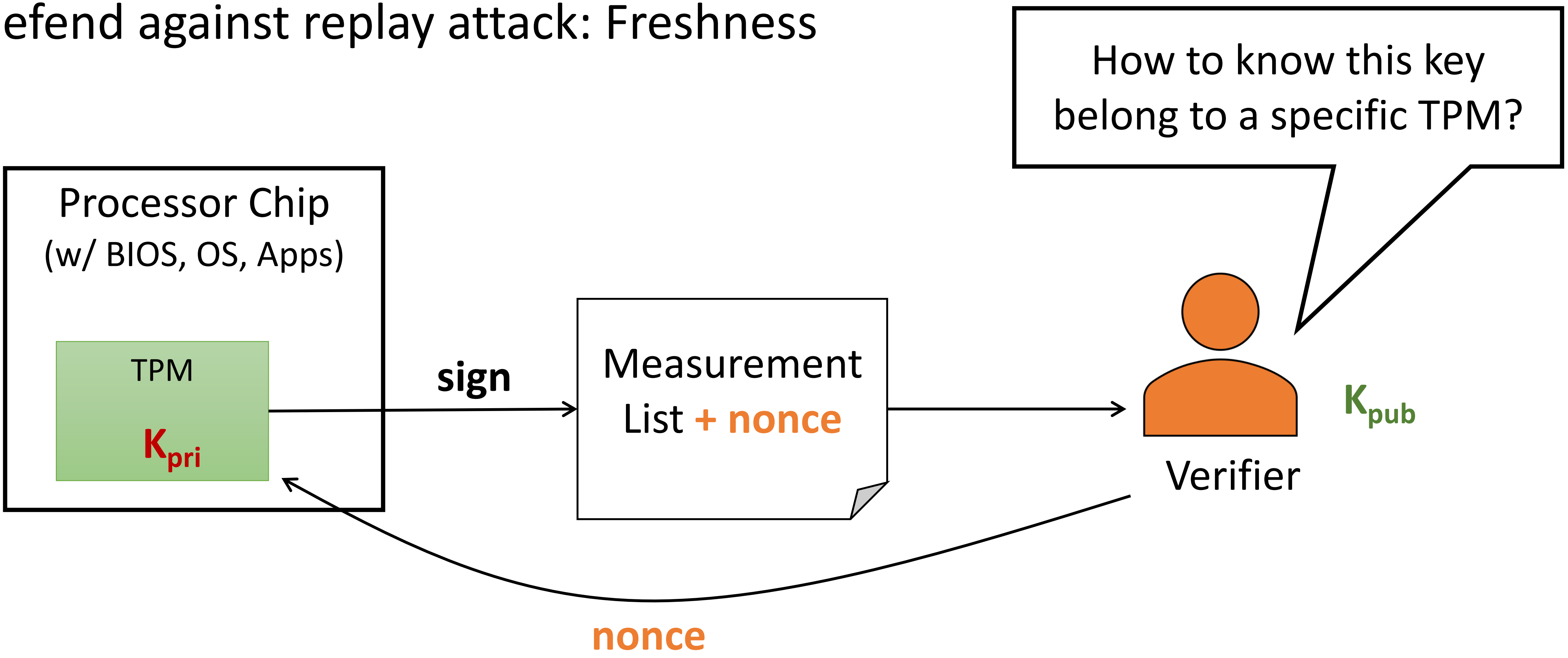
Software Attestation

- Defend against replay attack: Freshness



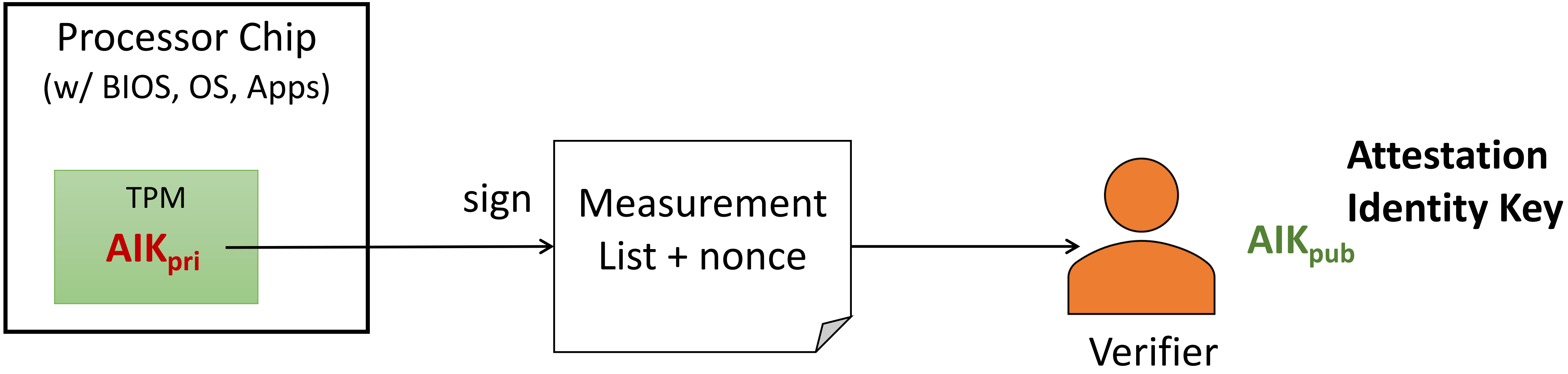
Software Attestation

- Defend against replay attack: Freshness



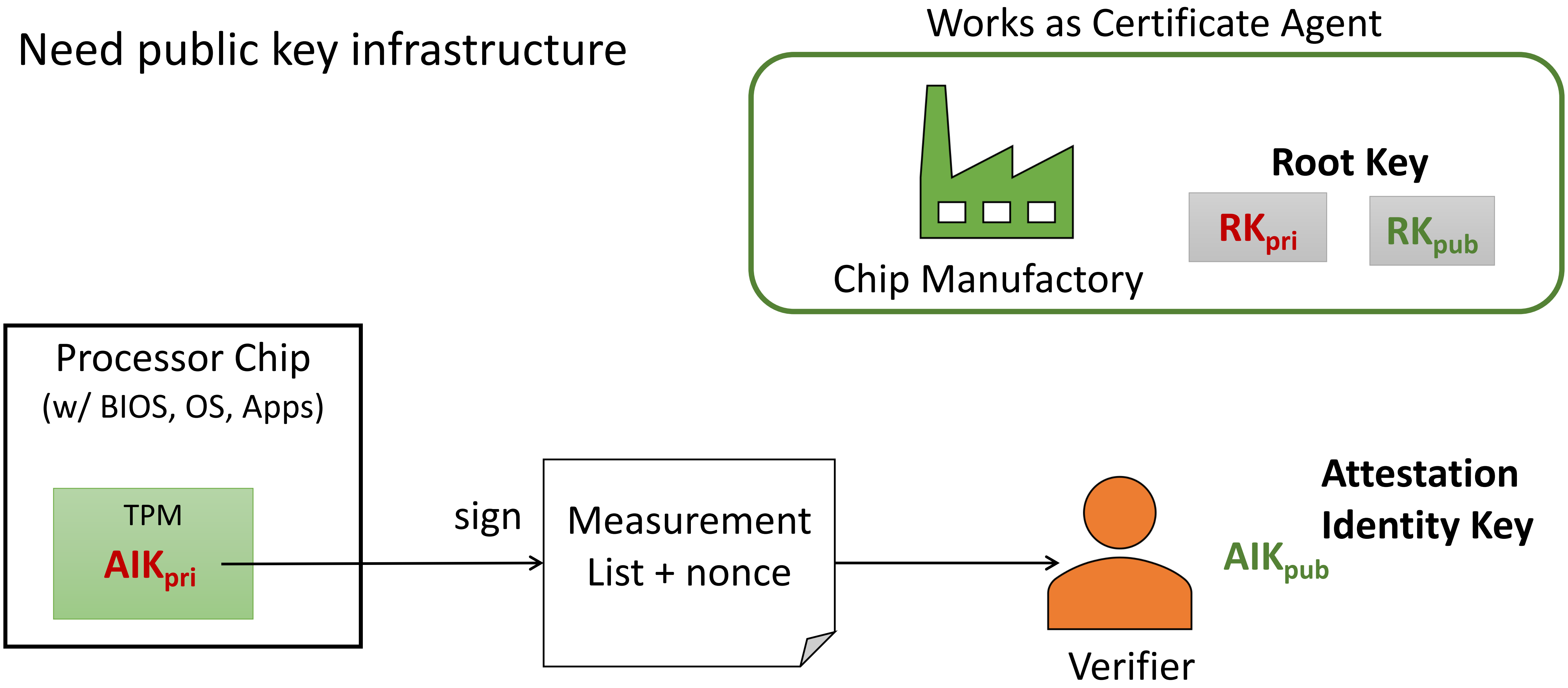
Software Attestation

- Need public key infrastructure



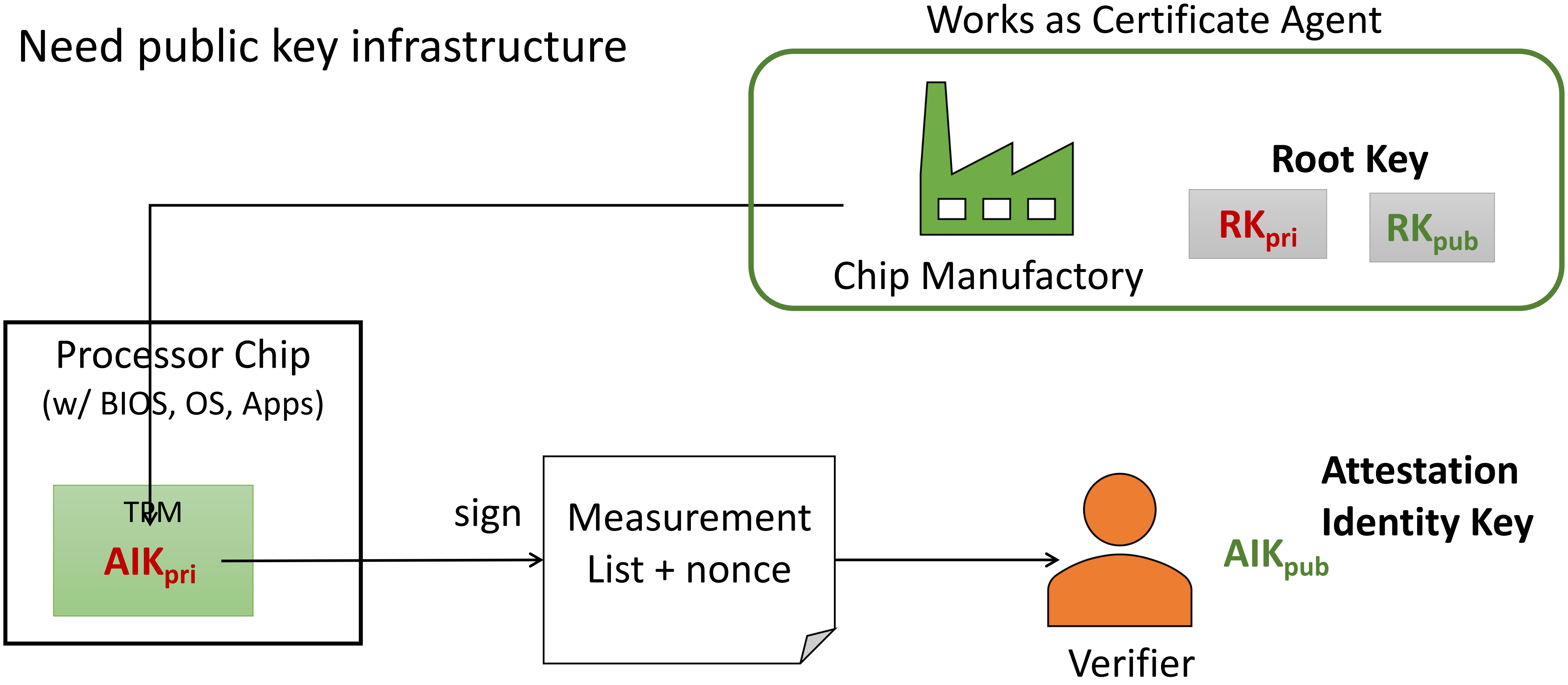
Software Attestation

- Need public key infrastructure



Software Attestation

- Need public key infrastructure



Works as Certificate Agent

Chip Manufactory

Root Key

RK_{pri} RK_{pub}

Processor Chip
(w/ BIOS, OS, Apps)

TPM

AIK_{pri}

sign

Measurement List + nonce

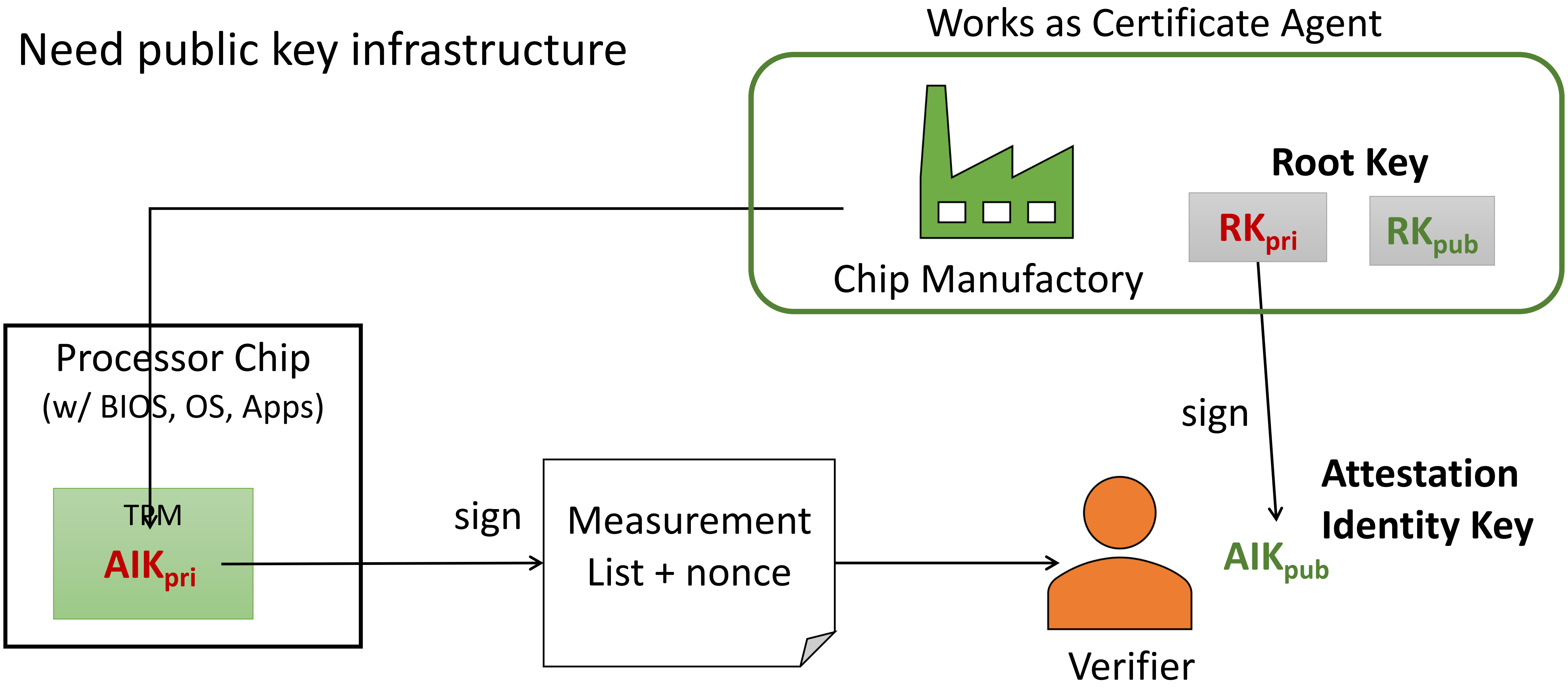
Verifier

AIK_{pub}

Attestation Identity Key

Software Attestation

- Need public key infrastructure



Security Objectives Summary

- Privacy
  - Alice sends msg *m* to Bob. Only Bob should be able to read *m*. (asymmetric or symmetric encryption)

- Integrity
  - Alice sends msgs *m1 … mn* to Bob.
  - Authenticity: Bob receives msg *p*. Bob can verify *p \in m1 … mn*. (Hash)
  - Freshness: Bob has received msgs *p1 … pn*. Bob can verify *pi = mi*. (Hash+nonce)

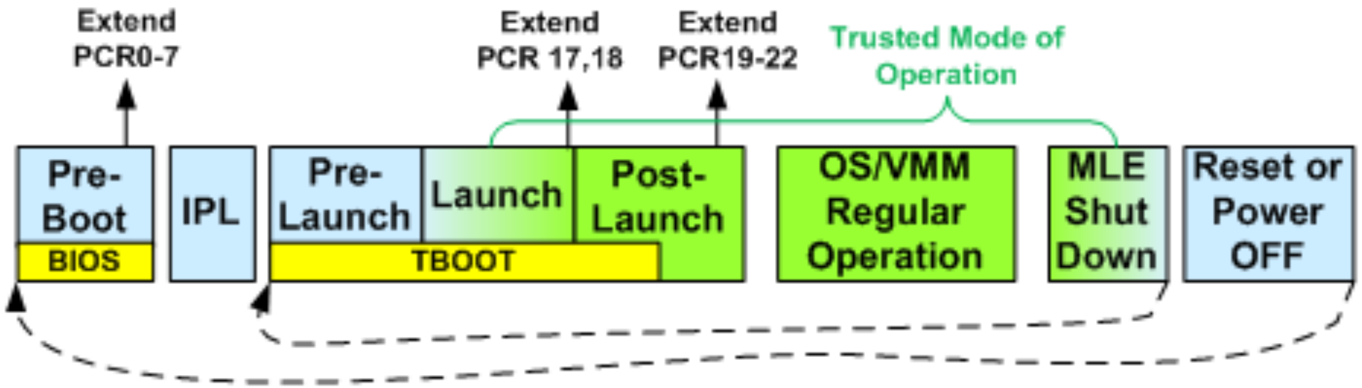
- Identity
  - Bob wants to know if Alice is really Alice.

- Availability
  - Does Bob ever see the n messages?

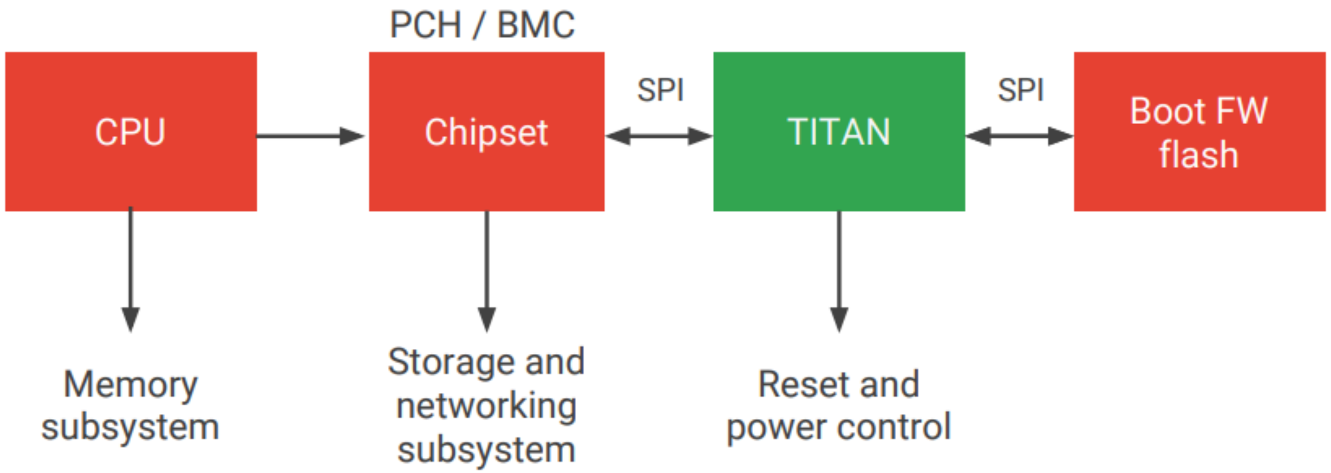
Protocols can be constructed using crypto primitives and infrastructures

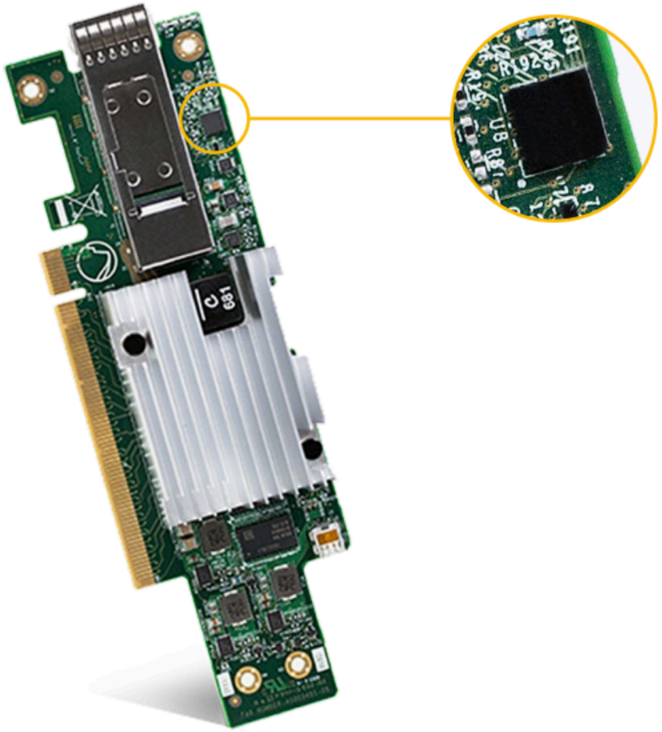
Intel TXT

- Uses TPM for software attestation
- Dynamic root of trust for measurement (DRTM)
  - PCRs 17-22 are reset by the SINIT ACM, every time a TXT VM is launched
- Marketed as more secure, but there are various attacks targeting TXT



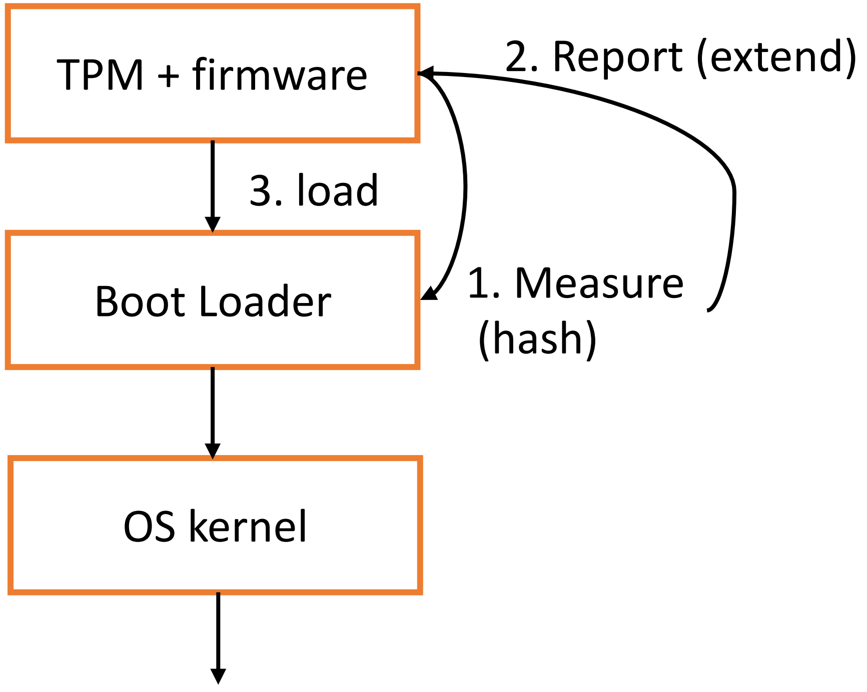
Open-source Choice: Google Titan





from https://www.hotchips.org/hc30/1conf/1.14_Google_Titan_GoogleFinalTitanHotChips2018.pdf

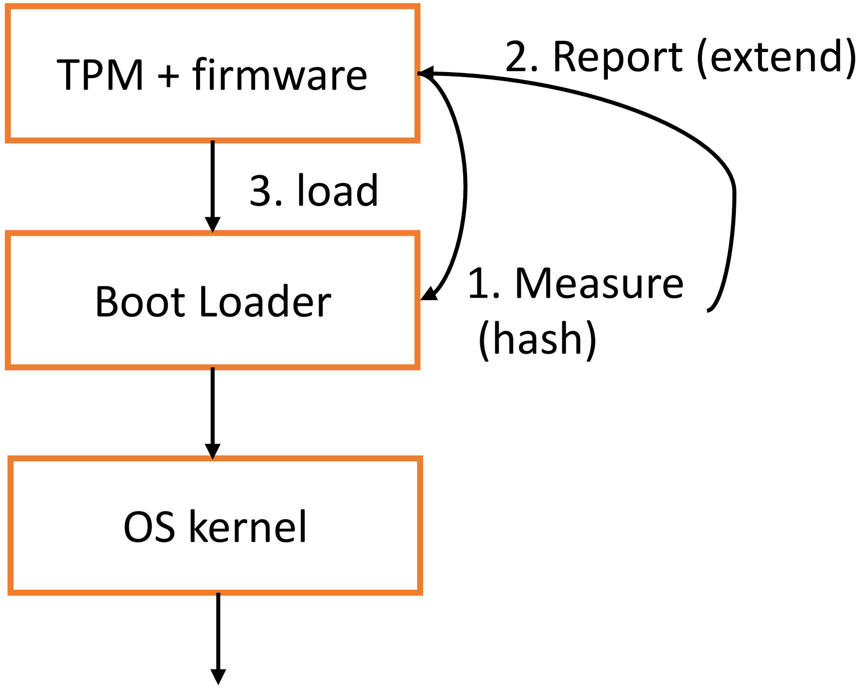
Security Vulnerabilities of Using TPM



Han et al. A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping. Usenix Security'18
Wojtczuk et al. Attacking Intel TXT® via SINIT code execution hijacking. 2011

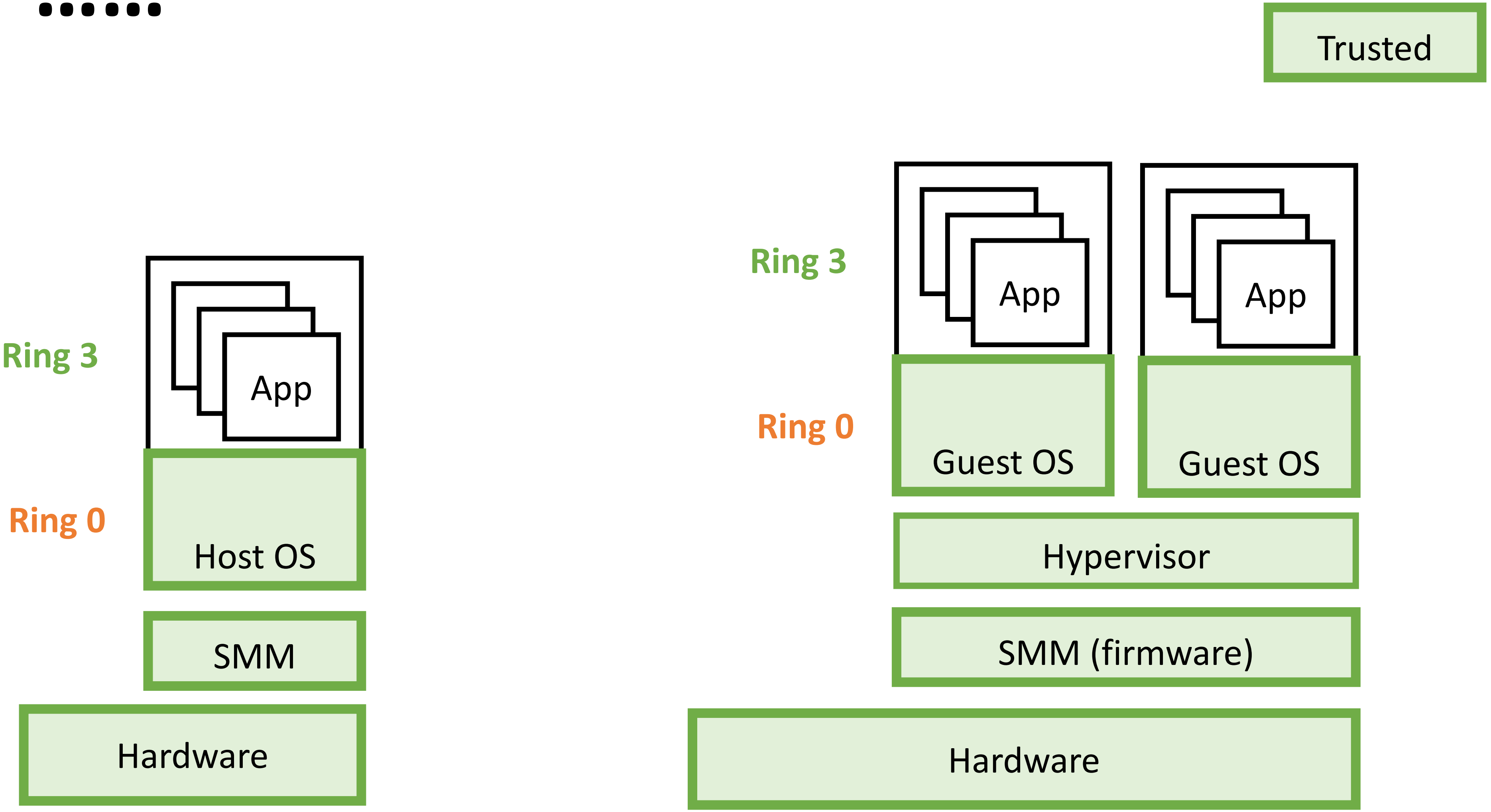
Security Vulnerabilities of Using TPM

- Vulnerable to bus sniffing attacks
- TPM Reset attacks
  - SW reports hash values
- Bugs in the trusted software



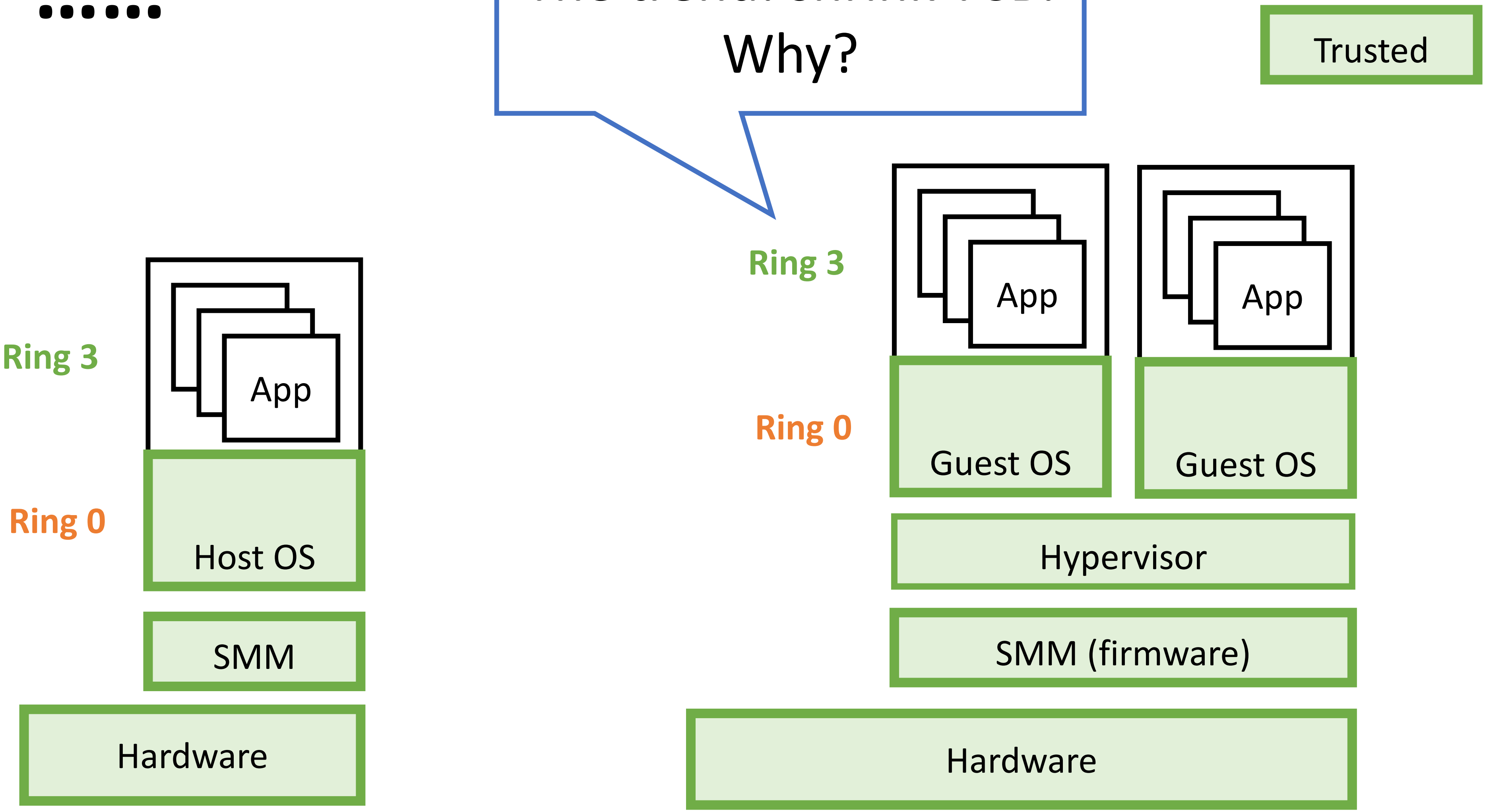
Han et al. A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping. Usenix Security'18
Wojtczuk et al. Attacking Intel TXT® via SINIT code execution hijacking. 2011

So Far ……

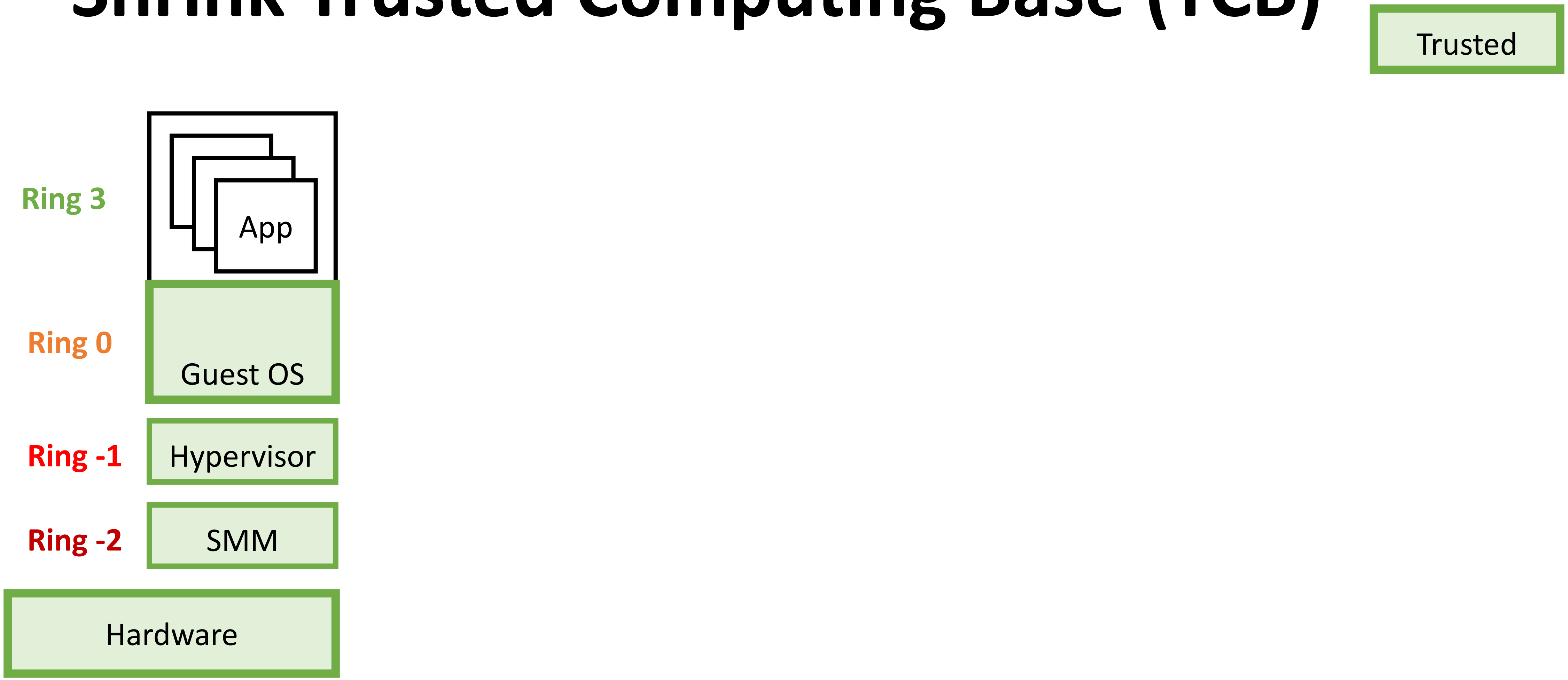

Trusted
Ring 3
App
Ring 0
Host OS
SMM
Hardware
Ring 3
App
App
Ring 0
Guest OS
Guest OS
Hypervisor
SMM (firmware)
Hardware

So Far ……

The trend: shrink TCB. Why?

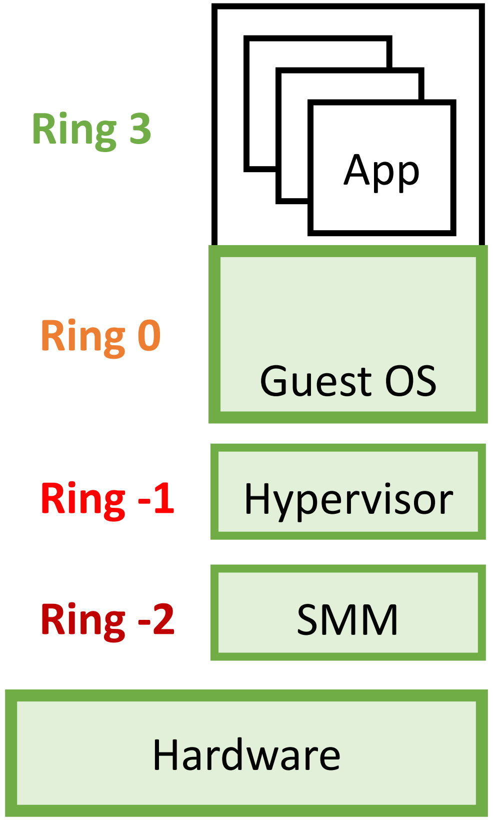


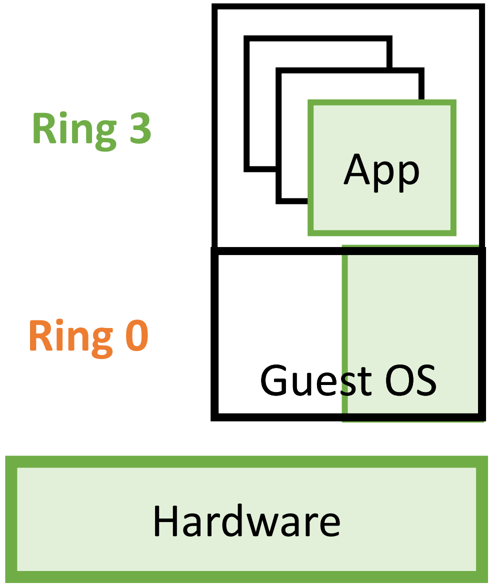
Shrink Trusted Computing Base (TCB)



Shrink Trusted Computing Base (TCB)

🟩 Trusted

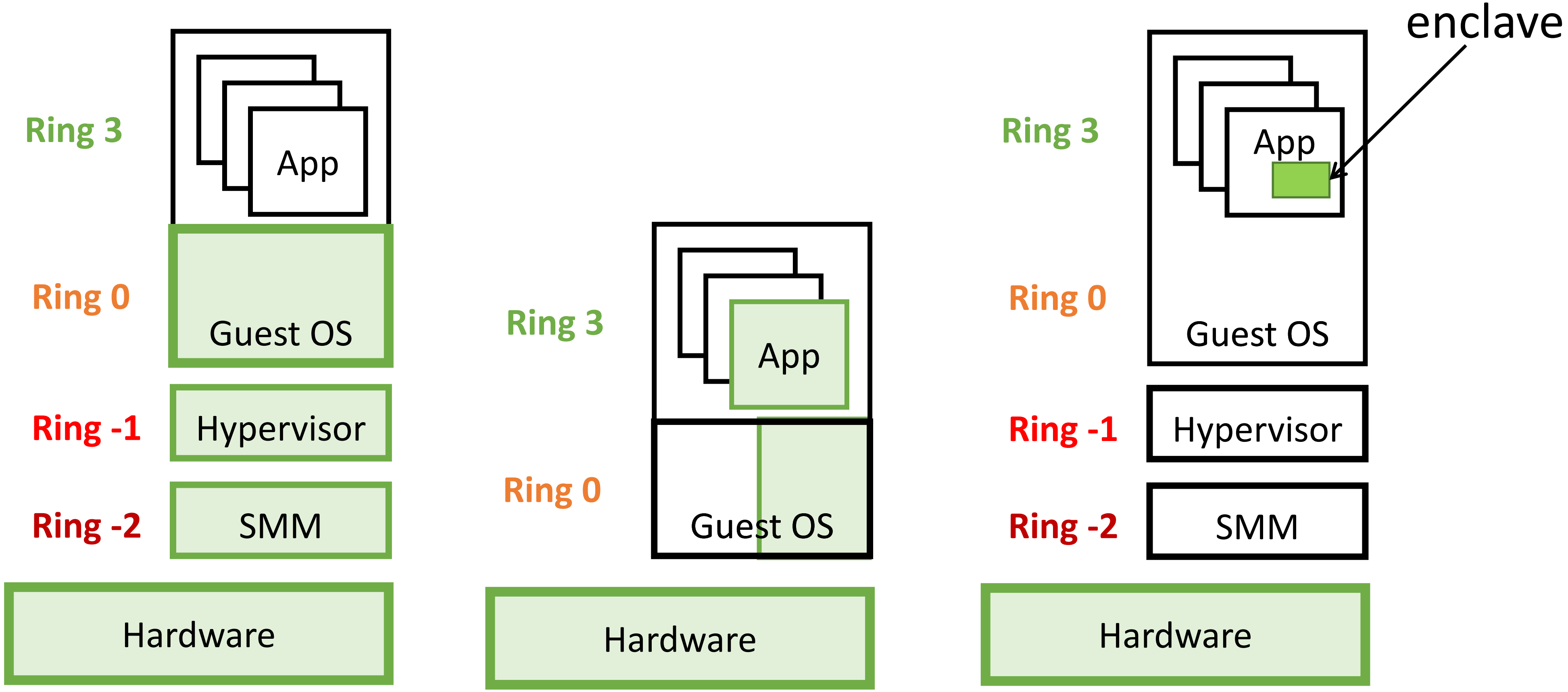




Arm TrustZone

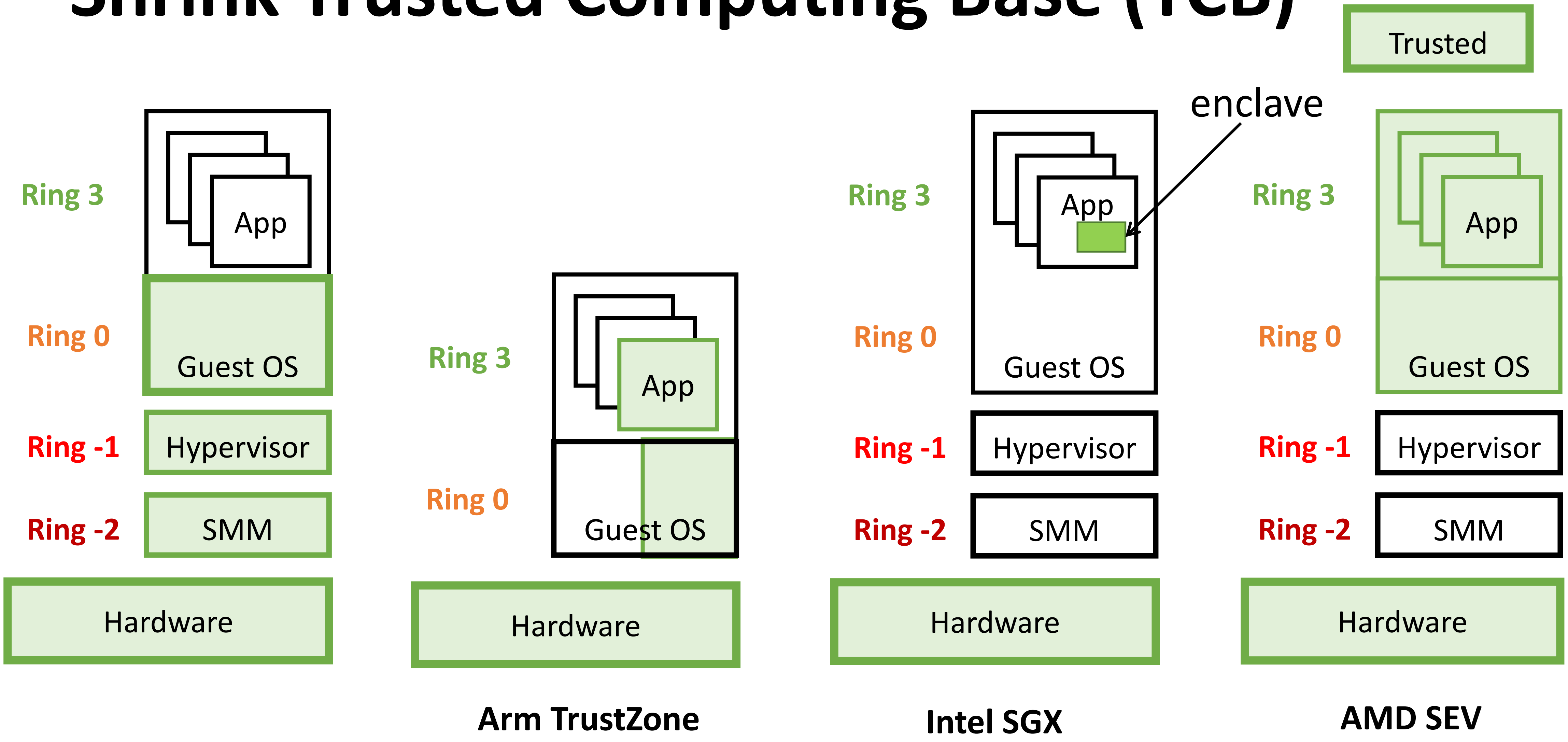
Shrink Trusted Computing Base (TCB)

🟩 Trusted

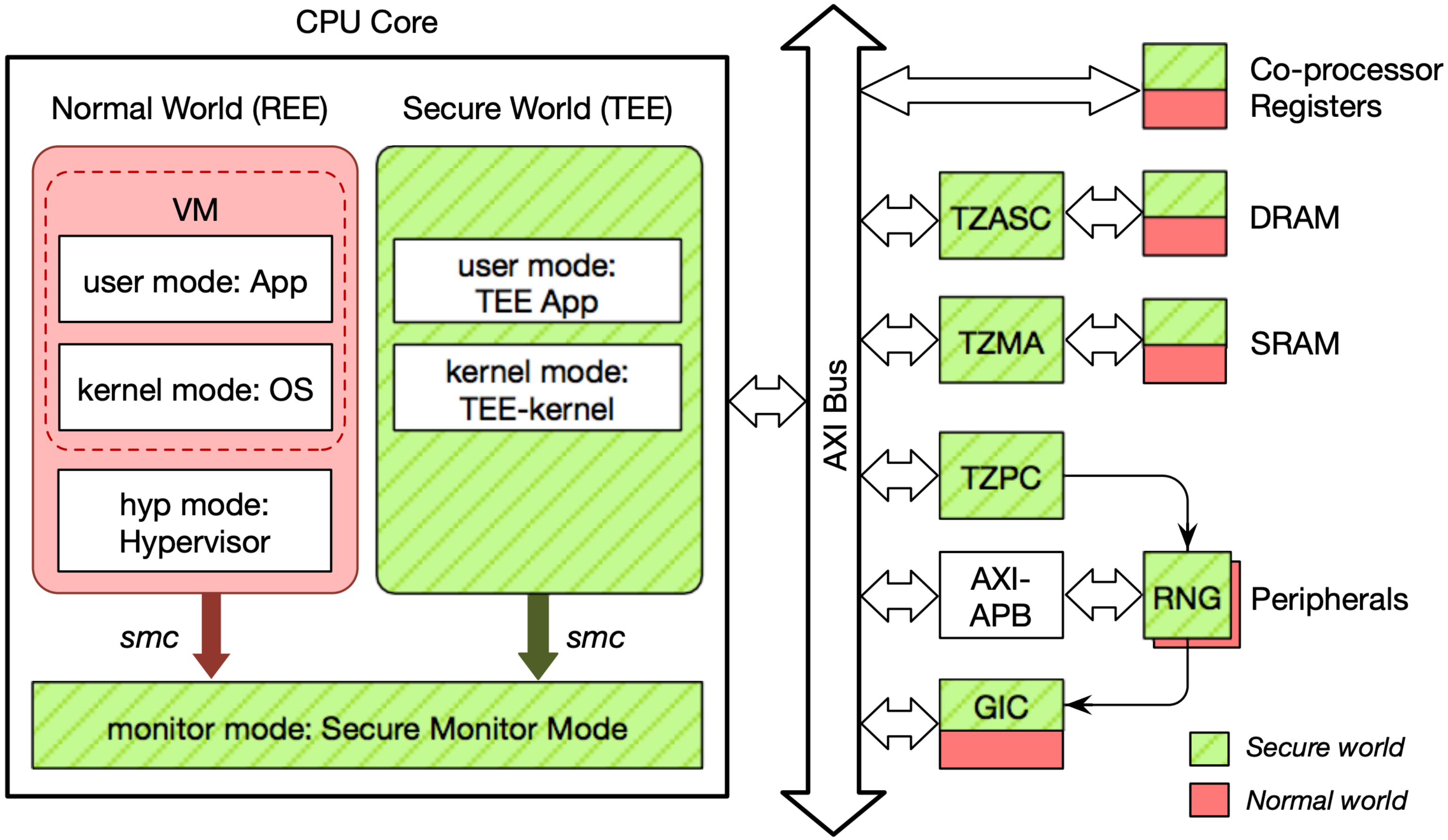


Arm TrustZone **Intel SGX**

Shrink Trusted Computing Base (TCB)

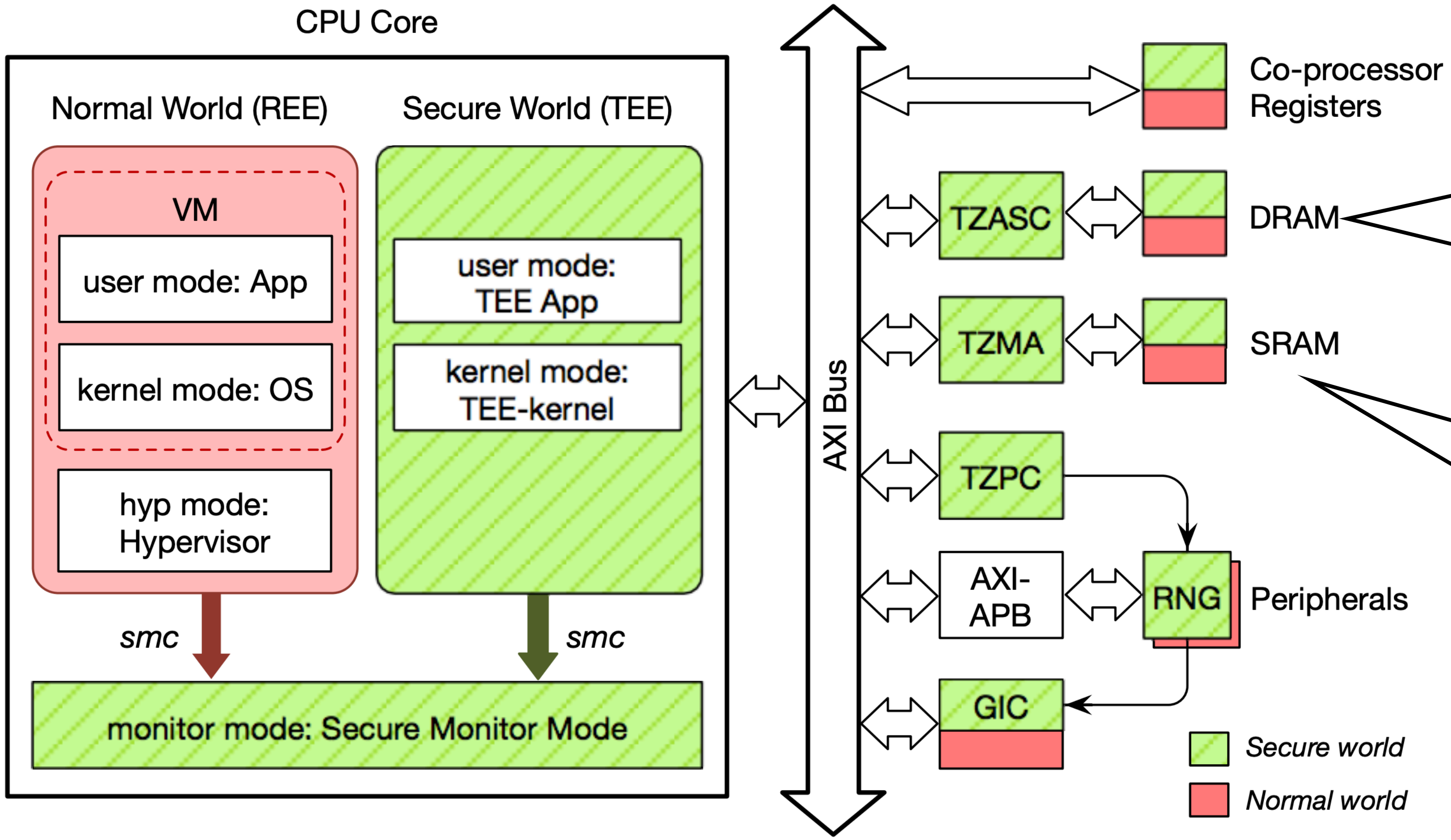


Arm TrustZone



from Hua et al. vTZ: Virtualizing ARM TrustZone. Usenix'17

Arm TrustZone



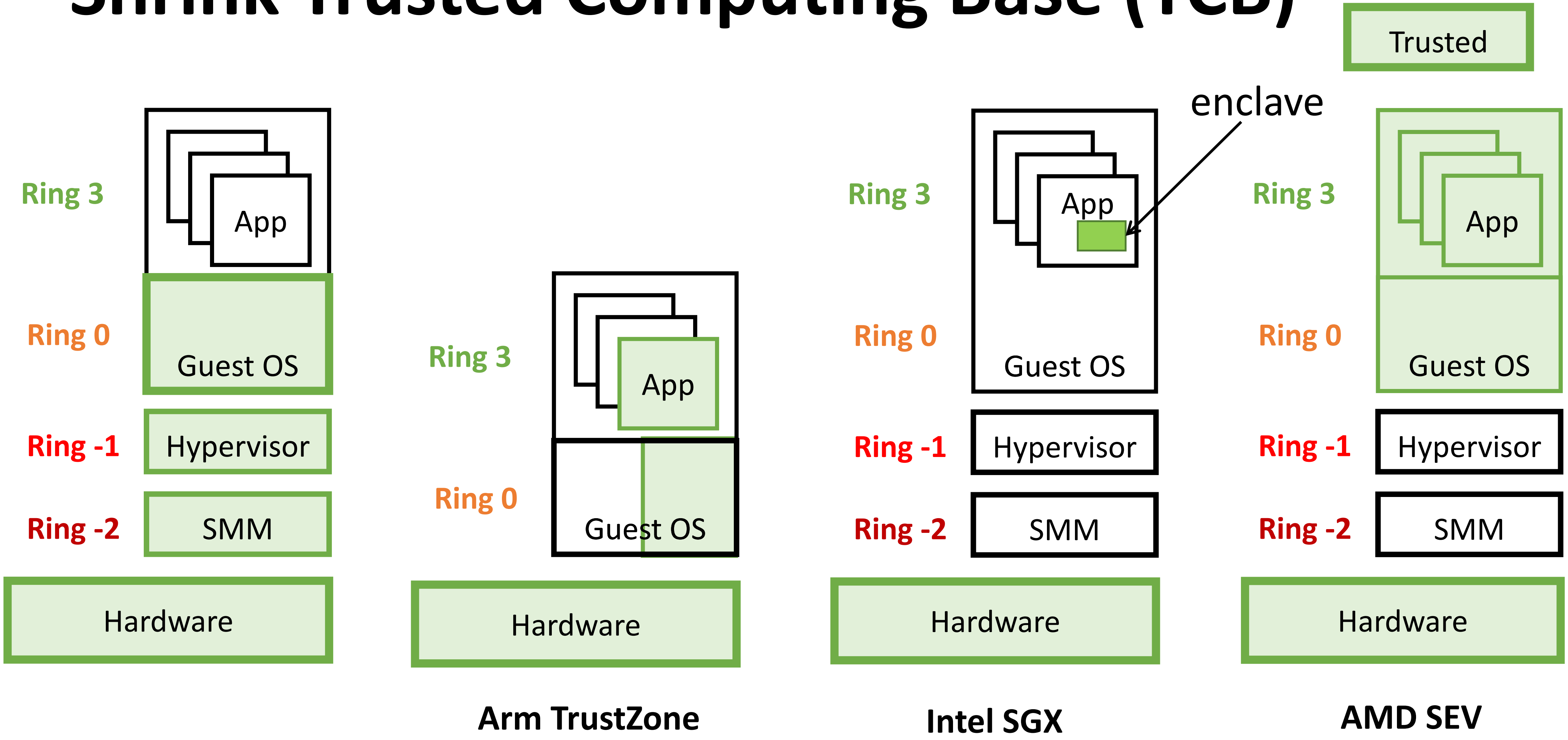
Recommend not to have secret data in DRAM.

But just recommendation.

Cache is dynamically partitioned by tagging each cache line with a security bit.

from Hua et al. vTZ: Virtualizing ARM TrustZone. Usenix'17

Shrink Trusted Computing Base (TCB)



Next Lecture:
Intel SGX