Hardware Security Module

Mengjia Yan

Spring 2025

Secure Processors/HSM

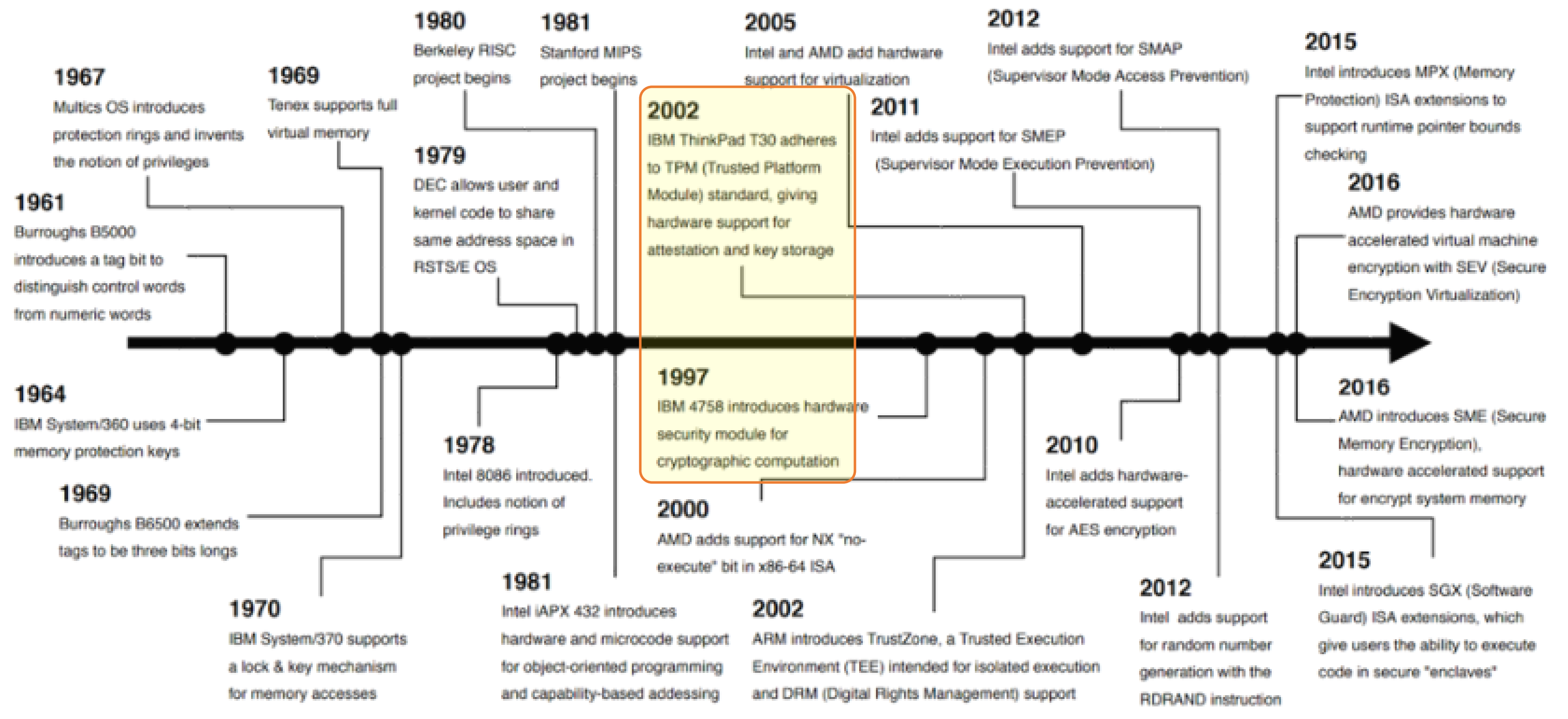
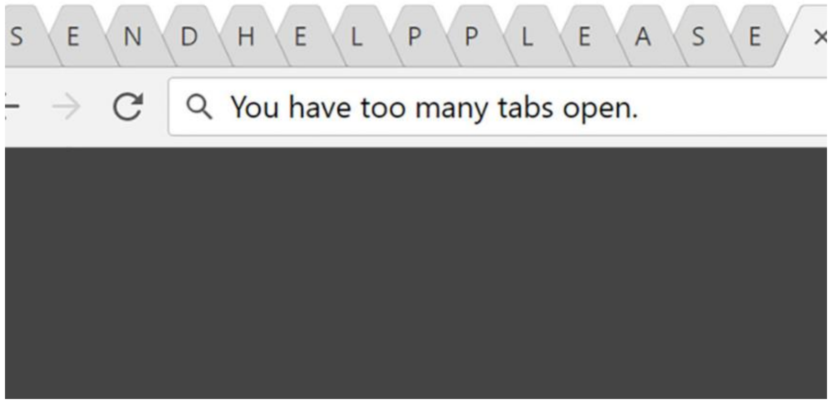
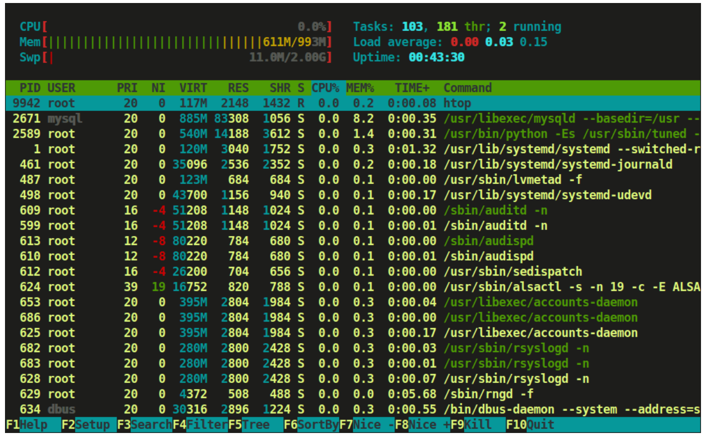


Figure 1: Sixty years of hardware support for security

Introduction to Security for Computer Architecture Students. Adam Hastings, Mohammed Tarek, Simha Sethumadhavan. https://www.cs.columbia.edu/~simha/ch1_supplement.pdf

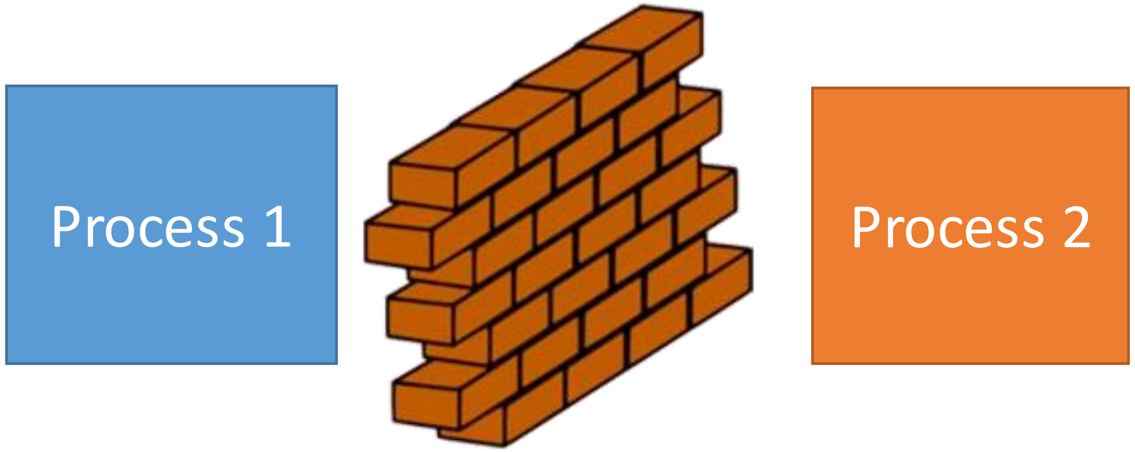
Security Context #1





- Problems:
  - Running random applications together with security-sensitive applications
  - Software can be buggy (or sometimes malicious)

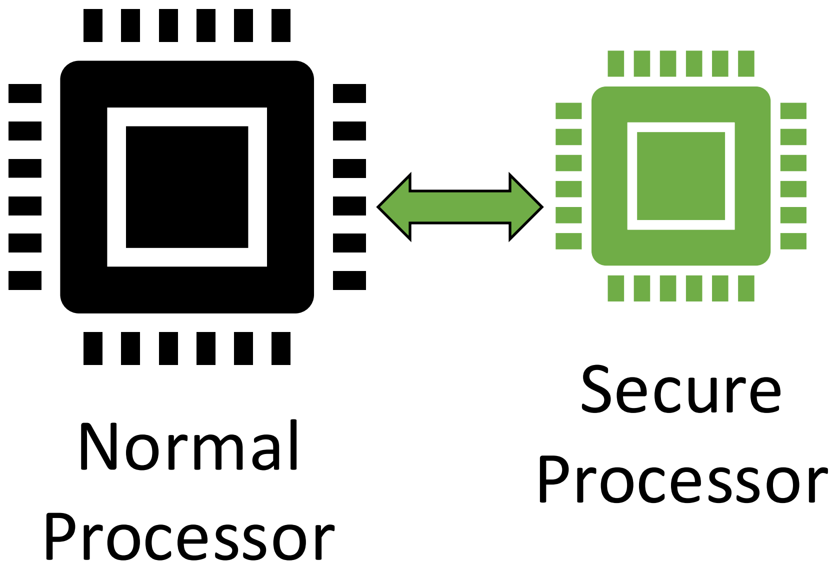
Isolation



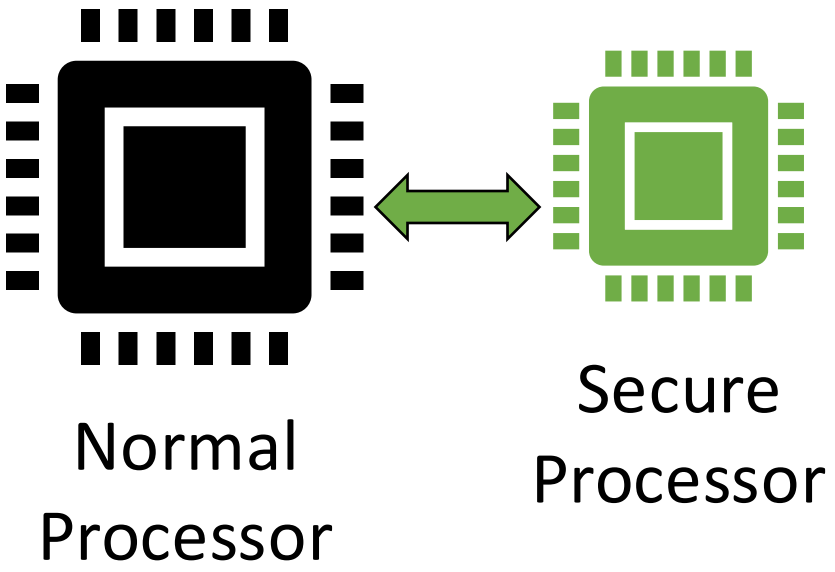
OS/Hypervisor enforced isolation

Can we do better than software-based isolation?

Physical Isolation



Secure Co-Processors

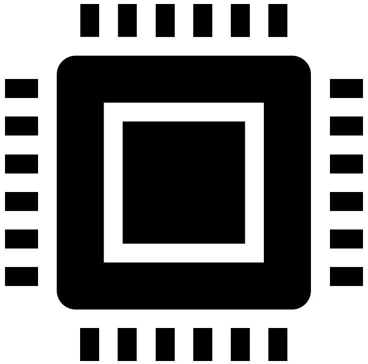


- Before IBM 4758 (1999):
  - Crypto accelerators (AES, RSA, etc.)
  - Store crypto keys inside the accelerator
  - Want to run more applications on the co-processor
- IBM 4758 (1999) -- 4765 (2012)
  - **Programmable** secure co-processor
  - Idea: create a virtual locker room

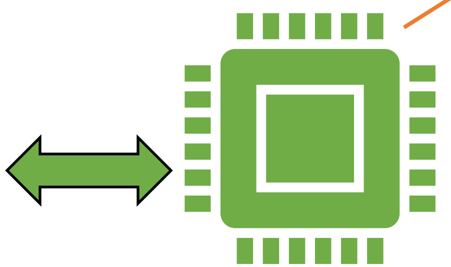
Secure Co-Processors

General-purpose processor, rather than ASIC, with isolated DRAM.

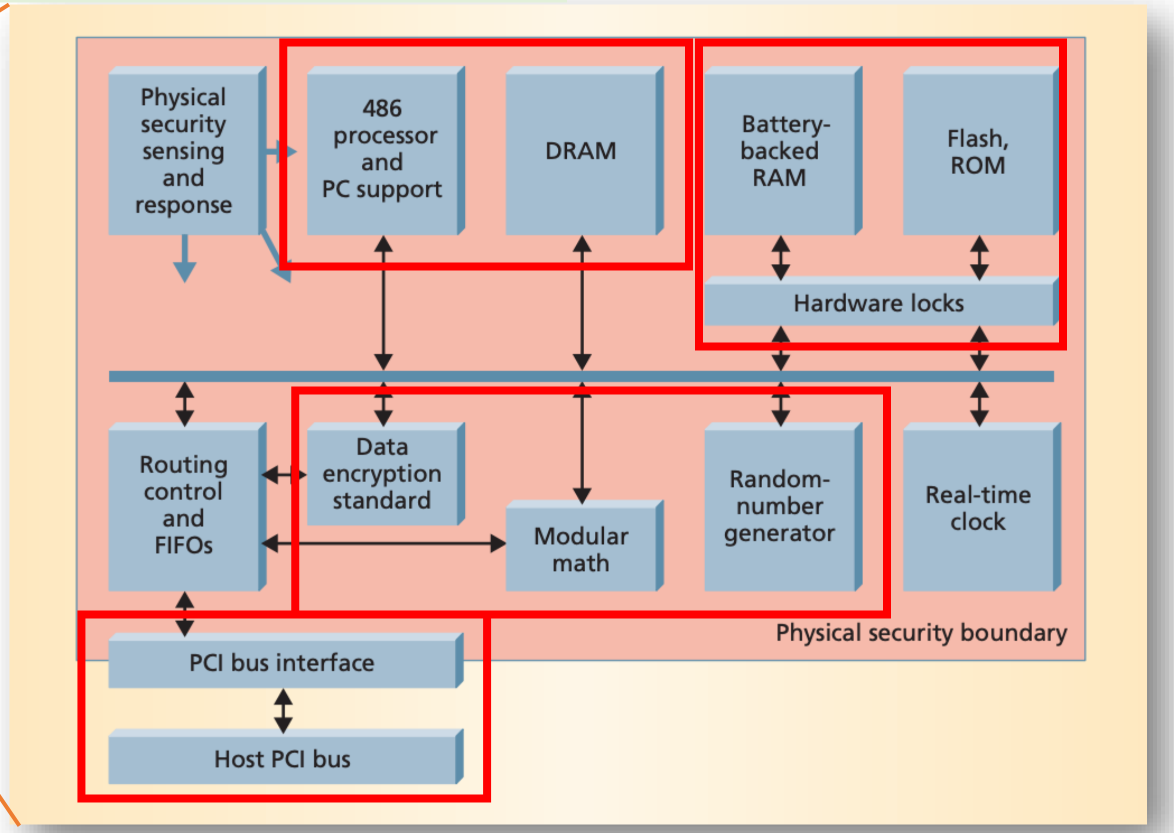
Hardware lock, resilient against physical attacks to modify firmware



Normal Processor

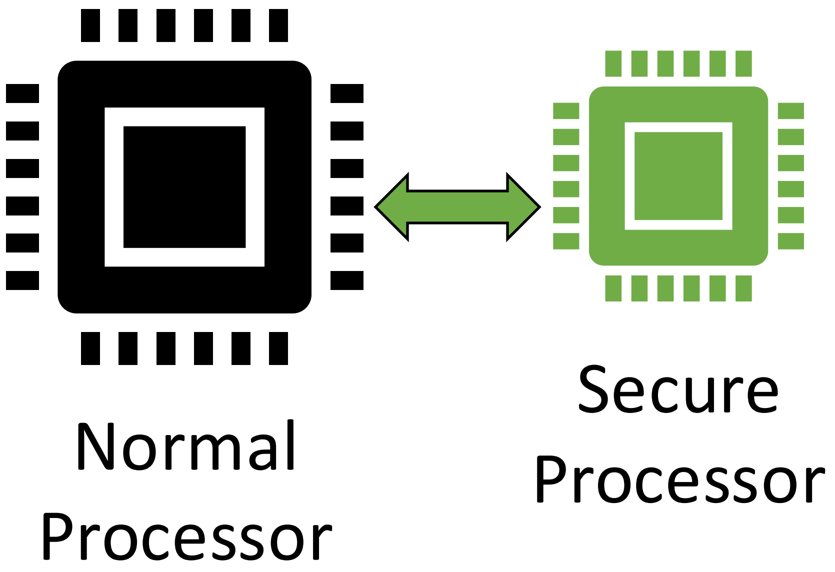


Secure Processor



Narrow interface, only interact with external worlds via APIs
(keys do not leave the co-processor)

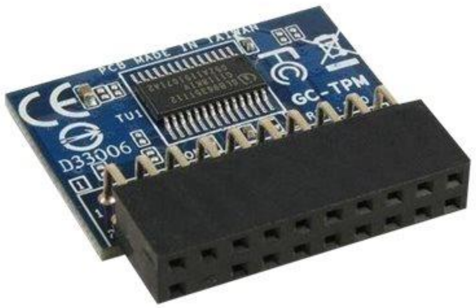
Secure Co-Processors

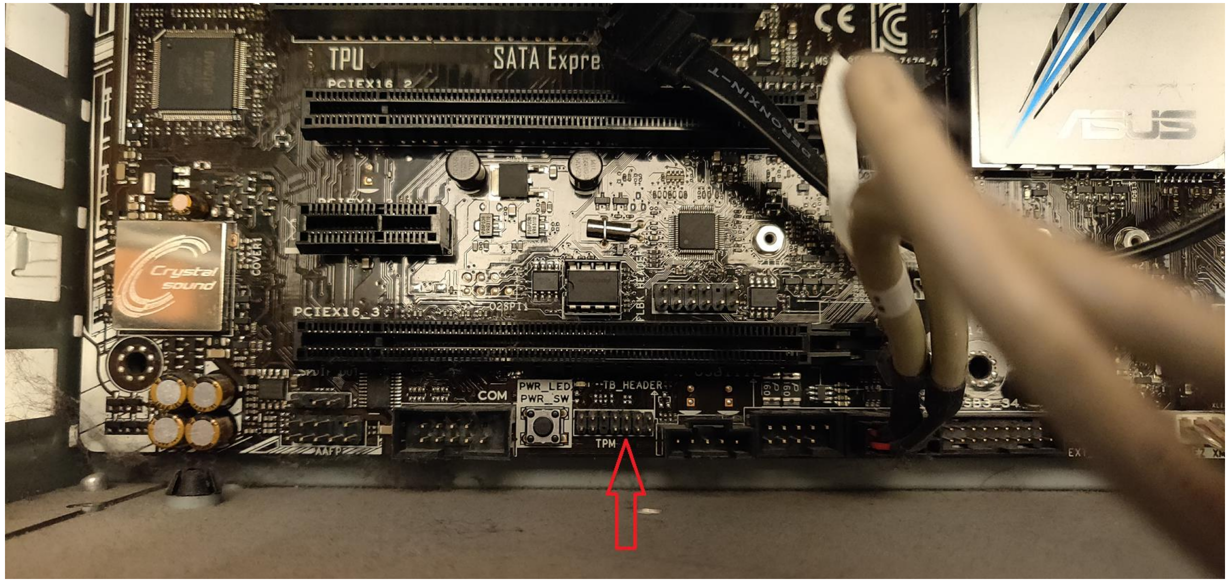


- Before IBM 4758 (1999):
  - Crypto accelerators (AES, RSA, etc.)
  - Store crypto keys inside the accelerator
  - Want to run more applications on the co-processor
- IBM 4758 (1999) -- 4765 (2012)
  - **Programmable** secure co-processor
  - Idea: create a virtual locker room
  - Problem?
  - The SWOFTWARE! Bad programmability.
  - Need to find a middle ground: run selected applications that offer strong security functionality

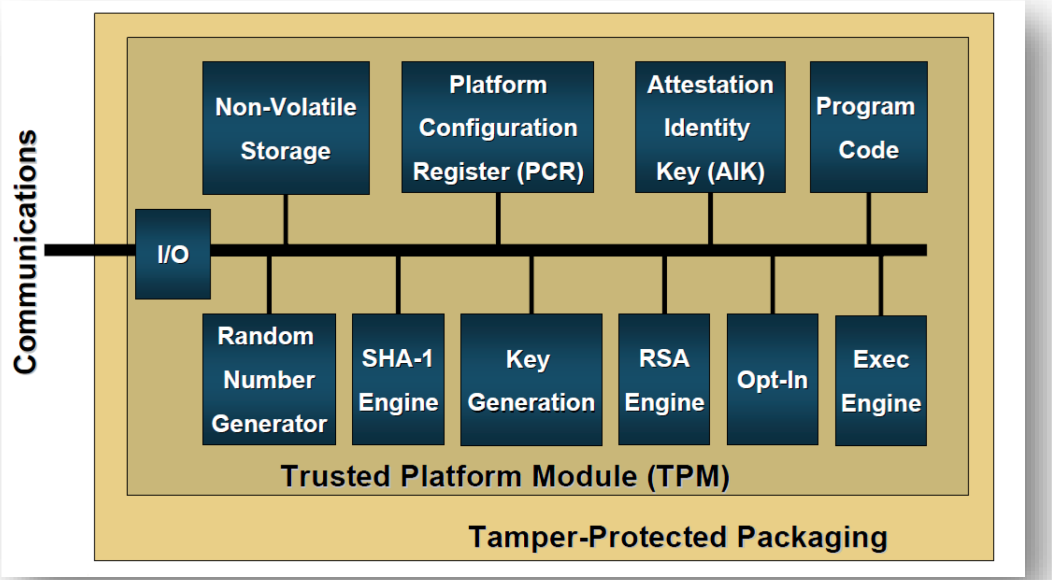
Trusted Platform Module (TPM)

- “*Commoditized* IBM 4758”: Standard LPC interface attaches to commodity motherboards



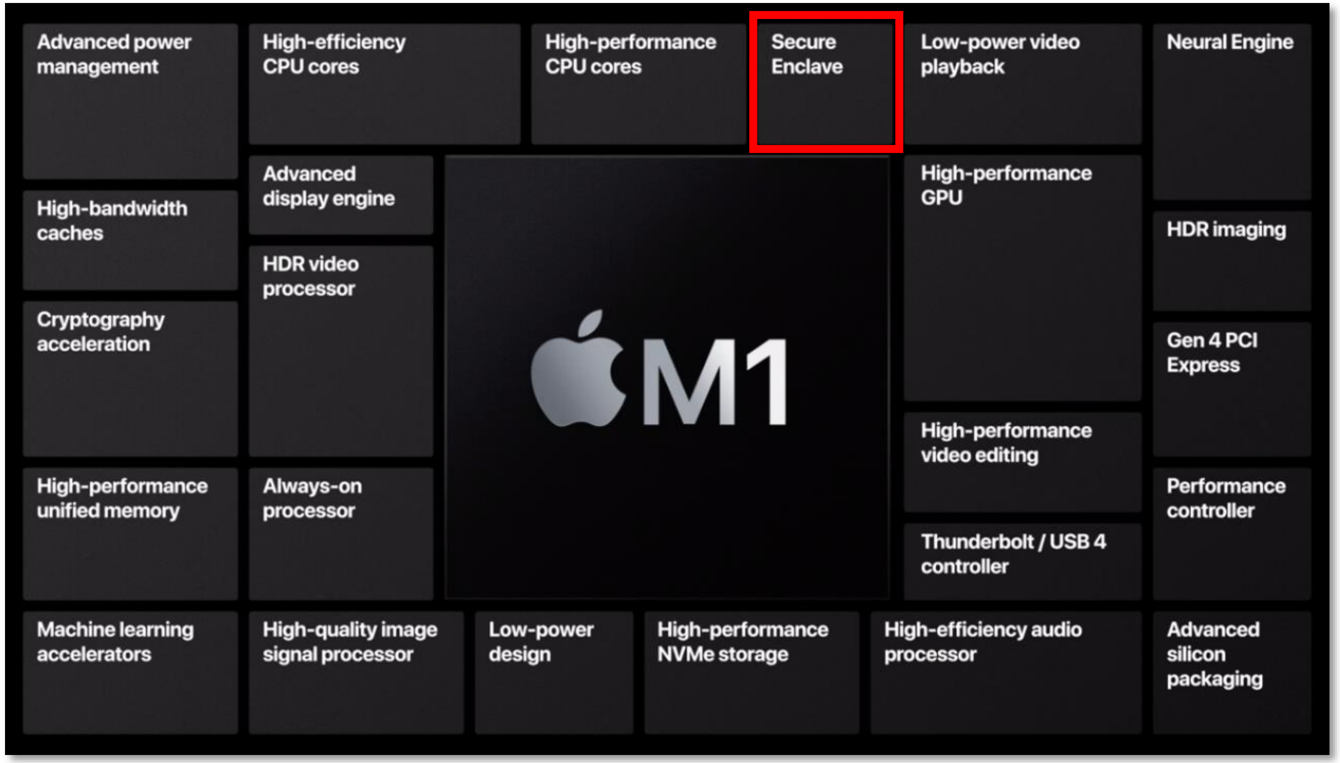


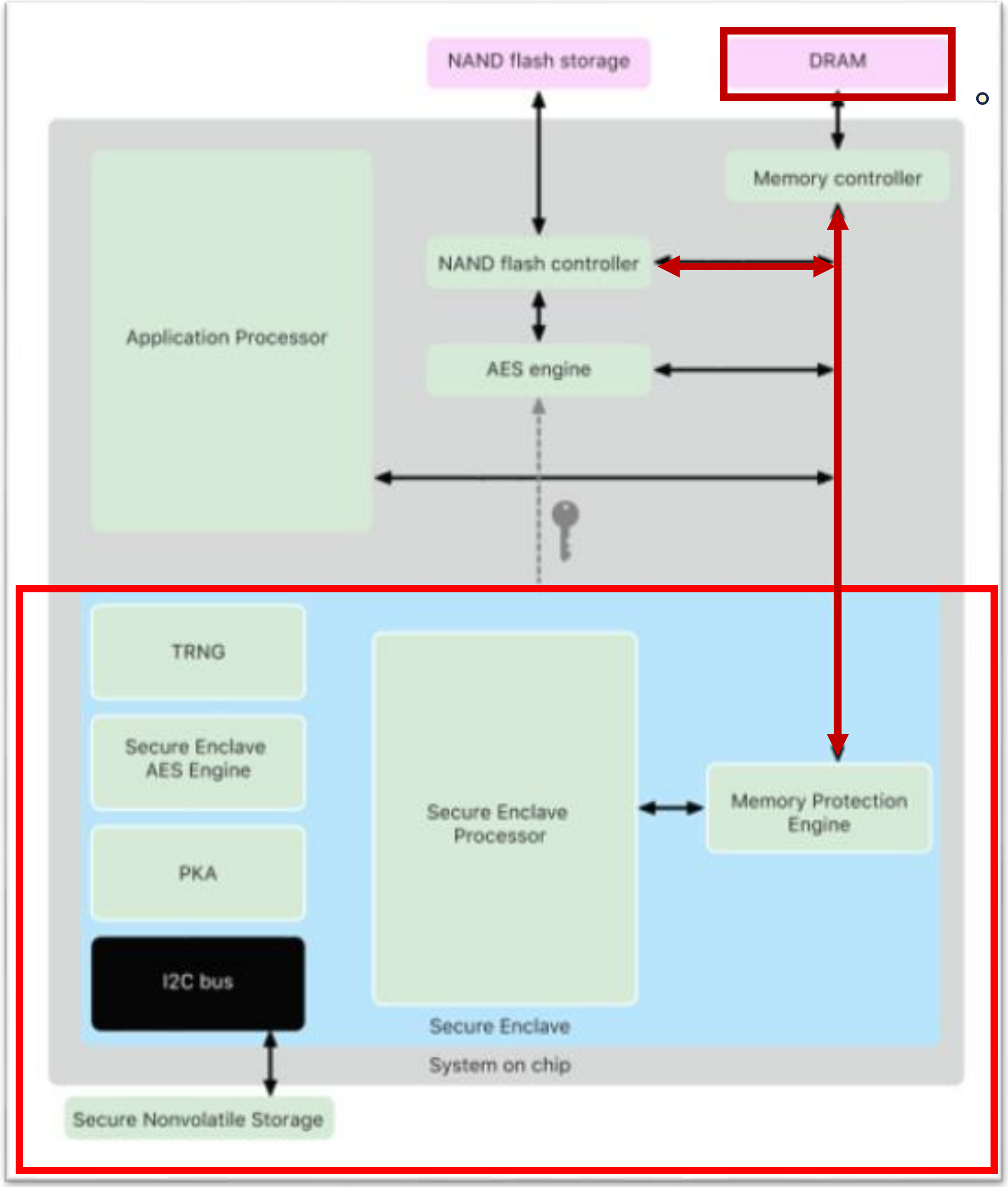
https://scotthelme.co.uk/upgrading-my-pc-with-a-tpm/

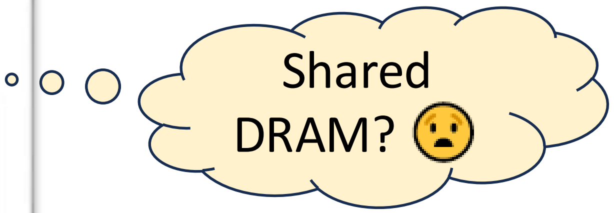


Apple Secure Enclave

- Advantage: one company controls both the hardware and the software
- Apple secure enclave runs a customized formally verified micro-kernel OS





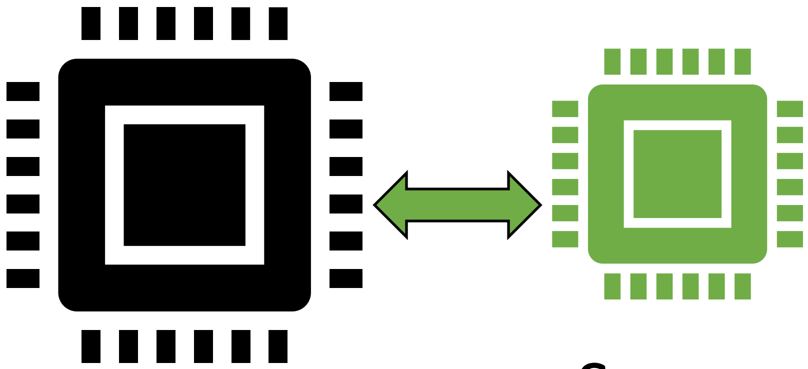


Encrypt enclave data and only decrypt at the memory protection engine

- Only run secure enclave functionality, no user code
- Block vulnerabilities due to software bugs (running L4 microkernel)
- Block uarch side channels

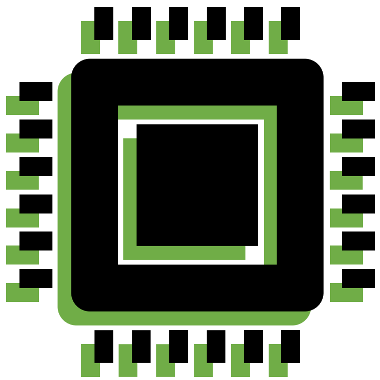
From Apple Platform Security (Page 5-96)

Make Physical Isolation More Flexible?



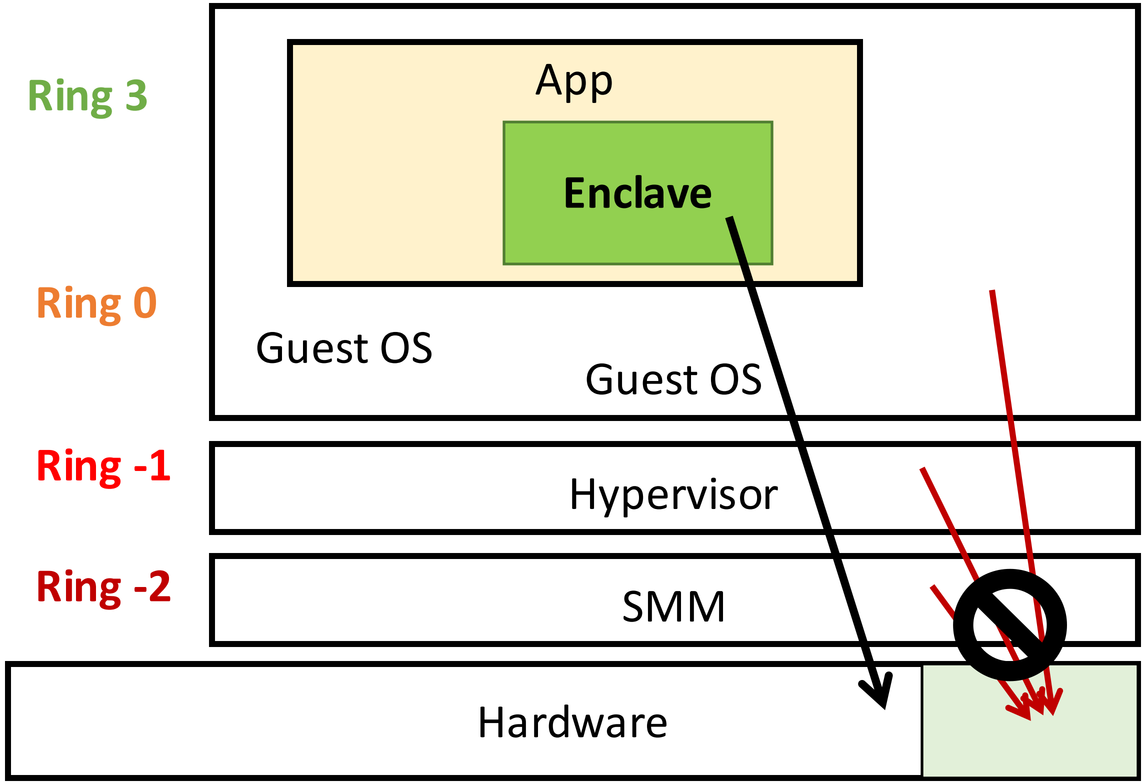
Normal Processor

Secure Processor

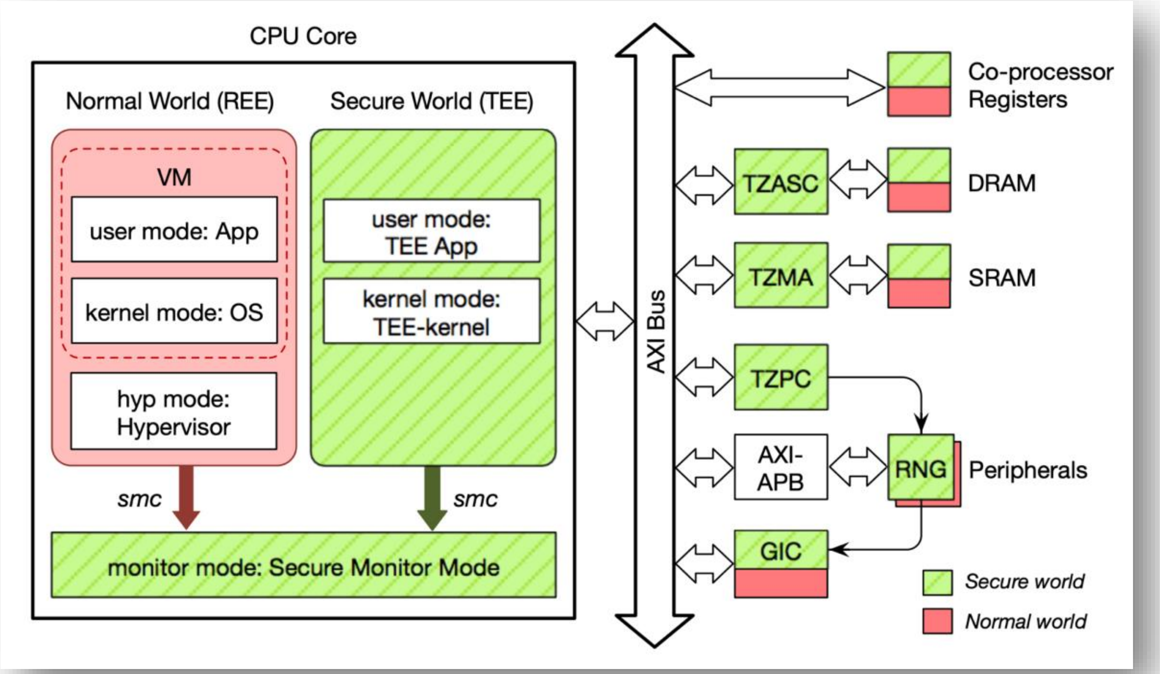


A Processor can switch between Normal and Secure Modes

The Trends (isolation with some sharing?)



Intel SGX model



ARM TrustZone

Security?

Usability?

Fixed Design (Static)

Flexible Design (Dynamic)

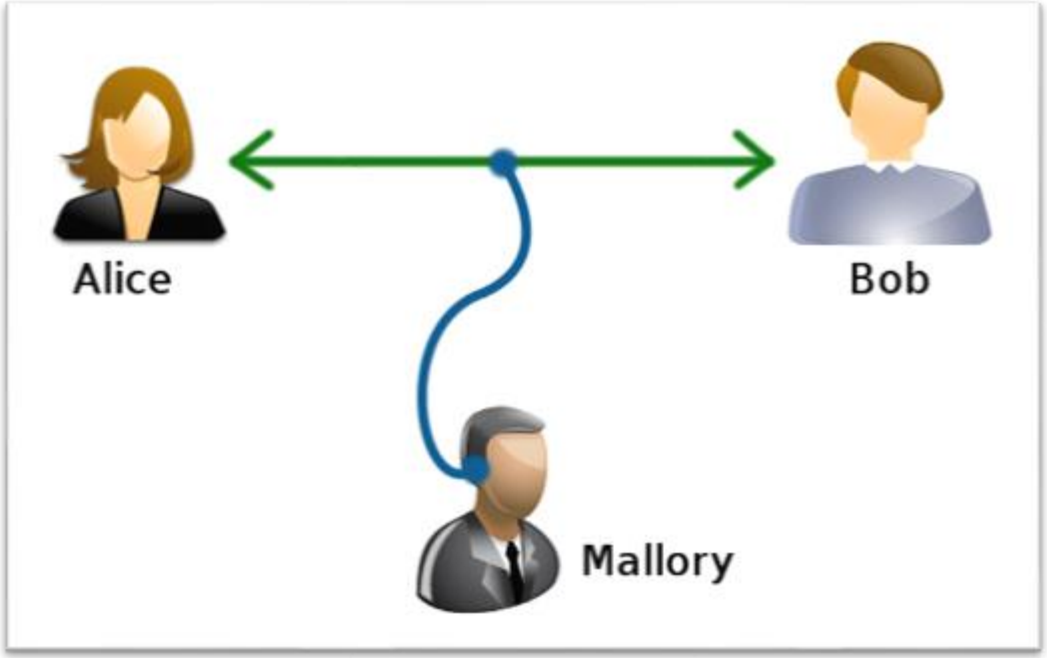
Security Context #2

Lost your device?

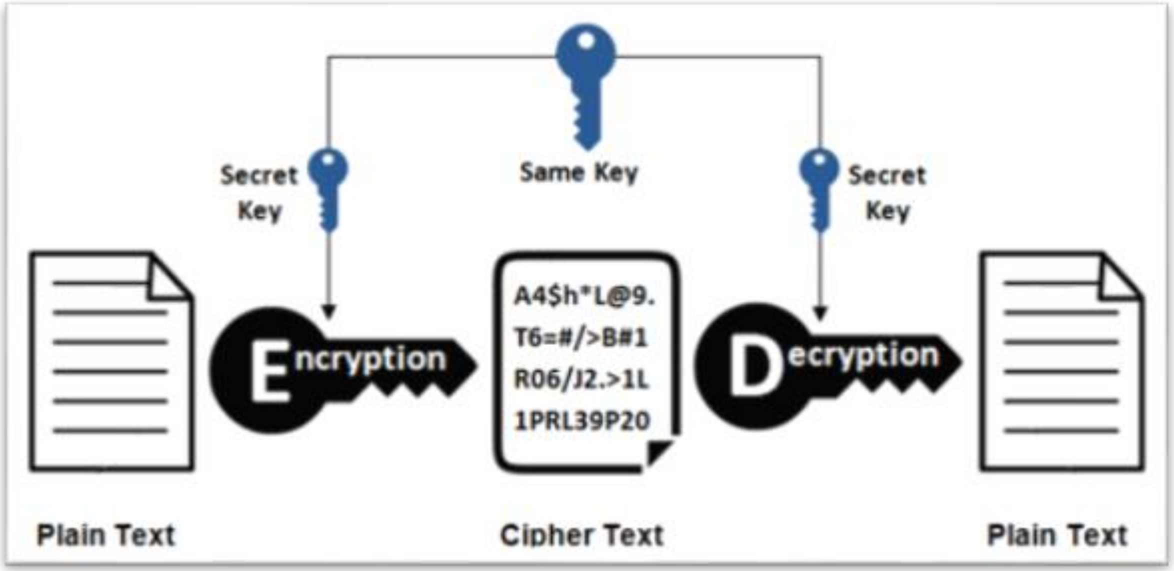
- Data leakage => confidentiality
- Rootkits => integrity

Security Property and Crypto Primitives

- Confidentiality
  - Symmetric
  - Asymmetric
- Integrity
- Freshness



Symmetric Cryptography



- One-time-pad (OTP)

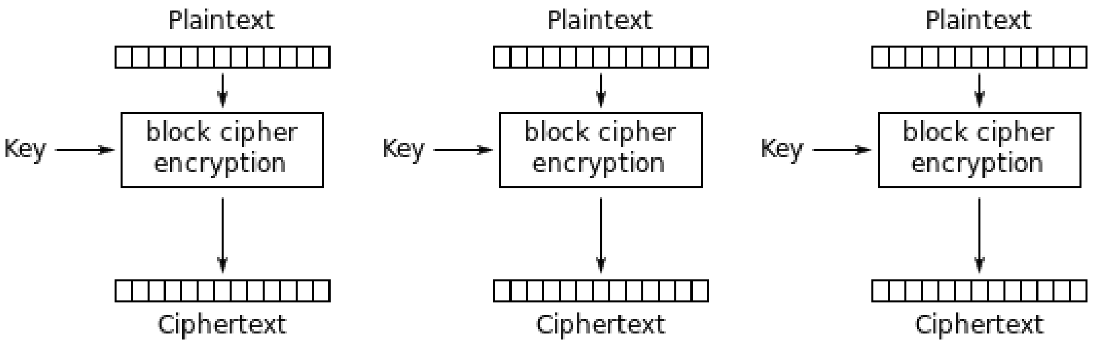
Encryption:
`ciphertext = key ⊕ plaintext`

Decryption:
`plaintext = key ⊕ ciphertext`

How about encrypting arbitrary length message? Any problems?

Block ciphers (e.g., DES, AES)

- Divide data in blocks and encrypt/decrypt each block
- Block ciphers are constructed using **one-way function** (see 6.1600)

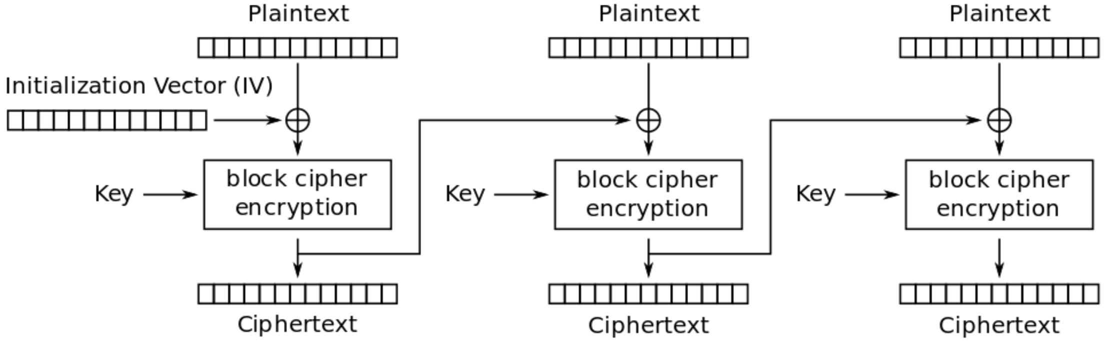


Electronic Codebook (ECB) mode encryption

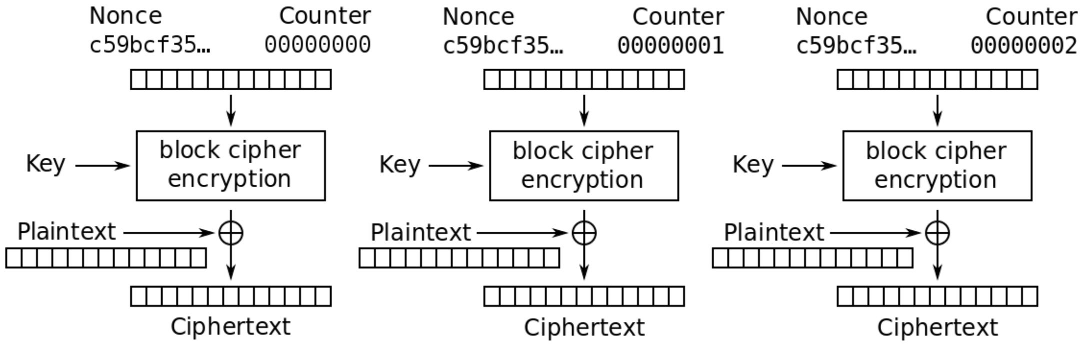
ECB IS NOT RECOMMENDED

Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

Other Block cipher Modes



Cipher Block Chaining (CBC) mode encryption



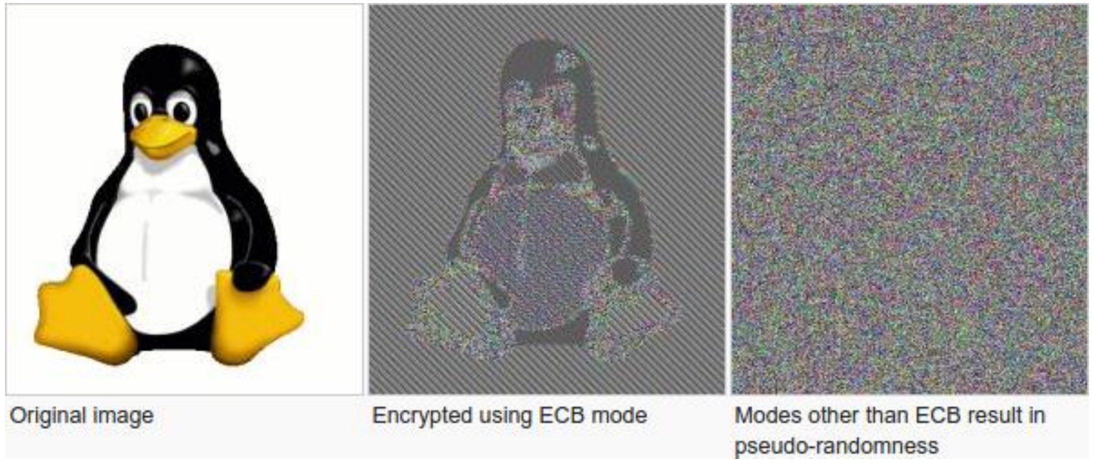
Counter (CTR) mode encryption

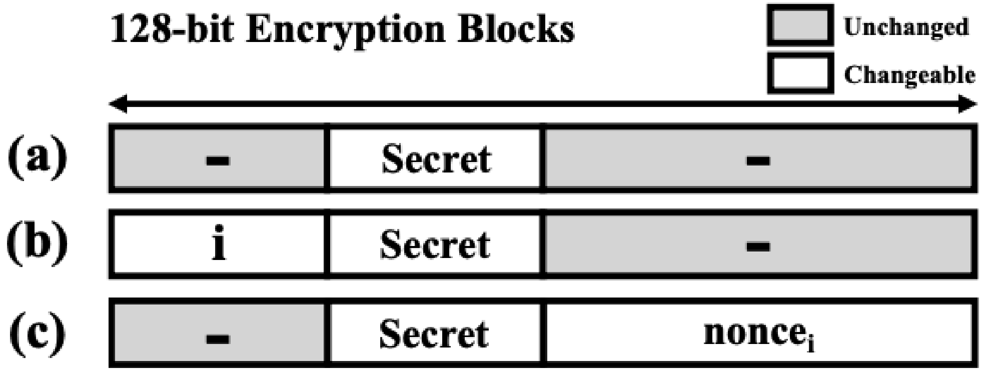
`IV` can be public, but need to ensure to not reuse `IV` for the same key.

Use cases: file/disk encryption and memory encryption.

Use Correct Crypto Primitives

- Ciphertext Side Channels on AMD SEV
- SEV's memory encryption engine uses an XOR-Encrypt-XOR (XEX) mode -> deterministic encryption during the lifetime of a VM



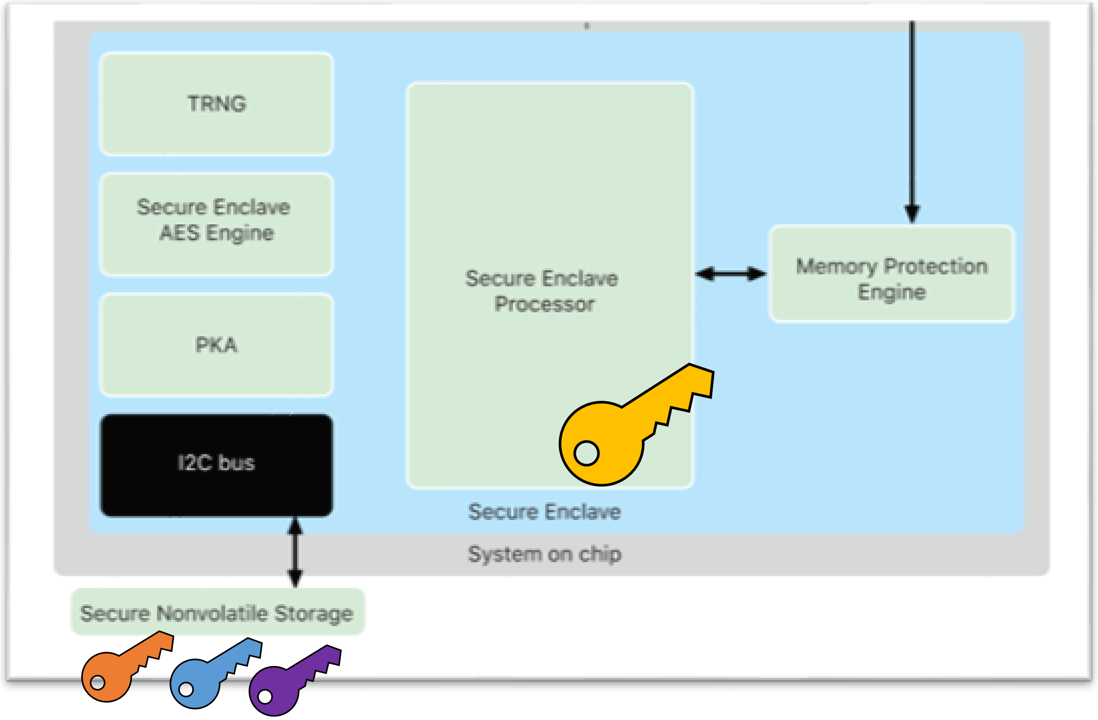


Li et al, CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel, USENIX'21
Li et al, A Systematic Look at Ciphertext Side Channels on AMD SEV-SNP, S&P'22

Encrypt using Short Passcode

- How many attempts do we need to brute-force 6-digit passcode?
- How to mitigate brute-force?
- How to deal with attacks who can copy the data across devices and brute-force in parallel?

Bind Crypto Keys to Device



User data encryption **keys**

A unique ID (**UID**) root cryptographic key.

- Unique to each device
- Randomly generated
- Fused into the SoC at manufacturing time
- Not visible outside the device

`Passcode + UID -> passcode entropy`

Brute-force has to be performed on the device under attack

Combine with other mitigations:

- Escalating time delays
- Erase data when exceeding attempt count

Integrity (MAC/Signature)

`Hash(m) = h`

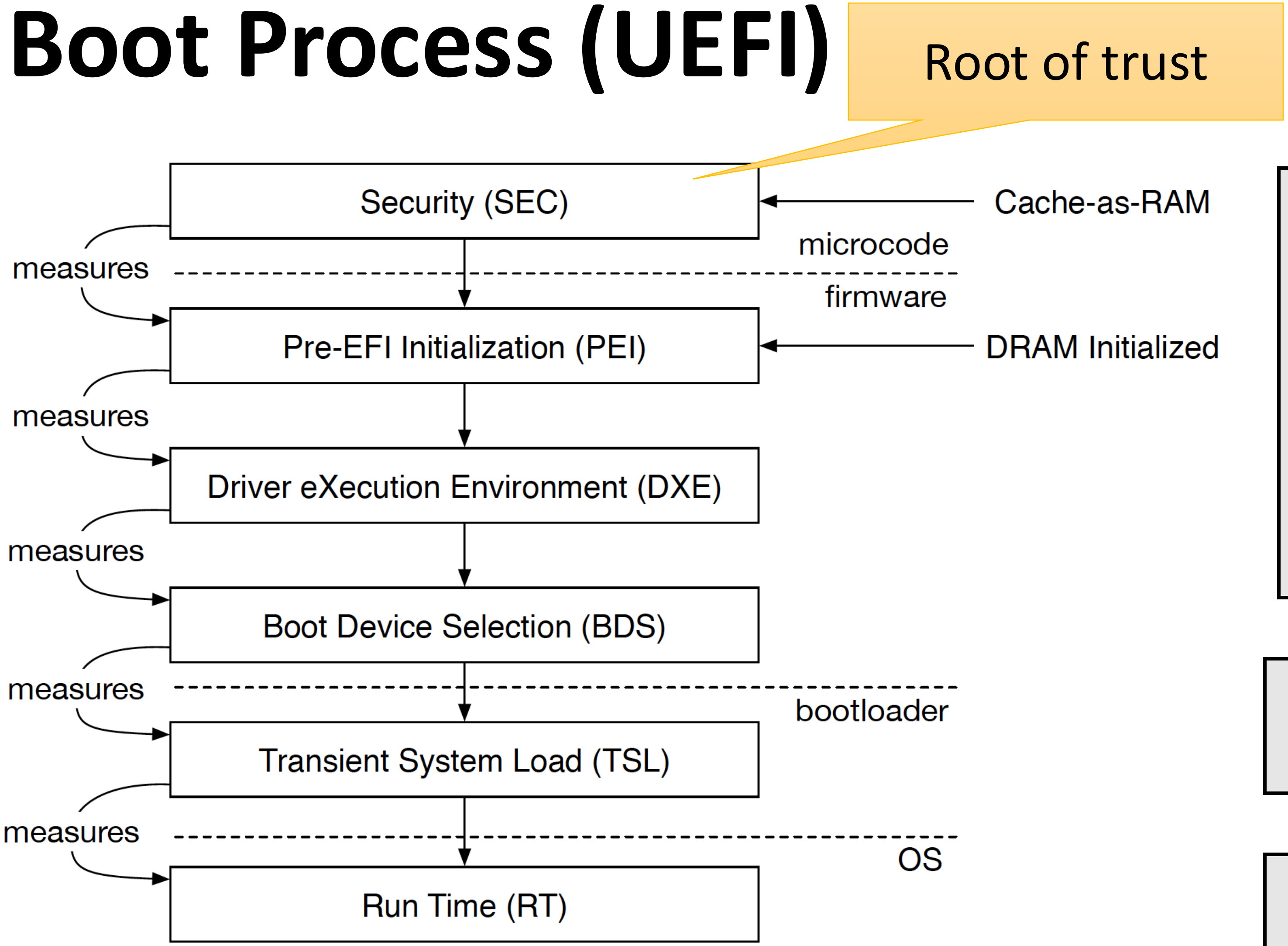
Message (long)

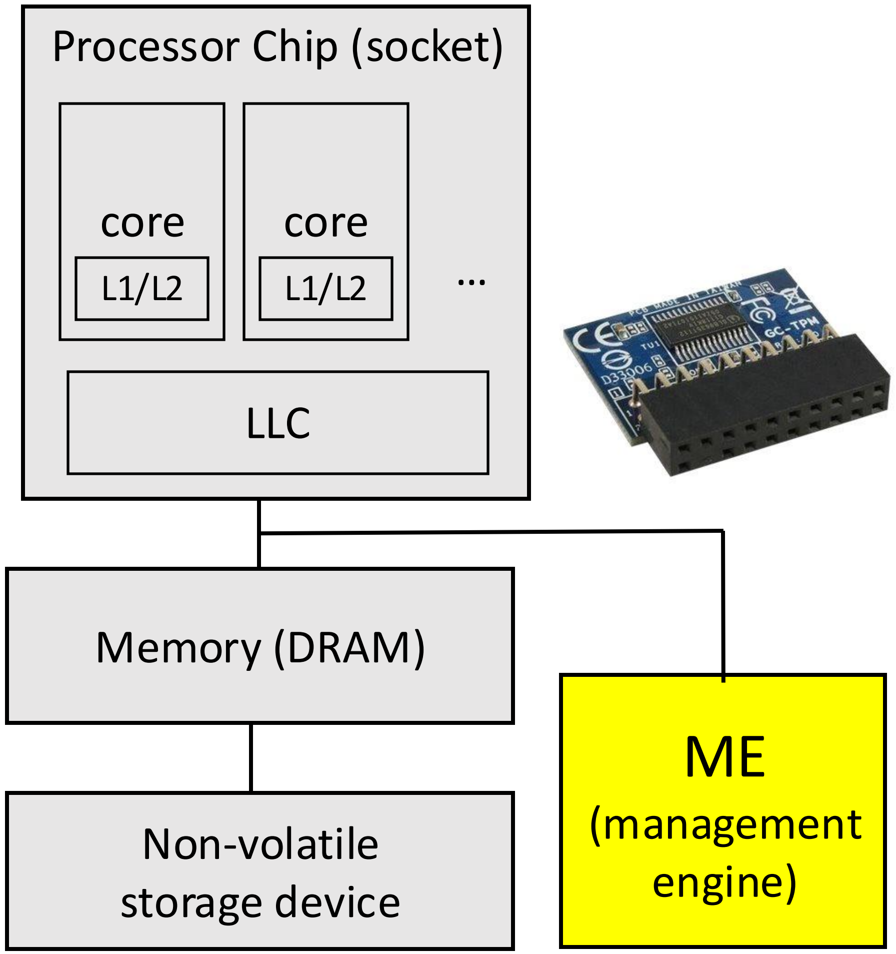
Hash value (length depends on algorithm)

Use as fingerprints

- One-way hash
  - Practically infeasible to invert, and difficult to find collision
- Avalanche effect
  - "Bob Smith got an A+ in ELE386 in Spring 2005"→ `01eace851b72386c46d`
  - "Bob Smith got an B+ in ELE386 in Spring 2005"→ `936f8991c111f2cefaw`
- When message is long
  - Divide message into blocks, and keep extending the hash by adding previous hash

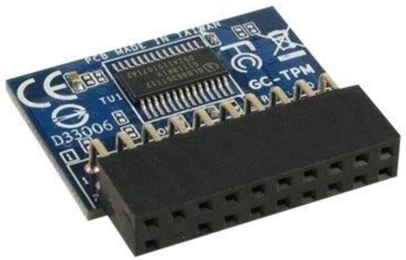
Boot Process (UEFI)

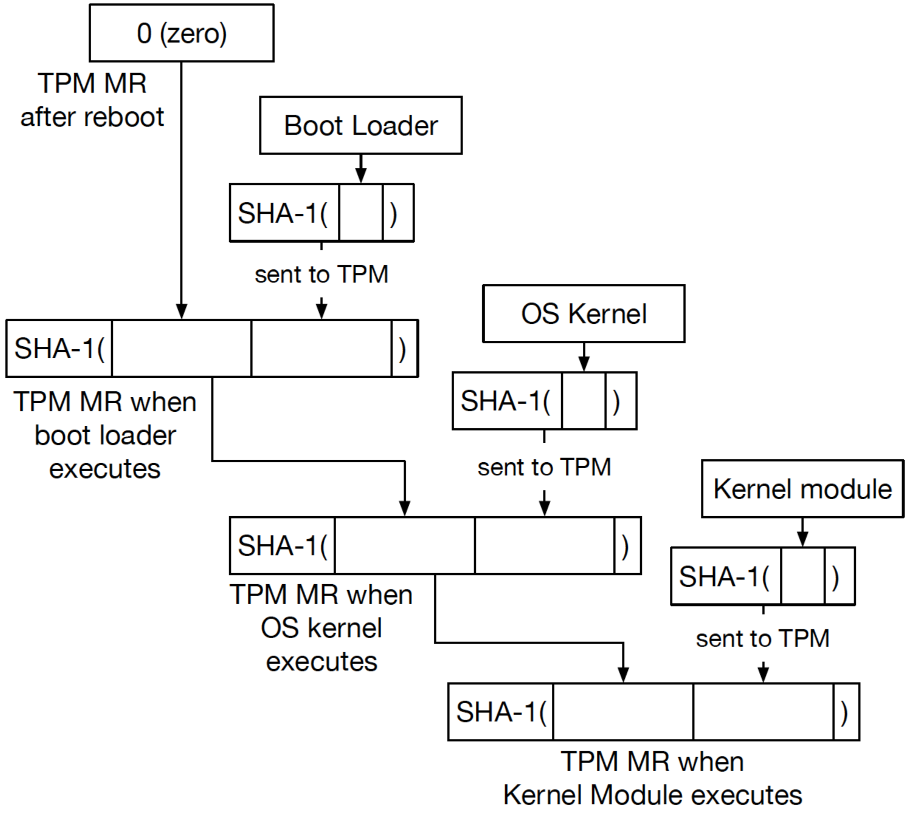


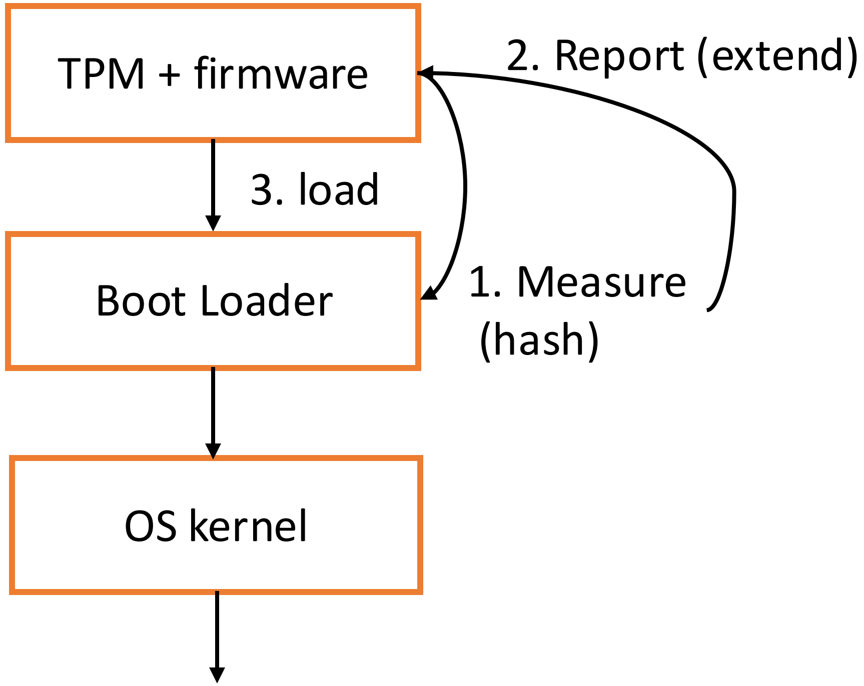


Always measure before executing …

Secure Boot using TPM





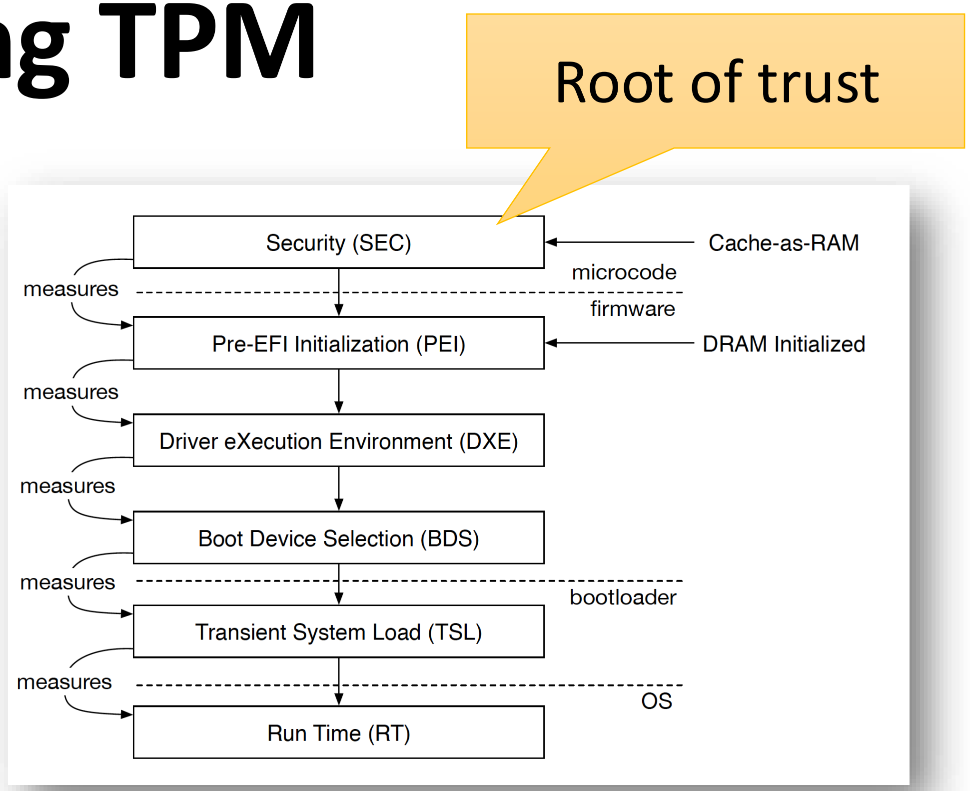


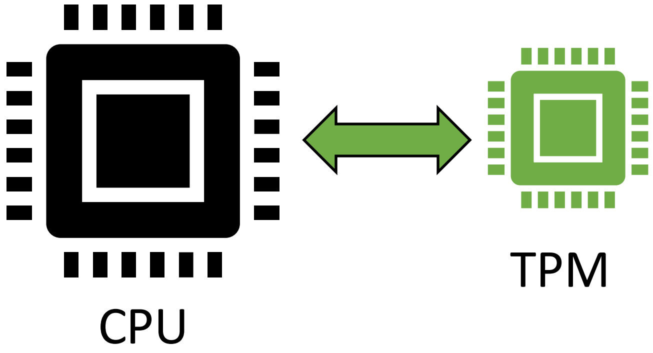
Each step, TPM compares to expected values locally or submitted to a remote attestor.

PCR: platform configuration register

Security Problems of Using TPM

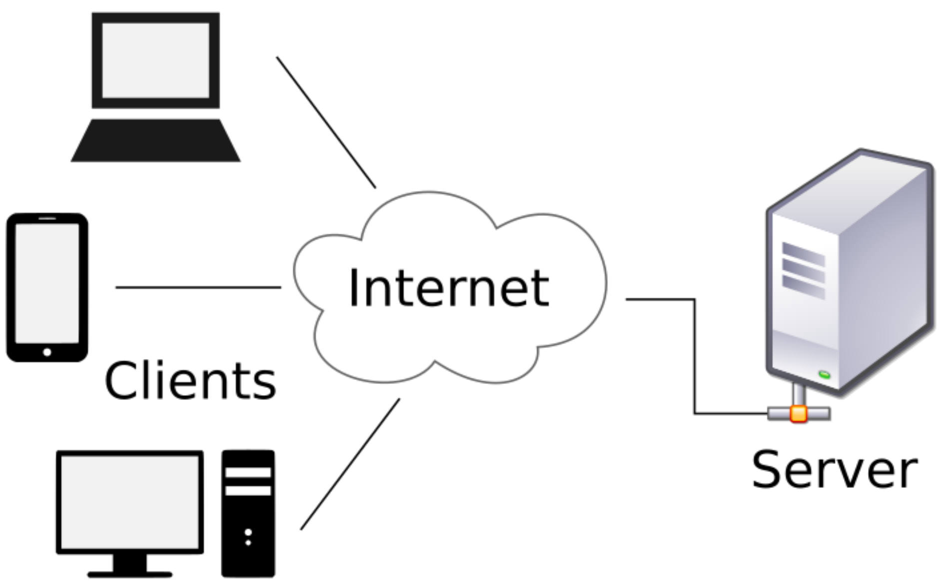
- Assume the first-stage bootloader is securely embedded in motherboard
- Not easy to use with frequent software/kernel update
- Time to check, time to use
- TPM Reset attacks
    - exploiting software vulnerabilities and using software to report false hash values





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 Context #3



a) A remote server wants to trust an end-user, e.g., when joining a company's highly-secure network.

b) A device wants to update/install an new version of OS/software approved by the vendor

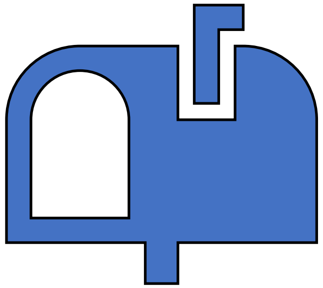
-> **Authentication and establishing trust**

Asymmetric Cryptography (e.g., RSA)

- A pair of keys:
  - Private key ($K_{private}$ – kept as secret)
  - Public key (K_{public} – safe to release publicly)

- Computation:
  - `Sign(plaintext, `$K_{private}$`) = signature`
  - `Verify(plaintext, signature, `K_{public}`) = T/F`

- How to announce and obtain the public key?

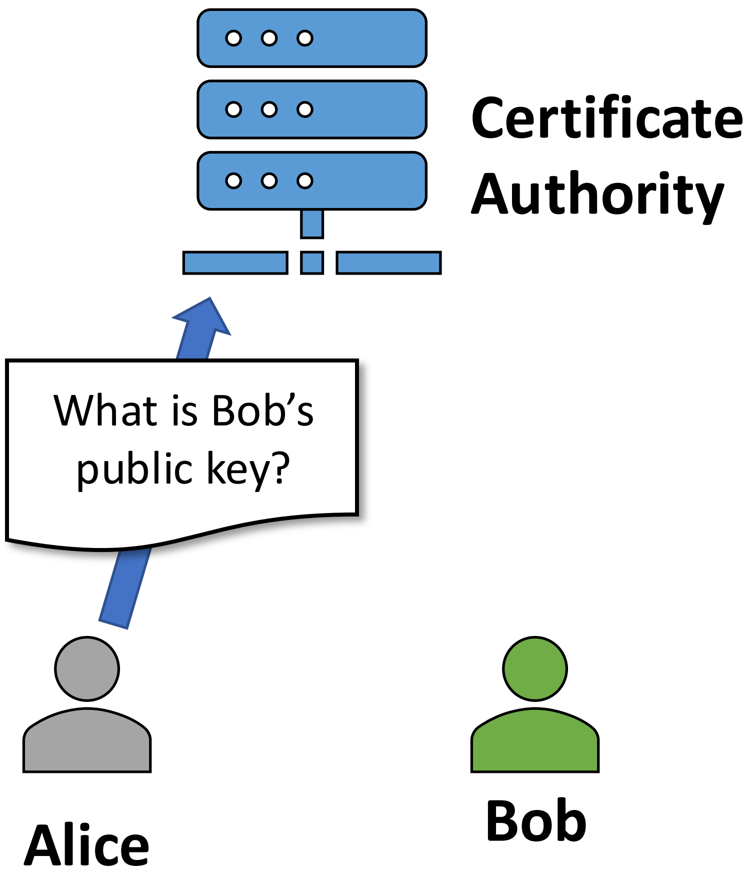


Mail box is public;
Box key is private

Public Key Infrastructures (PKIs)

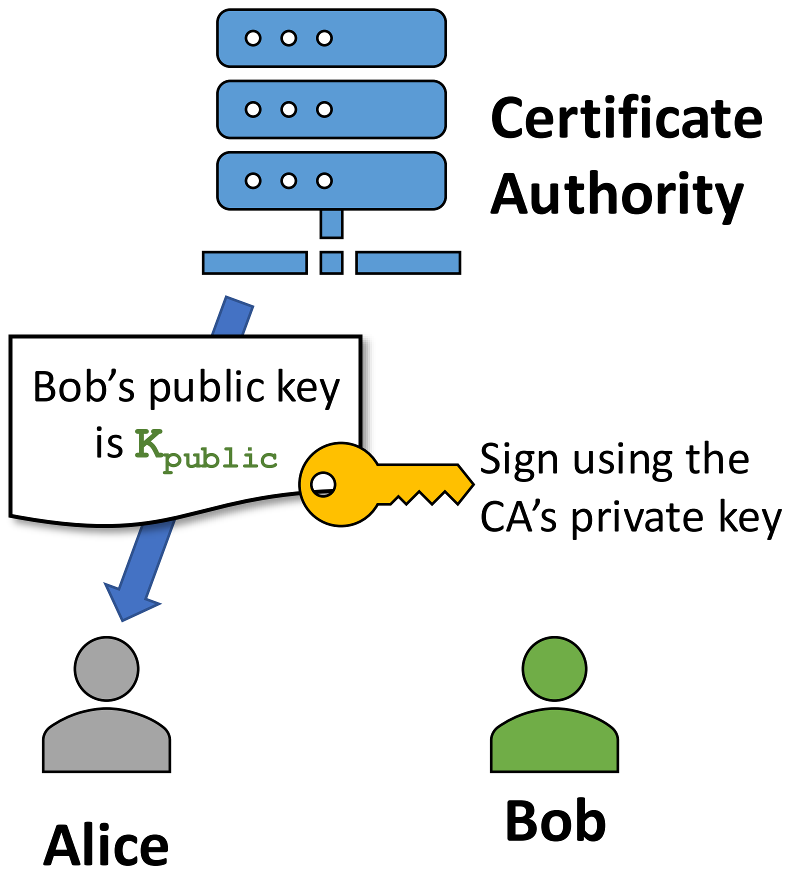
- Analogy: public key is like a government-issued ID, need to be validated by an authority.

- Bob has a private key $\mathbf{K}_{\mathtt{private}}$ and wants to claim he corresponds to a public key $\mathbf{K}_{\mathtt{public}}$

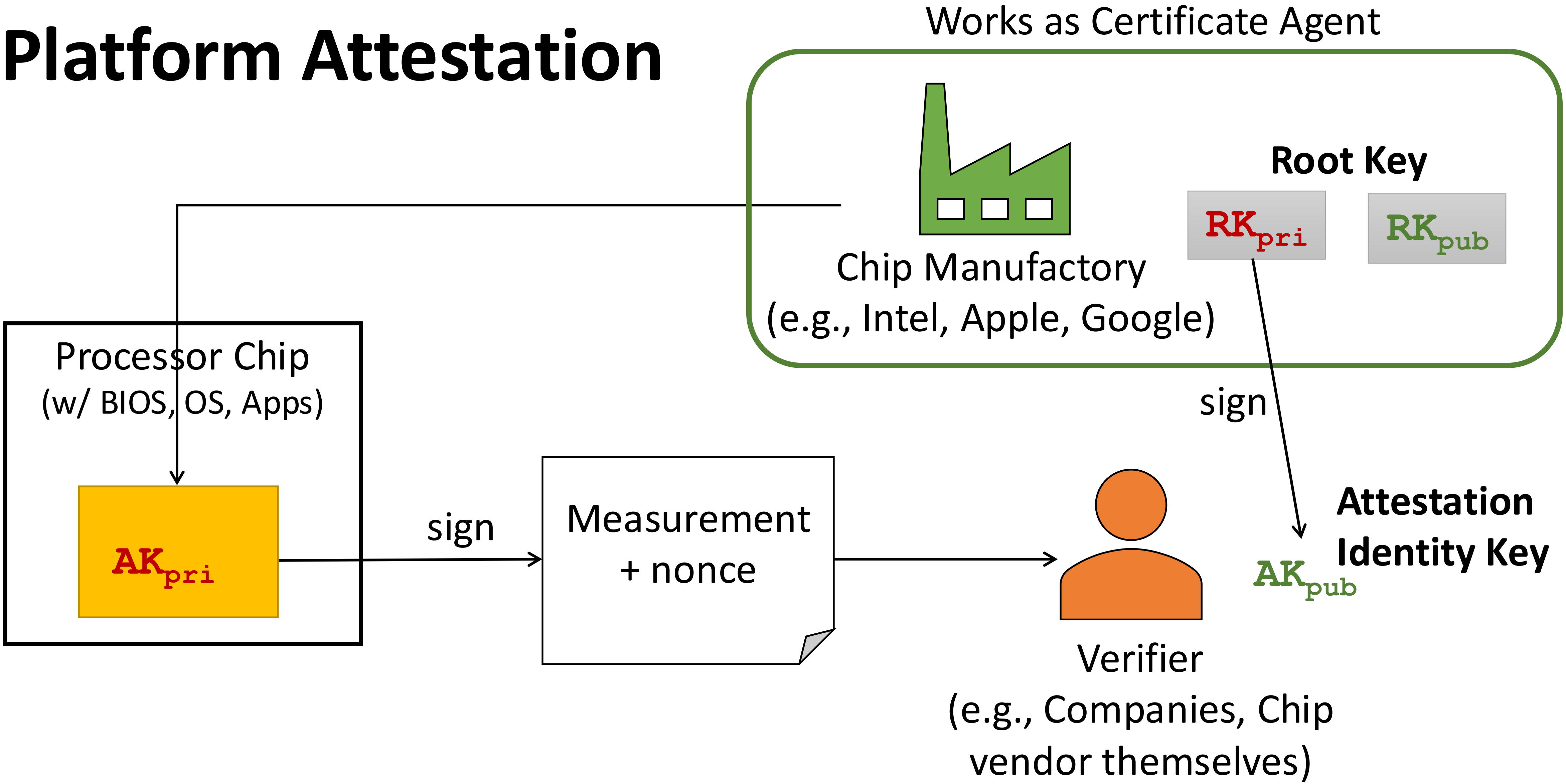


Public Key Infrastructures (PKIs)

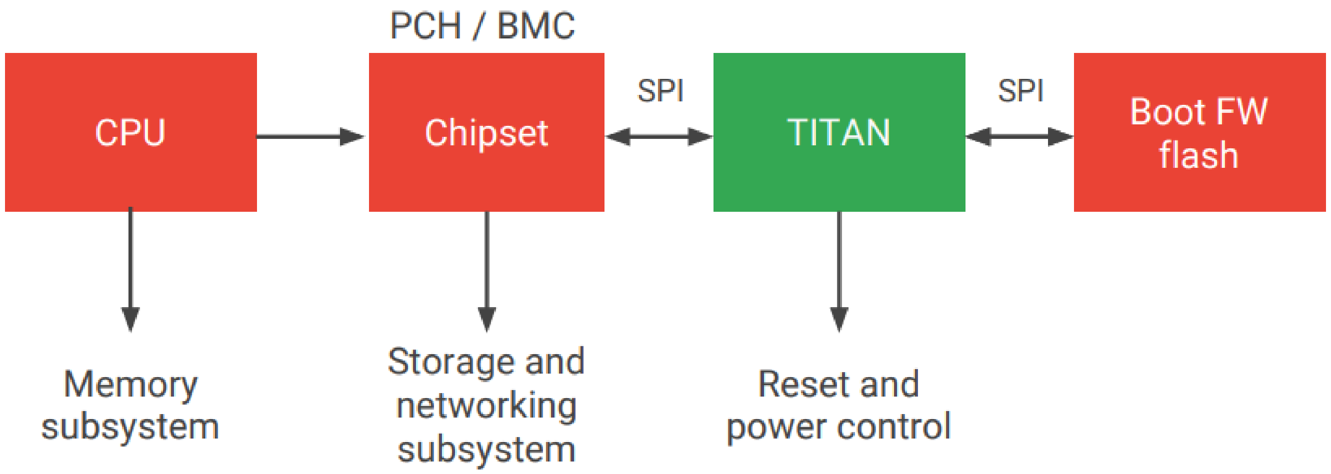
- Analogy: public key is like a government-issued ID, need to be validated by an authority.
- Bob has a private key $K_{private}$ and wants to claim he corresponds to a public key K_{public}
- Establish a chain of trust
- **Real-world use cases**: identify website, identify hardware chips/processors

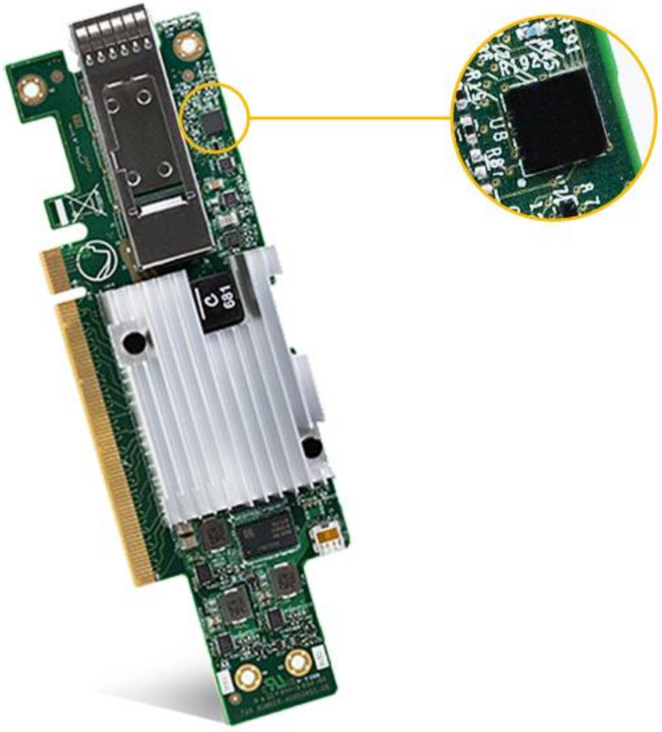


Platform Attestation



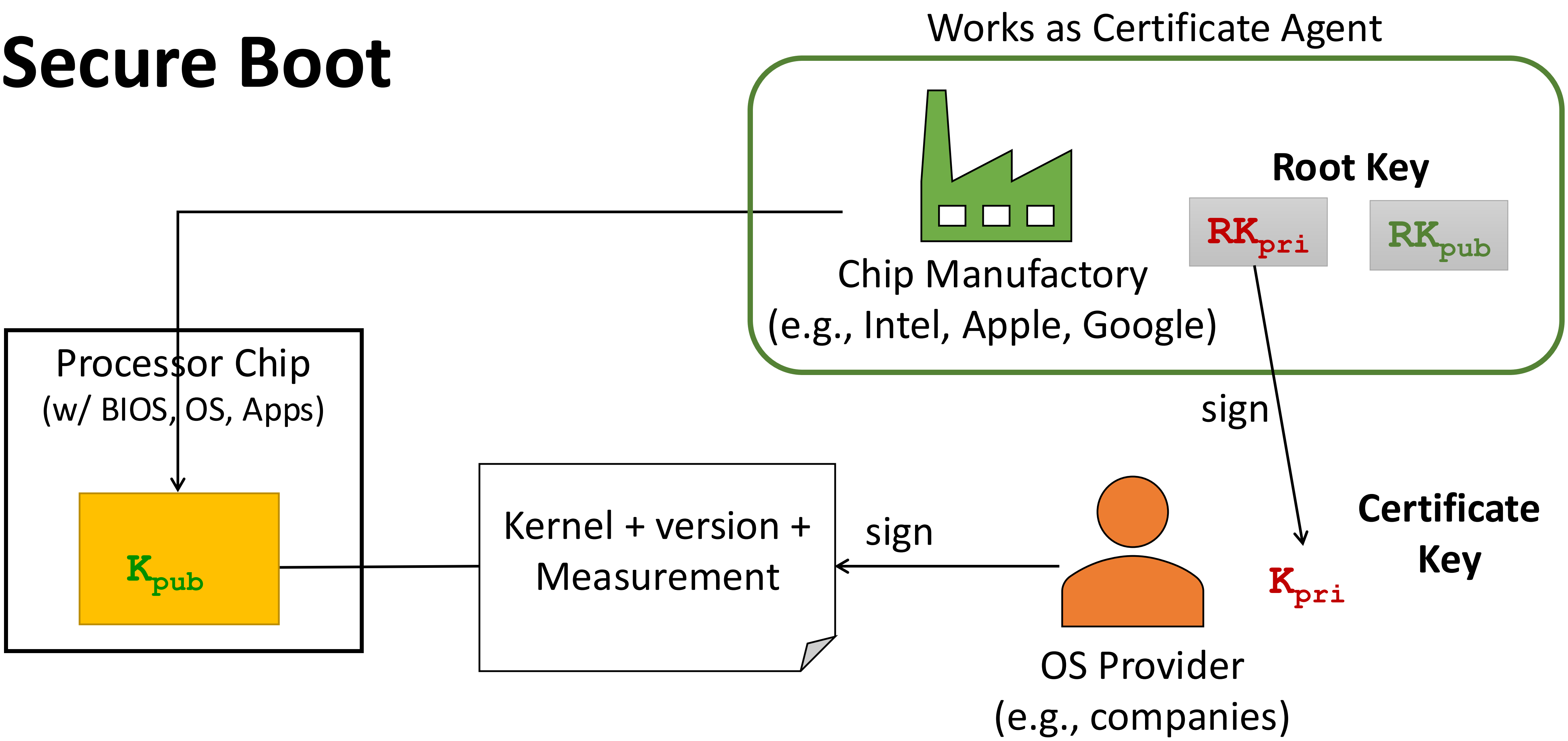
OpenTitan





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

Secure Boot

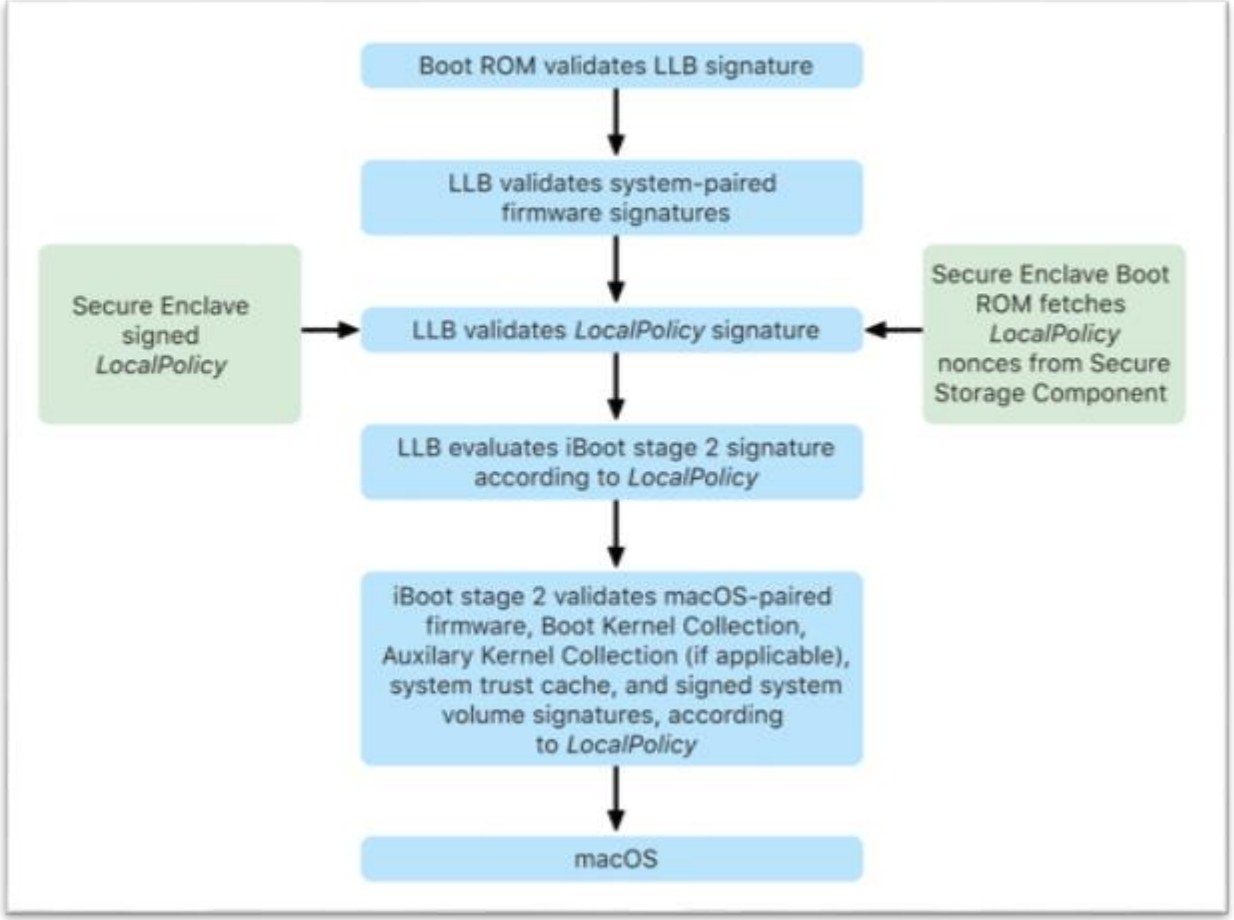


Works as Certificate Agent
Chip Manufactory
(e.g., Intel, Apple, Google)
Root Key
RK_{pri}
RK_{pub}
sign
Processor Chip
(w/ BIOS, OS, Apps)
K_{pub}
Kernel + version +
Measurement
sign
OS Provider
(e.g., companies)
K_{pri}
Certificate
Key

Secure Boot

Similar to TPM but with more constraints

- Each step is signed by Apple to prevent loading non-Apple systems
- Verify more components, including operating system, kernel extensions, etc.
- Keep track of version number to prevent rolling back to older/vulnerable versions



Summary

What Can Hardware Security Modules Offer?

- Physical isolation
- Bind data and applications with the hardware device
- Establish root of trust
- More efficient

Challenges: software support. Programmability.

Next:
IoT & Embedded Security

(Also with fancy demos 🔨🎩🤩✨)