Hardware-based Security Part I: Introduction & Basic concepts

Nicolò MAUNERO

Paolo PRINETTO

CINI Cybersecurity National Laboratory

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Acknowledgments

- The presentation includes material from
  - Gianluca ROASCIO – Politecnico di Torino
- His valuable contributions are here acknowledged and highly appreciated

Goal

- Presenting an overview of the most significant solutions aimed at resorting to hardware devices to protect the system from attacks that exploit vulnerabilities of *other* components of the system itself

Organization

- For sake of usability, the lecture is split into two parts:
  - *Part I : Introduction & Basic concepts*
  - *Part II : Implementations*

Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

- Security-oriented components
- Proprietary Solutions
- Open Security Platforms
- Built-in Security Features

Part I : Introduction & Basic concepts

Prerequisites

- Lecture:
  - *HS_1.1 - The role of Hardware in Security*

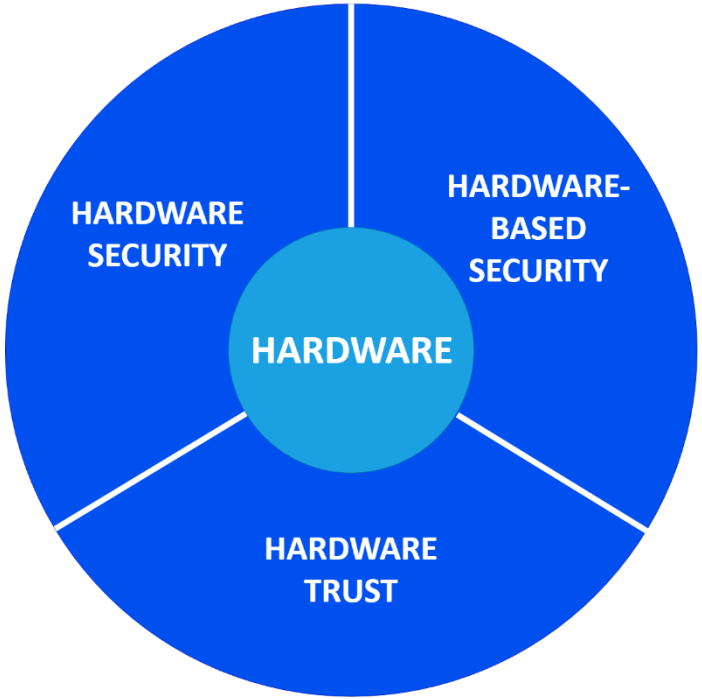
Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

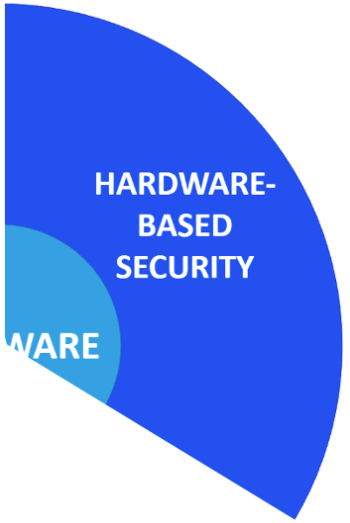
Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

The Role of Hardware in Cybersecurity



Hardware-based Security



- Refers to all those solutions aimed at resorting to hardware devices to protect the system from attacks that exploit vulnerabilities of *other* components of the system itself.

Hardware-based Security Role

- *”Although hardware-based security is not a silver bullet, it does provide a “chain of trust” rooted in silicon that makes the device and extended network more trustworthy and secure.”*

[https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/intel-security-essentials-solution-brief.pdf]

Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

Trust

- “A trusted component, operation, or process is one whose behavior is predictable under almost any operating condition and which is highly resistant to subversion by application software, virus, and a given level or physical interference.”

[ISO/IEC]

Root of Trust

- Component that needs to always behave in the expected manner because its misbehaviour cannot be detected

Root of Trust

- Trust in the *Roots of Trust* can be achieved through a variety of means including technical evaluation by competent experts

Root of Trust - Role

- Is used as basic block for the construction of a *Chain of Trust*

Root of Trust – Kinds & Applications

- A detailed analysis of Root of Trust application is presented later, when dealing with TPM (form slide #26 on)

Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

Trust Anchor

- ”A public or symmetric key that is trusted because it is directly built into hardware or software, or securely provisioned via out-of-band means, rather than because it is vouched for by another trusted entity (e.g., in a public key certificate).”

[https://csrc.nist.gov/glossary/term/trust-anchor]

Hardware Trust Anchor

- A Hardware Trust Anchor is a component that securely store and provide a unique secure identifier for the device

Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

System level solutions

- We shall focus on two significant standards:
  - *Trusted Platform Module*
  - *Trusted Execution Environments*

System level solutions

- We shall focus on two significant standards:
  - *Trusted Platform Module*
  - *Trusted Execution Environments*

Trusted Platform Module – TPM

- Standard guideline for developing chips with strong cybersecurity features
- Trustworthiness of TPM is based on different Root of Trust components and well-defined interactions among them

TPM History

- Specification initially released by the *Trusted Computing Group* in 2003

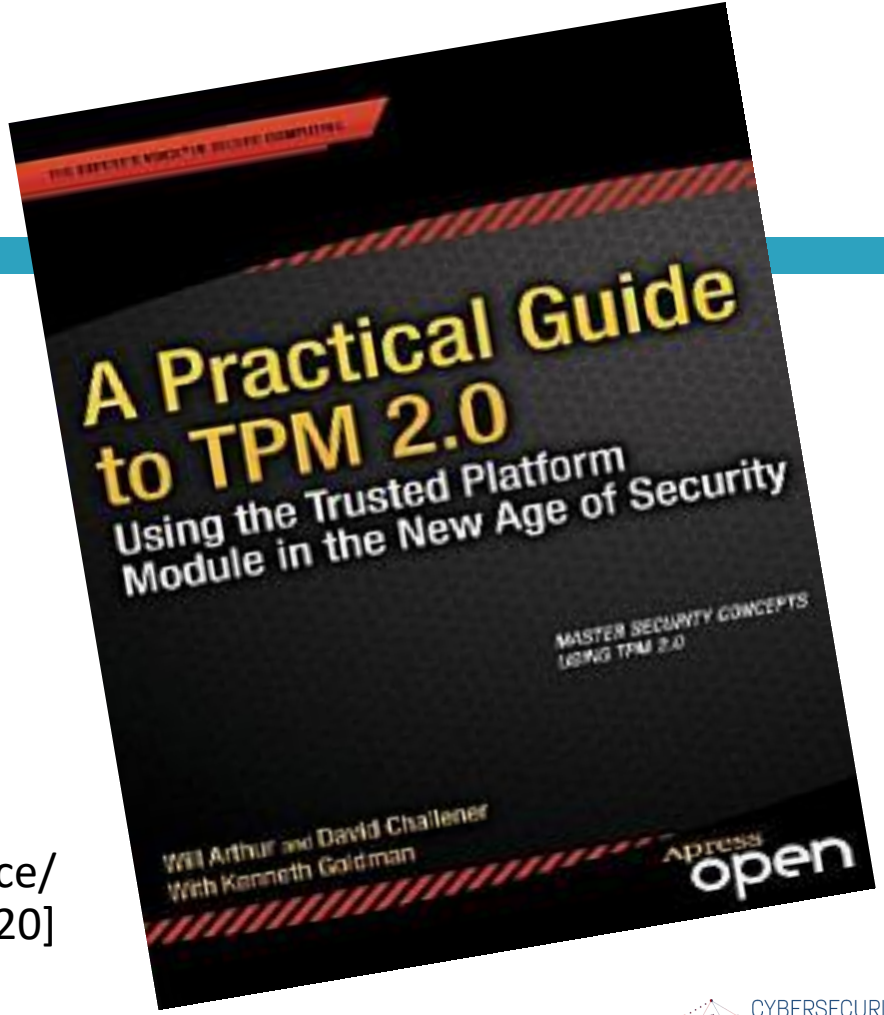
[https://trustedcomputinggroup.org/]

- The current version is TPM 2.0, which is standardized under ISO/IEC 11889

[https://www.iso.org/standard/66510.html]

[https://ebrary.net/24701/computer_science/a_practical_guide_to_tpm_20]

TPM 2.0



[https://ebrary.net/24701/computer_science/a_practical_guide_to_tpm_20]

TPM types

- There are five types of TPM: *Discrete, Integrated, Firmware, Software*, and *Virtual*
- Discrete TPM is the most common and the most secure form

Discrete TPM

- Discrete TPM is in the form of surface mounted integrated circuit on the computer's motherboard
- Many computers come with a TPM chip by default, but the TPM is inactive until it is enabled in the BIOS

Discrete TPM operations

- When you boot a computer, TPM checks the state of the computer and the state of the computer's environment
- If the computer is in a trustworthy state (i.e., it has not been tampered with), it will operate normally; else it will not boot
- TPM works by creating encryption codes. Half of the encryption key is stored on the TPM chip and the other half is stored on the computer hard drive, so if the TPM chip is removed, the computer will not boot

TPM adoption

- TPM has become standard in many consumer grade computers over the last few years
- In February 2019, the TPM was recommended for use in securing high-risk industrial devices in the newly released international standard IEC 62443-4-2
- In the automotive industry, TPM is used to guard a vehicle's software

Example of TPM adoption: Intel VPro

- The Intel platform gives a user control and adds more features to TPM, providing built-in security features for protection below the operating system
- In addition, the Intel VPro can do a secured remote diagnostic through TPM

[https://www.intel.com/content/www/us/en/architecture-and-technology/vpro/vpro-platform-general.html]

Example of TPM adoption: Microsoft's BitLocker

- BitLocker is a full volume encryption feature included with Microsoft Windows versions starting with Windows Vista
- It is designed to protect data by providing encryption for entire volumes
- By default, it uses the AES encryption algorithm in cipher block chaining or XTS mode with a 128-bit or 256-bit key

Example of TPM adoption: Microsoft's BitLocker

- Windows uses technologies including Trusted Platform Module (TPM), Secure Boot, and Measured Boot to help protect BitLocker encryption keys against attacks
- BitLocker is part of a strategic approach to securing data against offline attacks through encryption technology

Example of TPM adoption: Microsoft's BitLocker

- Microsoft's BitLocker requires TPM:
  - *"BitLocker uses the enhanced security capabilities of the TPM to make data accessible only if the computer's BIOS firmware code and configuration, original boot sequence, boot components, and BCD configuration all appear unaltered and the encrypted disk is located in the original computer."*

[https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-countermeasures]

Root of Trust in TPM

- TPM defines 3 kinds of RoTs:
  - Root of Trust for *Measurements* (RTM)
  - Root of Trust for *Storage* (RTS)
  - Root of Trust for *Reporting* (RTR)

Root of Trust for Measurement (RTM)

- Is a computing engine capable of making inherently reliable integrity measurements
- Provides measurement used by assertions protected via the RTI and attested to with the RTR
- Sends integrity-relevant information to the RTS

RTM Implementation

- Typically, the RTM is the CPU controlled by the *Core Root of Trust for Measurement* (CRTM)
- The CRTM is the first set of instructions executed when a new chain of trust is established
- When a system is reset, the CPU begins executing the CRTM
- The CRTM then sends values that indicate its identity to the RTS

Root of Trust for Storage (RTS)

- Is a computing engine capable of:
  - maintaining an accurate summary of values of integrity digests and the sequence of digests
- Provides a protected repository and a protected interface to store and manage keying material

[TCG Specification - Architecture Overview, Section 4.2, Trusted Computing Group]

[Guidelines on hardware-rooted security in mobile devices (Draft), NIST Special Publication 800-164]

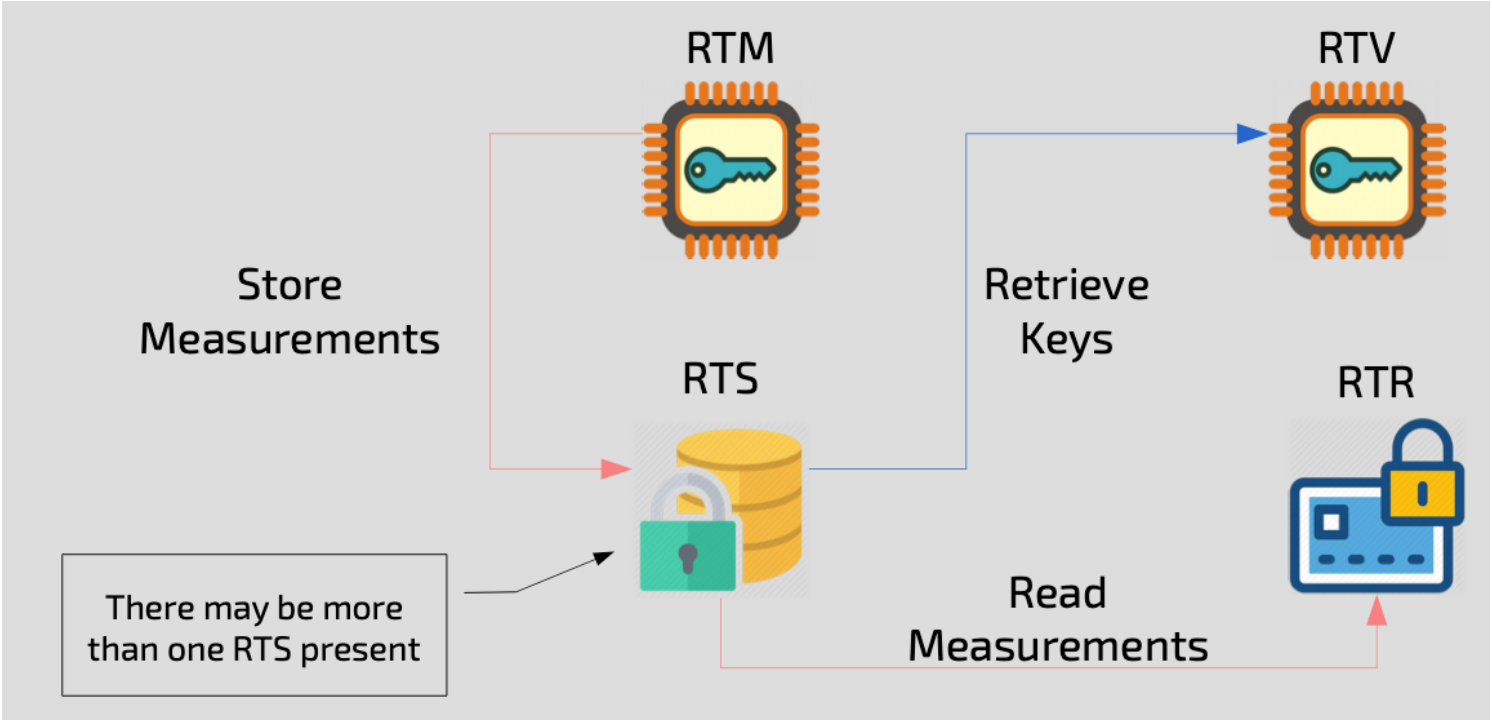
Root of Trust for Reporting (RTR)

- Is a computing engine capable of reliably reporting information held by the RTS
- Provides a protected environment and interface to manage identities and sign assertions

Root of Trust for Reporting (RTR)

- The typical values RTR report on are:
  - Audit logs
  - Key properties
- The interaction between the RTR and RTS is critical:
  - resistant to all forms of software attack and to the forms of physical attack implied by the TPM's Protection Profile
  - supply an accurate digest of all sequences of presented integrity metrics

Root of Trust Interactions



[TrenchBoot
Daniel P. Smith]

TPM Basic Features

- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

TPM Basic Features

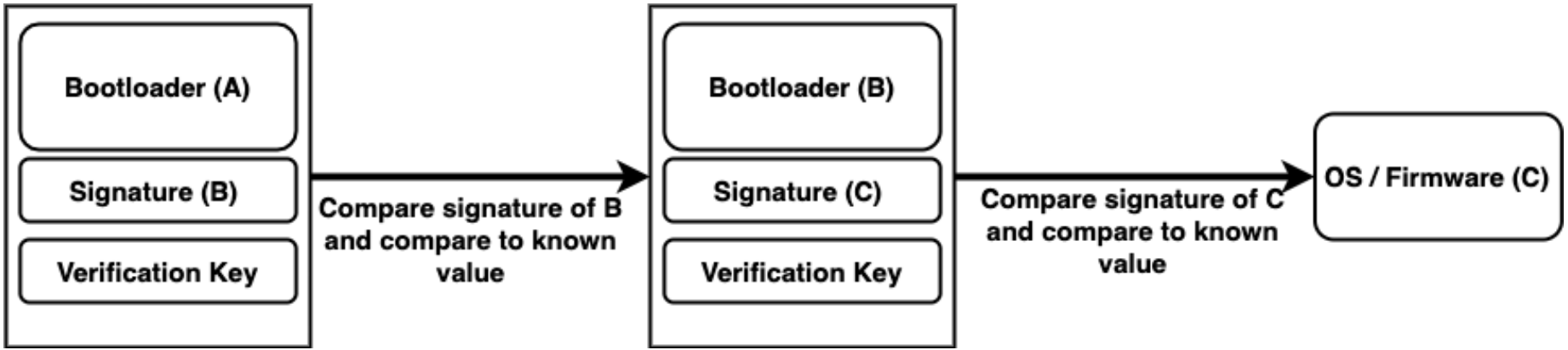
- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

Secure Boot & Firmware Integrity

- The environment in which the code runs must be controlled
- A power-on reset creates an environment in which the platform is in a well-known initial state
- *Secure Boot* is the act of establishing a secure initial state

Secure Boot & Firmware Integrity

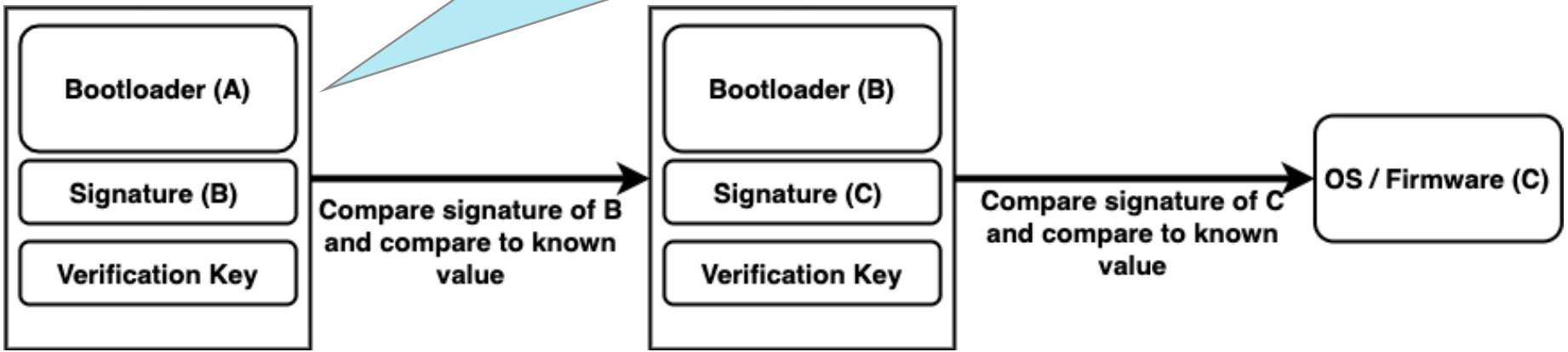
- The typical secure boot method verifies the authenticity of each component in the boot chain:



Secure Boot & Firmware Integrity

- The typi… …the authenti… …oot chain:

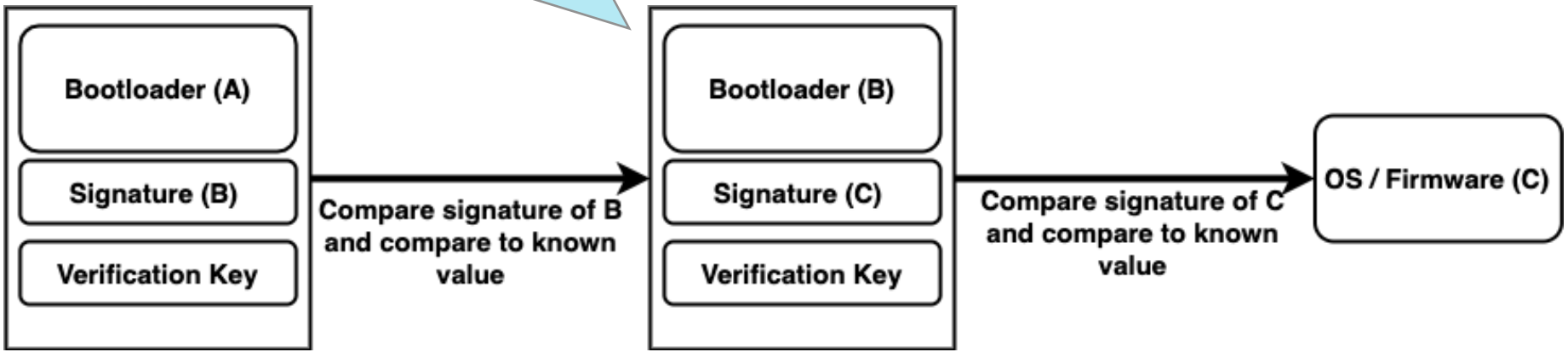
A 1st protected bootloader (A), stored in a secure memory, verifies the integrity and authenticity of a 2nd bootloader (B)



Secure Boot & Firmware Integrity

The 2nd bootloader (B) verifies the integrity and authenticity of the Operating System kernel and of the Firmware

- Th…es the au…e boot chain:



Secure Boot & Firmware Integrity

- Typically, the 1st bootloader (A) cannot be modified, whereas the 2nd bootloader (B) can be updated

TPM Basic Features

- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

Certification

- Certificate of authenticity should be available for the key shipped with the TPM
- Can be used to associate credential (certificate) with other TPMs
- A certified key that can be used for signing may be used to attest the platform data that affect the integrity (*trustworthiness*) of a platform

Certification

- Certificate and authentication credential can be stored in a Root of Trust for Storage (RTS) element

TPM Basic Features

- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

Attestation and Autentication

- Root of Trust components are usually the entities trusted when attesting to a devices
- Unique identifier can be stored in a Root of Trust element and used to identify the system
- Unique identifier can be obtained resorting to *Physically Unclonable Functions* (PUFs)

Attestation and Autentication

- Root of Trust components are usually the entities tr
- U
element an dentify the system
- Unique identifier ca be obtained resorting to *Physically Unclonable Functions* (PUFs)

See lecture:

HS_1.6 - Physically Unclonable Functions - PUFs

Attestation and Autentication

- Trusted platforms employ a hierarchy of attestation
- Extenal entities attest for different charateristics of the TPM
  - Genuine and compliant with the standard
  - Contains a RTM and exists a trusted path between the RTM and the TPM
  - A key pair is protected by a genuine TPM
  - …
- These various attestation takes the form of keys, certificates, software signature, etc. that are stored inside the TPM

TPM Basic Features

- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

Protected Location

- All information on a TPM is in a Shielded Location
- The contents of a Shielded Location are not disclosed unless intended: only the allowed entities can access the secure memory and the Root of Trust functionalities
- When sensitive data are not stored in a Shielded Location on the TPM they are encrypted

Protected Location

- Wherever sensitive data are stored outside TPM, it is in a *Protected Location*
- Encryption of Protected Locations uses multiple seeds and keys that never leave the TPM
- Tamper resistance devices are used to avoid disclose sensitive information to common physical attacks

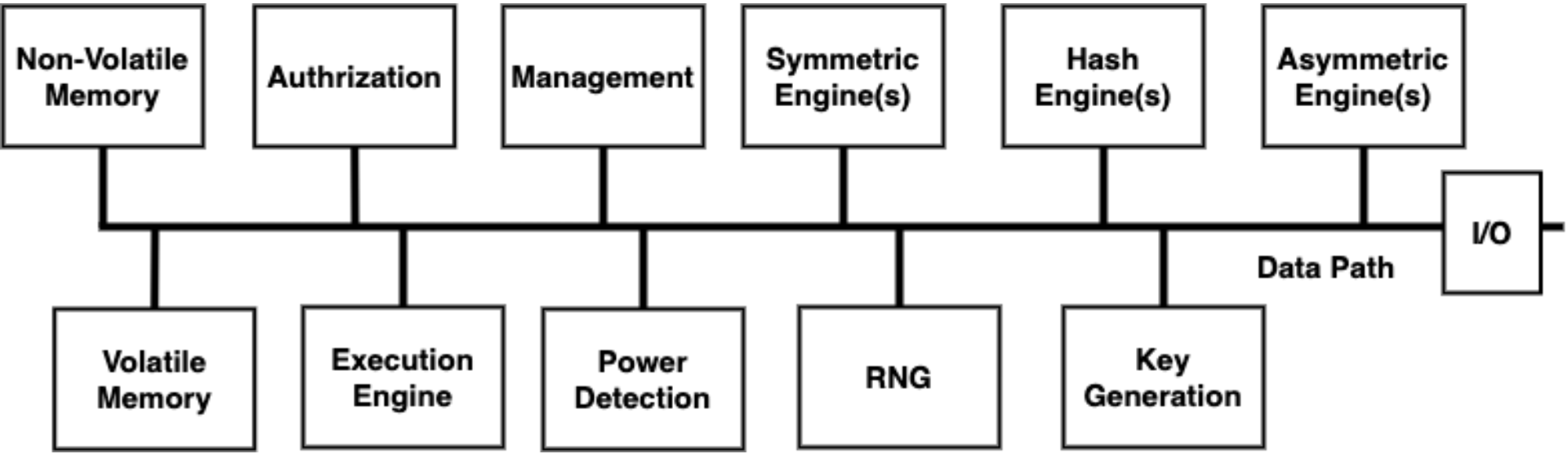
TPM Basic Features

- They include, among the others:
  - *Secure Boot & Firmware Integrity*
  - *Certification*
  - *Attestation and Autentication*
  - *Protected Location*
  - *Integrity Measurments and Reporting*

Integrity Measurments and Reporting

- An integrity measurement is a value that represents a possible change in the trust state of the platform
- The measured object may be:
  - A data value
  - The hash of code or data
- The digest of an arbitrary set of integrity measurements is statistically unique

TPM Architecture



System level solutions

- We shall focus on two significant standards:
  - *Trusted Platform Module*
  - *Trusted Execution Environments*

Trusted Execution Environments

- TEEs are secure area of a System-on-Chip that guarantee code and data protection
- They typically offer the minimal security required by low-end, closed embedded systems, such as IoT and “bare-metal” (i.e., without any Operating System) solutions

Trusted Execution Environment

- TEE was originally an initiative of Global Platform to standardize a part of the processor as a trusted secure part
- TEE has since evolved and covers in general the hardware modifications made to processors to provide isolation and attestation to software

Trusted Execution Environment

- TEE is a concept that provides a secure area of the main processor

  “to provide end-to-end security by protecting the execution of authenticated code, confidentiality, authenticity, privacy, system integrity and data access rights”

[Global Platform Device Committee, “EE protection profile,” version 1.2, Public Release, November 2014, Document Reference: GPD_SPE_021 https://csrc.nist.gov/publications/detail/fips/140/2/final]

Trusted Execution Environment

- Well known example is the ARM Trust Zone (see Part II)

Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

Architectural level solutions

- General purpose *Design-for-Security* solutions adopted at the architectural level, mainly to improve the security of the CPUs and of the involved memories

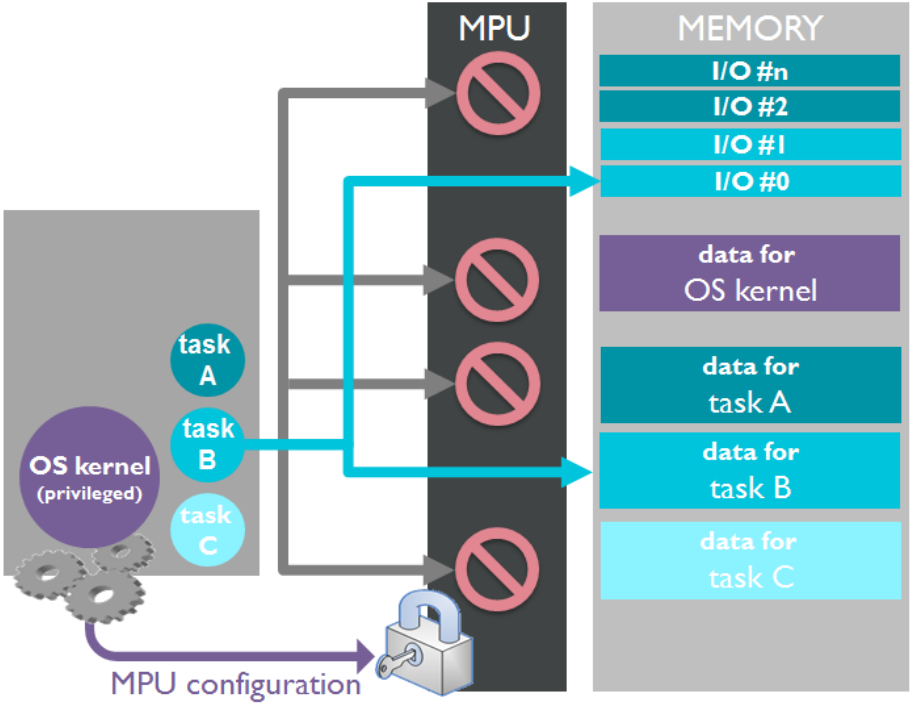
Architectural level solutions

- General purpose *Design-for-Security* solutions adopted at the architectural level, mainly to improve the security of the CPUs and of the involved memories

See lecture:
HS_1.8 - Architectural-level Protections

Memory Protection Unit - MPU

- Present in a wider and wider number of processors
- Each memory page can be read, written or executed just by a predefined set of tasks/processes
- Access rights are decided by the kernel, which runs privileged
- Addresses sent to the memory are automatically processed by the MPU without the intervention of the kernel
- Violations cause the immediate abortion of the task



Outline

- Introduction
- Roots of Trust
- Trust Anchors
- System level solutions
- Architectural level solutions
- Device level solutions

Device level solutions

- Set of solutions adopted at the device level to improve the device's resistance and resiliency to external attacks
- They include, among the others:
  - *Countermeasures for Side-channel attacks*
  - *Tamper-evident devices*
  - *Tamper-resistant devices*

Device level solutions

- Set of solutions adopted at the device level to improve the device's resistance and resiliency to external attacks
- They include, among the others:
  - *Countermeasures for Side-channel attacks*
  - *Tamper-evident devices*
  - *Tamper-resistant devices*

Countermeasures for Side-channel attacks

- *Device shielding*:
  - They typically aim at shielding the device against side-channels attacks (e.g., resorting to copper Faraday cages to prevent electromagnetic emissions)

Examples of Countermeasures for Side-channel attacks

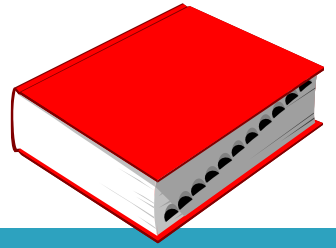
- *Side-channel Removal*: it aims at removing the sources of leakage information:
  - New EDA methodologies for security in order to force leaked information not to have any correlation between computation and data

Guilley, Sylvain, and Renaud Pacalet. "SoCs security: a war against side-channels."
In *Annales des télécommunications*, vol. 59, no. 7-8, pp. 998-1009. Springer-Verlag, 2004.

Device level solutions

- Set of solutions adopted at the device level to improve the device's resistance and resiliency to external attacks
- They include, among the others:
  - *Countermeasures for Side-channel attacks*
  - *Tamper-evident devices*
  - *Tamper-resistant devices*

Tamper-evident devices



- Devices that include some indicator of compromise, automatically activated when someone tries to mess with its physical integrity

Examples of Tamper-evident devices

- *Physical Level*
  - Packaging should maximize the evidence of tampering
  - Additional internal sensors (e.g., light detectors, temperature sensors, …) could be inserted in order to detect the presence of laser rays used to perform fault injection attacks

Waksman, Adam, and Simha Sethumadhavan. "Tamper evident microprocessors." In *2010 IEEE Symposium on Security and Privacy*, pp. 173-188. IEEE, 2010.

Examples of Tamper-evident devices

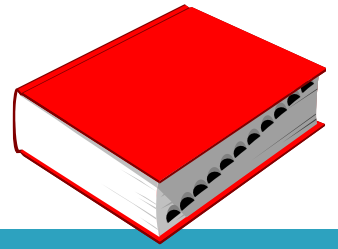
- *Software Level*
  - Appropriate auditing mechanisms through logging procedures to trace conducted activities and their sources

Waksman, Adam, and Simha Sethumadhavan. "Tamper evident microprocessors." In *2010 IEEE Symposium on Security and Privacy*, pp. 173-188. IEEE, 2010.

Device level solutions

- Set of solutions adopted at the device level to improve the device's resistance and resiliency to external attacks.
- They include, among the others:
  - *Countermeasures for Side-channel attacks*
  - *Tamper-evident devices*
  - *Tamper-resistant devices*

Tamper-resistant devices



- Device properly engineered in a way to reduce the surface for physical attacks

Examples of Tamper-resistant devices

- Strengthening of the hardware physical shielding
  - Device built in a way that an attempt of decapsulation will damage or destroy the entire chip (3D SiP)
  - Use of stronger coating for device packaging
  - Additional metallization layer on top of the actual circuit to prevent microprobing attacks
  - The circuit board need to be dipped in special material (like epoxy) to prevent easy access to the hardware

Ravi, Srivaths, Anand Raghunathan, and Srimat Chakradhar. "Tamper resistance mechanisms for secure embedded systems." In *17th International Conference on VLSI Design. Proceedings.*, pp. 605-611. IEEE, 2004.

Examples of Tamper-resistant devices

- Counterfeiting protections
  - Hiding device's names and their serial numbers from the packages in order to make more difficult for an attacker to exploit known vulnerabilities of the target devices

Ravi, Srivaths, Anand Raghunathan, and Srimat Chakradhar. "Tamper resistance mechanisms for secure embedded systems." In *17th International Conference on VLSI Design. Proceedings.*, pp. 605-611. IEEE, 2004.

Hardware-based Security Part I: Introduction & Basic concepts

Nicolò MAUNERO

Paolo PRINETTO

CINI Cybersecurity National Laboratory

https://cybersecnatlab.it