Sergio SAPONARA

The Role of Hardware in CyberSecurity

Outline

- The role of Hardware in Cybersecurity
- Hardware Security
- Hardware Trust
- Hardware-based Security

How to guarantee trust for your hardware? Can I trust this device?

Roles of hardware in cybersecurity

- The main roles of hardware (HW) when dealing with security are:
  - ***Hardware Security and Trust****: how to guarantee that the hardware used (independently if it provides or not security services) is secure and can be trusted or not (issues of backdoors, HW trojans, counterfeiting, side channel attacks,…)*
  - ***Hardware-based Security,*** *i.e. Hardware providing Security services (e.g. on-chip Random Number Generation, PUF-Physically Unclonable Functions, Cryptographic function accelerators, HSM: HW Security Modules, HW partitioning in trust zones, HW-based secure boot…)*

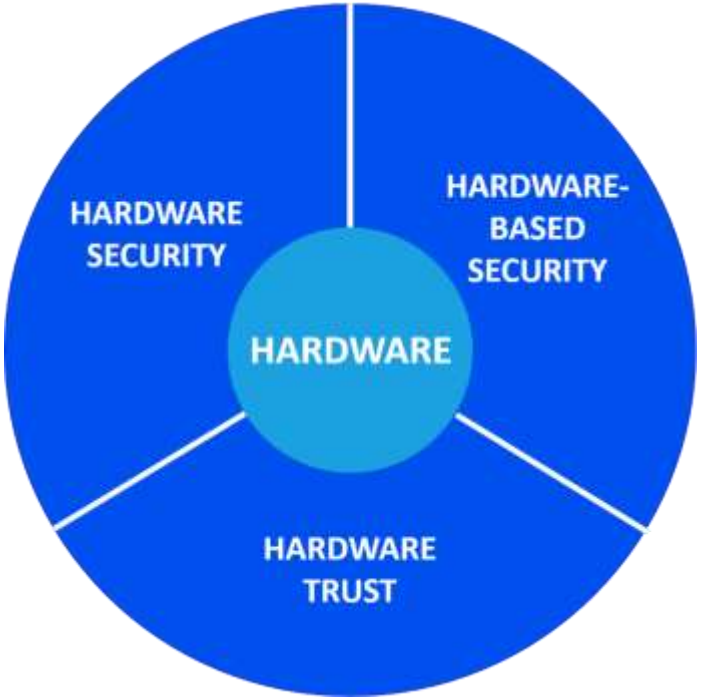
↳ Is HW secure and can it be trusted?

↳ HW can be used to provide security functions

↳ Why? HW is faster

The role of Hardware in Cybersecurity

HW security and trust are different concepts but are needed together



Hardware hierarchy in cybersecurity

This is the HW of an ECU (Electronic Control Unit) for automotive pumps control. This hardware (HW) does not provide security functions, it is a liquid (water or oil) pump!!!.
But **being related to mission-critical applications this HW should be secure and trusted.**

Hardware hierarchy in cybersecurity

To be secure and trusted, this HW needs to use a SoC (system on chip), TLE9893-2QKW62S IC, from Infineon that has on-chip HSM (hardware security module) to provide secure service

Notes on security and trust

- Hardware **Security** and **Trust** are different concepts but are strictly correlated and are needed together:

you can have a secure Hardware but if it is not identified, validated and certified you may be not trust it (consequences are extra costs, waste of functionalities, system overheads…)

or viceversa even worse a system can trust an hardware but due to an undetected cyber-attack, e.g. HW trojan or side-channel attack, or due to counterfeiting, it is no more secure (consequences are severe security issues for the whole system)

Trustness means capability to id. device, knowing its life cycle so we know it is not counterfeit ①

① This creates a cost: being able to id. the device. So we need to make sure that the money is worth it.

Notes

- Concerning Hardware-based security, the course project will be an example of a digital IP for cryptographic acceleration, i.e. how to design an HW in FPGA technology that provides a security service

Hardware to provide security function should be secure and trusted

- Obviously, you **can not have Hardware-based Security if your Hardware is not secure and/or not trusted**

Life cycle of *Hardware* Security?

The security and trustness of hardware must be analyzed, considering possible cyber attacks and their consequences,

during all phases of its life cycle:

Example of the car radio

Specification, → Someone enters design team and make fake specifications

Design, → platoon of big companies working together

Implementation,

Test,

Use,

Maintenance,

Dismission

NOTE: Checking the lifecycle is costly. So if you really need security you might spend more.

Motivations

HW as Root of Trust

- Hardware runs software and is, in fact, *the last line of defense*

Consequences (1)

- If the hardware is corrupted, all the mechanisms introduced to make the software secure (at any level) are useless

Important side effect

HW as Root of Trust

- Hardware runs software and is, in fact, *the last line of defence*

Consequences (2)

- A *trusted* and *secure* Hardware is needed to protect other system components (e.g., software, data communication infrastructures)

Important side effect

HW as Root of Trust

- Hardware runs software and is, in fact, *the last line of defence*

Consequences (3)

- If your Hardware is *not trusted* then also the software and data communication on it *are not trustable*

What are we talking about

- A multi-faceted reality

- A complex puzzle

Hardware & Security: a complex puzzle

- Hardware Vulnerabilities
- Hardware Attacks
- Hardware Root of Trust
- Hardware Counterfeiting
- Hardware-based Defenses
- Security-oriented Architectures
- Built-in security features
- HW-acceleration of security functions
- PUFs (Physically Unclonable Functions)

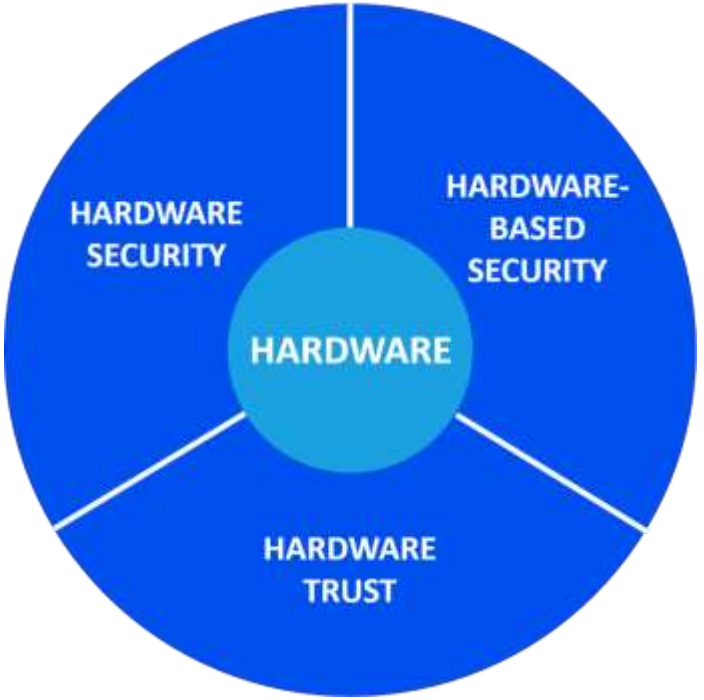
For each tile, many dimensions

- *Technology*
- *Target abstraction level*
- *Types of components*
- *Application domain*
- *System complexity*
- *System criticality*
- *…*

The role of Hardware in Cybersecurity

- Trying to move from a *mess* to a more *rigorous view*, the role of Hardware in security can be seen as follows (remember that HW security and trust are different concepts but are needed together):

Hardware has different dimensions in security:



Outline

- The role of Hardware in Cybersecurity
- Hardware Security
- Hardware Trust
- Hardware-based Security

Hardware Security: What

Of course you need to know the level of security you want to reach.

- 3 steps to achieve hardware security:
  - *Analyze hardware vulnerabilities*:
    - Their identification, detection, prevention, remediation, patching, …
    - prevention of their exploitation:
  - *Analyze possible ways of hardware attacks*:
    - Any technique and solution aimed at preventing, mitigating, defeating, making hardware attacks ineffective, regardless the tools and the abstraction levels (e.g., software or any upper level) used to carry them out
  - *Implement protection solutions*:
    - aimed at preventing hardware vulnerabilities and hardware attacks.

Remove all vulnerabilities is too costly

Prevent exploitation means understanding how attacks can be carried out.

Hardware Security: What Example1

- Over the air (OTA) update of the SW of an ECU:
  - *Analyze hardware vulnerabilities*: → Think of a car
    - the OTA update capability of an ECU can be exploited by an hacker to access remotely the internal parts of the ECU
  - *Analyze possible ways of hardware attacks*:
    - During the OTA update phase an attacker can insert a malicious SW in the ECU instead of a correct one
  - *Implement protection solutions*: confidentiality might be there for copyright, prevent someone from getting SW for free
    - Authenticate who is updating the SW (e.g. password + OTP code) and then authenticate the SW (digital signature of a certificate) that is uploaded; check the integrity of the SW that is uploaded (no manipulation of bits)
      ↳ During transfer there were errors or malicious actor modifying the code

Hardware Security: What Example2

- Diagnostic flag analysis and reconfiguration/calibration during maintenance: [for hardware]
  - *Analyze hardware vulnerabilities*: is
    - the fact that in maintenance mode the HW can be re-configured/re-calibrated or its registers can be read
  - *Analyze possible ways of hardware attacks*: [ex. sys temperature is too high and through serial port a can read]
    - An hacker can force a fault of the ECU, and when the ECU enters in maintenance/diagnostic mode the hacker can read secrets or can erroneously configure the HW
  - *Implement protection solutions*:
    - Implement panic modes to cancel secrets if the system is under attack*; authenticate who is doing maintenance of the ECU; authenticate the calibration and/or reconfiguration sequence (Validation of the reconfig)

* But after panic mode you need to reprogram HW. But it is extreme for commercial apps.

NOTE: this might be done to protect accidental changes too.

Hardware Security: When

- Hardware Security issues can be faced:
  - During the design and production phases (*Security-by-design*) →

    optimal and most effective solution

    → Also for legacy reasons
  - When hardware is already operating in the field →

    Non optimal since needs recall of the devices, adding patches…

Outline

- The role of Hardware in Cybersecurity
- Hardware Security
- Hardware Trust
- Hardware-based Security

Trust

can be applied to different elements

- "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]

I expect that under operating conditions behave in an expected way.
To trust something, we need to know what to expect from that component.

Authenticity and Trust

- *"An entity can be trusted if it always behaves in the expected manner for the intended purpose."*

[D. Grawrock, Dynamics of a Trusted Platform: A building block approach. Intel Press, 2008]

Hardware Trust : Role

Authenticate HW is easy. Authenticate the behaviour is harder

- Hardware trust mainly concerns *Authenticity,* that can be verified statically *checking for counterfeiting* and dynamically using *intrusion detection system, IDS,* to verify that during the life cycle a virus/malware has not taken the control of an hardware

ID check

Think of all the life cycle of the device.

- An asset treated or owned by an Information System component must come from an entity that is able to prove, beyond any reasonable doubt, its originality and genuineness.

for the HW.

NOTE: Anomaly detection ≠ Intruder detection

↓

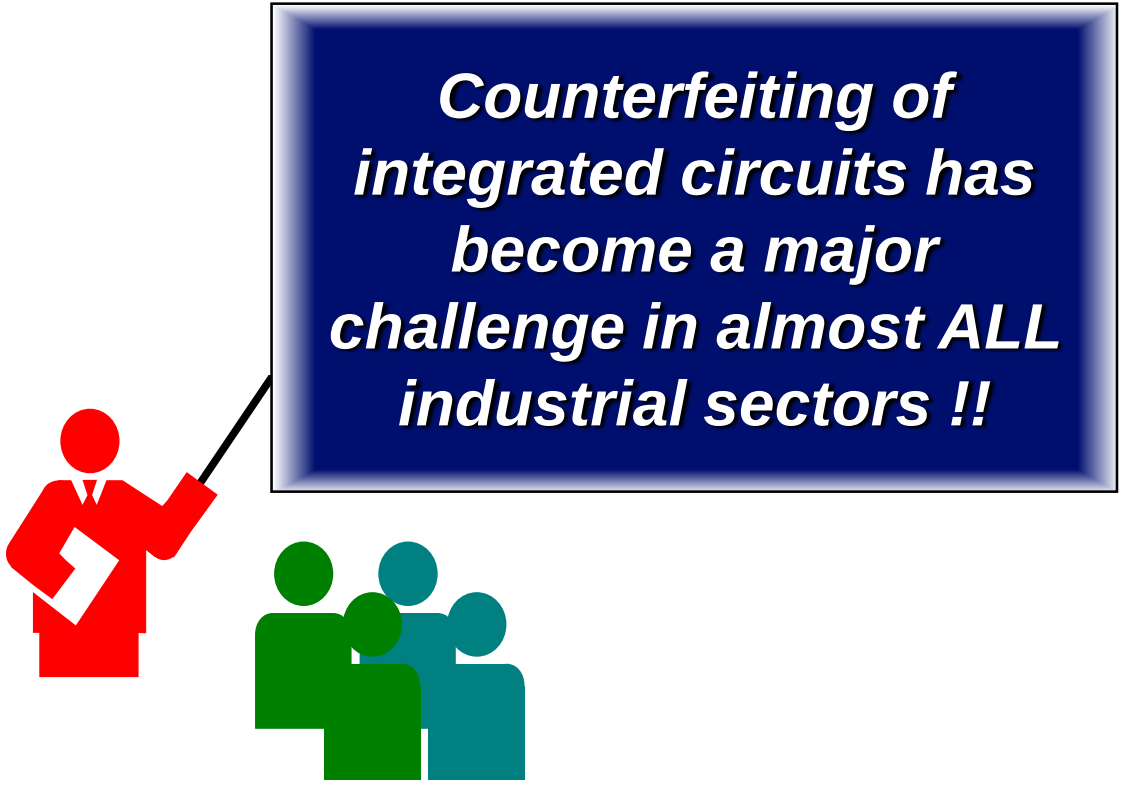
Looks for a persistent anomaly (for faults too), attackers are smarter

Hardware Trust : What

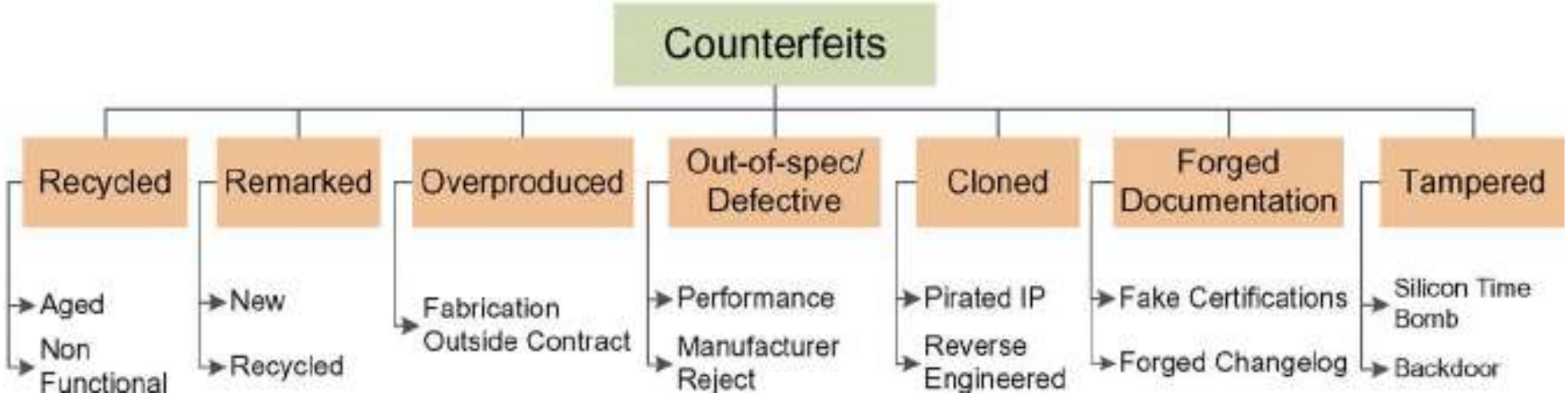
- *"Everything"* related to:
  - *hardware counterfeiting*:
    - Counterfeiting types
    - Counterfeiters
    - Counterfeiting detection approaches
    - Counterfeiting consequences
  - *protection from counterfeiting*:
    - Any technique and solution aimed at preventing counterfeiting in all the stages of the product lifecycle.

How to detect and who are the attackers

Alarm


Counterfeiting of integrated circuits has become a major challenge in almost ALL industrial sectors !!

Counterfeiting types



[Ujjwal Guin, Ke Huang, Daniel DiMase, John M. Carulli, Mohammad Tehranipoor, and Yiorgos Makris: "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain", in Proceedings of the IEEE · August 2014 - DOI: 10.1109/JPROC.2014.2332291]

Counterfeiting

How do you check your global partners?
How to check the supply chain?

Causes

- Complexity of the electronic systems significantly increased over the past few decades
- To reduce production cost, they are mostly fabricated and assembled globally (but designed somewhere else)

Who is designing, is not the one actually producing chips

Consequences

- This globalization has led to an illicit market willing to undercut the competition with counterfeit and fake parts

Counterfeiting

Lacks

- Deficiencies in the existing test solutions
- Lack of low-cost and effective avoidance mechanisms in place

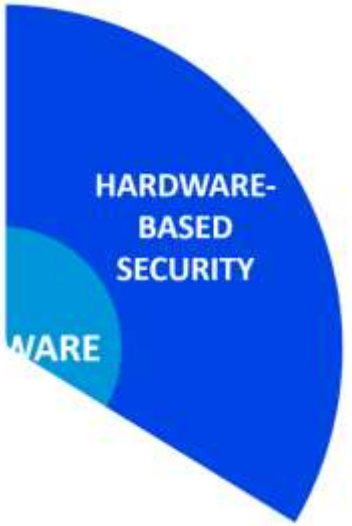
Problem is low-cost testing. Verification of HW has a cost. How can we do that at a good price!

Outline

For now we saw how to guarantee security and trust

- The role of Hardware in Cybersecurity
- Hardware Security
- Hardware Trust
- Hardware-based Security

Hardware-based Security



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

Remark

- To offer security features to upper layers, hardware itself must be secure at first
- From this point of view, *Hardware Security* play the role of a key *enabler* for *Hardware-based Security.*

Hardware-based Security Role

HW is the root of trust that starts the chain of trust.

- *”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

Hardware-based Implementations

- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platforms*

System level solutions

- Two significant De-facto standards:
  - *Trusted Platform Module (TPM)*
  - *Trusted Execution Environments (TEE)*

Trusted Platform Module – TPM

↳ is a guideline that offers rules for HW based S, now it has been standardised

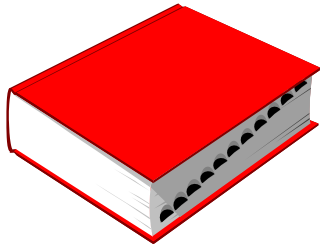
- Standard guideline for developing chips with strong cybersecurity features: a sort of rule a chip should follow to be compliant with the standard
- Trustworthiness of TPM is based on different *Root of Trust* components and well-defined interactions among them (on-chip Random Number Generators, secure key registers and data memory, on-chip HW modules for hashing and for symmetric and public key crypto functions, eFuse and/or One-Time-Programmable memory for secure boot code …)

TPM says on chip there should be: 1. an on-chip RNG (to create seeds for keys), secure key registers and data memory (you don't have a single set of keys and mem, but separated hw to store key and secure data. If the separation were virtual, you could violate virtualizity and access the wrong locations), modules on chip for hash, symmetric and public key encryption (a dedicated hw makes calculation at hw level), non volatile memory (One time programmable) in which you cannot reprogram boot code (e-fuse and/or one time programmable memory).

- Difference between TPM 1.0 and 2.0 etc, is just the version of algorithms supported or the level of entropy required for generation.

Thus ensures that hw becomes now the root of trust

Root of Trust



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

TPM History

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

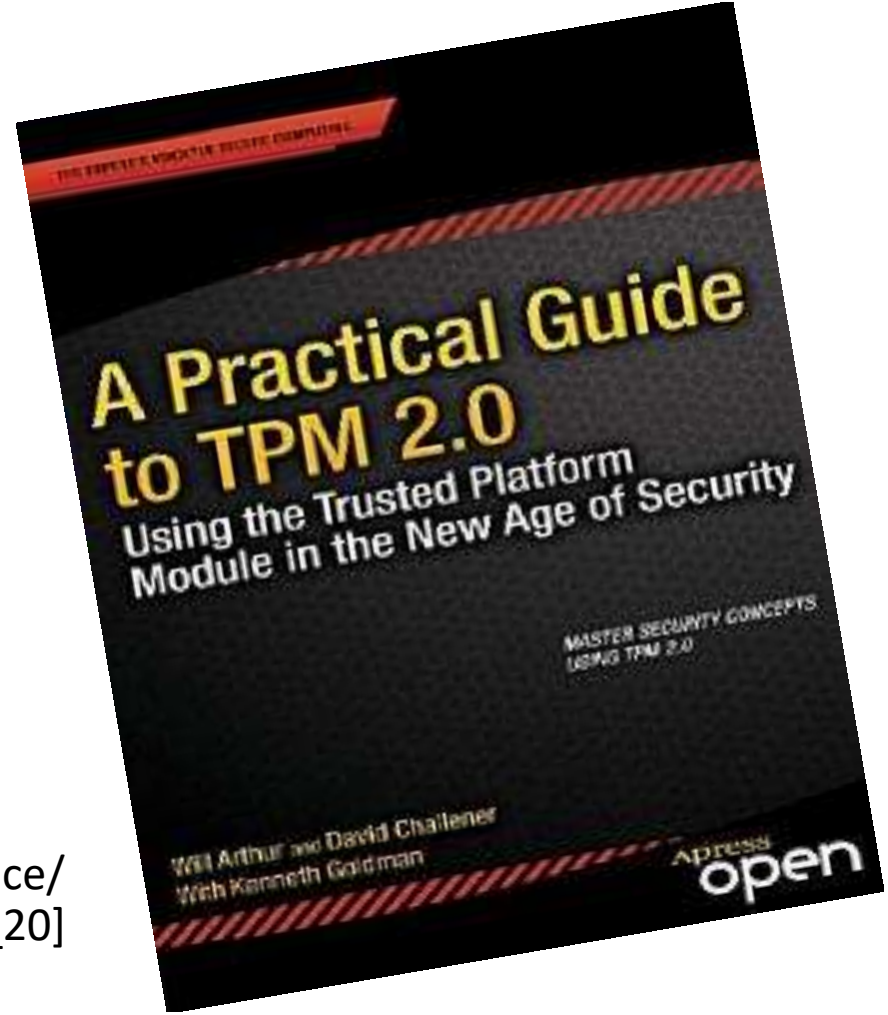
[https://trustedcomputinggroup.org/]

- The current version is TPM 2.0, which is standardized under ISO/IEC 11889: evolution is based on attacks evolution and cost efficiency increase

[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]

Trusted Execution Environment (TEE)

- 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"

  ↳ it is based on TPM but then you need middleware security (OS, Hypervisor).

  [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]

A system that is TPM compliant does not necessarily offer a TEE.

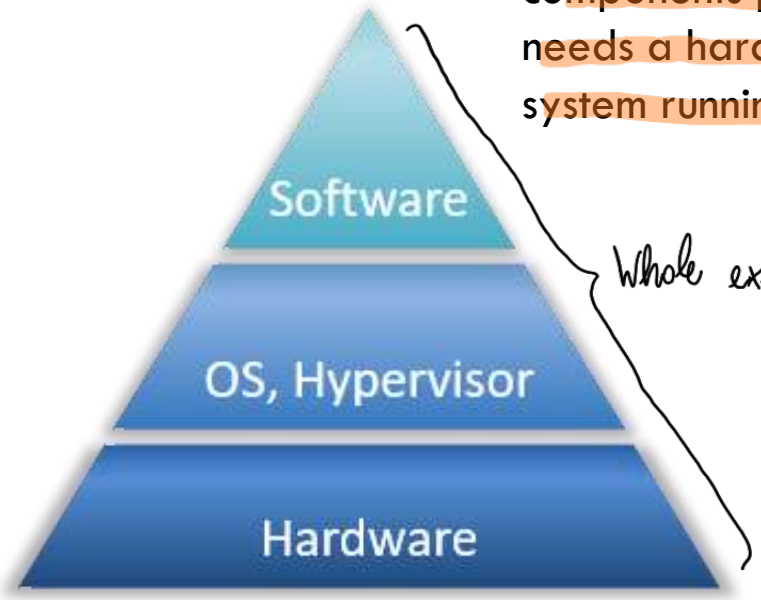
Trusted Execution Environments (TEE)

TPM is a specification of HW. A TEE is an environment in which you execute SW. In it you usually have hw providing an environment, the OS enriching it. Both of them provide security feature

- 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.

TEE and TPM

The TEE typically consists of a set of hardware and software components providing facilities necessary to support Application: TEE needs a hardware isolation mechanism, plus a secure operating system running on top of that isolation mechanism



Whole execution environment. TPM vs HW, TEE vs a set of HW and SW

Usually exam question

TEE and TPM

→ is one that sees the final user. TPM does not see the final user, which sees the TEE

A Trusted Execution Environment (TEE) for the final user includes not only a secure hardware but also secure components for the low-level SW i.e. for the operating system, drivers, hypervisor

Instead, the TPM refers only to the fact that the hardware has some built-in security features. A TPM may be a component of a TEE but alone the TPM is not enough to guarantee a TEE
https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/

Example of secure OS, secure hypervisor are the ProvenVisor and ProvenCore by provenrun
https://provenrun.com/products/provencore/
https://provenrun.com/products/provenvisor/

Standardization of TPM is useful to have SW that runs on different HWs because you have standards.

Hardware-based Implementations

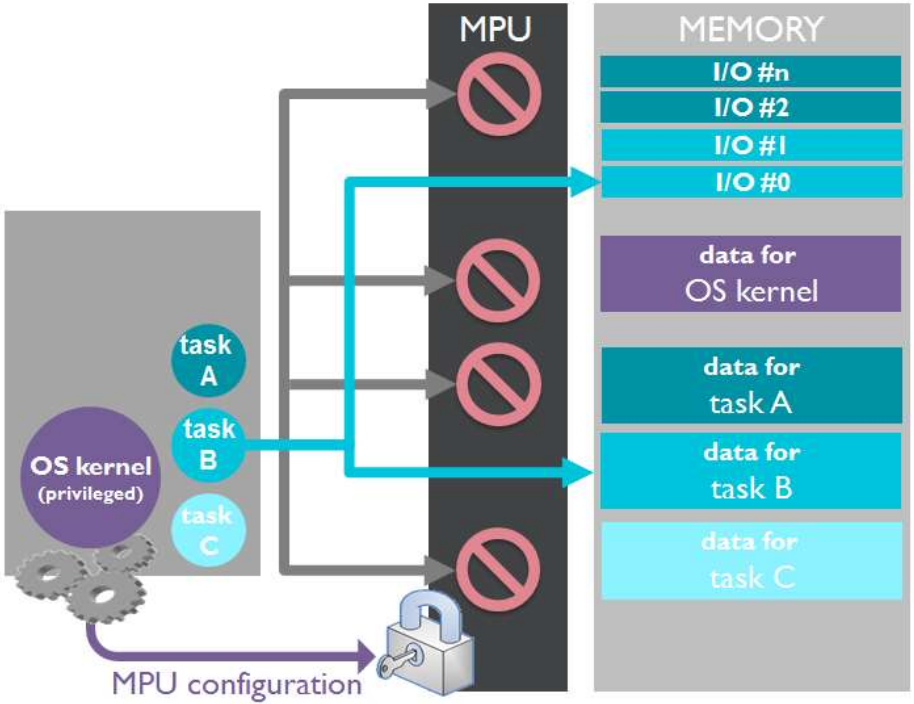
- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platform*

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.

Example of architectural-level security: 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/processes①
- Access rights are decided by the kernel, which runs in privileged session a MPU HW configuration or even more secure MPU configuration is stored in a NV memory read at boot phase
- Addresses sent to the memory are automatically processed by the HW MPU without the intervention of the kernel ②
- Violations cause the immediate abortion of the task



TPM does not specify how to deal with memory access or implementation of separated registers.

We add a sort of HW guardian. CPU has to execute tasks, but we check accesses on HW. Maybe a process has some labels associated, and those are checked by MPU.
② Additional protection layer. MPU is configured by boot software. So even in case of malwares, they still cannot change antifuses and OTP, so we keep security.
So the access rights to memory locations for processes is not handled by kernel.
The configuration is set at boot level.

Hardware-based Implementations

- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platforms*

Security-oriented components

- Set of custom, special purpose components used for performing specific security-oriented operations, including:
  - *Hardware Cyphers (i.e. Hardware accelerators for cryptography)*
  - *Smart Cards & SIM Cards*
  - *Secure storage devices and management (for users' data, programs, secret keys, …)*
  - *Random Number Generators*
  - *Physically Unclonable Functions*
  - *Anti tamper device*
  - *Secure boot process*
  - *Side-channel attack HW countermeasures*
  - *HW-based anomaly/intrusion detection and fingerprinting*
  - *…..*

devices separated physically

① circuit that exploit the HW vulnerability to create an internal signature

NOTE: if you put HW solution but you don't have drivers to use it, then they are useless. Important to know if HW is compliant with the OS.

TPM specifies a collection of these

① Physical attacks might be carried out. Attackers might measure electromagnetic emission, or DOS by damaging devices or modify it etc.

① Are systems more related to packaging to block access.

Ex: using a seal, but that works only with a very frequent visual inspection.

Modern approaches: magnets or sensors communicating in which if the sensors don't communicate (or accelerometers) they send alarms.

Also important for vandalism

② One of the basic concepts is secure boot: boot is that part of a program that checks if hardware is ok and then launches the OS. If altering the boot you modify that program that is dangerous. Since boot programs are getting large, the non volatile memory available is not enough to store boot program, you have external memory for the rest of the program. This is also because a part of the boot might be architecture dependent. And this can be a weakness. People can change that boot program part. Modern processors have now secure boot: outside sequence is encrypted, signed etc.

So, hw should have devices to evaluate decryption and signature

③ Means trying to get info from the system by measuring something correlated to it w. not directly but from a side channel.

④ Try to decorrelate emission with what you are transmitting: example:

- 01101101. Transmit this. You can use a NRZ to transmit in which 0 is low level, 1 is high level. Strong correlation between data and emission. What if: 0: ⌊ 1: ⌋⌈, this is a relax to zero encoding. Here correlation between emission level and content is reduced.
- Similar effect for clock getting. You can create a database of power consumption for what the processor is doing, you can look and say "hum... this chip is using ALU... now it's using memory etc.". To avoid this you could generate noise (so waste power) so you decorrelate. Of course you are using more power.
- Another example, to reduce correlation clock frequency can be spreaded to make synchronization for an attacker difficult.

⑤ If a malicious SW takes control of ECU, I want something to verify if the ECU is actually itself.

Fingerprinting: physical signatures to check for integrity.

- How to create this? We usually work with some physical aspects of chip.

↳ detection of intruder by comparing a physical signature with content of the packet or behaviour of chip. So if a board is behaving in a way that mismatches the digital signature, you may have an intruder.

Substitution attacks at low level are not stopped by SW level intrusion detection systems.

NOTE: All of these features require SW modifications to exploit them, but also you pay with overhead and cost.

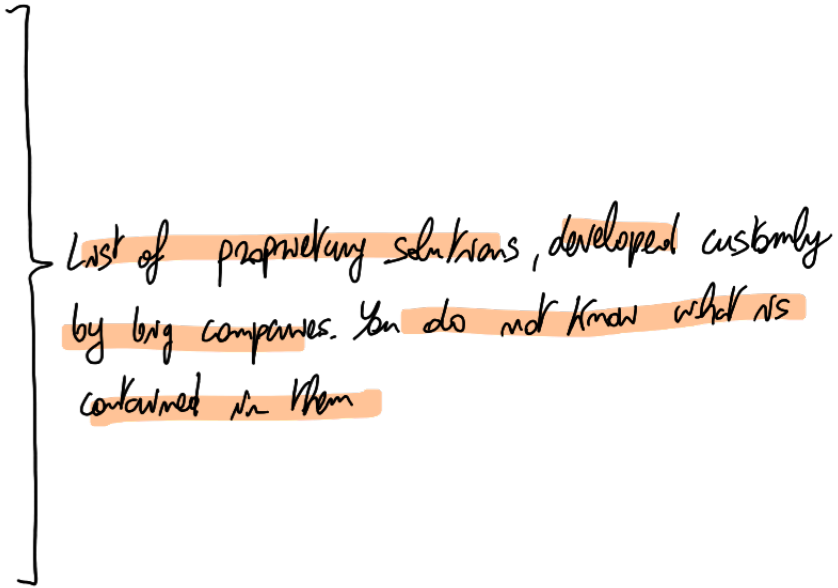
Hardware-based Implementations

- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platforms*

Transparency for security could leak info. Should security be hidden or not?

Proprietary Solutions

- *Intel® vPro® Platform*
- *AMD Secure Technology™*
- *ARM® TrustZone®*
- *Microsoft BitLocker*
- *Synopsys DesignWare® tRoot™*
- *Apple Secure Enclave Processor*
- *Google Titan*
- *Cisco® Trust Anchor*
- *…*



Open Security Platforms

- Platforms designed with cybersecurity in mind and packed with strong cybersecurity features available as open hw and including:
  - Hardware accelerators for cryptography
  - Anti tamper
  - Secure boot process
- They include for example:
  - SE*cube*™
  - USB Armory
  - OpenTitan

↓ Community of developers with licenses in which if you use them you have to share results.

↳ Maybe you find an open source HDL description for AES.

Does this increase risks? Only risk is if you have backdoor or trojan inside. Plus you give a bit of information about solution you are using.

opentitan

- https://opentitan.org/
- https://github.com/lowRISC/opentitan
- Reference to the opensource RISCV instruction set processor (Google, WesternDigital, Seagate, ETHZ, lowrisc consortium,..)

RISC 5: worldwide community developing an open instruction set to design processors.

SEcube™

https://www.secube.eu/

Reconfigurable opensecure HW and SW on FPGA technology

https://www.secube.blu5group.com/resources/open-sources-sdk/

You will need to check the licence associated to open IP. They might force you to release sdw for free.

Thanks for your attention
Any questions so far?