

# The Role of Hardware in CyberSecurity

# Outline

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

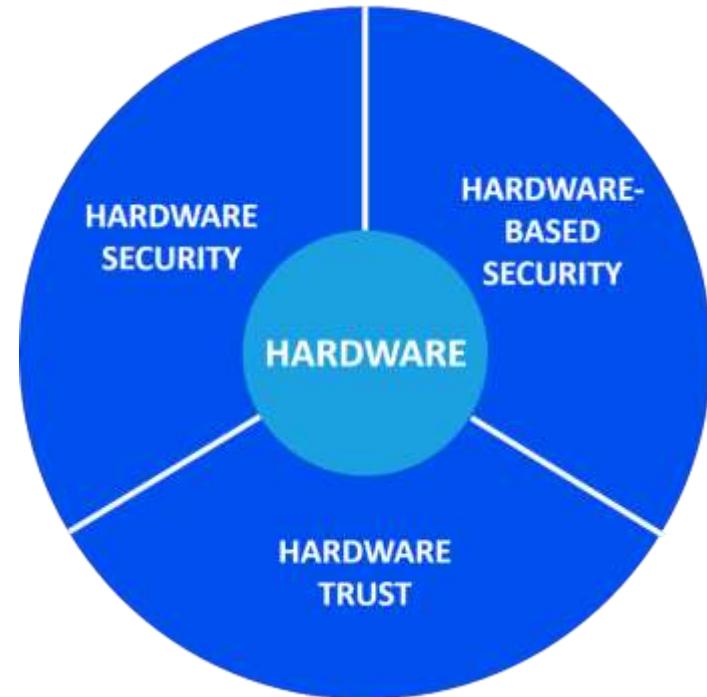
How to guarantee trust for your hardware? Can I trust this device?

# Roles of hardware in cybersecurity

- The main roles of hardware (HW) when dealing with security are:
  - Is HW secure and can it be trusted?
- **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,...)
  - HW can be used to provide security functions
- **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...)
  - 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)

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
- Obviously, you can not have **Hardware-based Security if your Hardware is not secure and/or not trusted**  
A handwritten note in black ink is positioned above the second bullet point. It consists of two parts: "Hardware to provide security function should be" on top and "secure and trusted" below it, connected by a single vertical arrow pointing upwards from the word "Hardware". The entire note is enclosed in a thin orange rectangular border.

# 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:

*Specification*, → Someone enters design team and make fake specifications

*Design*, → plan of big companies working together

*Implementation*,

*Test*,

*Use*,

*Maintenance*,

*Dismission*

Example of the car radio

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

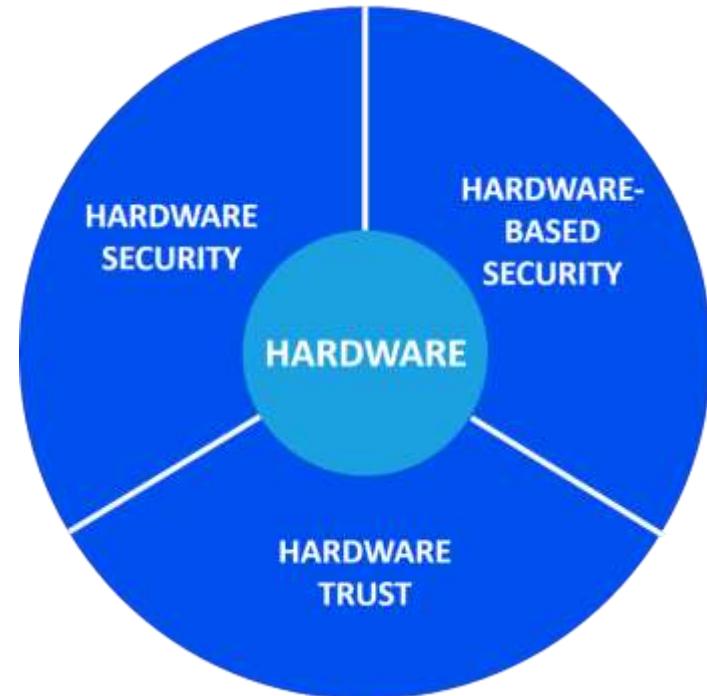


- *Technology*
- *Target abstraction level*
- *Types of components*
- *Application domain*
- *System complexity*
- *System criticality*
- ...

# 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

## ➤ 3 steps to achieve hardware security:

### ➤ Analyze hardware vulnerabilities:

- Their identification, detection, prevention, remediation, patching, ...
- prevention of their exploitation:

→ Remove all vulnerabilities is too costly

### ➤ 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.

→ 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 where for copyright, prevent someone from
    - Authenticate who is updating the SW (e.g. password + OTP code) and then get the SW authenticate the SW (digital signature of a certificate) that is uploaded; check the update integrity of the SW that is uploaded (no manipulation of bits)
      - ↳ During transmission 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:
      - 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]
      - 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 accelerated 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 of 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
  - ID check
- **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
  - 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 ≠ Malicious detection

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

# Hardware Trust : What



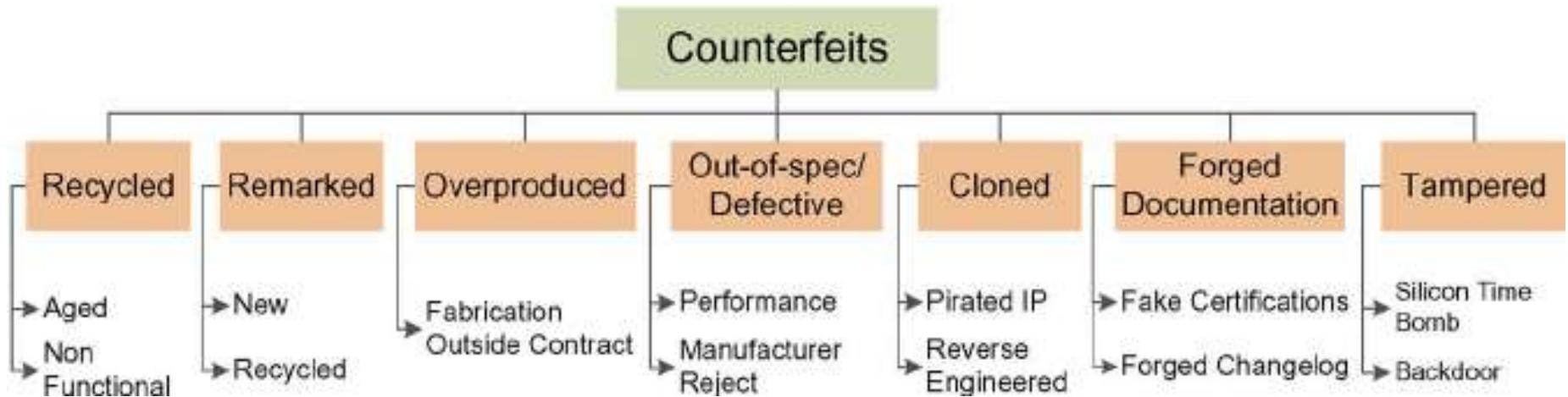
- “*Everything*” related to:
    - *hardware counterfeiting*:
      - Counterfeiting types
      - Counterfeitors
      - Counterfeiting detection approaches
      - Counterfeiting consequences
    - *protection from counterfeiting*:
      - Any technique and solution aimed at preventing counterfeiting 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

## 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

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

# 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."

STOP

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

# 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

- 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 (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 ...)

# 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

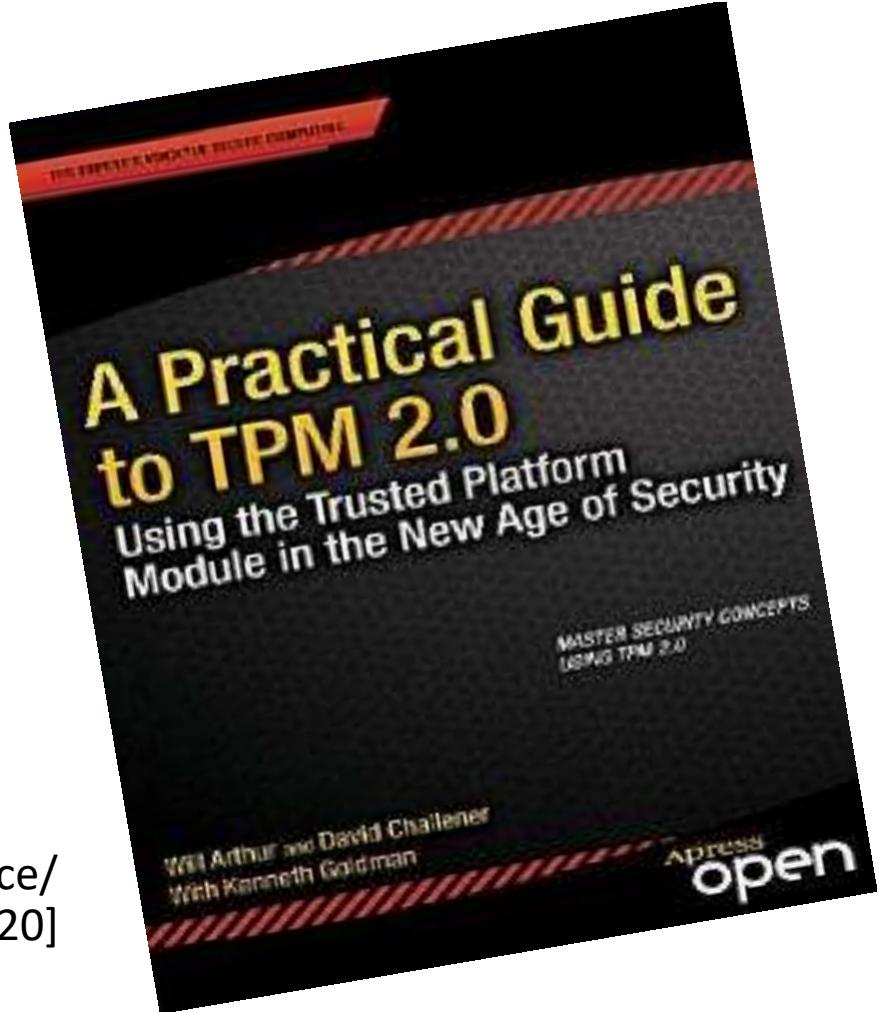
- Is used as basic block for the construction of a *Chain of Trust*

# 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  
[<https://www.iso.org/standard/66510.html>]  
[[https://ebrary.net/24701/computer\\_science/a\\_practical\\_guide\\_to\\_tpm\\_20](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](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”

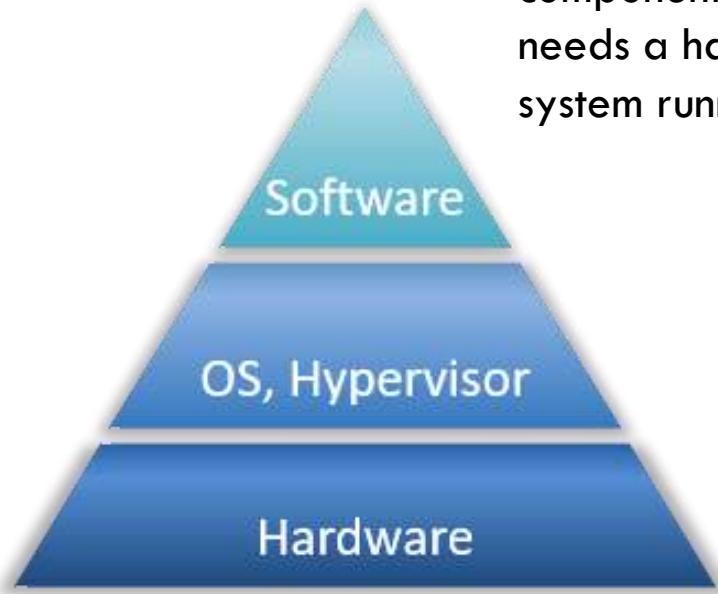
[Global Platform Device Committee, “EE protection profile,” version 1.2, Public Release, November 2014, Document Reference: GPD\_SPE\_021  
<https://csrc.nist.gov/publications/detail/fips/140/2/final>]

# Trusted Execution Environments (TEE)

- 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



# TEE and TPM

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/>

# Hardware-based Implementations

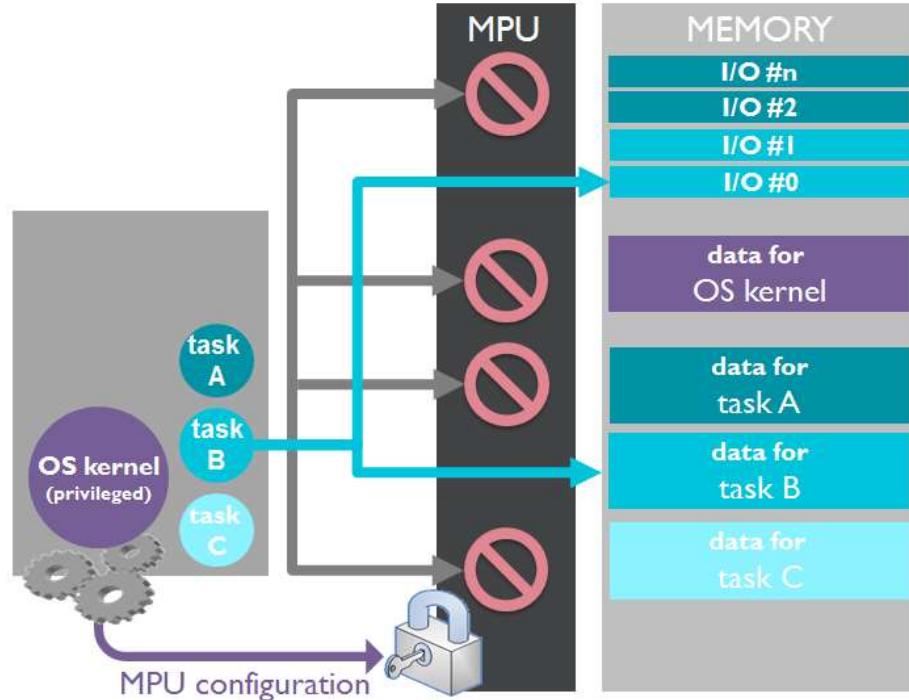
- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platform*

# 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/processes
- 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
- Violations cause the immediate abortion of the task



# 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*
  - .....

# Hardware-based Implementations

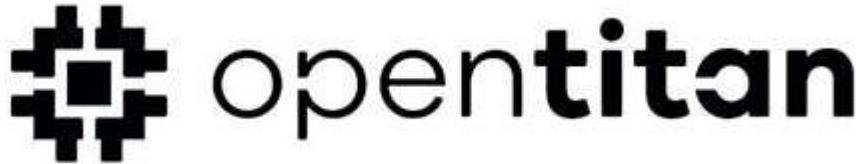
- Hardware-based Implementations can be clustered as:
  - *System level solutions*
  - *Architectural level solutions*
  - *Security-oriented components*
  - *Proprietary Solutions vs. Open Security Platforms*

# 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:
  - SEcube™
  - USB Armory
  - OpenTitan



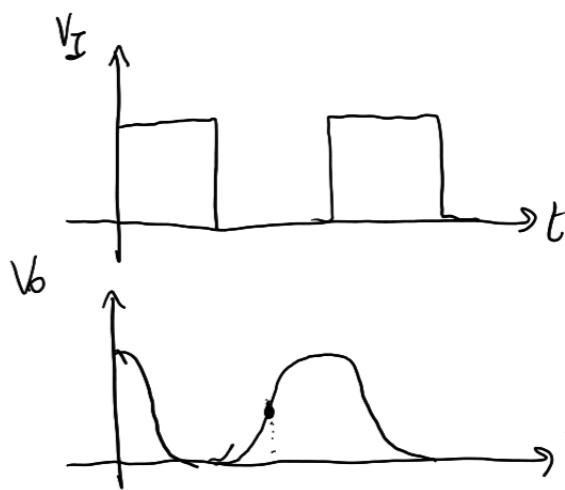
- <https://opentitan.org/>
- <https://github.com/lowRISC/opentitan>
- Reference to the opensource RISCV instruction set processor (Google, WesternDigital, Seagate, ETHZ, lowrisc consortium,..)



<https://www.secube.eu/>

Reconfigurable opensecure HW and SW on FPGA technology

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

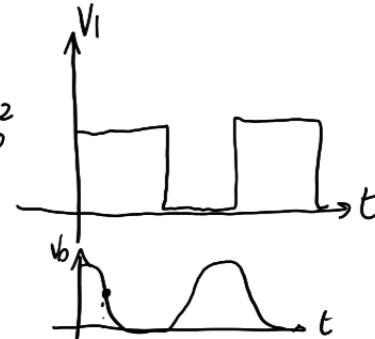


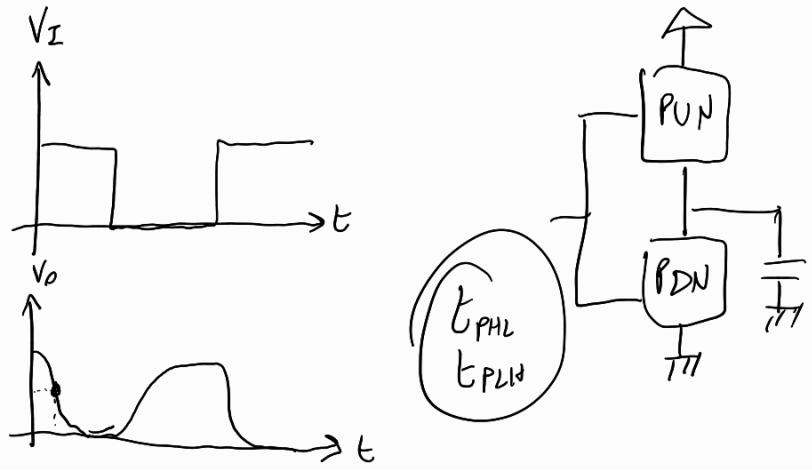
$$P_{\text{dym}} = f \cdot C_L V_{00}^2$$

Thanks for your attention  
Any questions so far?

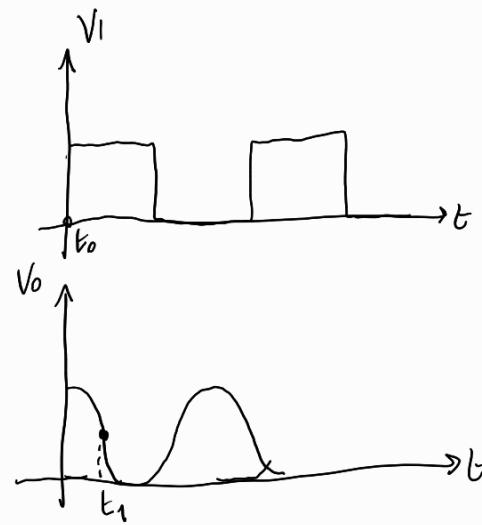
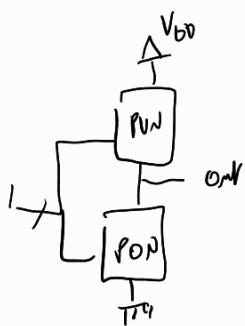
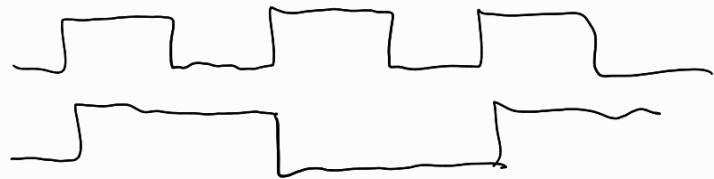
$$P_{\text{dym}} = f \cdot C_L \cdot V_{00}^2$$

$$P_{\text{dym}} \propto f_{\text{cur}} \cdot C_L V_{00}^2$$





$\text{PUN} = V_{DD}^2 C_L f$



$$\text{Pdgm} = \alpha_{0 \rightarrow 1} f_{ck} C_L V_{DD}^2$$

