ORSHIN
Trusted Life Cycle & Attack Defence Framework

Jan Pleskac, Tropic Square

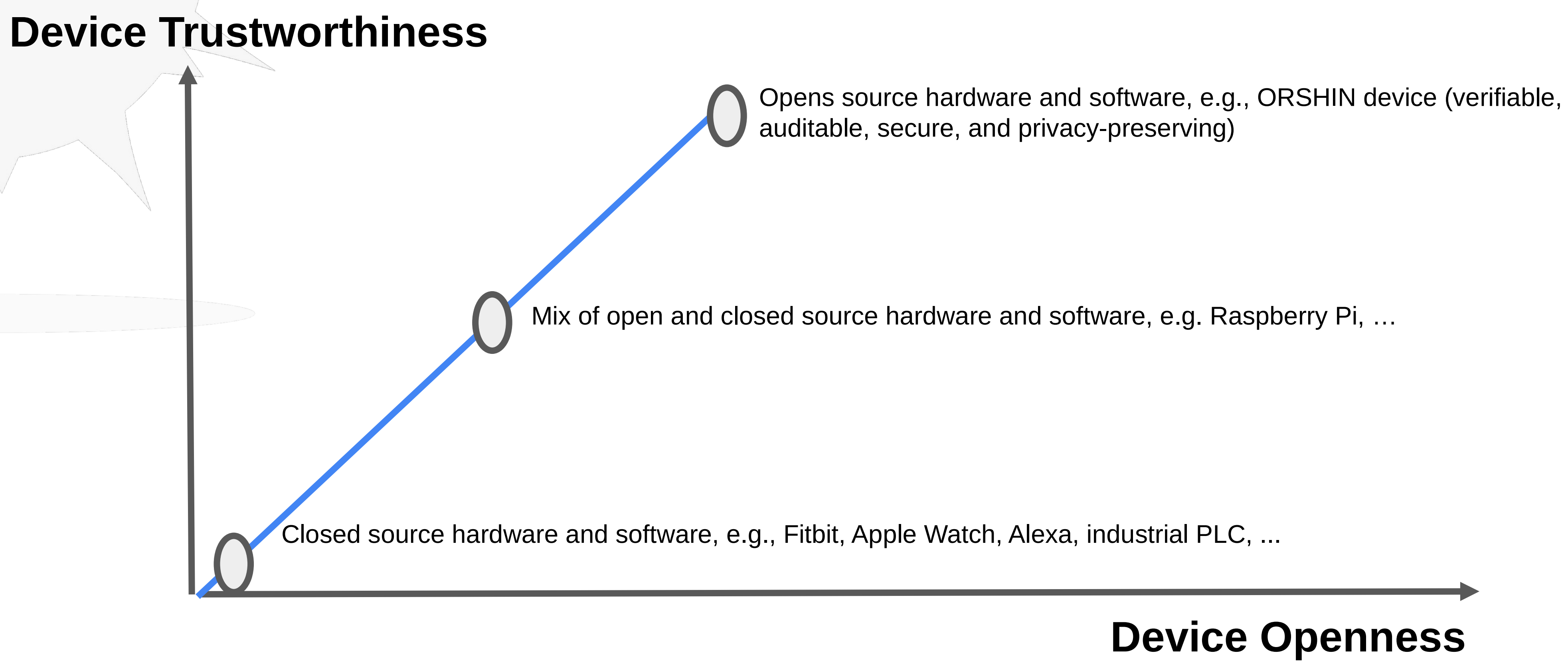
2024/06/28 RISC-V Summit CROSSCON & (Secure) Friends

horizon-orshin.eu

ORSHIN [ref]

- ORSHIN
  - Open-source ReSilient Hardware and software for Internet of thiNgs
- ORSHIN (electronic) device
  - Is resource constrained and connected
  - Provides secure and privacy-preserving services
  - Uses open-source hardware and software
  - Built and managed with a trusted life cycle
- ORSHIN use cases
  - IoT: smart wearables, digital payments, …
  - IIoT: industrial control, vehicle control, …
  - Critical infrastructure

ORSHIN Context

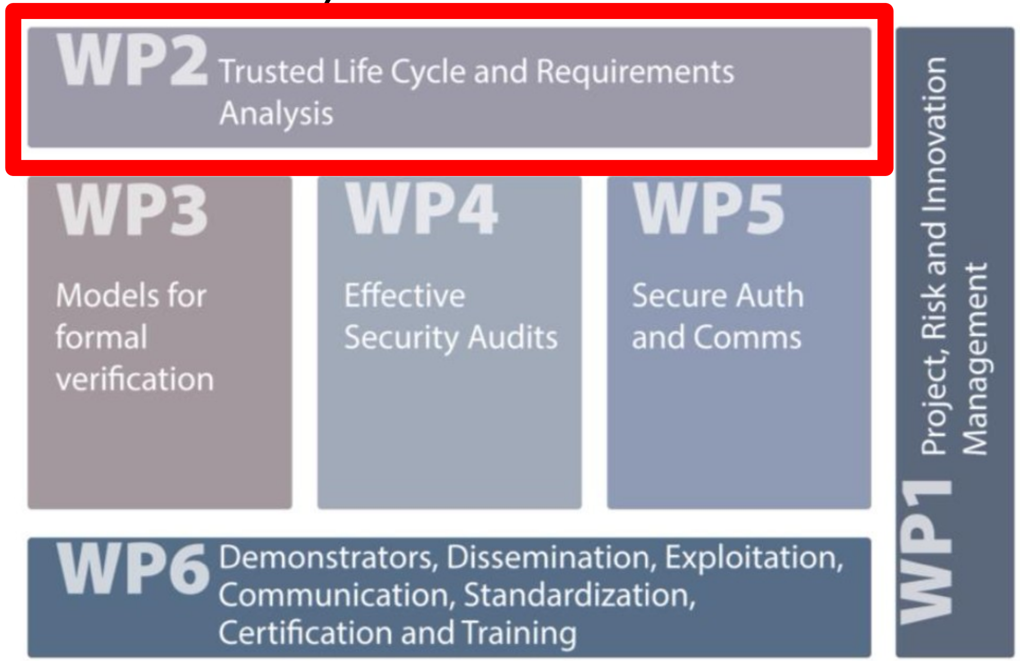


ORSHIN Consortium Members

- Academia
  - EURECOM (ECM):
  - Katholieke Universiteit Leuven (KUL):
- Industry (incl SME)
  - Security Pattern (SEP):
  - NXP Semiconductors (NXP):
  - Texplained (TEX):
  - Tropic Square (TRPC):
- Project management
  - Technikon (TEC):

Today’s focus – WP2

- Define a **methodology** to develop secure and privacy-preserving (I)IoT devices taking advantage of open-source hardware (and software)

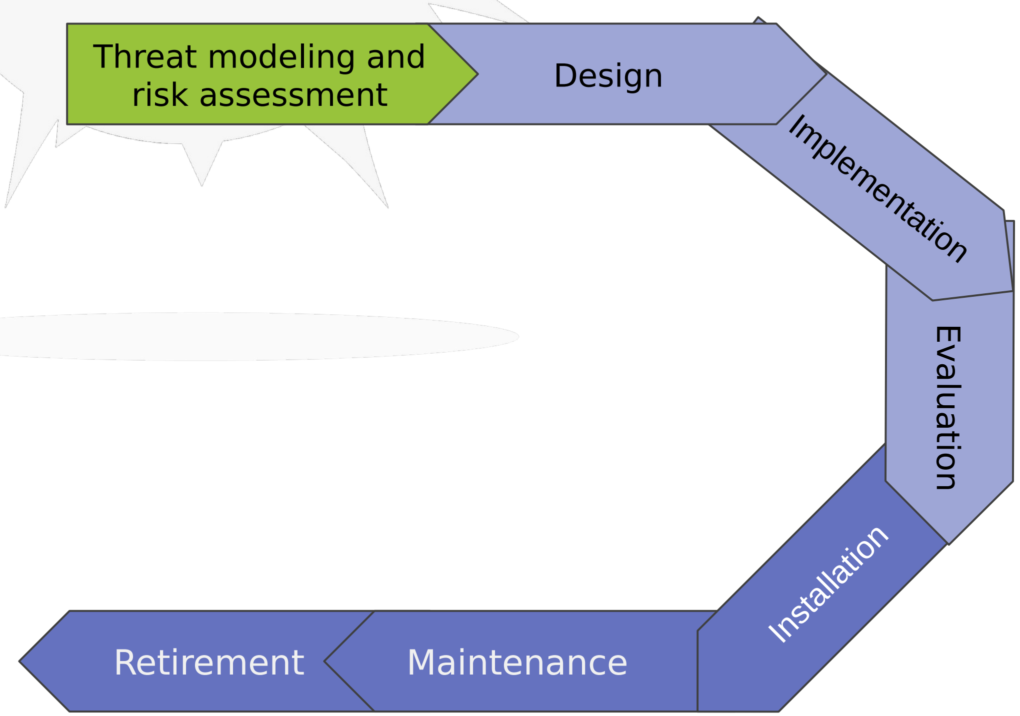


Tasks

- **Task 2.1: Methodology for the trusted life cycle of OSH**
  - Description of the **design methodology**
  - How a system developer should apply the methodology
  - Allow independent evaluation of OSH components
  - **Steps of the life cycles**
    - Threat modeling and Risk Assessment, Design, Implementation, Evaluation, Installation, Maintenance, Retirement
- **Task 2.2: Security requirements in the trusted life cycle of OSH**
  - How abstract security requirements map into **concrete policies** for the life cycle phases and into **concrete security requirements for the components of the device**

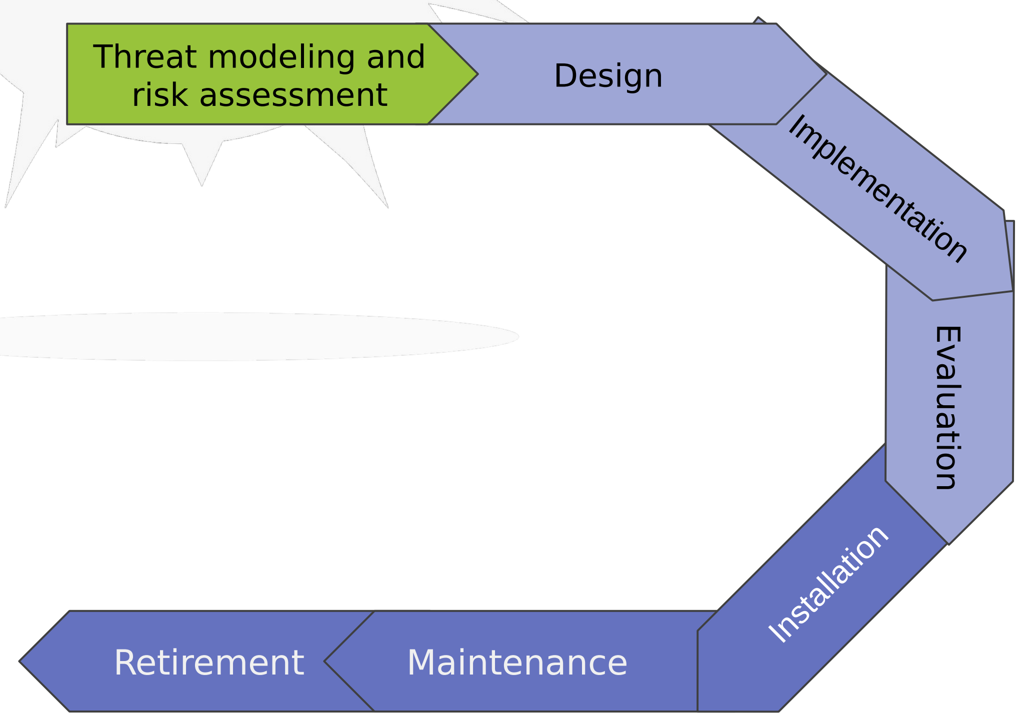
Methodology for the trusted life cycle of OSH

TLC Phases



- **Threat modeling and risk assessment**
  - Definition of security requirements, a set of product requirements for guaranteeing certain cybersecurity properties
- **Design**
  - The product hardware is designed
- **Implementation**
  - Development of hardware and software/firmware components
- **Evaluation**
  - Developed components are tested

TLC Phases



- **Installation**
  - Embedded devices get deployed and installed in the field in restricted or constrained environments
- **Maintenance**
  - After installation, embedded devices may be remotely monitored and maintained in the field
- **Retirement**
  - When a device reaches the end of its life, it is retired from the field

TLC process requirements definition: steps

- Started from **existing process requirements**, from some well-known cybersecurity standards
- **Adapted** these pre-existing requirements to the ORSHIN context
- Defined requirements that are specific to **hardware design** and **open source**
- In total, we defined 95 requirements
  - 73 arranged from existing
  - 22 new specific for HW and OS

ORSHIN Trusted Life Cycle vs State of the Art

- Started with several **existing standards** e.g.:
  - Industrial automation: ISA/IEC 62443
  - Automotive: ISO/SAE 21434
  - Medical: ISO 81001-5
- **None** of them is **addressing** the problematics of **Open Source HW/ Silicon** and we don't see a trivial way of applying one of them to this landscape
- Defined the **Trusted Life Cycle**
  - Phases of the TLC process requirements
  - Definition of open source HW & scoring

Example of new requirements

Selection of third-party components in Open Source (Process category)

To make it easier for others to replicate and modify the hardware, when possible it is better to prefer the use of free and open-source third-party components, as opposed to proprietary technology.

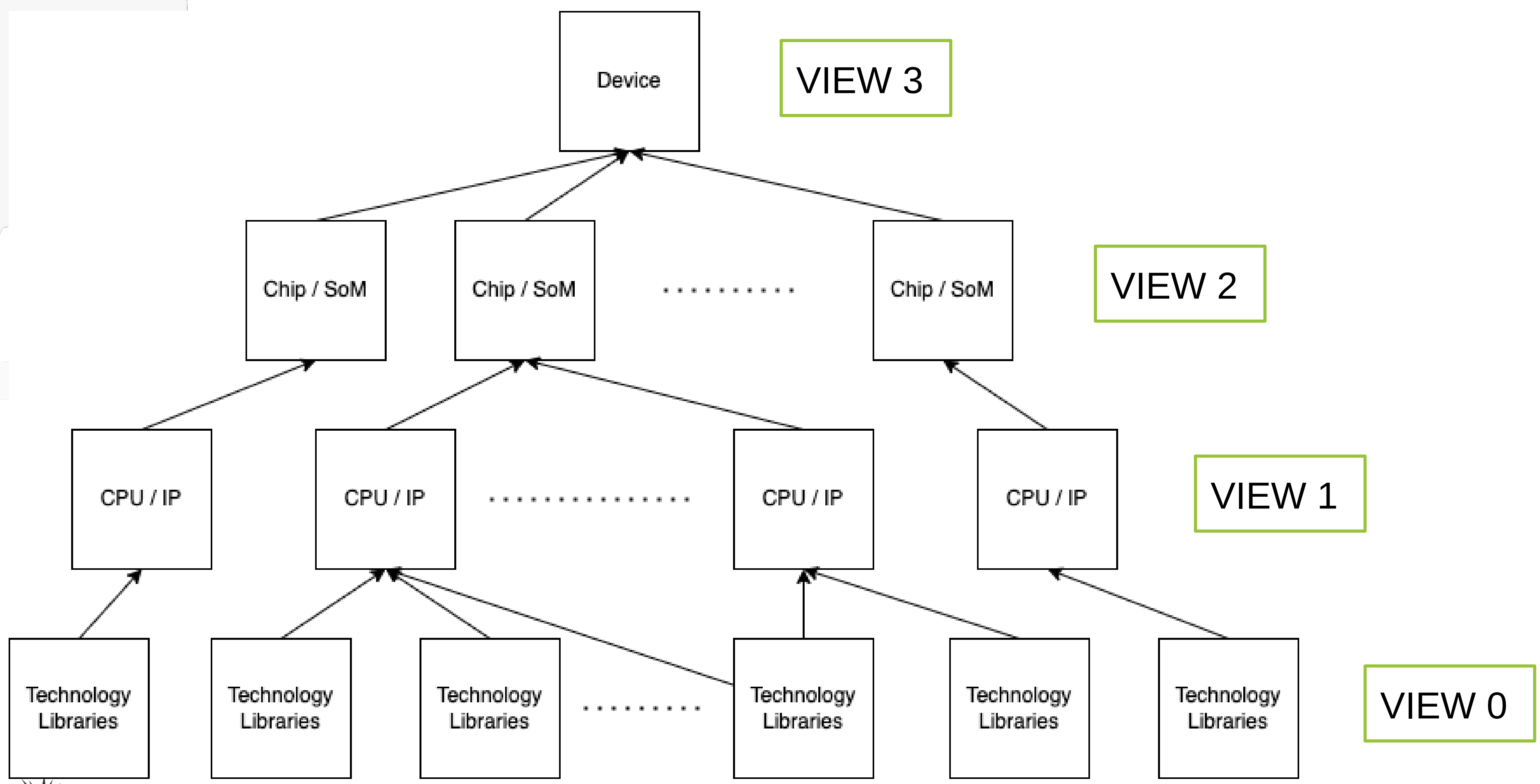
Apply hierarchical and modular design approach in Hardware design (Technology category)

Apply a hierarchical modular approach to design, by recursively divide systems into modules, reuse regular modules when possible, define well-formed interfaces between modules and sub-systems.

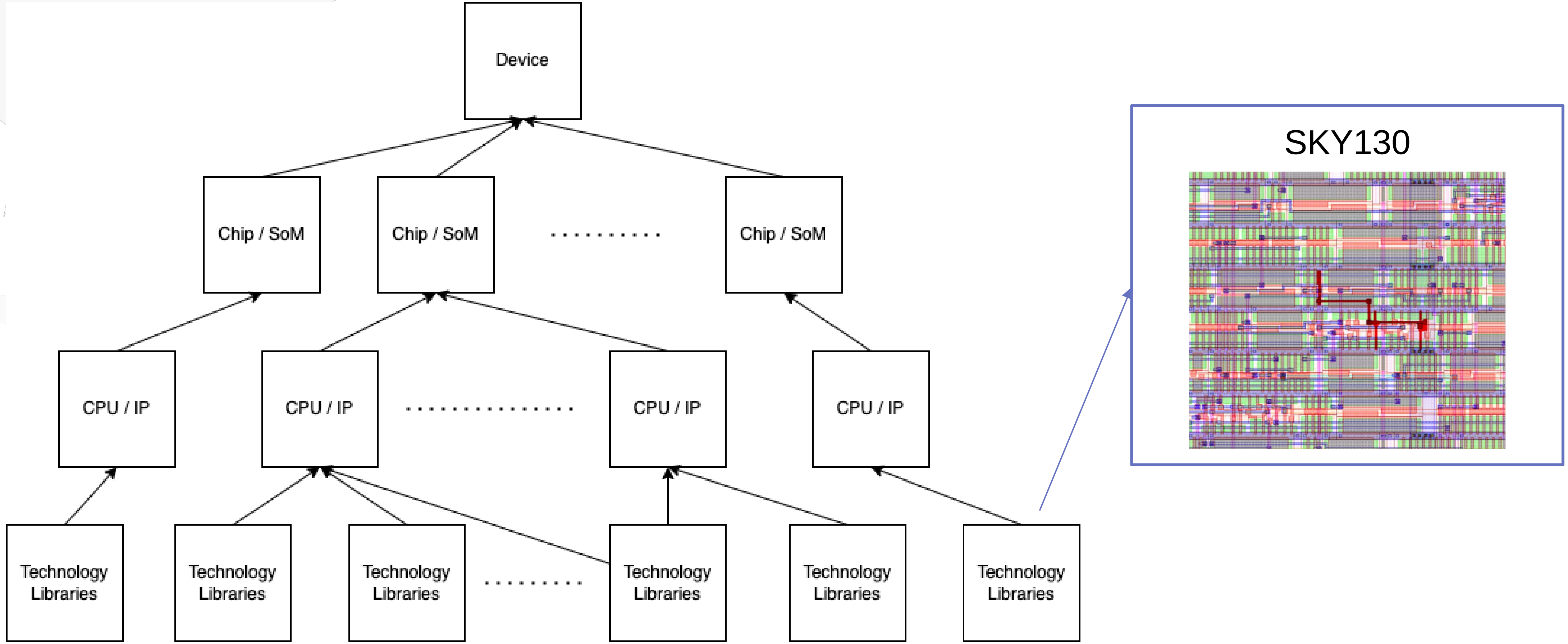
The idea & guiding principles

1. Clarify the **different components** of an open-source hardware
2. Assign level of how much each **component** is open-source separately
   a. Evaluate **properties specific** for the component
   b. Compute a **vector** with these evaluations
   c. Compute the **overall score**
3. Compute the score of the hardware considering the scores of **subcomponents**

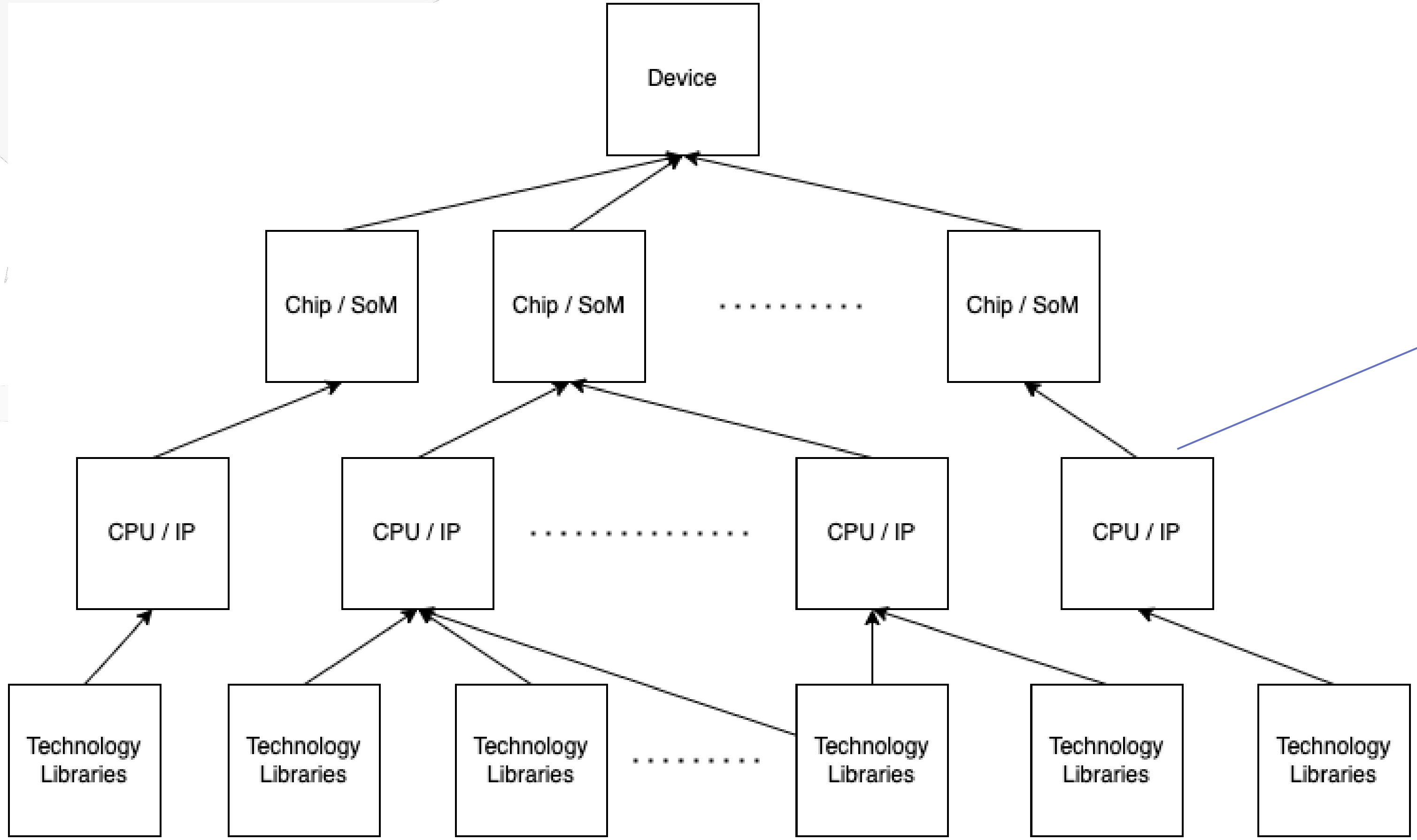
Definition of open source hardware: views

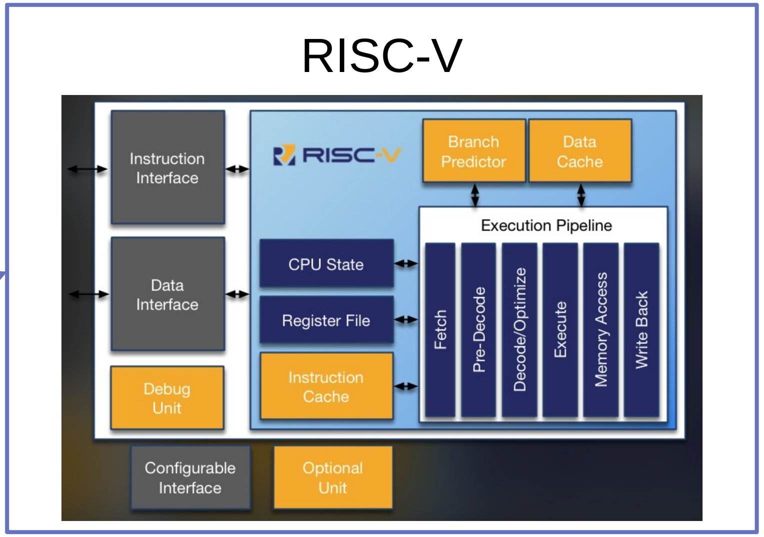


Definition of open source hardware: view 0

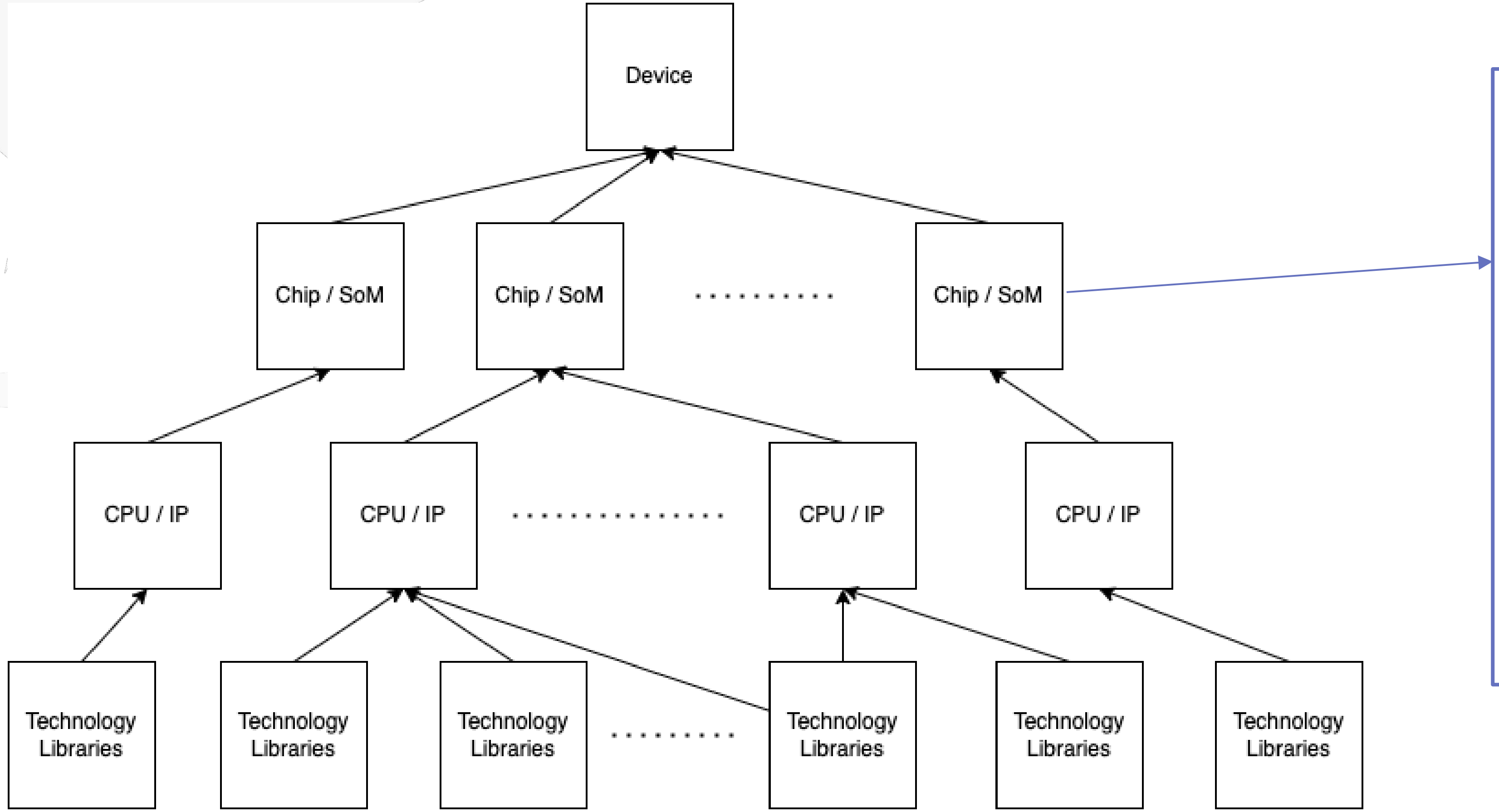


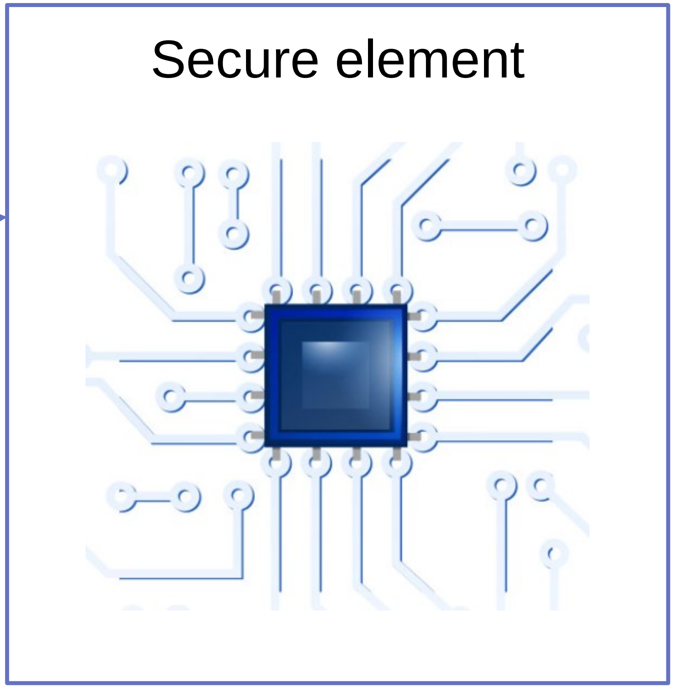
Definition of open source hardware: view 1



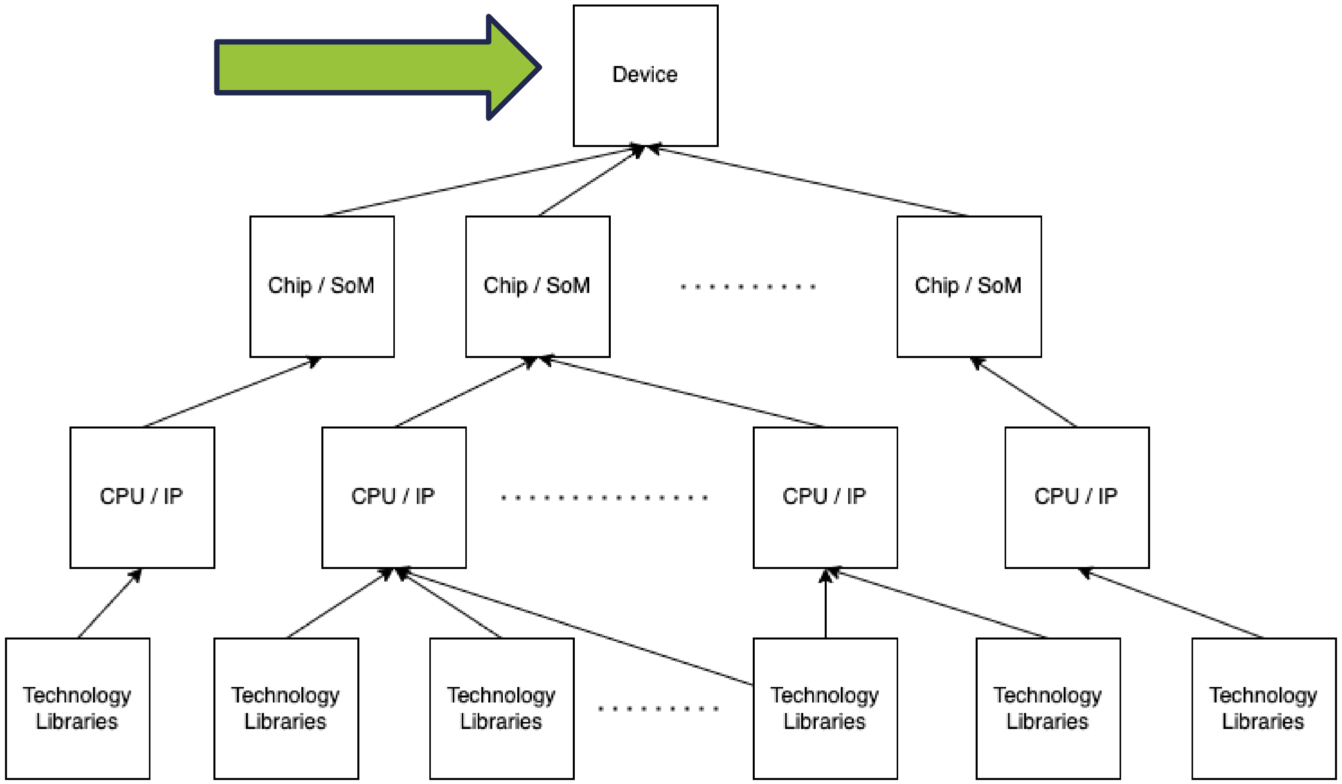


Definition of open source hardware: view 2



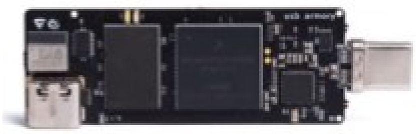


Definition of open source hardware: view 3



Raspberry Pi4 VS USB Armory Scores

RASPEBERRY Pi4	Properties	Score			Final score
COMPONENT	Source code and design files	2	2	2,416666667	2
	Licenses	2			
ECOSYSTEM	Design tools	3	3		
	Toolchain	3			
	Software ecosystem	3			
	Firmware	3			
INFRASTRUCTURE	Processes	1	2,25		
	Replicability	2			
	Documentation	3			
	Example code	3			

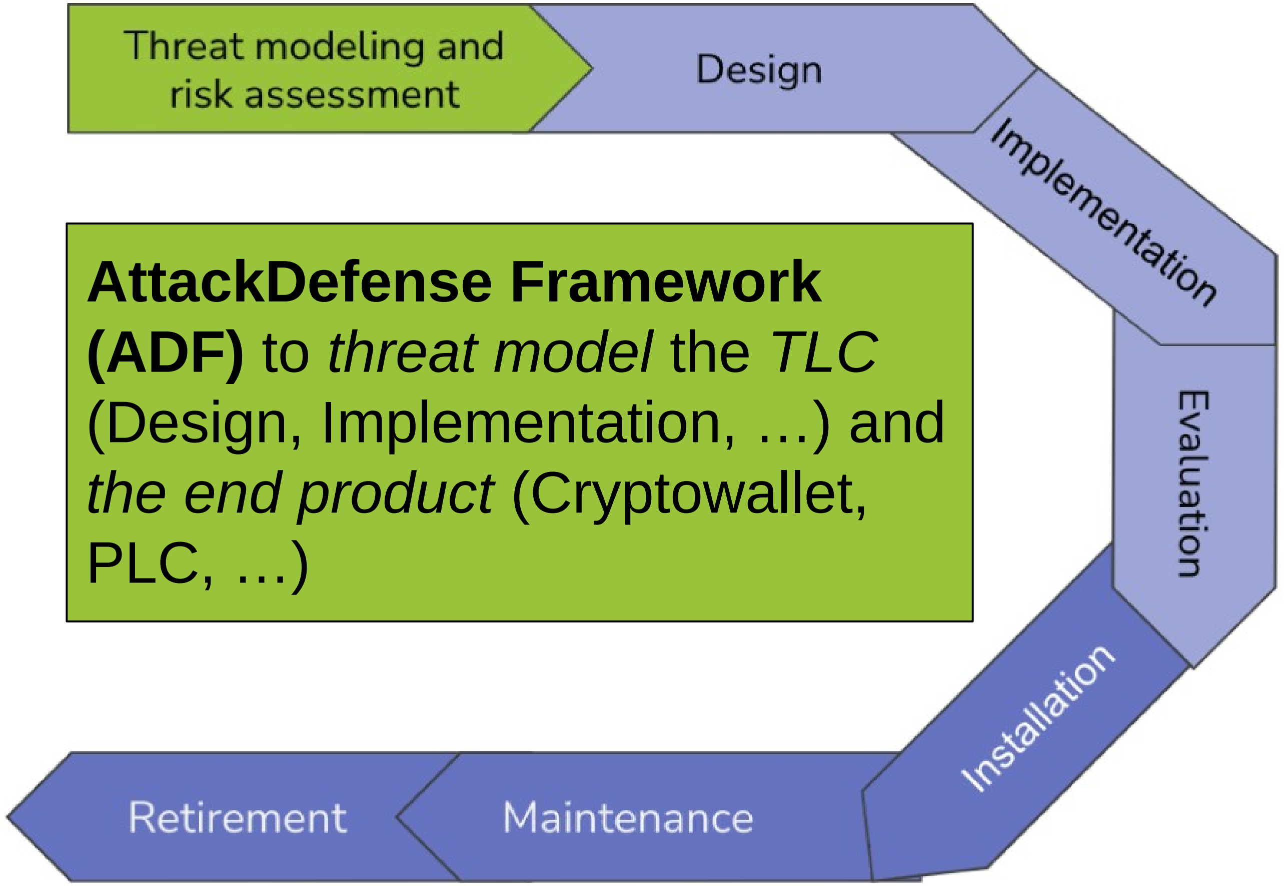


level	description
0	completely closed
1	more closed than open
2	more open than closed
3	completely open

USB Armory	Properties	Score			Final score
COMPONENT	Source code and design files	3	3	3	3
	Licenses	3			
ECOSYSTEM	Design tools	3	3		
	Toolchain	3			
	Software ecosystem	3			
	Firmware	3			
INFRASTRUCTURE	Processes	3	3		
	Replicability	3			
	Documentation	3			
	Example code	3			

Security requirements in the trusted life cycle of OSH

ADF augments ORSHIN's TLC



AttackDefense Framework (ADF) to *threat model* the *TLC* (Design, Implementation, …) and *the end product* (Cryptowallet, PLC, …)

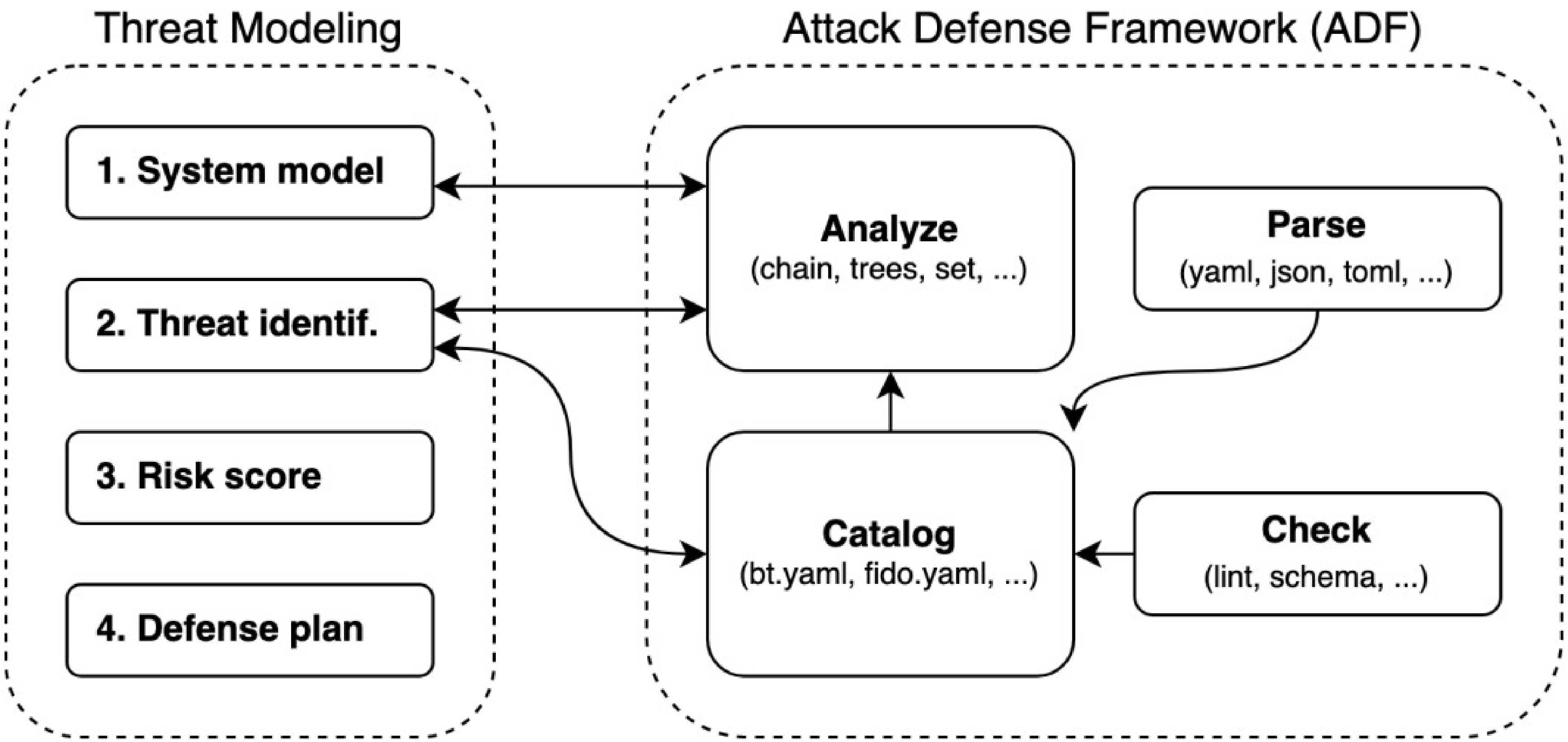
Threat Modeling (TM)

- **System and attacker model**
  - What do we want to protect? (cryptowallet)
  - From whom? (remote or physical threats)
- **Threat identification**
  - What are the possible attacks? (STRIDE, LINDDUN)
- **Risk scoring**
  - How serious they are? (CVSS)
- **Defense plan**
  - How do we mitigate/fix them?

AD Template (YAML)

```yaml
ad_name:
  # Primary fields
  a: attack
  d:
    policy1: [mech1, mech2]
    policy2: [mech1, mech2]
    ...
  surf: [surf, subsurf, subsubsurf, ...]
  vect: [vector1, vector2, ...]
  model: [model1, model2, ...]
  tag: [tag1, tag2, ...]
  # Optional fields
  risk: [score1, score2, ...]
  year: 2023
  cve: ["123", "456", ...]
  cwe: ["123", "456", ...]
  capec: ["123", "456", ...]
  vref: ["vendor-ref1", ...]
  ...: ...
```

ADF Overview



ADF Overview (2)

- **Catalog**
  - YAML files containing AD objects (ADs)
  - AD describes a threat (attack, defenses, attacker model, …)
- **Parse**
  - Extract ADs from YAML, XML, TOML, …
- **Check**
  - Semantic and syntax checks ADs
- **Analyze**
  - Process ADs (set, vector, tree, …)
  - Augment System modeling and threat identification

Generated 175 ADs

TM domain	Sec	Coverage	ADs	Files
ISA/IEC 62443-4-1 SecDev Lifecycle	5.3	LC, SE	40	62443-4-1/*.yaml
Physical Side-Channel and Fault inj.	5.4	PO, HW, SE, FW	20	sc-fi.yaml
Microarch. and Speculative Execution	5.5	PO, HW, SW, SE	14	microa.yaml
Presilicon RISC-V SE Testing	5.6	PO, HW, SW, FW, SE	8	presil.yaml
Invasive Physical IC Attacks	5.7	PO, HW, FW, SE, PR	26	physical.yaml
Bluetooth Protocol and Impl. Attacks	5.8	PO, SW, FW, PT, SE, PR	46	bt.yaml
FIDO2 Authentication Attacks	5.9	PO, HW, SW, FW, PT, SE	21	fido*.yaml

Coverage: **LC**: Lifecycle, **SE**: Security, **PO**: Product, **HW**: Hardware, **FW**: Firmware, **PR**: Privacy, **SW**: Software, **PT**: Protocols

Wireless/IoT AD: KNOB


```yaml
knob_ble:
  a: KNOB entropy downgrade attack on BLE pairing
  d:
    Mutually auth entropy negotiation: [Auth entropy with BLE pairing key]
    High key entropy: [Disallow entropy values lower than 16]
  surf: [BLE, Pairing, Entropy negotiation]
  vect: [Entropy downgrade, Key brute force]
  model: [Proximity, MitM]
  tag: [Protocol, SMP]
  risk: [cvss3_high, cvss2_medium]
  year: 2019
  cve: ["9506"]
  cwe: ["310", "327"]
  capec: ["668"]
```

Hardware AD: Invasive FIB

```yaml
attack_4:
  a: FIB modification
  d:
    Modifying or accessing internal signals should be rendered difficult.:
      - Packing the signals of interest.
  model:
    - invasive
  surf:
    - instruction skip
    - instruction modification
    - execution flow modification
    - counter-measure deactivation
    - read internal signals
  vect:
    - FIB editing
```

Low-level Software AD: Spectre

```
spectre-btb:
  a: Transient execution resulting from mispredicted indirect branches can cause
    persistent changes in the microarchitecture, which can be used to
    intentionally leak secrets from a victim process using a covert channel.
  d:
    "preventing speculation altogether" : [ "Inserting fence instructions at every
     indirect jump", "Disabling speculation in the hardware" ]
    "preventing speculation on secrets" : [ "Implementing a secure speculation
    scheme in the hardware, such as ProSpeCT" ]
    "removing the covert channel": [ "Cache partitioning", "Disabling
    hyperthreading", "(more depending on the microarchitectural side channels)" ]
  surf : [ "Shared resource enabling a covert channel between the victim and the
    attacker", "Shared branch target buffer (BTB) between the victim and the
    attacker" ]
  vect : [ "Controlling a shared resource leading to the covert channel", "
    Poisoning the BTB" ]
  model : [ "code execution", "remote" ]
  tag : [ "transient attack" ]
  year : 2018
  cve : [ "CVE-2017-5753", "CVE-2017-5715" ]
```

Life cycle AD: ISA/IEC 62443-4-1


```yaml
sm_1_dev-proc:
  a: Undefined development/maintenance/support processes
  d:
    Implement config mgmt with change control and audit logging: ["Redmine"]
    Require product desc and reqs def with req traceability:: ["Redmine"]
    Define design practices: [Addressed in @sd-4-secure-design-best-practices]
    Define implementation practices: [Addressed in @si-2-secure-coding-standards]
    Implement repeatable testing and validation processes: [Addressed in @svv-*]
    Enforce review and approval of all development process records: [Addressed in
    @sm-12-process-verification]
    Implement life-cycle support: ["..."]
  surf: [Processes]
  vect: [Unclear definition]
  tag: [Processes, Requirements, Design, Implementation, Testing, Review,
    Vulnerability management, Maintenance]
```

Conclusions

Conclusion

- **TLC**: Flexible process for device manufactures to show the evidences of cybersecurity dimension in projects with Open Source components
- **AttackDefense Framework**: novel framework for threat modelling
  - Machine readable format, connecting & referencing
- **ORSHIN**
  - Project in 2nd half, October 2022 - September 2025
  - Security Pattern, EURECOM lead contributors (TLC & ADF)

ORSHIN Grant Agreement No. 101070008

If you need further information, please contact the coordinator:

TECHNIKON Forschungs- und Planungsgesellschaft mbH

Burgplatz 3a, 9500 Villach, AUSTRIA
Tel: +43 4242 233 55   Fax: +43 4242 233 55 77

E-Mail: coordination@horizon-orshin.eu

Funded by the European Union

Funded by the European Union under grant agreement no. 101070008. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.