27 March 2026

Verifying Secure Memory Compartmentalization in CHERI Processors at the RTL

Johannes Müller
RPTU Kaiserslautern-Landau

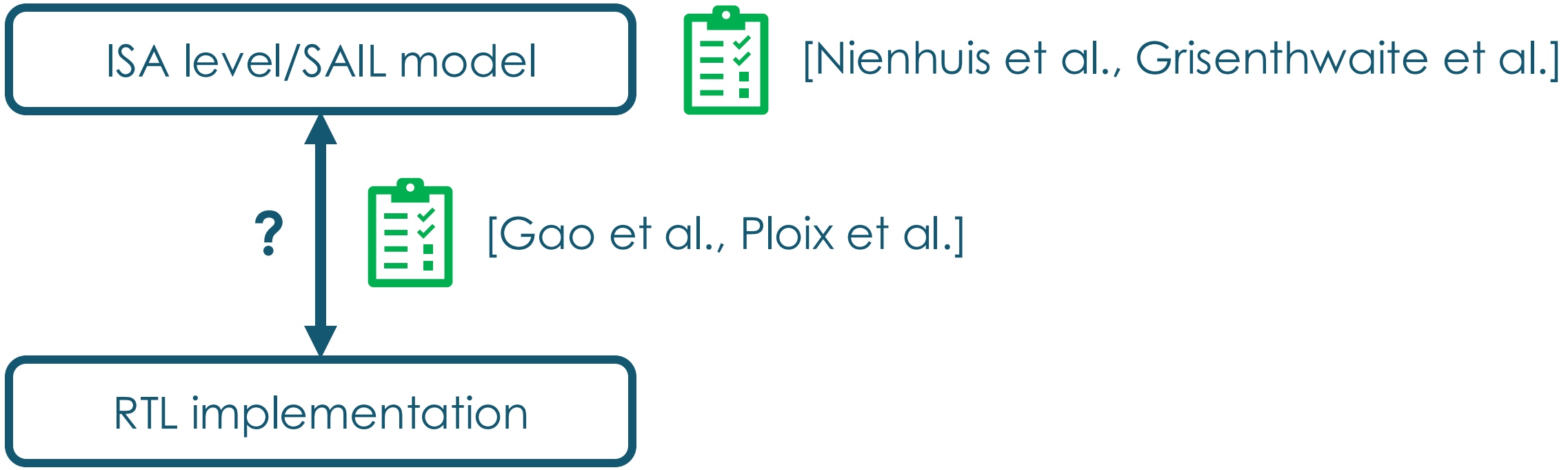
Motivation

- Security features in HW are the **foundation** of security for **entire system stack**
- CHERI offers **sophisticated** HW security features

→ **HW verification** of high importance

→ **Security guarantees** desirable

Related Verification Approaches



Related Verification Approaches

Pitfalls:

- High manual effort or proof runtimes
- Security vulnerabilities of the microarchitecture may be missed

Our Approach

~~ISA level/SAIL model~~

Our Approach

Benefits:

- **Exhaustive proofs** w.r.t. security goals at the RTL

  → Covering vulnerabilities in the microarchitecture

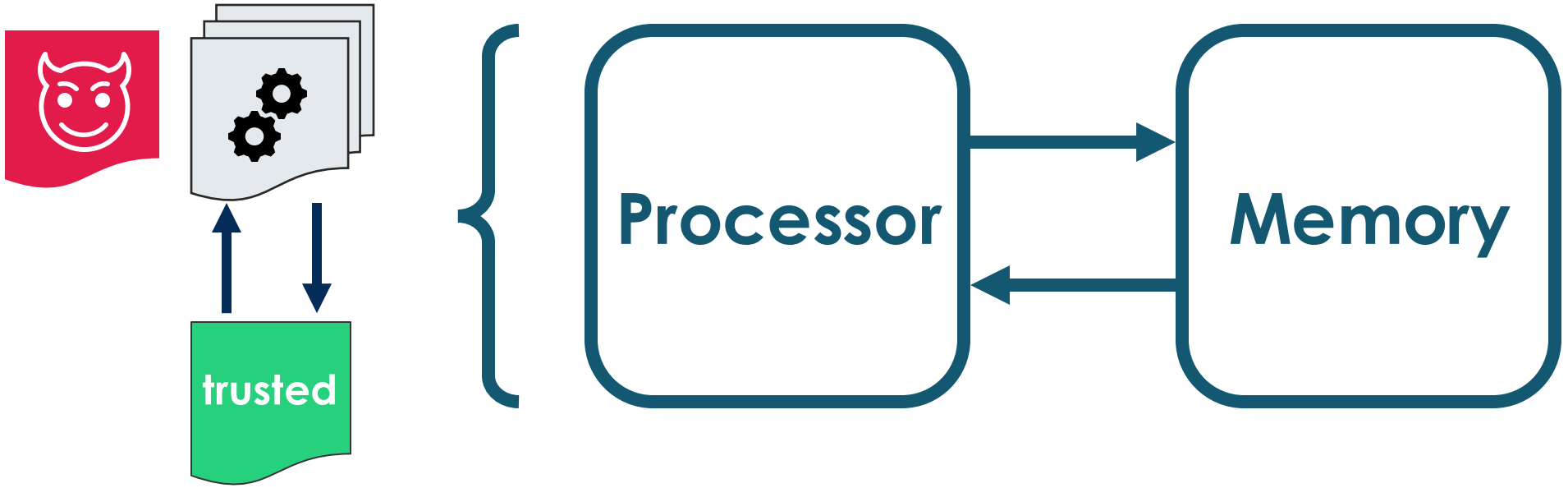
- **Scalable proofs** by focusing on security goals

Security Goal

Our Focus:

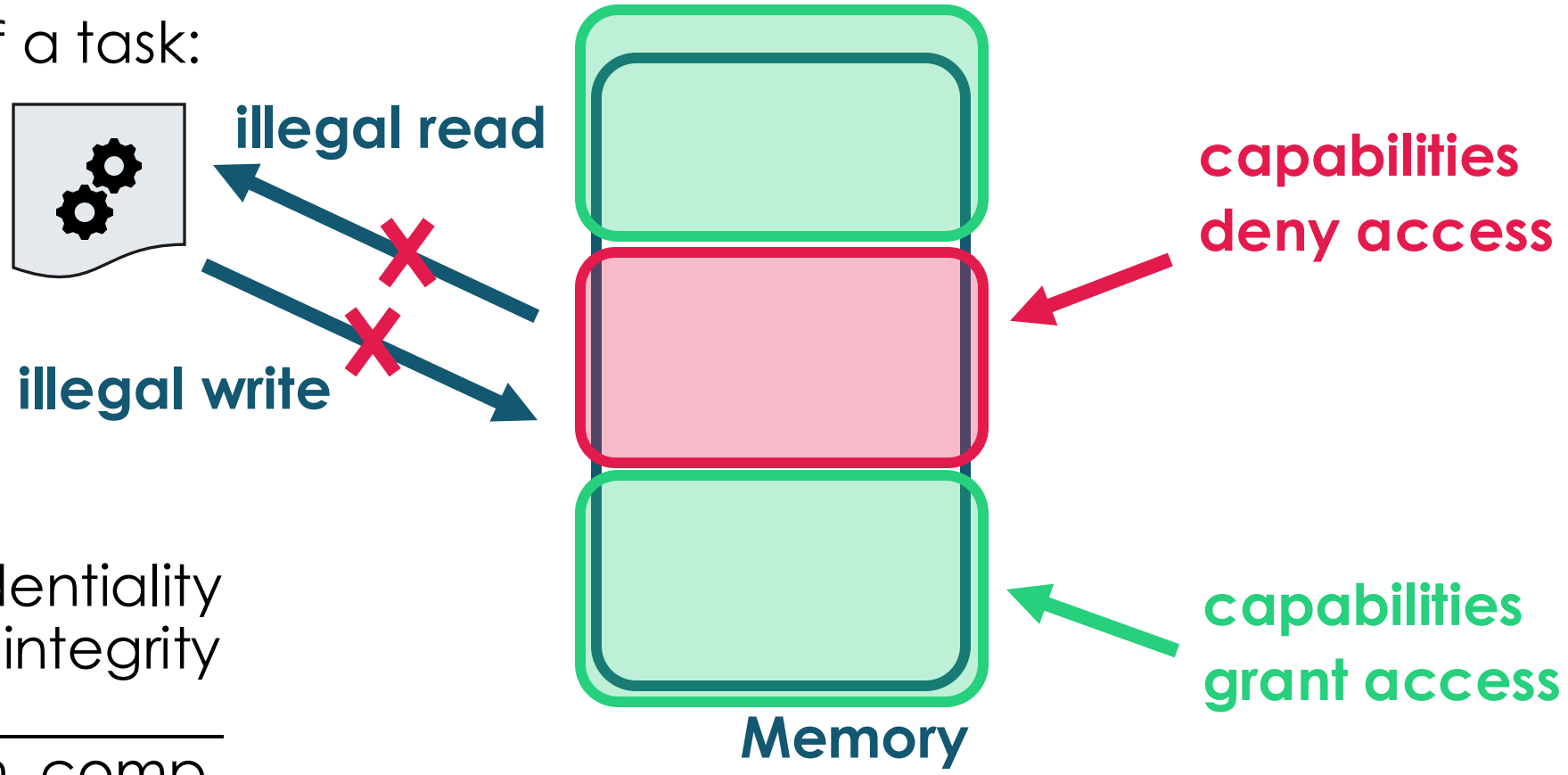
Secure memory compartmentalization

Secure Memory Compartmentalization



Secure Memory Compartmentalization

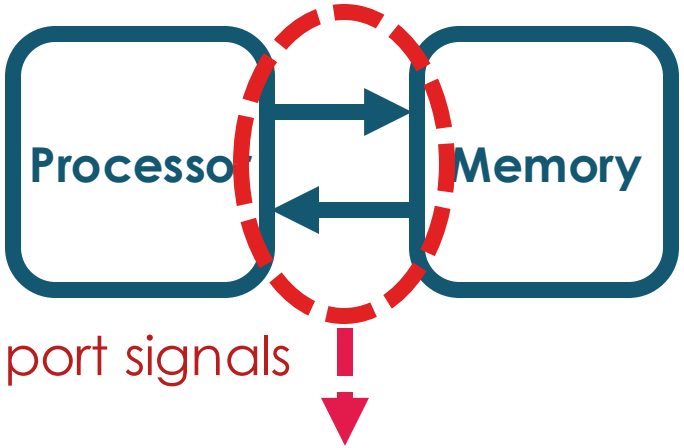
Perspective of a task:

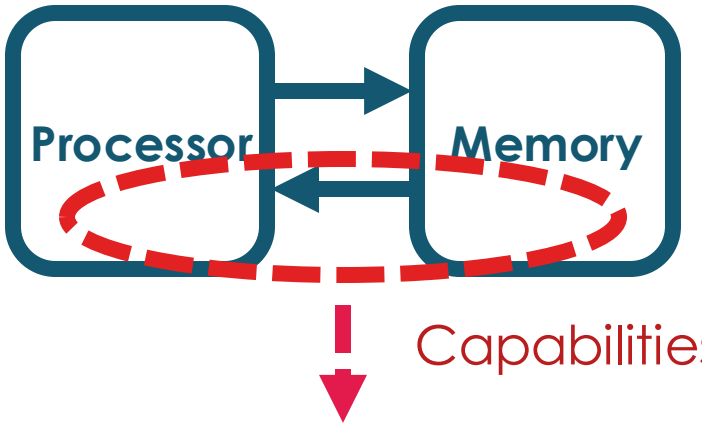


data confidentiality
\+ data integrity

secure mem. comp.

Security Properties

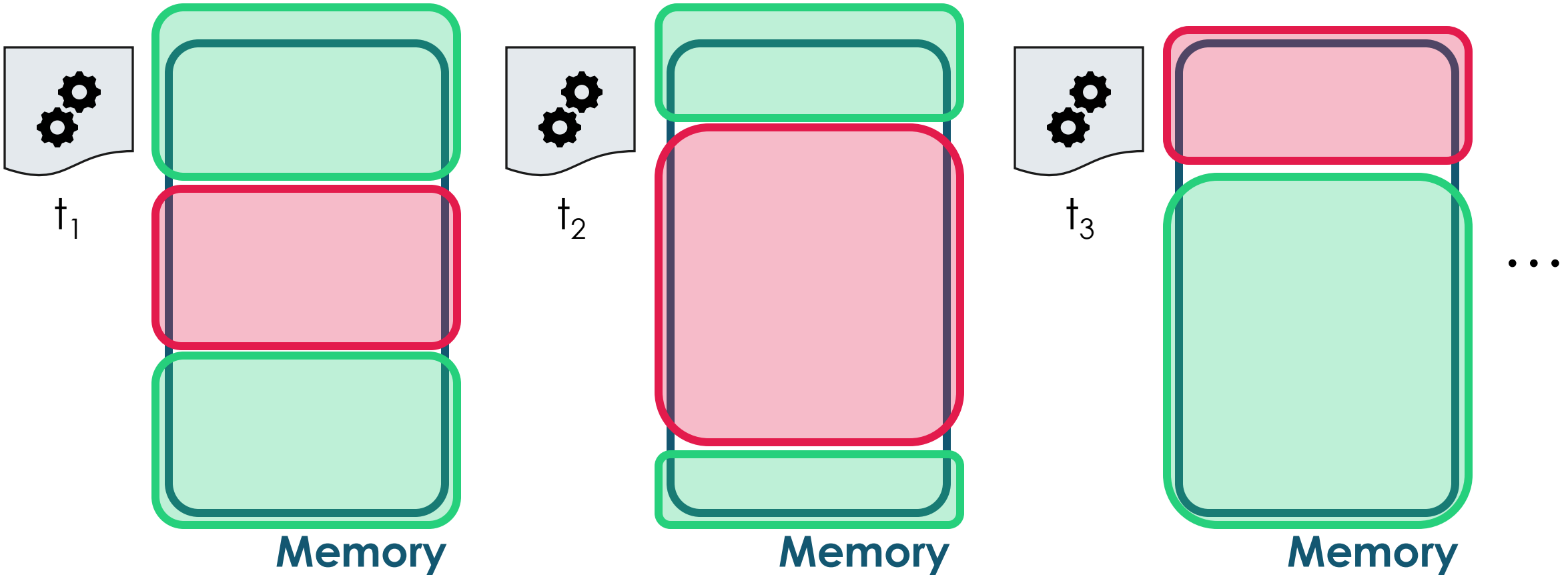




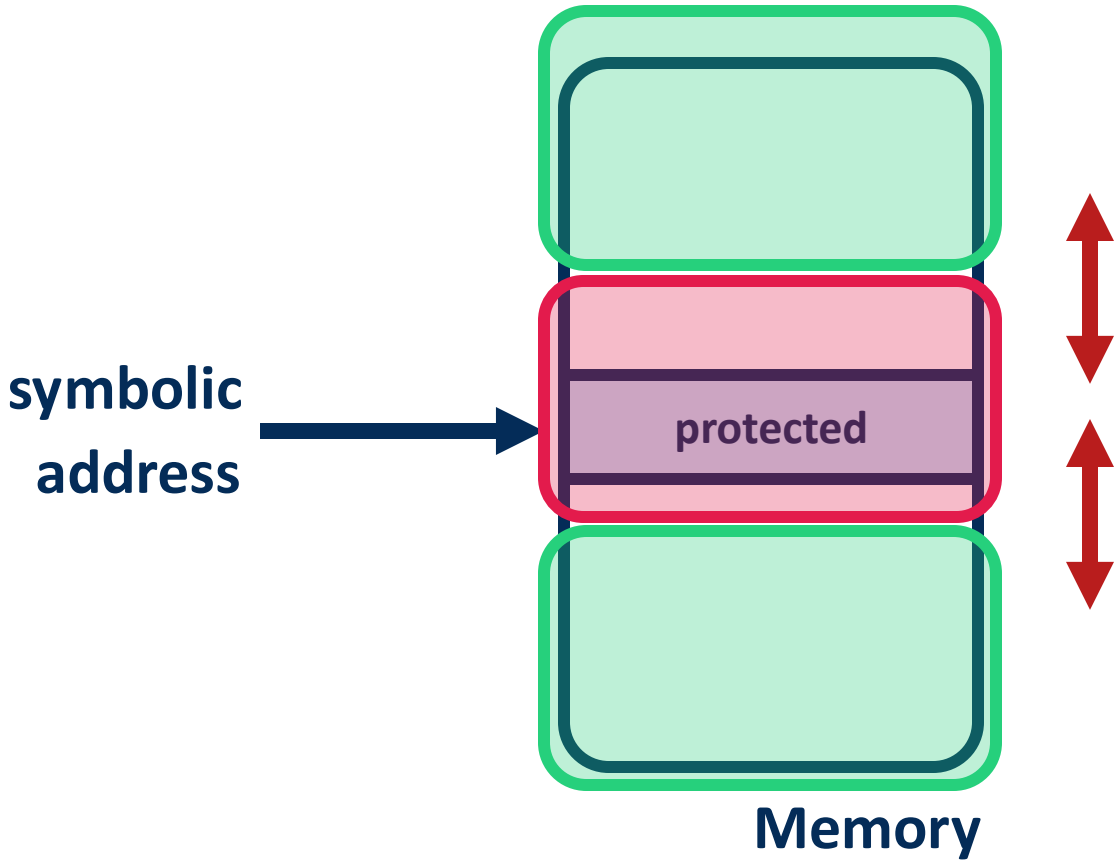
Confidentiality: `task_executing(t) → not(read_access() and protected(data_addr))`

Integrity: `task_executing(t) → not(write_access() and protected(data_addr))`

Challenge 1: One Property per Task



Solution: Symbolic Capabilities



Capabilities are freely chosen by the solver

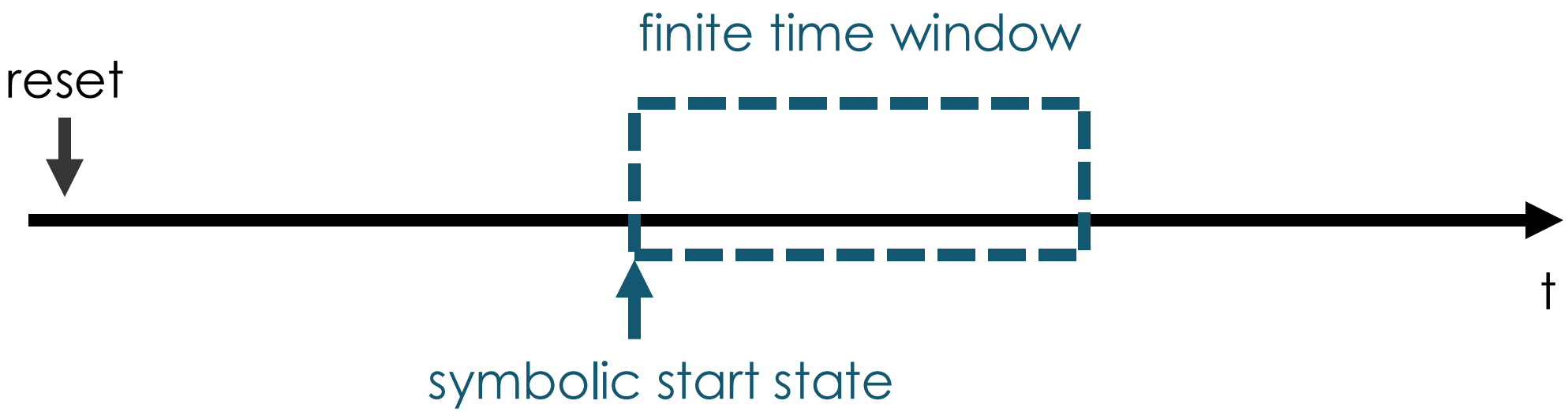
except:

symbolic address must be **protected**

Challenge 2: Scalable Properties

How can we obtain properties that are both, **scalable** and **exhaustive** w.r.t. our security goal?

Solution: Interval Properties



- Design is unrolled to match time window
- SAT-based property checking → commercial tools available

Interval Properties

Confidentiality interval property:

```
t   : protected(symbolic_addr)
implies
t+1: not(read_access() and
         data_addr == symbolic_addr)
```

Integrity interval property:

```
t   : protected(symbolic_addr)
implies
t+1: not(write_access() and
         data_addr == symbolic_addr)
```

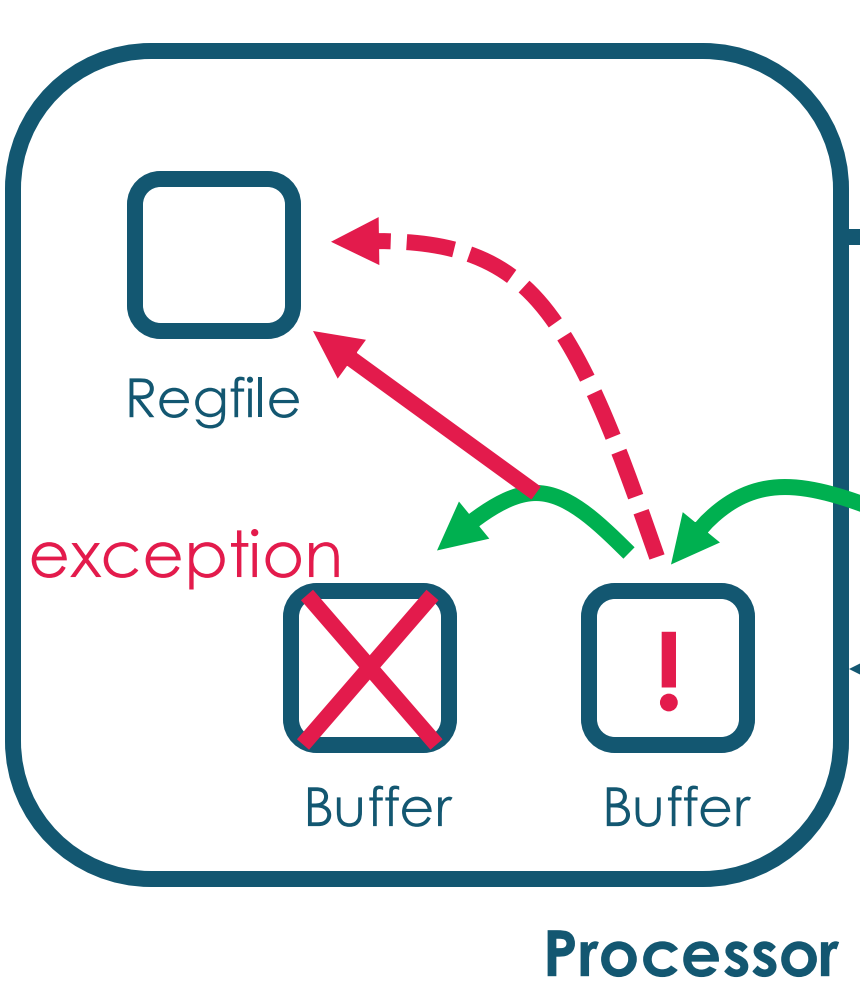
Monotonicity Invariant

Monotonicity interval property:

```
t   : protected(symbolic_addr)
implies
t+1: protected(symbolic_addr) or
     task_exit()
```

Task cannot increase access rights

Problem with Confidentiality Property

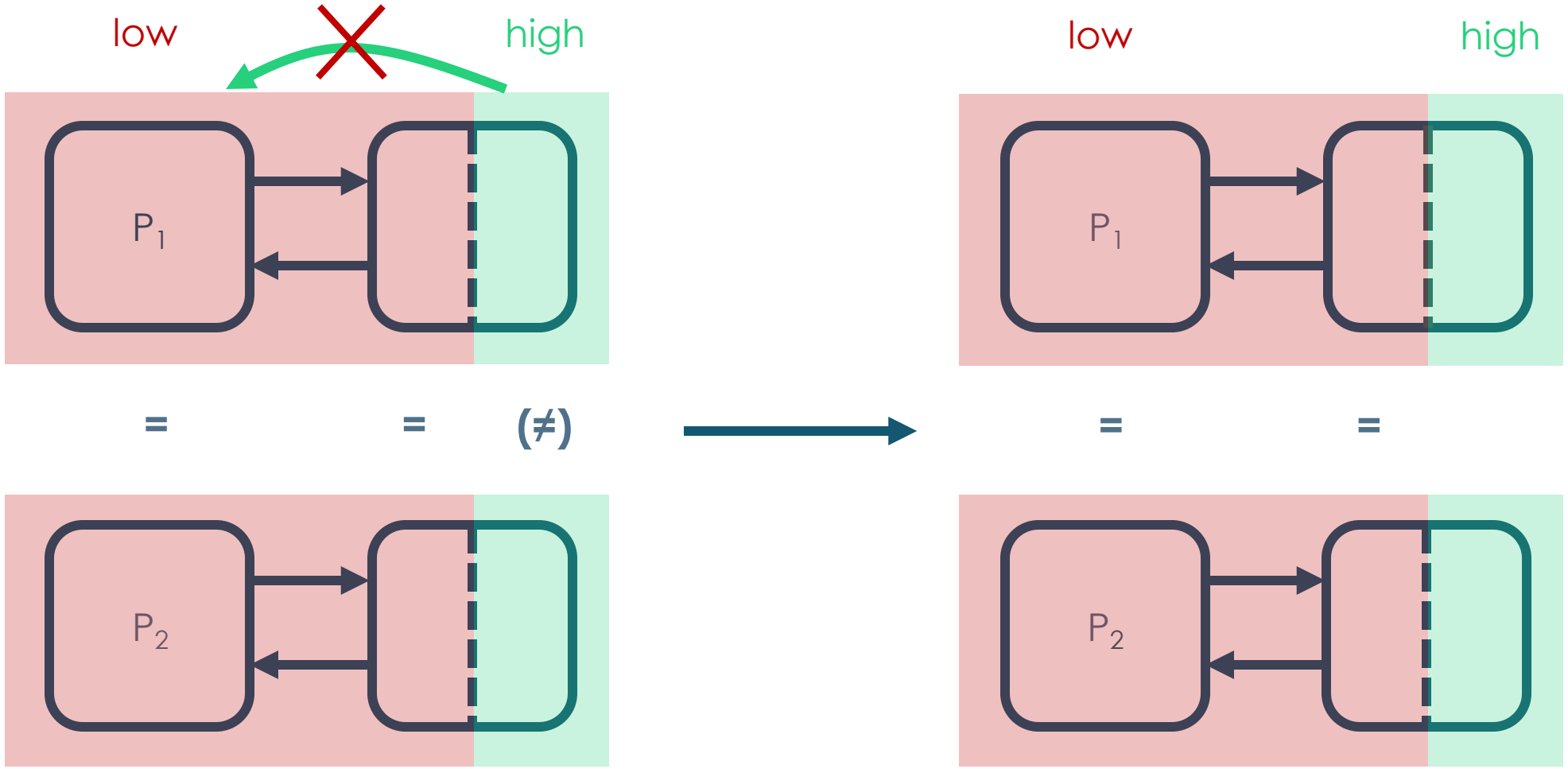


architectural vs. **microarchitectural** state variables

illegal memory access

exception is triggered

Noninterference



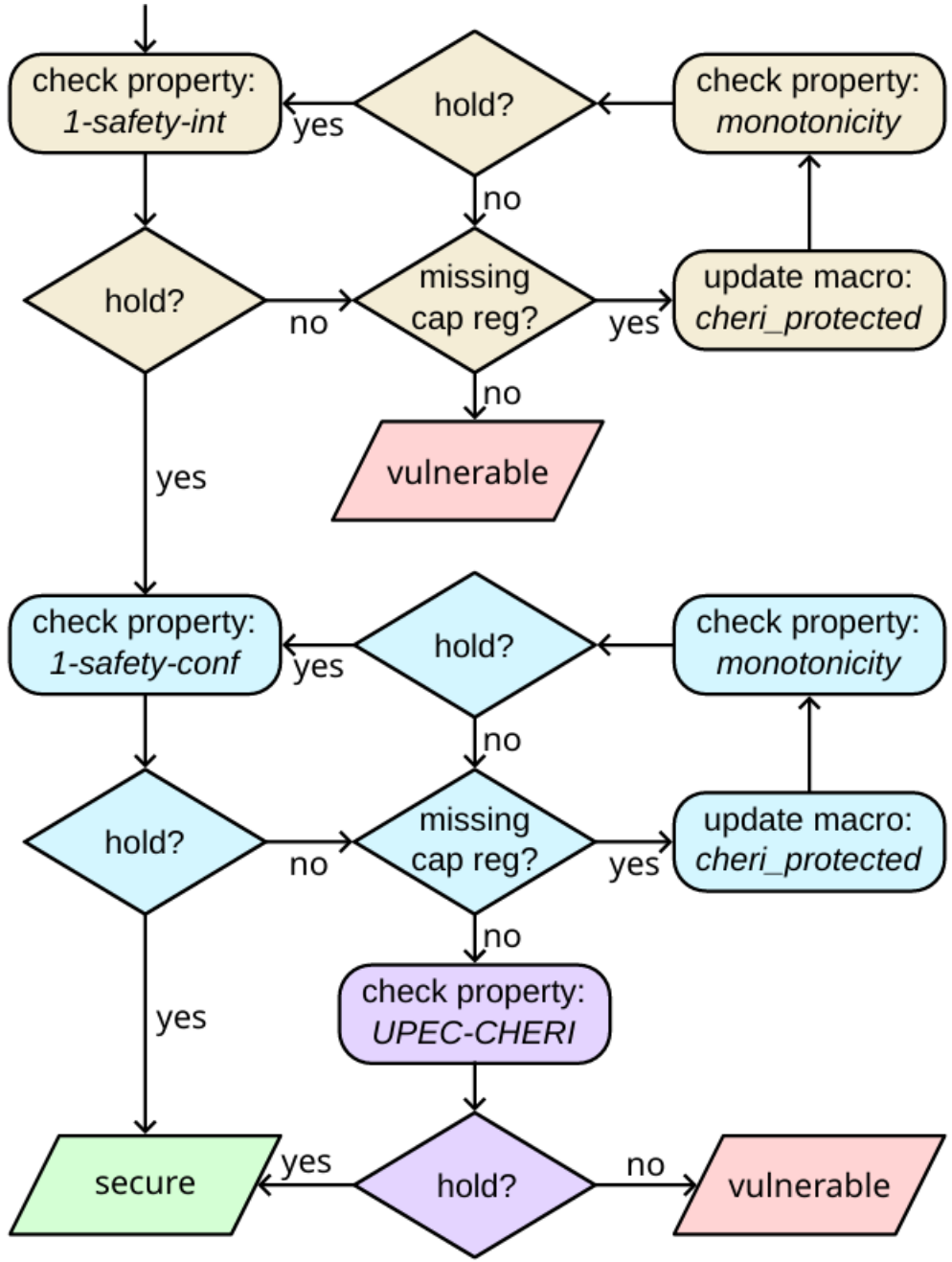
2-safety Interval Property

2-safety interval property:

```
t … t+n   : protected(symbolic_addr)
t … t+n   : data_addr != symbolic_addr
            → data_rdata1 == data_rdata2
t         : marchstate1 == marchstate2
implies
t+1 … t+n: archstate1 == archstate2
```

Similar to our previous work on UPEC

VeriCHERI


check property:
1-safety-int
yes
hold?
check property:
monotonicity
no
hold?
no
missing
cap reg?
yes
update macro:
cheri_protected
no
vulnerable
yes
check property:
1-safety-conf
yes
hold?
check property:
monotonicity
no
hold?
no
missing
cap reg?
yes
update macro:
cheri_protected
no
check property:
UPEC-CHERI
yes
secure
yes
hold?
no
vulnerable

Case Study: CHERIoT-Ibex

Property	Iteration	Result	Runtime	Memory	Description
1-safety-integrity	1	fail	< 1 min	4.3 GB	*Bug*: setup guide specification of protection enable pin
	2	fail	< 1 min	4.7 GB	*Bug*: capability stores across capability bounds
	3	hold	7 min	4.8 GB	-
Monotonicity	1-9	fail	≤ 1 min	4-5 GB	Missing capability register or pipeline buffer
	10	hold	15 min	6.2 GB	-
1-safety-confidentiality					
⟶ data	1	hold	7 min	7.3 GB	-
⟶ instructions	1	fail	< 1 min	4.8 GB	Instruction fetched from outside PCC bounds
UPEC-CHERI	1	fail	31 min	3.7 GB	*Side channel*: exception timing depends on fetched data
	2	hold	18 min	6.3 GB	-

Case Study CHERIoT-Ibex

We detected a vulnerability to a potential **Transient Execution Attack**:

- Branch to address **outside of PCC bounds**
- Illegal instruction fetch **raises an exception**
- Exception execution is delayed **depending on two bits** of the fetched data

→ By measuring the (overall) execution time, or reading the performance counter an attacker **can probe the two bits for an arbitrary protected address**

Open-Source Verification IP

- Open-Source verification IP at GitHub
- Cookbook for refining properties for a CHERI processor
- Two example refinements

GitHub Repo:

CHERI

THANK YOU

ICCAD'24 Paper

GitHub Repo

Contact johannes.mueller@rptu.de