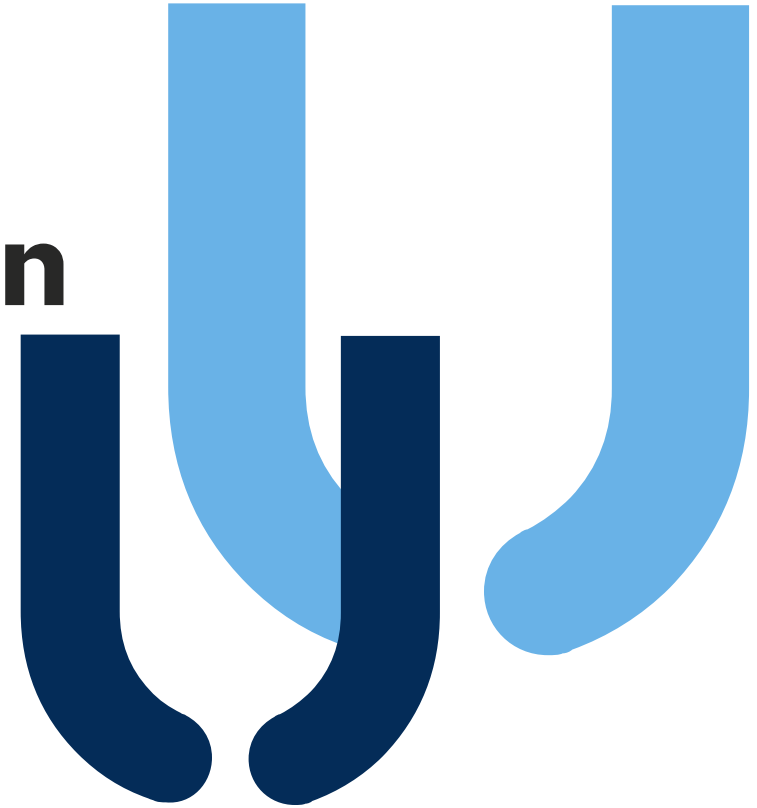
VeriCHERI: Exhaustive Formal Security Verification of CHERI at the RTL



Verification Futures Conference UK - 2025

Speaker: Johannes Müller

Joint work with: Anna Duque Antón, Philipp Schmitz, Tobias Jauch, Alex Wezel, Lucas Deutschmann, Mo Fadiheh, Dominik Stoffel and Wolfgang Kunz

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Motivation

> Memory protection mechanisms are a key ingredient for security of the entire system stack.

> Capability Hardware Enhanced RISC Instructions (CHERI) is a promising candidate.

> CHERI is gaining traction in industry:

  > ARM taped out a CHERI enhanced processor “Morello”

  > Microsoft actively develops CHERI enhanced IoT processor “CHERIoT Ibex”

  > Codasip includes CHERI in RISC-V processor

  > Open-Source RISC-V processor designs implementing CHERI are available

→ Security verification of CHERI designs is needed,
but creating trust for the entire system stack is challenging.

Motivation

Related verification approaches:

> Verified security properties on a formal ISA model with an effort of 3-4 person years [1], [2]

> Manually translated a subset of the formal ISA model into SVA properties and verified them on the HW [3]

> Introduced a reference model for the formal ISA model in RTL and verified observational correctness of the HW [4]

→ Security verification based on a time-abstract ISA model misses non-functional security vulnerabilities like timing side channels.

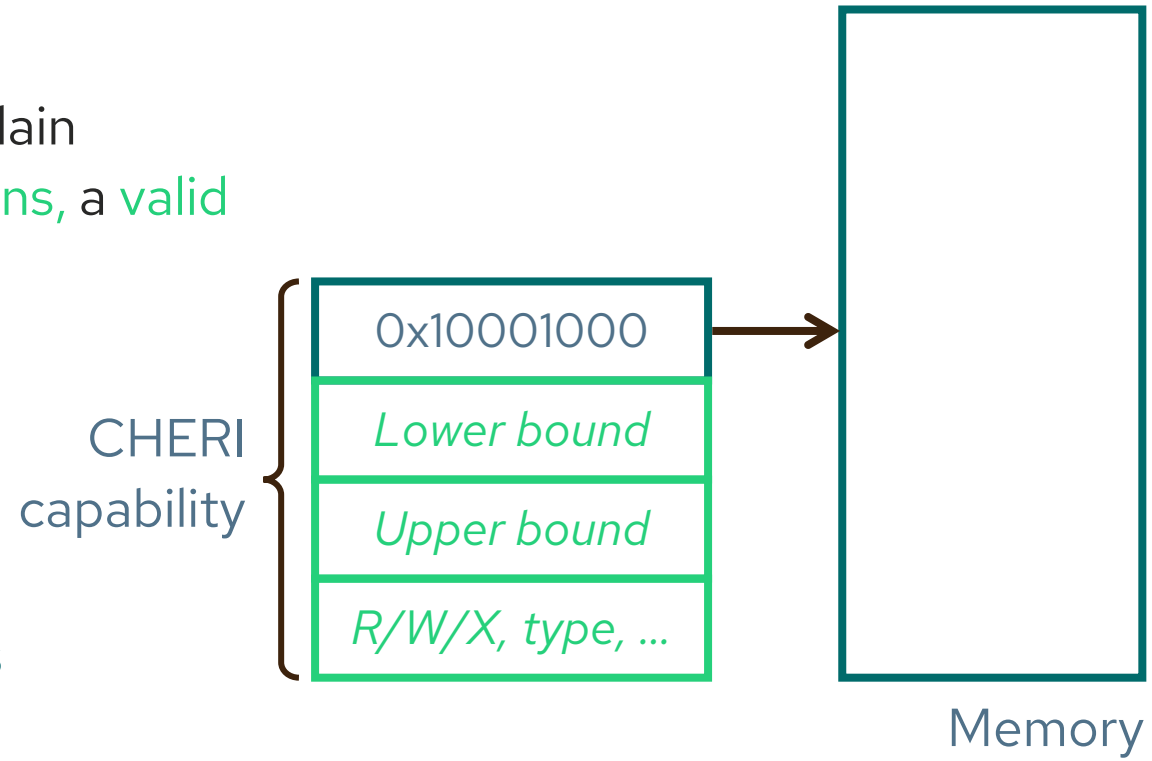
Our goal:

Prove global security objectives (confidentiality, integrity) on the timing-accurate RTL implementation.

[1] Grisenthwaite et al., The Arm Morello Evaluation Platform–Validating CHERI-Based Security in a High-Performance System, 2023, IEEE Micro
[2] Nienhuis et al., Rigorous engineering for hardware security: Formal modelling and proof in the CHERI design and implementation process, 2020, IEEE S&P
[3] Gao et al., End-to-end formal verification of a risc-v processor extended with capability pointers, 2021, IEEE FMCAD
[4] Ploix et al., Comprehensive Formal Verification of Observational Correctness for the CHERIoT-Ibex Processor, 2025, arxiv preprint

CHERI Protection

- CHERI enhances RISC ISAs with fine-grained memory protection implemented in HW
- Capabilities extend classical address pointers (plain integers) with address bounds, access permissions, a valid tag and an object type
- Legal memory accesses require
  - valid capabilities that hold
  - corresponding permissions and
  - whose bounds include the accessed address
  - ...



CHERI Protection

> ISA is extended by additional instructions for manipulating capabilities.

> Capability manipulations also require valid capabilities with the correct permissions

> Capability Monotonicity: new capabilities must not exceed the access rights

→ HW needs to accommodate for

  > Additional CHERI instructions and

  > Access control checks, and for

  > Storing capabilities in the register file and memory

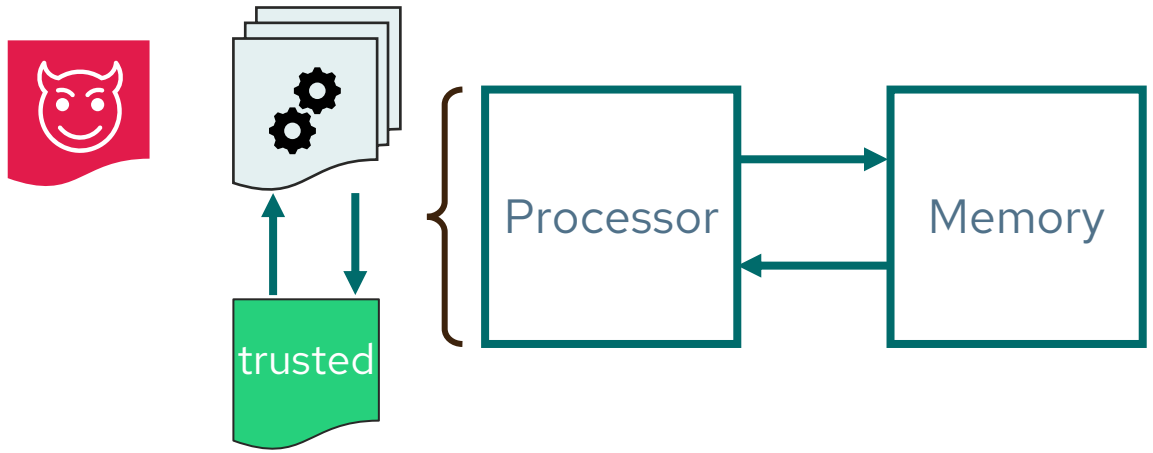
Attacker Model

We assume:

> A capability-enhanced single-core processor with memory for instructions and data.

> The processor executes mutually distrusting tasks including an attacker task and

> A trusted entity (e.g., OS or hypervisor) to securely manage context switches between tasks.



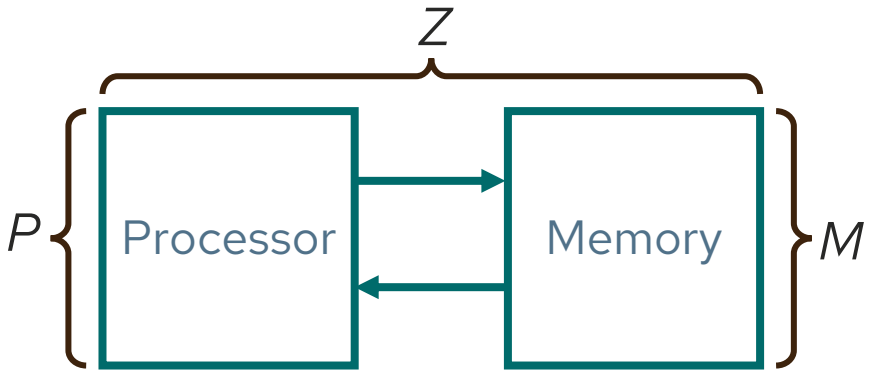
Formal Model: Modeling HW

> HW consists of a processor and memory modeled as FSM:
$(S, S_0, I, O, \lambda, \sigma)$

> FSM states in S are valuations to state variables (state-holding RTL signals). We denote by *Z* the set of all state variables. *Z* comprises:

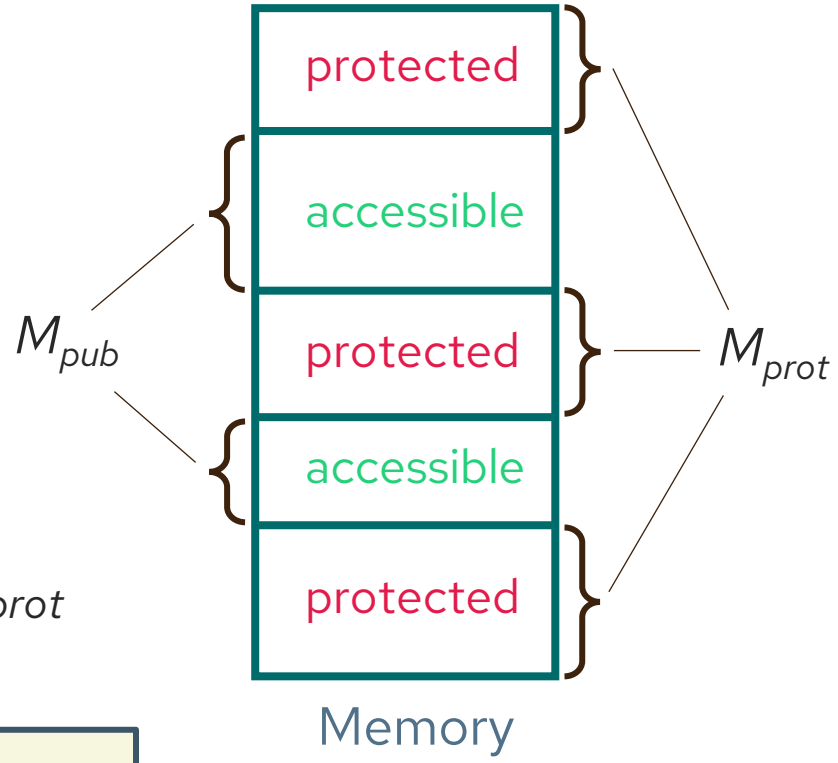
  > *P* : all state variables belonging to the processor

  > *M*: all state variables belonging to the memory



Formal Model: Modeling HW

> For any task running on the processor, M can be divided into:
  > M_{pub} : all state variables in the memory accessible to the current task
  > M_{prot}: all state variables in the memory protected from the current task
> Compartmentalization of M into M_{pub} and M_{prot} is enforced by CHERI capabilities



In our model, two tasks only differ in the compartmentalization of M into M_{pub} and M_{prot}

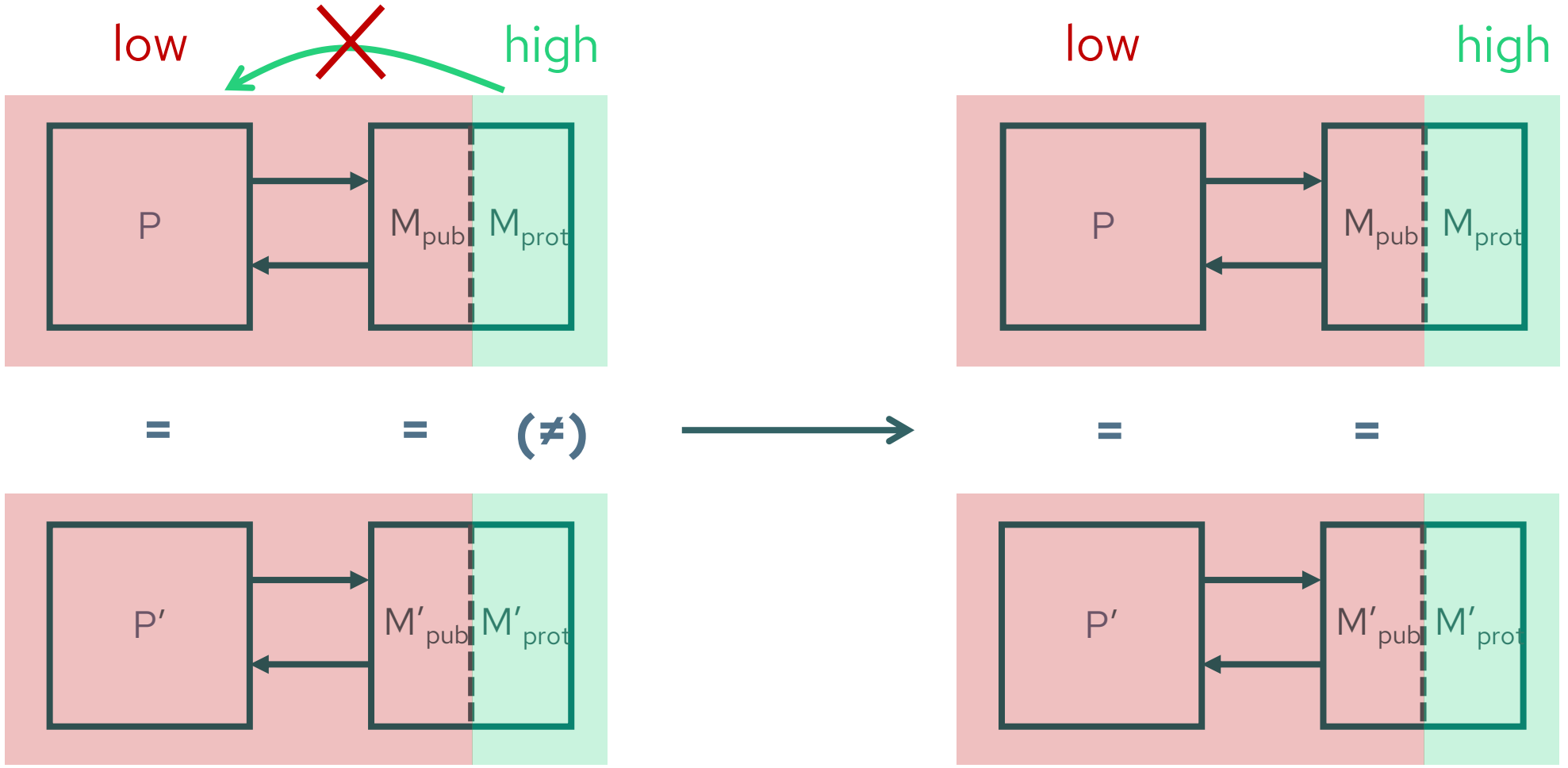
Formal Model: Security Objective

Our goal:
Prove global security objectives (confidentiality, integrity) on the timing-accurate RTL implementation.

We model confidentiality and integrity objectives using non-interference.

- Strong formulation of security
- Well known and widely adapted

Formal Model: Non-interference



Formal Model: CTL properties

Confidentiality non-interference CTL property for our threat model:

$$\mathrm{AG}(\ \$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow \mathrm{AG}(\$M_{pub} = \$M'_{pub} \wedge \$P = \$P')\ )$$

Integrity non-interference CTL property for our threat model:

$$\mathrm{AG}(\ \$M_{prot} = \$M'_{prot} \rightarrow \mathrm{AG}(\$M_{prot} = \$M'_{prot})\ )$$

Notations:

$\$M$: set of values assigned to variables in M

M': state set M from second instance of same design

Formal Model: Challenges

Proving the two CTL non-interference properties poses two main challenges:

1. Modeling M_{pub} and M_{prot} for RTL designs, i.e.,
   dividing of memory into public and protected locations in terms of CHERI capabilities

2. Covering all possible tasks, i.e.,
   modeling all possible divisions of memory locations into M_{pub} and M_{prot}

Formal Model: Reduction to 1-safety

Our key ideas to overcome these challenges:

> We model M_{pub} and M_{prot} by their addresses:
Violation of CTL properties is only possible if processor makes a memory access to M_{prot}

Confidentiality non-interference CTL-property:

$$\mathrm{AG}(\ \$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow\ \mathrm{AG}(\$M_{pub} = \$M'_{pub} \wedge \$P = \$P')\ )$$

→ Memory read access to M_{prot}

Integrity non-interference CTL-property:

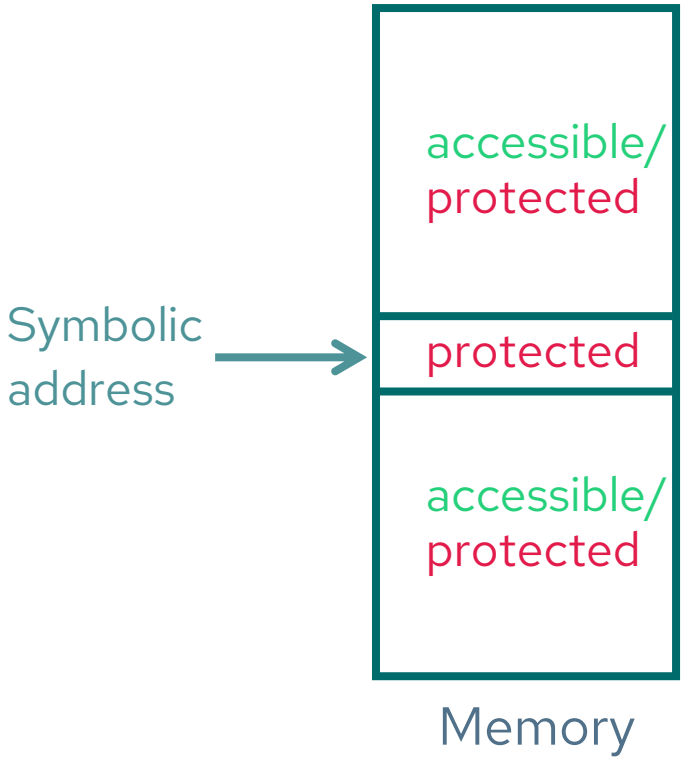
$$\mathrm{AG}(\ \$M_{prot} = \$M'_{prot} \rightarrow\ \mathrm{AG}(\$M_{prot} = \$M'_{prot})\ )$$

→ Memory write access to M_{prot}

Formal Model: Modeling HW

- We introduce a new symbolic address that can be chosen freely by the solver.
- We introduce a new macro cheri_protected(symbolic_addr), which constrains all capabilities available to an attacker task
  - Constrained capabilities are fully symbolic, except
  - They deny access to the symbolic address.

→ This is how we model all possible divisions into M_{pub} and M_{prot} and thus all possible attacker tasks.



Formal Model: Reduction to 1-safety

Confidentiality 1-safety property:

$$\mathrm{AG}(\ \mathrm{cheri\_protected}(\mathrm{symbolic\_addr}) \rightarrow (\mathrm{read\_mem\_access} \rightarrow \mathrm{mem\_addr} \neq \mathrm{symbolic\_addr})\ )$$

Confidentiality interval property:

```
t : cheri_protected(symbolic_addr)
implies
t: !read_mem || mem_addr != symbolic_addr
```

The property describes behavior in a single clock cycle

→ Scalable proofs

Formal Model: Reduction to 1-safety

Integrity 1-safety property:

$$\mathrm{AG}(\ \mathrm{cheri\_protected}(\mathrm{symbolic\_addr}) \rightarrow (\mathrm{write\_mem\_access} \rightarrow \mathrm{mem\_addr} \neq \mathrm{symbolic\_addr})\ )$$

Integrity interval property:

```
t : cheri_protected(symbolic_addr)
implies
t: !write_mem || mem_addr != symbolic_addr
```

Formal Model: Proving Monotonicity

Integrity interval property:

```
t : cheri_protected(symbolic_addr)
implies
t: !write_mem ||
    mem_addr != symbolic_addr
```

Capability Monotonicity is a fundamental concept and invariant used in CHERI:

> *Access permissions can only ever decrease*.

Monotonicity interval property:

```
t  : cheri_protected(symbolic_addr)
implies
t+1: cheri_protected(symbolic_addr)
```

Formal Model: UPEC-CHERI

Confidentiality non-interference CTL property:

$$\mathrm{AG}(\ \$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow \mathrm{AG}(\$M_{pub} = \$M'_{pub} \wedge \$P = \$P')\ )$$

What if the confidentiality 1-safety property fails?

> Protected data propagates to state variables of the processor P

> But: not all state variables in the processor are visible to an attacker task

> We define the subset:

  > $P_{arch} \subseteq P$: all architectural state variables in the processor, i.e., state variables visible to an attacker task

Formal Model: UPEC-CHERI

Confidentiality non-interference CTL property:

$$\mathrm{AG}(\ \$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow \mathrm{AG}(\$M_{pub} = \$M'_{pub} \wedge \$P = \$P')\ )$$

Less conservative non-interference CTL-property for confidentiality:

$$\mathrm{AG}(\$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow \mathrm{AG}\ (\$P_{arch} = \$P'_{arch})\ )$$

→ Reformulation of UPEC for our threat model

Formal Model: UPEC-CHERI

UPEC-CHERI interval property:

```
t   : cheri_protected(symbolic_addr)
t   : $M_pub == $M'_pub && $P == $P'
implies
t+n: $P_arch == $P'_arch
```

Less conservative confidentiality
non-interference CTL property:

$$\mathrm{AG}(\ \$M_{pub} = \$M'_{pub} \wedge \$P = \$P' \rightarrow \mathrm{AG}\ (\$P_{arch} = \$P'_{arch})\ )$$

Specific reformulation of UPEC for a symbolic attacker task:

- Property assumes `cheri_protected(symbolic_addr)`
- Property covers all breakout Transient Execution Attacks
  (but does not target poisoning attacks)

Formal Model: UPEC-CHERI

Integrity non-interference CTL property:

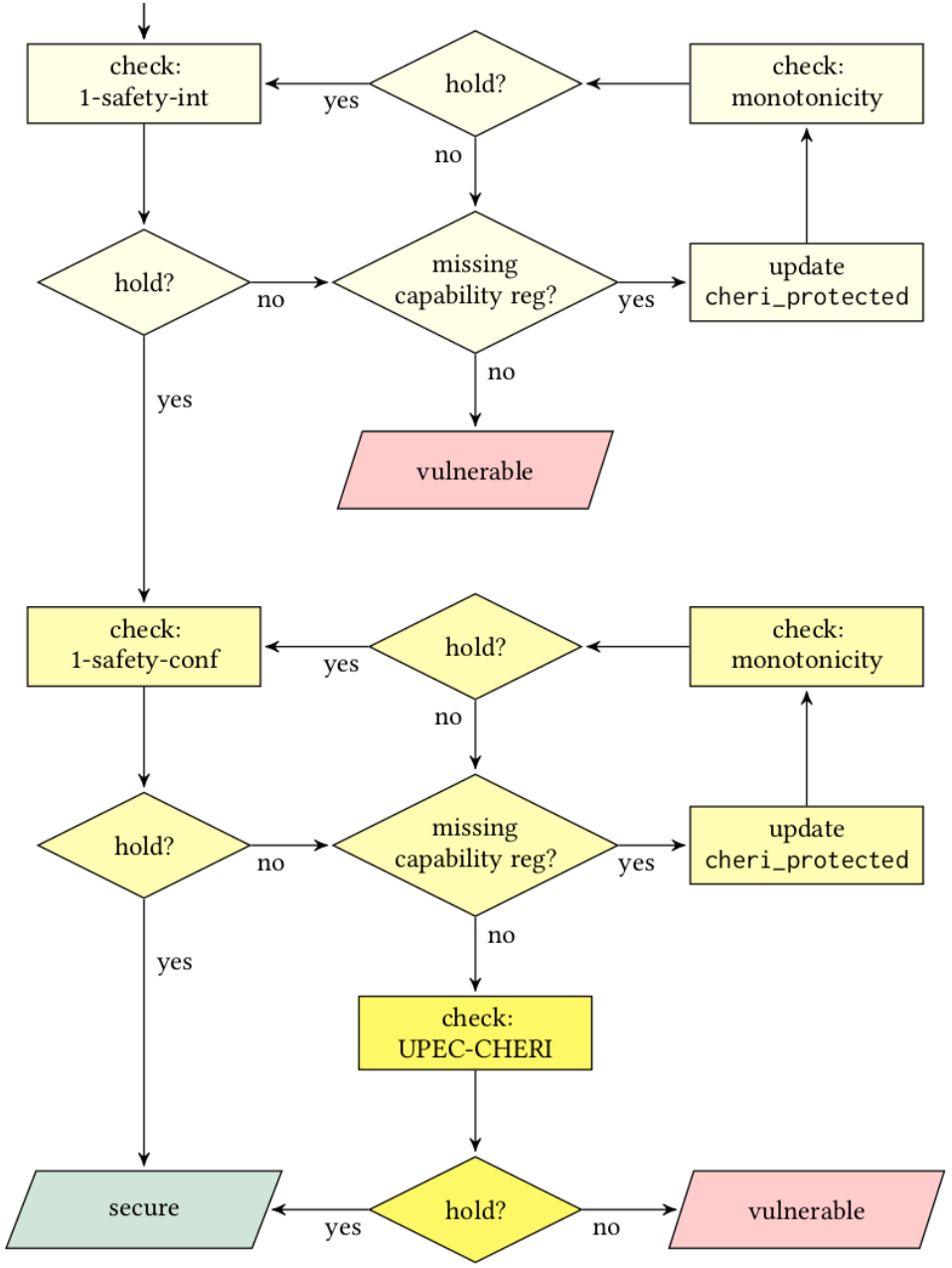
$$\mathrm{AG}(\ \$M_{prot} = \$M'_{prot} \rightarrow\ \mathrm{AG}(\$M_{prot} = \$M'_{prot})\ )$$

What if the integrity 1-safety property fails?

> After a write access to M_{prot} enters the memory, integrity is violated

→ 1-safety property is sufficiently precise

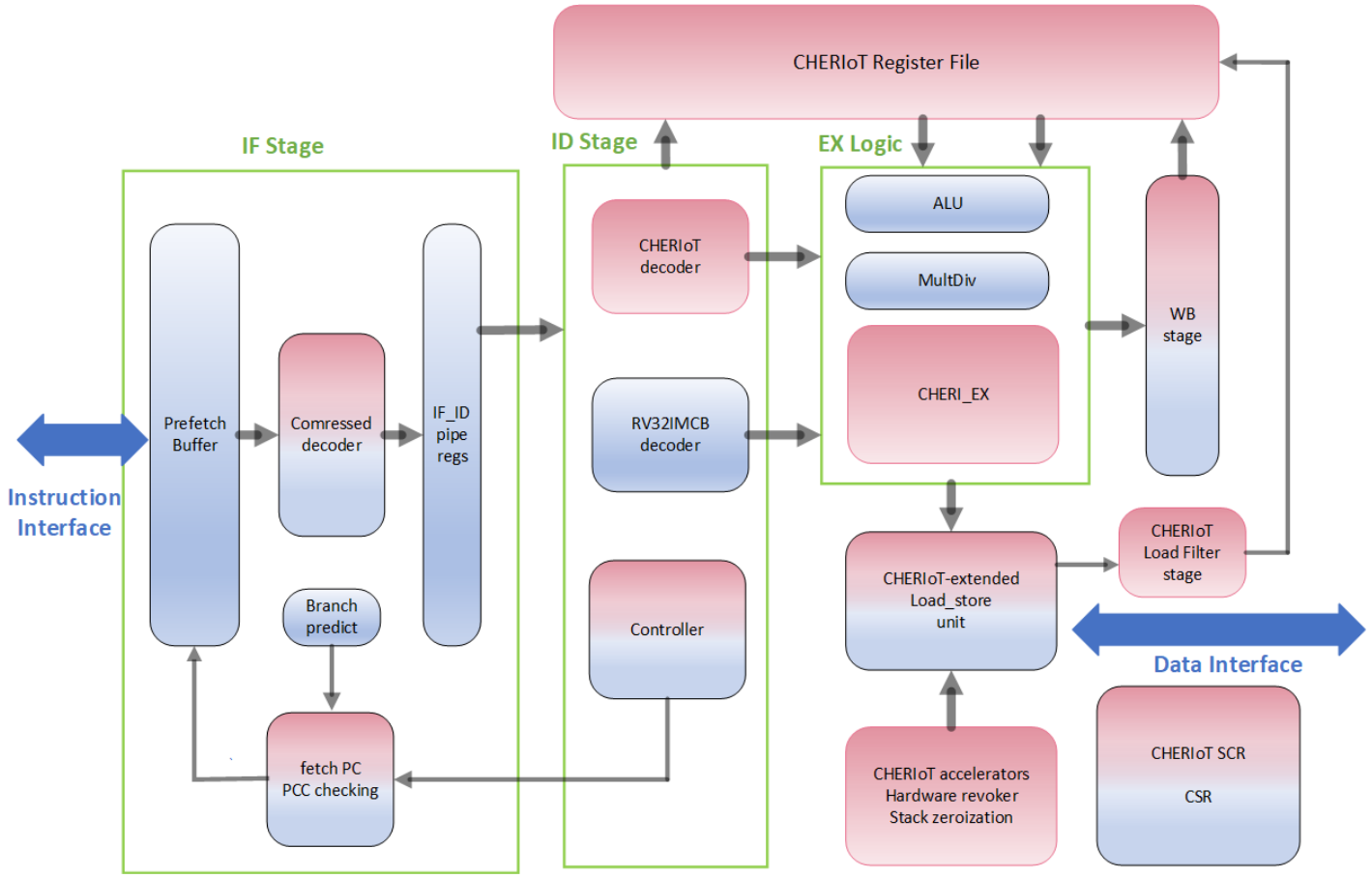
VeriCHERI Flow



RPTU

Case Study: CHERIoT Ibex Processor

CHERIoT Ibex implements a variant of RISC-V CHERI tailored to IoT and real-time applications



Source: https://github.com/microsoft/cheriot-ibex

RPTU

Case Study: CHERIoT Ibex Processor

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 Processor

UPEC-CHERI 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

> Performance counter change depending on the two bits

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

Conclusion

> VeriCHERI detected several security issues including a vulnerability to a Transient Execution Attack, which is not detectable by previous methods

> Formulating the security objective as single-cycle interval properties allows us to introduce a scalable iterative verification flow

> The developed invariants are implemented as symbolic verification IPs which may be reused for similar designs