BLACKOUT
Data-Oblivious Computation with Blinded Capabilities

ERICSSON

UNIVERSITY OF WATERLOO

CHERITech'25 Conference, Manchester UK

Merve Gülmez*, Thomas Nyman
Ericsson, Sweden

Hossam ElAtali*, N. Asokan
University of Waterloo, Canada

*) Joint first authors

Motivation: memory safety and side channels

ERICSSON

Weak **memory-safety** and **side-channel leakage** are major implementation challenges for **cryptographic and security-critical software**

Languages like C and C++

lack memory-safety, leading to security vulnerabilities; memory-safe languages and hardware (e.g., CHERI) can mitigate these risks.

Constant-time programming

prevents timing side-channels but is difficult to implement correctly; even compilers and hardware optimizations can introduce side-channel leakage.

CHERI hardware

addresses memory safety but does not inherently defend against side-channel attacks; data can leak silently.

BLACKOUT extends CHERI

with *blinded capabilities* and *hardware-enforced taint tracking* that mark secret data in registers and propagate secrecy through arithmetic and logical operations.

Data-oblivious computation

is needed by constant-time code; BLACKOUT enforces that data accessed by blinded capablities is handled in data-oblivious manner.

BLACKOUT prototype

on a CHERI-RISC-V FPGA softcore and integrated with CheriBSD OS shows minimal performance overhead (1.5% vs. baseline CHERI-RISC-V)

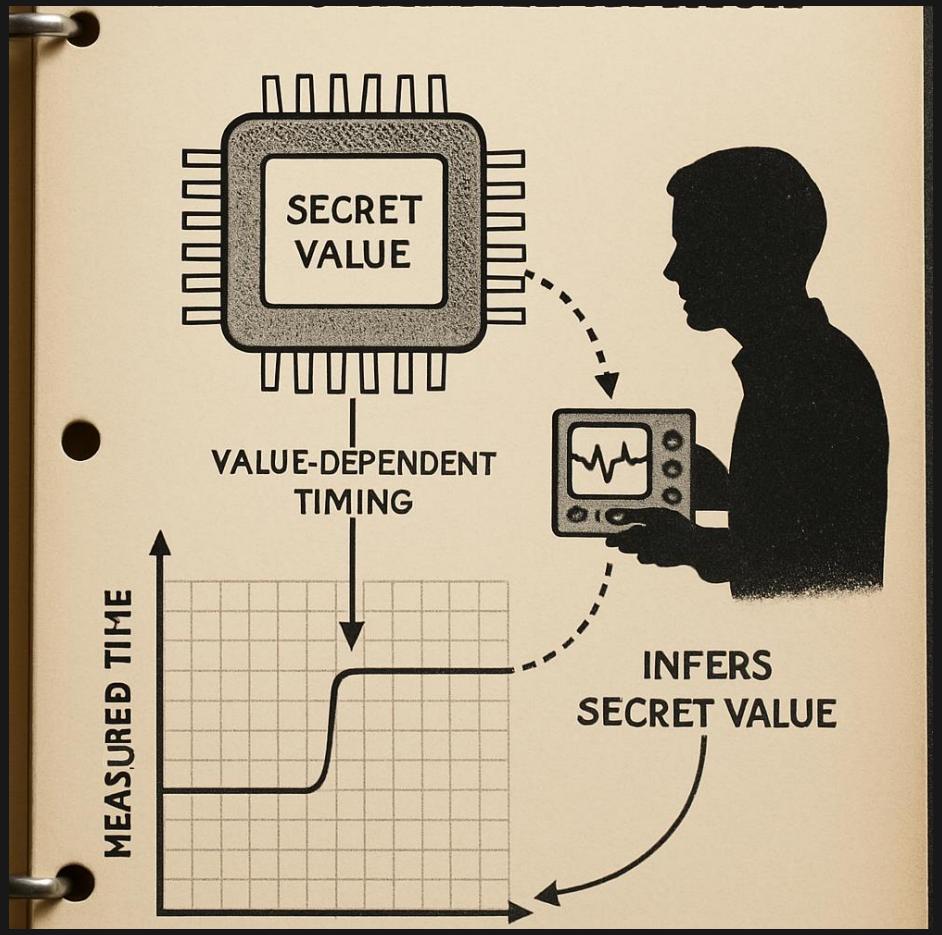
Background: Timing side channels in CPUs

Unintended outputs of a system caused by

- timing-dependence on (secret) data operands

Root causes:

- Logical and functional properties of processors (branch predictors, caches, translation lookaside buffers, execution ports, arithmetic circuitry[1])
- Software-visible physical properties of circuits (power consumption or frequency scaling)
- Misspeculation in speculative execution



[1] Ken Shirrf, The Pentium contains a complicated circuit to multiply by three, Ken Shirrf's blog. 2025-03-02.

Example: Cache side channels

substitution lookup table

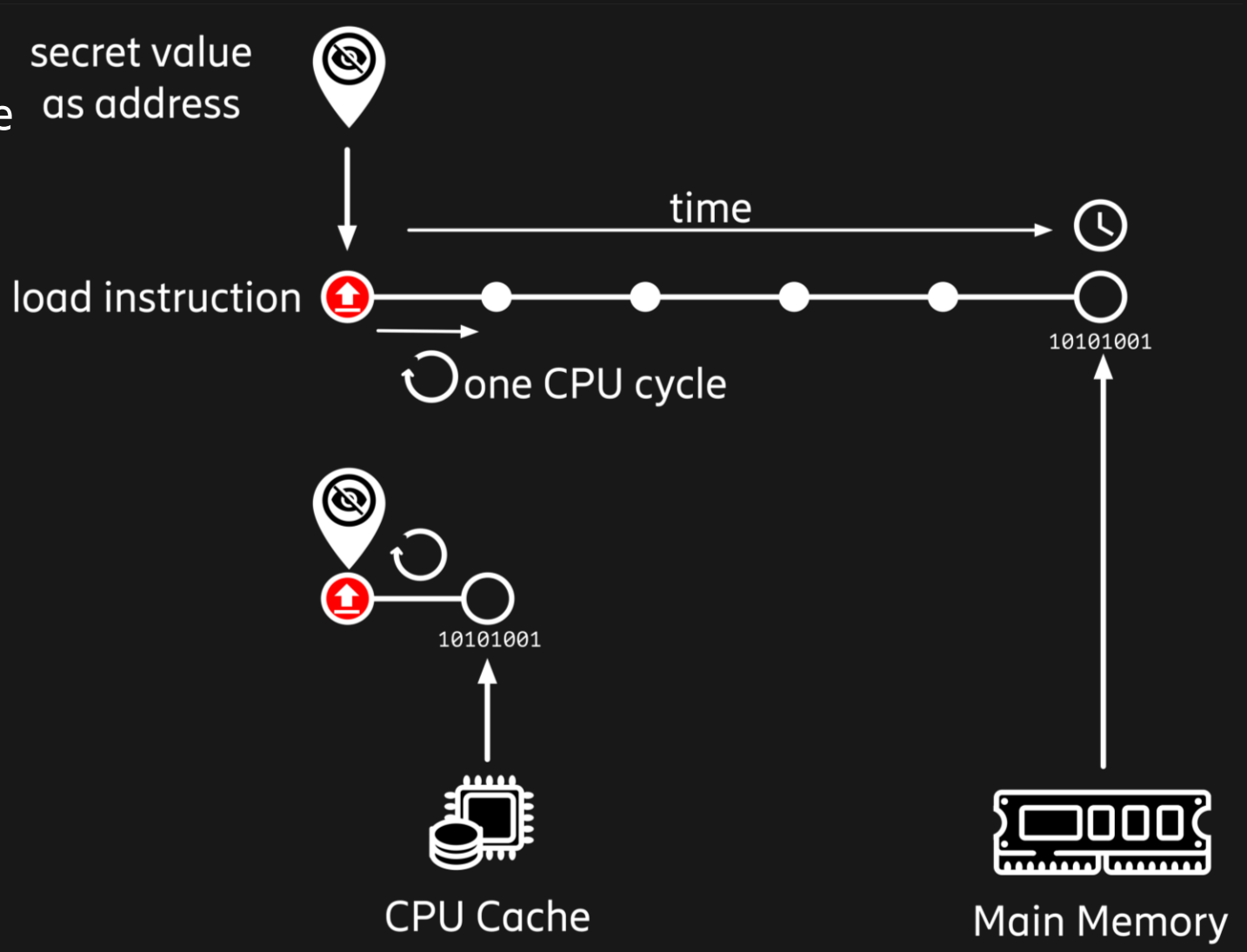
plaintext byte

secret key byte

```
blk0 = sbox[ blk0 ^ key0 ]
```

- A secret value used as *address* (pointer or index) controls *which* cache lines a victim process touches

- Deterministic mapping turns cache occupancy into fingerprint of secret



Example: Cache side-channel attacks

ERICSSON

Adversary model:

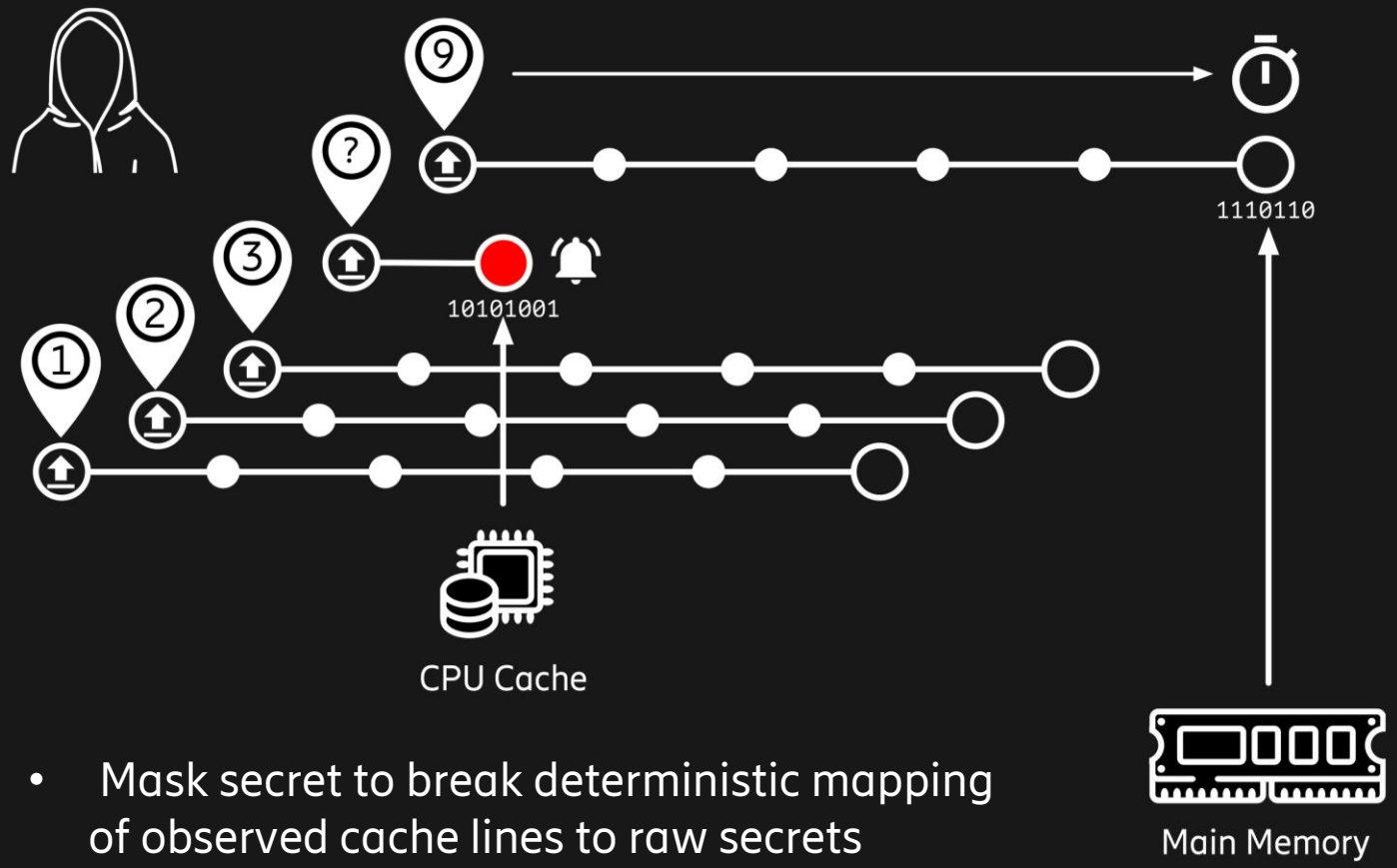
- Adversary and victim processes share same CPU/cache (same core, sibling SMT thread)
- Adversary cannot read secret directly but can measure timing/latency differences caused by the victim's memory accesses

What the adversary observes:

- Access times to many candidate lines to infer which lines the victim recently used, and further infer the secret address

Potential mitigations:

- Isolate or partition µarchitectural resources (process/core isolation, disable SMT)



- Mask secret to break deterministic mapping of observed cache lines to raw secrets

Data-oblivious code and ISAs

Programs where the execution path and memory access patterns are independent of input data

Achieved by eliminating data-dependent behavior:

- conditional branching on secret data, and
- addressing memory using secret data as address

Caution: seemingly data-oblivious source code can still lead to side channels and silent leakage!

- Optimizing compilers can introduce side channels into generated machine code, e.g., branch of secrets
- Hardware optimizations can cause side channels even with data-oblivious machine code

Data-oblivious instruction-set architectures (ISAs) enforce lack of data-dependent behavior run-time

Enabled through:

- Hardware taint-tracking of secret data in register and throughout memory (using memory tags)
- Enforcing tainted data cannot be used in branching decisions or as memory addresses

Prior Work:

- OISA (Oblivious Instruction Set Architecture) [YHEF+19]
- BliMe (Blinded Memory) [EGLA+24]

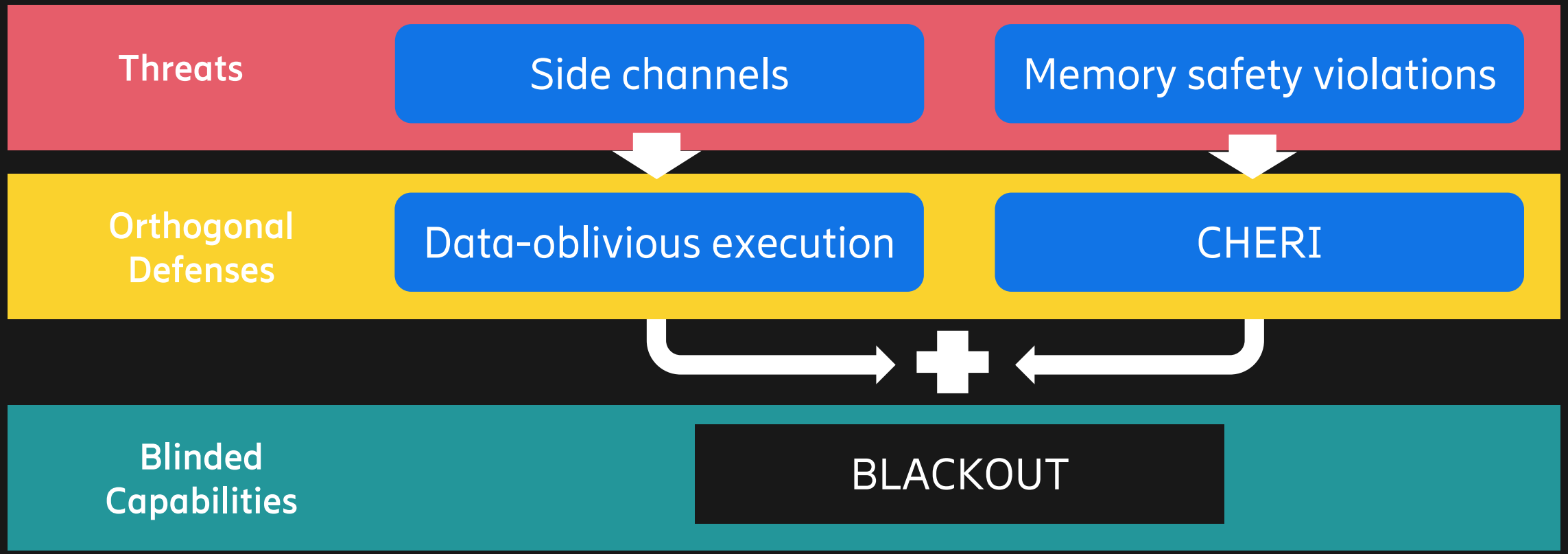
Limitations: high (memory tagging) overhead + difficult to program correctly

[EGLA+24] "BliMe: Verifiably Secure Outsourced Computation with Hardware-Enforced Taint Tracking", NDSS (2024)
[YHEF+19] "Data Oblivious ISA Extensions for Side Channel-Resistant and High Performance Computing", NDSS (2019)

Can we use CHERI capabilities (instead of memory tagging) to track tainted data in memory?

Goal: Memory safety and side-channel confidentiality

Provide a unified, efficient hardware-software solution for memory safety and side-channel confidentiality



BLACKOUT Adversary Model

ERICSSON

Same adversary model as CHERI:

- Capability-enhanced speculative processor
- Mutually distrusting userspace tasks
- Trusted OS securely manages context switches and reclaiming resources of completed tasks

In addition:

- adversary has ability to observe side channels during program execution, e.g., from concurrent "spy" task

In scope: SCAs mitigatable by data-oblivious code

- Cache timing
- Instruction timing (with DIT mode)
- Port-contention timing
- Transient execution attacks that leak information **loaded** by mis-speculated instructions in same address space, e.g., Spectre (all variants)

Out of scope:

- Transient execution attacks that load data across address spaces, such as Meltdown and microarchitectural data sampling
- Transient capability forgery, such as Meltdown-CF (but preventable by capability speculation contracts)

BLACKOUT Overview

ERICSSON

BLACKOUT hardware uses notion of "blindedness" to taint register file and track secret/sensitive data

- Blindedness tracked by one extra bit per register (blindedness bit transparent to software)
- Moves, arithmetic operations etc. propagate blindedness between registers (blinded registers)

Loads to and stores from blinded register must occur through new *blinded capabilities* (BCs)

- BCs ensured to have exclusive access to their area of memory (blinded data)
- Memory loaded with BCs is tainted in registers with blindedness bit set

Compiler and software stack guides developer towards data-oblivious code

- Compiler generates BCs for stack variables (from annotations) and analyzes code for leakage
- Runtime initialization code generates BCs for global blinded data
- Blinded `malloc()` generates BCs for heap and clears blinded data on revocation

BLACKOUT's Blinded Capabilities

Secret data can only be stored in special "blinded" memory areas via blinded capabilities

- Misuse of data accessed through blinded capabilities (e.g., secret-dependent control flow or loads) cause faults, preventing side-channel leakage

Blinded capabilities turn silent side-channel leakage into explicit programming errors!

Blinded CHERI-Tooba Design

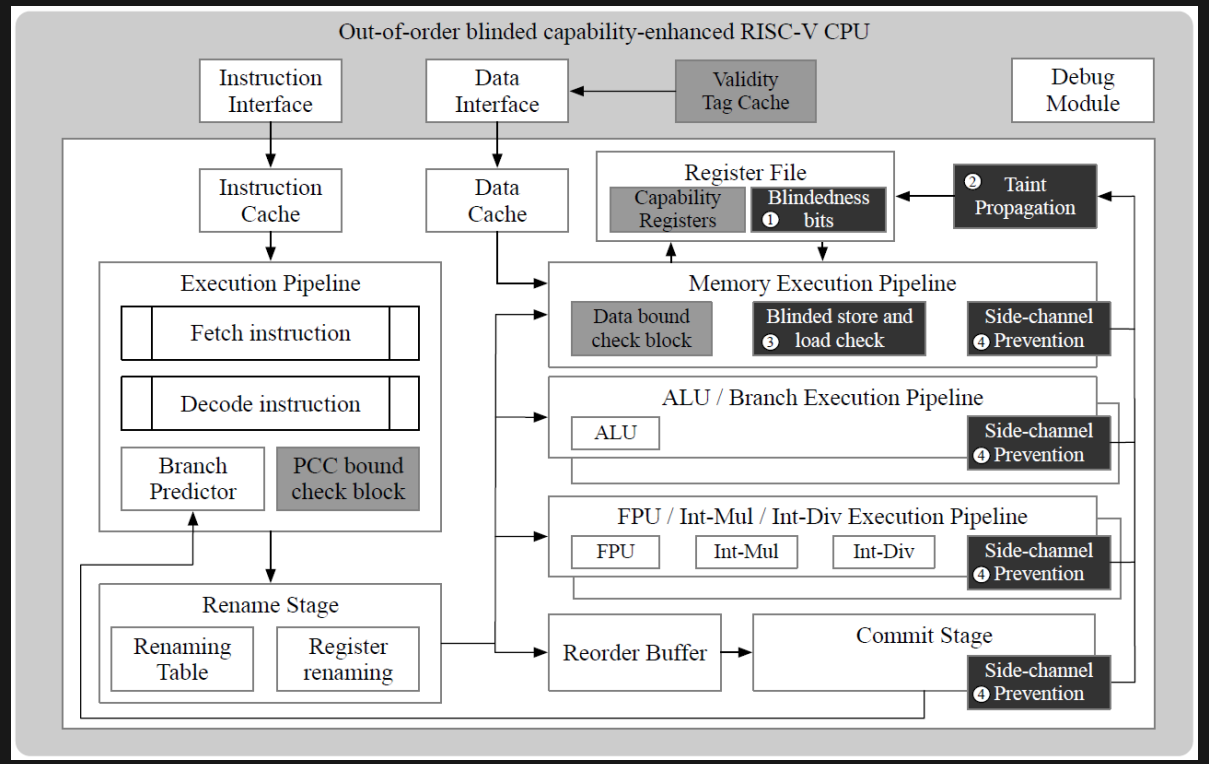
ERICSSON

1. Blindedness bits in register file
2. Taint-propagation logic
3. Side-channel prevention (see below)
4. Blinded store and load check

Side-channel prevention 3 enforces operations on blinded operands must be data-oblivious

- No control-flow decisions
- No load or store address operands

ISA-level changes limited to new permission bit and `csc/clc` handling of spilled blinded registers



Ensuring exclusive access to blinded memory

ERICSSON

Exclusive access invariants:

1. Blinded data cannot be stored using non-BCs → enforced by hardware
2. No capabilities can be blinded → enforced by hardware
3. Bounds of valid BCs and non-BCs must not overlap → enforced by software

Compiler and heap allocator ensure that BC and non-BC bounds cannot overlap

- Blinded data is cleared before memory region is reused
- Compiler handles blinded local variables on stack frame pop
- Heap allocator uses Cornucopia[F+20]-style reclamation for blinded data

[F+20] "Cornucopia: Temporal Safety for CHERI Heaps" in IEEE Symposium on Security and Privacy 2020

Empirical Evaluation on FPGA Softcore

ERICSSON

Evaluation of BLACKOUT-enabled CHERI-Toooba Virtex Ultrascale+ VCU118 FPGA at 25MHz

Area and Power expressed in number of lookup tables (LUTs), registers, and Watts :

	logic	Δ(%)	memory	Δ(%)	registers	Δ(%)	power	Δ(%)
CHERI-Toooba Core	697508	–	20852	–	412493	–	6.205	–
Blinded CHERI-Toooba Core	705863	1.2	20855	0.0	412913	0.1	6.536	5.3

Low area overhead: **≈1%**

Moderate power overhead: **≈5%**

Performance evaluation on CheriBSD

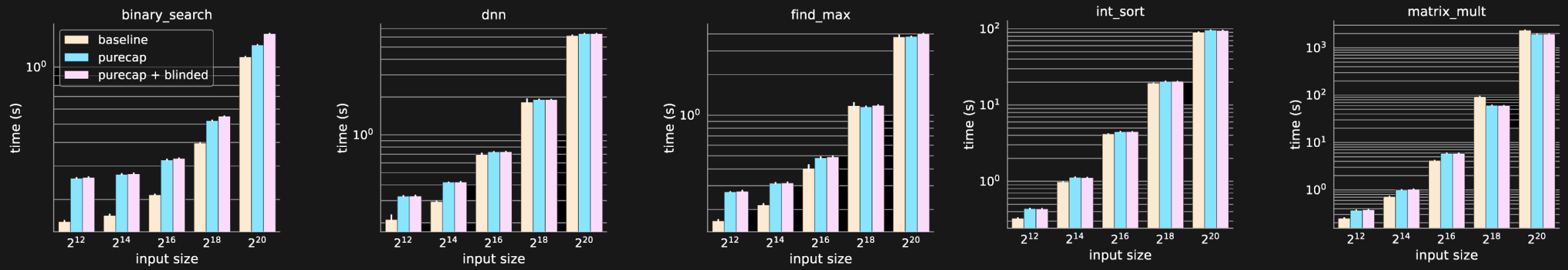
Performance impact measured using OISA[YHEF+19] benchmark of data-oblivious code samples:

- Porting OISA benchmarks to BLACKOUT took 1-5 LoC changes to adapt annotations (<1%)
- **1.5% overhead** over CHERI in purecap mode (g.m.)

Note: Measure *cost of data-oblivious enforcement* (not cost of converting code to data-oblivious)

Performance impact on code without blinded capabilities measured using CoreMark benchmark:

- Neglible performance impact: ≈**0.1%**



Empirical Security Evaluation: Spectre and Non-Interference

Resistance against Spectre evaluated using Fuchs et al.'s test suite [F+21]:

- Baseline CHERI-Toooba vulnerable to Spectre-BTB, RSB and STL (but resistant to Spectre-PHT)
- BLACKOUT CHERI-Toooba successfully stops all above Spectre variants against blinded data

	Spectre variant			
	PHT	BTB	RSB	STL
CHERI-Toooba	✓	✗	✗	✗
Blinded CHERI-Toooba	✓	✓	✓	✓

In addition, replicated ProSpeCT's [D+23] RISC-V SpectreGuard [FFY19] tests:

- CHERI inherently precludes dereferencing non-capability data and thus resistant to all tests

Non-interference of data-oblivious code verified from signal traces

- Based on methodology used in *Libra* [F+24]
- Collect signal traces from hardware simulation across multiple runs and different inputs
- Determine whether side-channel relevant signals are data-oblivious (same across runs)

[FFY19] "SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks" in ACM/IEEE Design Automation Conferece (DAC) 2019.
[F+21] "Developing a test suite for transient-execution attacks on RISC-V and CHERI-RISC-V" in Computer Architecture with RISC-V workshop (CARRV) 2021.
[D+23] "ProSpeCT: Provably Secure Speculation for the Constant-Time Policy" in USENIX Security 2023.
[F+24] "Libra: Architectural Support for Principled, Secure and Efficient Balanced Execution on High-End Processors" in ACM CCS 2024.

Summary

BLACKOUT combines CHERI memory safety with side-channel confidentiality

- Blinded capabilities allow access to secret data only in data-oblivious manner
- Data-oblivious programming model and compiler support leads to safer and easier to maintain constant-time code
- BLACKOUT is demonstrated on CHERI-RISC-V FPGA with minimal performance, area and power impact relative to baseline CHERI
  - Prototype evaluated for non-interference (side-channel relevant hardware signals are data-oblivious) and Spectre PHT, BTB, RSB, and STL variants.
- Published and previously presented at ACM CCS '25 in Taipei, Taiwan

Pre-print available:

https://arxiv.org/abs/2504.14654

ERICSSON

https://www.ericsson.com/en/security