Data Oblivious ISA Extensions for Side Channel-Resistant and High-Performance Computing

Network and Distributed System Security Symposium (NDSS), 2019
Distinguished Paper Award Honorable Mention/CSAW Applied Research Competition Finalist

Jiyong Yu, Lucas Hsiung, Mohamad El Hajj, Christopher W. Fletcher
University of Illinois at Urbana-Champaign

Introduction

Microarchitecture side channel attacks:

- Huge privacy threat
- Fundamental problem: secret data affects HW resource pressure
- Various attacks proposed for different HW resources

Difficulty in solving this type of attacks:

- No contract between HW and SW
  - SW doesn't know what HW leaks
  - HW doesn't know what is secret in SW

Data Oblivious Programming

Definition:

- SW solution to block microarchitecture side channel attacks
- Rewrite programs to be data oblivious, i.e., remove visible data-dependent behaviors
- Data oblivious programs can be represented as a static data-flow graph

Security assumptions:

- Instructions are evaluated in data-independent manner
- Data is transferred in data-dependent manner
- Instruction sequence is not a function of data

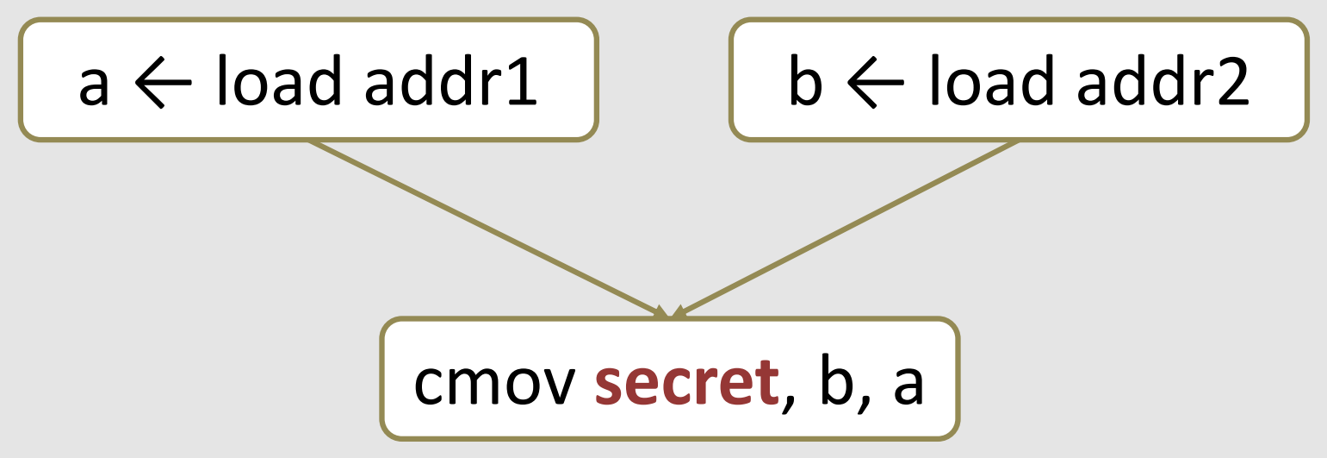
HW optimizations undermine all of them!

```
// source code
if (secret)
    a ← load (addr1);
else
    a ← load (addr2);
```

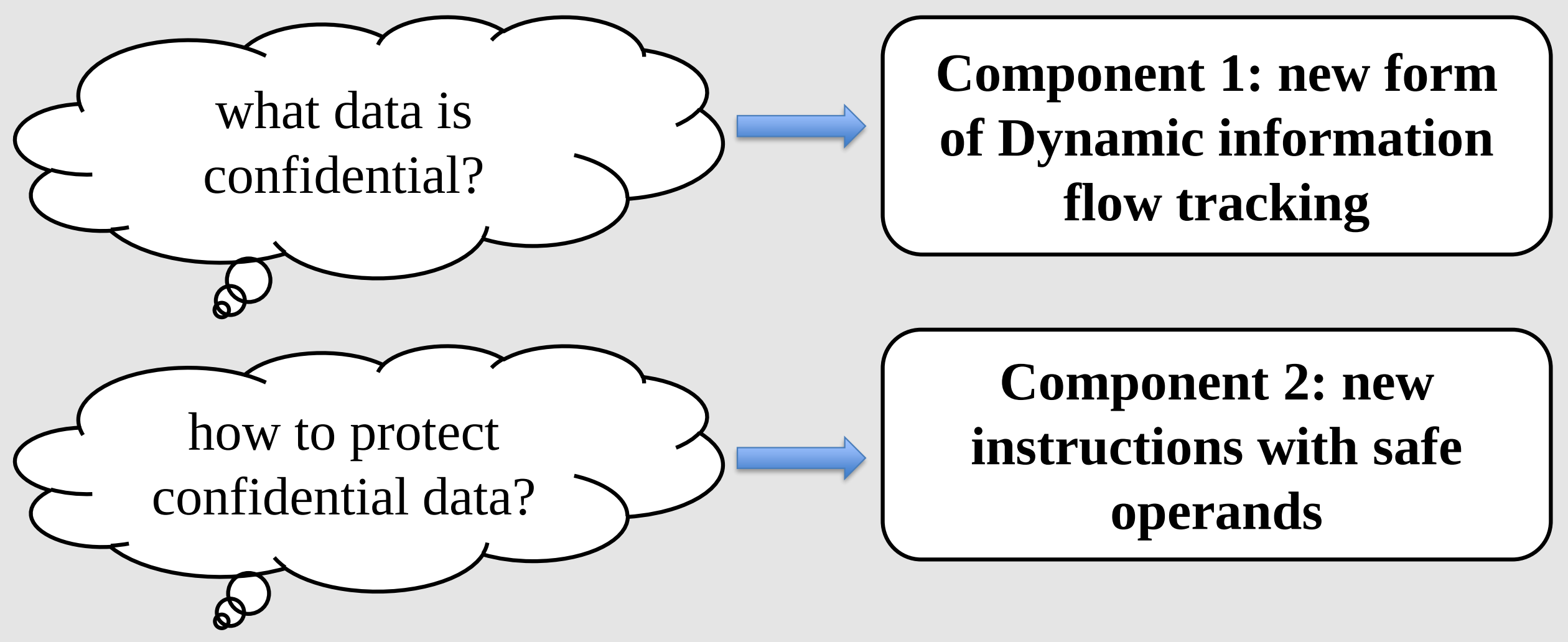
→

```
// data-oblivious code
a ← load (addr1);
b ← load (addr2);
cmov secret, a, b;
    // a = secret? b : a
```

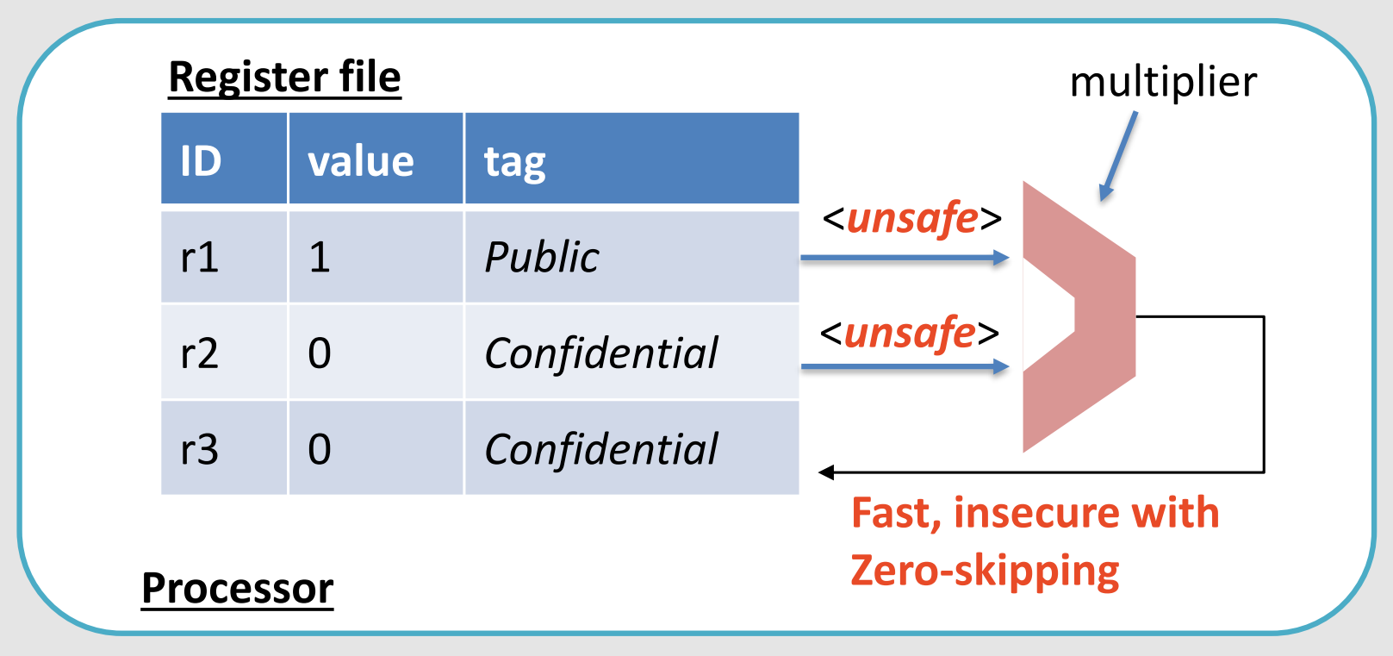
→



Data Oblivious ISA (OISA) Design

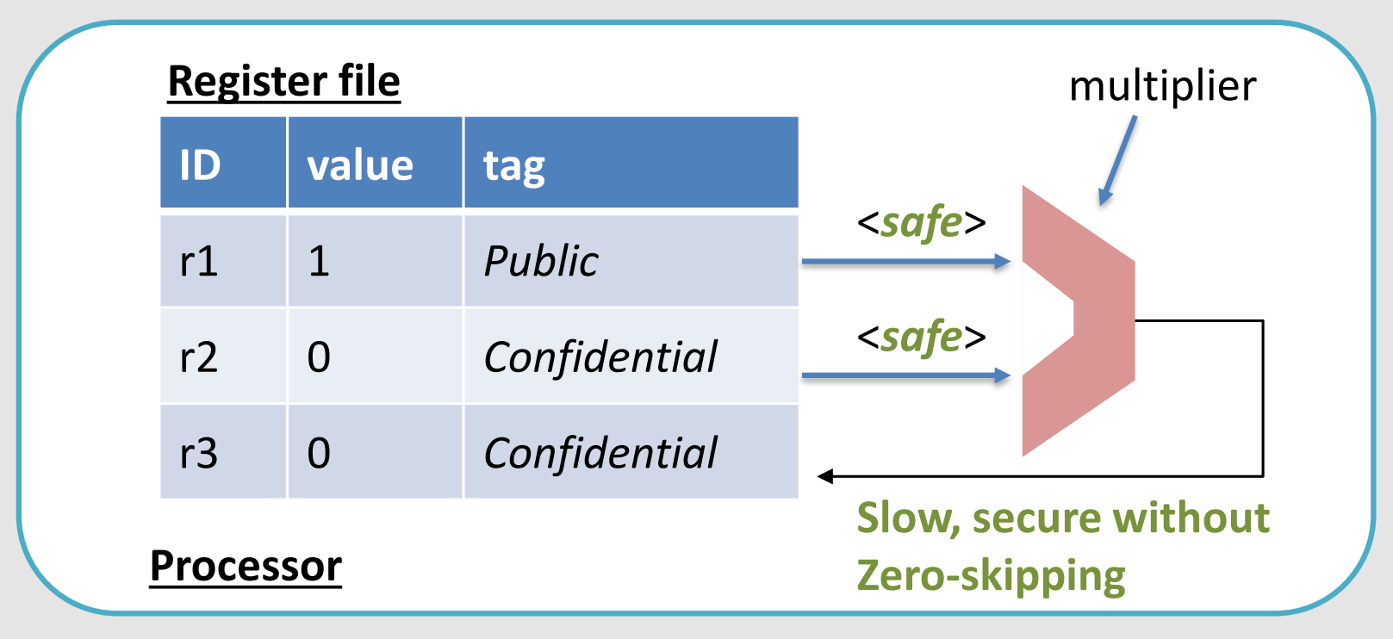

what data is confidential?
Component 1: new form of Dynamic information flow tracking
how to protect confidential data?
Component 2: new instructions with safe operands

- **Security:**
  - ISA specifies what data is secret
  - ISA specifies what operation can leak
- **Efficiency:** HW free to data oblivious operations
- **Portability:** ISA and security specification is fixed across hardware implementations



Register file

ID	value	tag
r1	1	*Public*
r2	0	*Confidential*
r3	0	*Confidential*



Register file

ID	value	tag
r1	1	*Public*
r2	0	*Confidential*
r3	0	*Confidential*

New Dynamic Information Flow Tracking (DIFT):

- Programmers declare data as *Public* or *Confidential*
- *Confidential* data is tracked in hardware using DIFT
- *Rules:*
  - *Public data needs no protection*
  - *Confidential data must be protected*

Instruction with Safe/Unsafe Operand

- Each input operand is defined as *Safe* or *Unsafe*
  - *Safe* operand blocks side channel from that operand
  - *Unsafe* operand provides no protection

Putting it all together

ISA designers decide instructions with Safe/Unsafe operands

Hardware designers augment processors with logic to enable/disable hardware optimizations

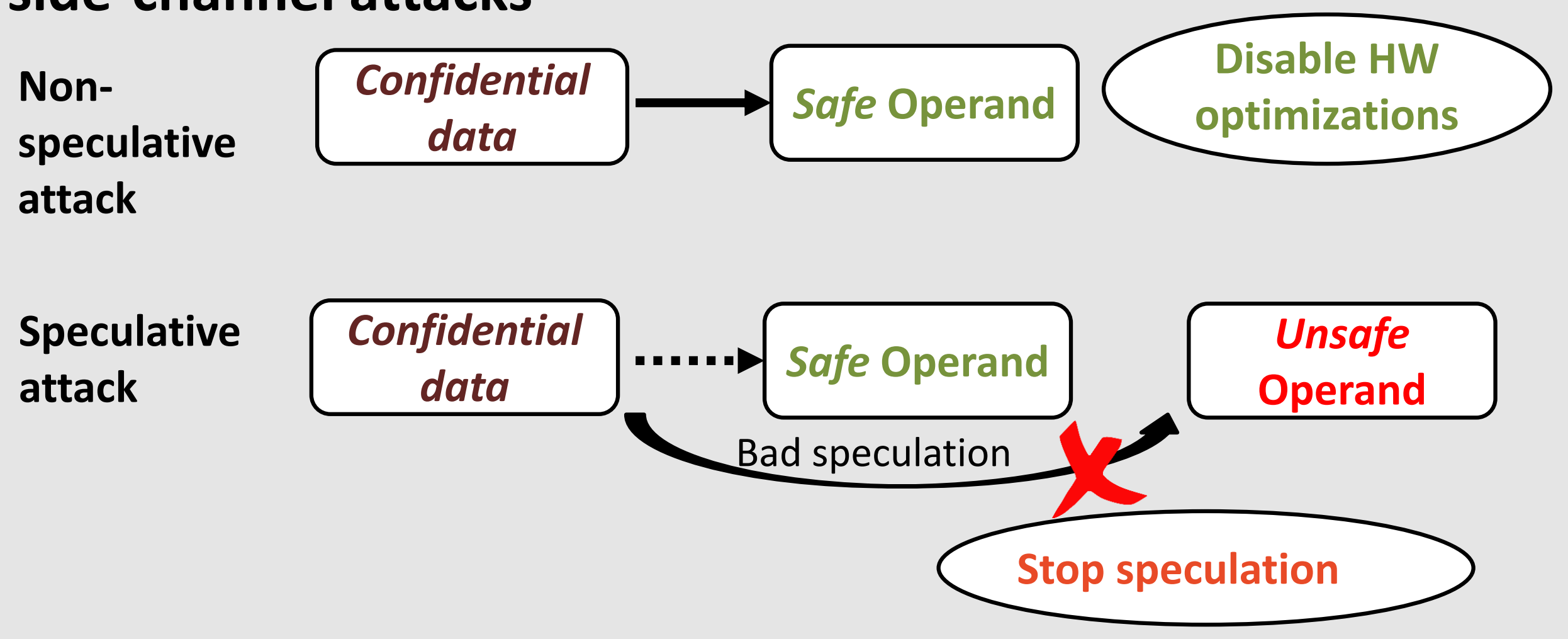
Programmers annotate data as Confidential/Public

Processors enforce taint propagation and transition rules:

- *Public* data → *Safe* operand >> <u>no protection</u>
- *Public* data → *Unsafe* operand >> <u>no protection</u>
- *Confidential* data → *Safe* operand >> <u>disable HW optimizations</u>
- *Confidential* data → *Unsafe* operand >> <u>exception!</u>

Design Feature

Security: Defense against non-speculation & speculative side-channel attacks



Efficiency: Design space for safe optimizations

- Case 1: Oblivious load operation: O(N) → O(logN) → O(1)
- Case 2: Oblivious sort operation: $O(N\log^2 N)$ → O(NlogN)

Portability: Consistent security guarantee across hardware instances

Hardware Implementation

Hardware prototyping on RISC-V BOOM

Performance Evaluation

- Achieving speedup of upto 8.8x over baseline data oblivious programming
- Case studies:
  - AES: 4.4x speedup over bitslice AES
  - Memory oblivious library: more than 4.6x speedup over ZeroTrace [SGF'18]

Security Evaluation

- Proving non-interference property for the trace of observable processor states
- Challenges:
  - Formalizing attacker's observability
  - Modeling complicated modern processors