Hardware-Software Contracts for Safe and Secure Systems

Jan Reineke @ UNIVERSITÄT DES SAARLANDES

This work has received funding from an Intel Strategic Research Alliance (ISRA) and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101020415)"

The Need for New HW/SW Contracts

“Stone-age” Computing

Applications implemented data transformations:
e.g. payroll processing

Hardware:

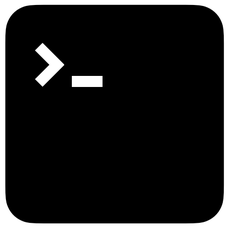
- isolated, on-site
- limited interaction with environment

IBM System 360/30

HW/SW Contract: Instruction set architecture (ISA)

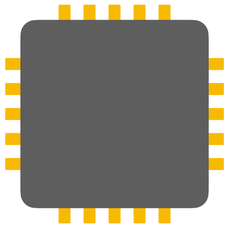
ISA: Benefits

High-level language



Instruction set architecture (ISA)

Microarchitecture



Can program **independently** of microarchitecture

Can implement **arbitrary optimizations** as long as ISA semantics are obeyed

“Modern” (?) Computing

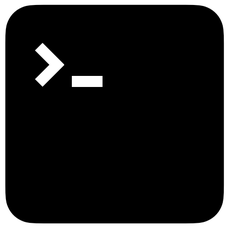
Applications are:

- ***Data-driven***: e.g. deep neural networks
- ***Distributed***: e.g. locally + in the cloud
- ***Open***: e.g. untrusted code in the browser 
- ***Real-time***: interacting with the physical environment 

What are the implications for HW/SW contracts?

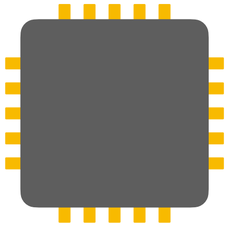
Inadequacy of the ISA: Real-time Systems

High-level language



Instruction set architecture (ISA)

Microarchitecture



Impossible to program
real-time systems on top of ISA

Abstracts from time
No control over time

Can implement arbitrary
unpredictable optimizations as long
as ISA is implemented correctly

Wanted: Timed HW/SW Contracts

Programs have a **timed semantics** that is **efficiently predictable**
Programs have **control** over timing

Timed Instruction Set Architecture

Admit a **wide range** of **timing-predictable**, yet high-performance microarchitectural **implementations**

Wanted: Timed HW/SW Contracts

Some answers:

E. Lee, J. Reineke, and M. Zimmer:
Abstract PRET Machines
RTSS 2017

Determinism

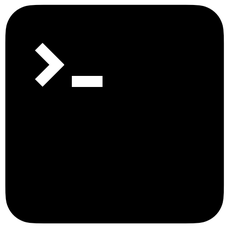
S. Hahn and J. Reineke:
Design and Analysis of SIC:
A Provably Timing-Predictable Pipelined Processor Core
RTSS 2018 (🥇 Best Student Paper Award)

Monotonicity

G. Stock, S. Hahn and J. Reineke:
Cache Persistence Analysis: Finally Exact
RTSS 2019 (🥇 Best Paper Award)

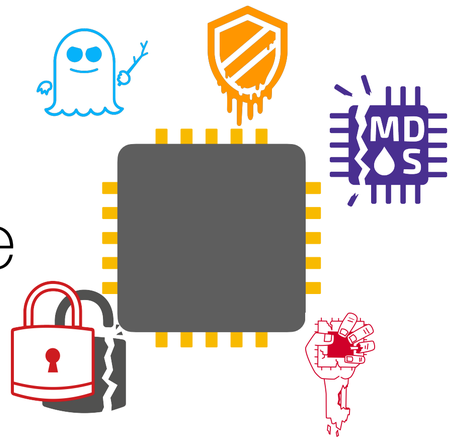
Inadequacy of the ISA: Side channels

High-level language



Instruction set architecture (ISA)

Microarchitecture

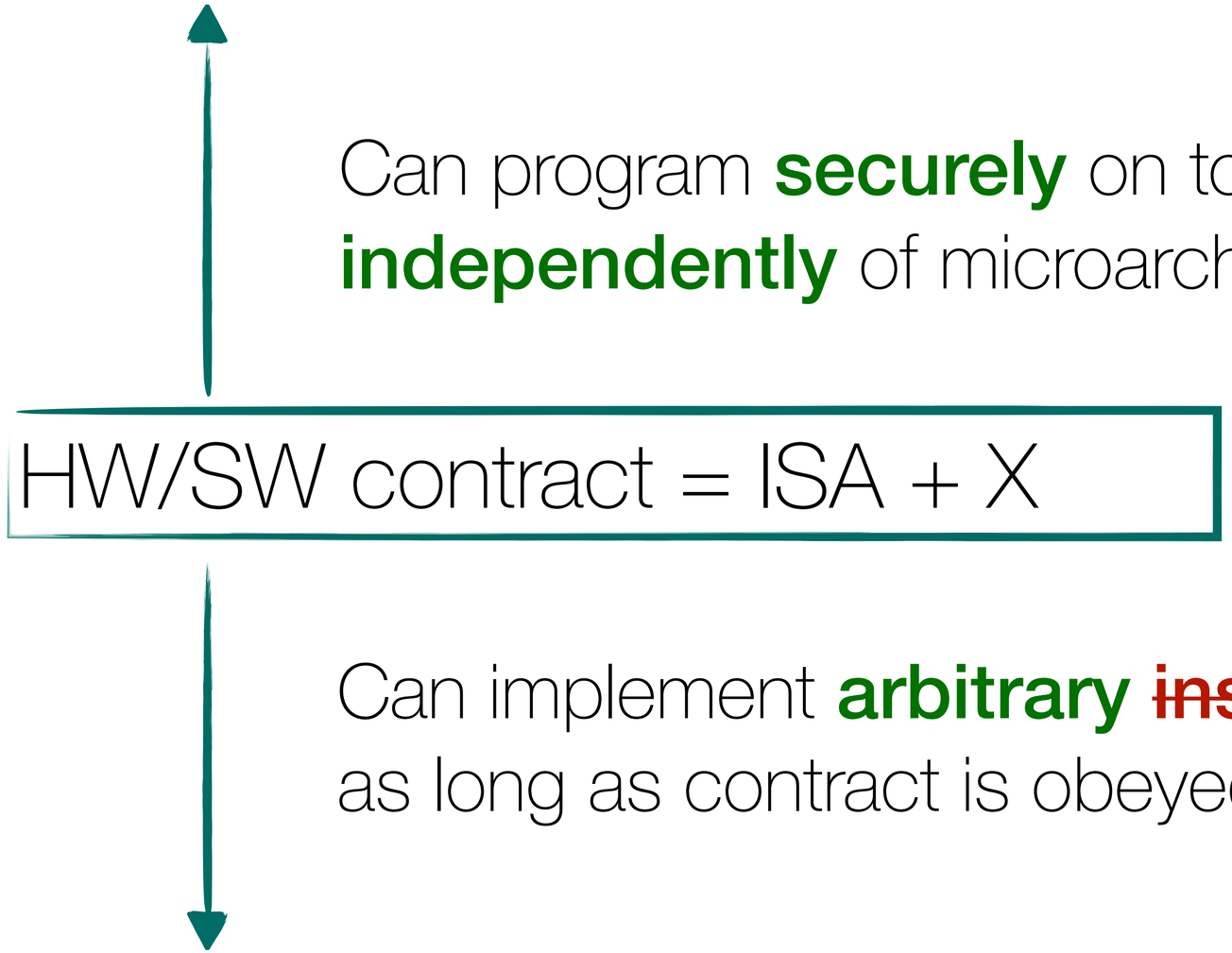


Impossible to program securely
cryptographic algorithms?
sandboxing untrusted code?

No guarantees about side channels

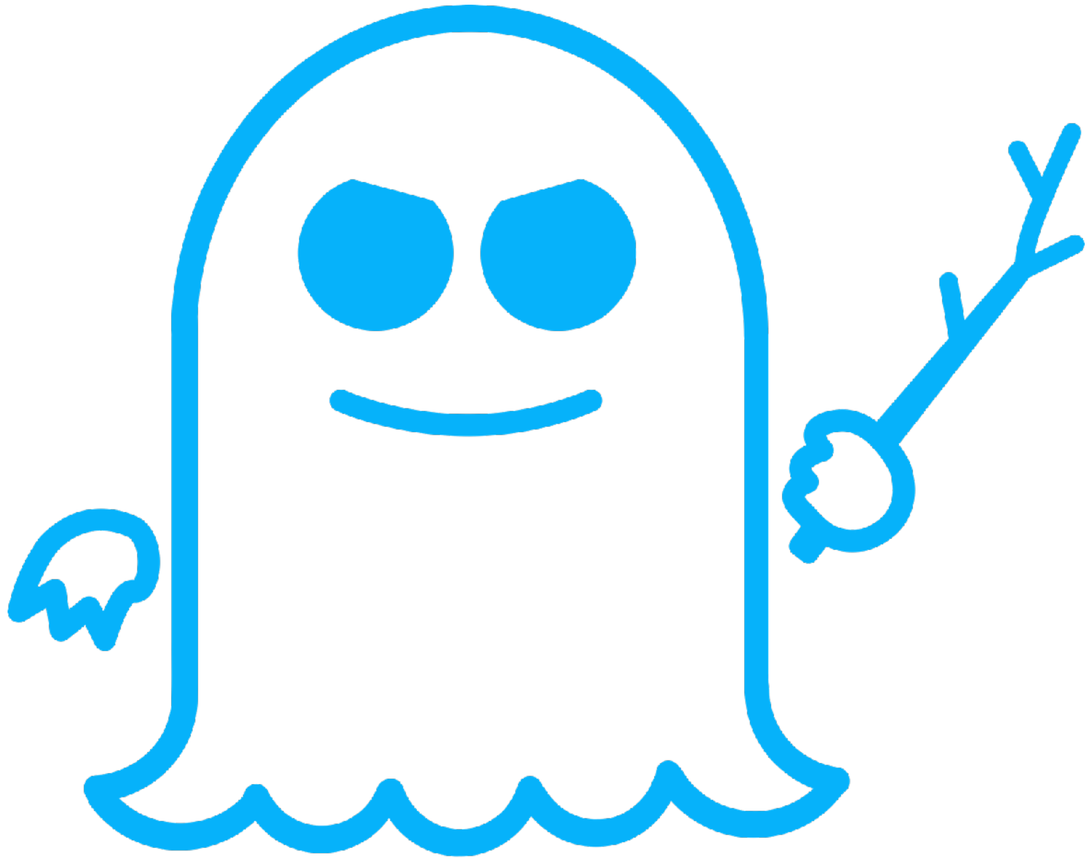
Can implement arbitrary **insecure** optimizations as long as ISA is implemented correctly

A Way Forward: HW/SW Security Contracts



Succinctly captures possible information leakage

A Concrete Challenge: Spectre



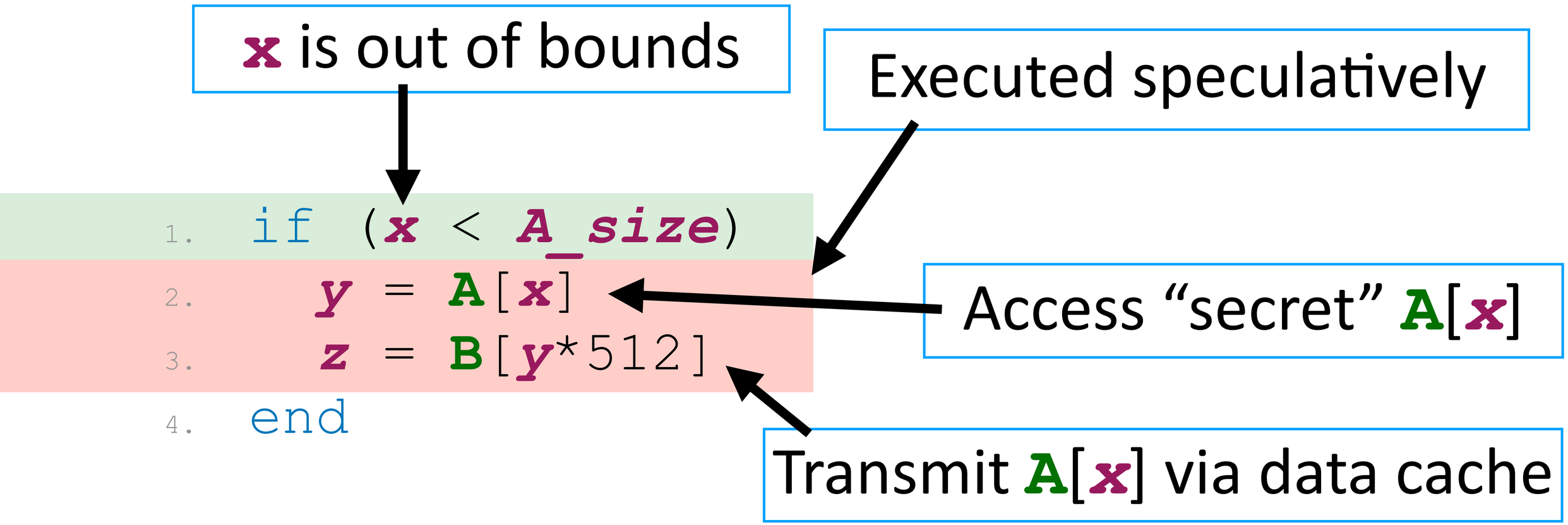
SPECTRE

Exploits *speculative execution*

Almost *all* modern *CPUs* are *affected*

P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, Y. Yarom — | Spectre Attacks: Exploiting Speculative Execution — S&P 2019

Example: Spectre v1 Gadget



Hardware Countermeasures

InvisiSpec: Making Speculative E
Invisible in the Cache

Mengjia Yan†, Jiho Choi†, Dimitrios Skarlatos, Ad
University of Illinois at
{myan8, jchoi42, skarlat2}@i

NDA: Preventing Speculative Execution Attacks at Their Source

Kevin Loughlin
University of Michigan

Ian Neal
University of Michigan

Baris Kasikci
University of Michigan

Ofir Weisse
University of Michigan

Thomas F. Wenisch
University of Michigan

Efficient Invisible Speculative Execution through Selective Delay and Value Prediction

Alberto Ros
University of Murc
Murcia, Spain
aros@ditec.um

Stefanos Kaxiras
Uppsala University
Uppsala, Sweden
stefanos.kaxiras@it.uu.se

Magnus Själander
Norwegian University of Science and Technology
Trondheim, Norway
magnus.sjalander@ntnu.no

os Sakalis
rsity

orean

CleanupSpec: An "Undo" Approach to Safe Speculation

Gururaj Saileshwar
gururaj.s@gatech.edu
Georgia Institute of Technology

Moinuddin K. Qureshi
moin@gatech.edu
Georgia Institute of Tech

Speculative Taint Tracking (STT): A Comprehensive Protection for Speculatively Accessed Data

Mengjia Yan
University of Illinois at Urbana-Champaign
myan8@illinois.edu

Artem Khyzha
Tel Aviv University
artkhyzha@mail.tau.ac.il

Josep Torrellas

Christopher W. Fletcher

Examples

```
1.  if (x < A_size)
2.      y = A[x]
3.      z = B[y*512]
4.  end
```

Delay loads until they can be retired

[Sakalis et al., ISCA'19]

Delay loads until they cannot be squashed

[Sakalis et al., ISCA'19]

Taint speculatively loaded data + delay tainted loads

[STT and NDA, MICRO'19]

Examples

```
1.  y = A[x]
2.  if (x < A_size)
3.     z = B[y*512]
4.  end
```

Delay loads until
they can be retired
[Sakalis et al., ISCA'19]

Delay loads until they cannot
be squashed
[Sakalis et al., ISCA'19]

Taint speculatively loaded data
+ delay tainted loads
[STT and NDA, MICRO'19]

What security properties do HW countermeasures enforce?

How can we program securely?

Some Answers:

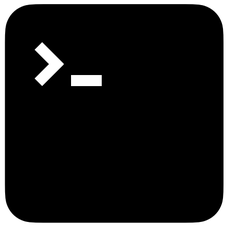
M. Guarnieri, B. Köpf, J. Reineke, and P. Vila

Hardware-Software Contracts for Secure Speculation

S&P (Oakland) 2021 (🏅 Best Paper Award)

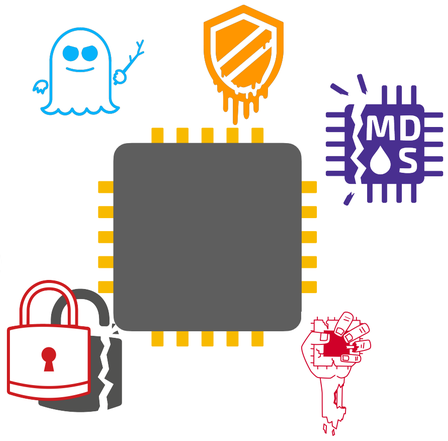
Research Landscape around HW/SW Contracts

High-level language



Hardware-Software Contracts

Microarchitecture



Program Analysis
Abstract Interpretation
Symbolic Execution

How to capture guarantees?

Verification SMT Solvers
+ Invariant Synthesis

Testing Fuzzing

Design Domain Specific Language

Happy to chat about PhD and internship opportunities

Thank you for your attention