Microarchitecture Vulnerabilities

Past, Present and Future

Daniel Gruss (Graz University of Technology)
Anders Fogh (Intel Corporation)

Introduction

Daniel Gruss
Graz University of Technology

Anders Fogh
Intel

Daniel and Anders
do not always agree!!


Past

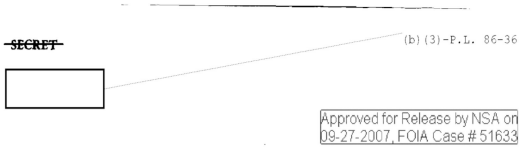
Past – earliest days

Side Channels always existed

Past – earliest days

Side Channels always existed

First scientific observations in 1943



TEMPEST: A Signal Problem

**The story of the discovery
of various compromising radiations
from communications and Comsec equipment.**

impractical. Hydraulic techniques—to replace the electrical—were tried and abandoned, and experiments were made with different types of batteries and motor generators, in attempts to lick the power-line problem. None was very successful.

During this period, the business of discovering new TEMPEST threats, or refining techniques and instrumentation for detecting, recording, and analyzing these signals, progressed more swiftly than the art of suppressing them. Perhaps the attack is more exciting than the defense—something more glamorous about finding a way to read one of these signals than going through the drudgery necessary to suppress that whacking great spike first seen in 1943. At any rate, when they turned over the next rock, they found the acoustic problem under it. Phenomenon No. 5.

Acoustics

We found that most acoustic emanations are difficult to exploit if the microphonic device is outside of the room containing the source equipment; even a piece of paper inserted between, say, an offending keyboard and a pick-up

Past – earliest days

Side Channels always existed

First scientific observations in 1943

Concept of “covert channels” in 1973

Operating Systems
C. Weissman Editor

A Note on the Confinement Problem

Butler W. Lampson
Xerox Palo Alto Research Center

This note explores the problem of confining a program during its execution so that it cannot transmit information to any other program except its caller. A set of examples attempts to stake out the boundaries of the problem. Necessary conditions for a solution are stated and informally justified.

Communications of the ACM
October 1973 Volume 16 Number 10

Past – earliest days

Side Channels always existed

First scientific observations in 1943

Concept of “covert channels” in 1973

1974-1980: Provable secure operating systems with exceptions for side channels

1985: Orange book. Covert channels with low bandwidth not a problem

1996: Paul Kocher’s seminal work on timing attacks

FIGURE 1: RSAREF Modular Multiplication Times

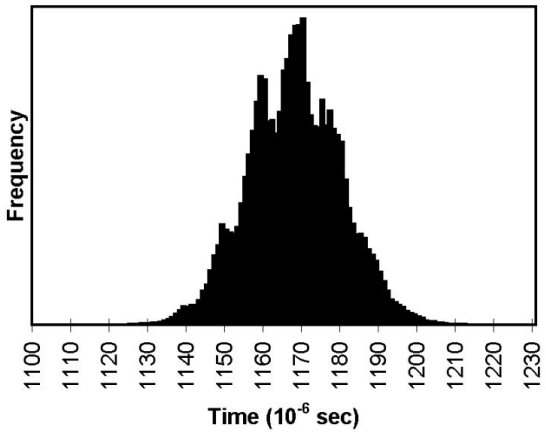
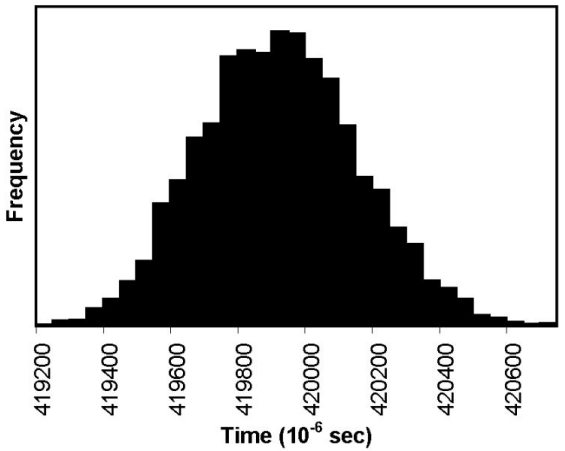
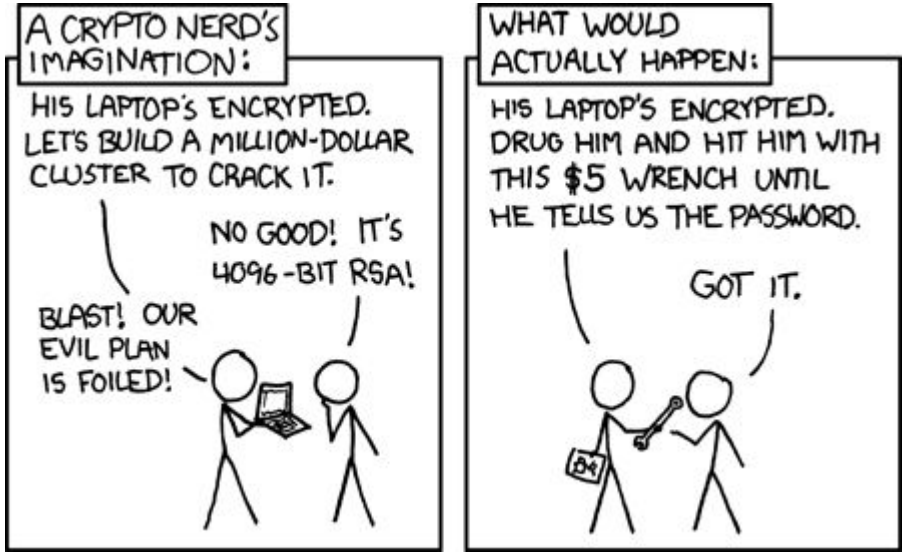


FIGURE 2: RSAREF Modular Exponentiation Times



Past: cryptographic attacks

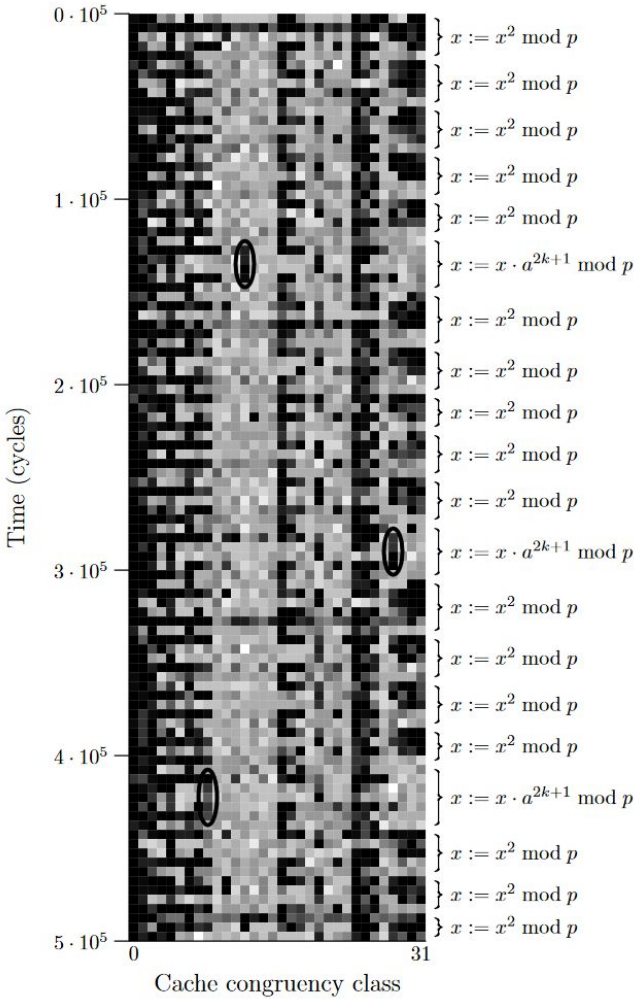
1996-2015 Mainly side channels on cryptography (threat model!)



Past: cryptographic attacks

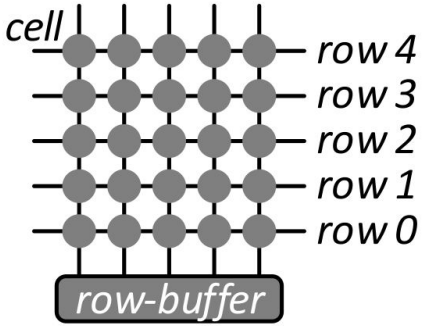
1996-2015 Mainly side channels on cryptography (threat model!)

Colin Percival (2005): “Cache Missing for fun and profit”

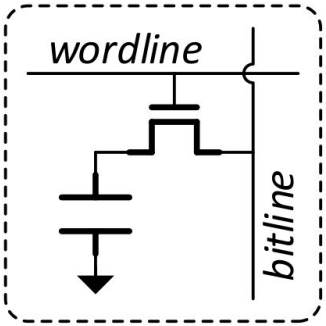


Past:
Moving beyond crypto

ISCA 2014 + BlackHat US 2015:
Rowhammer



a. Rows of cells

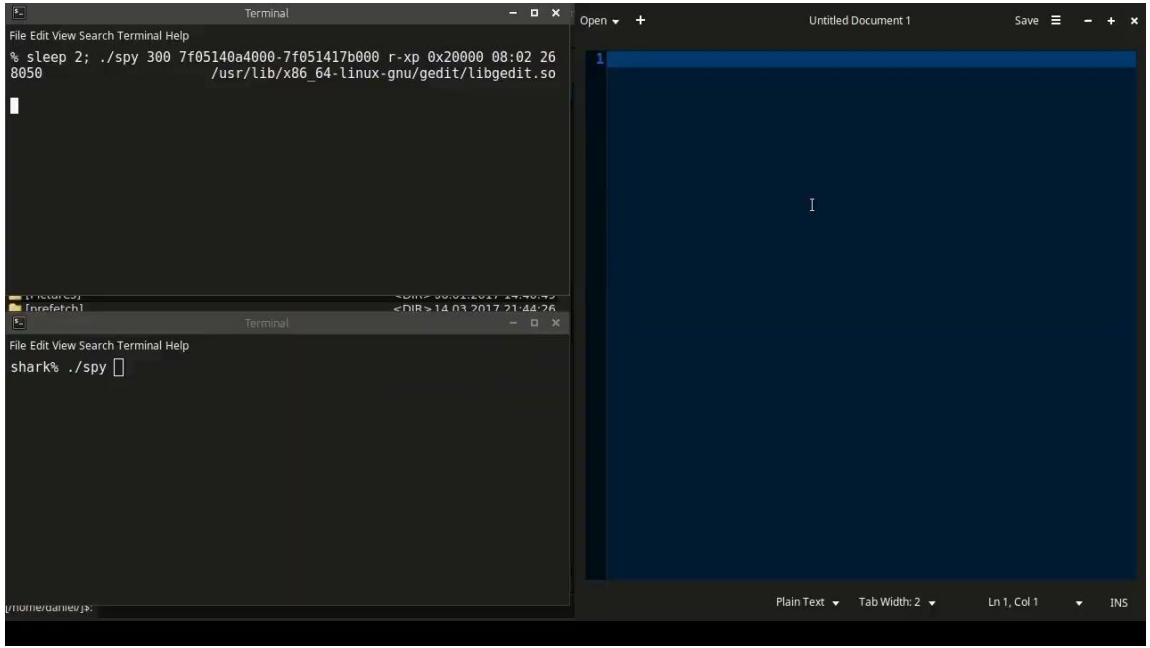


b. A single cell

Past:
Moving beyond crypto

ISCA 2014 + BlackHat US 2015:
Rowhammer

USENIX Security 2015:
Cache Template Attacks



Past:
Moving beyond crypto

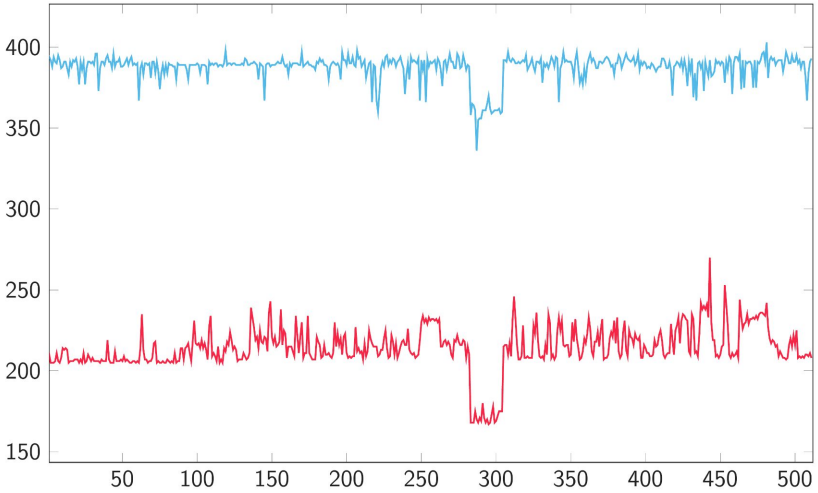
ISCA 2014 + BlackHat US 2015:
Rowhammer

USENIX Security 2015:
Cache Template Attacks

CCS + BlackHat US 2016:
Breaking KASLR

Breaking Kernel Address Space Layout Randomization with Intel TSX

Yeongjin Jang, Sangho Lee, and Taesoo Kim
Georgia Institute of Technology



**Prefetch Side-Channel Attacks:
Bypassing SMAP and Kernel ASLR**

Daniel Gruss* Clémentine Maurice* Anders Fogh†
Moritz Lipp* Stefan Mangard*
* Graz University of Technology † G DATA Advanced Analytics

Past:
Moving beyond crypto

ISCA 2014 + BlackHat US 2015:
Rowhammer

USENIX Security 2015:
Cache Template Attacks

CCS + BlackHat US 2016:
Breaking KASLR

2017: Many academic works on **attacking TEEs with side channels**

CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache Attacks

Fergus Dall[1], Gabrielle De Micheli[2], Thomas Eisenbarth[3,4], Daniel Genkin[2,5], Nadia Heninger[2], Ahmad Moghimi[4] and Yuval Yarom[1,6]

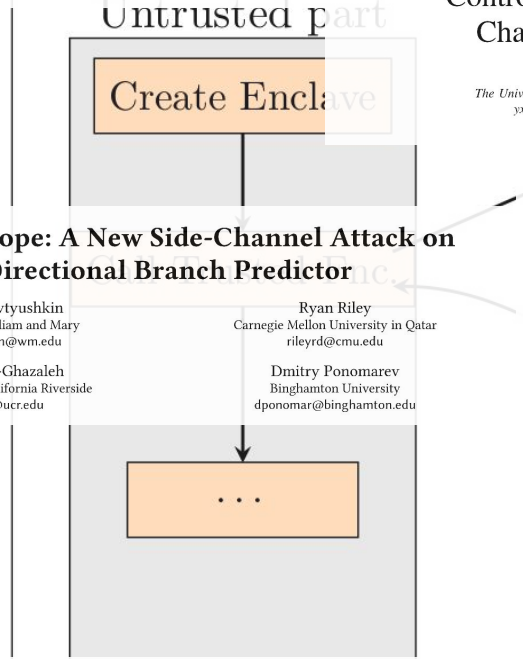
BranchScope: A New Side-Channel Attack on Directional Branch Predictor

Dmitry Evtyushkin
College of William and Mary
devtyushkin@wm.edu

Ryan Riley
Carnegie Mellon University in Qatar
rileyrd@cmu.edu

Nael Abu-Ghazaleh
University of California Riverside
naelag@ucr.edu

Dmitry Ponomarev
Binghamton University
dponomar@binghamton.edu



Application

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems

Yuanzhong Xu
The University of Texas at Austin
yxu@cs.utexas.edu

Weidong Cui
Microsoft Research
wdcui@microsoft.com

Marcus Peinado
Microsoft Research
marcuspe@microsoft.com

CopyCat: Controlled Instruction-Level Attacks on Encla

Daniel Moghimi[1], Jo Van Bulck[2], Nadia Heninger[3], Frank Piessens[2], and Ber

[1]Worcester Polytechnic Institute, Worcester, MA, USA
[2]imec-DistriNet, KU Leuven, Leuven, Belgium
[3]University of California, San Diego, CA, USA

How Trusted Executi
Environments Fuel P
on Microarchitectura

Session K4: Secure Enclaves

CCS'17, October 30-November 3, 2017, Dallas, TX, U

CacheZoom: How SGX Amplifies The Power of Cache Attacks

Ahmad Moghimi
Worcester Polytechnic Institute
amoghimi@wpi.edu

Gorka Irazoqui
Worcester Polytechnic Institute
girazoki@wpi.edu

Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX

Wenhao Wang[1], Guoxing Chen[3], Xiaorui Pan[2], Yinqian Zhang[3], XiaoFeng Wang[2], Vincent Bindschaedler[4], Haixu Tang[2], Carl A. Gunter[4]*
[1]SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences & Indiana University Bloomington

Past:
Moving beyond crypto

ISCA 2014 + BlackHat US 2015:
Rowhammer

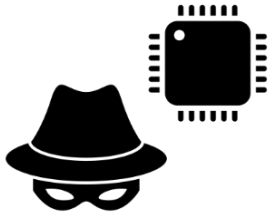
USENIX Security 2015:
Cache Template Attacks

CCS + BlackHat US 2016:
Breaking KASLR

2017: Many academic works on **attacking TEEs with side channels**

USENIX + BlackHat US 2018, S&P 2019:
Spectre & Meltdown

1 preface



architectural

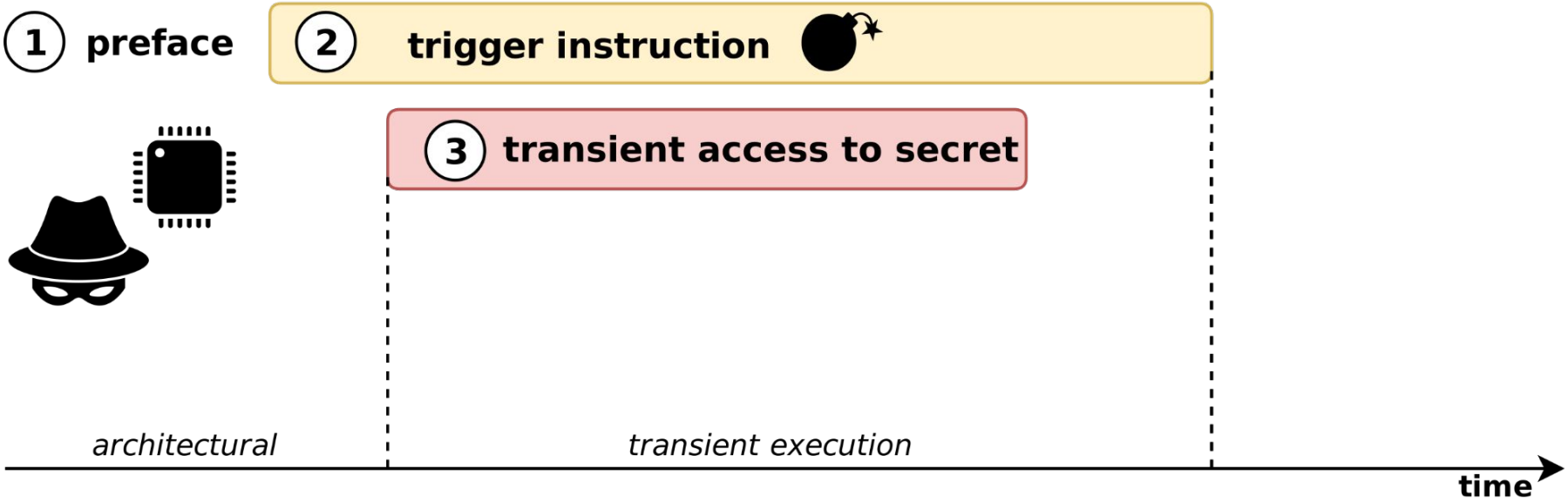
time

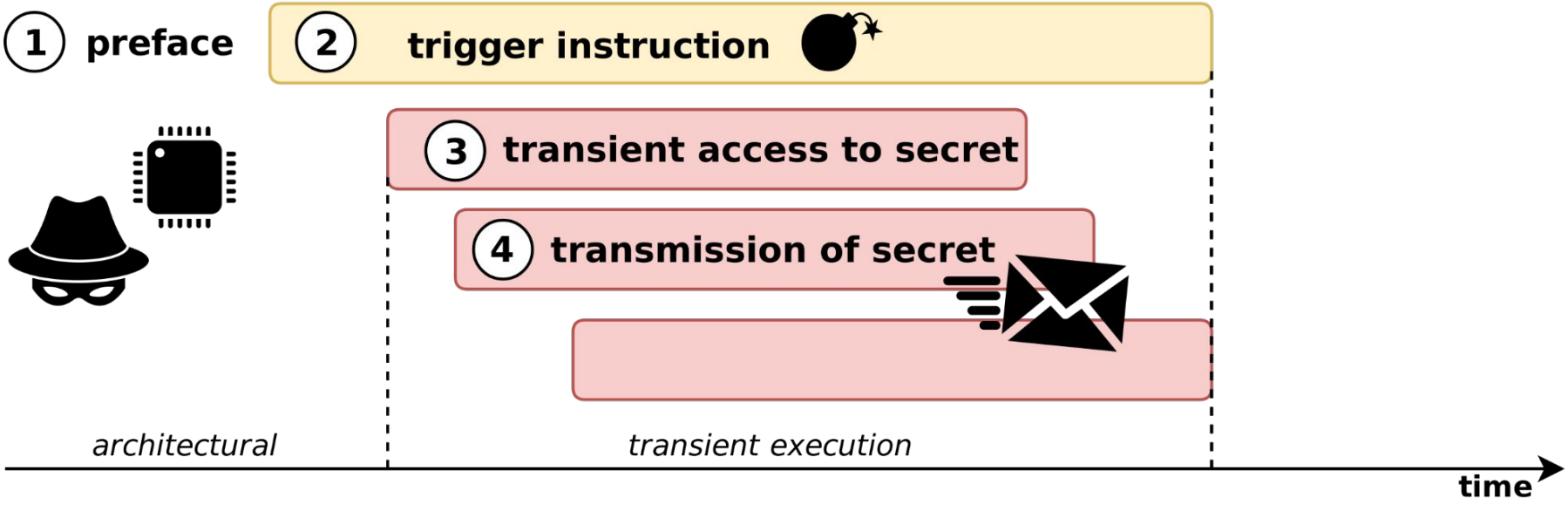
1 **preface**

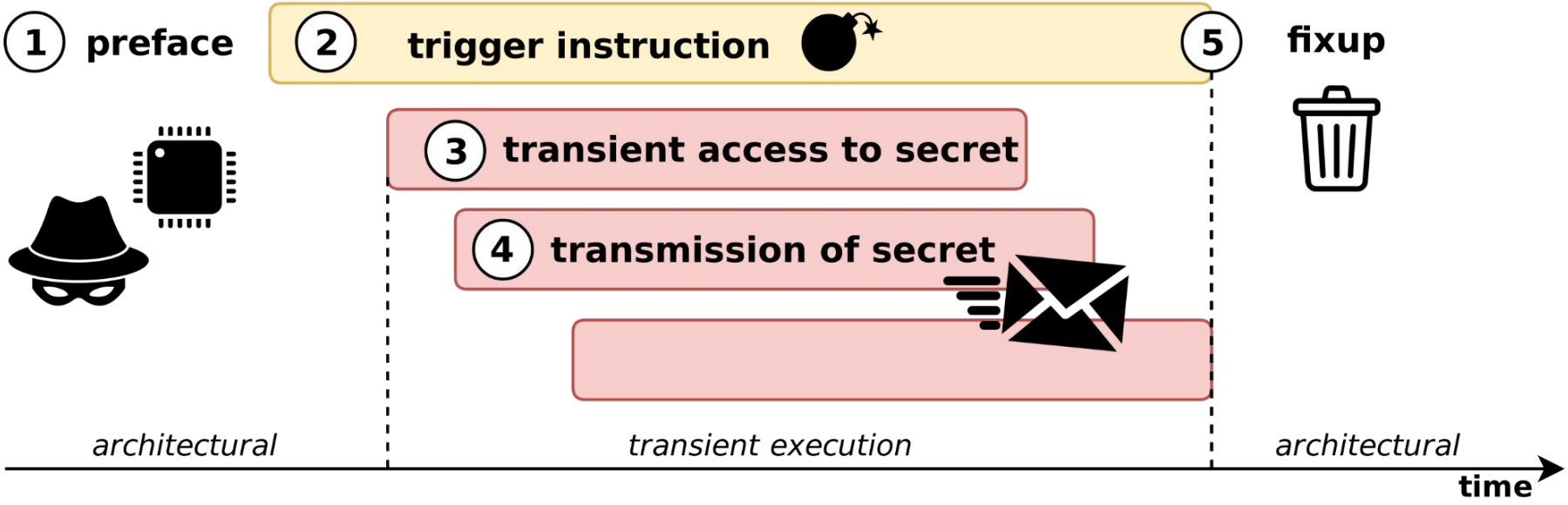
2 **trigger instruction**

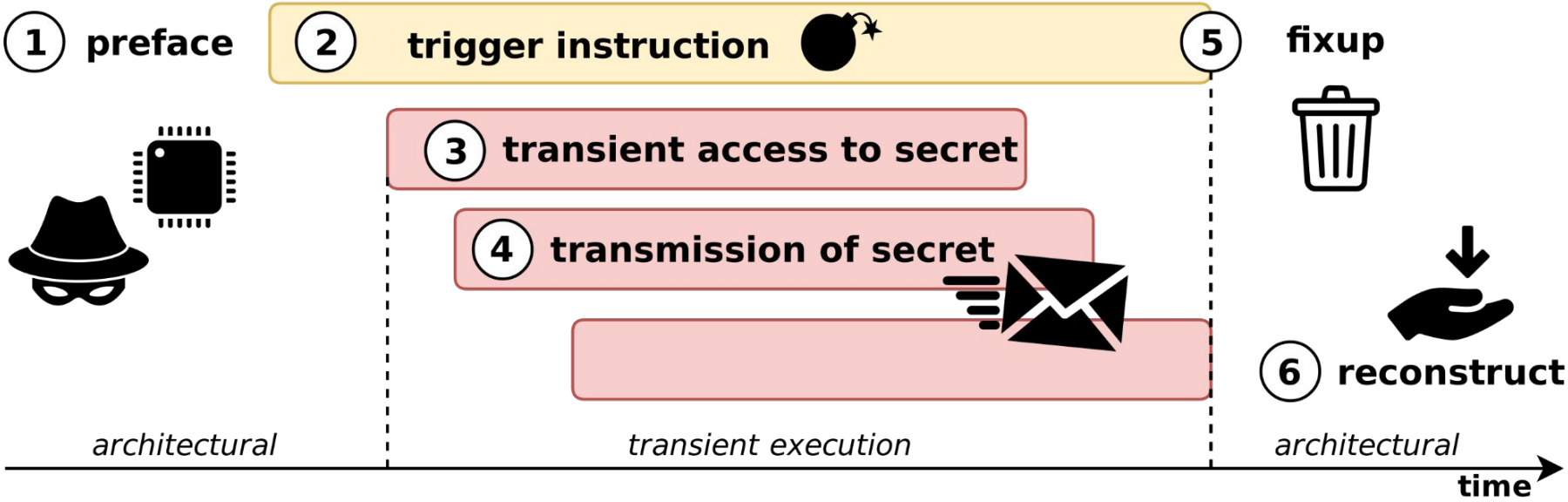
architectural *transient execution*

time

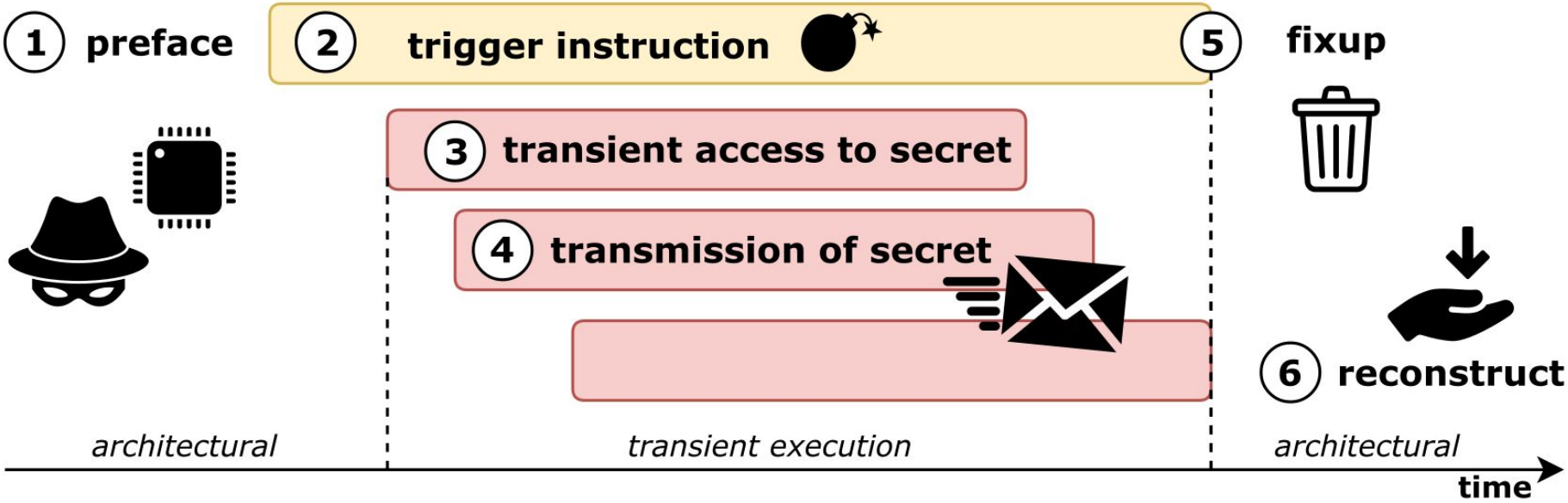

1 preface
2 trigger instruction
3 transient access to secret
architectural
transient execution
time

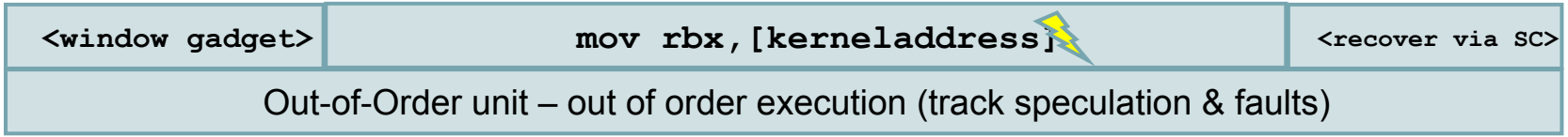

1 preface
2 trigger instruction
3 transient access to secret
4 transmission of secret
architectural
transient execution
time


1 preface
2 trigger instruction
5 fixup
3 transient access to secret
4 transmission of secret
architectural
transient execution
architectural
time

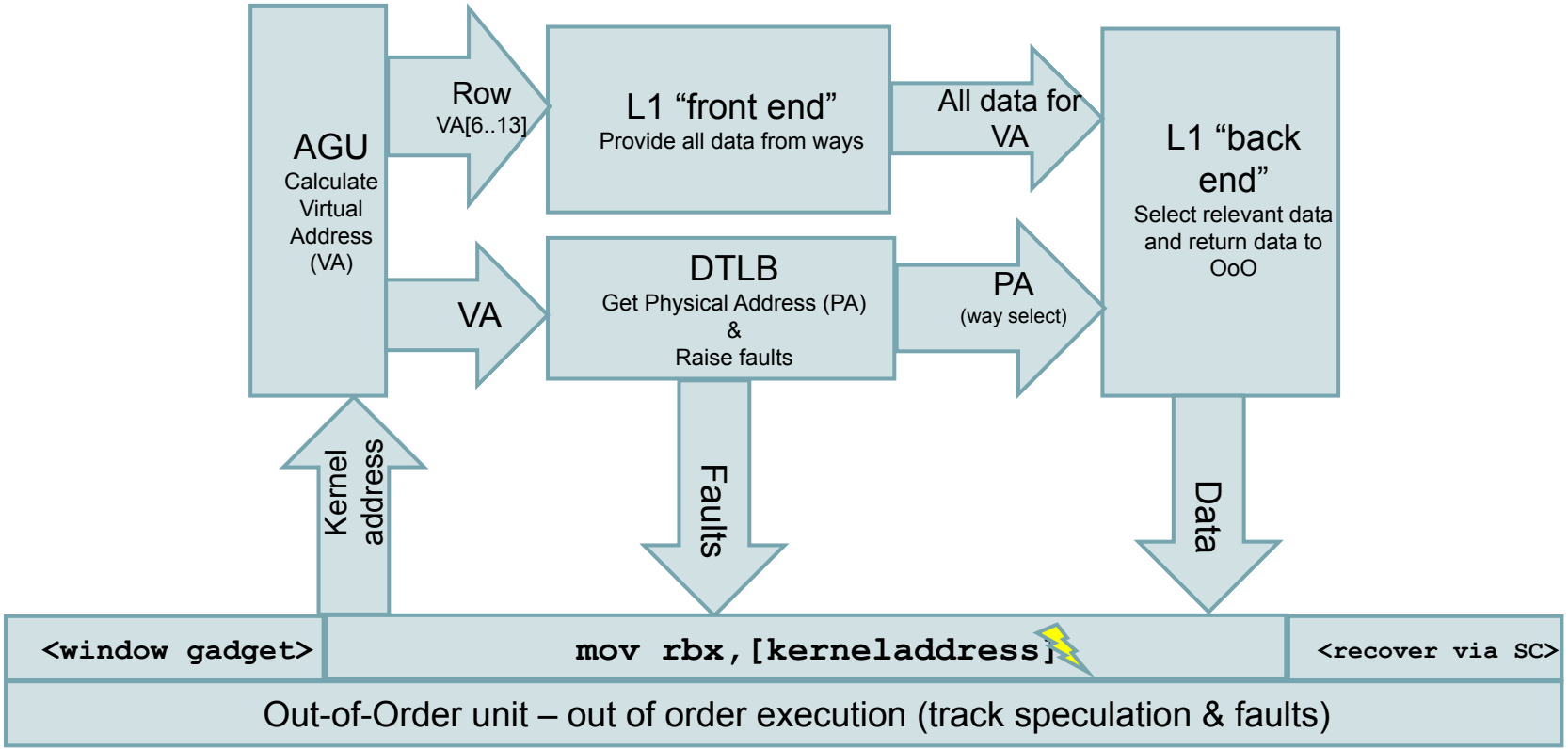

1 preface
2 trigger instruction
5 fixup
3 transient access to secret
4 transmission of secret
6 reconstruct
architectural
transient execution
architectural
time

Past: Meltdown



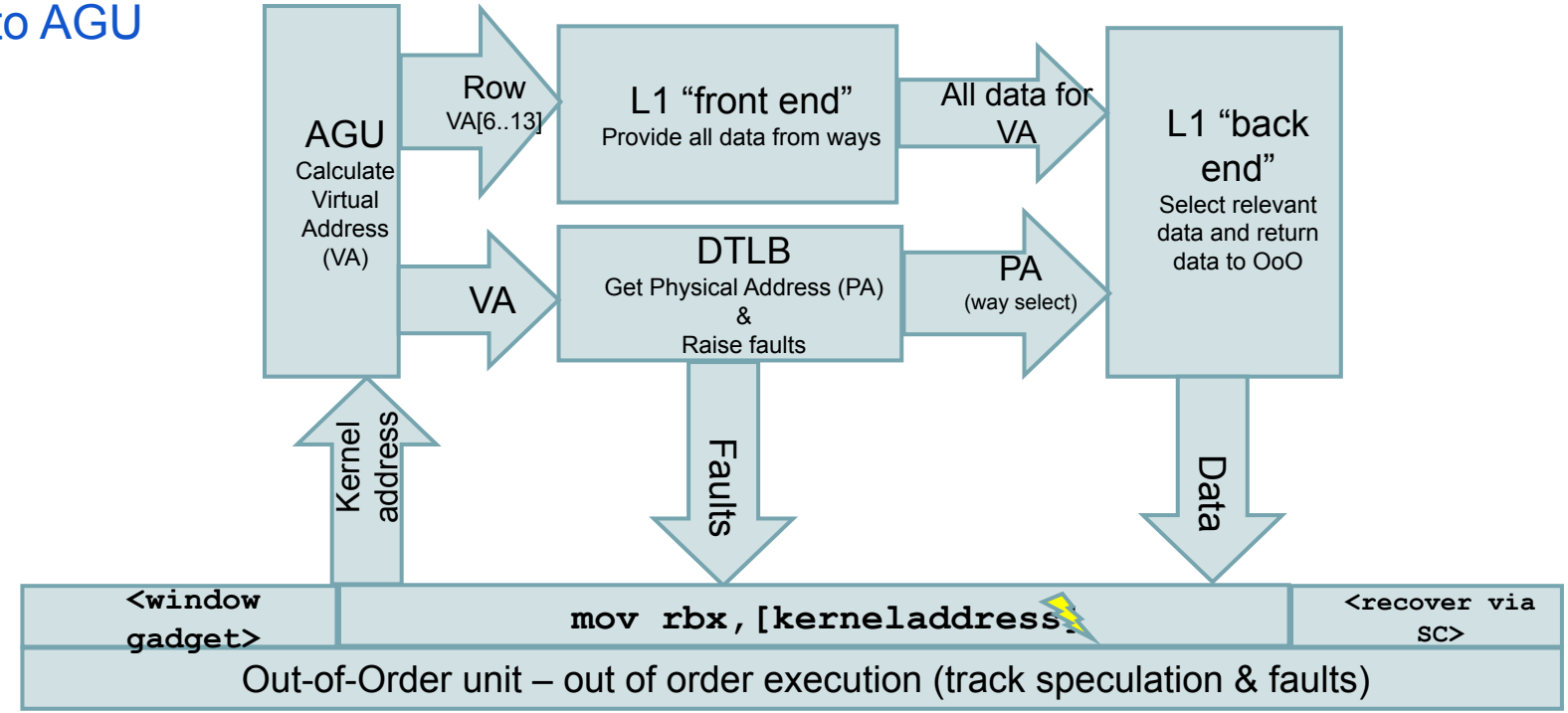


Meltdown: Details



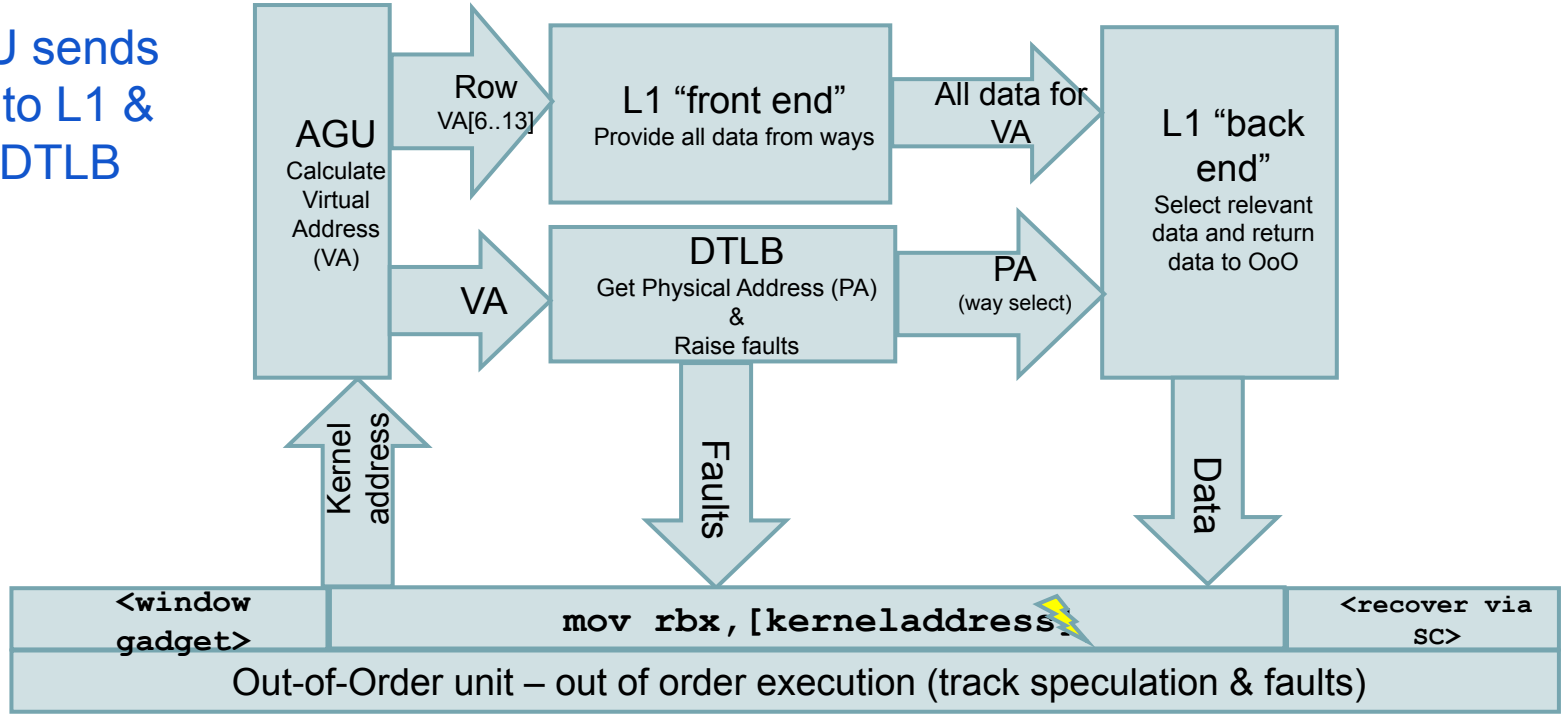
Meltdown: Details

1. OoO Trigger load to AGU



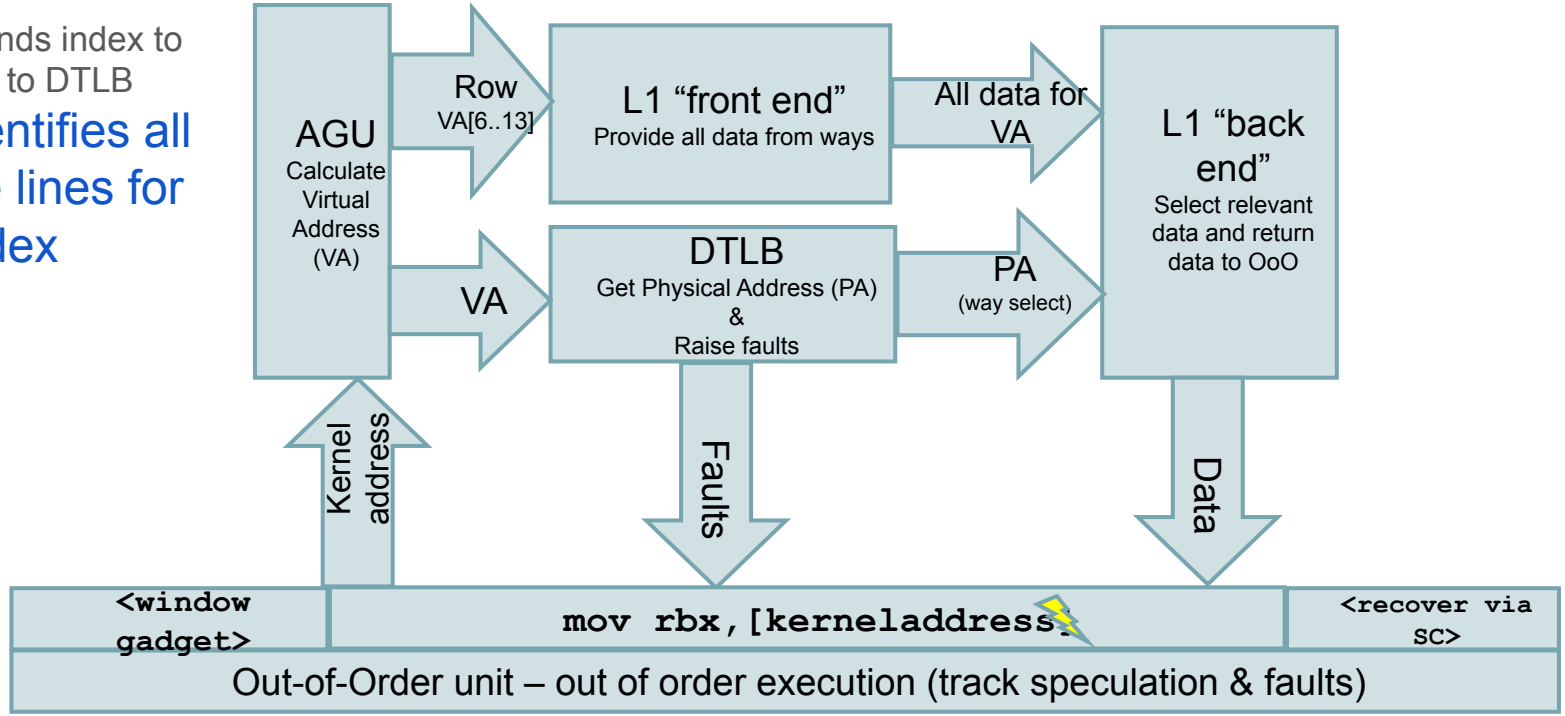
Meltdown: Details

1. 1.OoO Trigger load to AGU
2. 2.AGU sends index to L1 & VA to DTLB



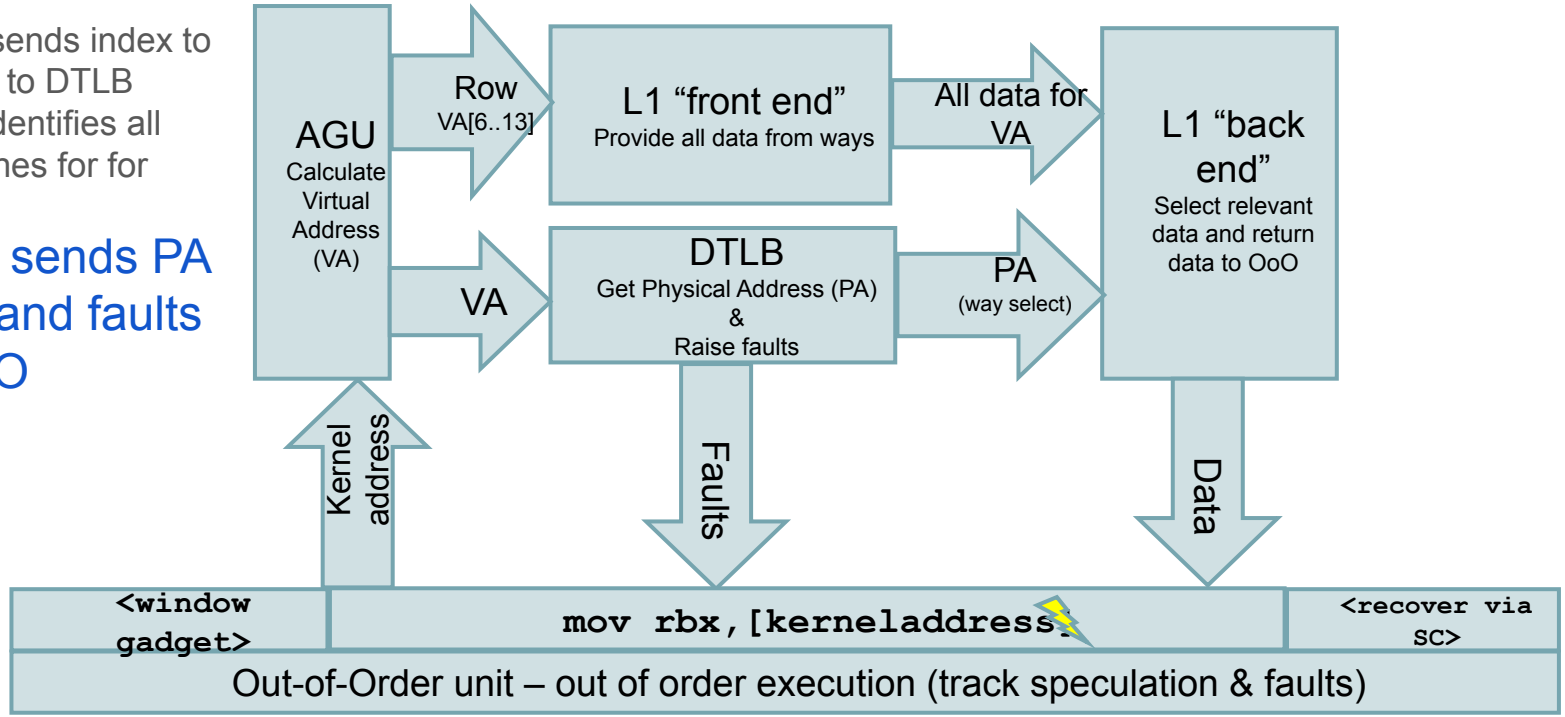
Meltdown: Details

1. OoO Trigger load to AGU
2. AGU sends index to L1 & VA to DTLB
3. L1 identifies all cache lines for for index



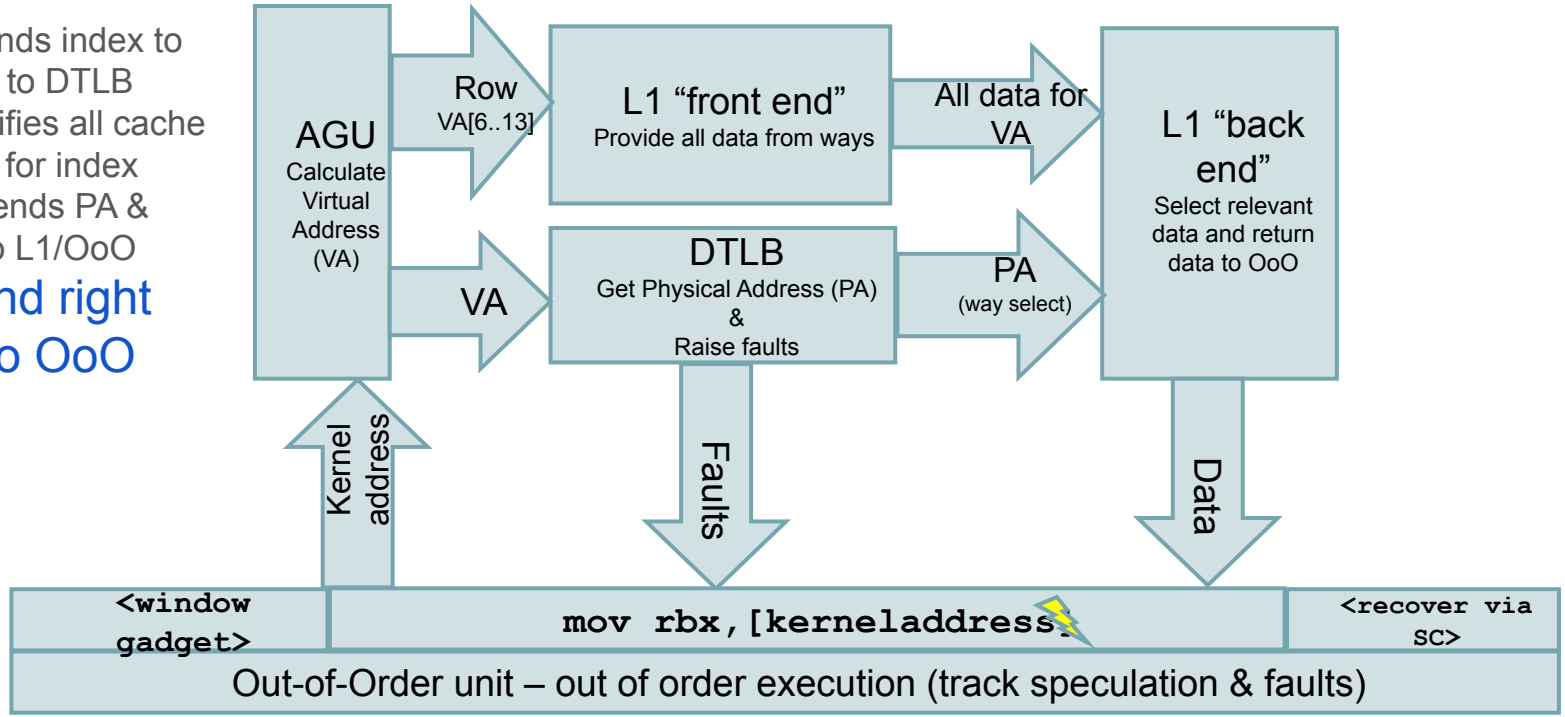
Meltdown: Details

1. 1.OoO Trigger load to AGU
2. 2.AGU sends index to L1 & VA to DTLB
3. 3.a L1 identifies all cache lines for for index
4. DTLB sends PA to L1 and faults to OoO



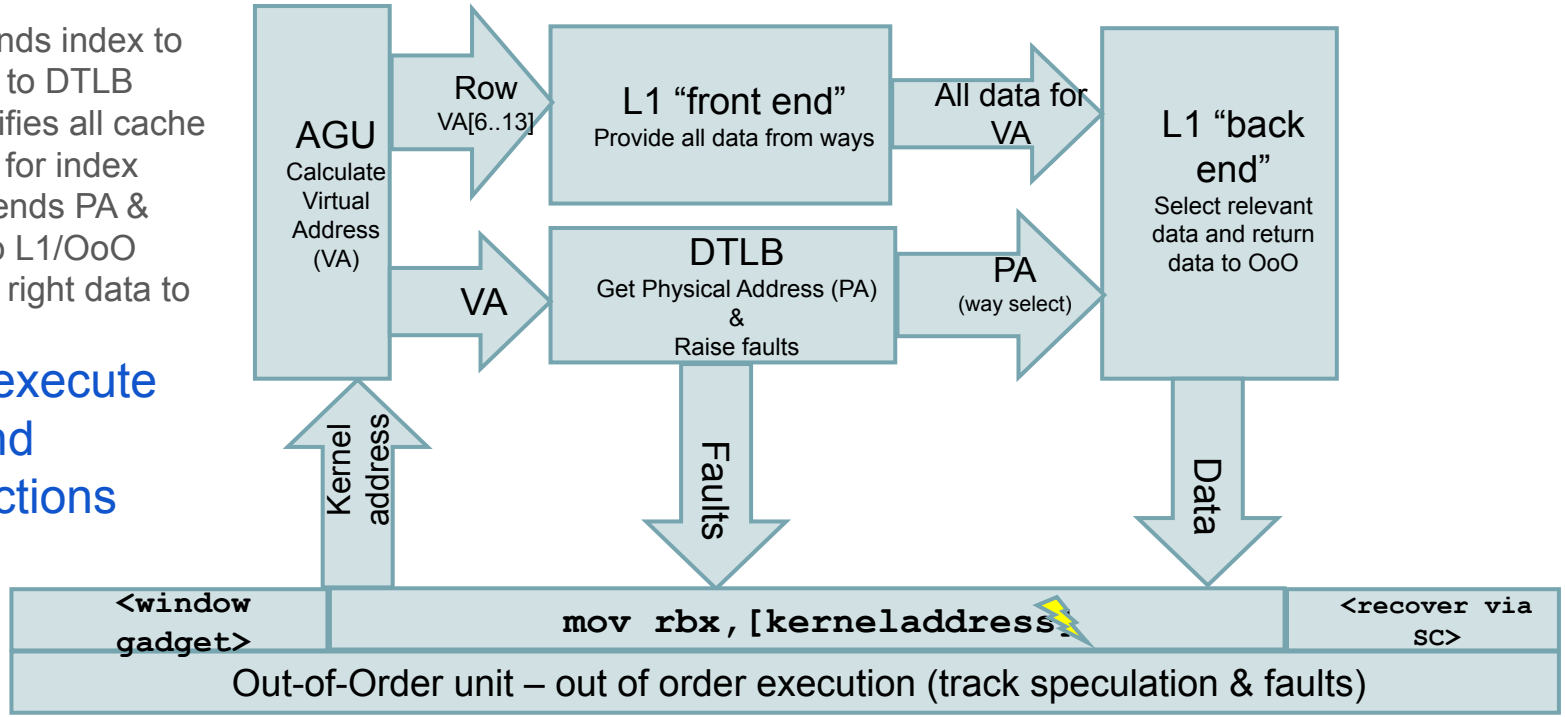
Meltdown: Details

1. OoO Trigger load to AGU
2. AGU sends index to L1 & VA to DTLB
3. L1 identifies all cache lines for for index
4. DTLB sends PA & faults to L1/OoO
5. L1 send right data to OoO

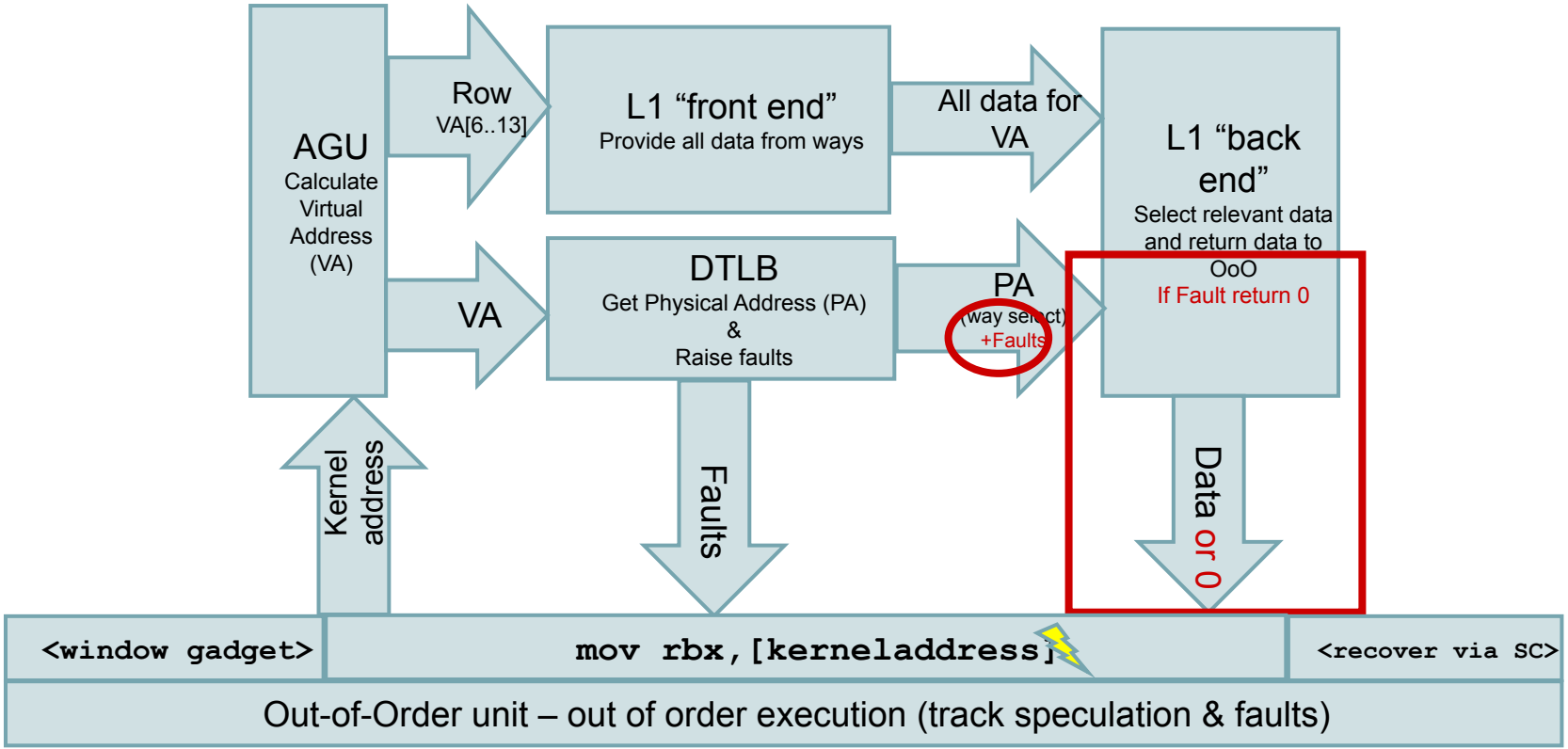


Meltdown: Details

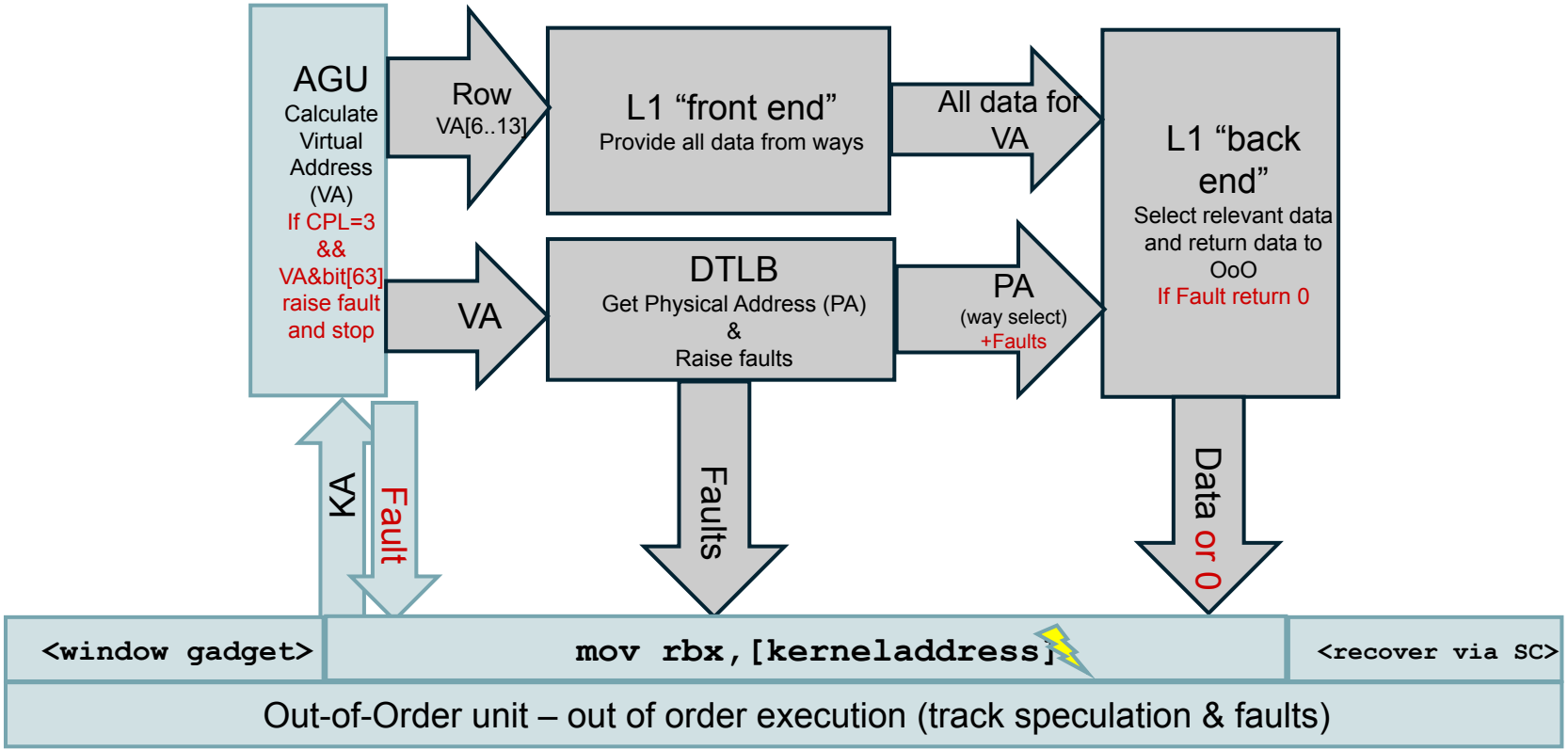
1. OoO Trigger load to AGU
2. AGU sends index to L1 & VA to DTLB
3. L1 identifies all cache lines for for index
4. DTLB sends PA & faults to L1/OoO
5. L1 send right data to OoO
6. OoO execute depend instructions



The First Meltdown Mitigations



Meltdown defense in depth (LASS)



Spectre and LVI

μ-Arch Buffer \ Methodology		Leakage	Injection
Prediction history	PHT	BranchScope [79], Bluethunder [131]	Spectre-PHT [174]
	BTB	SBPA [8], BranchShadow [182]	Spectre-BTB [174]
	RSB	Hyper-Channel [46]	Spectre-RSB [177, 200]
	STL	—	Spectre-STL [128]
Program data	NULL	EchoLoad [49]	LVI-NULL [311]
	L1D	Meltdown [193], Foreshadow [310]	LVI-L1D [311]
	FPU	LazyFP [291]	LVI-FPU [311]
	SB	Store-to-Leak [270], Fallout [48]	LVI-SB [311]
	LFB/LP	ZombieLoad [276], RIDL [267]	LVI-LFB/LP [311]


Present

Present: Trends

Attack type	Activity level	(Point) Mitigation	Notable
Crypto side channels		Guidance & DOIT	Data dependent features for example data dependent prefetchers
Transient execution vulnerabilities		Hardware + Software +on/off switches Workarounds	Predictive store forwarding
Stale data vulnerabilities		Microcode Patches or SW Mitigation (if possible)	Not any recent attacks
Logical bugs		Microcode Patches (if possible)	Reptar, CacheWarp
Physical properties			Hertzbleed, Collide+Power
Exploitation methods			Spectre & Power

Logic Issues

Reptar - What's supposed to happen

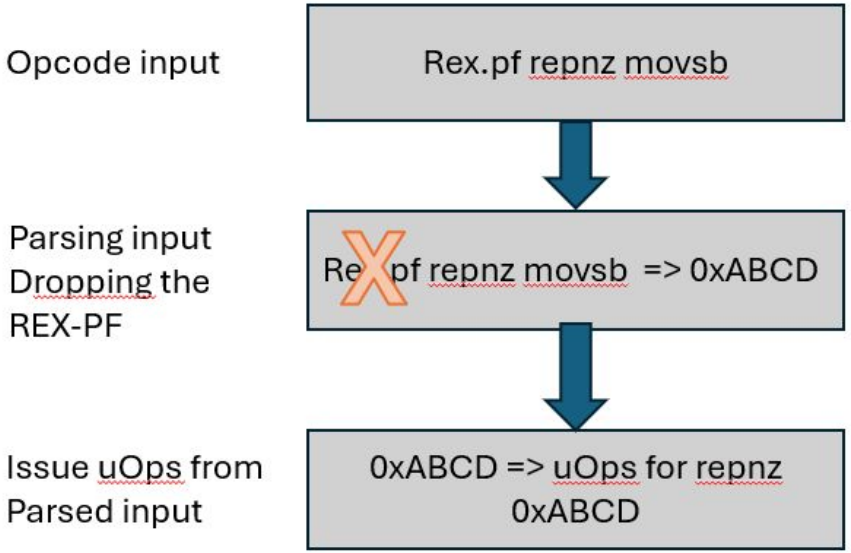
REPNZ is a prefix that will repeat an operation until the Z-flag becomes zero.

MOVSB will copy a single byte from DS:[RSI] to ES:[RDI] and increment both registers and decrement RCX & update flags.

REPNZ MOVSB is thus a simple memcpy.

The REX-prefix (REX.PF) changes the meaning of how explicit operands of an instruction are interpreted. MOVSB doesn't have any explicit operands.

If you use the REX-prefix with REPNZ MOVSB the CPU should ignore the prefix entirely



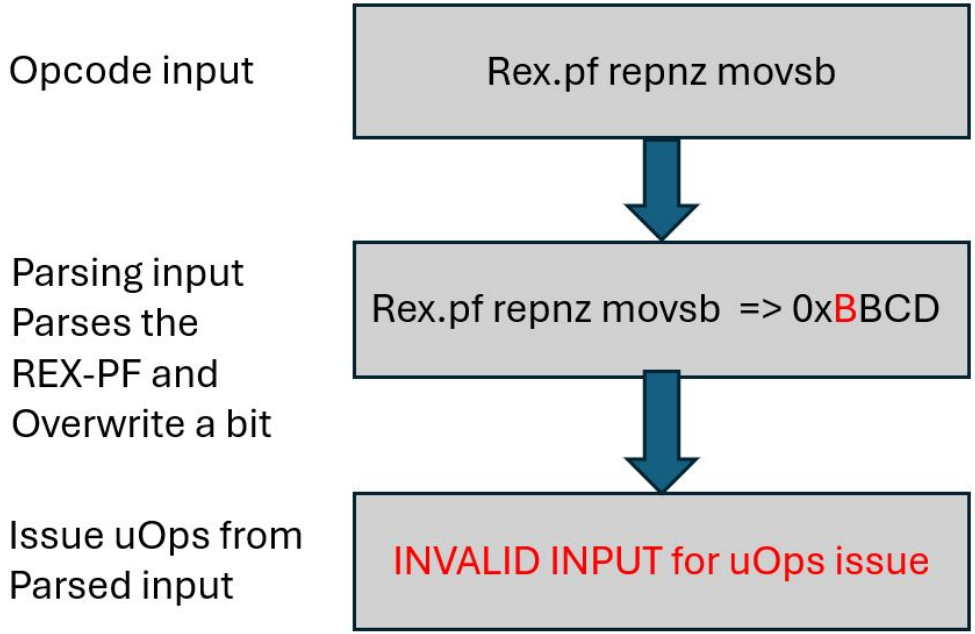
Reptar - The bug

When the REX-prefix is parsed instead of ignored a single bit is overwritten.

This cause an invalid input to be used to generate uOps.

Under certain conditions this leads to a machine check. Careful analysis found that a condition could potentially lead to privilege escalation.

A microcode change that mitigates the issue has been made public.



Cachewarp

Confidential VM (encrypted but basically no data integrity)

invd instruction can invalidate a single cache line

Attack in three steps:

1. let confidential VM modify a target cache line
2. use **invd** to drop the modification
3. confidential VM continues with an outdated value

CacheWarp: Software-based Fault Injection using Selective State Reset

Ruiyi Zhang
CISPA Helmholtz Center for Information Security

Lukas Gerlach
CISPA Helmholtz Center for Information Security

Daniel Weber
CISPA Helmholtz Center for Information Security

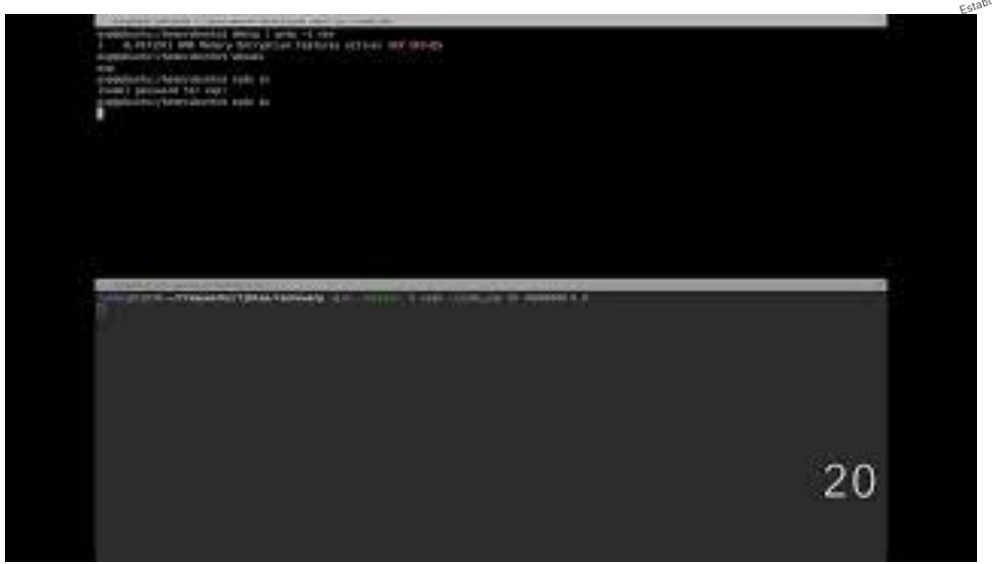
Lorenz Hetterich
CISPA Helmholtz Center for Information Security

Youheng Lü
Independent

Andreas Kogler
Graz University of Technology

Michael Schwarz
CISPA Helmholtz Center for Information Security

fssec erc
European Research Council
Established by the European Commission



Zenbleed

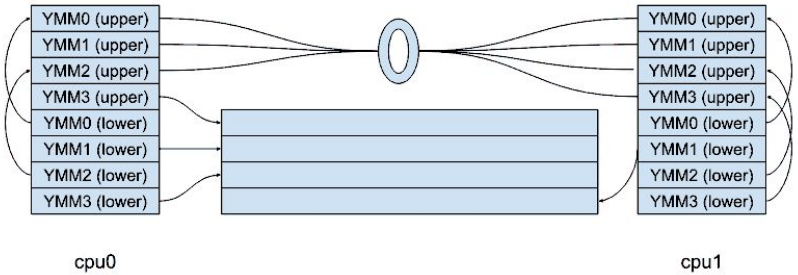
Register names are just for the user, CPU uses register file

XMM Register Merge Optimization: merge registers (e.g. zero registers)

also: for zero just set a zero-bit

Zenbleed:

1. misspeculation
2. **vzeroupper** → set zero-bit
3. merge → storage in register file released
4. victim stores data in this register
5. unroll misspeculation
6. architectural access to a victim data



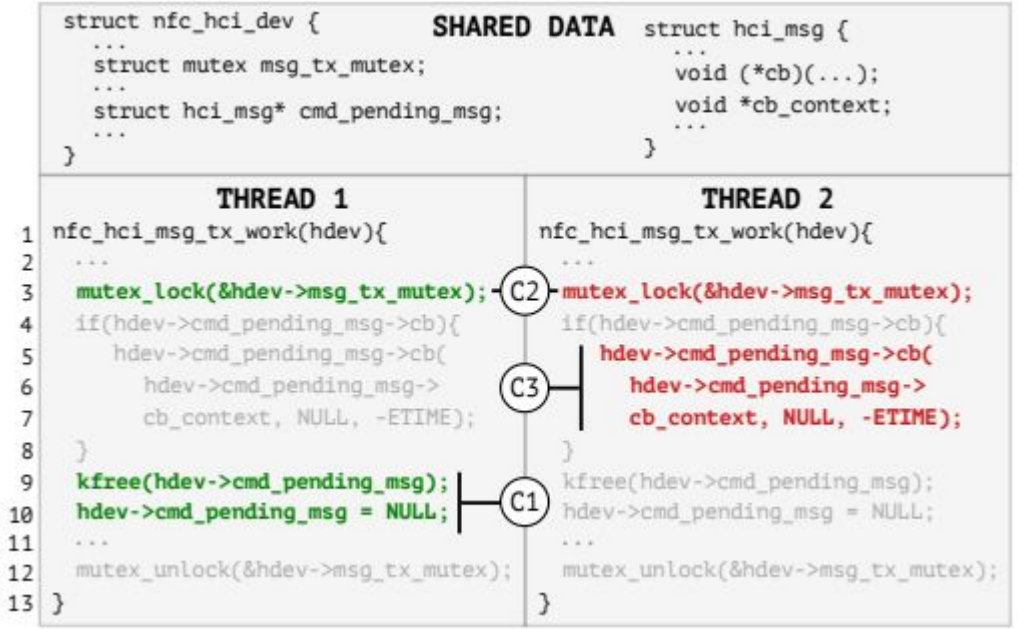
Exploitation Techniques

Exploitation techniques - example

GhostRace: Exploiting and Mitigating Speculative Race Conditions - Hany Ragab et. al.

Spectre v1. variant that speculatively bypasses synchronization primitives.

Existing methods of mitigating Spectre v1 remain effective.

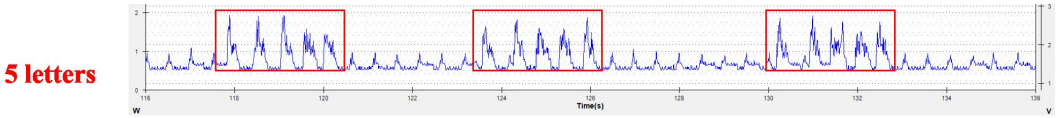


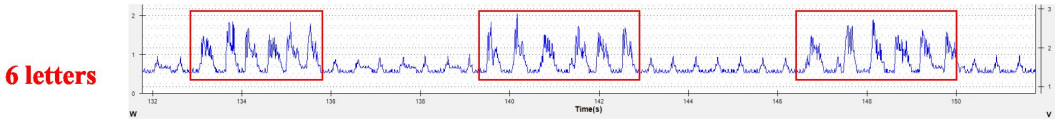
Quote from the papers abstract:
"*There's is security, and then there's just being ridiculous*" - Linus Torvalds, on Speculative Race Conditions

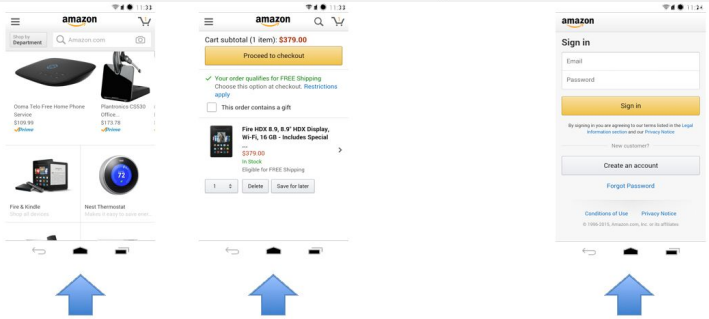
Physical Domain in Software

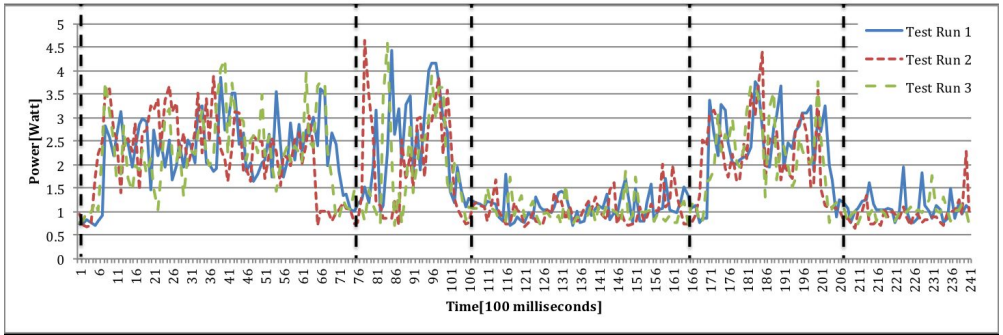
Software-based Power Analysis

before 2020: mainly fingerprinting









Software-based Power Analysis

before 2020: mainly fingerprinting

2020: Platypus
full recovery of cryptographic keys

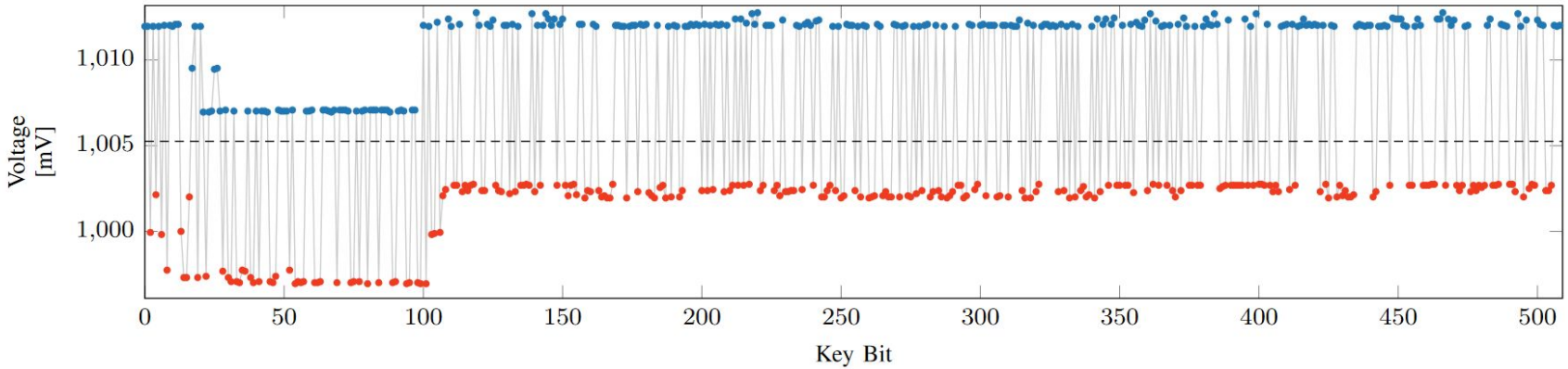


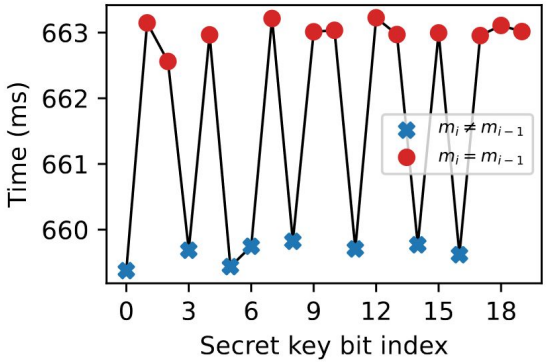
Fig. 13: Core voltage per measured instruction for each key bit offset in the fixed window length implementation of mbed TLS inside an SGX enclave on the Xeon E3-1275 v5. The blue marks represent 1 bits, while the red marks represent 0 bits. Using a threshold (dashed line), they can easily be distinguished.

Software-based Power Analysis

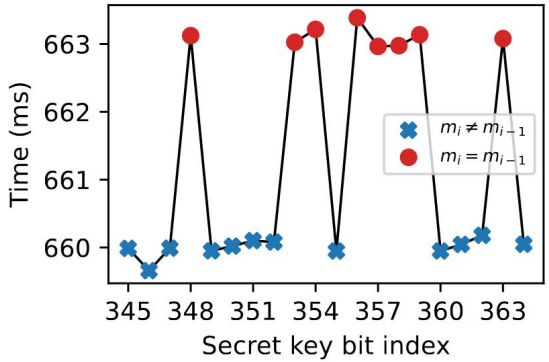
before 2020: mainly fingerprinting

2020: Platypus
full recovery of cryptographic keys

2023: Hertzbleed
DVFS makes timing a proxy for energy consumption → remote attacks



(a) CIRCL first 20 bits



(b) CIRCL last 20 bits

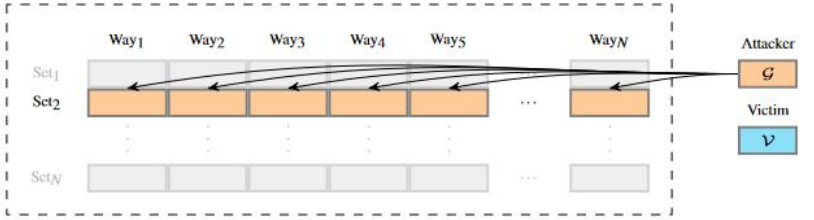
Software-based Power Analysis

before 2020: mainly fingerprinting

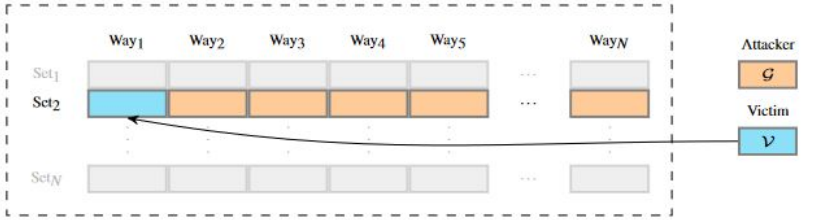
2020: Platypus
full recovery of cryptographic keys

2023: Hertzbleed
DVFS makes timing a proxy for energy consumption → remote attacks

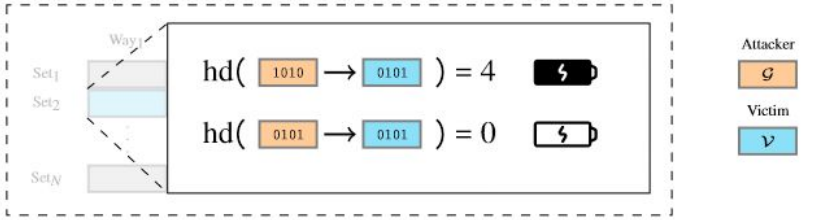
2023: Collide+Power
Generic Attacks (not just crypto)



(a) **Step 1:** The attacker primes each cache line of the target cache set with the attacker-controlled guess $\mathcal{G}$.



(b) **Step 2:** The victim accesses the secret $\mathcal{V}$ and forces a cache line to change from $\mathcal{G}$ to $\mathcal{V}$.



(c) **Step 3:** The energy consumption during this change is proportional to the number of bit changes between $\mathcal{G}$ and $\mathcal{V}$.

Software-based Fault Attacks

since 2015: Rowhammer
still not solved!

ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms

Patrick Jattke† Max Wipfli† Flavien Solt Michele Marazzi Matej Bölcskei Kaveh Razavi

ETH Zurich

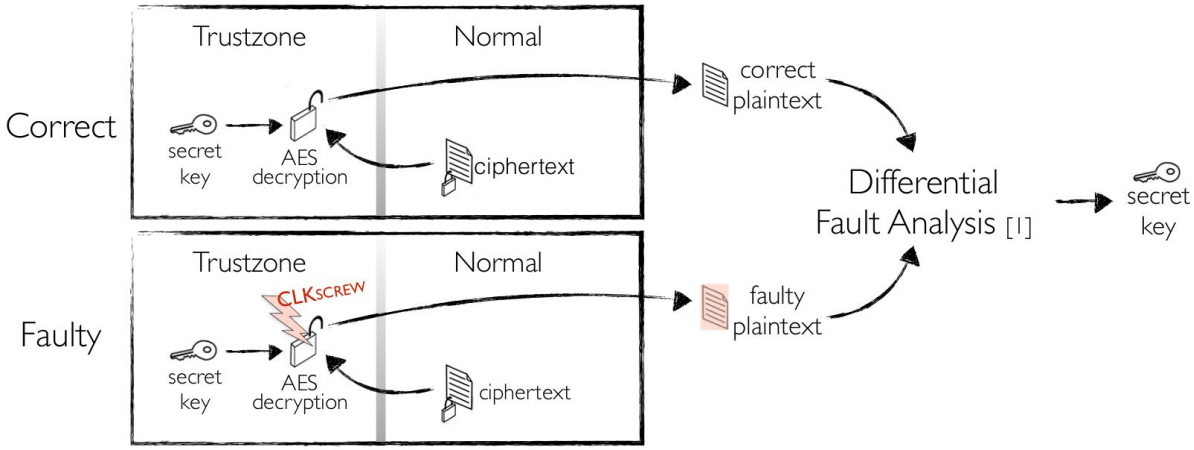
Table 10. Analysis of the bit flip exploitability found during the sweep over 256 MiB on AMD *Zen 2*, *Zen 3*, and Intel *Coffee Lake*. For each attack, we indicate the number of exploitable bit flips (#Ex.) and average time to find an exploitable bit flip (Time). We mark DIMMs with a single exploitable bit flip by (*). We omit DIMMs without any exploitable bit flips.

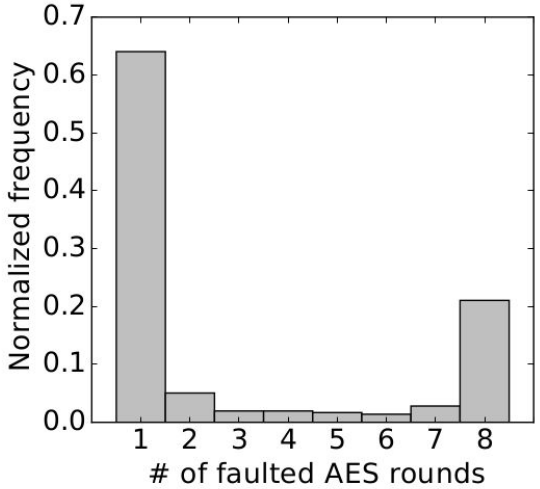
DIMM	PTE [36]						RSA-2048 [34]						sudo [11]					
	Zen 2		*Zen 3*		*Coffee Lake*		*Zen 2*		*Zen 3*		*Coffee Lake*		*Zen 2*		*Zen 3*		*Coffee Lake*	
	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	Time	#Ex.	T.	#Ex.	Time	#Ex.	Time
$\mathcal{S}_0$	7	6m 4s	7	2m 55s	3	4m 15s	17	2m 47s	37	46s	14	1m 36s	–	–	4	3m 13s	1	*23m 49s
$\mathcal{S}_1$	90	9s	1474	2s	846	2s	6	2m 2s	27	30s	21	26s	–	–	1	*6m 50s	1	*1m 20s
$\mathcal{S}_2$	641	21s	5326	1s	126	11s	30	2m 16s	170	6s	6	1m 59s	–	–	12	1m 17s	–	–
$\mathcal{S}_3$	142	9s	61	32s	–	–	7	2m 21s	–	–	–	–	–	–	–	–	–	–
$\mathcal{S}_4$	220	28s	3	23m 52s	2658	1s	7	12m 29s	1	*23m 52s	53	26s	–	–	–	–	4	5m 16s
$\mathcal{S}_5$	102	6s	625	2s	330	4s	6	1m 14s	28	33s	11	1m 5s	–	–	2	5m 58s	3	2m 34s
$\mathcal{H}_0$	11	53s	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

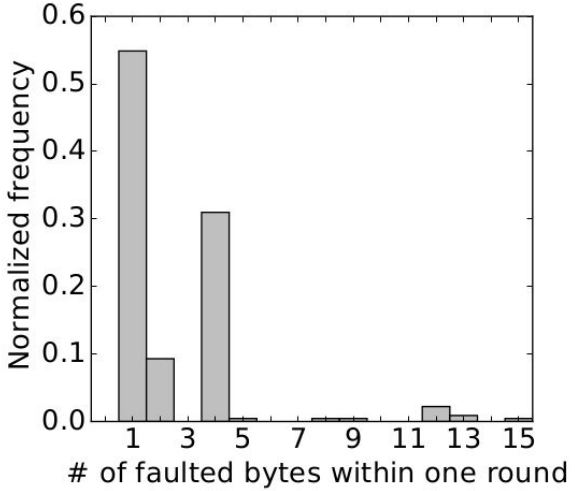
Software-based Fault Attacks

since 2015: Rowhammer
still not solved!

2017: CLKScrew
overclock and attack Arm TrustZone





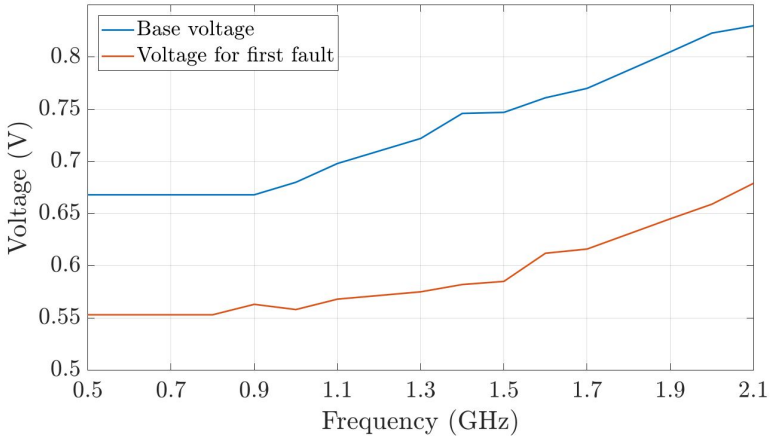


Software-based Fault Attacks

since 2015: Rowhammer
still not solved!

2017: CLKSkrew
overclock and attack Arm TrustZone

2020: Plundervolt (VoltJockey, V0ltpwn, VoltPillager)
undervolt and attack Intel SGX




```
Summary
-------------------------------------------------
Iterations:           10000000
Start Voltage:        -252
End Voltage:          0
Stop after x drops:   10
Voltage steps:        1
Threads:              4
Operand1:             0x00000000deadbeef
Operand2:             0x1122334455667788
Operand1 is:          fixed value
Operand2 is:          fixed value
```

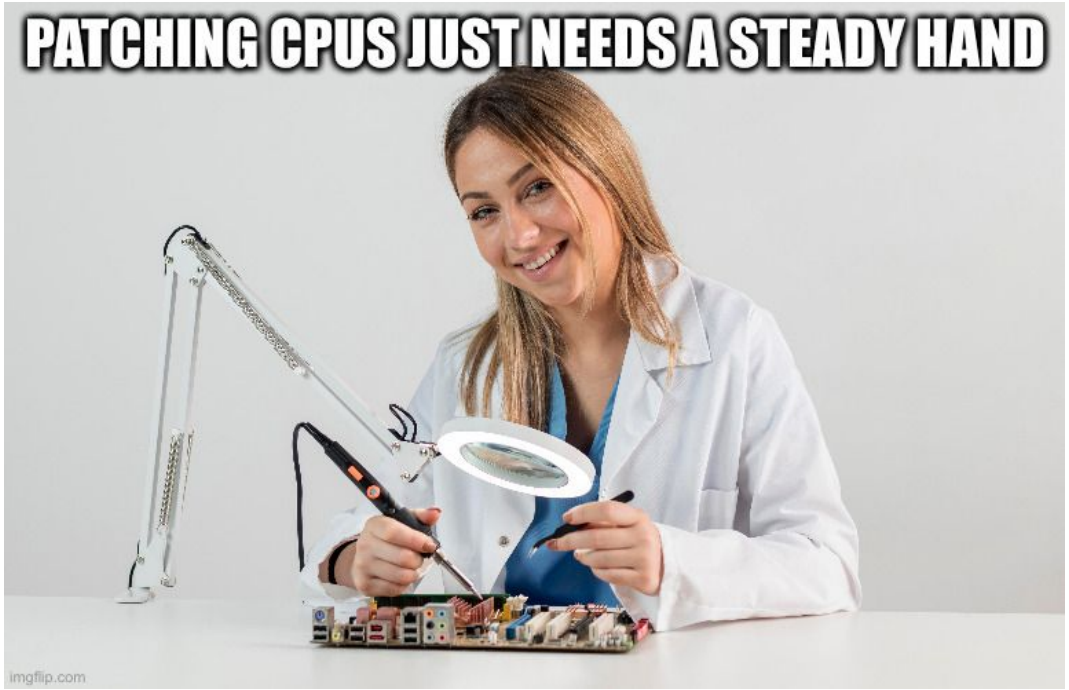
Mitigation efforts

Limitations of mitigations

Physical hardware cannot be changed in the field

Limitations of mitigations

Physical hardware cannot be changed in the field



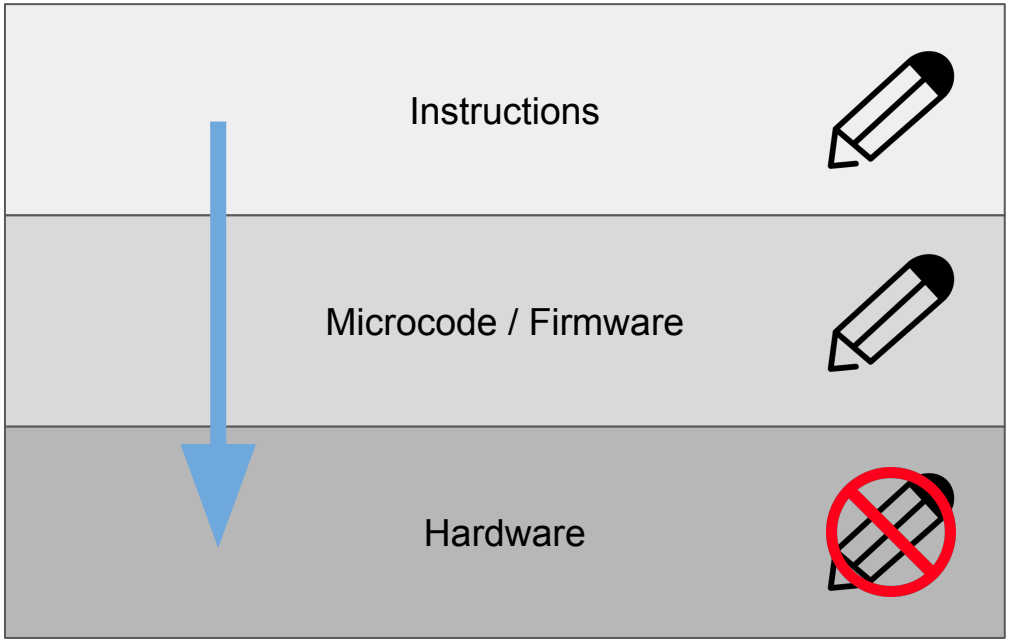
Limitations of mitigations

Physical hardware cannot be changed in the field

Vendors build in “Survivability features”

Microcode is the most common used tool for mitigations.

Other firmware is also used



Limitations of mitigations

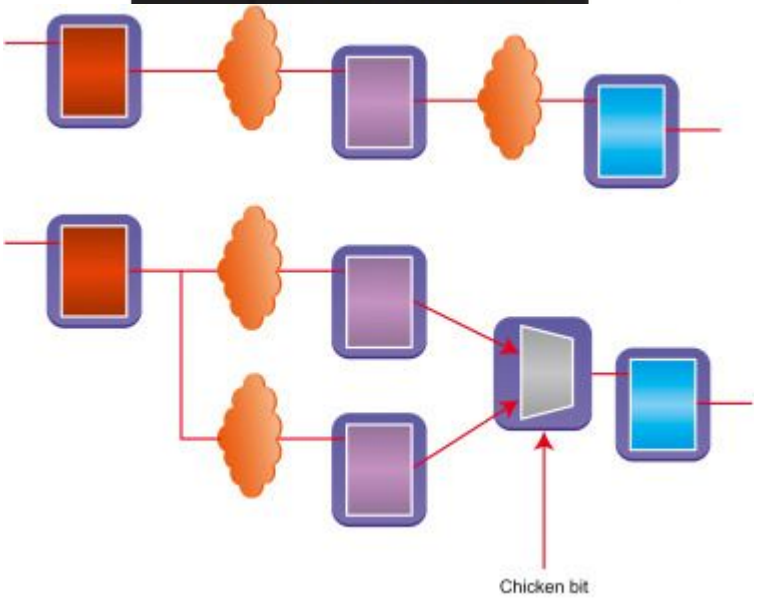
Physical hardware cannot be changed in the field

Vendors build in “Survivability features”

Microcode is the most common used tool for mitigations.

Other firmware is also used

“Chicken bits” to disable / change behavior



Limitations of mitigations

Physical hardware cannot be changed in the field

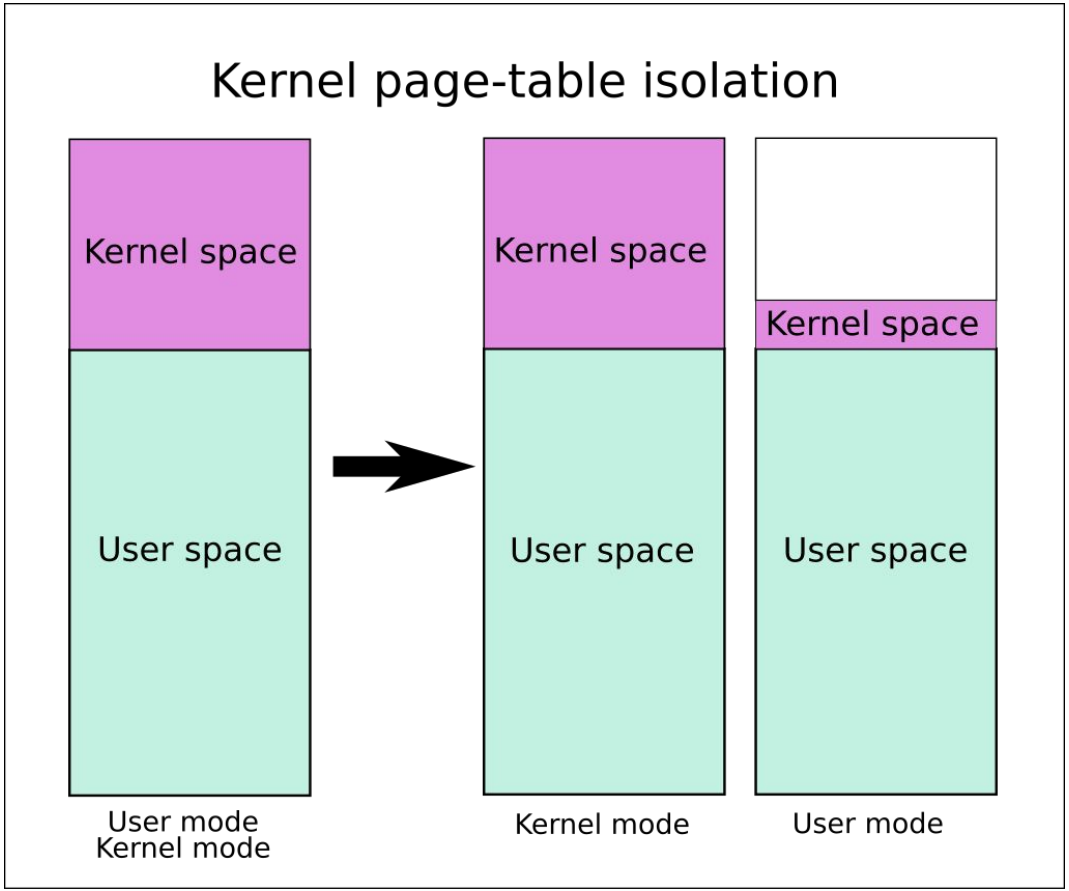
Vendors build in “Survivability features”

Microcode is the most common used tool for mitigations.

Other firmware is also used

“Chicken bits” to disable / change behavior

Some issues are best mitigated in software



Limitations of mitigations

Physical hardware cannot be changed in the field

Vendors build in “Survivability features”

Microcode is the most common used tool for mitigations.

Other firmware is also used

“Chicken bits” to disable / change behavior

Some issues are best mitigated in software

Mitigations are **not always possible/reasonable** and almost always **difficult** and **time-consuming** to engineer

Prevention Pre-silicon

Prevention starts before the product exist: pre-silicon

Pre-silicon is slow and cumbersome as the chips are emulated or simulated.

This makes security validation & research significantly different from software validation

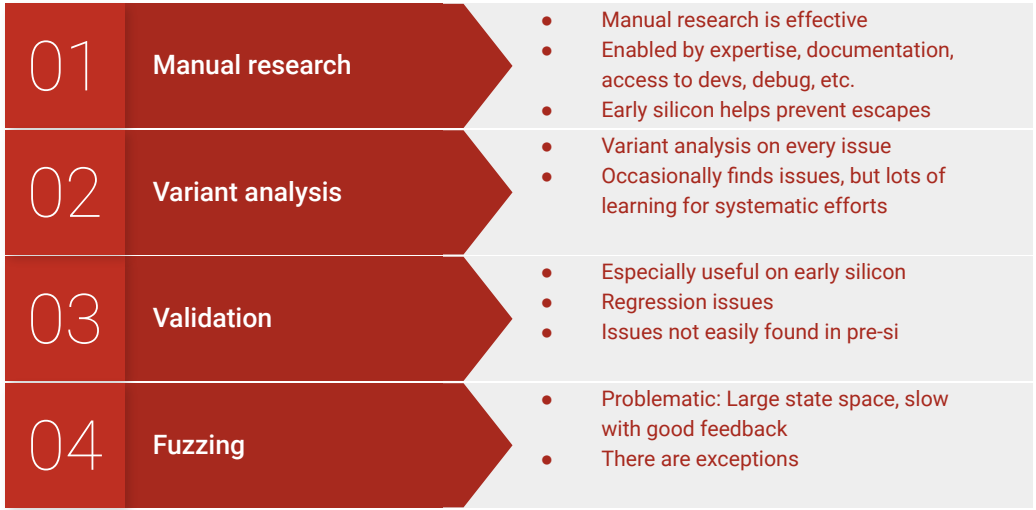
Post-silicon

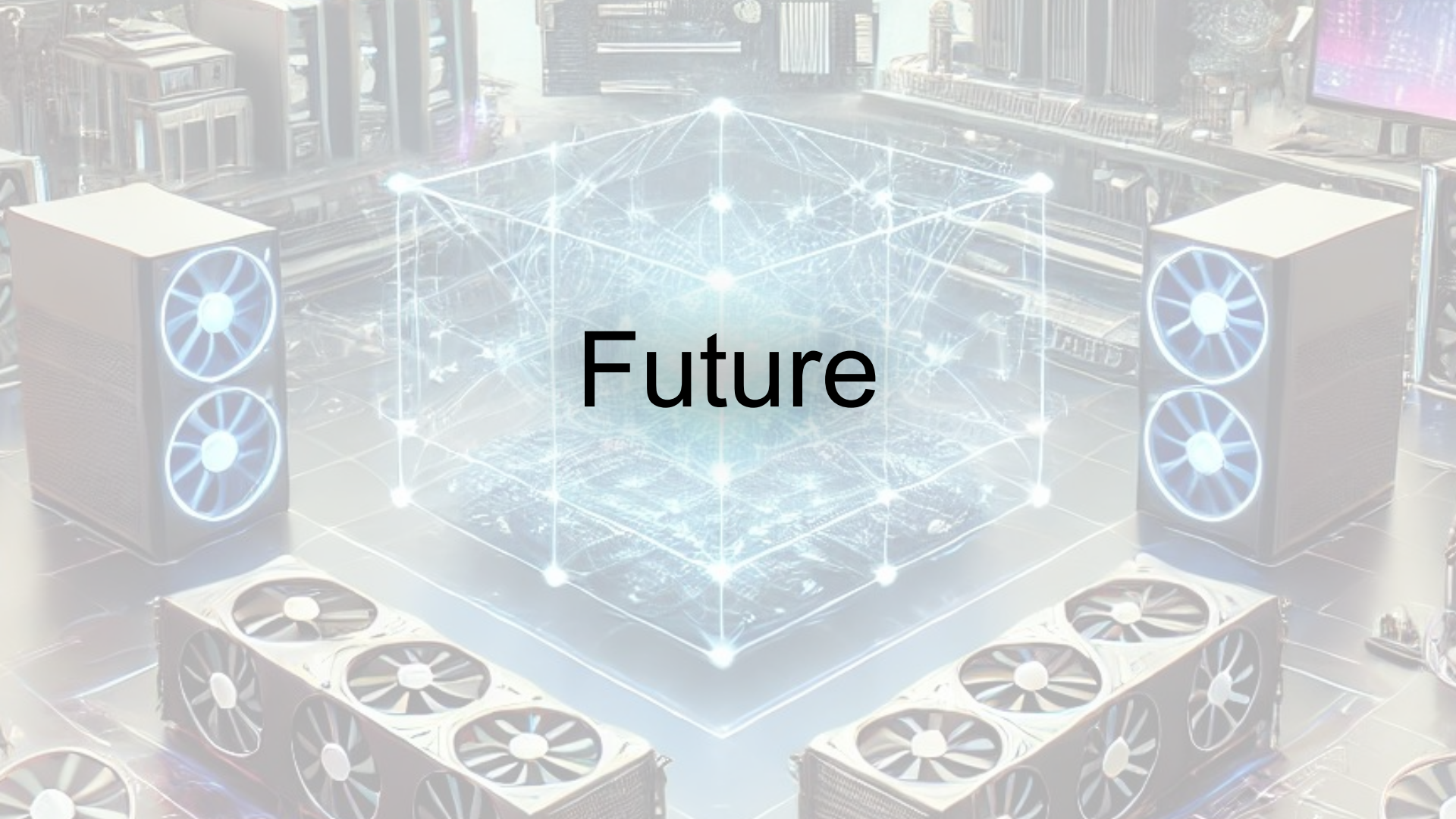
Prevention in silicon happens before product ship from A0 to shipping systems.

Some issues are best found in post-silicon.

Post-silicon issues are particularly difficult.

Learning from issues on last generation hardware is critically important.




Future

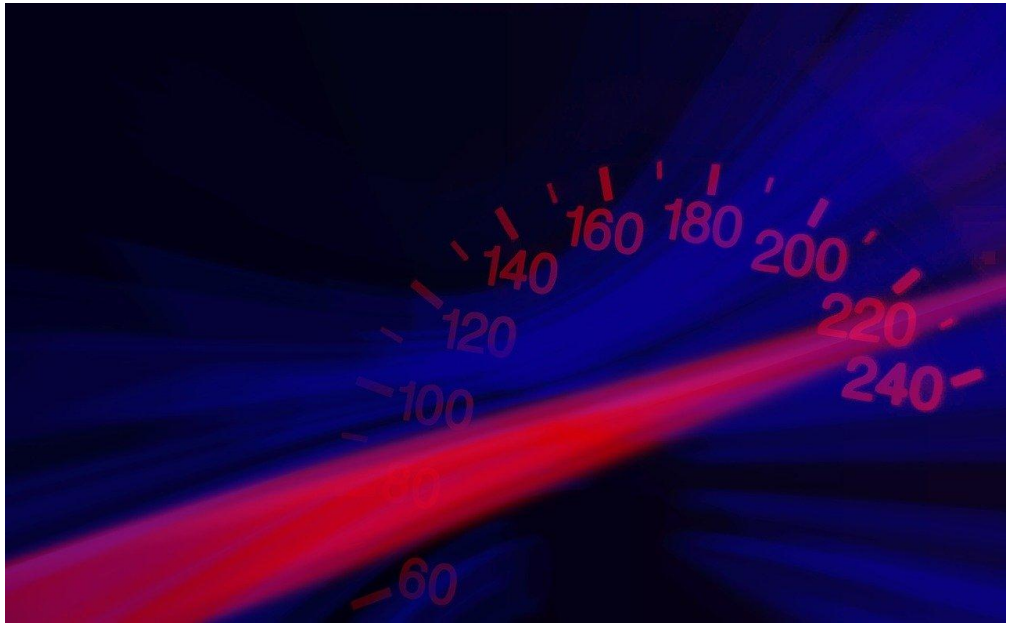
Future of uArch security is future of uArch

Silicon performance is the main underlying driver for growth in compute ecosystem

Performance comes from 3 sources

- New process technology
- uArch improvements
- Adaptation to changed workloads

uArch improvements & Changed workloads will lead to new security challenges



uArch security future

Offense

New kinds of prediction & data dependent behaviors (memory latency!). Memory is order of magnitude slower than compute. Some examples:

- New kinds of caches and bigger caches
- Work load specific prefetchers
- Different kinds of value prediction
- Cache & memory compression
- Growth in reorder buffer sizes
- New exploitation techniques

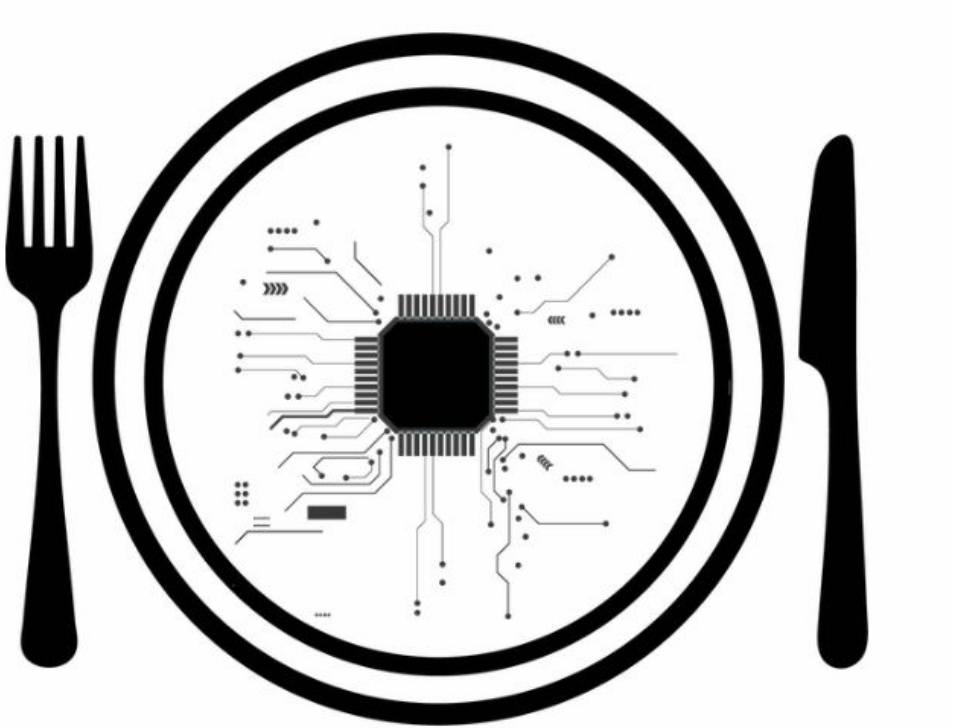
Defense

- Increased maturity
  - Better tooling
  - More defense in depth
- New microarchitecture security features
- More configurability of security
  - Ex.PSF switch on AMD
- Improved support for software influence
  - Ex. Local configuration switches

New kinds of compute

more heterogeneous - but all have uArch:

- GPU (new use cases)
  - Remote accessible
  - Increased complexity and new work loads
  - Example: “LeftoverLocals” by Trails of Bits
- Neural Processing Units
  - New model of compute
  - New threats: Integrity of models
  - Attack vector against system
- AI training accelerators in the cloud
  - Soon: shared resources + multi tenant
- More generally: More kinds of compute, more accelerators



Defensive side of things

Huge gap between academia and industry:

Academia

- provable Rowhammer mitigations available
- provable secure cache available

Industry

- probabilistic Rowhammer mitigations
- secure caches not adopted (but non-inclusive LLCs)

uArch in uArch

Embedded processors everywhere -- already with speculation:

Speculation vs confidentiality?

- Threat models rarely contain arbitrary execution
  → constrains attackers
- Embedded processors often provide low-level access → new and different kinds of assets

Take Aways

Side channels are **here to stay**

- Side channels **can be managed**

more aspects of microarchitecture and different kinds of issues

- Hard work for both offensive research and defense
- Defense is maturing

Microarchitecture is a **growth area**, so is microarchitecture security

Microarchitecture matters, so does microarchitecture security

Microarchitecture Vulnerabilities

Past, Present and Future

Daniel Gruss (Graz University of Technology)
Anders Fogh (Intel Corporation)