The Security Arms Race in Confidential Computing

Jo Van Bulck

DistriNet, KU Leuven, Belgium jo.vanbulck@cs.kuleuven.be vanbulck.net

Seminar in Cybersecurity, KU Leuven – March 10, 2026

DistriNet

KU LEUVEN

The Big Picture: Protecting Private Data

Data in transit

Data in use

Data at rest

The Big Picture: Protecting Private Data

Data in transit

- SSL/TLS etc.

Data in use

Data at rest

- Full disk encryption

The Big Picture: Protecting Private Data

Data in transit

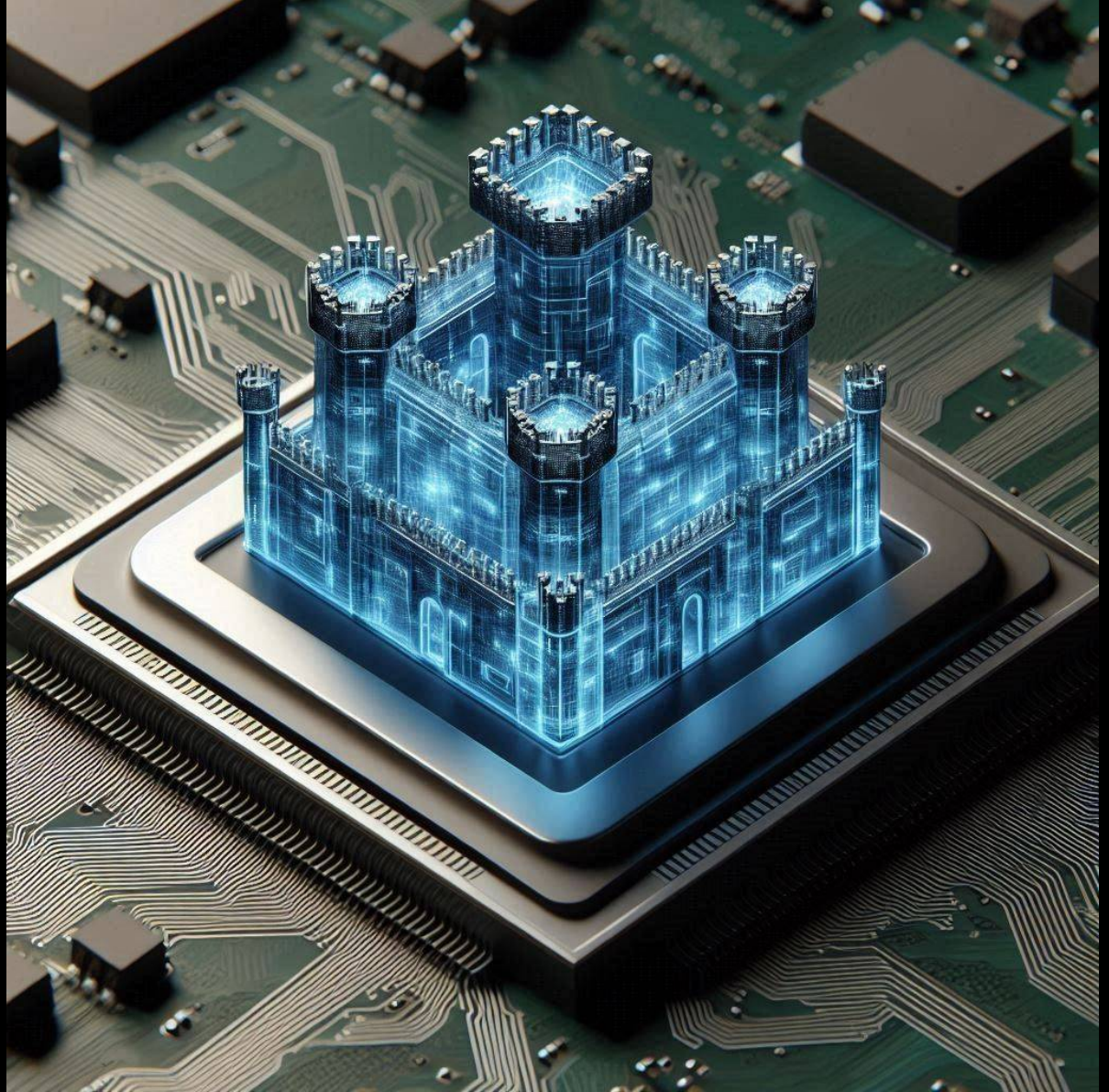
- ✓ SSL/TLS etc.

Data in use

- ? **Homomorphic encryption?**
- ? **Trusted Execution?**
  = *Confidential Computing*
  = *Hardware Enclaves*

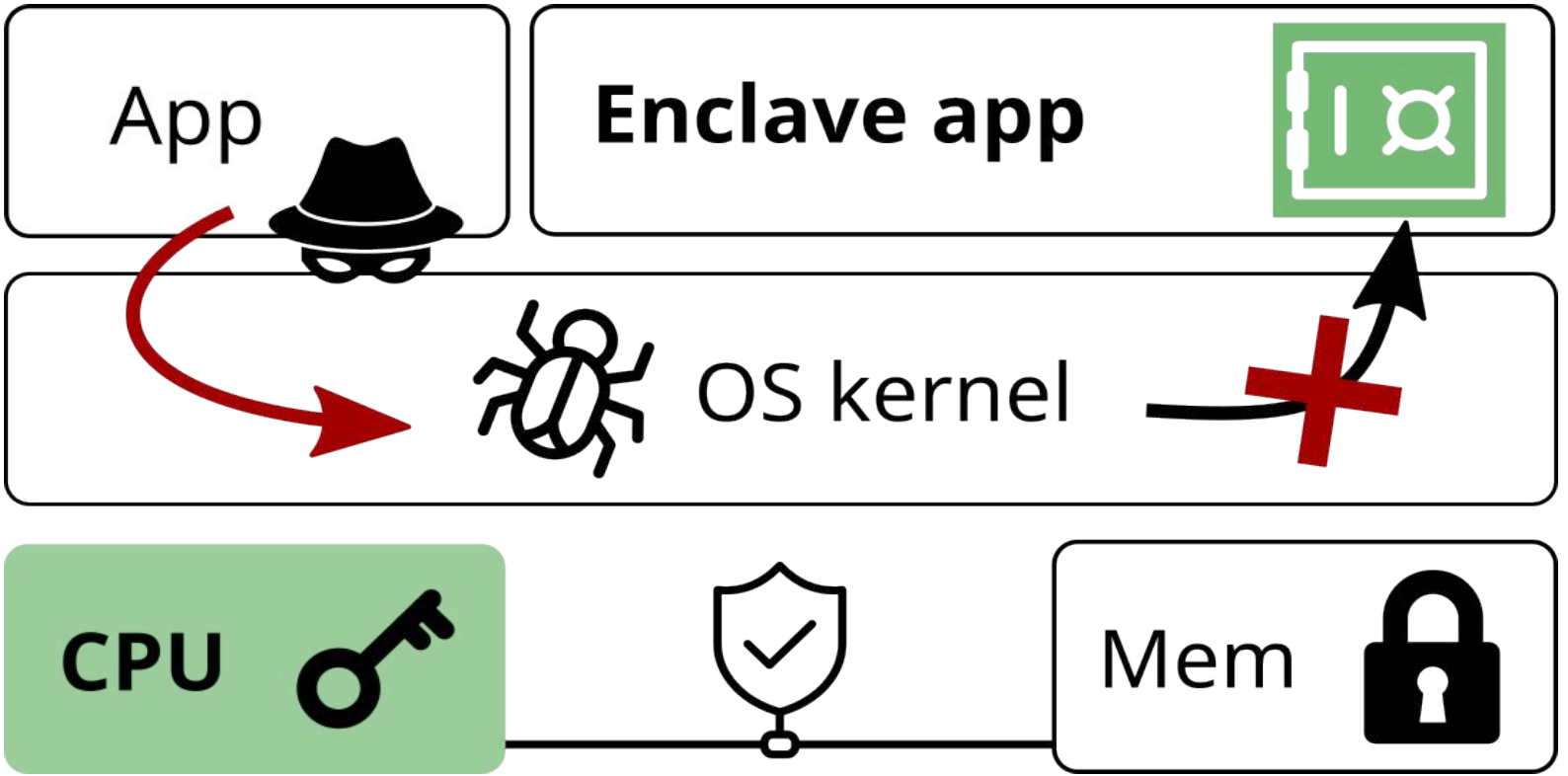
Data at rest

- ✓ Full disk encryption



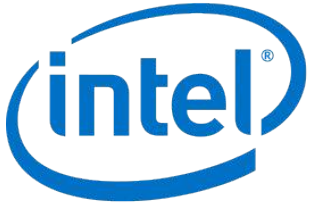
DALL·E 3

Confidential Computing: Reducing Attack Surface



Trusted execution: Hardware-level isolation and attestation

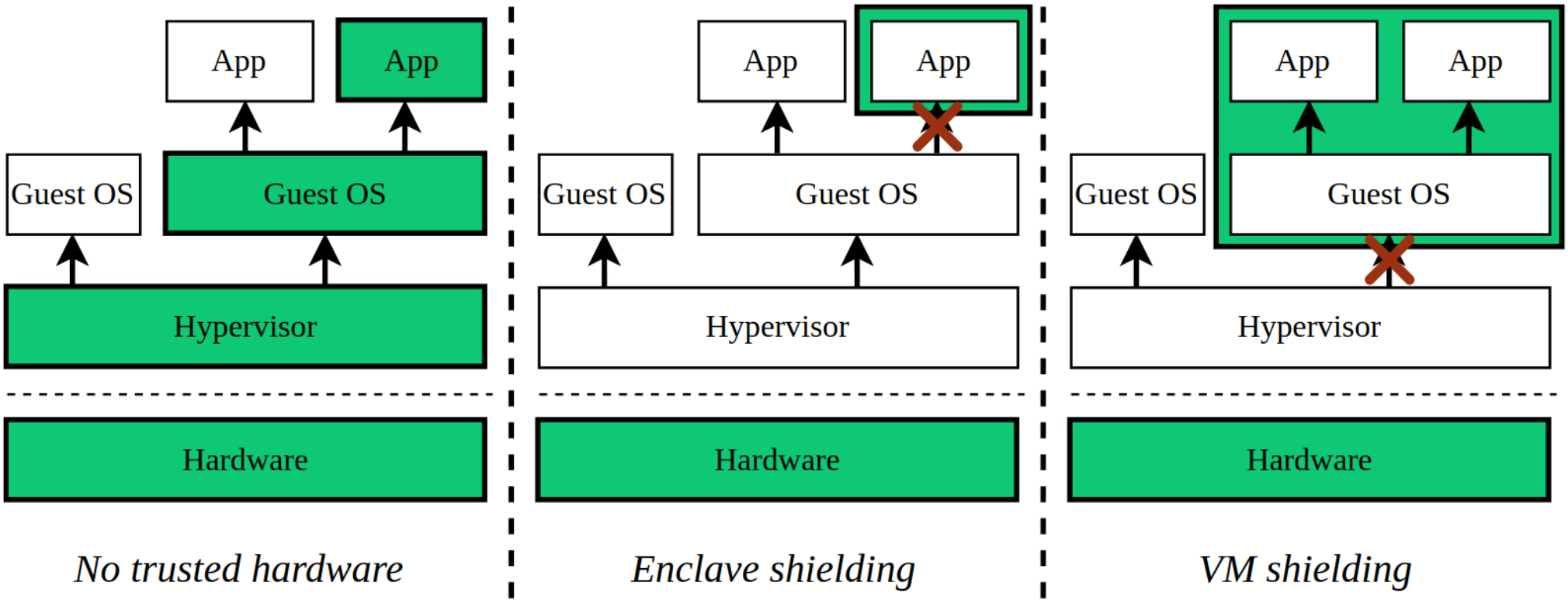
The Rise of Trusted Execution Environments (TEEs)



- 2004: ARM TrustZone
- 2015: **Intel Software Guard Extensions (SGX)**
- 2016: AMD Secure Encrypted Virtualization (SEV)
- 2018: IBM Protected Execution Facility (PEF)
- 2020: AMD SEV with Secure Nested Paging (SEV-SNP)
- 2022: Intel Trust Domain Extensions (TDX)
- 2023: ARM Confidential Compute Architecture (CCA)
- 2024: NVIDIA Confidential Computing

TEEs are here to stay...

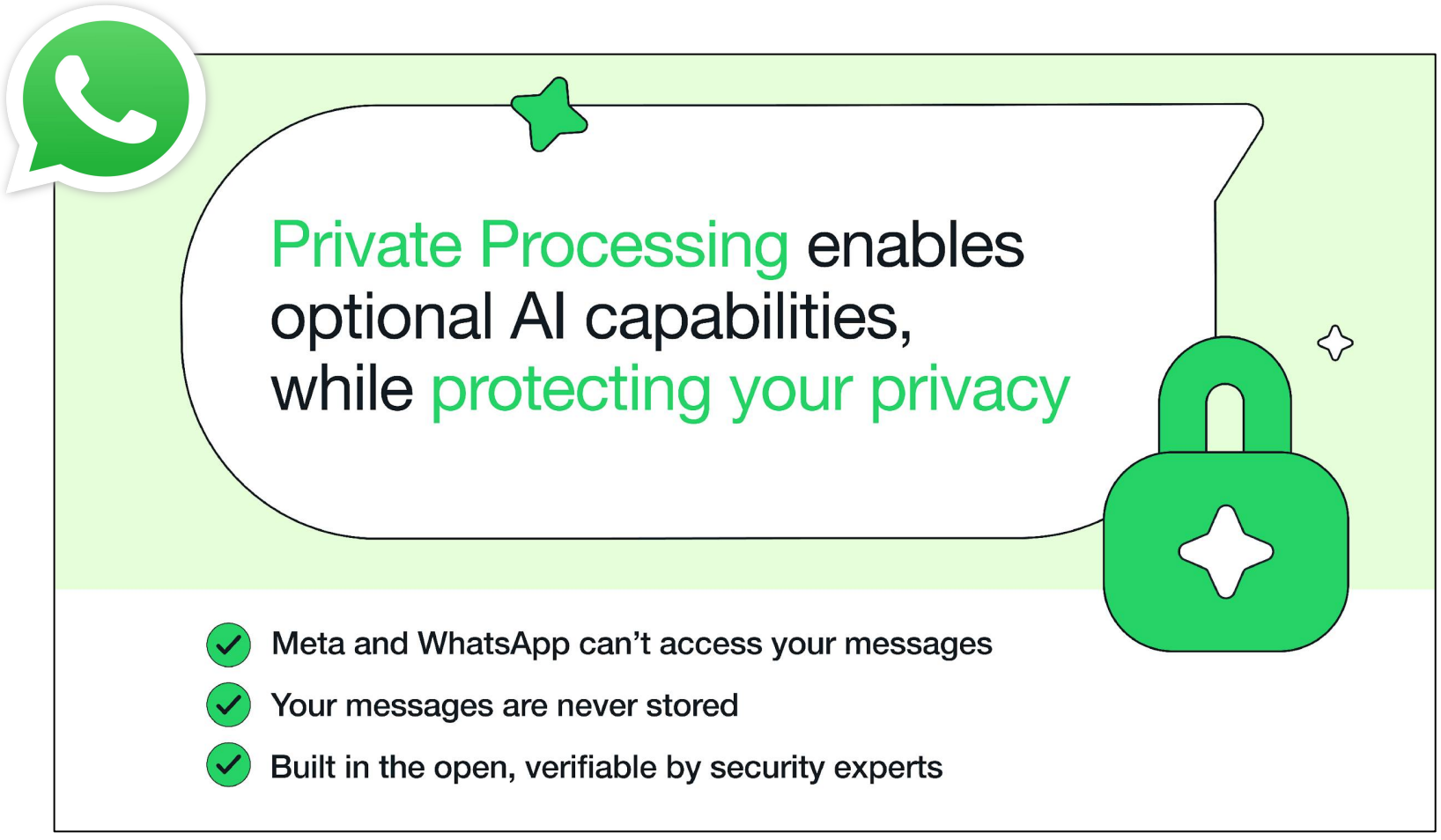
Confidential Computing Isolation Paradigms



Intel SGX, Arm TrustZone-M

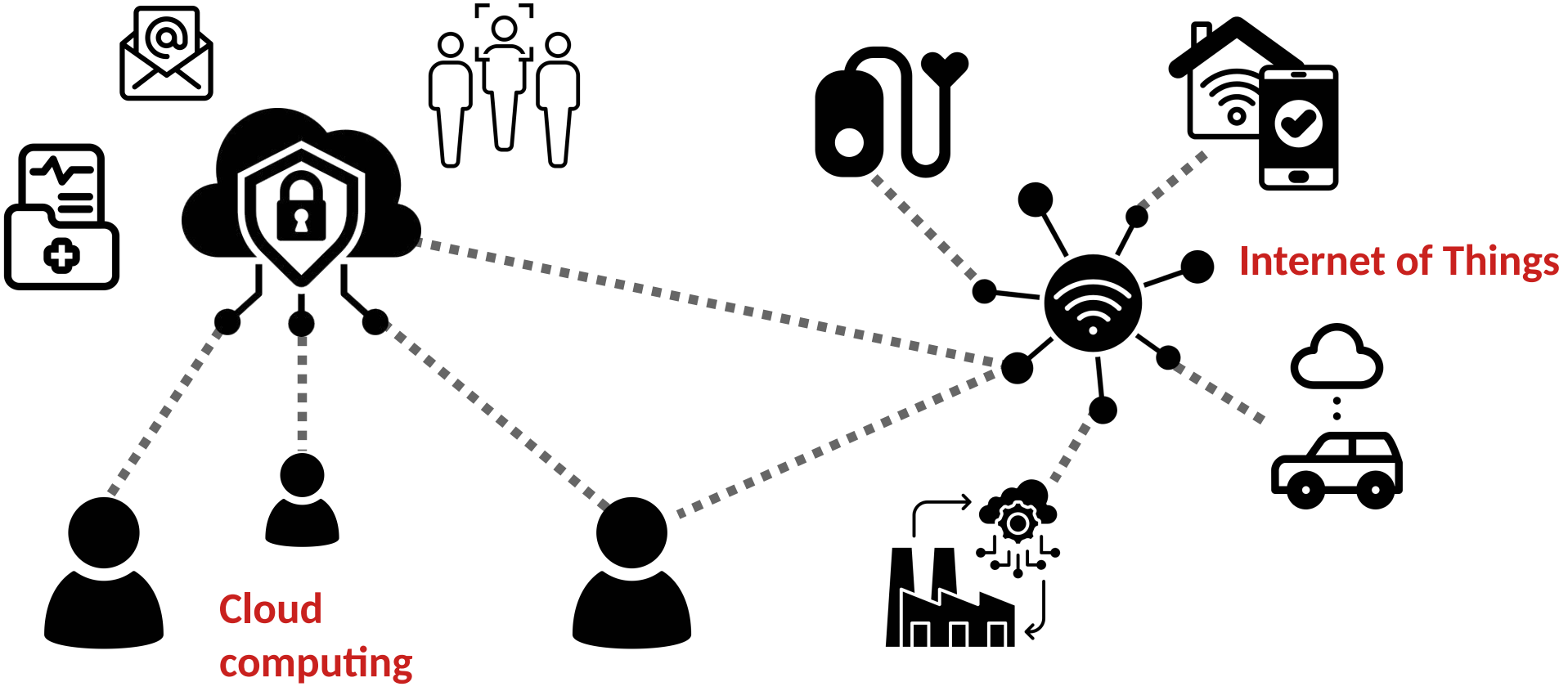
AMD SEV, Intel TDX, Arm CCA
Arm TrustZone-A

Example: WhatsApp Private Processing Confidential AI



https://engineering.fb.com/2025/04/29/security/whatsapp-private-processing-ai-tools/

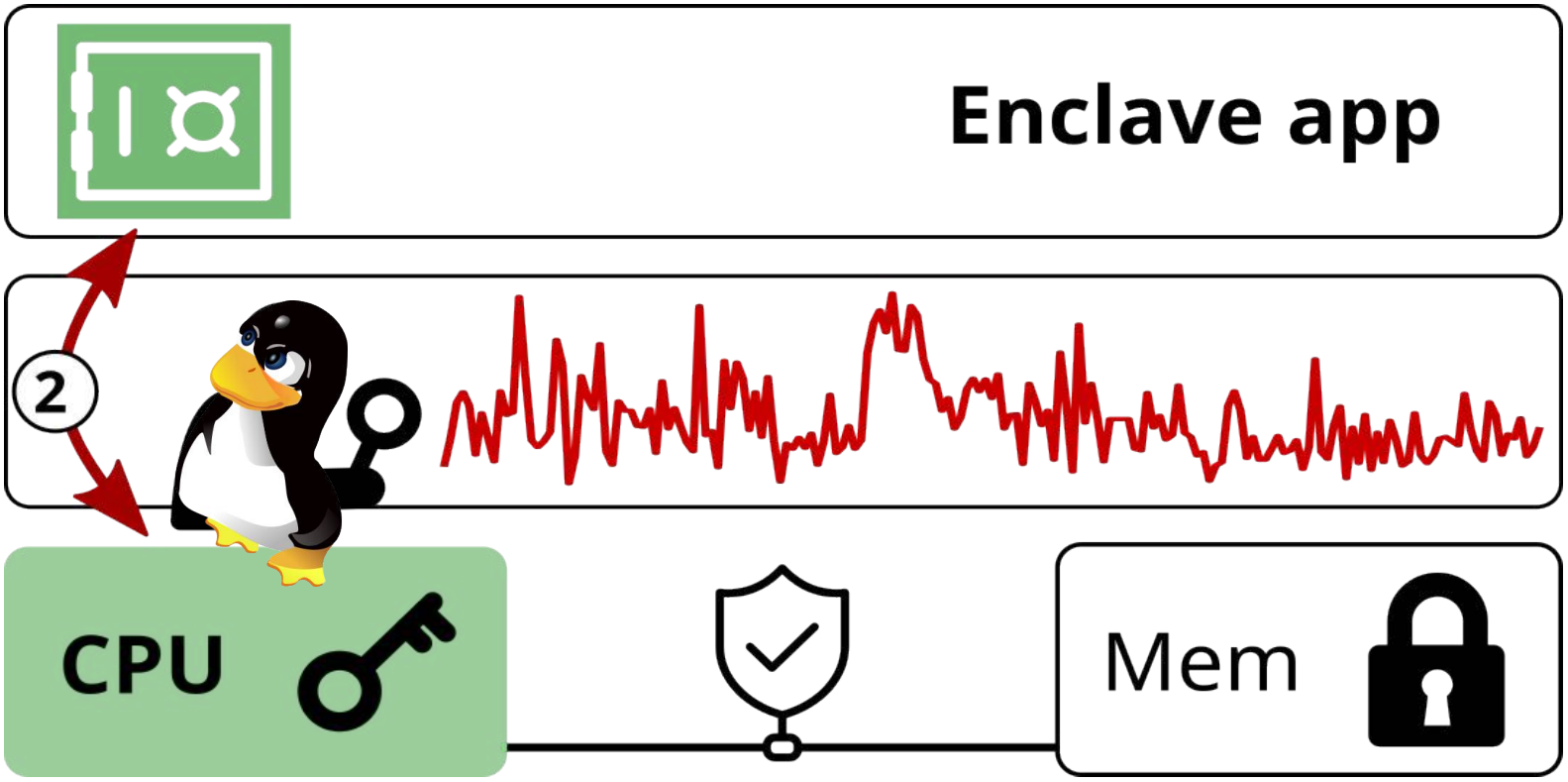
“Confidential Computing Today, Just Computing Tomorrow” *



* *Mark Russinovich, CTO Microsoft Azure*

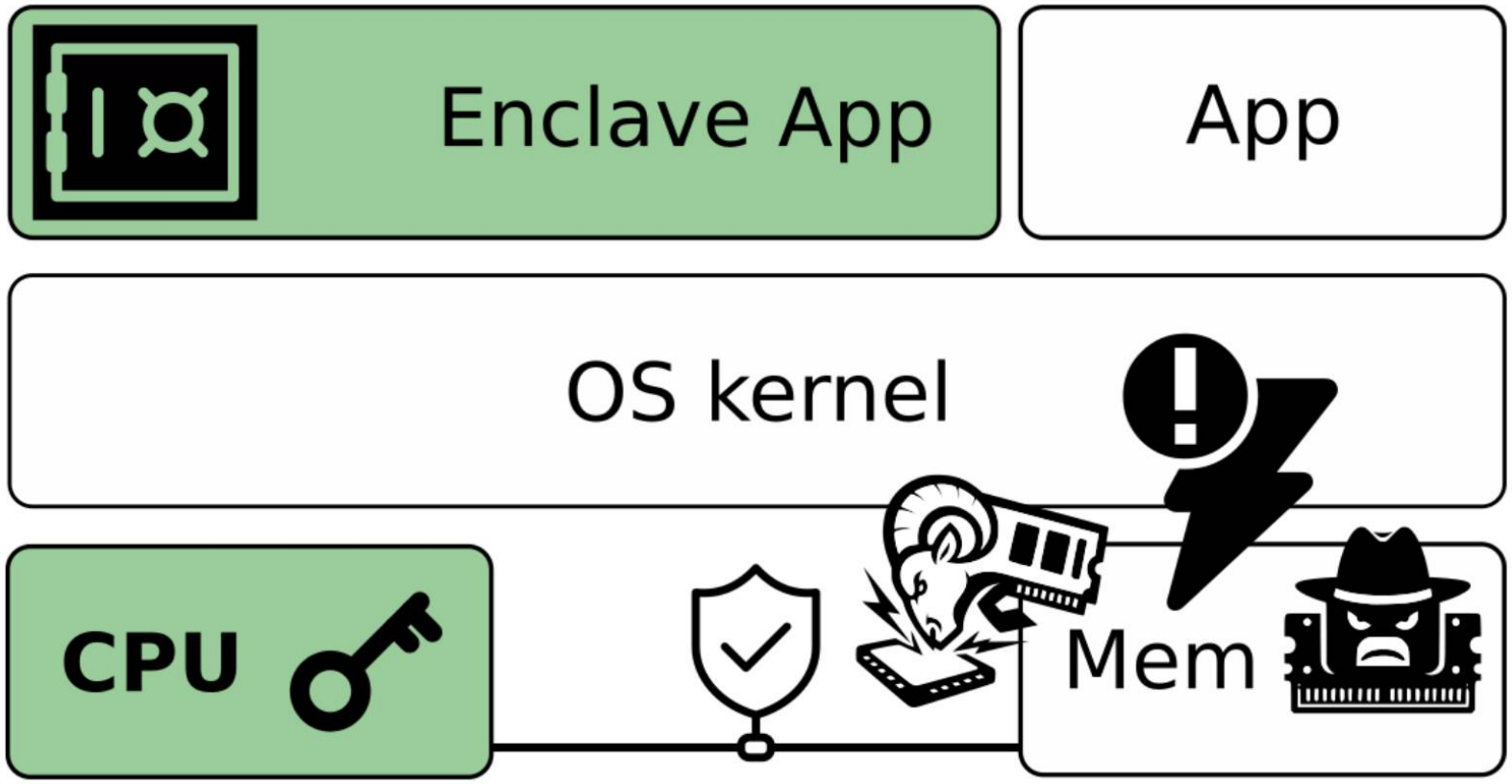
Confidential Computing: The Weakest Link?

Lecture Overview: Privileged Software Interface *(Part 1)*



Untrusted OS → New class of powerful software-based side channels!

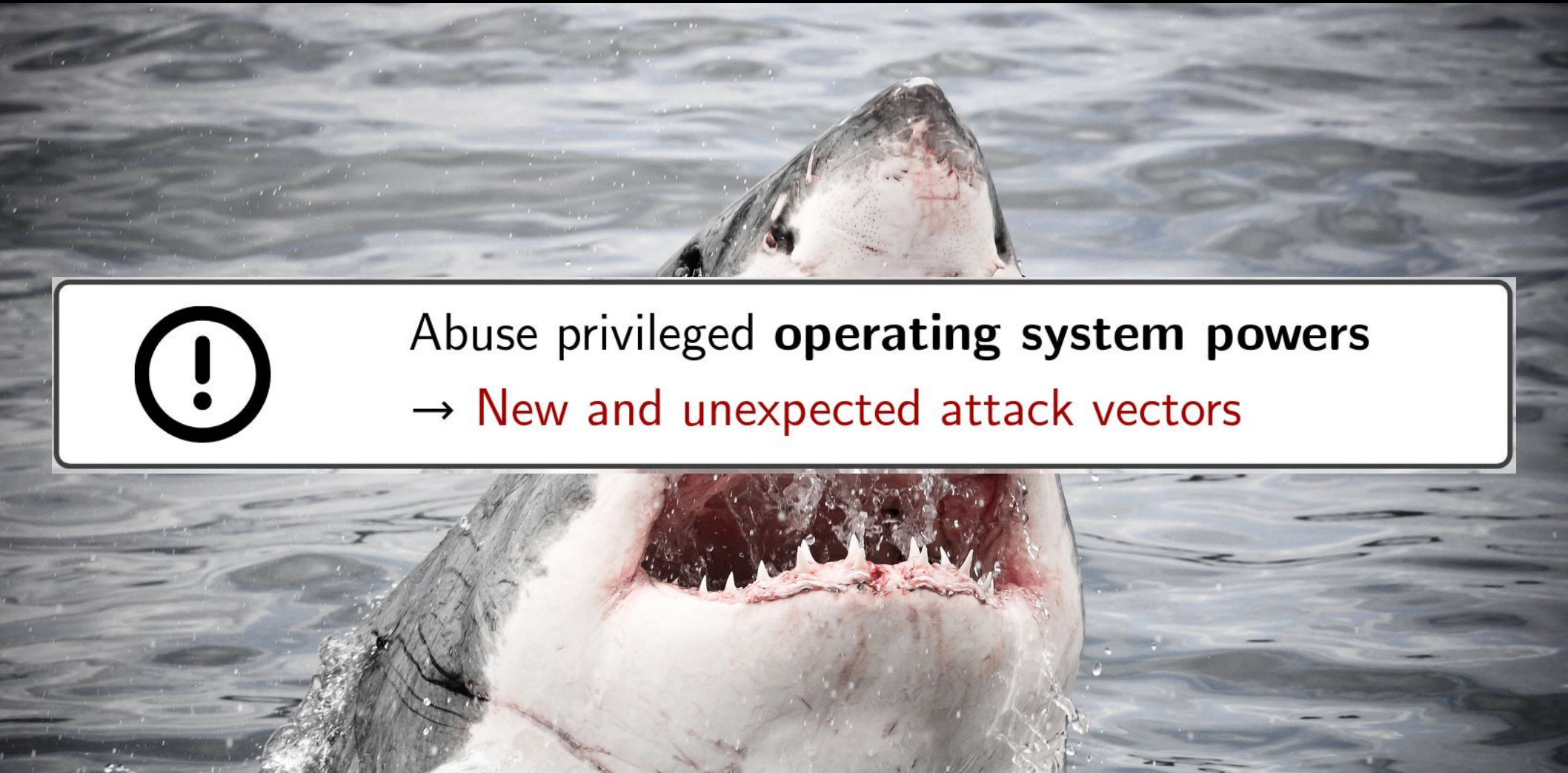
Lecture Overview: Privileged Hardware Interfaces *(Part 2)*



CPU == trust boundary → Hardware-level interposer attacks!

Game Changer: Privileged “Bottom-Up” Adversary Model

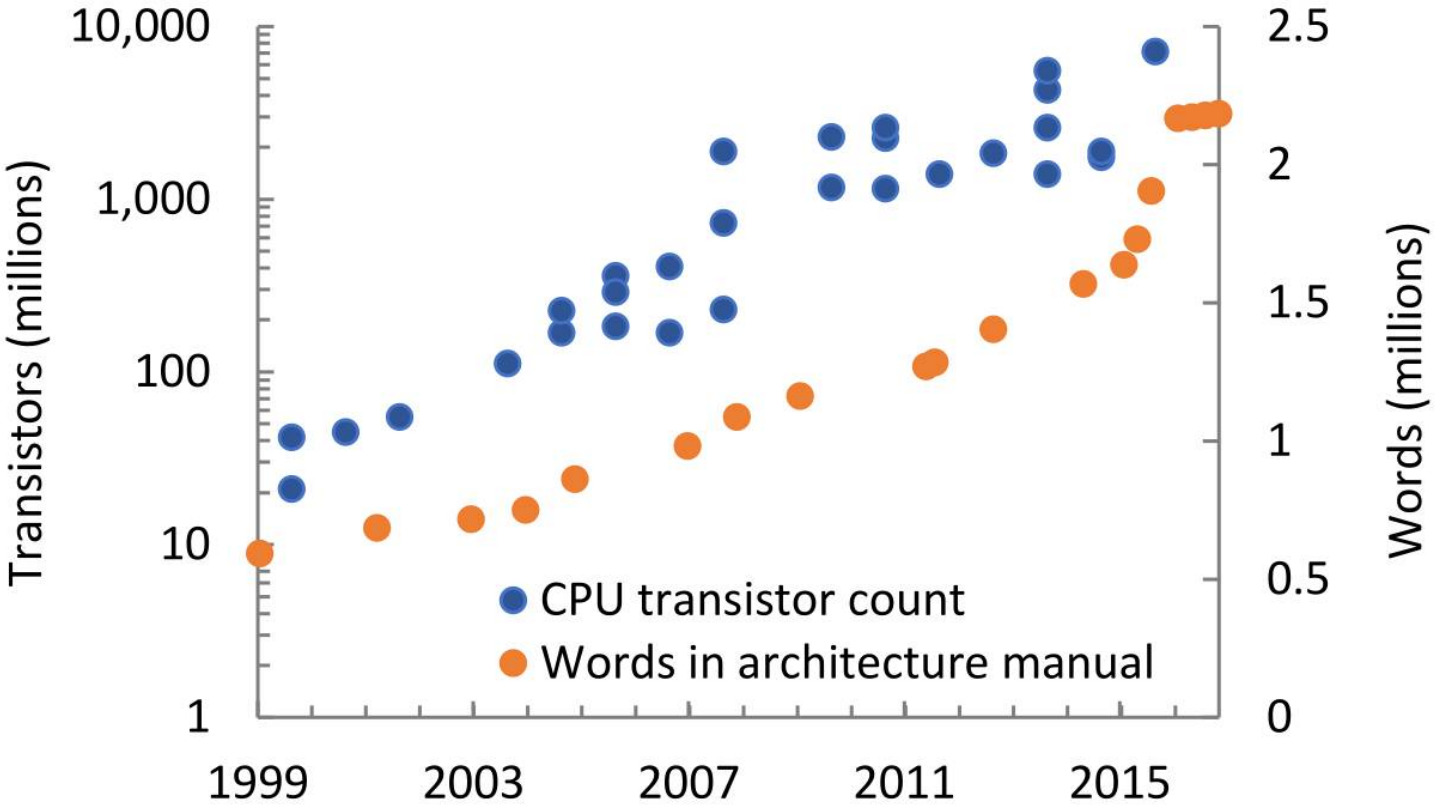
Game Changer: Privileged “Bottom-Up” Adversary Model



Abuse privileged **operating system powers**

→ New and unexpected attack vectors

Privileged x86 Interface and Complexity Growth…



Baumann, "Hardware is the New Software", HotOS'17.

Systematizing the SGX Attack Landscape

	Attack \ Properties	x86 Interface						Constraints			
		PTE	GDT	IRQ	MSR	CR0	PMC	SMT	REP	SA	Granular
μ-arch contention	Cache priming [181, 92, 29, 225, 79]	◐	○	◐	◐	○	◐	◐	●	○	64 B
	Branch prediction [156, 59, 105]	◐	○	◐	◐	◐	◐	◐	●	○	Inst
	DRAM row buffer conflicts [263]	○	○	○	◐	◐	○	○	●	○	1-8 KiB
	False dependencies [180]	○	○	○	◐	○	○	●	●	○	4 B
	Interrupt latency [**256**, 95, 208]	◐	○	●	◐	○	○	○	●	○	Inst
	Port contention [7]	○	○	○	◐	○	○	●	●	○	μ-op
Control channel	Page faults [277]	●	○	○	○	○	○	○	○	○	4 KiB
	Page table A/D [**258**, 263]	●	○	◐	○	◐	○	◐	○	○	4 KiB
	Page table flushing [**258**]	●	○	◐	○	○	○	◐	○	○	32 KiB
	Interrupt counting [**182**]	●	○	●	○	○	○	○	○	○	Inst
	IA32 segmentation faults [**91**]	◐	●	◐	○	○	○	○	○	○	1 B-4 KiB
	Alignment faults [**254**]	○	○	●	○	●	○	○	○	○	1 B
Transient	Foreshadow L1D extraction [**249**]	●	○	◐	○	○	○	◐	○	○	–
	Data sampling [**223**, 216, 211]	●	○	◐	○	○	○	◐	◐	○	–
	Spectre [39, 148]	◐	○	◐	○	○	○	◐	◐	◐	–
	Load value injection [**251**]	●	○	◐	○	○	○	◐	◐	◐	–
Interface	Memory safety [**254**, 154, 24, 266]	◐	○	◐	○	○	○	○	◐	◐	–
	Undervolting [**188**, 137, 210]	◐	○	◐	●	○	○	◐	●	◐	–
	Off-chip memory address bus [152]	●	○	◐	○	◐	○	○	○	○	64 B

Possibly every privileged CPU feature can be abused!

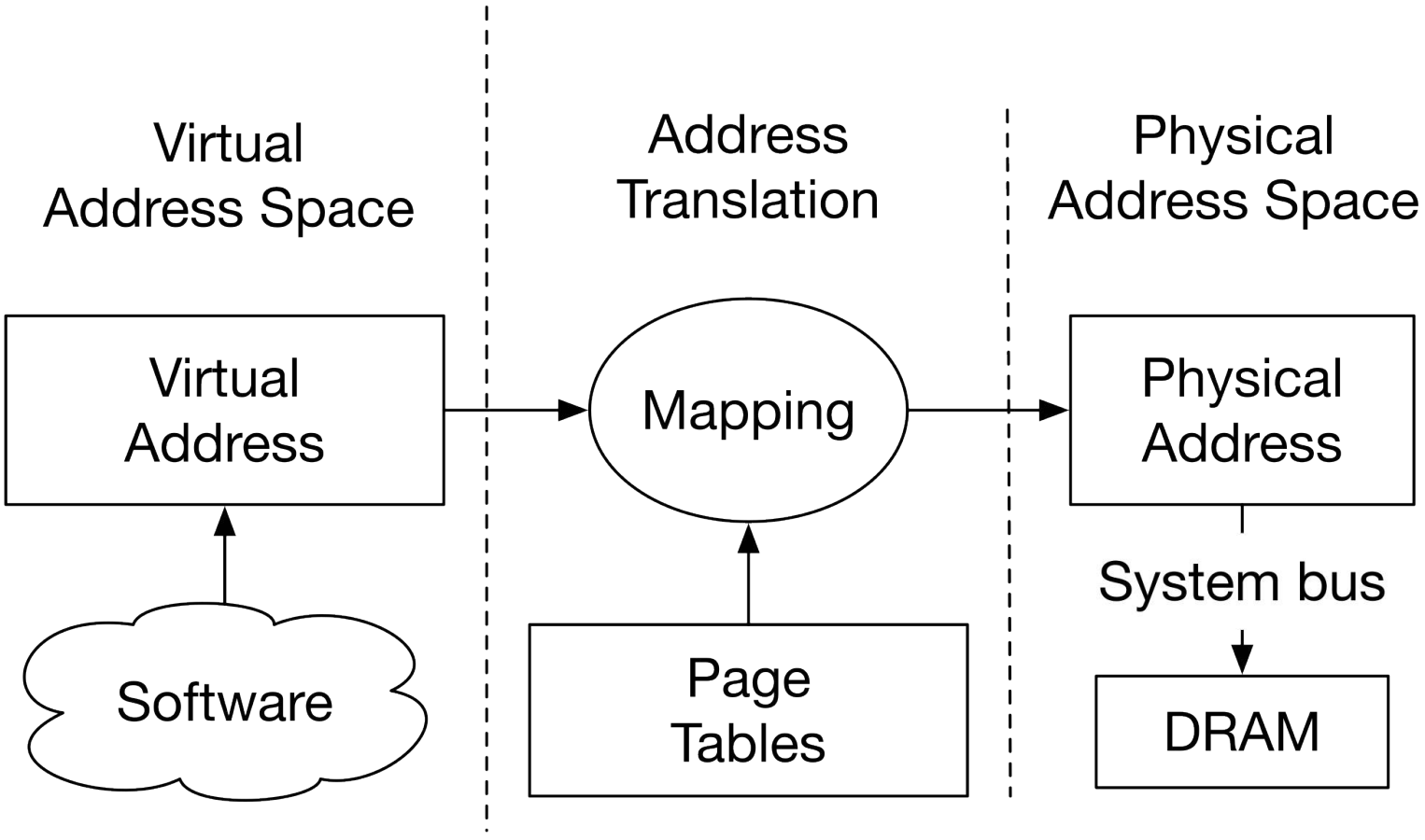
TEE Attack Research Leads the Way . . .

TEE Attack Research Leads the Way . . .

- Privileged TEE attacker models sets the bar!
- Idealized execution environment for attack research
- **Generalizations:** e.g., Foreshadow-NG, branch prediction, address translation, etc.

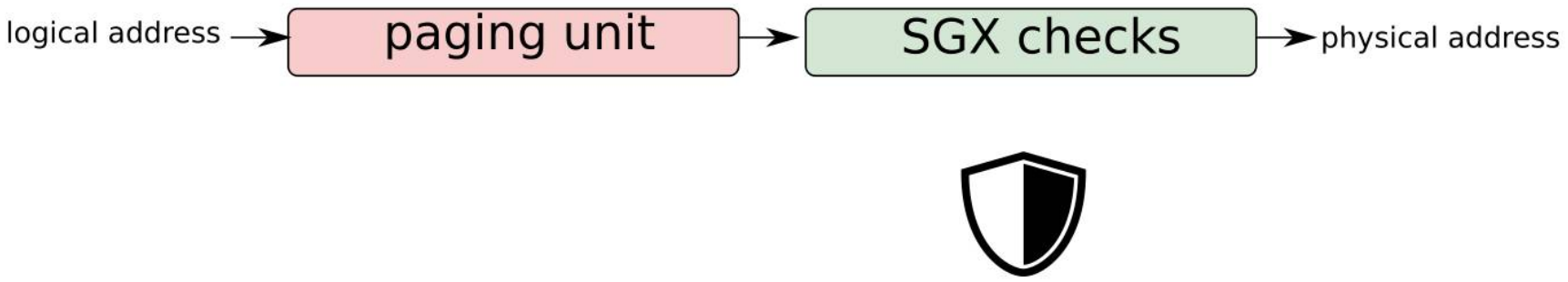
Idea #1 - Software-Based Attacks: Spatial Resolution?

Background: The Virtual Memory Abstraction



Costan et al. “Intel SGX explained”, IACR 2016.

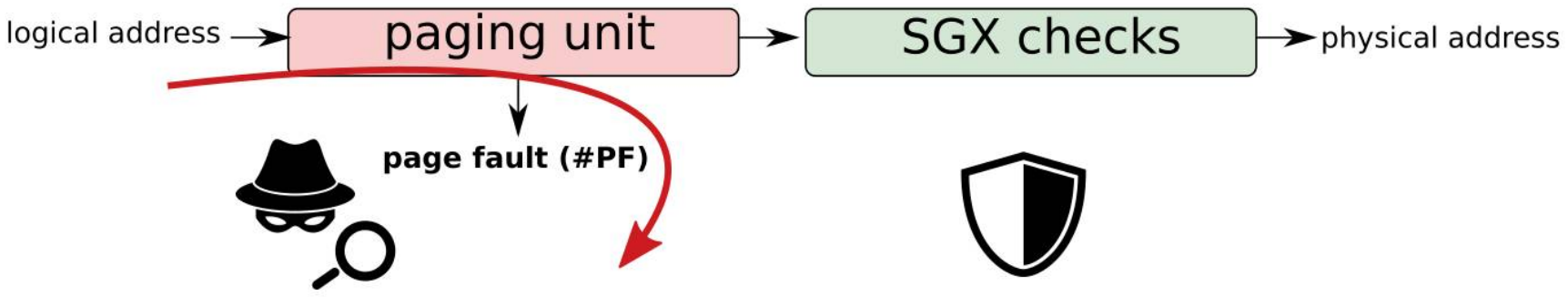
Idea: Page Faults as a Side Channel



SGX machinery protects against direct address remapping attacks

Xu et al. “Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems”, IEEE S&P 2015.

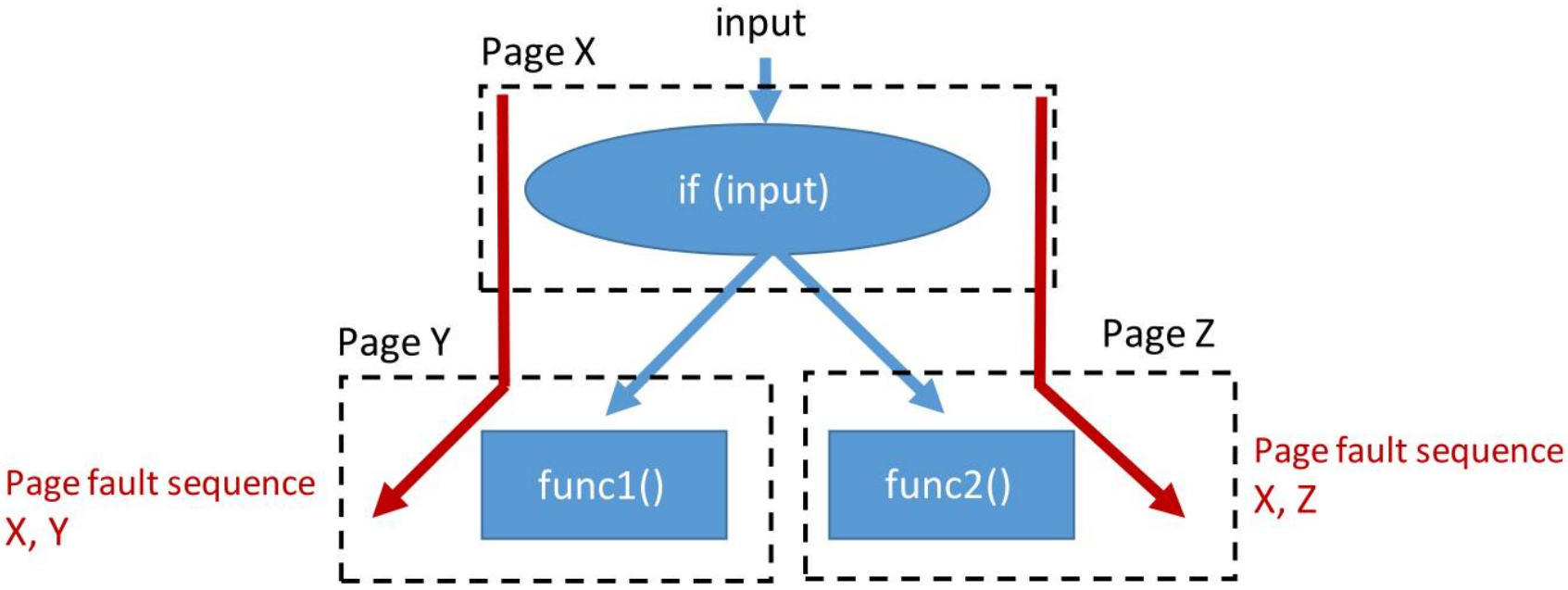
Idea: Page Faults as a Side Channel



. . . but untrusted address translation may **fault(!)**

Xu et al. “Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems”, IEEE S&P 2015.

Intel SGX: Page Faults as a Side Channel

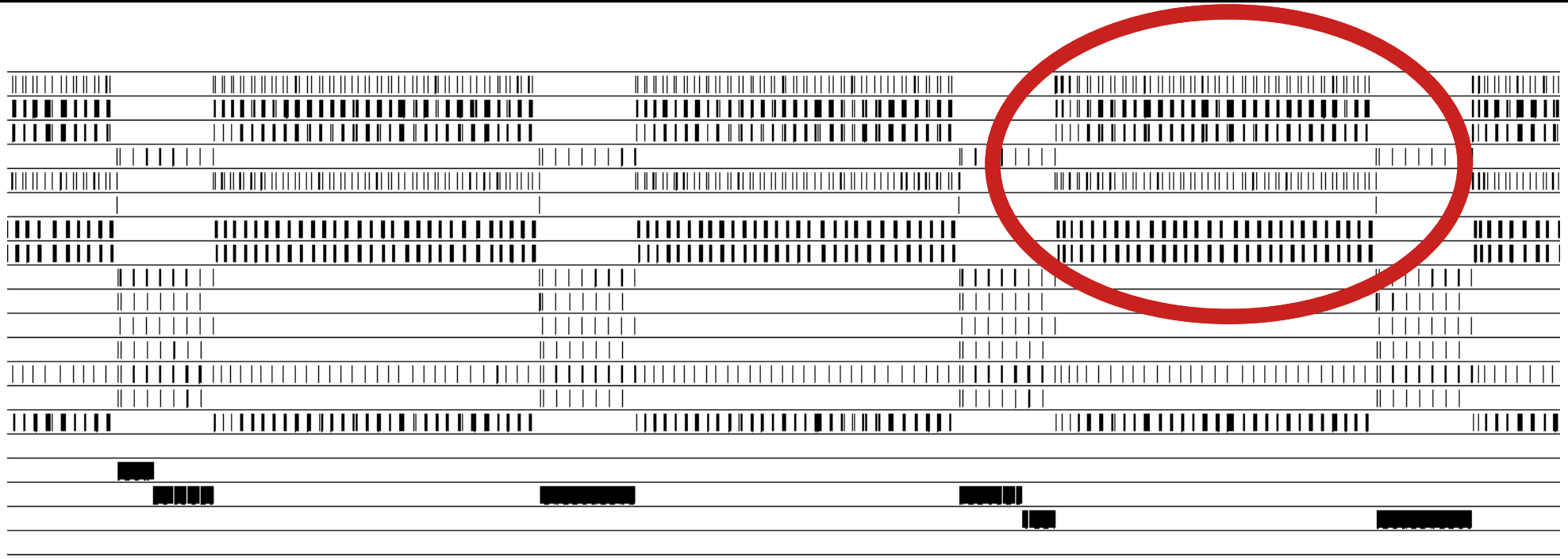


Xu et al.: "Controlled-channel attacks: Deterministic side channels for untrusted operating systems", Oakland 2015.

⇒ Page fault traces leak **private control data/flow**

Xu et al. "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems", IEEE S&P 2015.

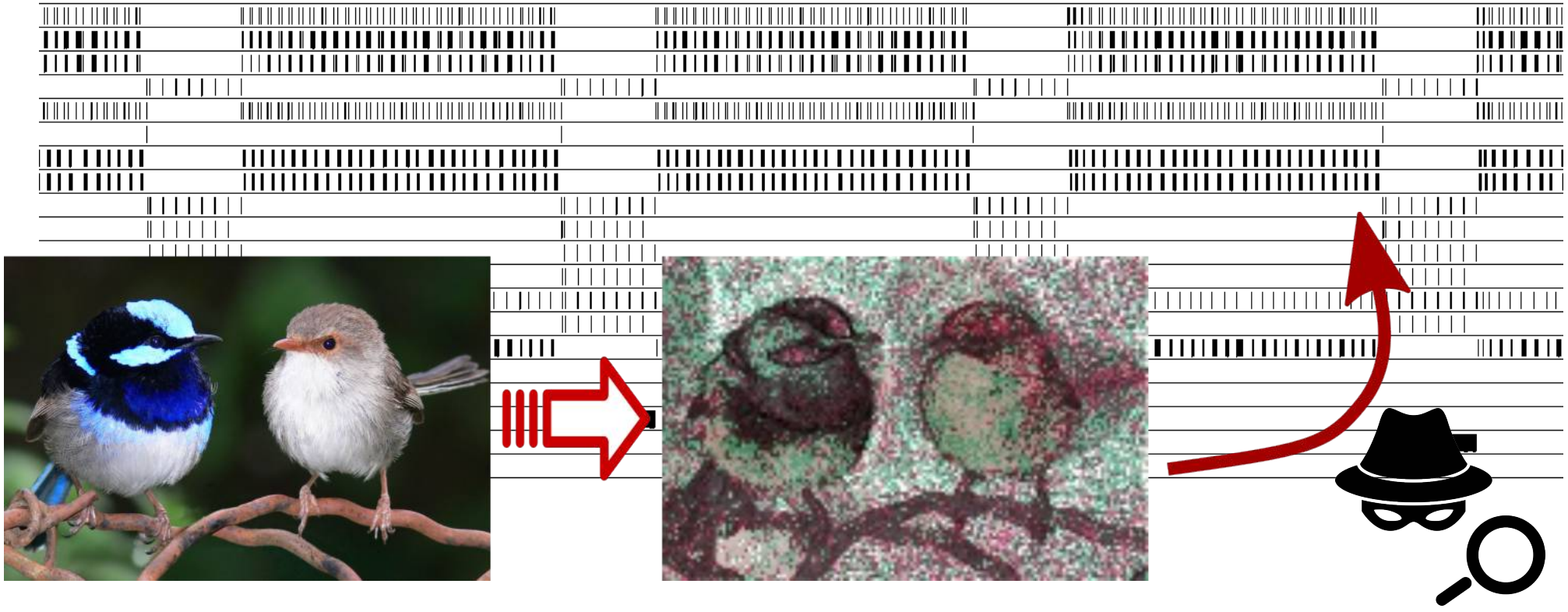
Spatial Resolution: Page-Granular Memory Access Traces



Detailed trace of (coarse-grained) code and data accesses over time…

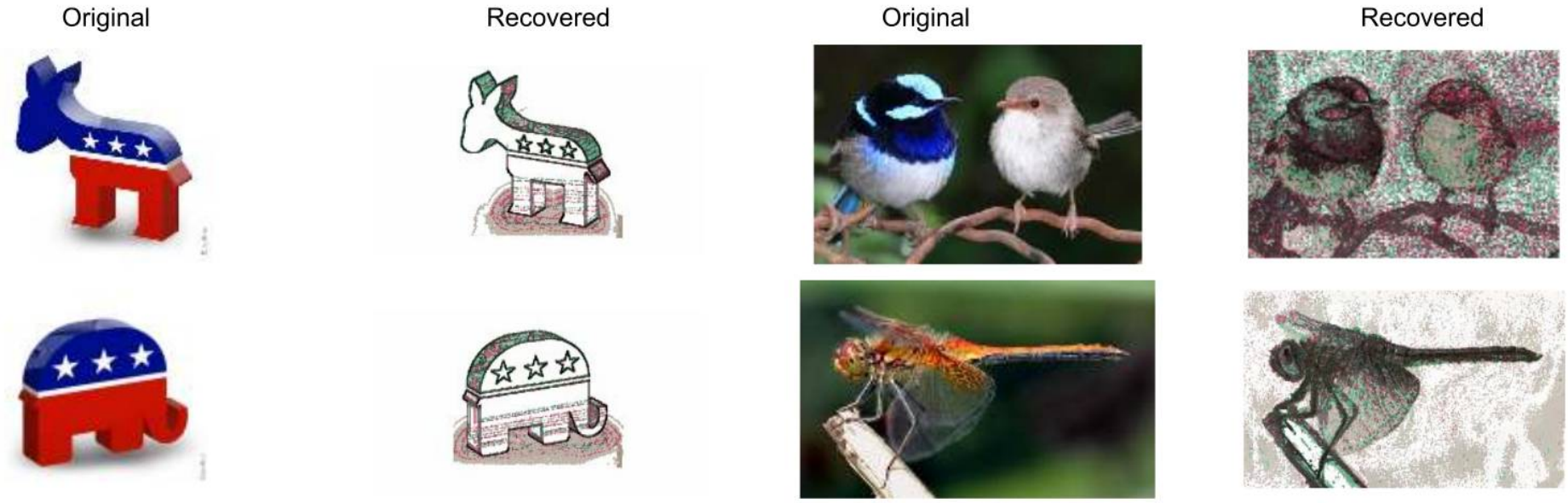
Vanoverloop et. al “TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms”, USENIX’25.

Spatial Resolution: Page-Granular Memory Access Traces



Vanoverloop et. al “TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms”, USENIX’25.

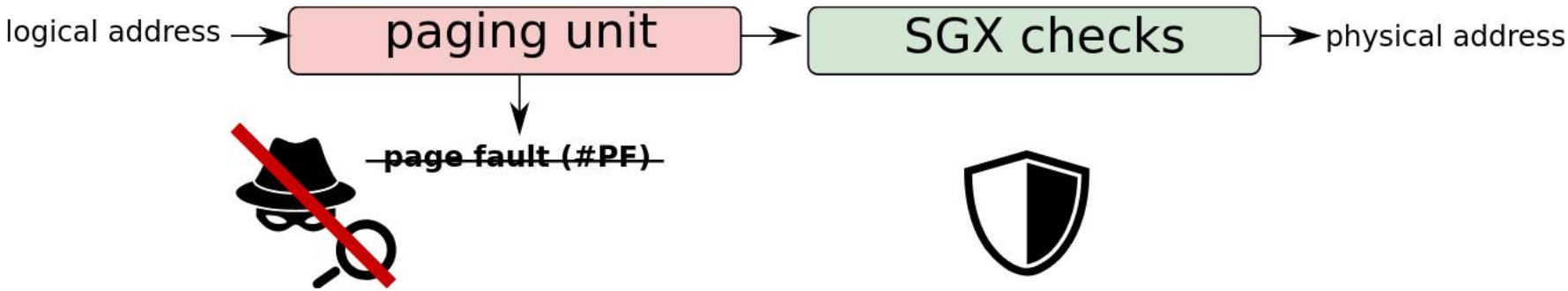
Spatial Resolution: Page-Granular Memory Access Traces



Xu et al.: "Controlled-channel attacks: Deterministic side channels for untrusted operating systems", Oakland 2015.

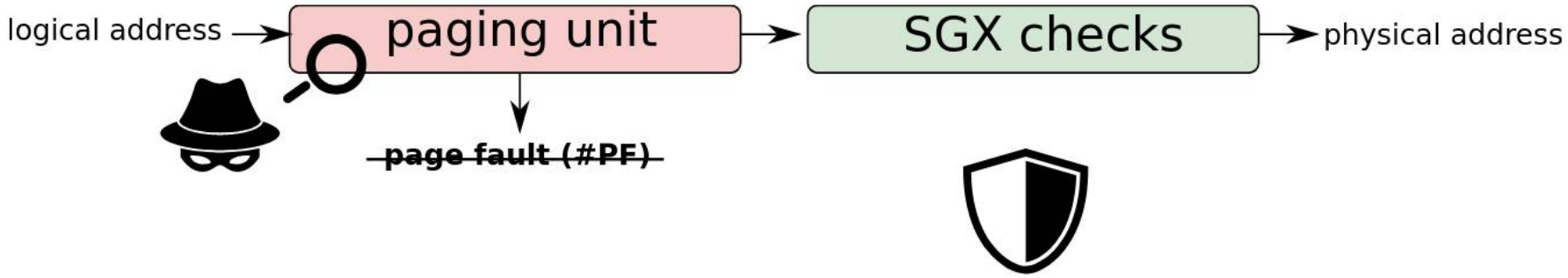
. . . but **many faults** and a coarse-grained **4 KiB granularity**

Naive Solutions: Hiding Enclave Page Faults



- Shih et al. "T-SGX: Eradicating controlled-channel attacks against enclave programs", NDSS 2017.
- Shinde et al. "Preventing page faults from telling your secrets", AsiaCCS 2016.

Naive Solutions: Hiding Enclave Page Faults



. . . But stealthy attacker can learn page visits without triggering faults!

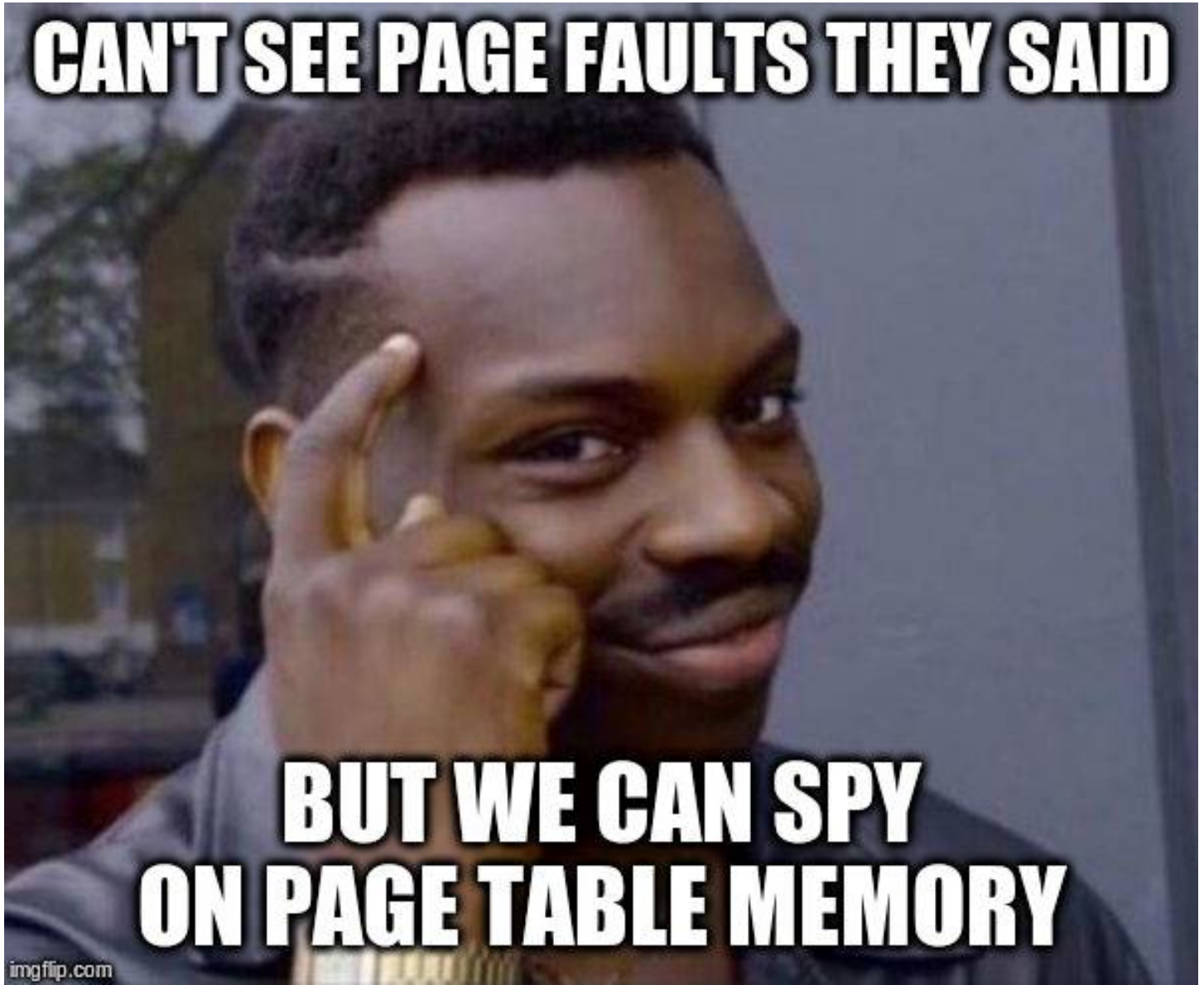
Documented Side-Effects of Address Translation

4.8 ACCESSED AND DIRTY FLAGS

For any paging-structure entry that is used during linear-address translation, bit 5 is the **accessed** flag.[2] For paging-structure entries that map a page (as opposed to referencing another paging structure), bit 6 is the **dirty** flag. These flags are provided for use by memory-management software to manage the transfer of pages and paging structures into and out of physical memory.

Whenever the processor uses a paging-structure entry as part of linear-address translation, it sets the accessed flag in that entry (if it is not already set).

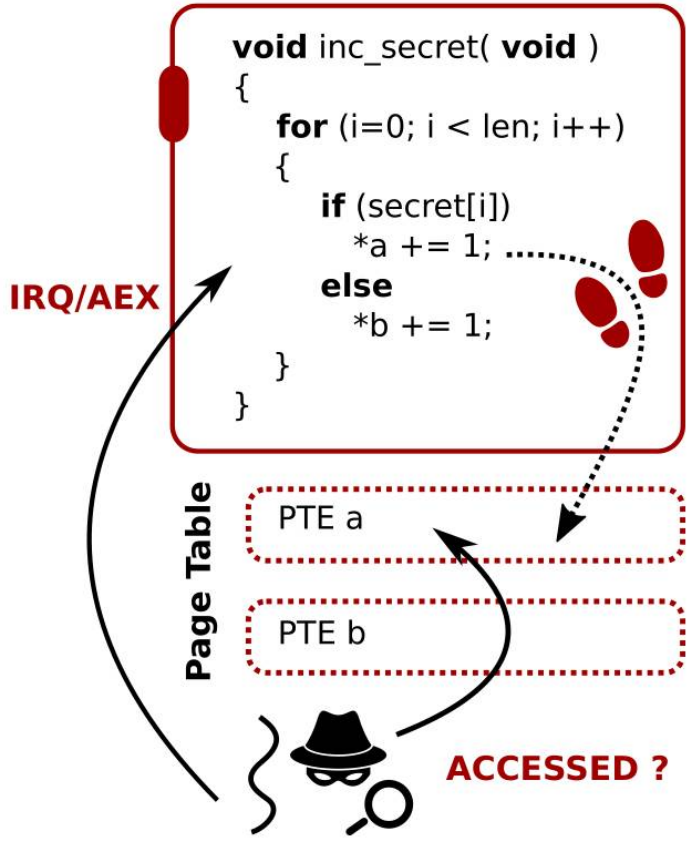
Whenever there is a write to a linear address, the processor sets the dirty flag (if it is not already set) in the paging-structure entry that identifies the final physical address for the linear address (either a PTE or a paging-structure entry in which the PS flag is 1).


CAN'T SEE PAGE FAULTS THEY SAID
BUT WE CAN SPY
ON PAGE TABLE MEMORY
imgflip.com

Telling your Secrets without Page Faults

1. Attack vector: PTE **status flags**:
   - A(ccessed) bit
   - D(irty) bit

   ⇝ Also updated in enclave mode!



Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

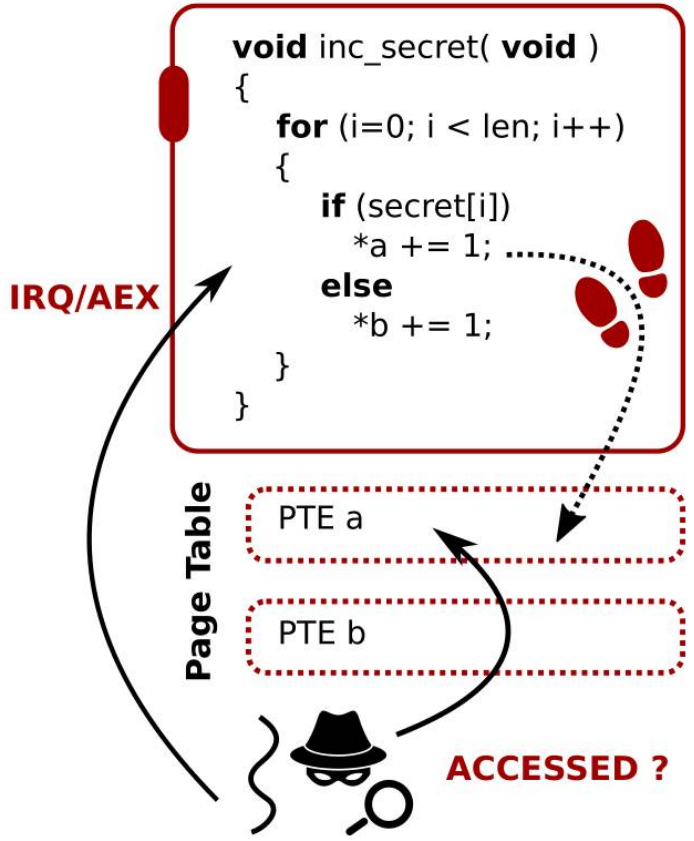
Telling your Secrets without Page Faults

1. Attack vector: PTE **status flags**:
   - A(ccessed) bit
   - D(irty) bit

   ⇝ Also updated in enclave mode!

2. Attack vector: Unprotected **page table memory**:
   - Cached as regular data
   - Accessed during address translation

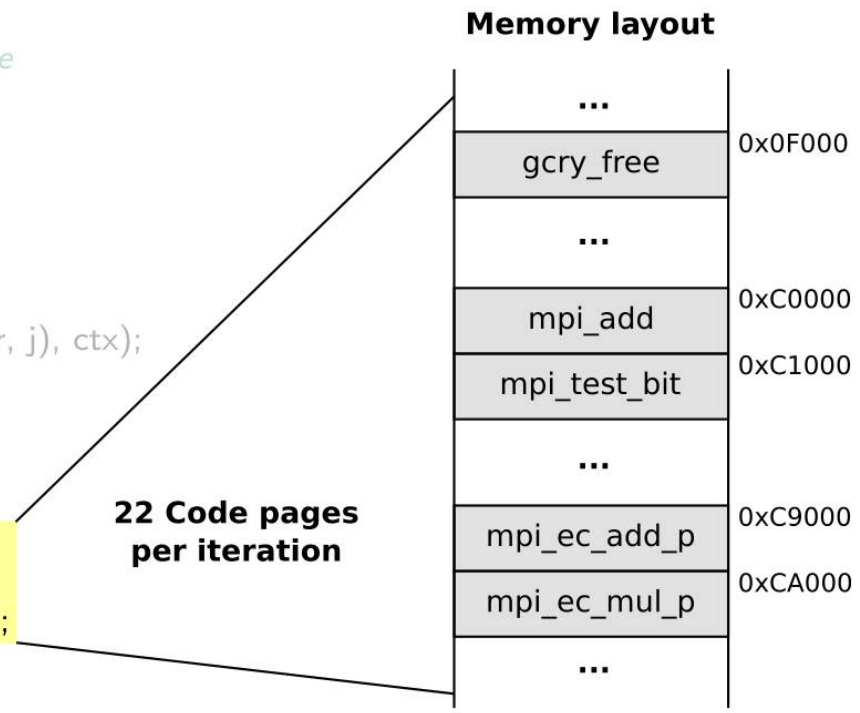
   ⇝ Flush+Reload cache timing attack!



Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

Attacking Libgcrypt EdDSA (Simplified)

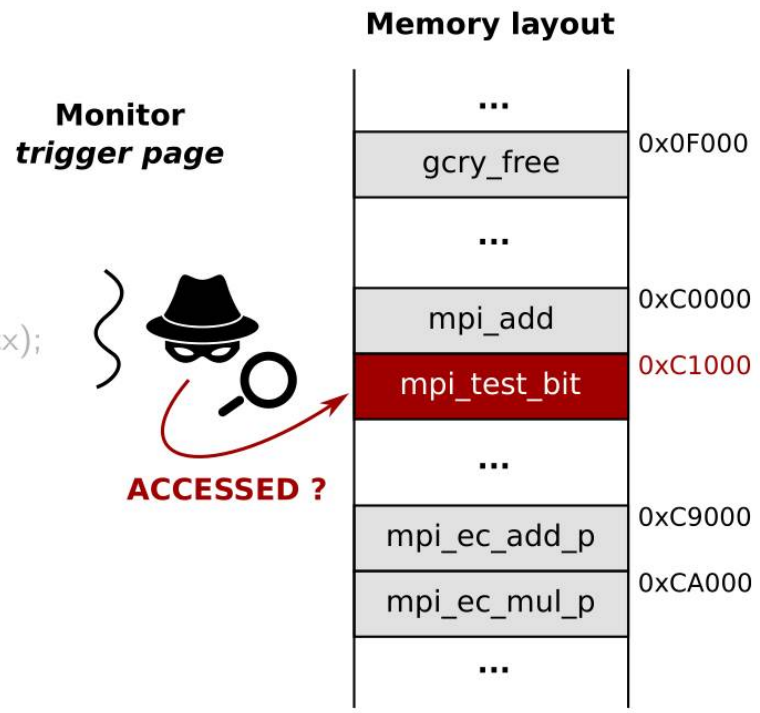
```
1  if (mpi_is_secure (scalar)) {
2      /* If SCALAR is in secure memory we assume that it is the
3         secret key we use constant time operation. */
4      point_init (&tmppnt);
5
6      for (j=nbits-1; j >= 0; j--) {
7          _gcry_mpi_ec_dup_point (result, result, ctx);
8          _gcry_mpi_ec_add_points (&tmppnt, result, point, ctx);
9          point_swap_cond (result, &tmppnt, mpi_test_bit (scalar, j), ctx);
10     }
11     point_free (&tmppnt);
12 } else {
13     for (j=nbits-1; j >= 0; j--) {
14         _gcry_mpi_ec_dup_point (result, result, ctx);
15         if (mpi_test_bit (scalar, j))
16             _gcry_mpi_ec_add_points (result, result, point, ctx);
17     }
18 }
```



Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

Attacking Libgcrypt EdDSA (Simplified)

```
1  if (mpi_is_secure (scalar)) {
2      /* If SCALAR is in secure memory we assume that it is the
3         secret key we use constant time operation. */
4      point_init (&tmppnt);
5
6      for (j=nbits-1; j >= 0; j--) {
7          _gcry_mpi_ec_dup_point (result, result, ctx);
8          _gcry_mpi_ec_add_points (&tmppnt, result, point, ctx);
9          point_swap_cond (result, &tmppnt, mpi_test_bit (scalar, j), ctx);
10     }
11     point_free (&tmppnt);
12 } else {
13     for (j=nbits-1; j >= 0; j--) {
14         _gcry_mpi_ec_dup_point (result, result, ctx);
15         if (mpi_test_bit (scalar, j))
16             _gcry_mpi_ec_add_points (result, result, point, ctx);
17     }
18 }
```



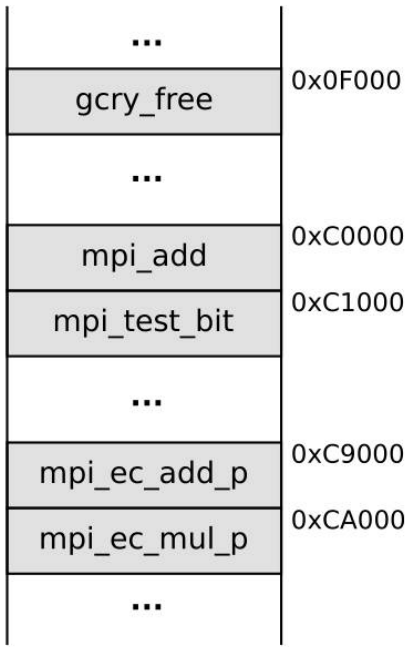
Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

Attacking Libgcrypt EdDSA (Simplified)

```
1  if (mpi_is_secure (scalar)) {
2      /* If SCALAR is in secure memory we assume that it is the
3         secret key we use constant time operation. */
4      point_init (&tmppnt);
5
6      for (j=nbits-1; j >= 0; j--) {
7          _gcry_mpi_ec_dup_point (result, result, ctx);
8          _gcry_mpi_ec_add_points (&tmppnt, result, point, ctx);
9          point_swap_cond (result, &tmppnt, mpi_test_bit (scalar, j), ctx);
10     }
11     point_free (&tmppnt);
12 } else {
13     for (j=nbits-1; j >= 0; j--) {
14         _gcry_mpi_ec_dup_point (result, result, ctx);
15         if (mpi_test_bit (scalar, j))
16             _gcry_mpi_ec_add_points (result, result, point, ctx);
17     }
18 }
```

INTERRUPT

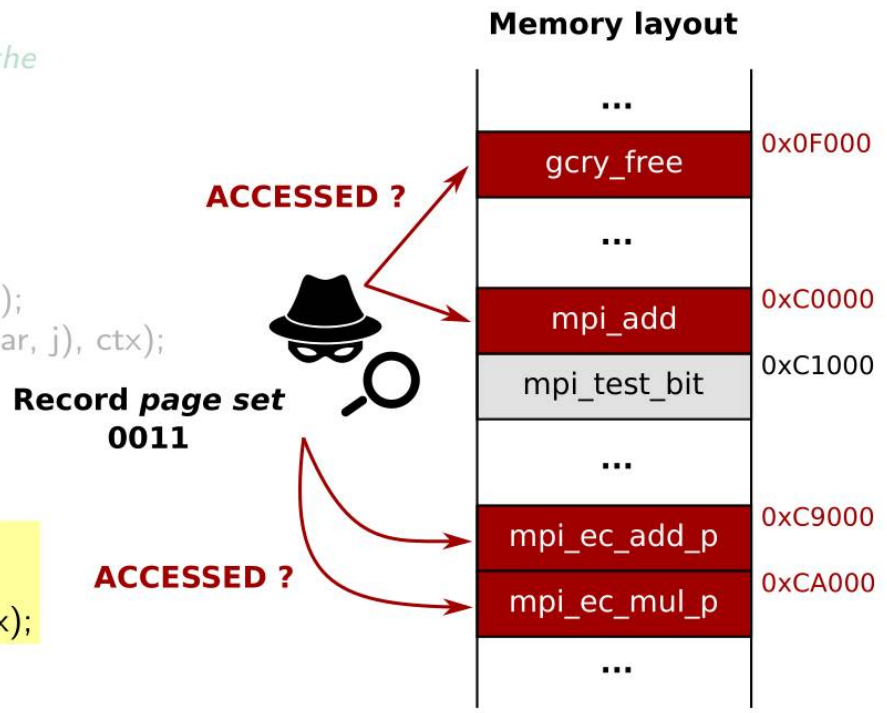
Memory layout



Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

Attacking Libgcrypt EdDSA (Simplified)

```
1  if (mpi_is_secure (scalar)) {
2      /* If SCALAR is in secure memory we assume that it is the
3         secret key we use constant time operation. */
4      point_init (&tmppnt);
5
6      for (j=nbits-1; j >= 0; j--) {
7          _gcry_mpi_ec_dup_point (result, result, ctx);
8          _gcry_mpi_ec_add_points (&tmppnt, result, point, ctx);
9          point_swap_cond (result, &tmppnt, mpi_test_bit (scalar, j), ctx);
10     }
11     point_free (&tmppnt);
12 } else {
13     for (j=nbits-1; j >= 0; j--) {
14         _gcry_mpi_ec_dup_point (result, result, ctx);
15         if (mpi_test_bit (scalar, j))
16             _gcry_mpi_ec_add_points (result, result, point, ctx);
17     }
18 }
```



Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

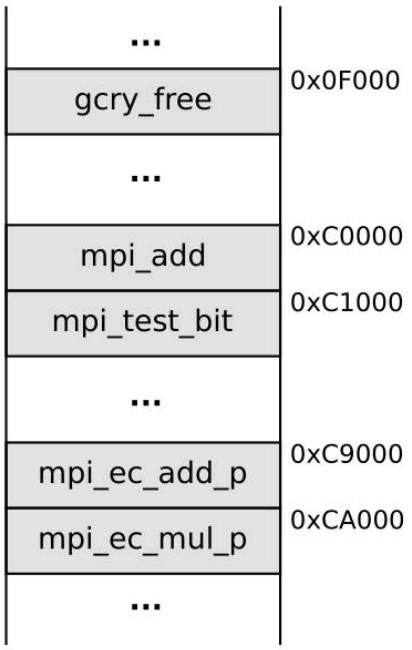
Attacking Libgcrypt EdDSA (Simplified)

```
1  if (mpi_is_secure (scalar)) {
2      /* If SCALAR is in secure memory we assume that it is the
3         secret key we use constant time operation. */
4      point_init (&tmppnt);
5
6      for (j=nbits-1; j >= 0; j--) {
7          _gcry_mpi_ec_dup_point (result, result, ctx);
8          _gcry_mpi_ec_add_points (&tmppnt, result, point, ctx);
9          point_swap_cond (result, &tmppnt, mpi_test_bit (scalar, j), ctx);
10     }
11     point_free (&tmppnt);
12 } else {
13     for (j=nbits-1; j >= 0; j--) {
14         _gcry_mpi_ec_dup_point (result, result, ctx);
15         if (mpi_test_bit (scalar, j))
16             _gcry_mpi_ec_add_points (result, result, point, ctx);
17     }
18 }
```

RESUME

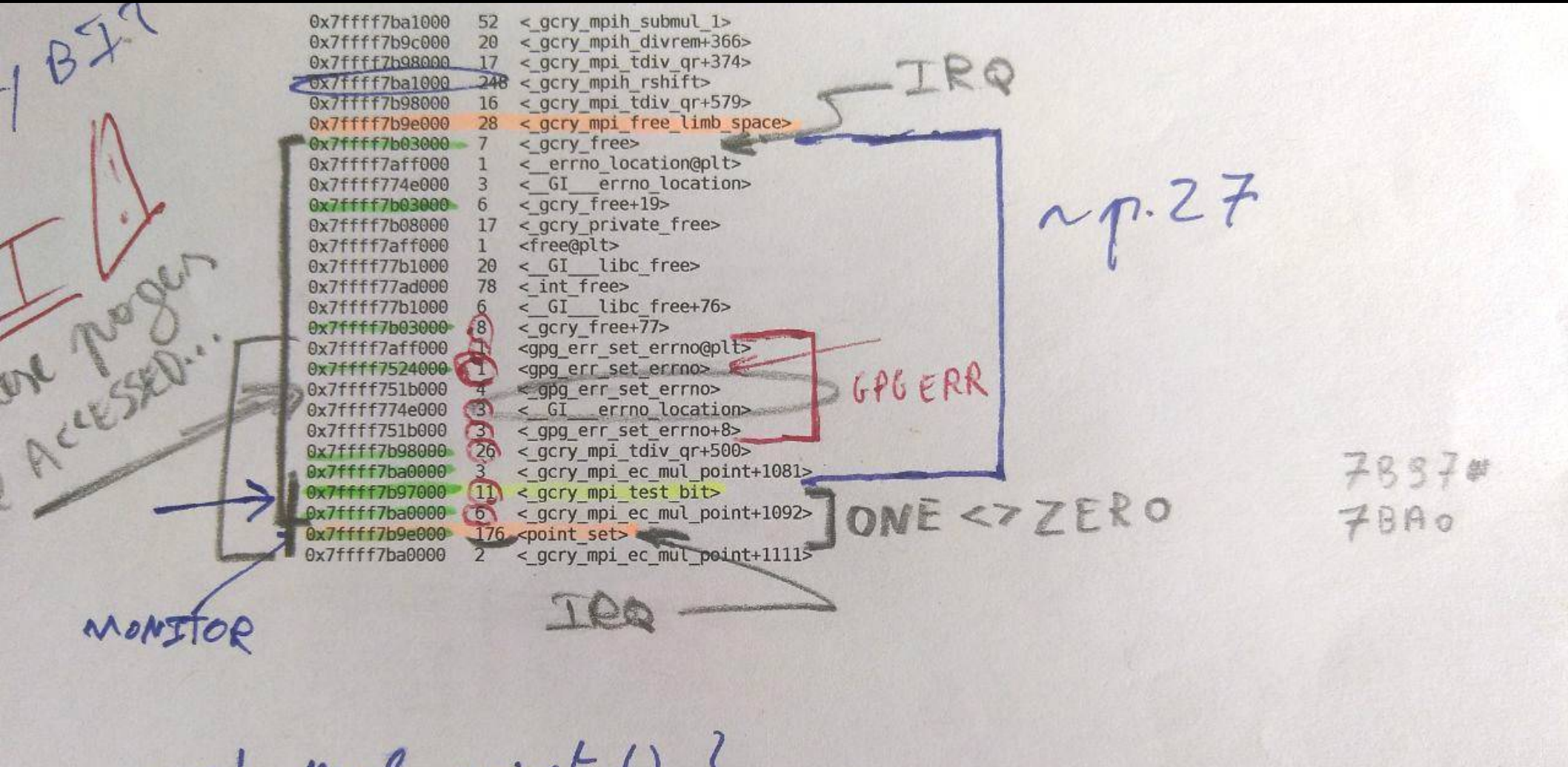
Full 512-bit key recovery, single run

Memory layout



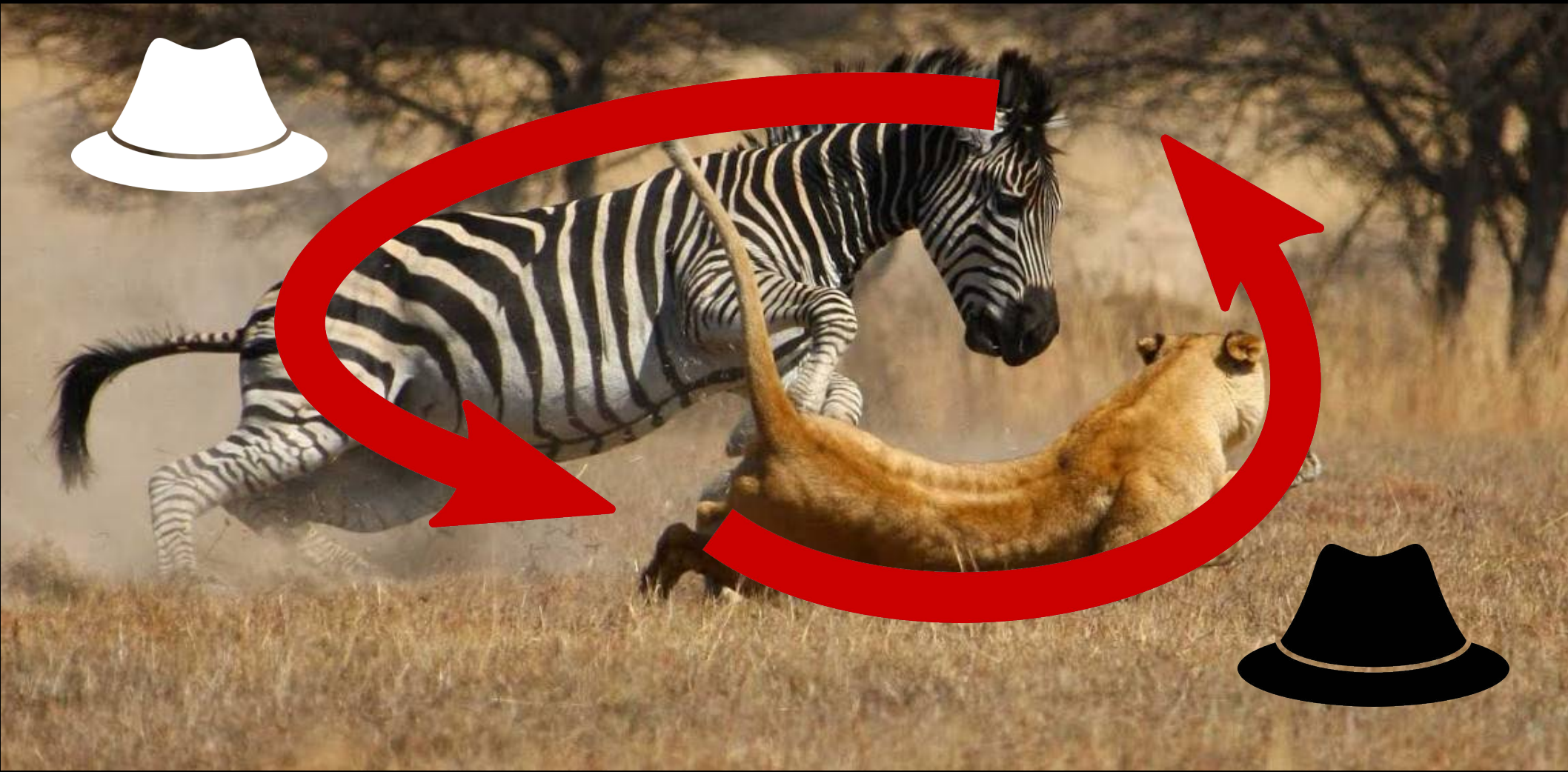
Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017.

Side-channel Analysis: From Metadata Patterns to Secrets

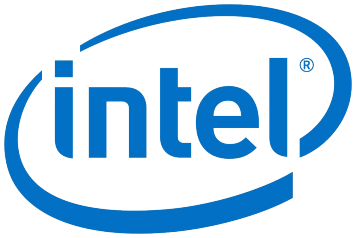


Scientific Understanding Driven by Attacker-Defender Race...

Scientific Understanding Driven by Attacker-Defender Race...



Idea #2 - Software-Based Attacks: Temporal Resolution?



Protection from Side-Channel Attacks

Intel® SGX does not provide explicit protection from side-channel attacks. It is the enclave developer's responsibility to address side-channel attack concerns.

In general, enclave operations that require an OCall, such as thread synchronization, I/O, etc., are exposed to the untrusted domain. If using an OCall would allow an attacker to gain insight into enclave secrets, then there would be a security concern. This scenario would be classified as a side-channel attack, and it would be up to the ISV to design the enclave in a way that prevents the leaking of side-channel information.

An attacker with access to the platform can see what pages are being executed or accessed. This side-channel vulnerability can be mitigated by aligning specific code and data blocks to exist entirely within a single page.

More important, the application enclave should use an appropriate crypto implementation that is side channel attack resistant inside the enclave if side-channel attacks are a concern.

Temporal Resolution Limitations for the Page-Fault Oracle

```c
size_t strlen (char *str)
{
  char *s;

  for (s = str; *s; ++s);
  return (s - str);
}
```

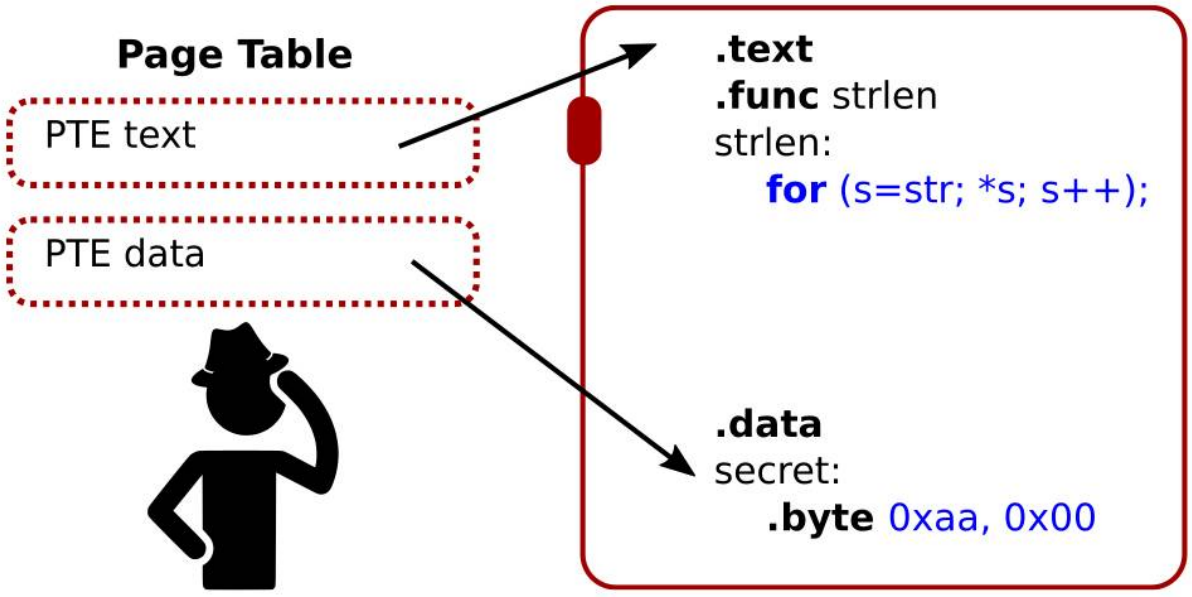
```
    mov  %rdi,%rax
1:  cmpb $0x0,(%rax)
    je   2f
    inc  %rax
    jmp  1b
2:  sub  %rdi,%rax
    retq
```

$\Rightarrow$ tight loop: 4 instructions, single memory operand, single code + data page

Counting `strlen` loop iterations?

Note: Page-fault attacks cannot make progress for 1 code + data page

Temporal Resolution Limitations for the Page-Fault Oracle

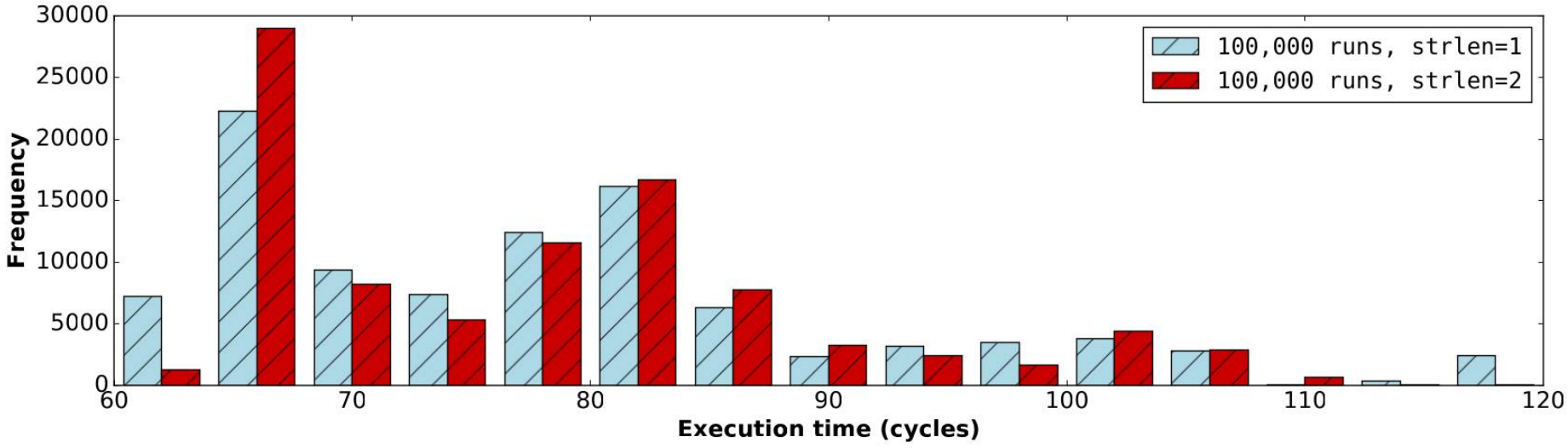


Counting `strlen` loop iterations?

Progress requires both pages present (non-faulting) ↔ page fault oracle

Building the Side-Channel Oracle with Execution Timing?

Too noisy: modern x86 processors are lightning fast. . .



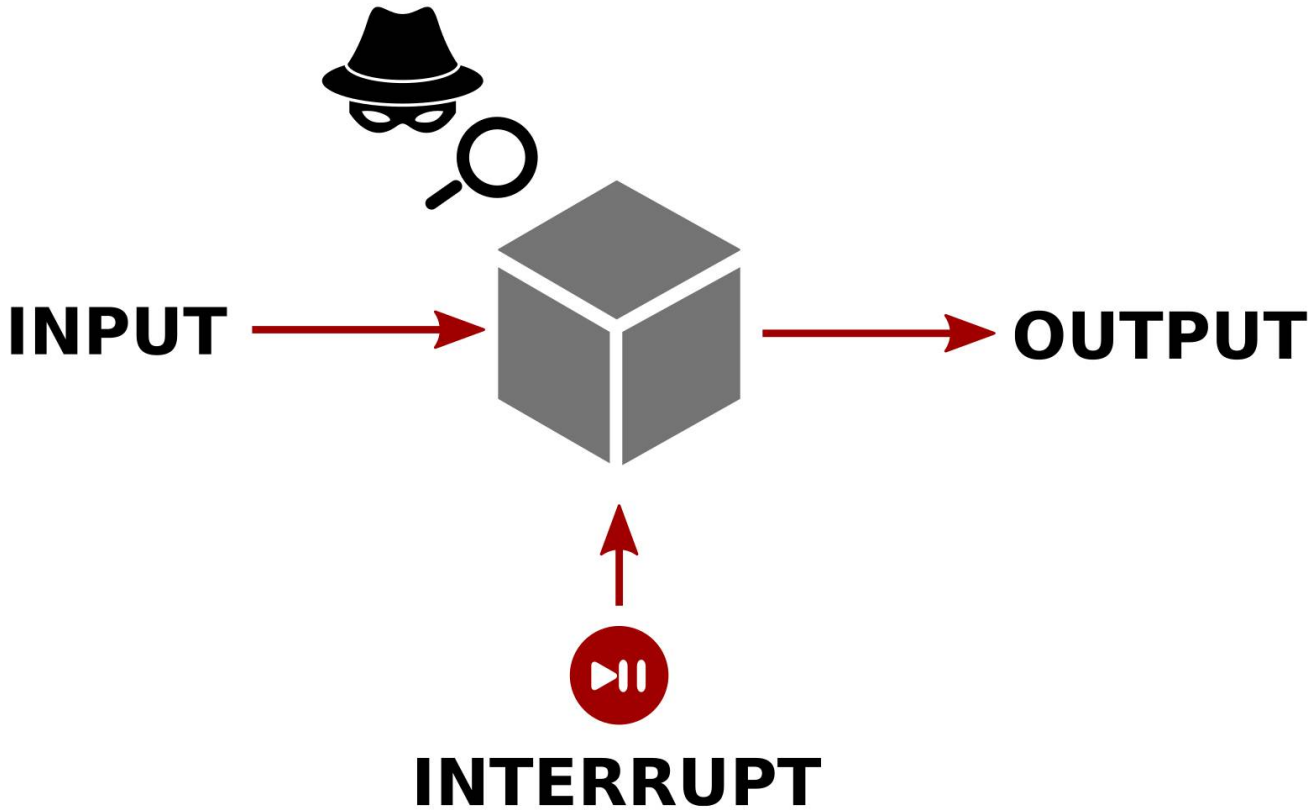
Challenge: Side-Channel Sampling Rate

Slow
shutter speed

Medium
shutter speed

Fast
shutter speed

SGX-Step: Executing Enclaves one Instruction at a Time



SGX-Step: Executing Enclaves one Instruction at a Time

SGX-Step

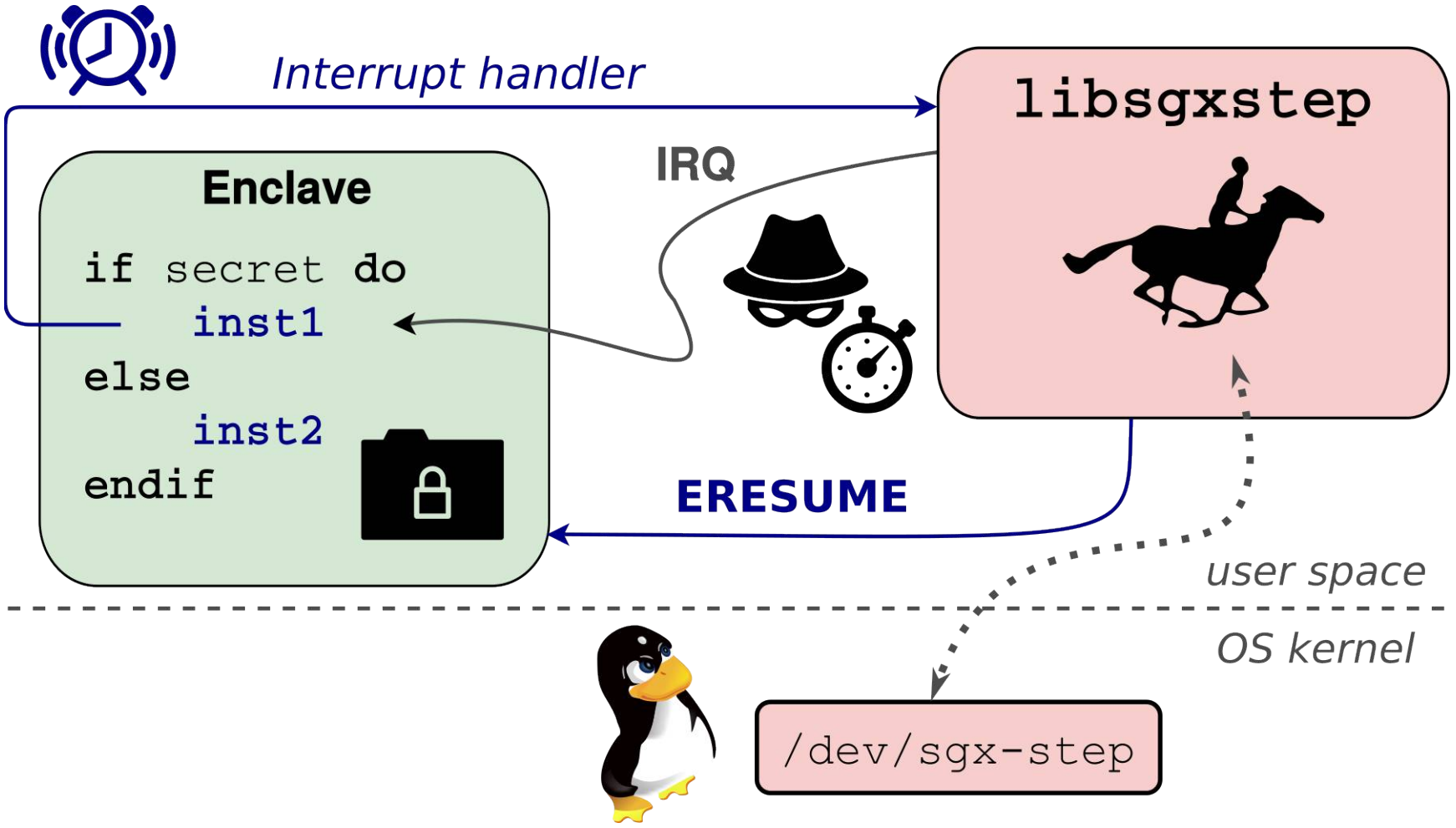
ACSAC 2023
Cybersecurity Artifacts
Impact Award

https://github.com/jovanbulck/sgx-step

Watch	22	Star	245	Fork	52

Van Bulck et al., “SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control”, SysTEX 2017.

SGX-Step: Executing Enclaves one Instruction at a Time



SGX-Step Demo: Single-Stepping Password Comparison

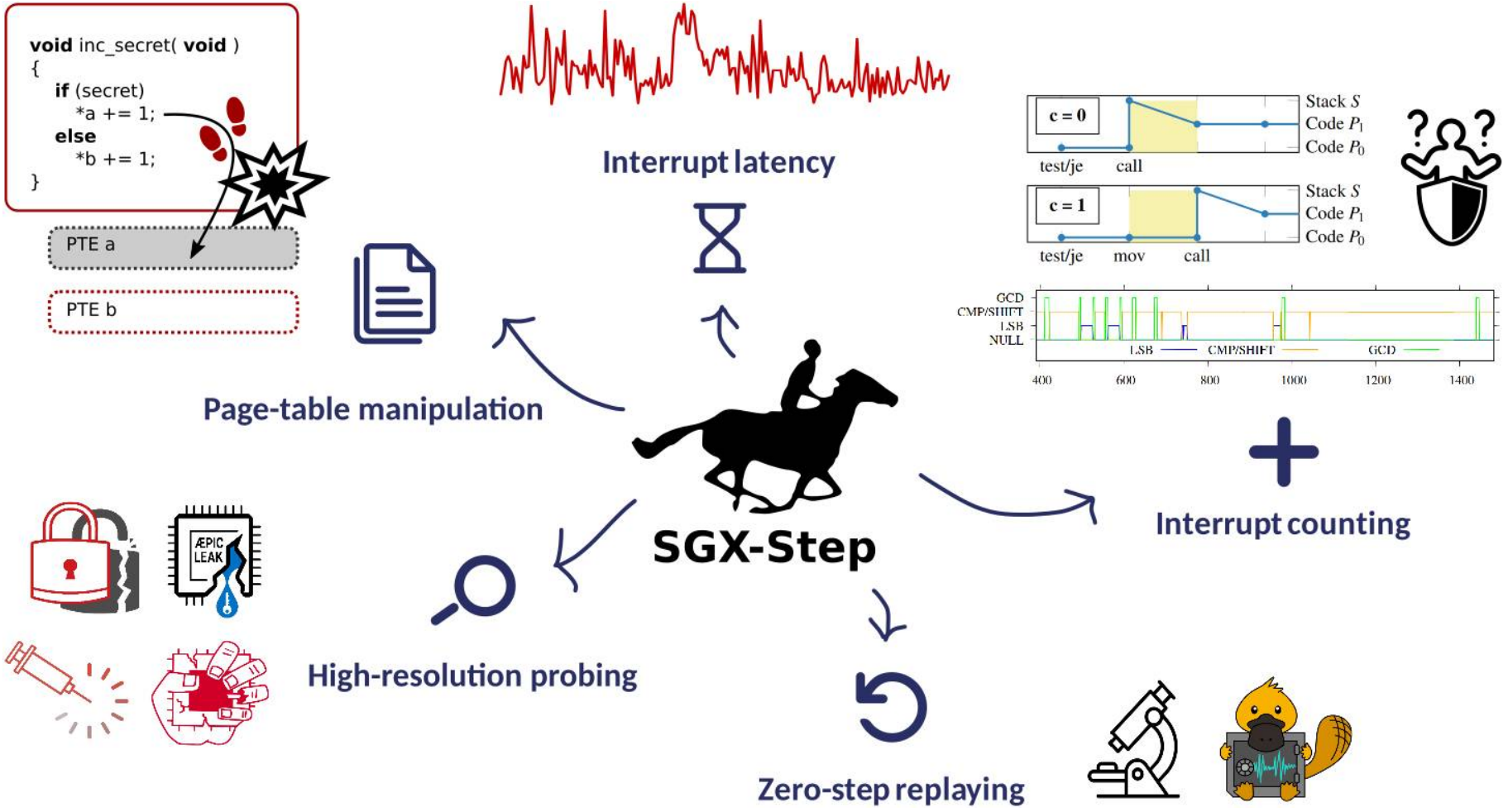
```
[idt.c] IDT[ 45] @0x7f83225492d0 = 0x556b4424b000 (seg sel 0x10); p=1; dpl=3; type=14; ist=0
[file.c] reading buffer from '/dev/cpu/1/msr' (size=8)
[apic.c] established local memory mapping for APIC_BASE=0xfee00000 at 0x7f8322548000
[apic.c] APIC_ID=2000000; LVTT=400ec; TDCR=0
[apic.c] APIC timer one-shot mode with division 2 (lvtt=2d/tdcr=0)

------------------------------------------------------------------------------
[main.c] recovering password length
------------------------------------------------------------------------------

[attacker] steps=15; guess='******'
[attacker] found pwd len = 6

------------------------------------------------------------------------------
[main.c] recovering password bytes
------------------------------------------------------------------------------

[attacker] steps=35; guess='SECRET' --> SUCCESS

[apic.c] Restored APIC_LVTT=400ec/TDCR=0)
[file.c] writing buffer to '/dev/cpu/1/msr' (size=8)
[main.c] all done; counted 2413/2336 IRQs (AEP/IDT)
jo@breuer:~/sgx-step-demo$
```

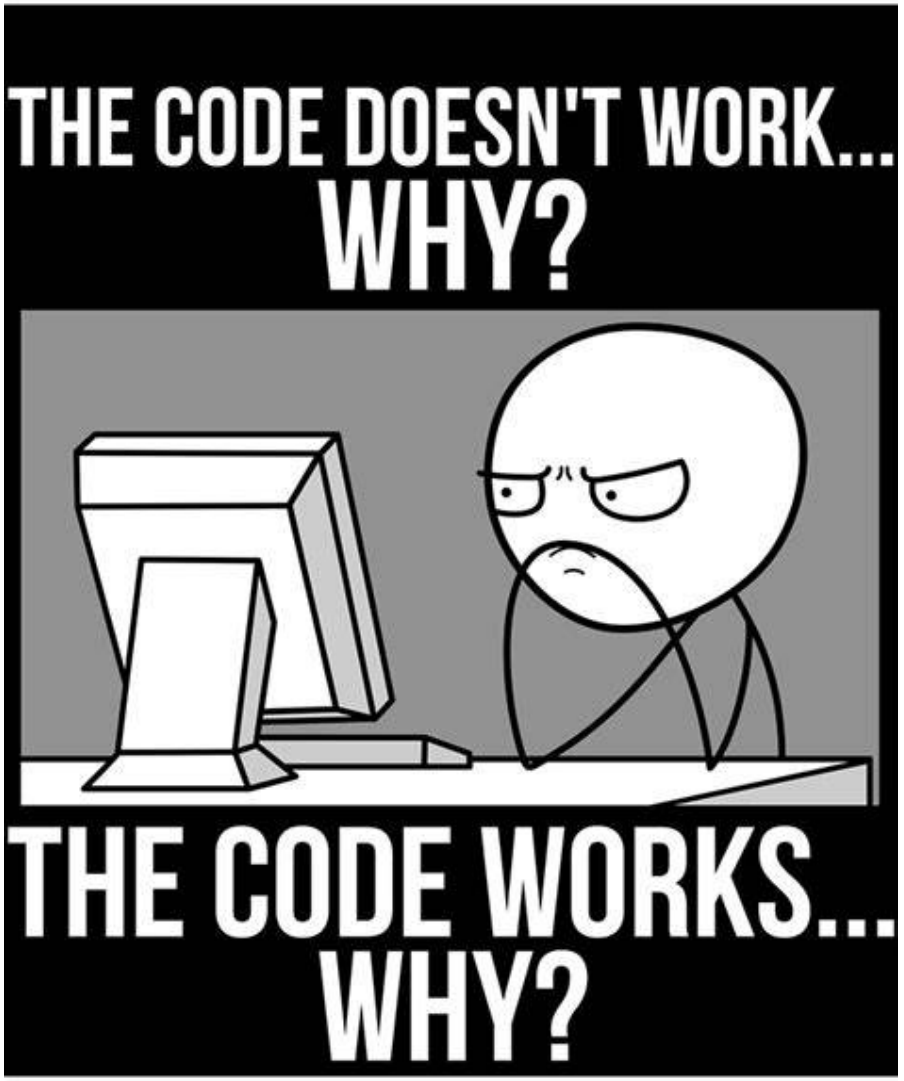
SGX-Step: Enabling a New Line of High-Resolution Attacks

Yr	Venue	Paper	Step	Use Case	Drv
'15	S&P	Ctrl channel	~ Page	*Probe* (page fault)	✓
'16	ESORICS	AsyncShock	~ Page	*Exploit* (mem safety)	–
'17	CHES	CacheZoom	✗ >1	*Probe* (L1 cache)	✓
'17	ATC	Hahnel et al.	✗ 0 - >1	*Probe* (L1 cache)	✓
'17	USENIX	BranchShadow	✗ 5 - 50	*Probe* (BPU)	✗
'17	USENIX	Stealthy PTE	~ Page	*Probe* (page table)	✓
'17	USENIX	DarkROP	~ Page	*Exploit* (mem safety)	✓
'17	SysTEX	SGX-Step	✓ 0 - 1	*Framework*	✓
'18	ESSoS	Off-limits	✓ 0 - 1	*Probe* (segmentation)	✓
'18	AsiaCCS	Single-trace RSA	~ Page	*Probe* (page fault)	✓
'18	USENIX	Foreshadow	✓ 0 - 1	*Probe* (transient exec)	✓
'18	EuroS&P	SgxPectre	~ Page	*Exploit* (transient)	✓
'18	CHES	CacheQuote	✗ >1	*Probe* (L1 cache)	✓
'18	ICCD	SGXlinger	✗ >1	*Probe* (IRQ latency)	✗
'18	CCS	Nemesis	✓ 1	*Probe* (IRQ latency)	✓
'19	USENIX	Spoiler	✓ 1	*Probe* (IRQ latency)	✓
'19	CCS	ZombieLoad	✓ 0 - 1	*Probe* (transient exec)	✓
'19	CCS	Fallout	–	*Probe* (transient exec)	✓
'19	CCS	Tale of 2 worlds	✓ 1	*Exploit* (mem safety)	✓
'19	ISCA	MicroScope	~ 0 - Page	*Framework*	✗
'20	CHES	Bluethunder	✓ 1	*Probe* (BPU)	✓
'20	USENIX	Big troubles	~ Page	*Probe* (page fault)	✓
'20	S&P	Plundervolt	–	*Exploit* (undervolt)	✓
'20	CHES	Viral primitive	✓ 1	*Probe* (IRQ count)	✓
'20	USENIX	CopyCat	✓ 1	*Probe* (IRQ count)	✓
'20	S&P	LVI	✓ 1	*Exploit* (transient)	✓

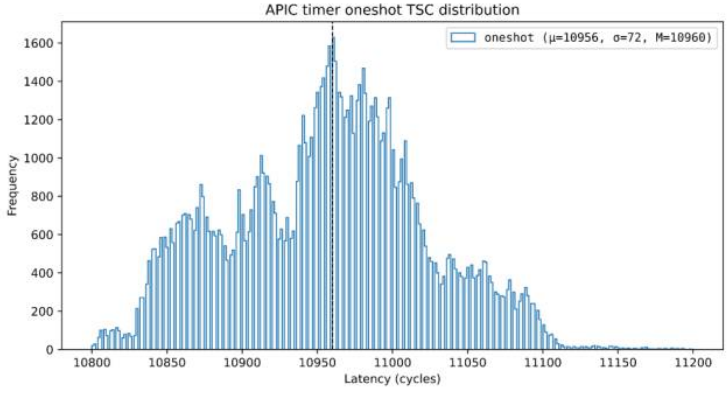
Yr	Venue	Paper	Step	Use Case	Drv
'20	CHES	A to Z	~ Page	*Probe* (page fault)	✓
'20	CCS	Déjà Vu NSS	~ Page	*Probe* (page fault)	✓
'20	MICRO	PTHammer	–	*Probe* (page walk)	✓
'21	USENIX	Frontal	✓ 1	*Probe* (IRQ latency)	✓
'21	S&P	CrossTalk	✓ 1	*Probe* (transient exec)	✓
'21	CHES	Online template	✓ 1	*Probe* (IRQ count)	✓
'21	NDSS	SpeechMiner	–	*Framework*	✓
'21	S&P	Platypus	✓ 0 - 1	*Probe* (voltage)	✓
'21	DIMVA	Aion	✓ 1	*Probe* (cache)	✓
'21	CCS	SmashEx	✓ 1	*Exploit* (mem safety)	✓
'21	CCS	Util::Lookup	✓ 1	*Probe* (L3 cache)	✓
'22	USENIX	Rapid prototyping	✓ 1	*Framework*	✓
'22	CT-RSA	Kalyna expansion	✓ 1	*Probe* (L3 cache)	✓
'22	SEED	Enclyzer	–	*Framework*	✓
'22	NordSec	Self-monitoring	~ Page	*Defense* (detect)	✓
'22	AutoSec	Robotic vehicles	✓ 1 - >1	*Exploit* (timestamp)	✓
'22	ACSAC	MoLE	✓ 1	*Defense* (randomize)	✓
'22	USENIX	AEPIC	✓ 1	*Probe* (I/O device)	✓
'22	arXiv	Confidential code	✓ 1	*Probe* (IRQ latency)	✓
'23	ComSec	FaultMorse	~ Page	*Probe* (page fault)	✓
'23	CHES	HQC timing	✓ 1	*Probe* (L3 cache)	✓
'23	ISCA	Belong to us	✓ 1	*Probe* (BPU)	✓
'23	USENIX	BunnyHop	✓ 1	*Probe* (BPU)	✓
'23	USENIX	DownFall	✓ 0 - 1	*Probe* (transient exec)	✓
'23	USENIX	AEX-Notify	✓ 1	*Defense* (prefetch)	✓

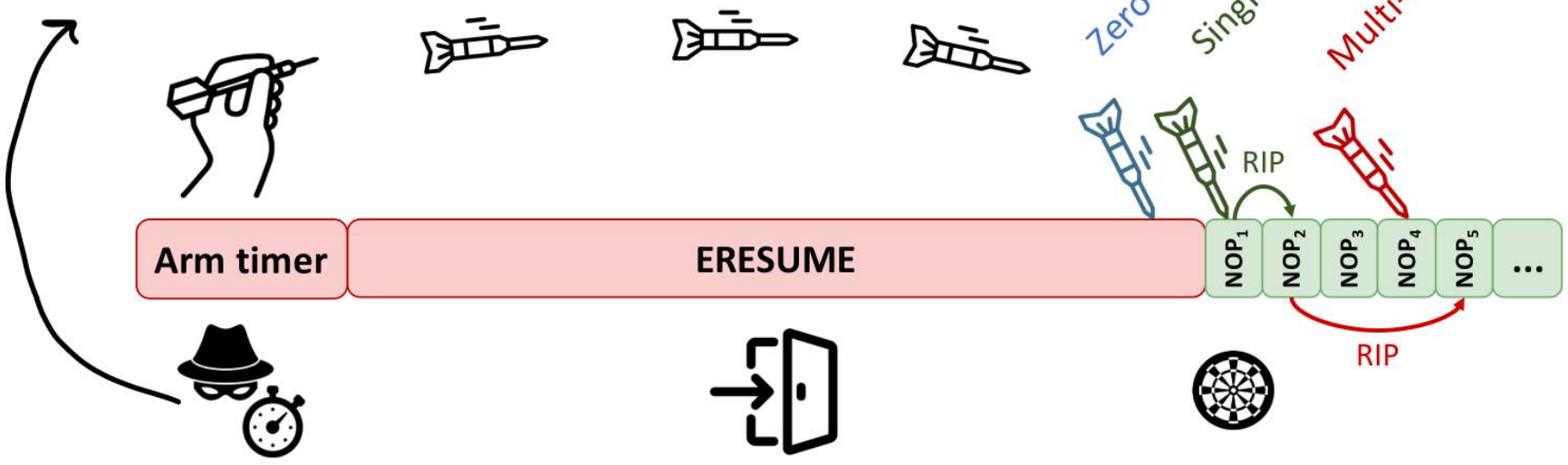
SGX-Step: A Versatile Open-Source Attack Framework




THE CODE DOESN'T WORK...
WHY?
THE CODE WORKS...
WHY?

Root-Causing SGX-Step: Aiming the Timer Interrupt

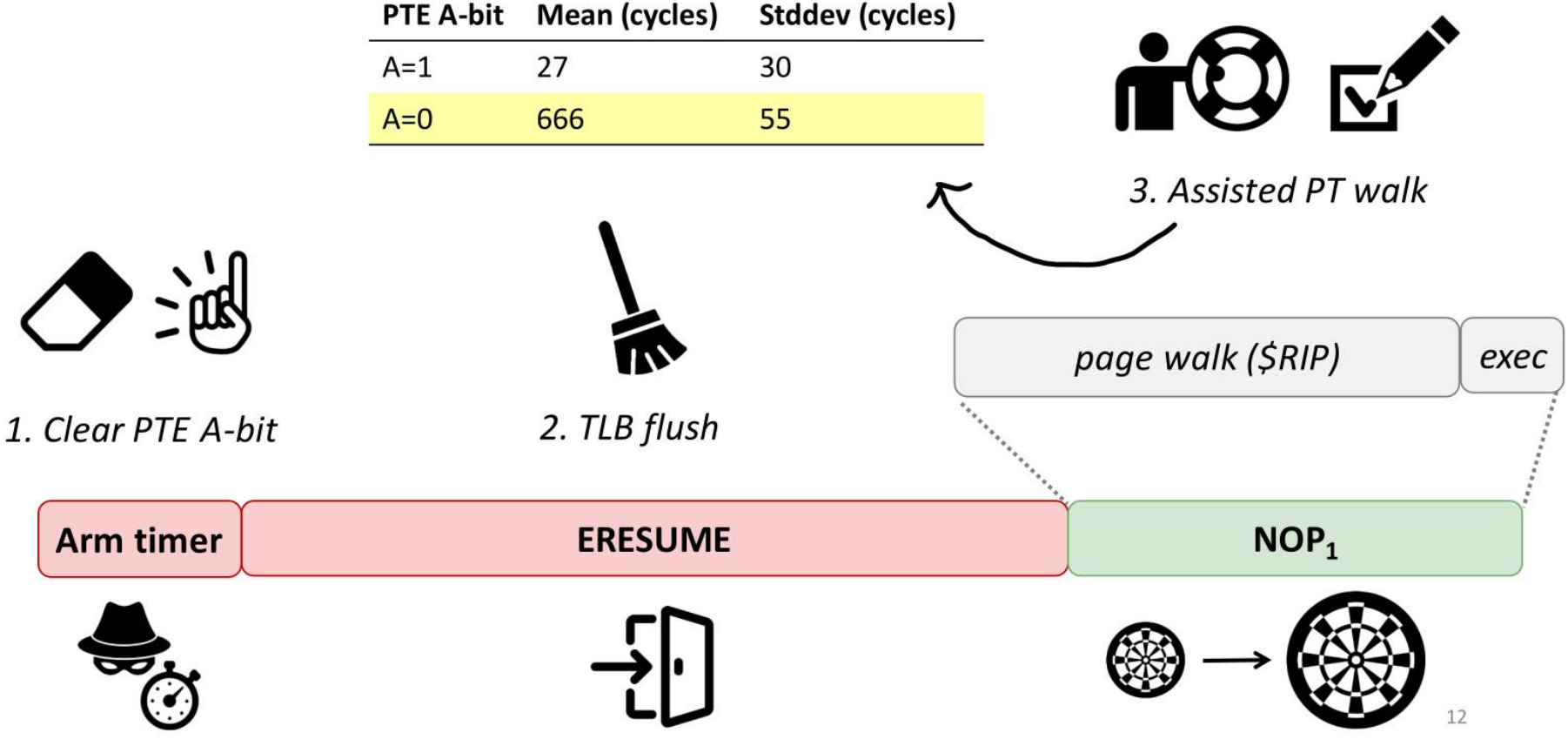




Constable et al. “AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves”, USENIX Security 2023.

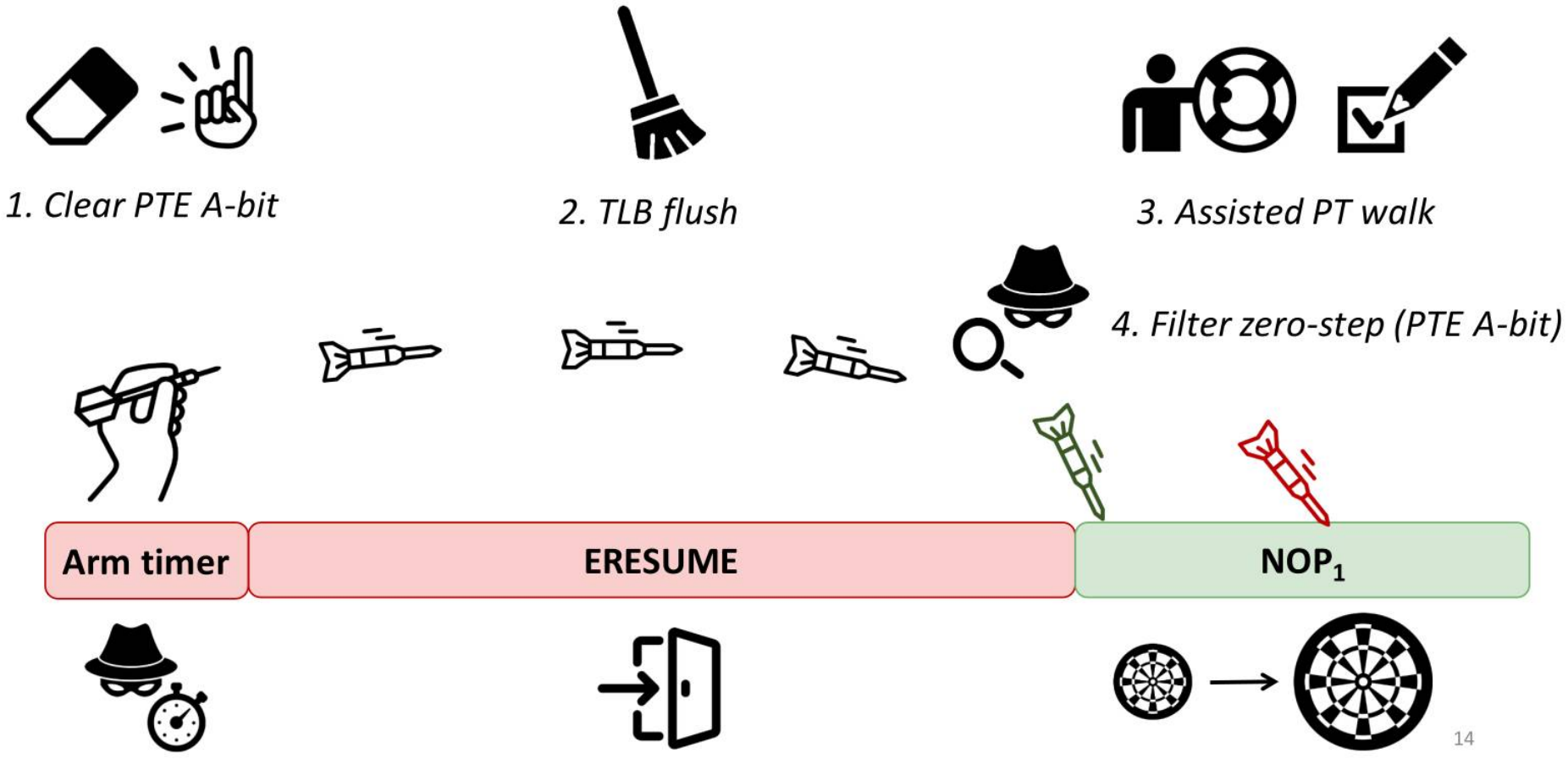
Root-Causing SGX-Step: Microcode Assists to the Rescue!

PTE A-bit	Mean (cycles)	Stddev (cycles)
A=1	27	30
A=0	666	55



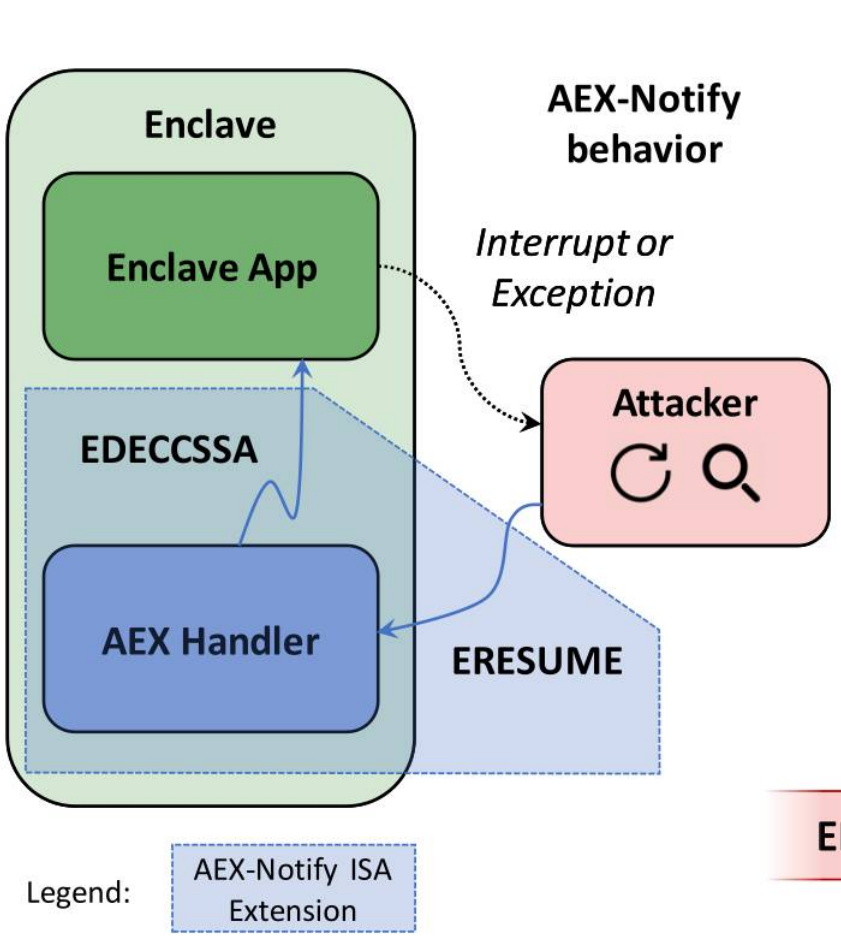
Constable et al. “AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves”, USENIX Security 2023.

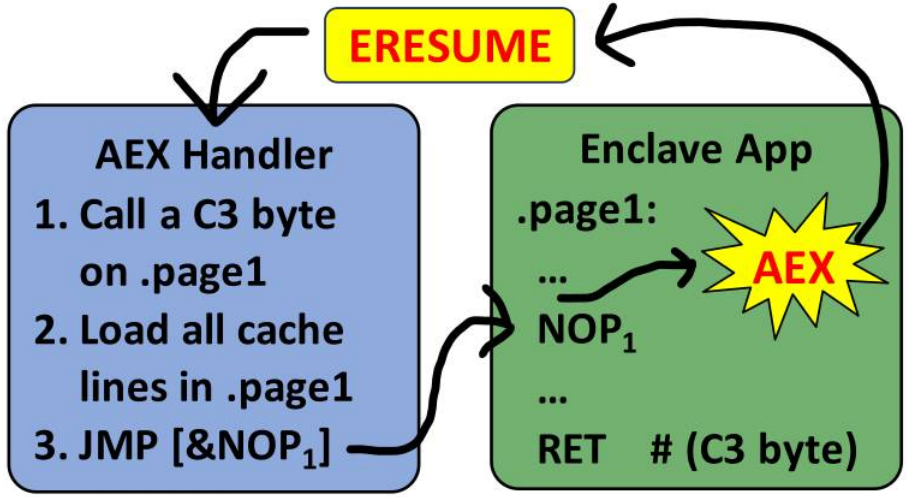
Root-Causing SGX-Step: Microcode Assists to the Rescue!

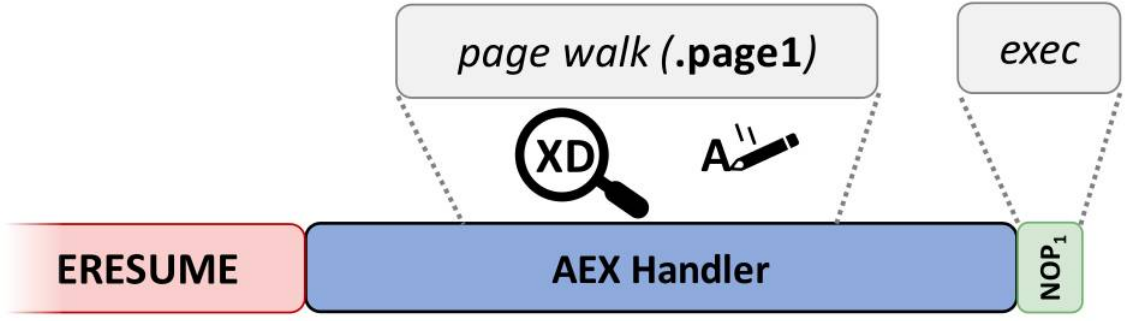


Constable et al. “AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves”, USENIX Security 2023.

AEX-Notify: Hardware-Software Co-Design Solution



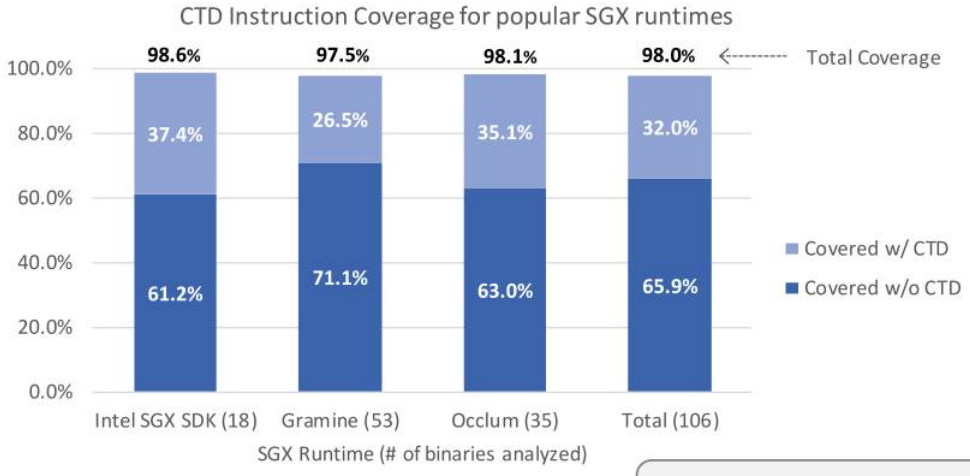


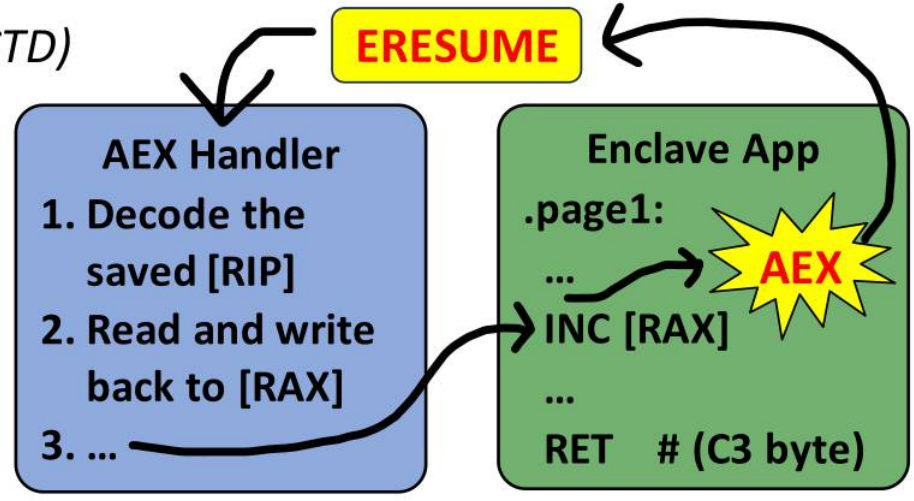


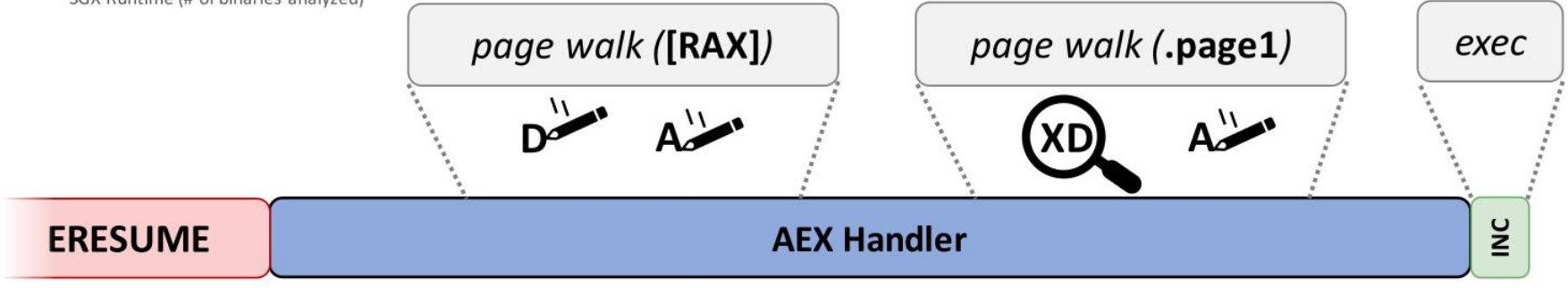
Constable et al. “AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves”, USENIX Security 2023.

AEX-Notify: Hardware-Software Co-Design Solution

We implemented a fast, constant-time decoder (CTD)







Constable et al. “AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves”, USENIX Security 2023.

intel

CHAPTER 8
ASYNCHRONOUS ENCLAVE EXIT NOTIFY AND THE EDECCSSA USER LEAF FUNCTION

8.1 INTRODUCTION

Asynchronous Enclave Exit Notify (AEX-Notify) is an extension to Intel® SGX that allows Intel SGX enclaves to be notified after an asynchronous enclave exit (AEX) has occurred. EDECCSSA is a new Intel SGX user leaf function (ENCLU[EDECCSSA]) that can facilitate AEX notification handling, as well as software exception handling. This chapter provides information about changes to the Intel SGX architecture that support AEX-Notify and ENCLU[EDECCSSA].

The following list summarizes the a
details are provided in Section 8.3)

- SECS.ATTRIBUTES.AEXNOTIFY
- TCS.FLAGS.AEXNOTIFY: This e
- SSA.GPRSGX.AEXNOTIFY: Enclave-writable byte that allows enclave software to dynamically enable/disable AEX notifications.

An AEX notification is delivered by ENCLU[ERESUME] when the following conditions are met:

SGX-Step led to **new x86 processor instructions!**

→ shipped in millions of devices ≥ 4th Gen Xeon CPU

phoronix

ARTICLES & REVIEWS | NEWS ARCHIVE | FORUMS | PREMIUM | CONTACT | CATEGORIES

Intel AEX Notify Support Prepped For Linux To Help Enhance SGX Enclave Security

Written by Michael Larabel in Intel on 6 November 2022 at 06:01 AM EST. 5 Comments

Future Intel CPUs and some existing processors via a microcode update will support a new feature called the Asynchronous EXit (AEX) notification mechanism to help with Software Guard Extensions (SGX) enclave security. Patches for the Linux kernel are pending for implementing this Intel AEX Notify support with capable processors.

Intel's Asynchronous EXit (AEX) notification mechanism lets SGX enclaves run a handler after an AEX event. Those handlers can be used for things like mitigating SGX-Step as an attack framework for precise enclave execution control.

Code 1 in intel/linux-sgx Filter

sdk/trts/linux/trts_mitigation.S

```
48   * Description:
49   *     The file provides mitigations for SGX-Step
50   */
71   *  Function:
constant_time_apply_sgxstep_mitigation_and_continue_execution
72   *      Mitigate SGX-Step and return to the point at which the
most recent
73   *      interrupt/exception occurred.
```

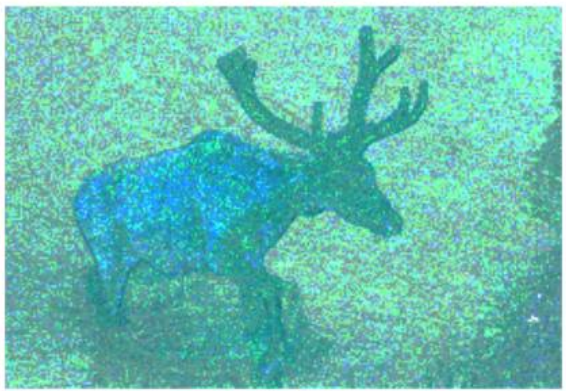
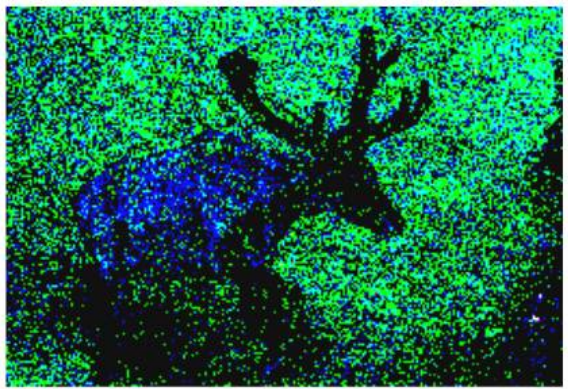
SGX-Step led to **changes in major OSs and enclave SDKs**

There's a Catch...

Finally note that our proposed mitigation does not protect against interrupting enclaves and observing application code and data page accesses at a coarse-grained 4 KiB spatial resolution. In contrast to the fine-grained, instruction-granular interrupt-driven attacks we consider in this work, such controlled-channel attacks have received ample attention [18, 47, 56, 59] from the research community.

Constable et al. "AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves", USENIX Security 2023.

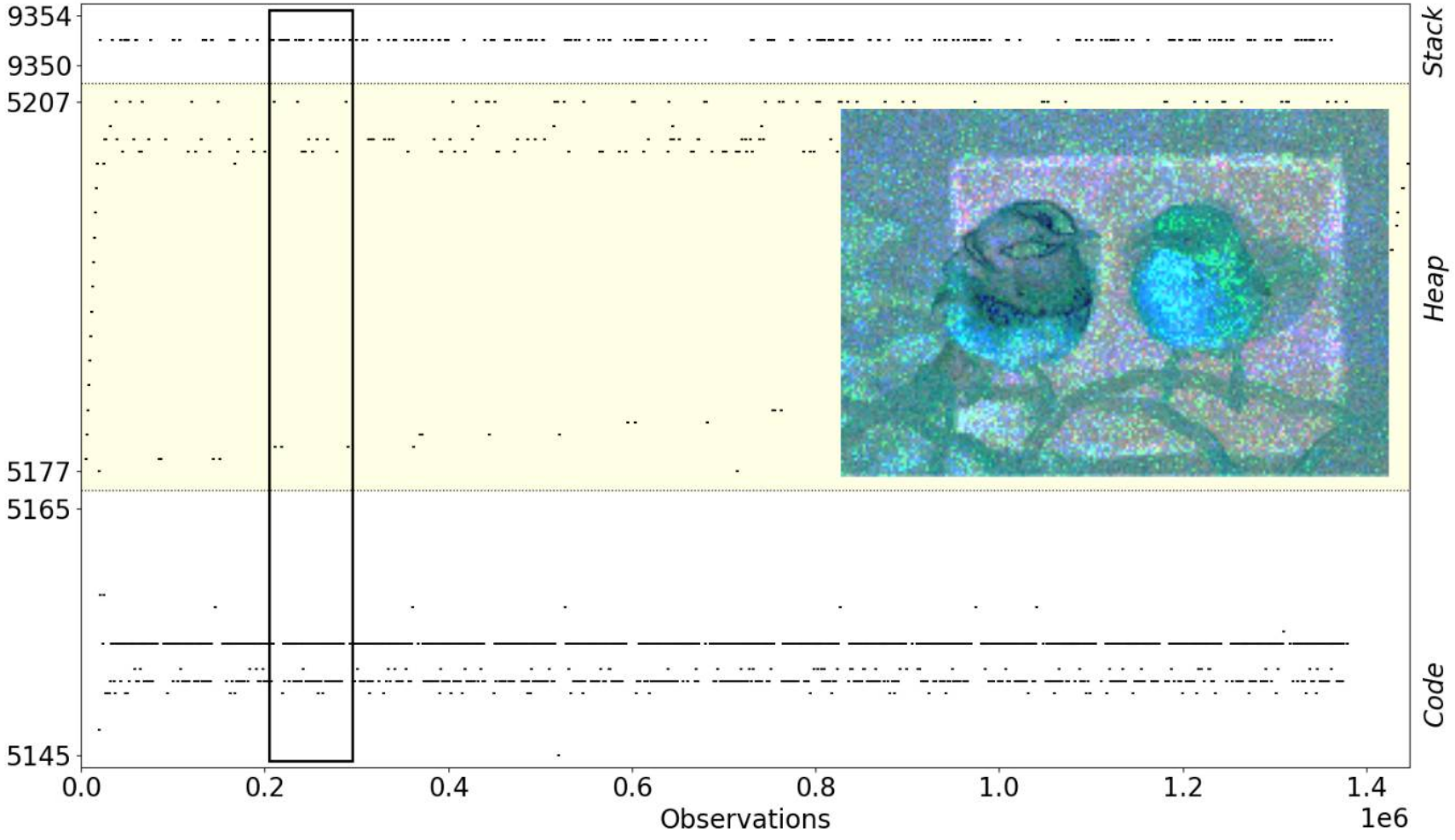
Why Mitigating Single-Stepping is Not Enough

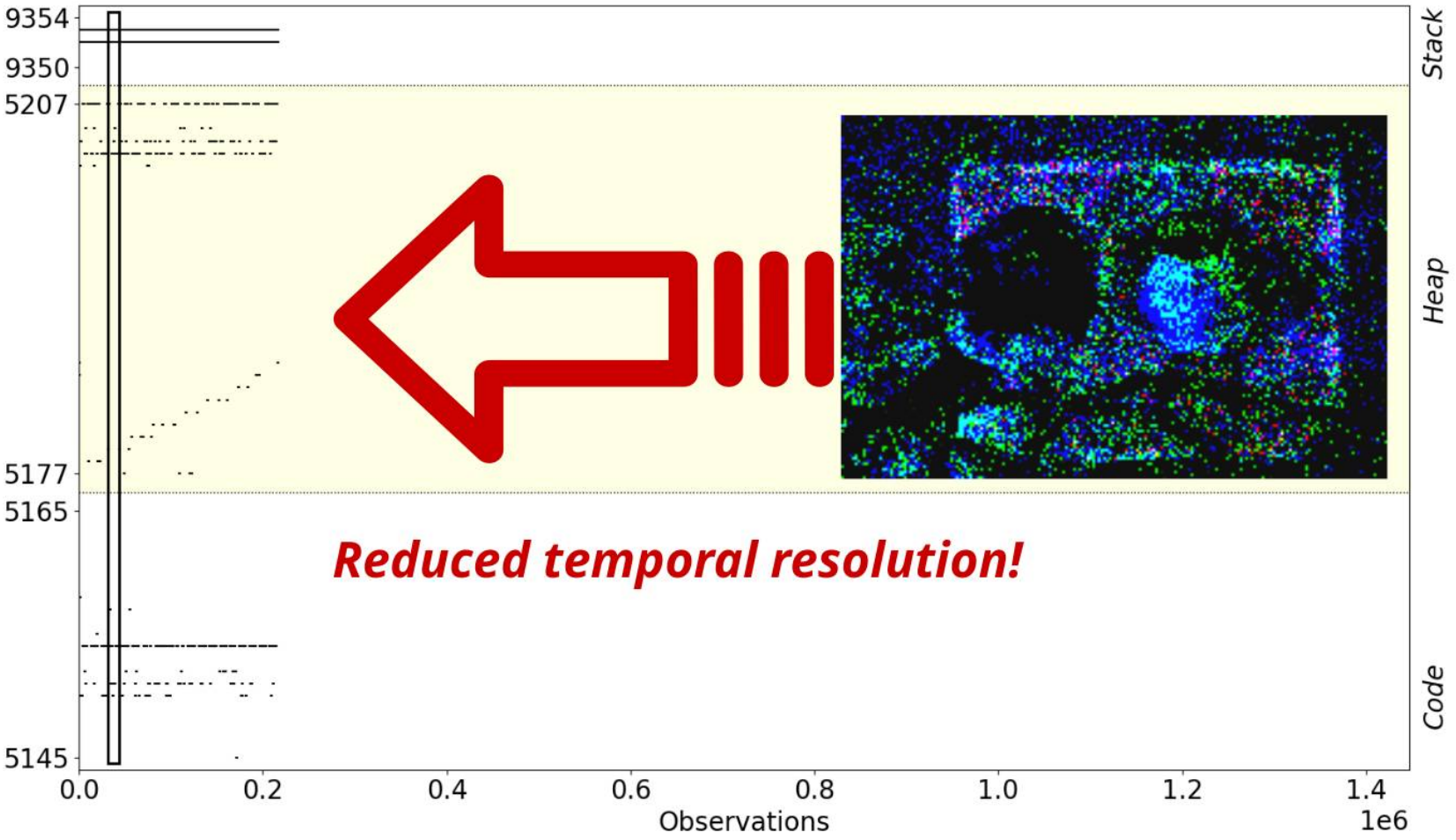
Original (left), Xu et al. (middle), our attack with AEX-Notify single-stepping defense (right)

Vanoverloop et. al “TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms”, USENIX’25.

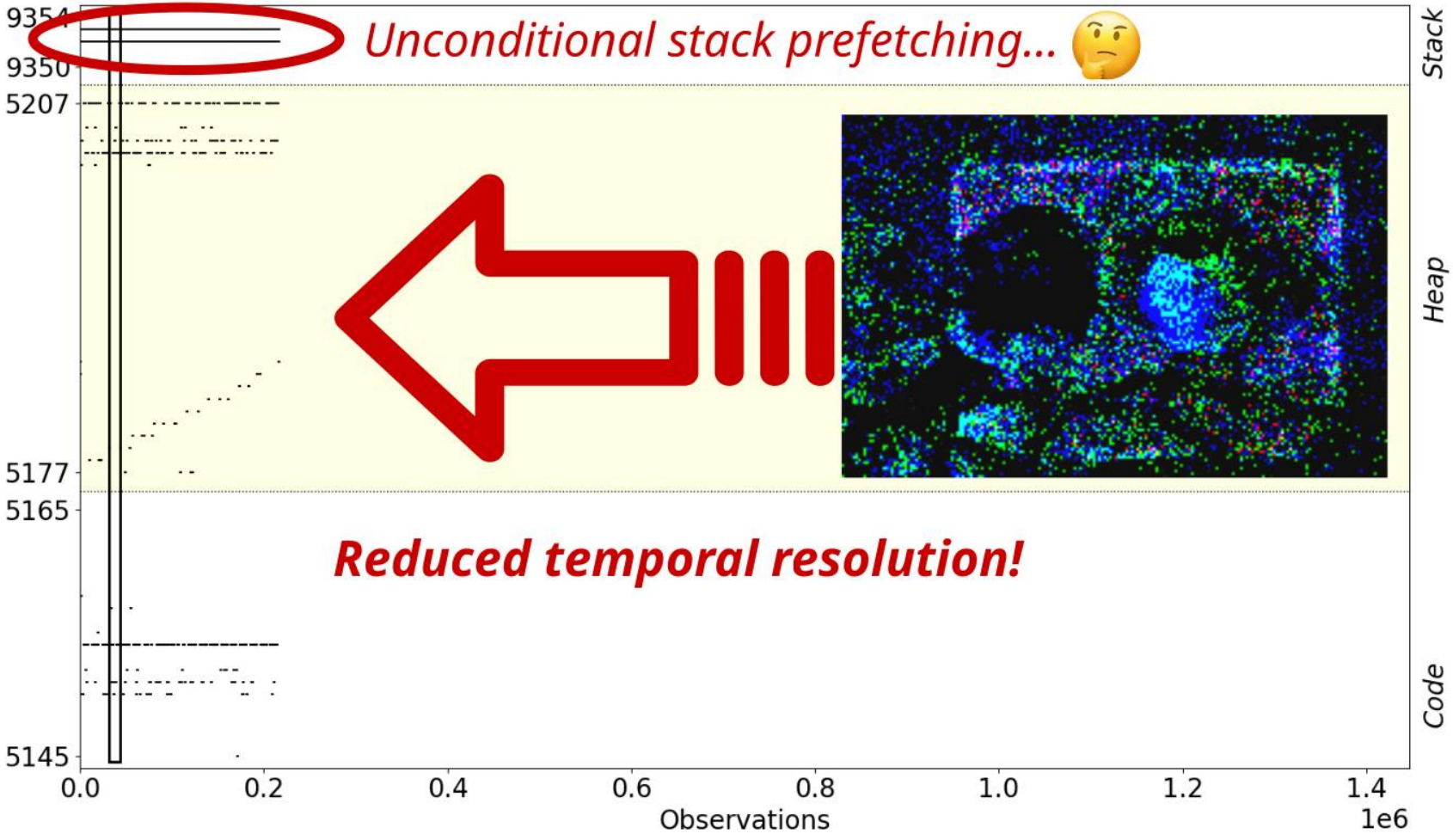
Libjpeg: AEX-Notify's Temporal Reduction in Practice



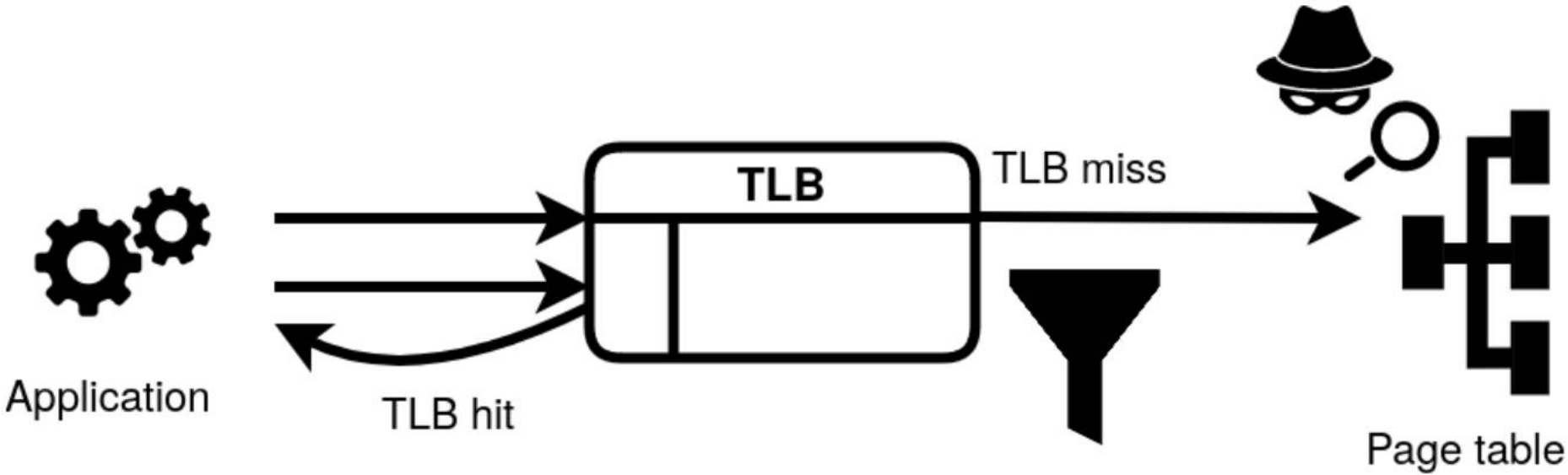
Libjpeg: AEX-Notify's Temporal Reduction in Practice



Libjpeg: AEX-Notify's Temporal Reduction in Practice

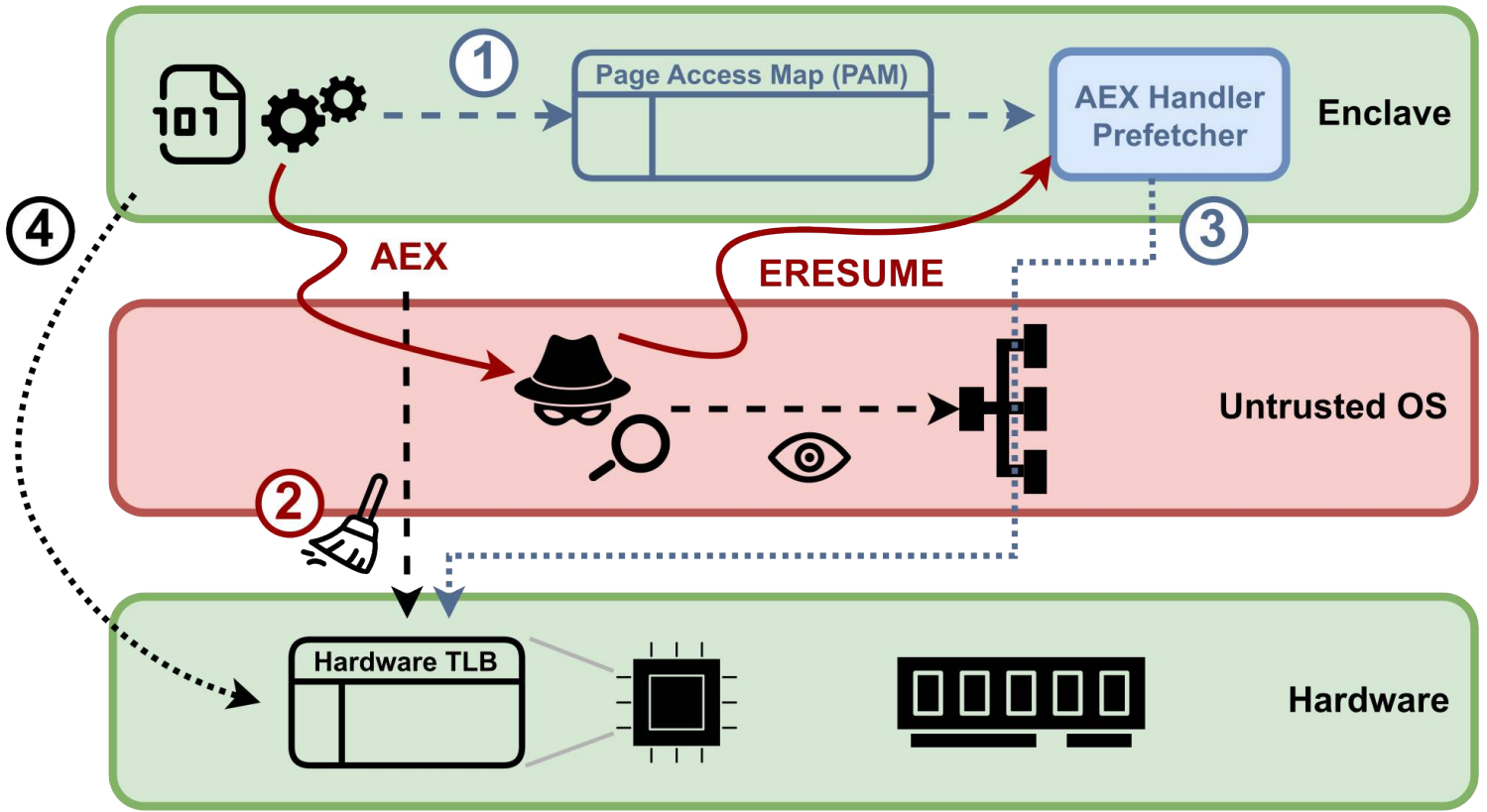


Idea: TLB as a “Filter” to Hide Page Accesses



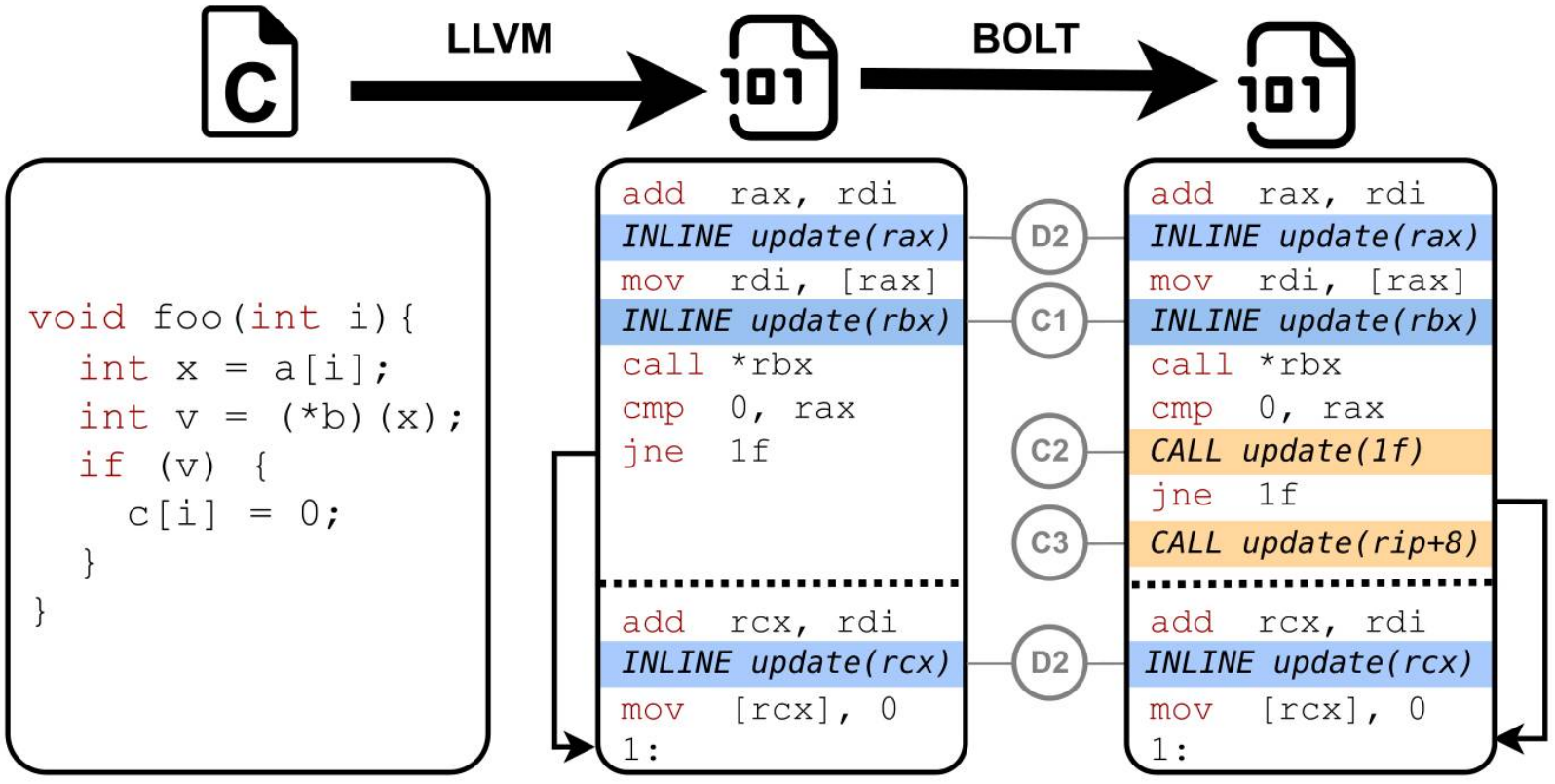
Vanoverloop et. al “TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms”, USENIX’25.

TLBlur: Self-Monitoring and Restoring Enclave Page Accesses



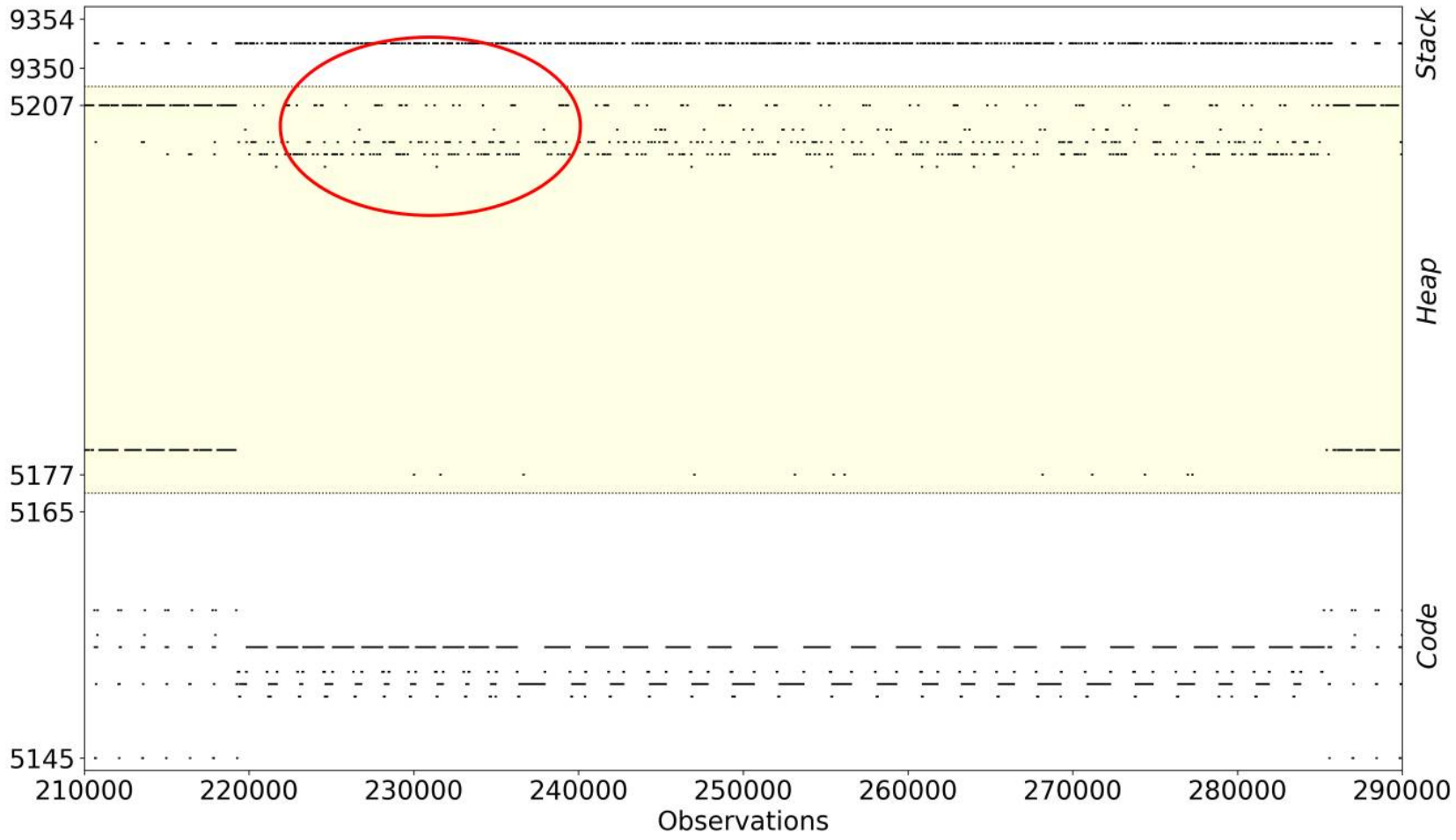
Vanoverloop et. al "TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms", USENIX'25.

Instrumentation to Self-Monitor Page Accesses at Runtime

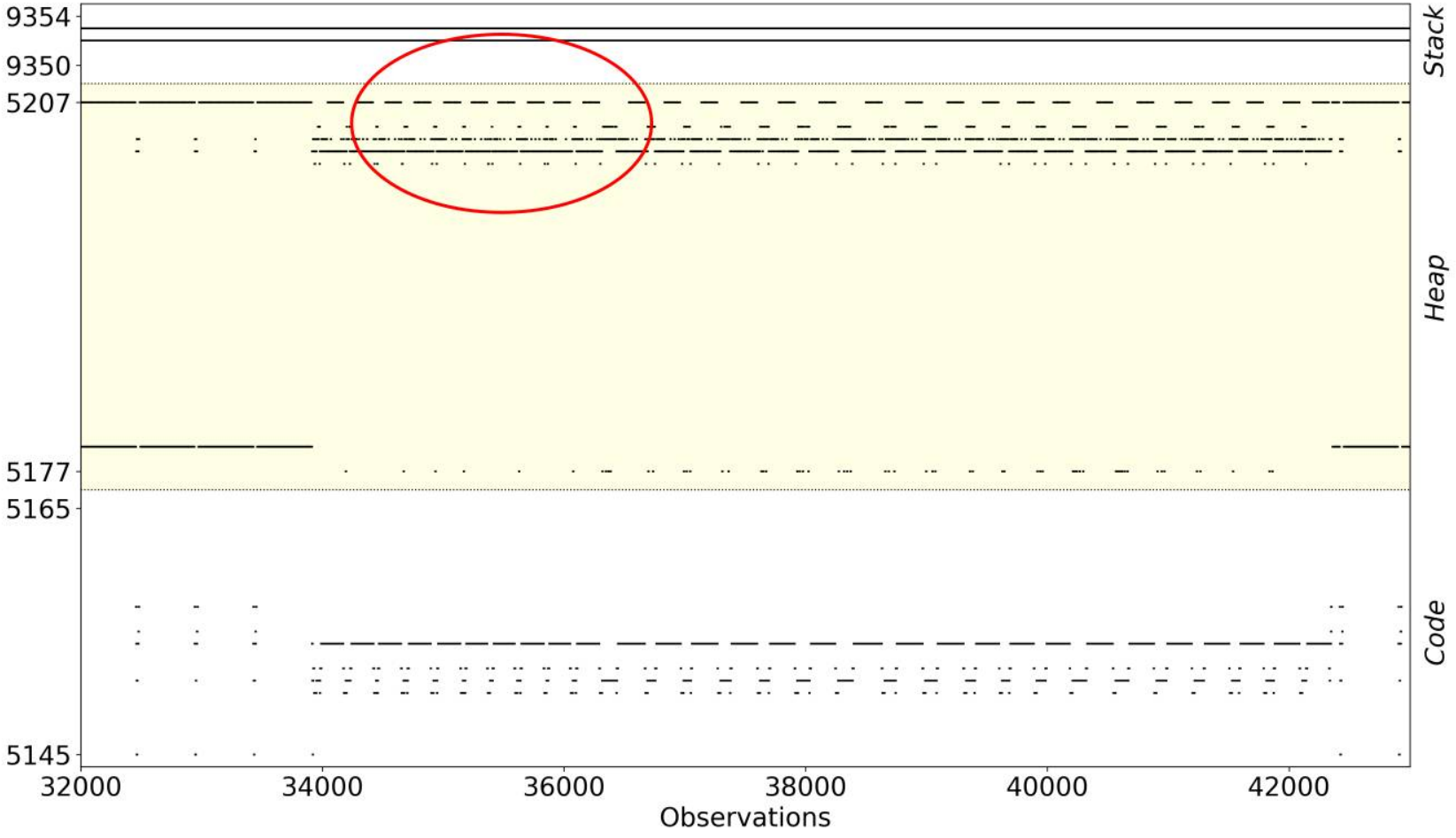


Vanoverloop et. al "TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms", USENIX'25.

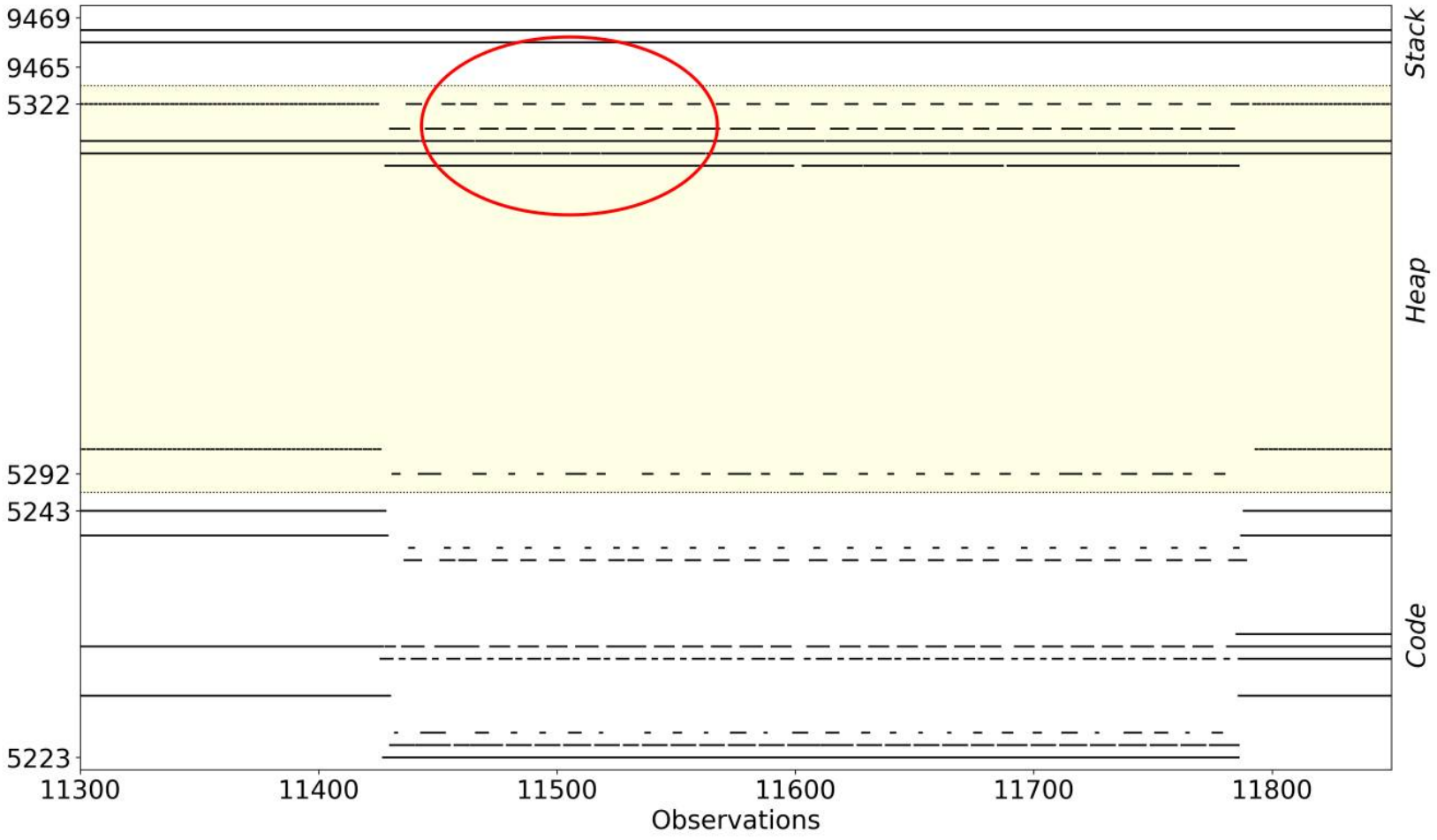
Leakage Reduction in Practice: Libjpeg Single-Stepping



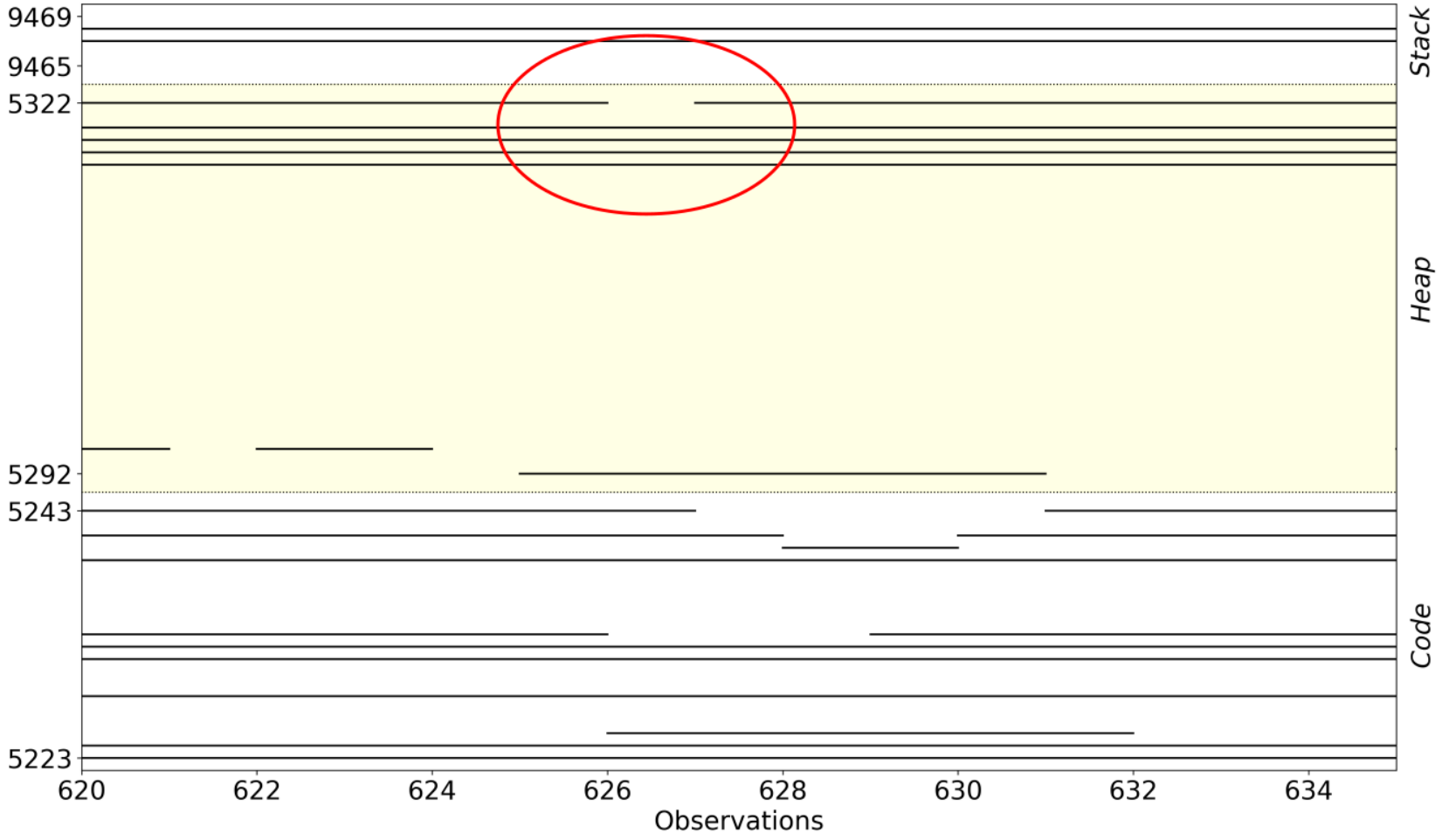
Leakage Reduction in Practice: Libjpeg Page Faults



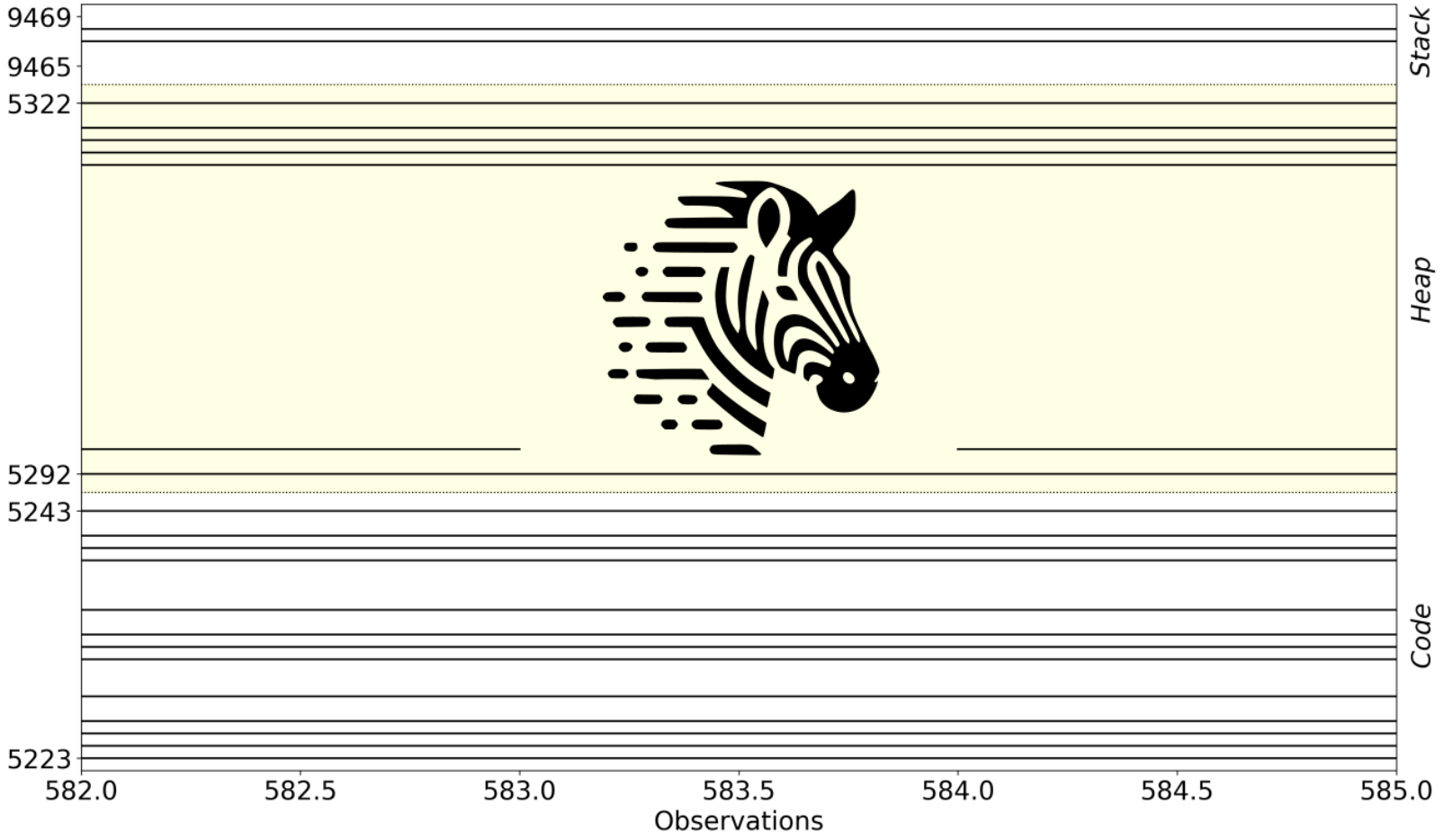
Leakage Reduction in Practice: Libjpeg TLBlur (N=10)



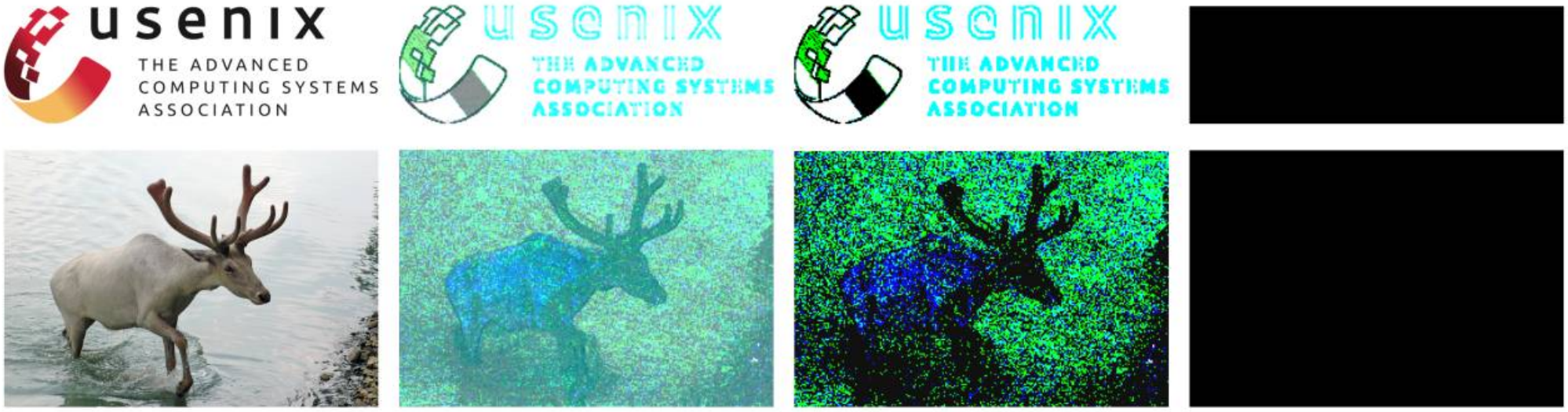
Leakage Reduction in Practice: Libjpeg TLBlur (N=20)



Leakage Reduction in Practice: Libjpeg TLBlur (N=30)



TLBlur: Compiler-Assisted Leakage Reduction in Practice



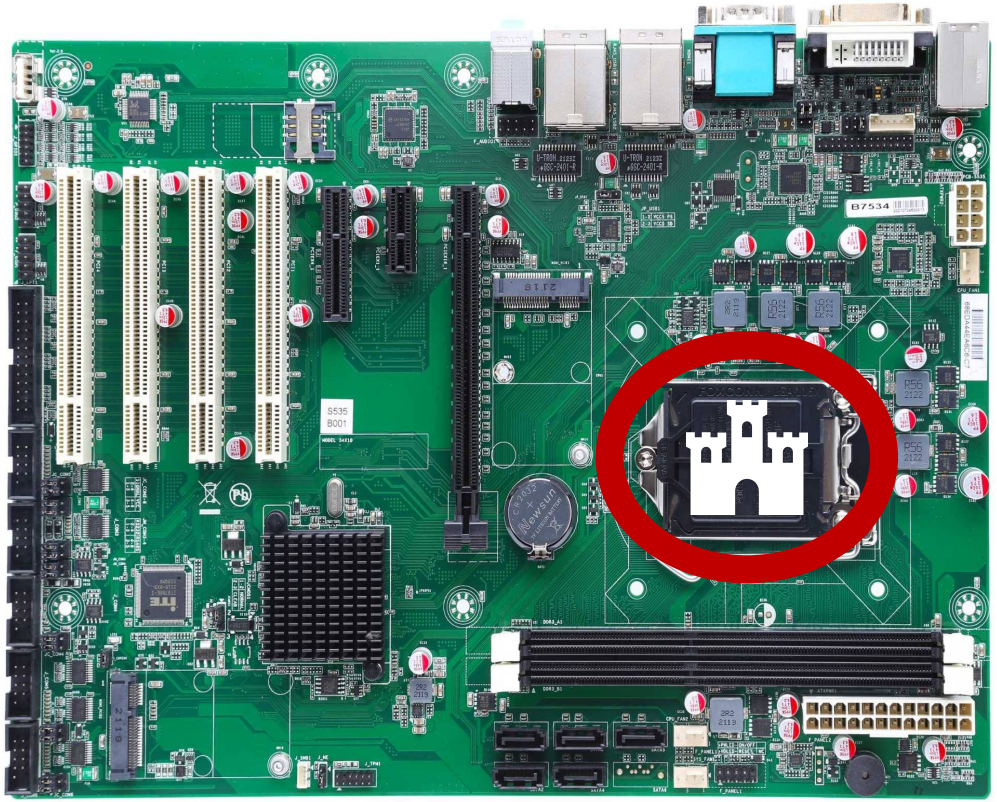
Automated “blurring” of page-access traces in space and time

Vanoverloop et. al “TLBlur: Compiler-Assisted Automated Hardening against Controlled Channels on Off-the-Shelf Intel SGX Platforms”, USENIX’25.

Idea #3 - Hardware-Based Attacks: Encrypted Memory Interface?

Confidential Computing: Trust Boundary

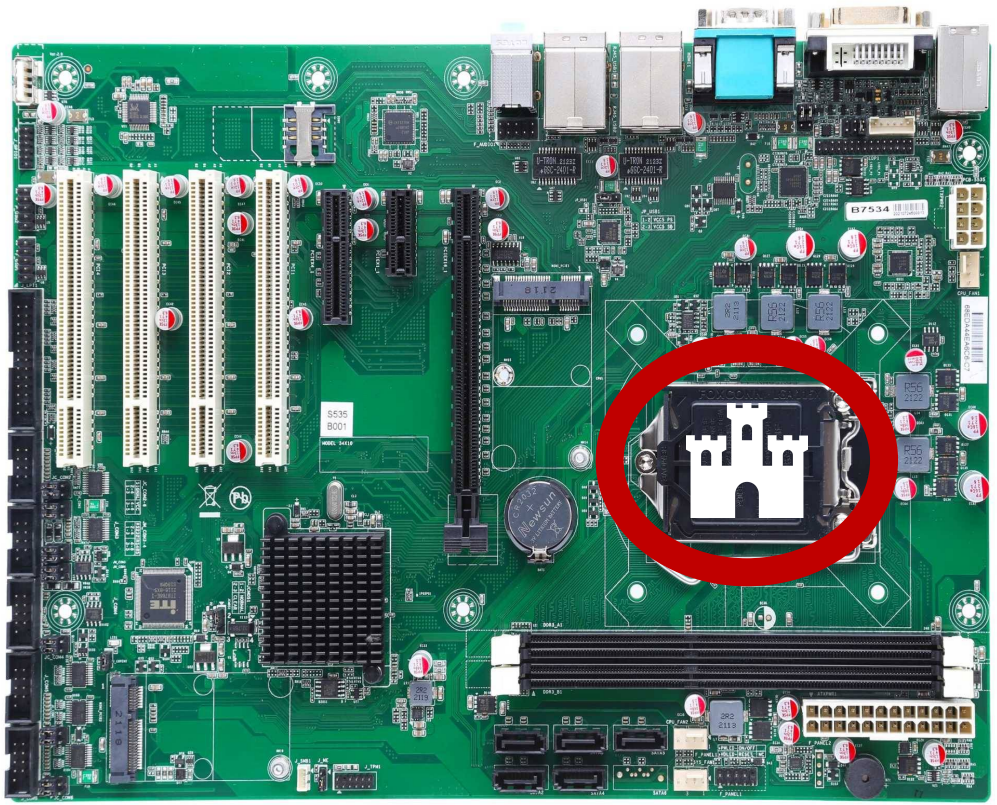
- **CPU package** = trust boundary



Confidential Computing: Trust Boundary

- **CPU package** = trust boundary
- **Memory encryption** to protect against <u>physical access:</u>

1. Rogue cloud provider employees
2. Supply-chain adversaries
3. Local law enforcement



A Brief History of Commercial Memory Encryption

intel

SGX

2015

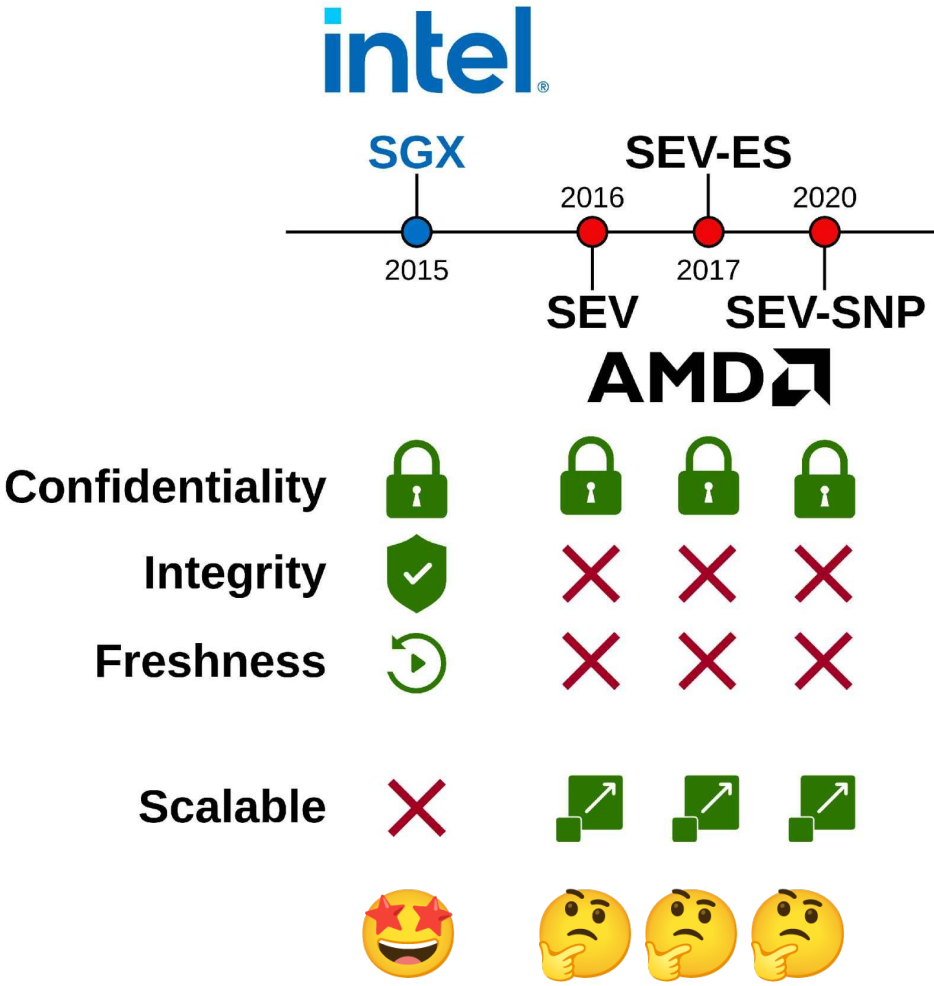
Confidentiality

Integrity

Freshness

Scalable

A Brief History of Commercial Memory Encryption



A Brief History of Commercial Memory Encryption

CLOUD | OPERATIONS & MANAGEMENT | NEWS

Why Google Cloud Turned to AMD to Solve for Runtime Encryption

AMD's latest server chips enabled better scalability, less lag, and more memory than Intel SGX, the cloud provider said.

Maria Korolov
July 21, 2020

5 Min Read

https://www.datacenterknowledge.com/cloud/why-google-cloud-turned-to-amd-to-solve-for-runtime-encryption

A Brief History of Commercial Memory Encryption

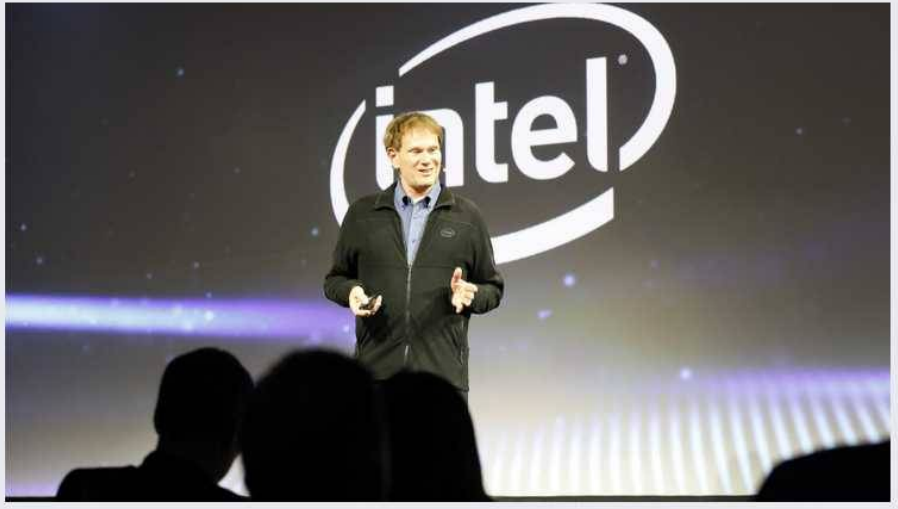
ars TECHNICA SECTIONS FORUM SUBSCRIBE SIGN IN

PUTTING ON A BRAVE FACE

Intel promises Full Memory Encryption in upcoming CPUs

Intel's security plans sound a lot like "we're going to catch up to AMD."

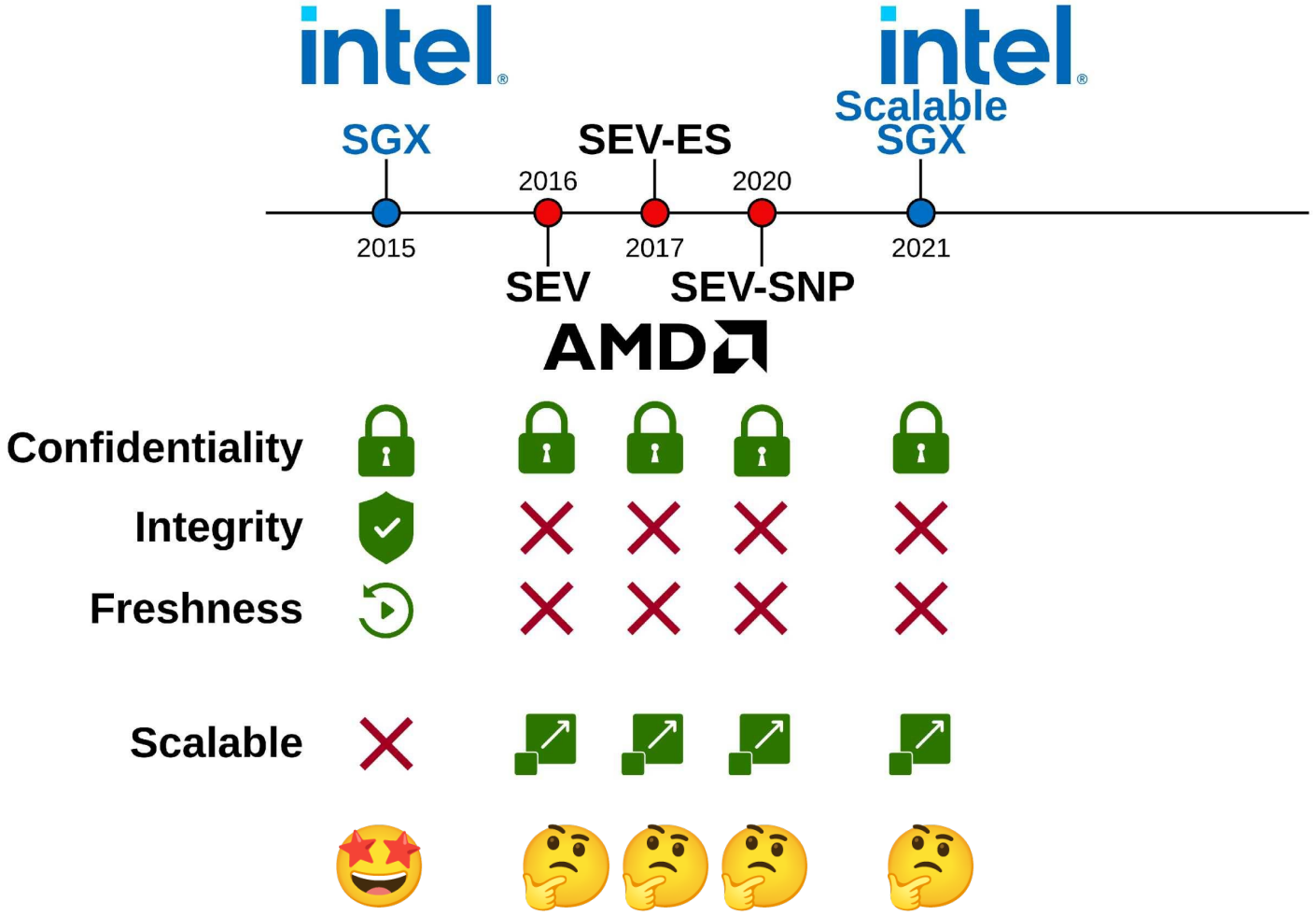
JIM SALTER – FEB 26, 2020 8:29 PM | 120



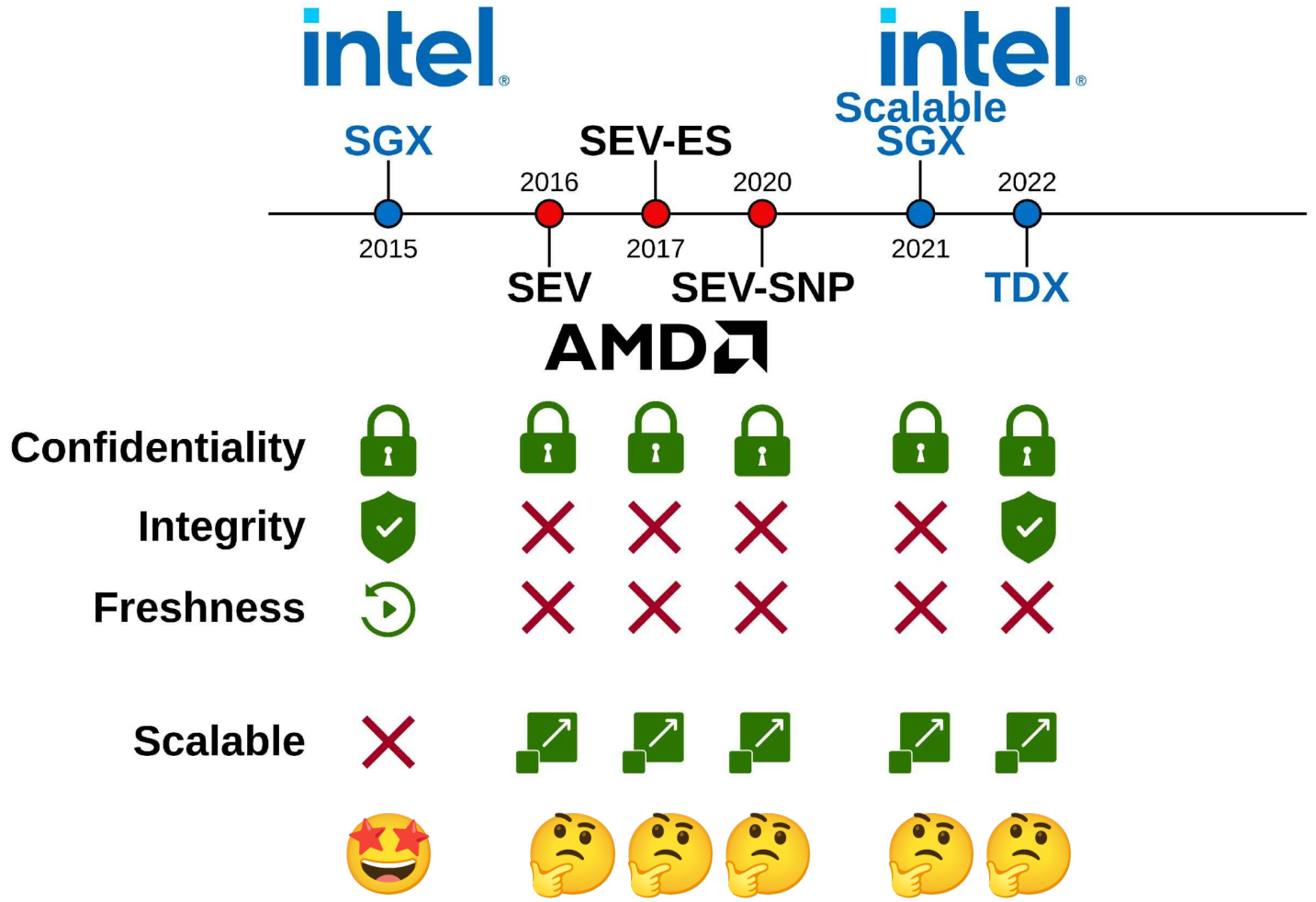
Intel Security Architecture and Technology Director John Sell provided an overview of Intel's mission to provide common security capabilities across all architectures. Credit: Intel Corporation

https://arstechnica.com/gadgets/2020/02/intel-promises-full-memory-encryption-in-upcoming-cpus/

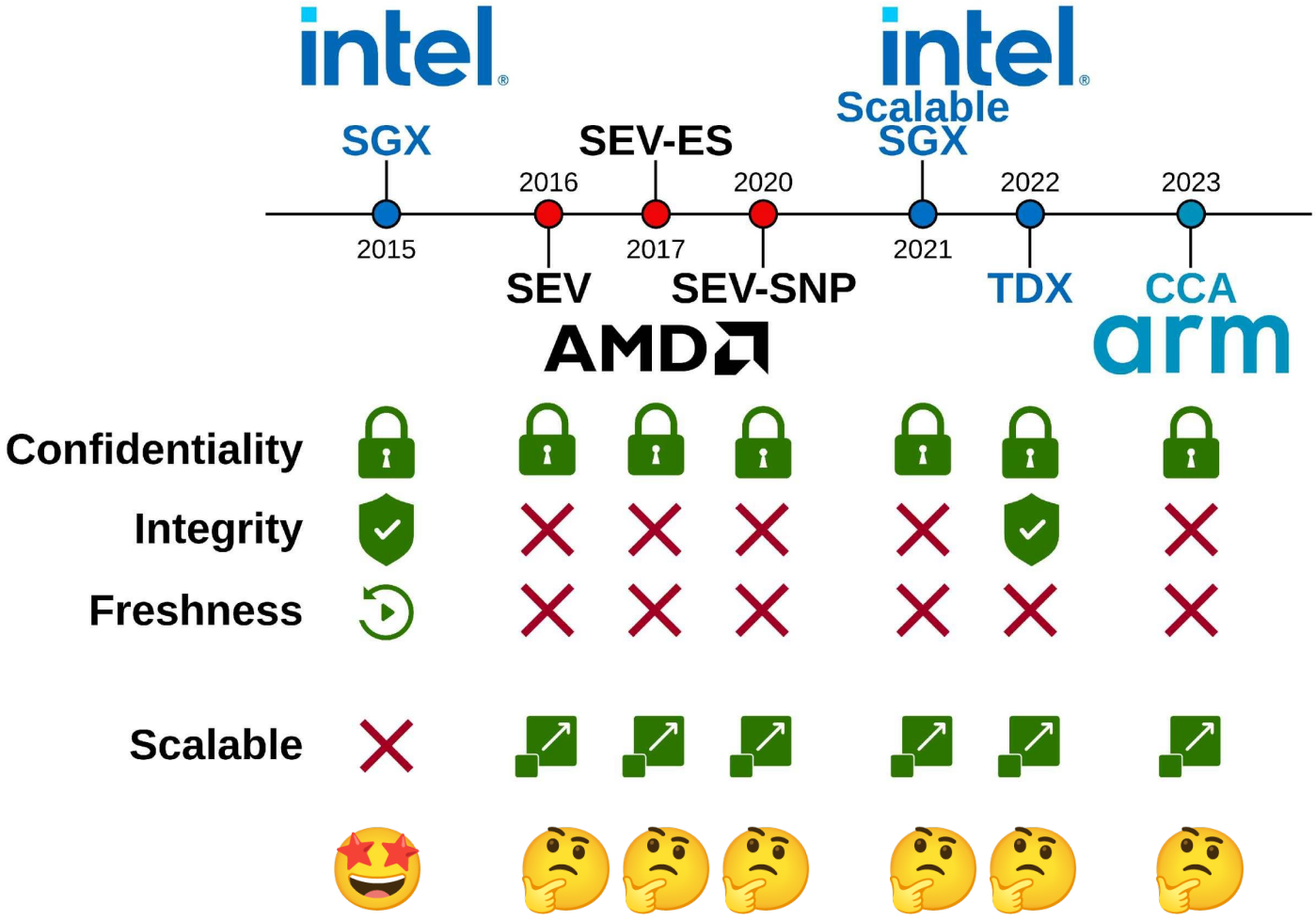
A Brief History of Commercial Memory Encryption



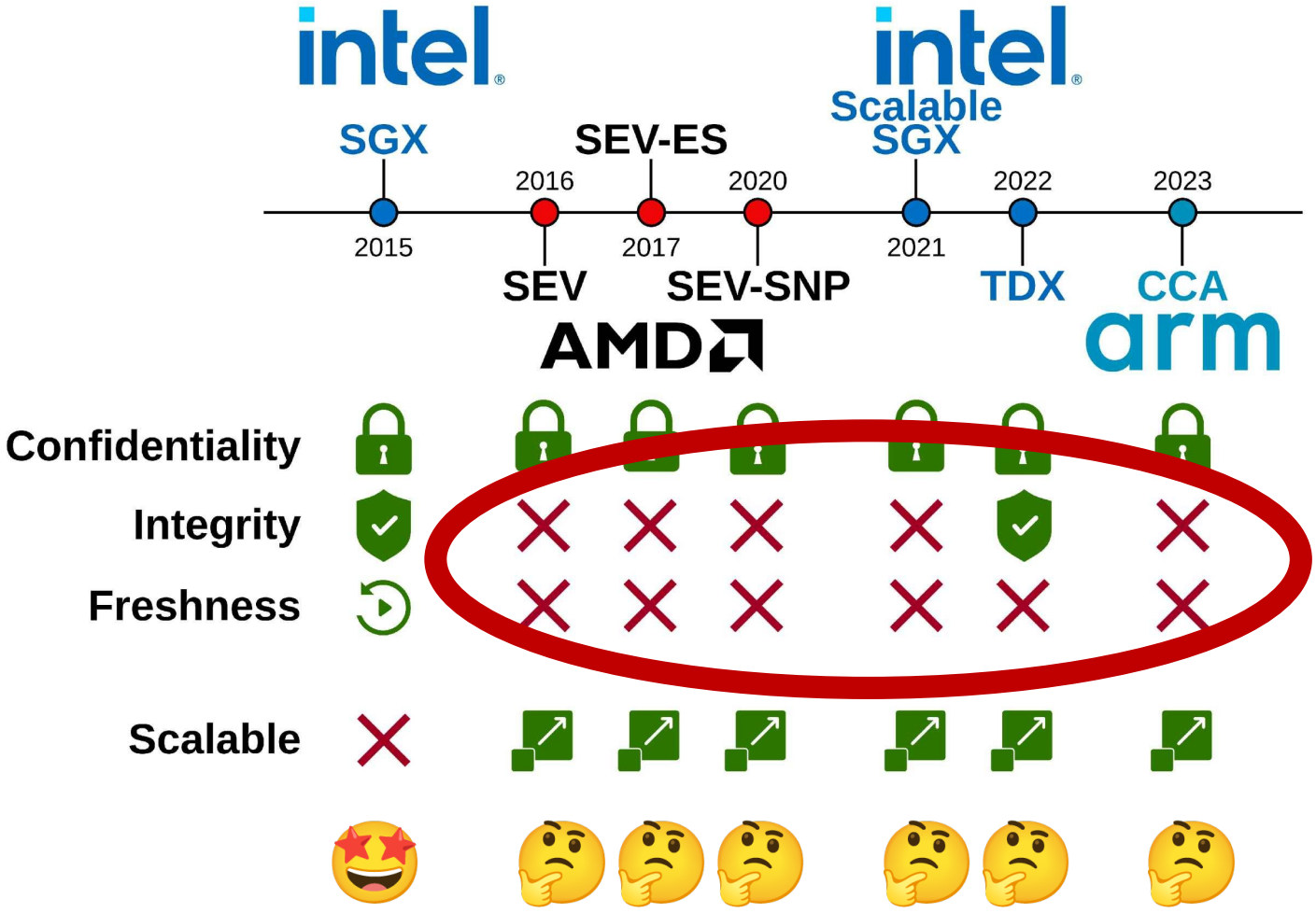
A Brief History of Commercial Memory Encryption

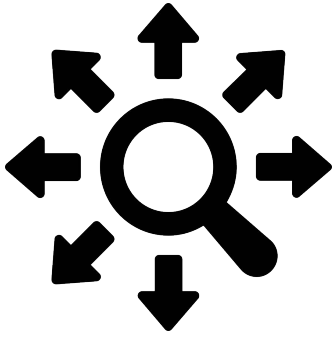


A Brief History of Commercial Memory Encryption

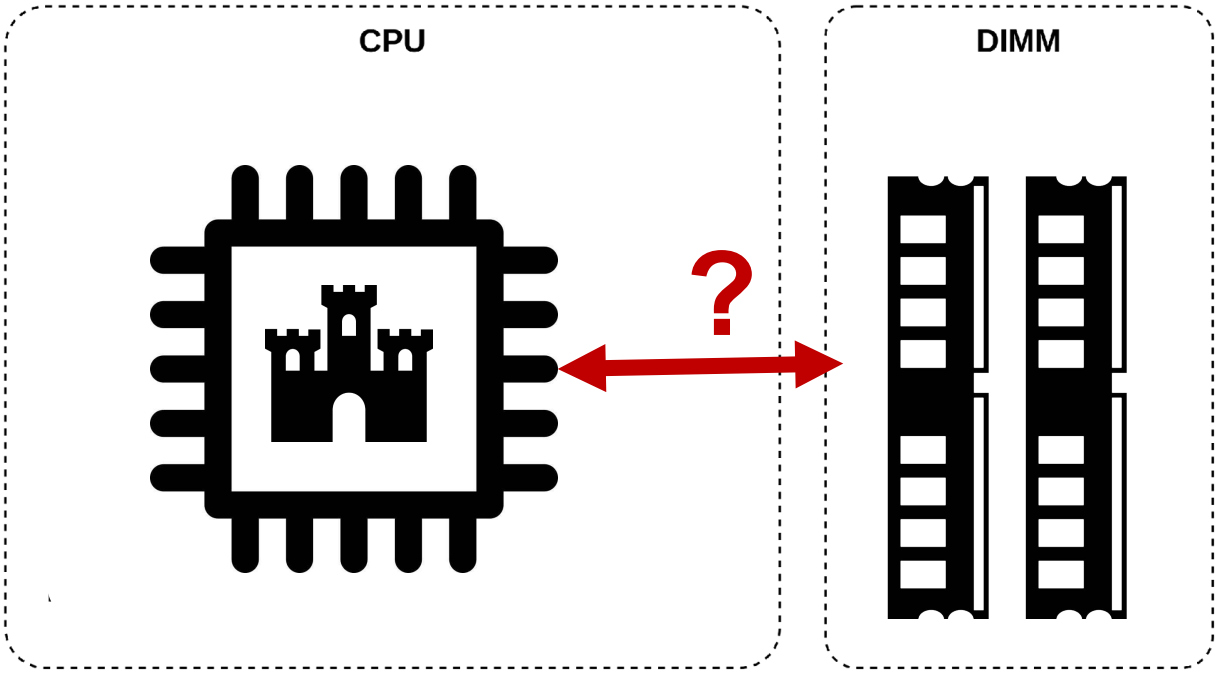


A Brief History of Commercial Memory Encryption

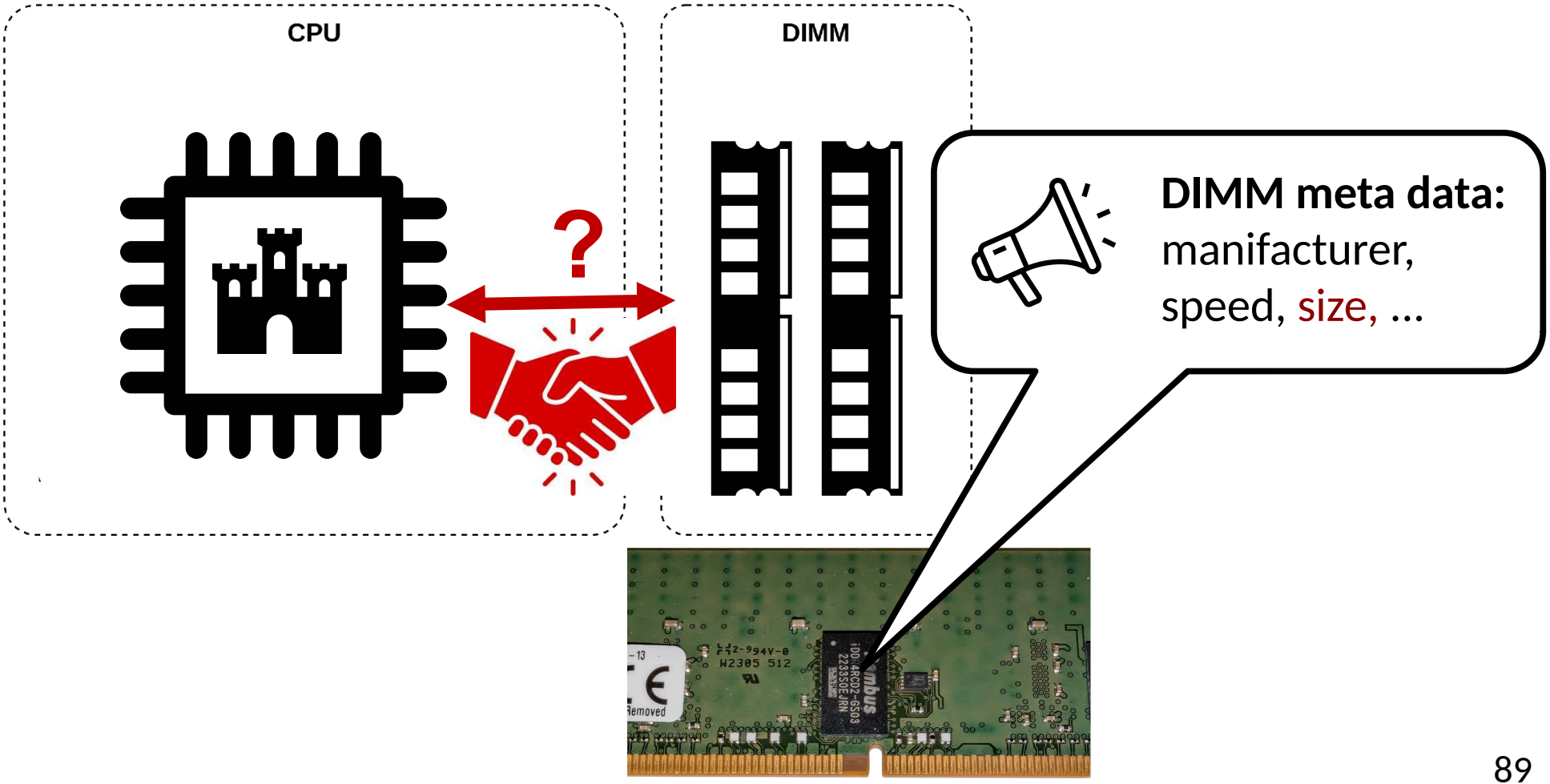




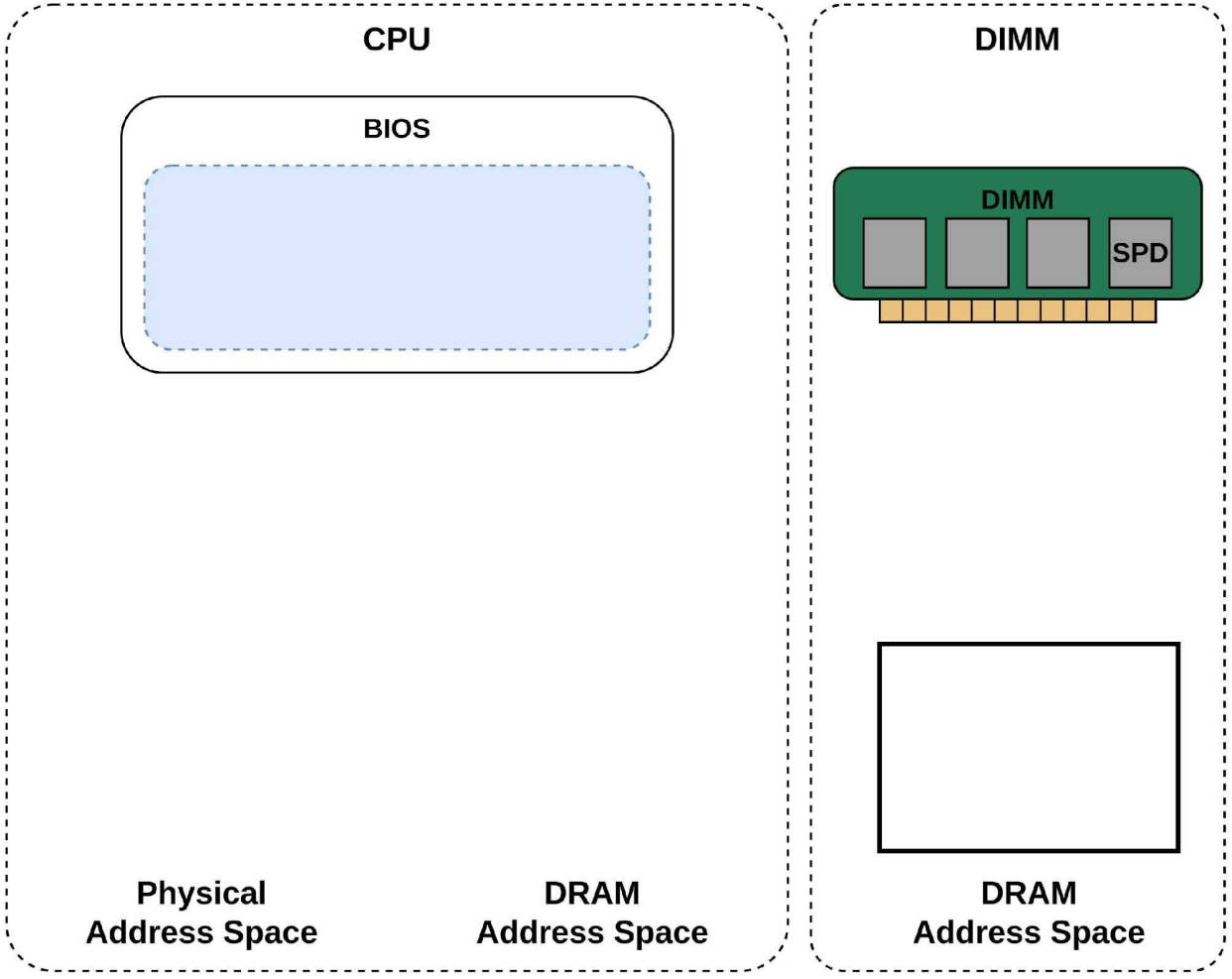
Background: Memory Initialization



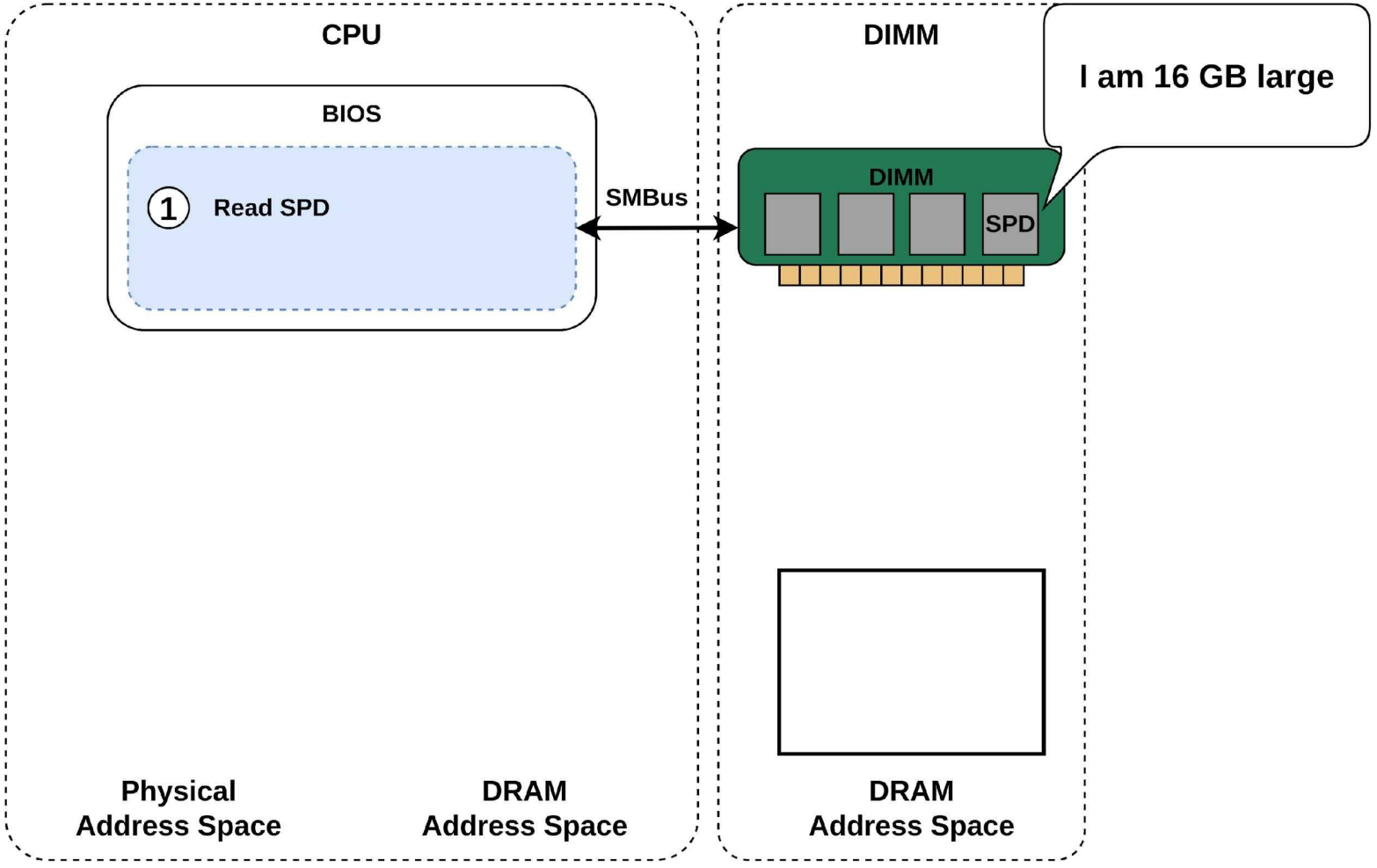
Background: Memory Initialization - Serial Presence Detect (SPD)



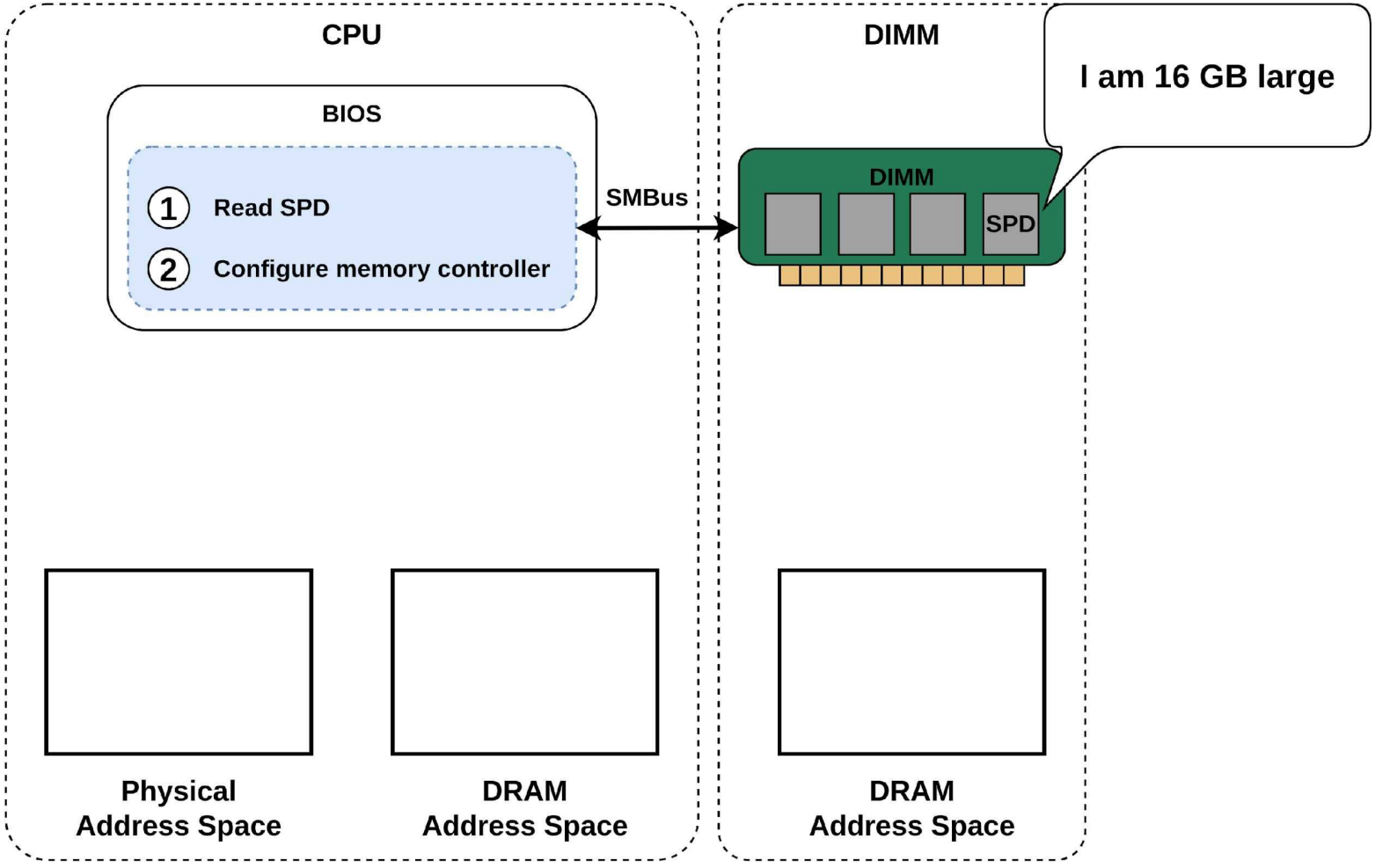
BadRAM: What if Your DRAM Lies to You?



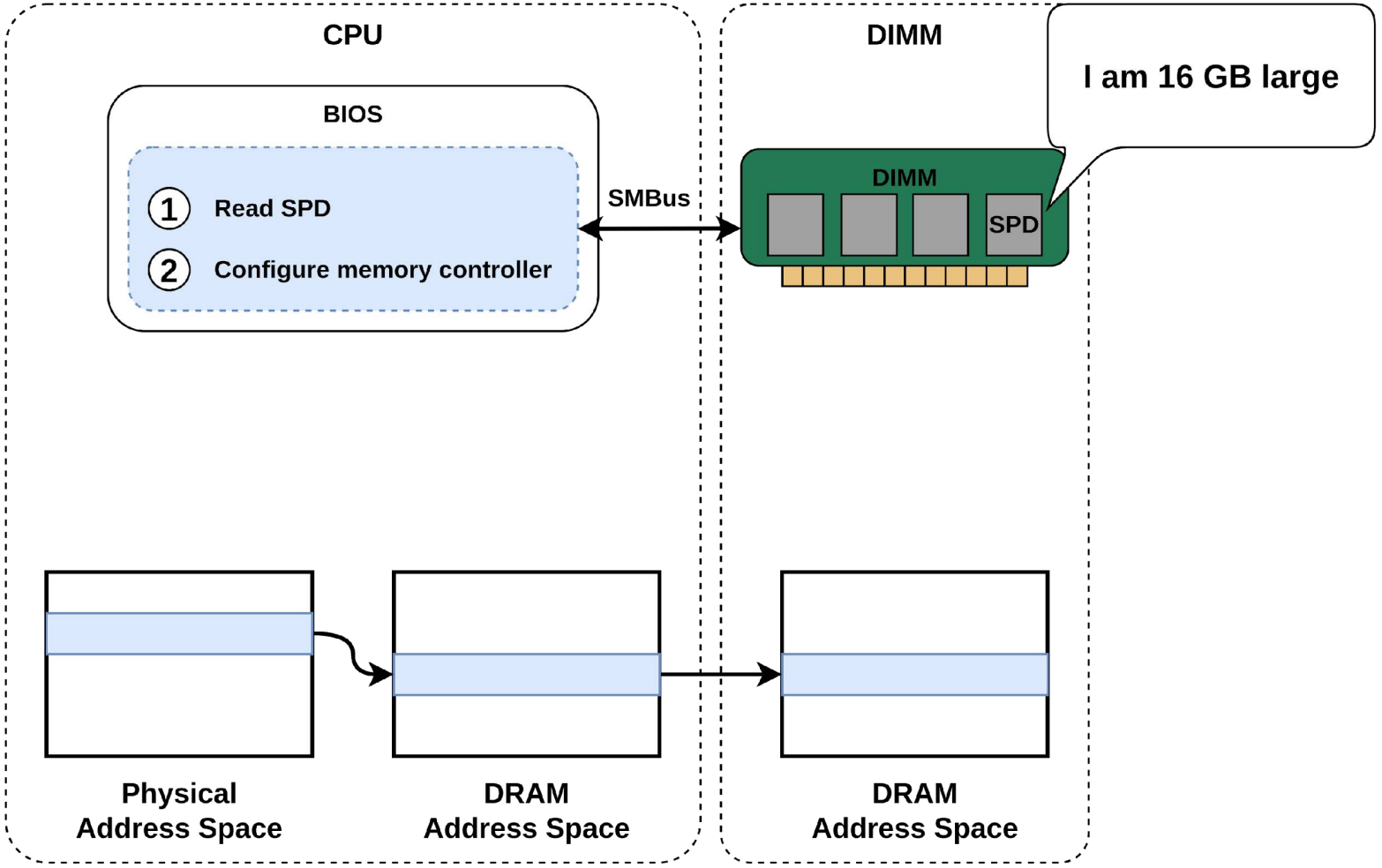
BadRAM: What if Your DRAM Lies to You?



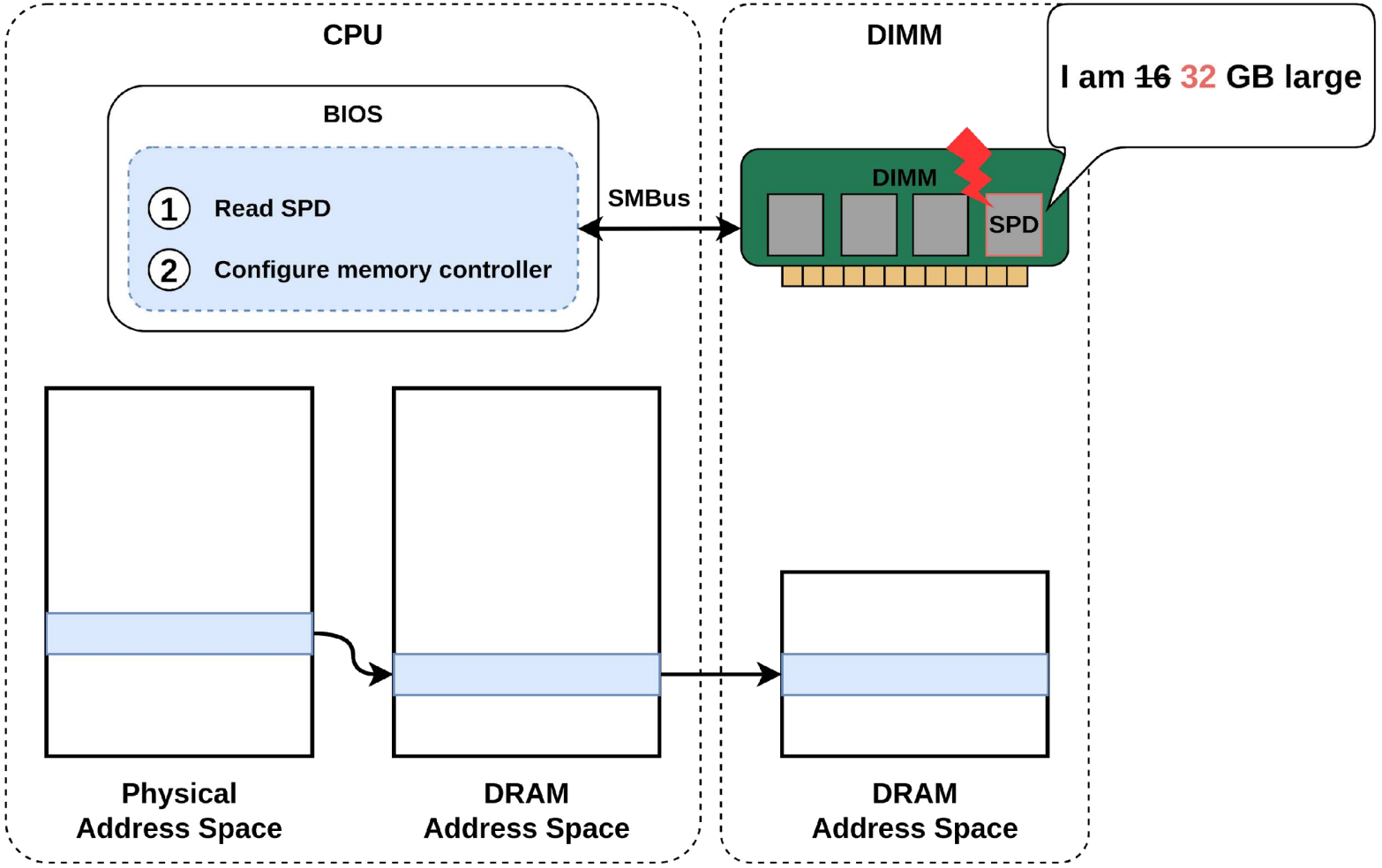
BadRAM: What if Your DRAM Lies to You?



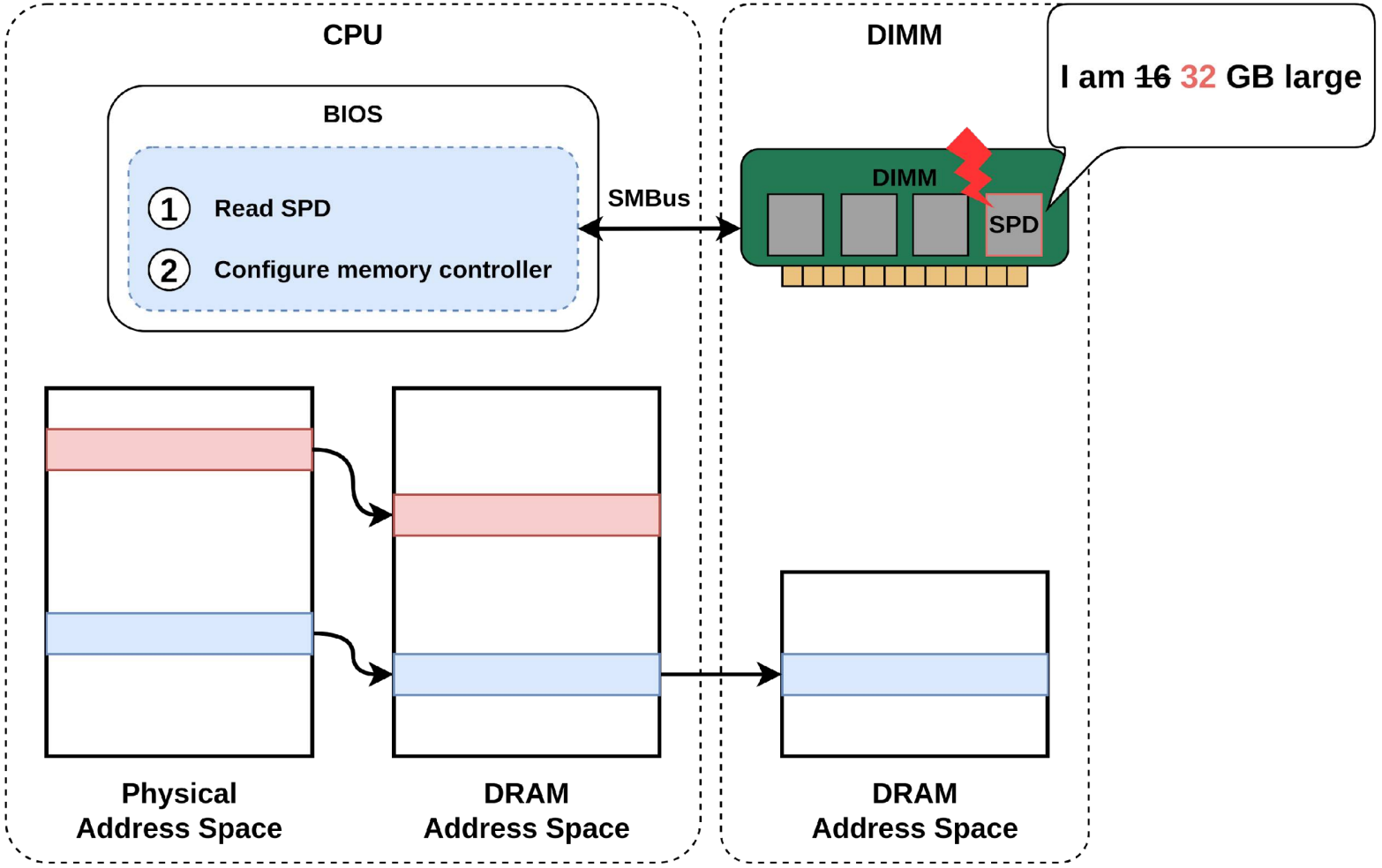
BadRAM: What if Your DRAM Lies to You?



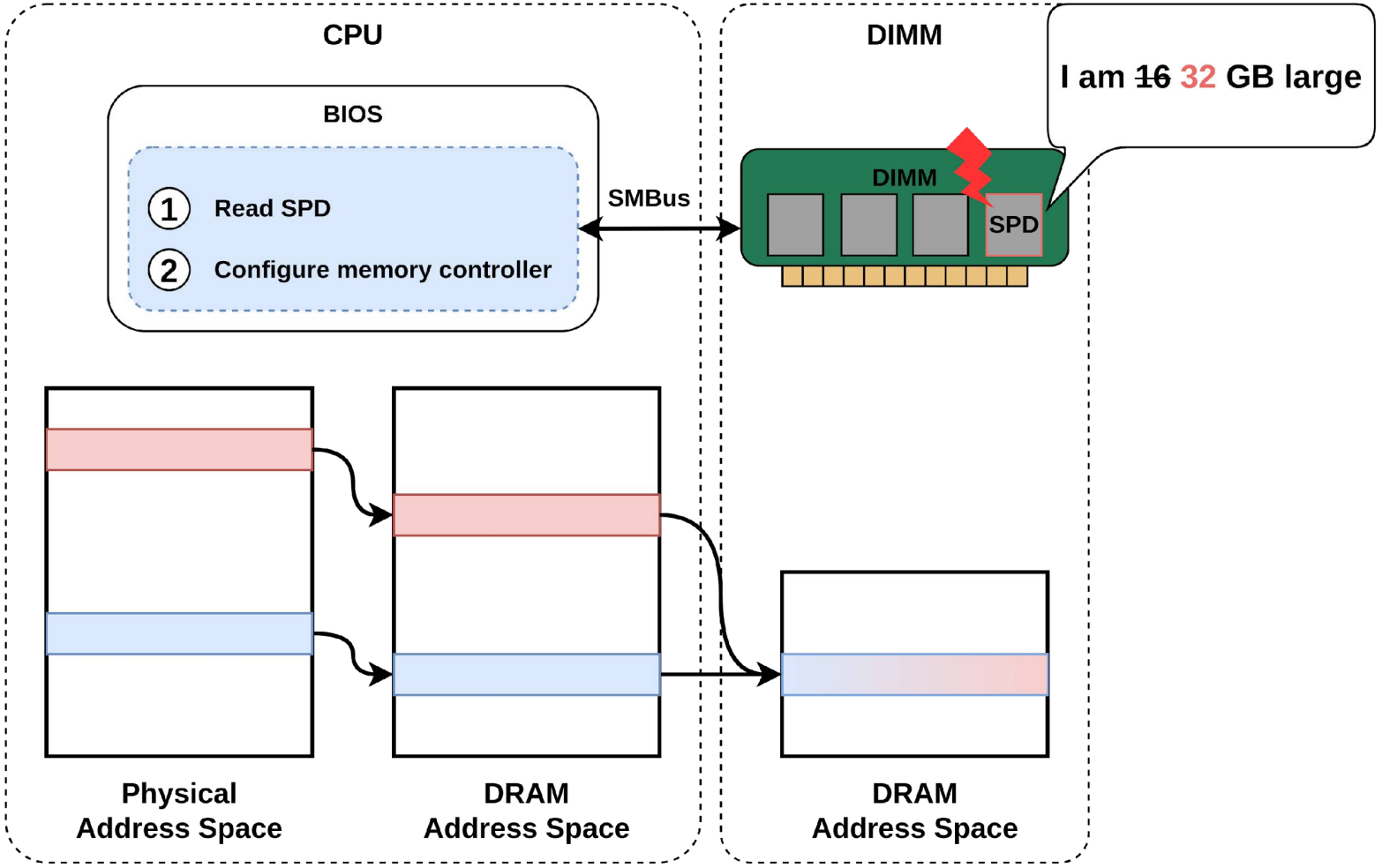
BadRAM: What if Your DRAM Lies to You?



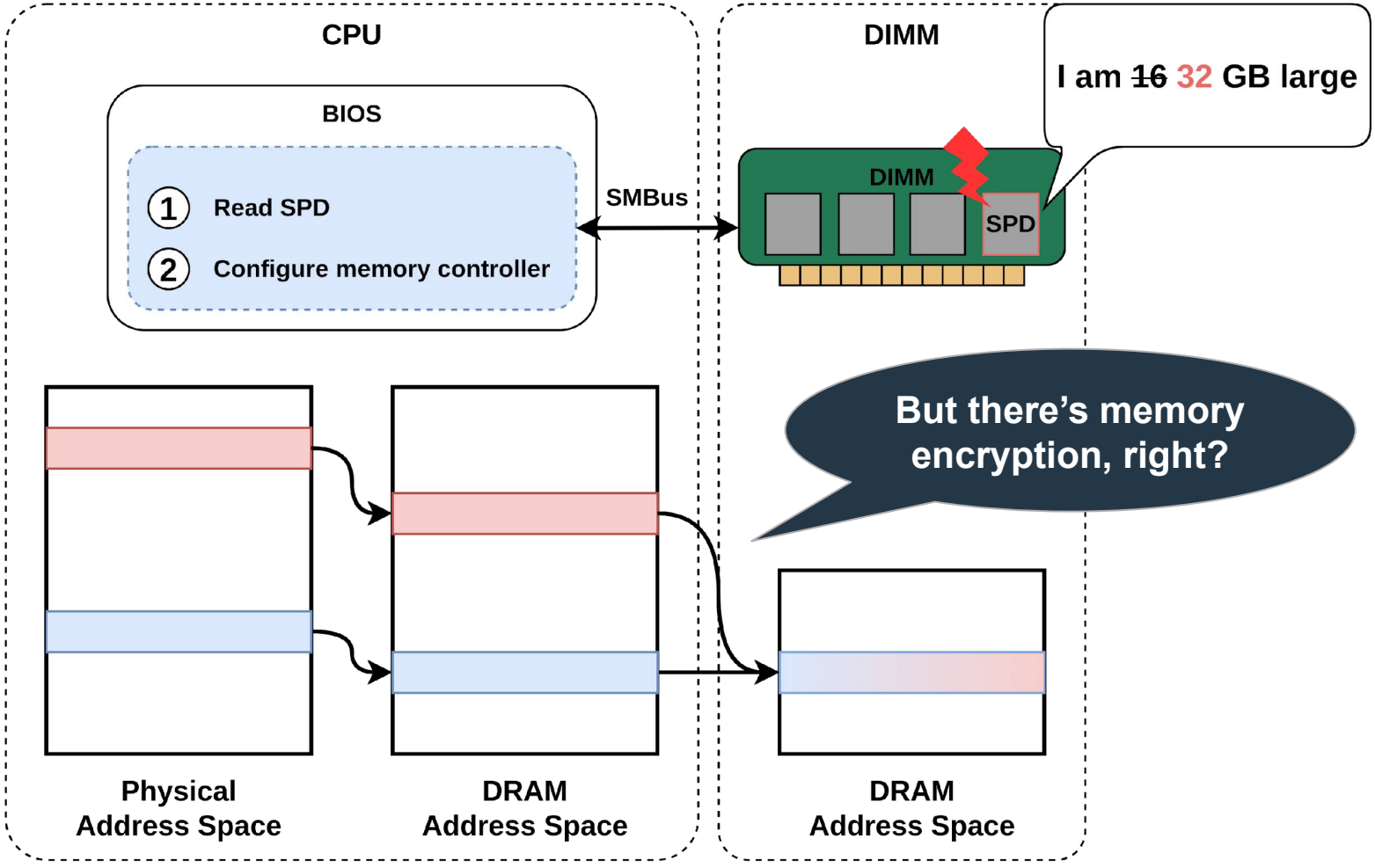
BadRAM: What if Your DRAM Lies to You?



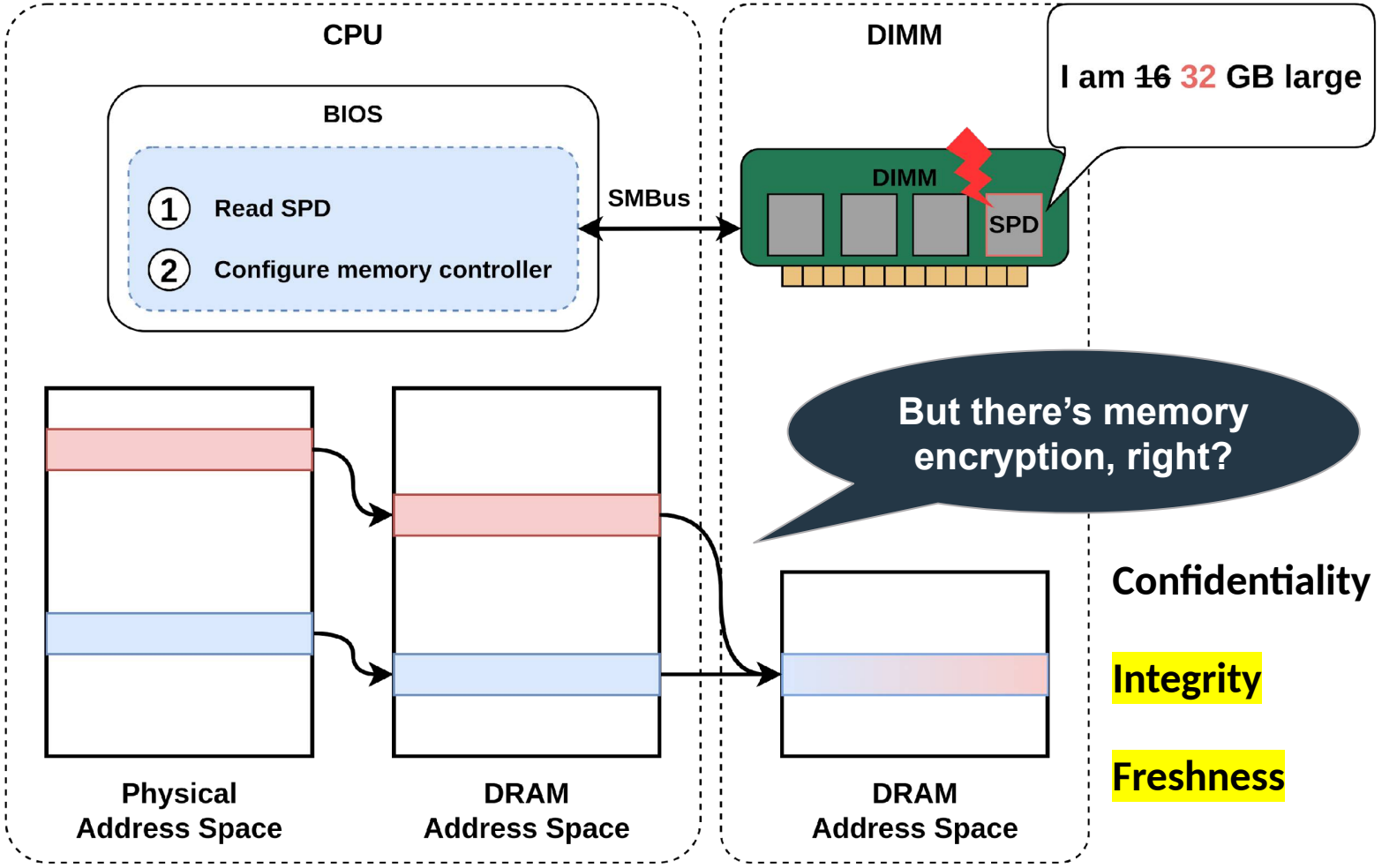
BadRAM: What if Your DRAM Lies to You?



BadRAM: What if Your DRAM Lies to You?



BadRAM: What if Your DRAM Lies to You?



Demo

Replaying encrypted memory

```
*****     AMD SEV-SNP Victim VM     *****

Initialized 64-byte memory buffer
 --> Virtual address:  0x7f485c6e7000
 --> Physical address: 0x26f5e000

Buffer initialized to zero:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Capture the ciphertext, then press enter

Setting buffer to 0xff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Replay the old ciphertext, then press enter

Printing memory buffer
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
jesse@jesse-test:~$
```

```
*****     Privileged Software Attacker     *****

Translating victim address
 --> Guest physical address: 0x26f5e000
 --> Host physical address:  0x18415e000

Calculating memory alias
 --> Original address: 0x18415e000
 --> Alias address:    0x58415e000

Press Enter to capture ciphertext

Successfully captured victim ciphertext:
0d 52 96 9c b1 b3 0a e9 b5 f0 7e d9 06 e7 2f 5c fa 17 cf aa 88 c8 4e 8b c5 26 f7 28 5e 57 1f c4
e8 b5 09 44 00 a3 af 6f b9 12 61 aa 27 b7 e4 df 65 6f 39 ec a7 ee c5 50 f3 c9 56 51 dc 84 ee b3


Press enter to start replay

Replayed victim ciphertext
jesse@horus:~/badram/sev-attacks/simple-replay$
```

The Register

AMD secure VM tech undone by DRAM meddling

Boffins devise BadRAM attack to pilfer secrets from SEV-SNP e

Thomas Claburn

L'Echo

Des chercheurs de Louvain ont piraté la sécurité des puces AMD

BadRAM attack breaches AMD secure VMs using a Raspberry Pi Pico, DDR socket, and a 9V battery

News By Mark Tyson published December 11, 2024

has now issued firmware updates for cloud providers.

KWETSBAARHEDEN

KU Leuven legt kwetsbaarheden in AMD-processoren bloot

ars TECHNICA AI BIZ & IT CARS RIBE

BEWARE OF GHOSTS

AMD's trusted execution environment blown wide open by new BadRAM attack

Attack bypasses AMD protection promising security, even when a server is compromised.

DAN GOODIN – 10 DEC 2024 18:08 | 112

Credit: Getty Images

ITdaily.

KU Leuven legt kwetsbaarheid in AMD-processoren bloot

mory Modules

https://badram.eu/

DE TIJD

Leuvense onderzoekers kraken beveiliging AMD-chips

Vendor Response: Boot-Time Firmware Mitigations

Undermining Integrity Features of SEV-SNP with Memory Aliasing

AMD ID: AMD-SB-3015
Potential Impact: Loss of Integrity
Severity: Medium

Summary

A team of researchers has reported to AMD that it may be possible to modify serial presence detect (SPD) metadata to make an attached memory module appear larger than it is, potentially allowing an attacker to overwrite physical memory.

Guest Attestation Report [Attestation method for Guest VM]

ATTESTATION_REPORT Structure PLATFORM_INFO field in Byte offset 0h bit 5 contains indication that the mitigation has been applied and confirmed.

Byte Offset	Bits	Name	Description
00h	63:6	-	Reserved.
	5	ALIAS_CHECK_COMPLETE	Indicates that alias detection has completed since the last system reset and there are no aliasing addresses. Resets to 0.

https://www.amd.com/en/resources/product-security/bulletin/amd-sb-3015.html

Vendor Response: Boot-Time Firmware Mitigations

Undermining Integrity Features of SEV-SNP with Memory Aliasing

AMD ID: AMD-SB-3015
Potential Impact: Loss of Integrity
Severity: Medium

Summary

A team of researchers has reported to AMD that it may be possible to modify serial presence detect (SPD) metadata to make an attached memory module appear larger than it is, potentially allowing an attacker to overwrite physical memory.

What if we can introduce aliases at runtime (post-boot)?

Guest Attestation Report [Attestation method for Guest VM]

ATTESTATION_REPORT Structure PLATFORM_INFO[illegible] offset 0h bit 5 contains indication that the mitigation has been applied and confirmed.

Byte Offset	Bits	Name	
00h	63:6	-	Reser[illegible]
	5	ALIAS_CHECK_COMPLETE	Indicates that alias detection has completed since the last system reset and there are no aliasing addresses. Resets to 0.

https://www.amd.com/en/resources/product-security/bulletin/amd-sb-3015.html


I HAVE...
AN EVIL PLAN
quickmeme.com

Interfering at Runtime: Commercial DRAM Interposers?






Genuine New MW-Keysight U4972A DDR4 Protocol Debugging and Analysis Solution Logic Analyzers Factory Wholesale Price

US$782,016.00
1 Set (MOQ)

Send Inquiry | Chat Now

Product Details

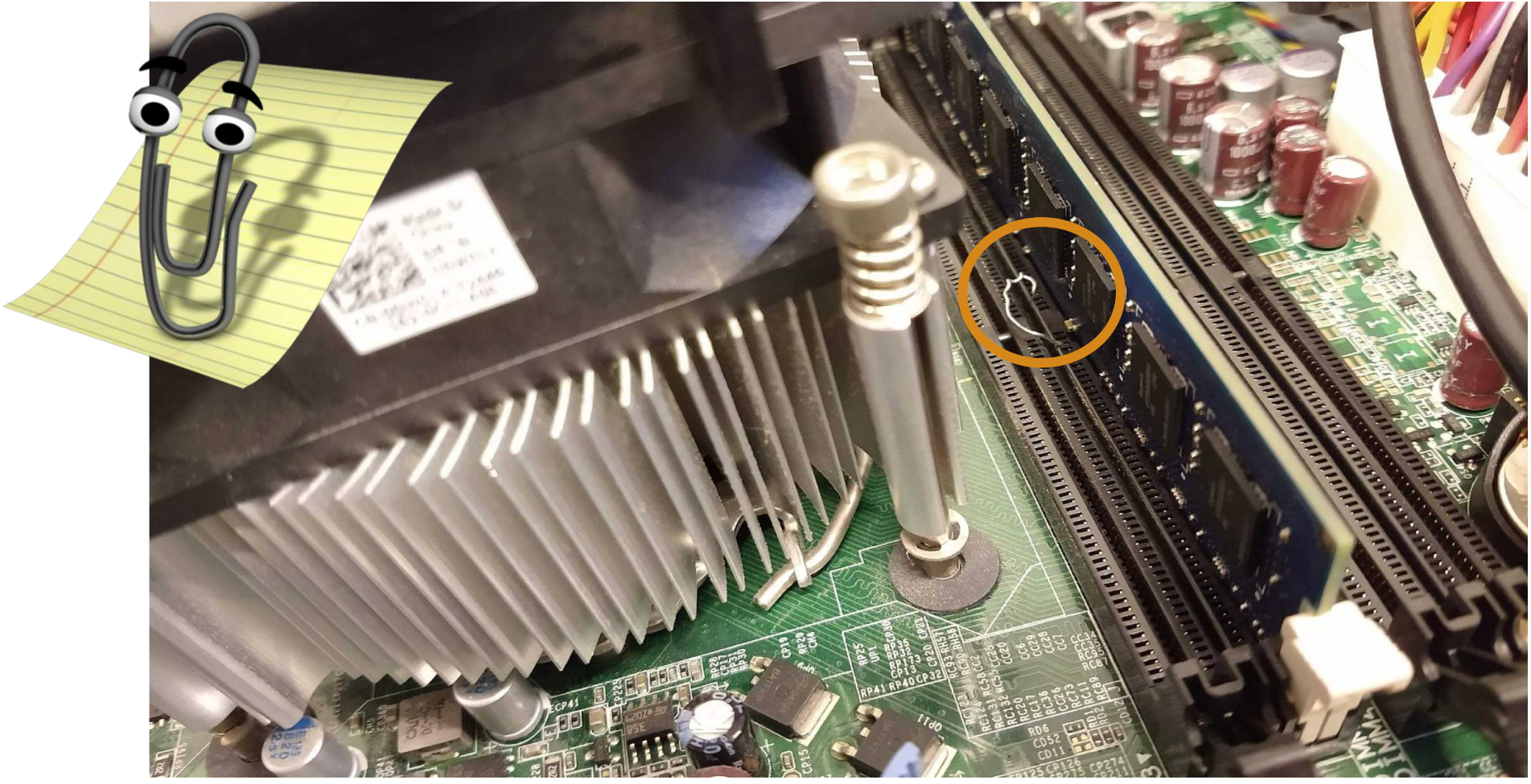
Customization:	Available
After-sales Service:	12 Months
Warranty:	12 Months

Shenzhen Leading International Trading Co., Ltd. >

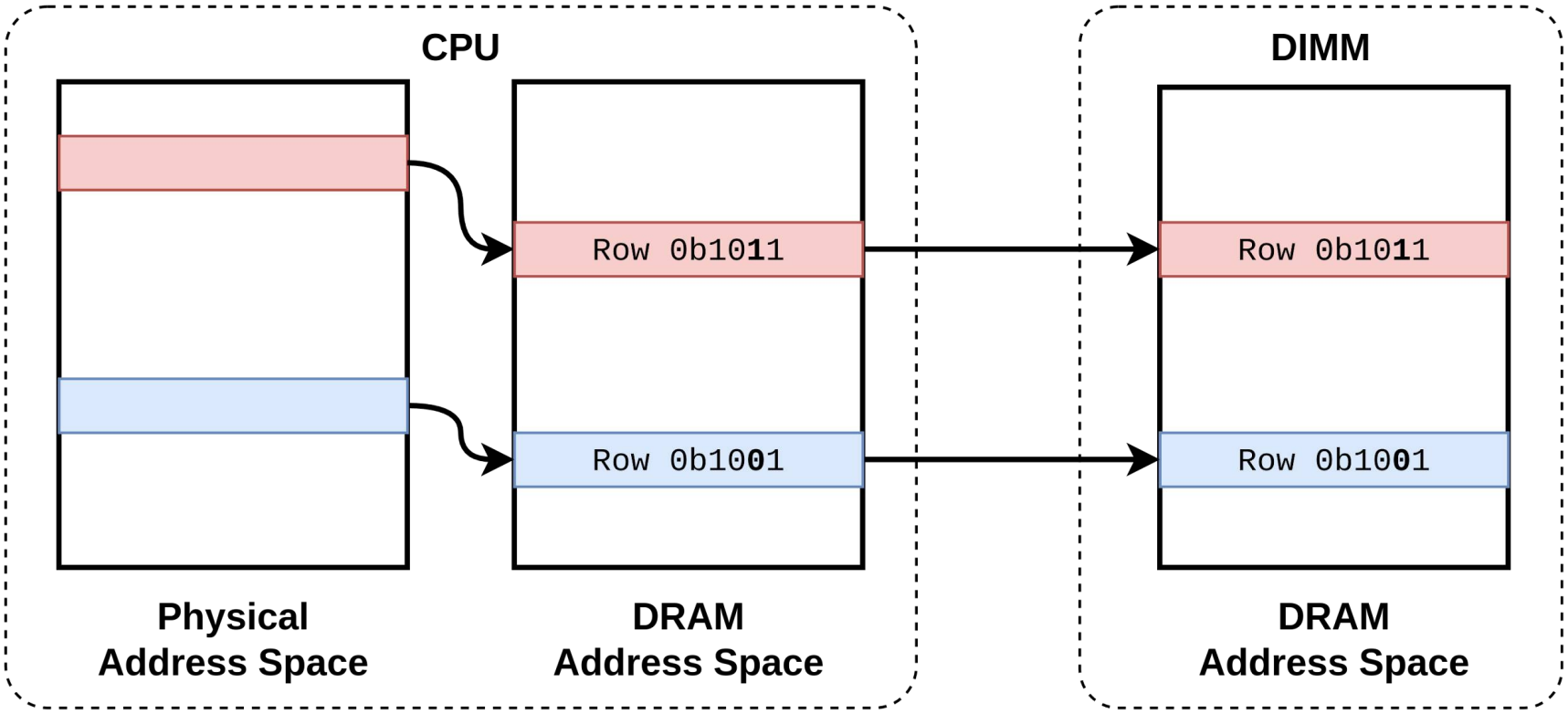
Gold Member Since 2024

Audited Supplier

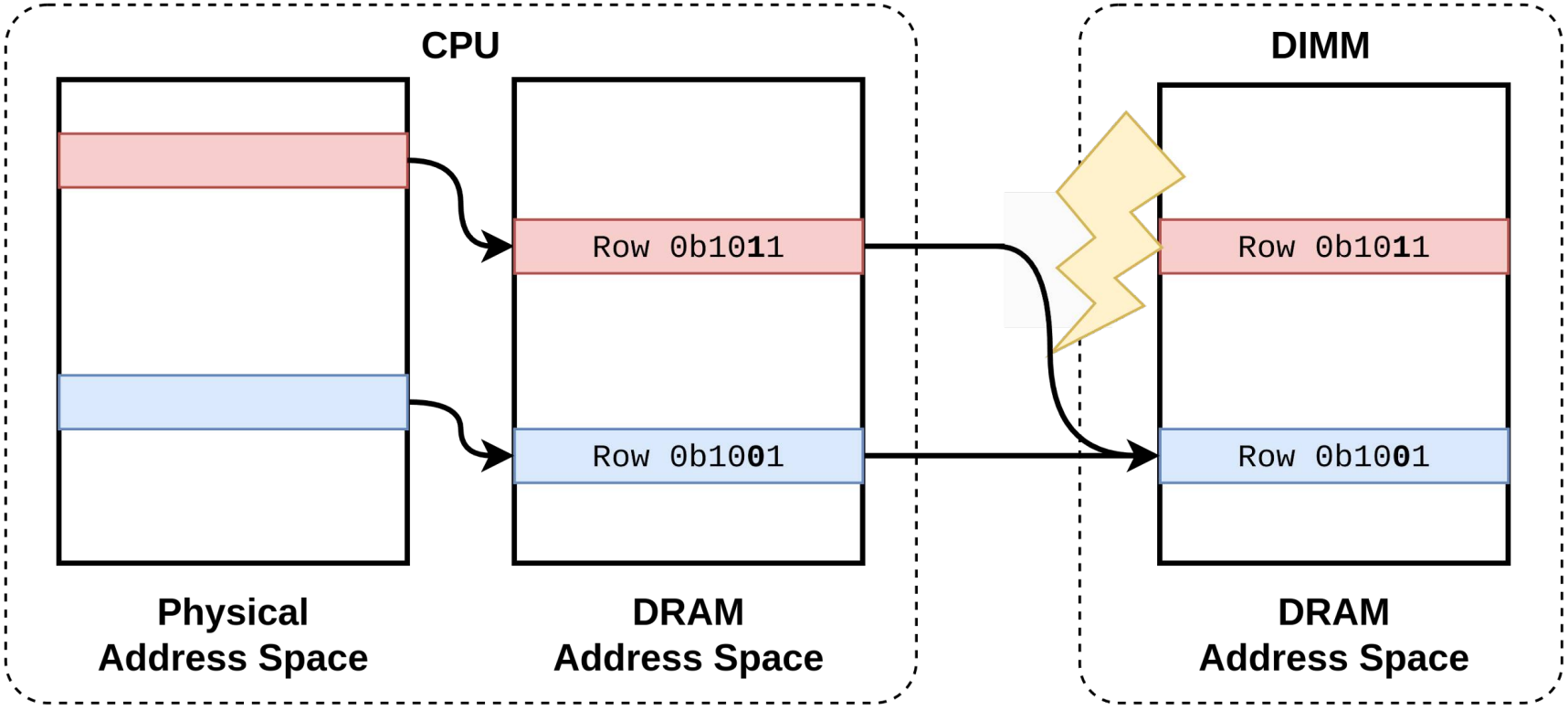
Tampering with Addressing at Runtime



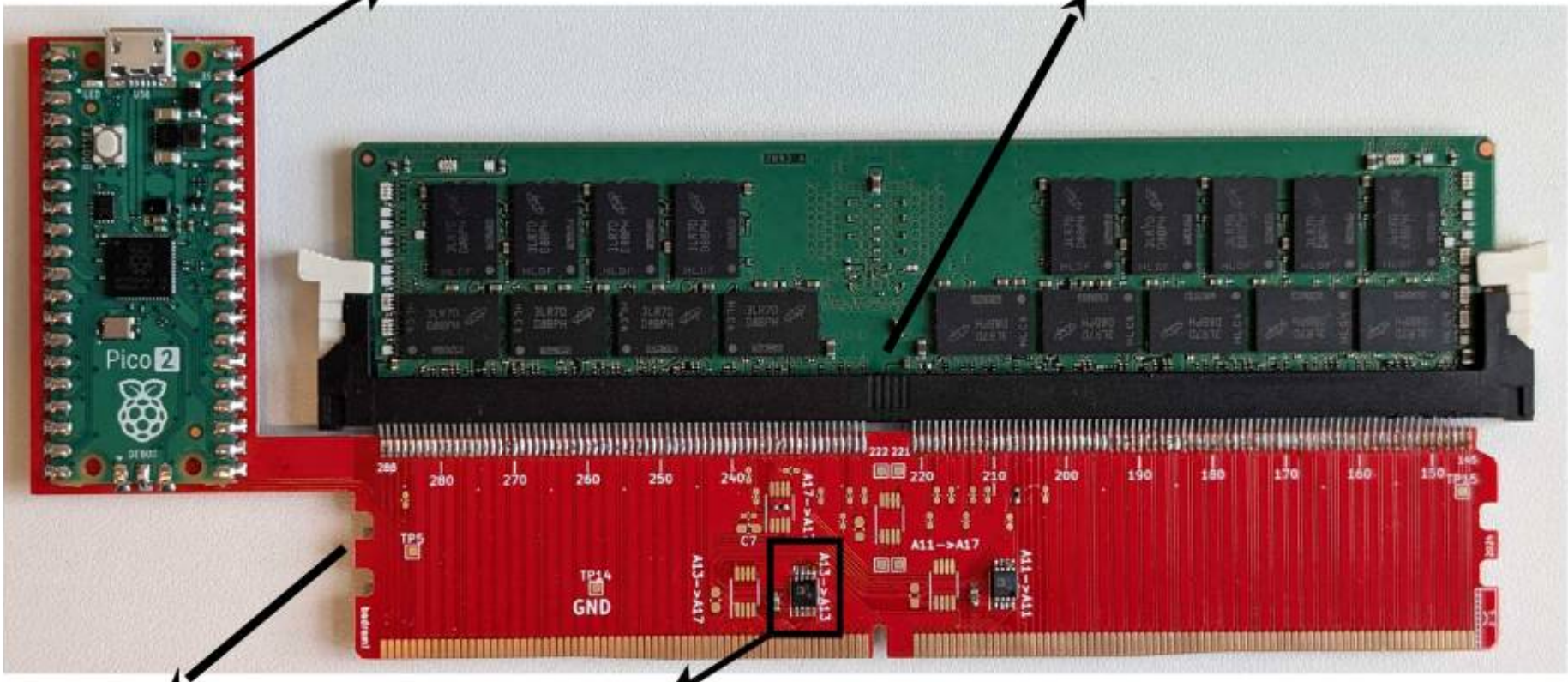
Battering RAM: Tampering with Addressing at Runtime



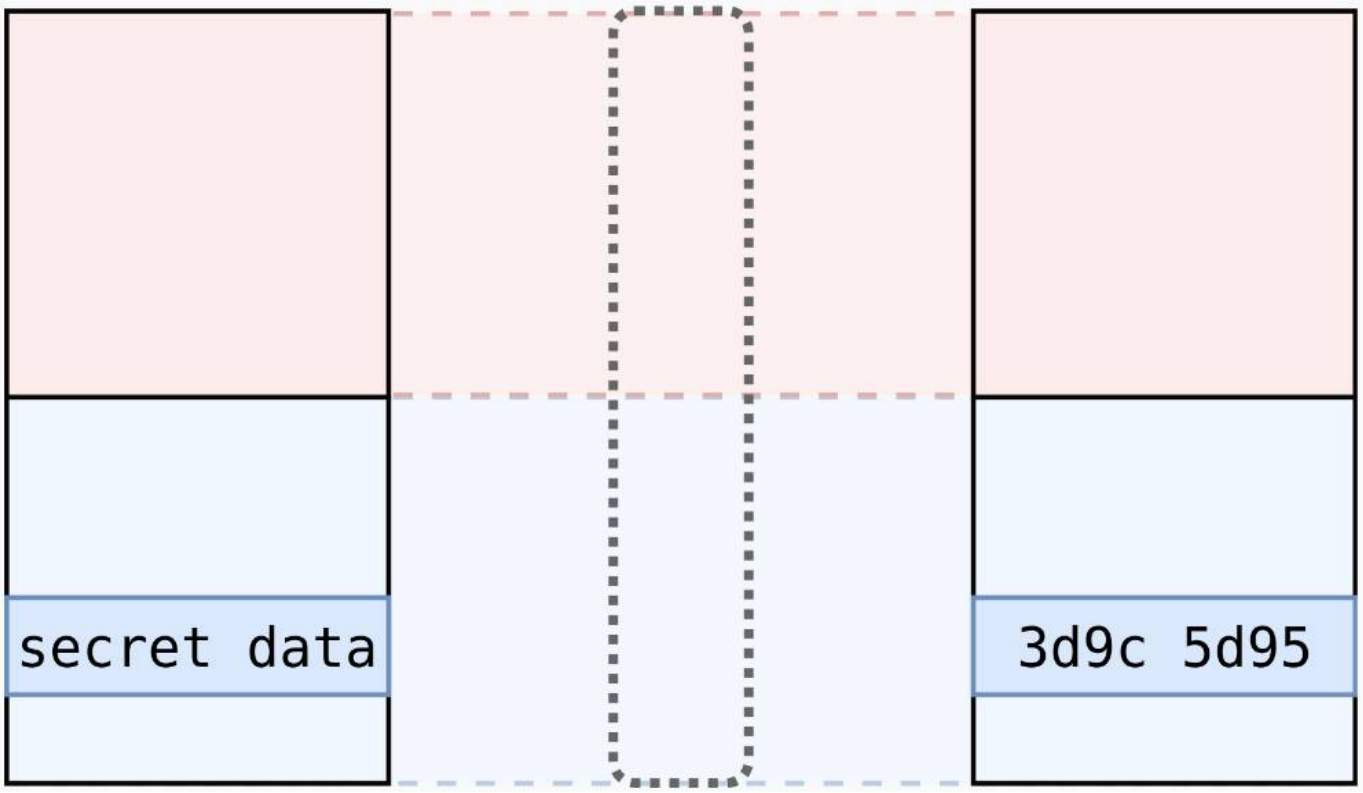
Battering RAM: Tampering with Addressing at Runtime



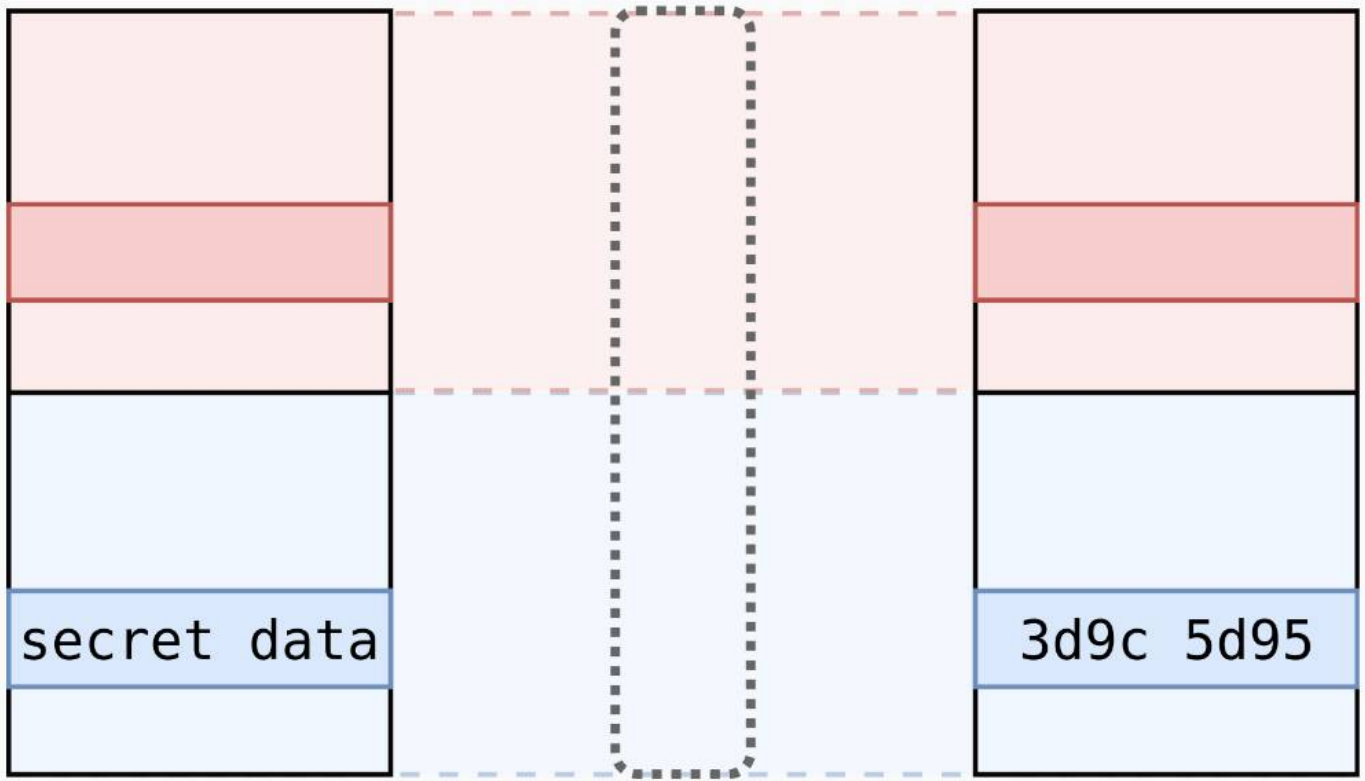
Battering RAM: Tampering with Addressing at Runtime



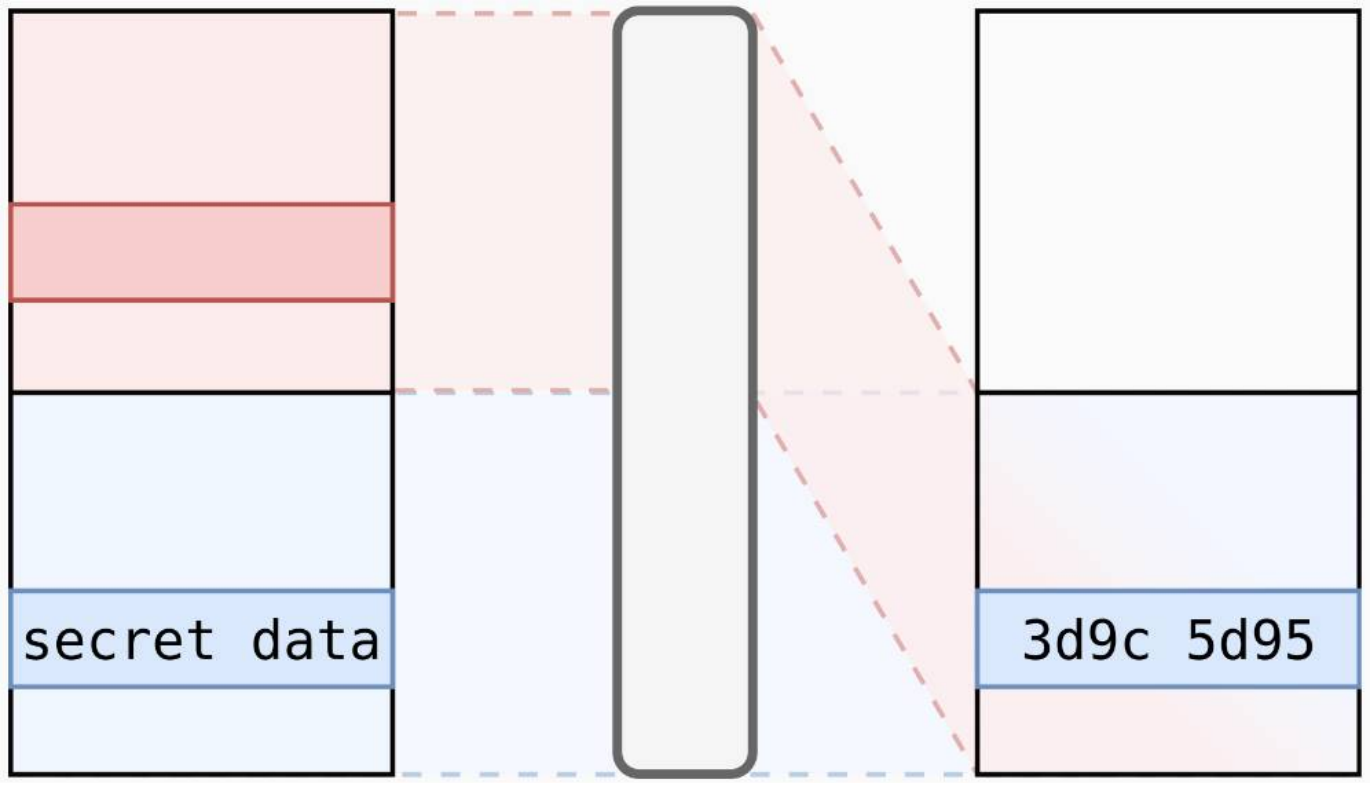
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



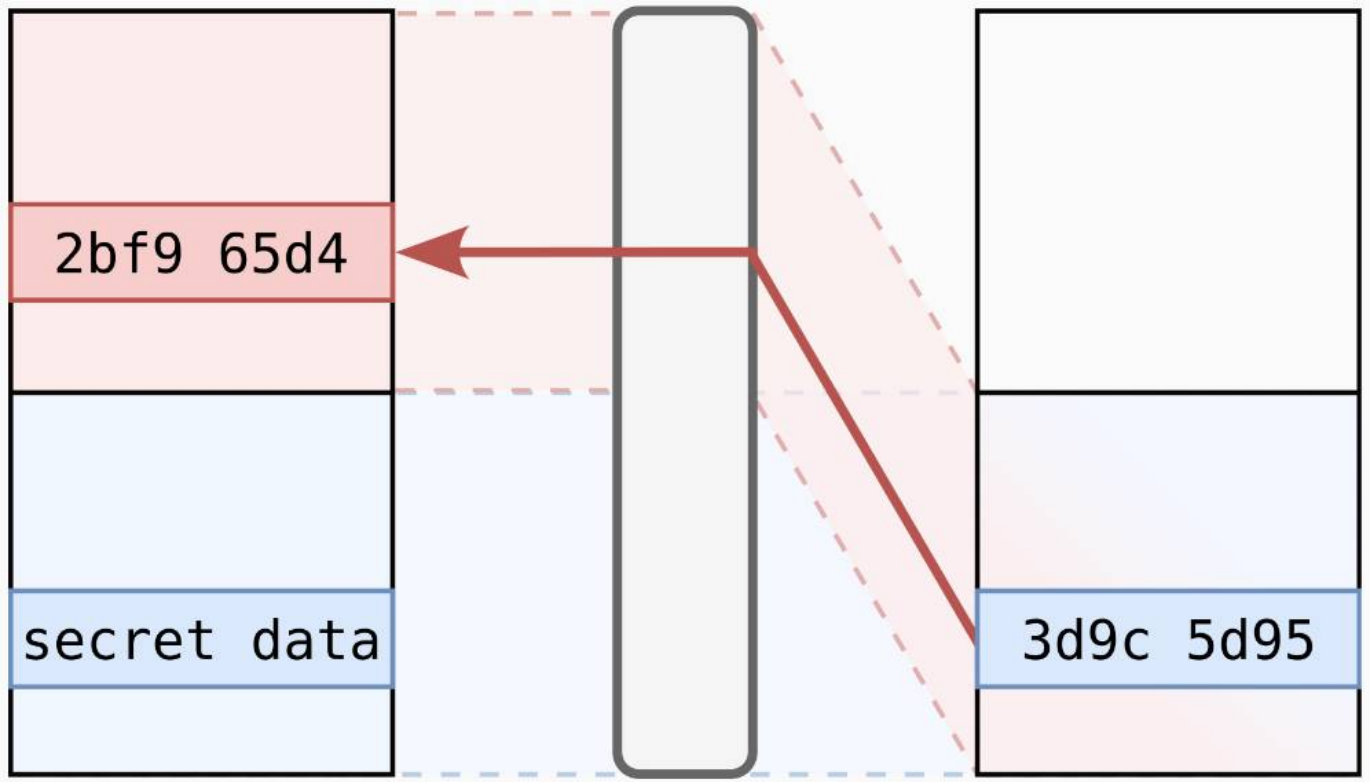
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



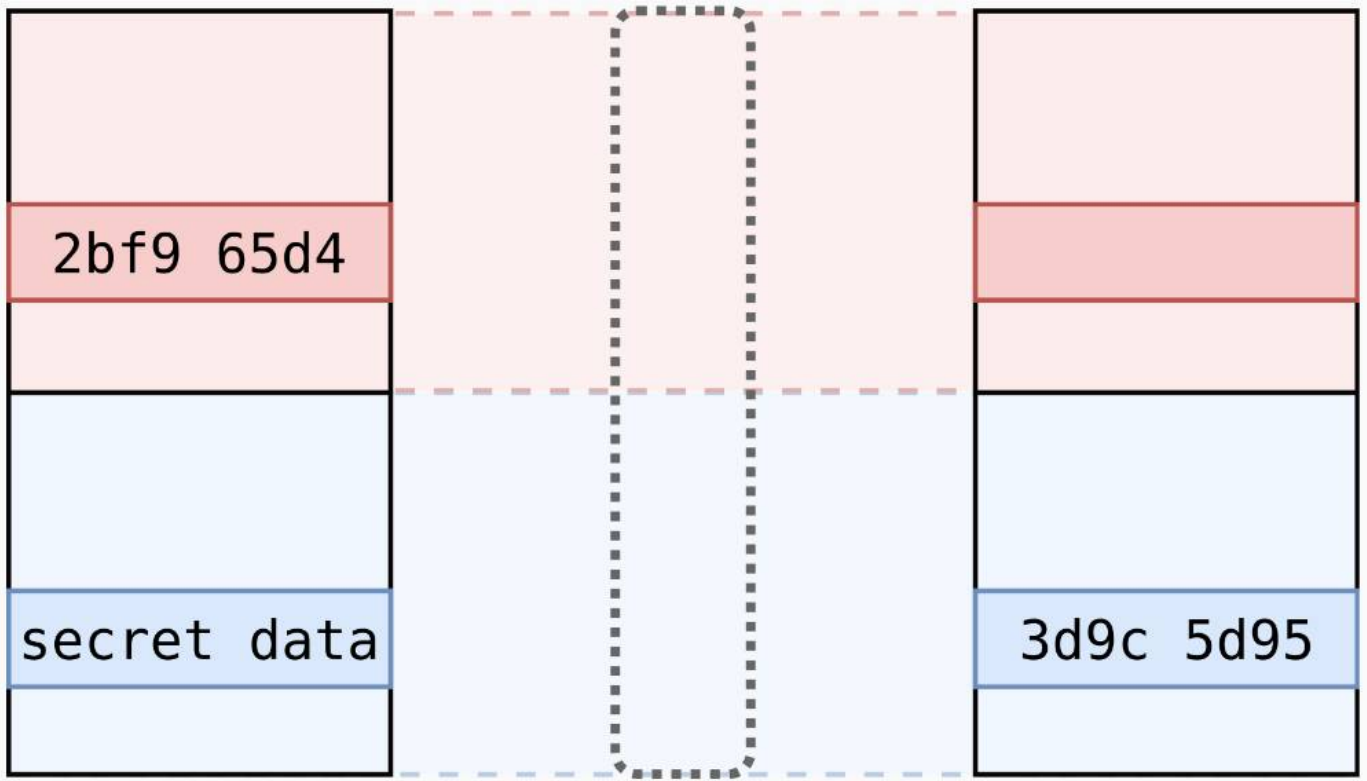
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



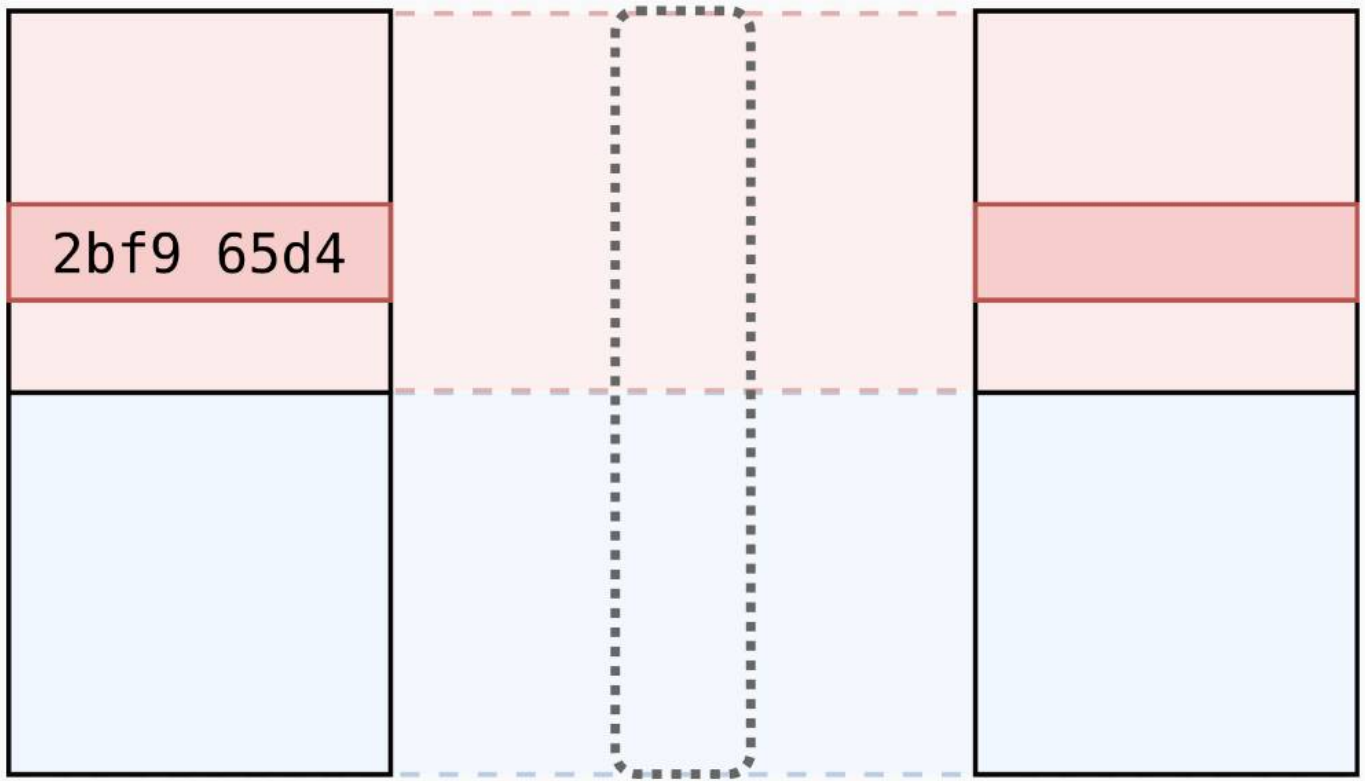
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



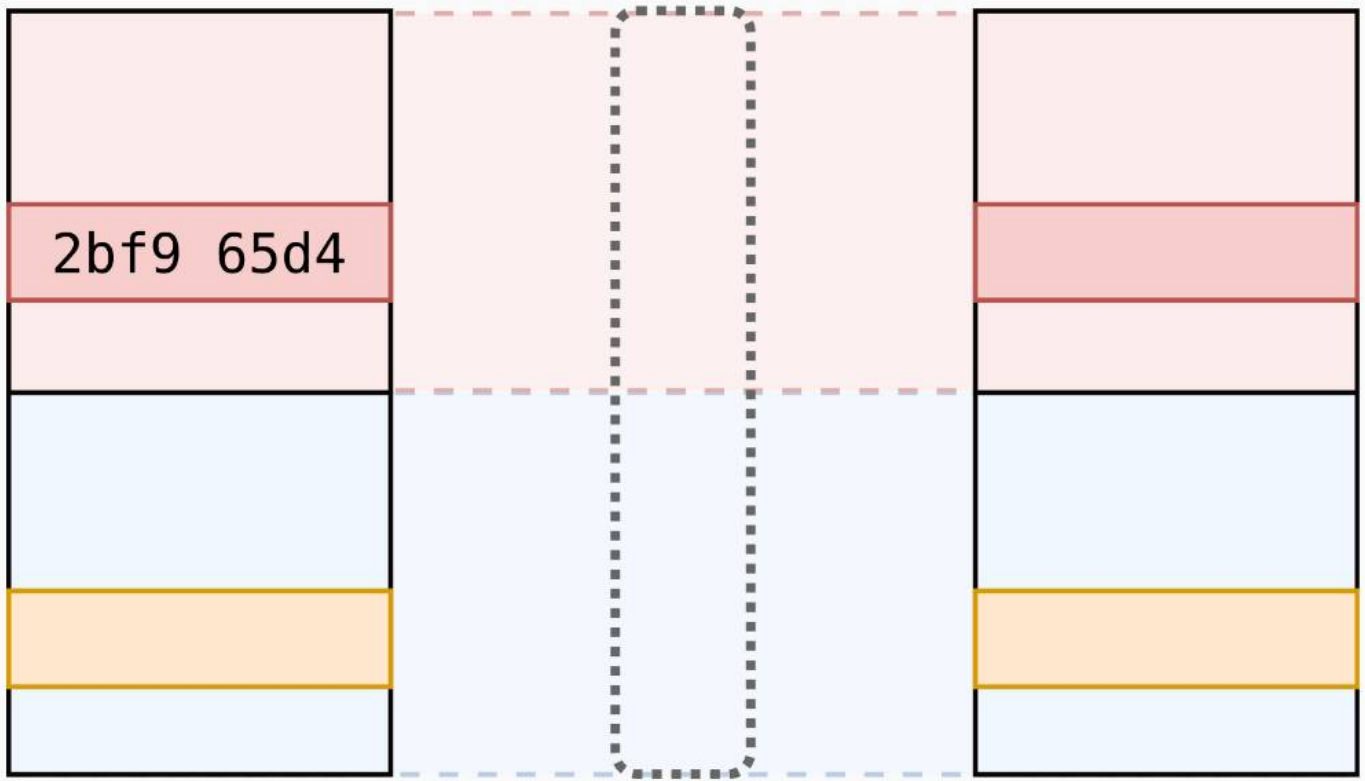
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



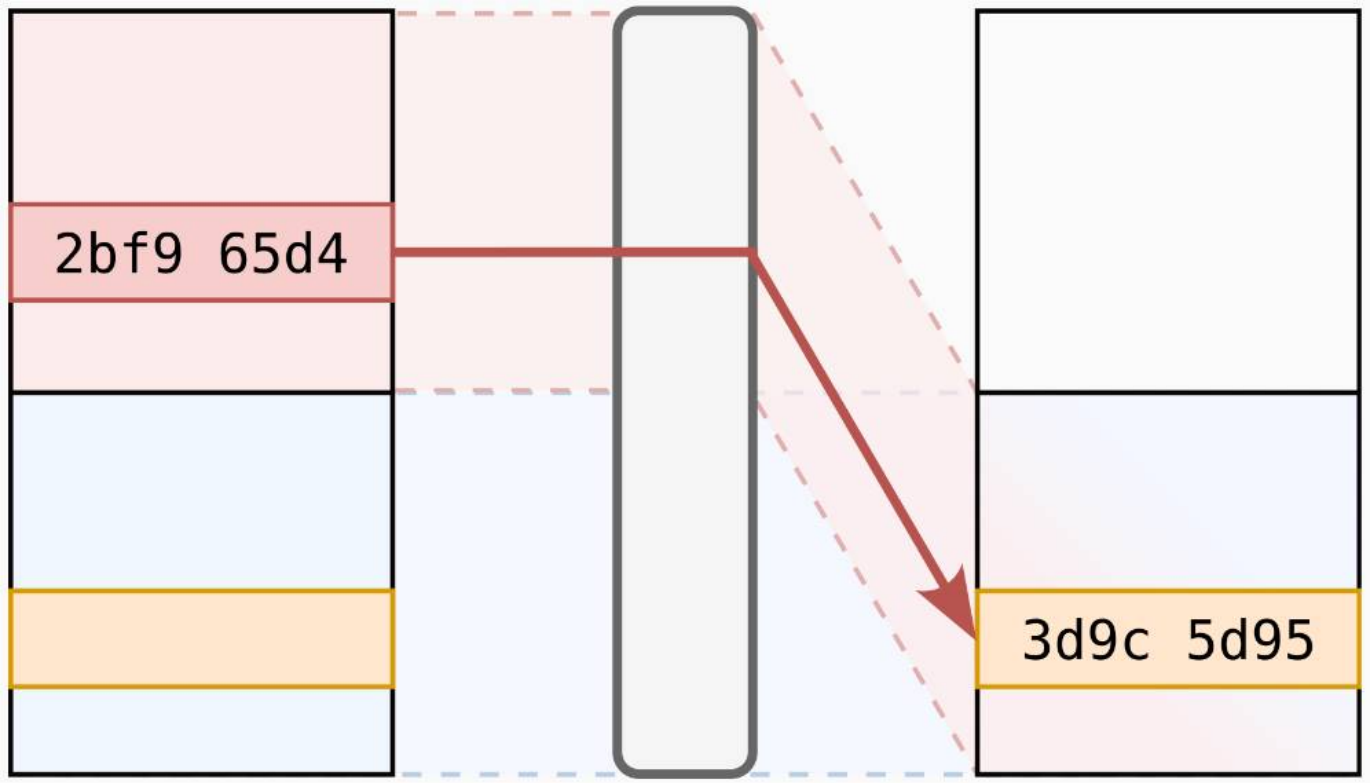
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



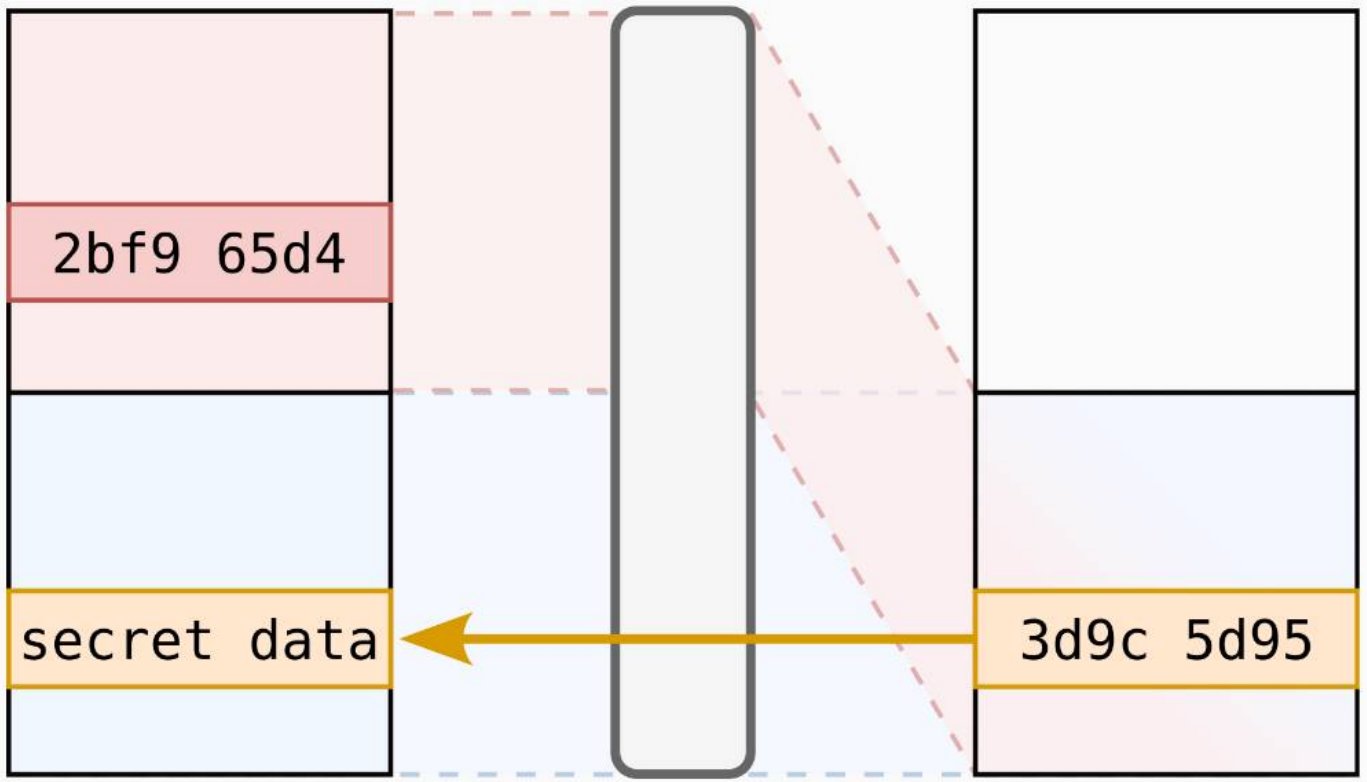
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



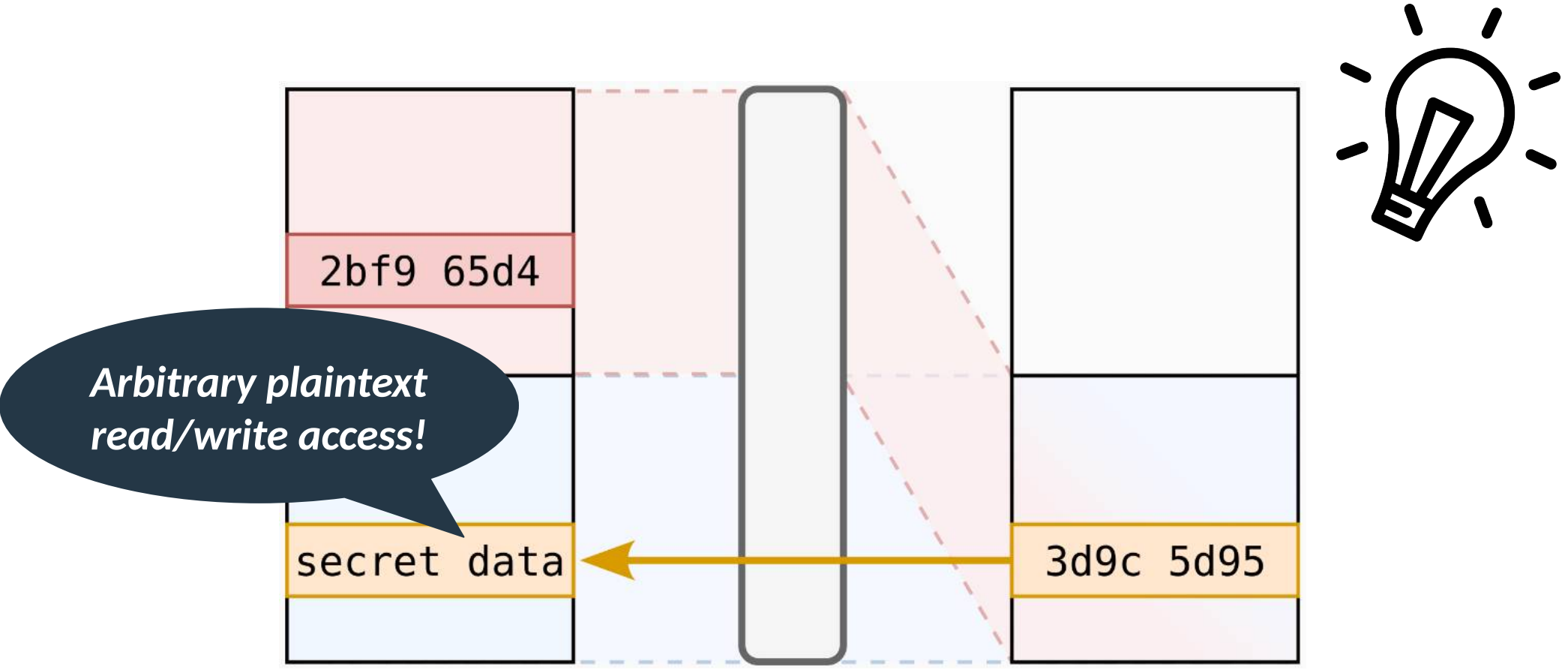
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



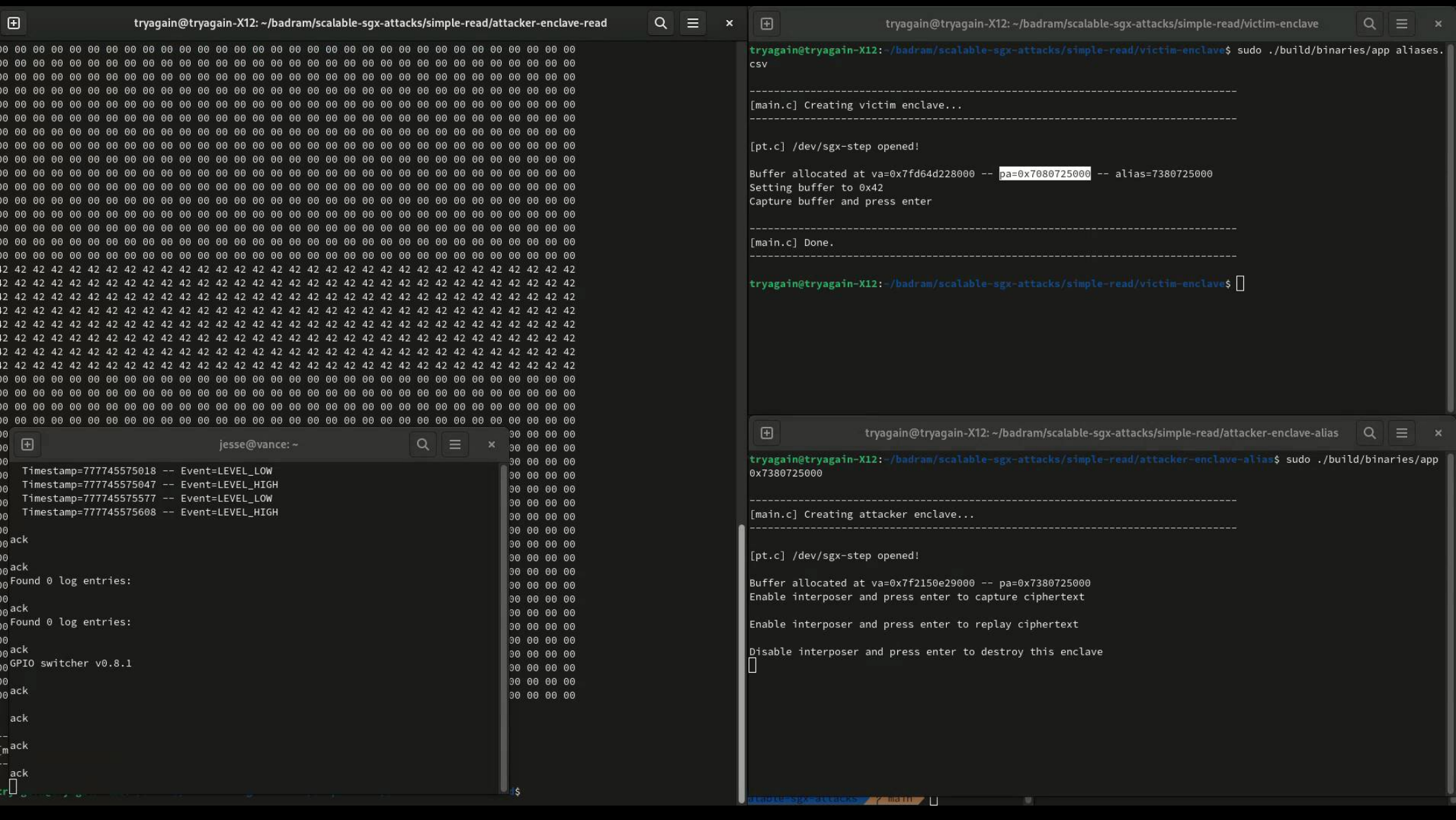
Battering RAM: Breaking Intel Scalable SGX Memory Encryption



Battering RAM: Breaking Intel Scalable SGX Memory Encryption




Demo
Arbitrary plaintext access on Intel Scalable SGX


tryagain@tryagain-X12: ~/badram/scalable-sgx-attacks/simple-read/attacker-enclave-read
tryagain@tryagain-X12: ~/badram/scalable-sgx-attacks/simple-read/victim-enclave
tryagain@tryagain-X12:~/badram/scalable-sgx-attacks/simple-read/victim-enclave$ sudo ./build/binaries/app aliases.csv
[main.c] Creating victim enclave...
[pt.c] /dev/sgx-step opened!
Buffer allocated at va=0x7fd64d228000 -- pa=0x7080725000 -- alias=7380725000
Setting buffer to 0x42
Capture buffer and press enter
[main.c] Done.
tryagain@tryagain-X12:~/badram/scalable-sgx-attacks/simple-read/victim-enclave$
jesse@vance: ~
Timestamp=777745575018 -- Event=LEVEL_LOW
Timestamp=777745575047 -- Event=LEVEL_HIGH
Timestamp=777745575577 -- Event=LEVEL_LOW
Timestamp=777745575608 -- Event=LEVEL_HIGH
Found 0 log entries:
Found 0 log entries:
GPIO switcher v0.8.1
tryagain@tryagain-X12: ~/badram/scalable-sgx-attacks/simple-read/attacker-enclave-alias
tryagain@tryagain-X12:~/badram/scalable-sgx-attacks/simple-read/attacker-enclave-alias$ sudo ./build/binaries/app 0x7380725000
[main.c] Creating attacker enclave...
[pt.c] /dev/sgx-step opened!
Buffer allocated at va=0x7f2150e29000 -- pa=0x7380725000
Enable interposer and press enter to capture ciphertext
Enable interposer and press enter to replay ciphertext
Disable interposer and press enter to destroy this enclave

ars TECHNICA AI BIZ & IT CARS CULTURE GAMING HEALTH POLICY SCIENCE SECURITY SPACE TECH FORUM SUBSCRIBE SIGN IN

Intel and AMD trusted enclaves, a foundation for network security, fall to physical attacks

The chipmakers say physical attacks aren't in the threat model. Many users didn't get the memo.

NewScientist
IDEEËN DIE DE WERELD VERANDEREN

'Voor zo'n hackaanval had je ooit tonnen nodig, nu kan het met minder dan vijftig euro'

Onderzoekers KU Leuven hacken hyperbeveiligde cloudsoftware met paneeltje van 50 euro: "Zeker niet zomaar afgaan op de beweringen van technologiebedrijven"

vrt nws

CLOUD SECURITY

Battering RAM Attack Breaks Intel and AMD Security Tech With $50 Device

Intel and AMD say the research is not in scope of their threat model because the attack requires physical access to a device.

By Eduard Kovacs | October 1, 2025 (4:50 AM ET)

🌍 https://batteringram.eu/

SEV-SNP Physical Memory Aliasing

AMD ID: AMD-SB-3024
Potential Impact: N/A

Summary

Researchers have reported a method for privileged attackers with physical access to a motherboard to potentially compromise confidentiality and integrity of AMD Secure Encrypted Virtualization – Secure Nesting Paging (SEV-SNP) guests.

AMD does not plan to release any mitigations in response to this report because the reported exploit is outside the scope of the published threat model for SEV-SNP, as detailed in Table 1 of the AMD SEV-SNP technical paper.

https://www.amd.com/en/resources/product-security/bulletin/amd-sb-3024.html

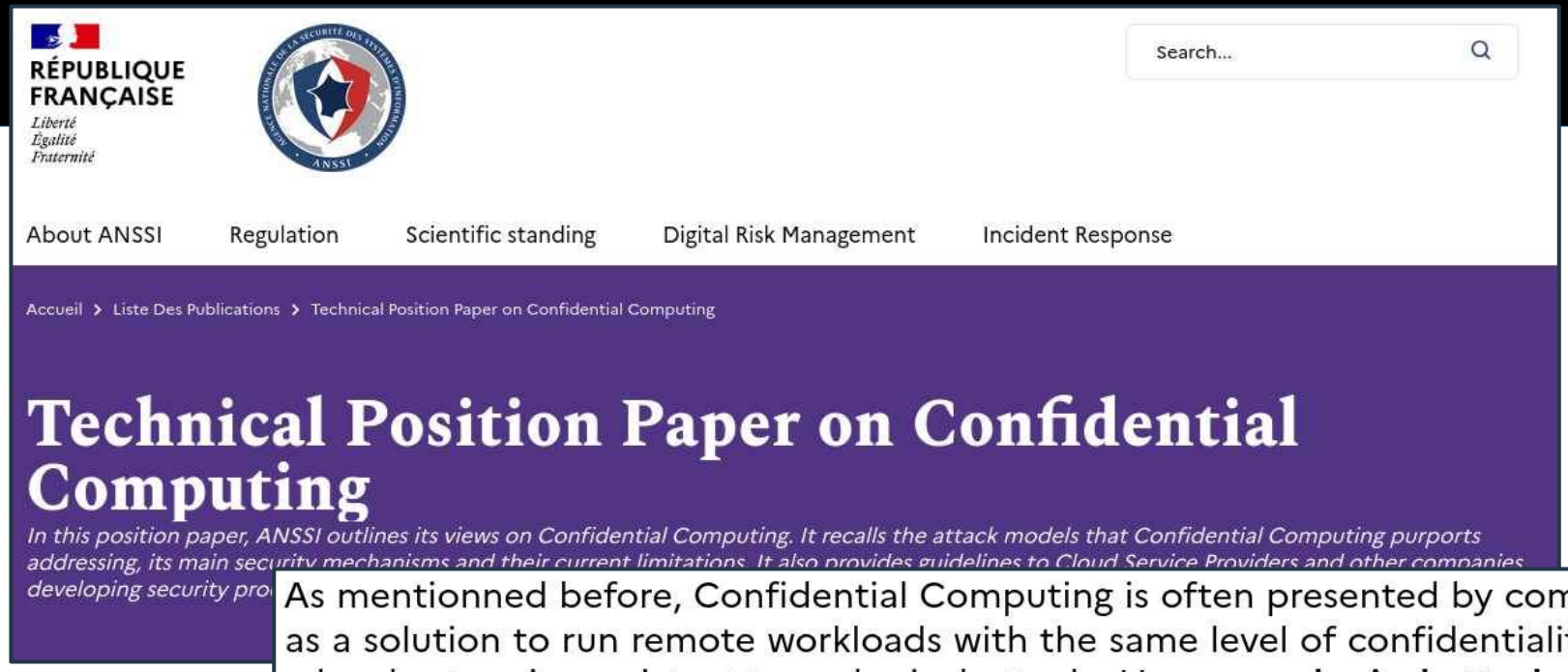
More Information on Encrypted Memory Frameworks for Intel Confidential Computing

ID	Updated	Version	
865767	10/27/2025	1.0	Public

In the *Battering RAM* paper, researchers from KU Leuven and University of Birmingham developed a custom interposer to actively alias memory and gain arbitrary read/write access into Intel SGX-protected memory.

Both research teams assume a physical adversary has direct access to the hardware with a memory bus interposer. Both methods can then be used to attack Intel SGX-protected assets, including Intel SGX attestation keys. In a separate disclosure to Intel, Fortanix provided a potential attack that requires a replay-capable physical interposer. Such attacks are outside the scope of the boundary of protection offered by Advanced Encryption Standard-XEX-based Tweaked Codebook Mode with Ciphertext Stealing (AES-XTS) based memory encryption, as originally stated in the 2021 Intel publication Supporting Intel® SGX on Multi-socket Platforms. As it provides limited confidentiality protection, and no integrity or anti-replay protection against attackers with physical capabilities, Intel does not plan to issue a CVE.

https://www.intel.com/content/www/us/en/developer/articles/news/more-information-encrypted-memory-frameworks.html



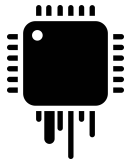
As mentionned before, Confidential Computing is often presented by commercial providers as a solution to run remote workloads with the same level of confidentiality and integrity as a local setup, *i.e.* resistant to a physical attack. However, **physical attacks are explicitly out-of-scope of the security target defined by hardware vendors**. This means in particular that if a user is concerned about a cloud-provider conducting targeted attacks, instead of relying on a Confidential Computing approach they need to switch to a cloud-provider they trust, *i.e.* with strong counterparts or control capabilities, or use their own hardware with physical security protection measures. Likewise, the security of Confidential Computing assumes an uncompromised Manufacturer TCB: manufacturer and supply-chain attackers, including state-level ones, are thus explicitly out-of-scope.

https://cyber.gouv.fr/en/publications/technical-position-paper-confidential-computing

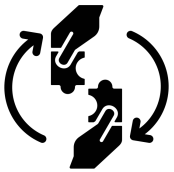
DALL·E 3

Conclusions and Take-Away

New era of **confidential computing** for the cloud



… but current architectures are **not perfect!**



Scientific understanding driven by **attacker-defender race**