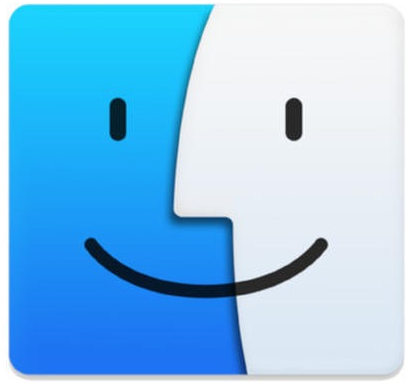
Towards Practical Interrupt Side-Channel Attacks on macOS for Apple Silicon

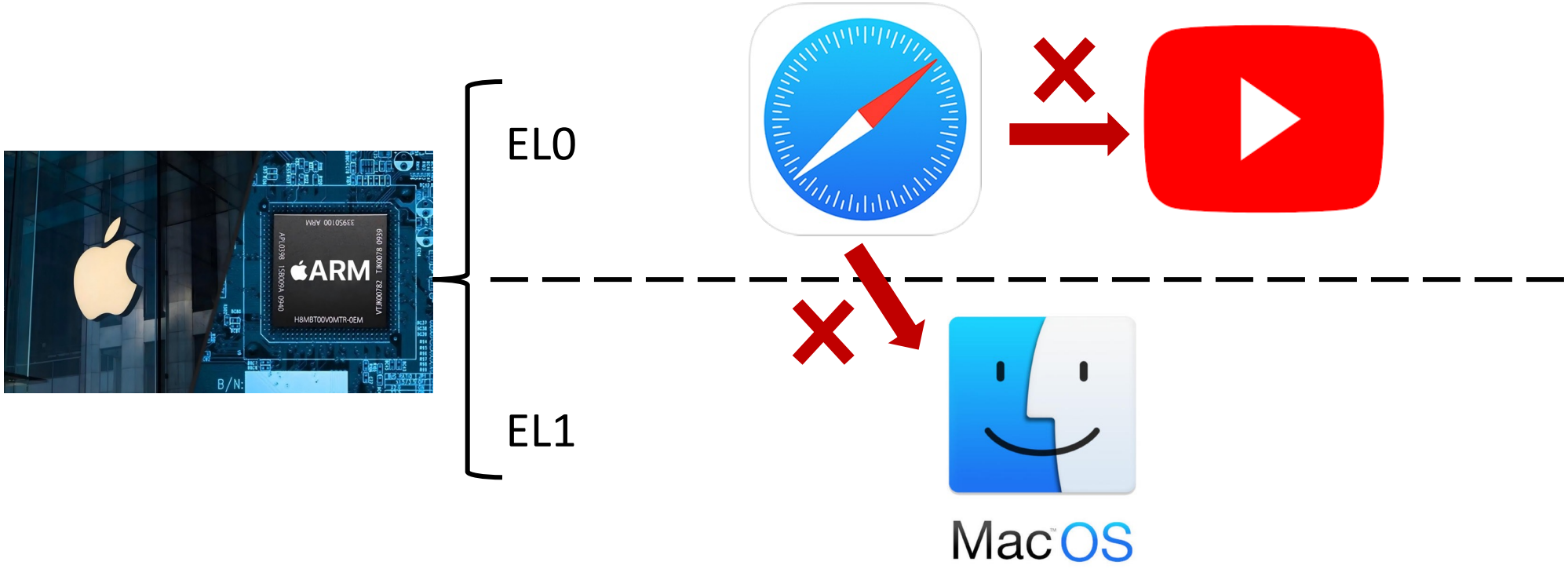
Xin Zhang, Chang Liu, Jiajun Zou, Yi Yang, Qingni Shen, Zhi Zhang, Trevor E. Carlson

Apple ecosystem

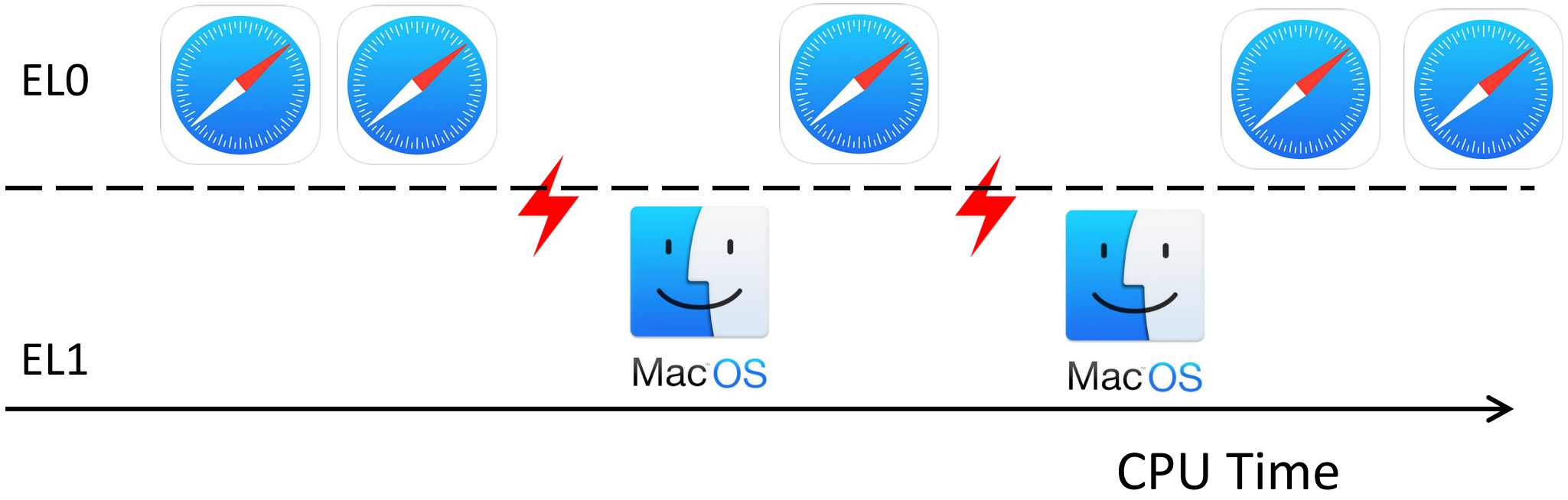
The high-end CPU landscape has shifted from an x86-dominated market to increasingly featuring high-performance Arm designs, most notably Apple’s M-series processors.



Privileges

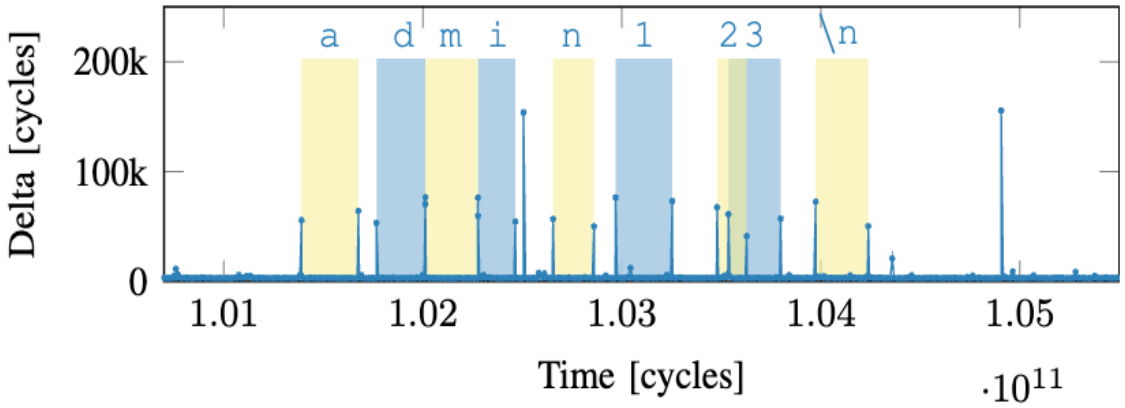


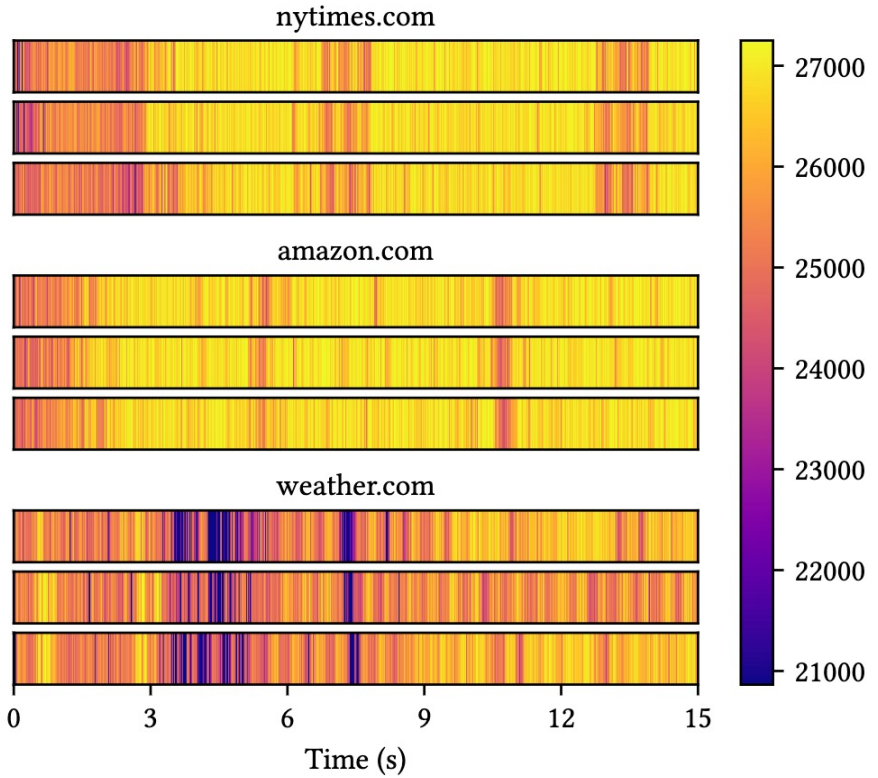
Interrupts



Interrupt Timing is Sensitive

Interrupt side channels have been widely applied into various attack scenarios.





[1] KeyDrown: Eliminating Software-Based Keystroke Timing Side-Channel Attacks, published in Network and Distributed System Security (NDSS) Symposium 2018.

[2] There's Always a Bigger Fish: A Clarifying Analysis of a Machine-Learning-Assisted Side-Channel Attack, published in International Symposium on Computer Architecture (ISCA) 2022.

Challenges

- Software detection: macOS does not provide `/proc/interrupts` interfaces
- Hardware detection: Existing timer-free techniques rely on x86-specific mechanisms
- Interrupt delivery: Apple uses proprietry interrupt controller design

Research Question

- Can we precisely detect interrupts from user-space on Apple silicon?

- Do established interrupt side-channel conclusions generalize to Apple silicon, or does Apple’s interrupt delivery design invalidate them?

Our Work

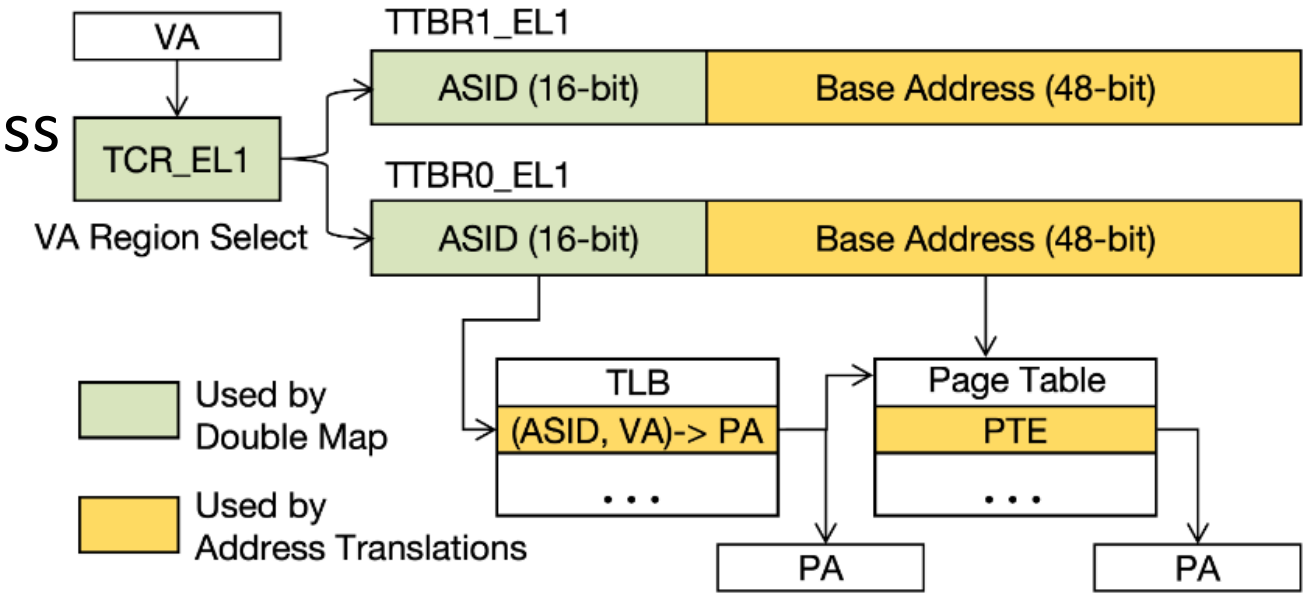
- We propose TIDE, a *timer-less interrupt detection* technique for macOS on M-series CPUs, revealing an unintended side effect of Apple's *Double Map* implementation on the Arm architecture.

- We use TIDE to reverse-engineer the *Apple interrupt delivery* mechanism, results of which show that it *uniformly* delivers shared peripheral interrupts to *every active core*.

Double Map

- To mitigate Meltdown attack, Apple introduced the Double Map feature in macOS 10.13.2 and iOS 11.2 (XNU kernel 4570.31.3)

- TTBR controls the base address of the page tables.
- TCR controls how virtual addresses are translated.



Double Map

- Under Double Map, two macros are defined to be executed before accessing the kernel stack.

- Both use x18 as a scratch register, thus overwriting its original value from user-space.

```
.macro MAP_KERNEL
    /* Switch to the kernel ASID for the task. */
    mrs     x18, TTBR0_EL1
    orr     x18, x18, #(1 << TTBR_ASID_SHIFT)
    msr     TTBR0_EL1, x18
    /* Update the TCR to map the kernel using the
     kernel ASID. */
    MOV64   x18, TCR_EL1_BOOT
    msr     TCR_EL1, x18
    isb     sy
.endmacro

.macro BRANCH_TO_KVA_VECTOR
    /* Find the kernelcache table for the exception
     vectors by accessing the per-CPU data. */
    mrs     x18, TPIDR_EL1
    ldr     x18, [x18, ACT_CPUDATAP]
    ldr     x18, [x18, CPU_EXC_VECTORS]
    /*Branch to the corresponding handler.*/
    ldr     x18, [x18, #($1 << 3)]
    br      x18
.endmacro
...
Lel0_irq_vector_64:
    MAP_KERNEL
    BRANCH_TO_KVA_VECTOR Lel0_irq_vector_64_long, 9
```

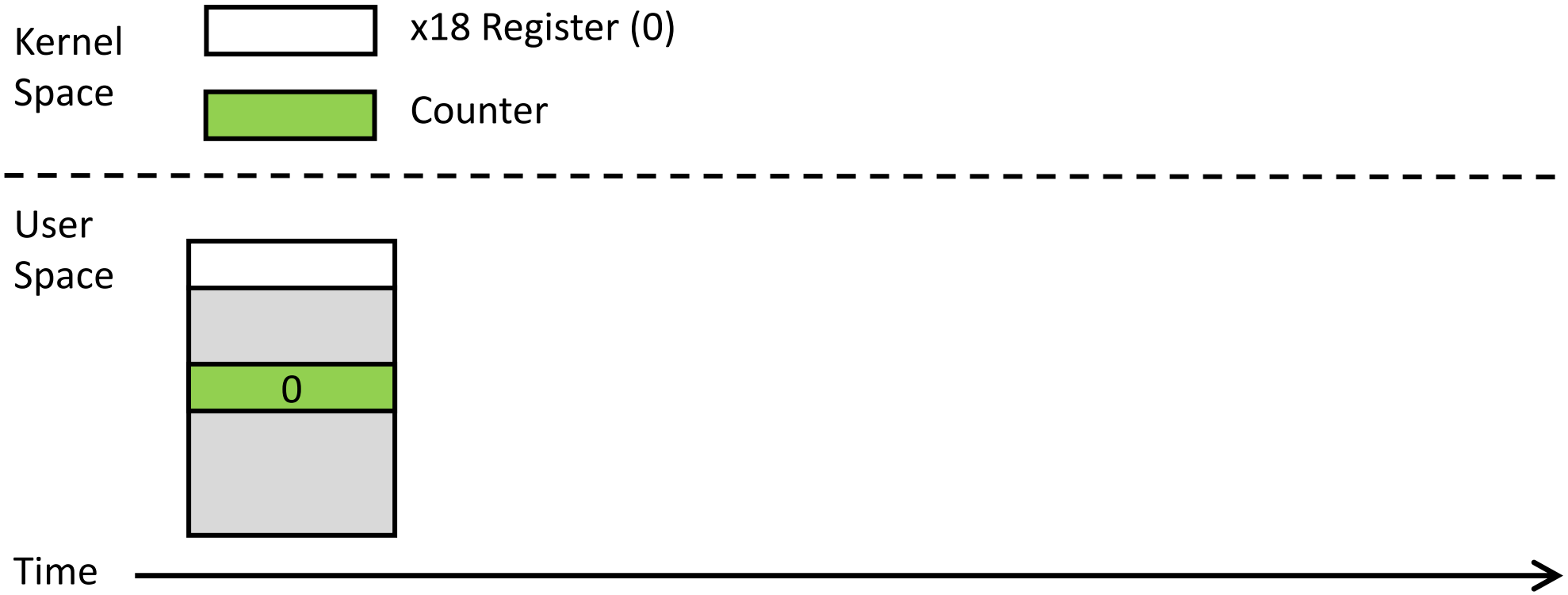
Platform Registers

The x18 register is always cleared during each context switches !

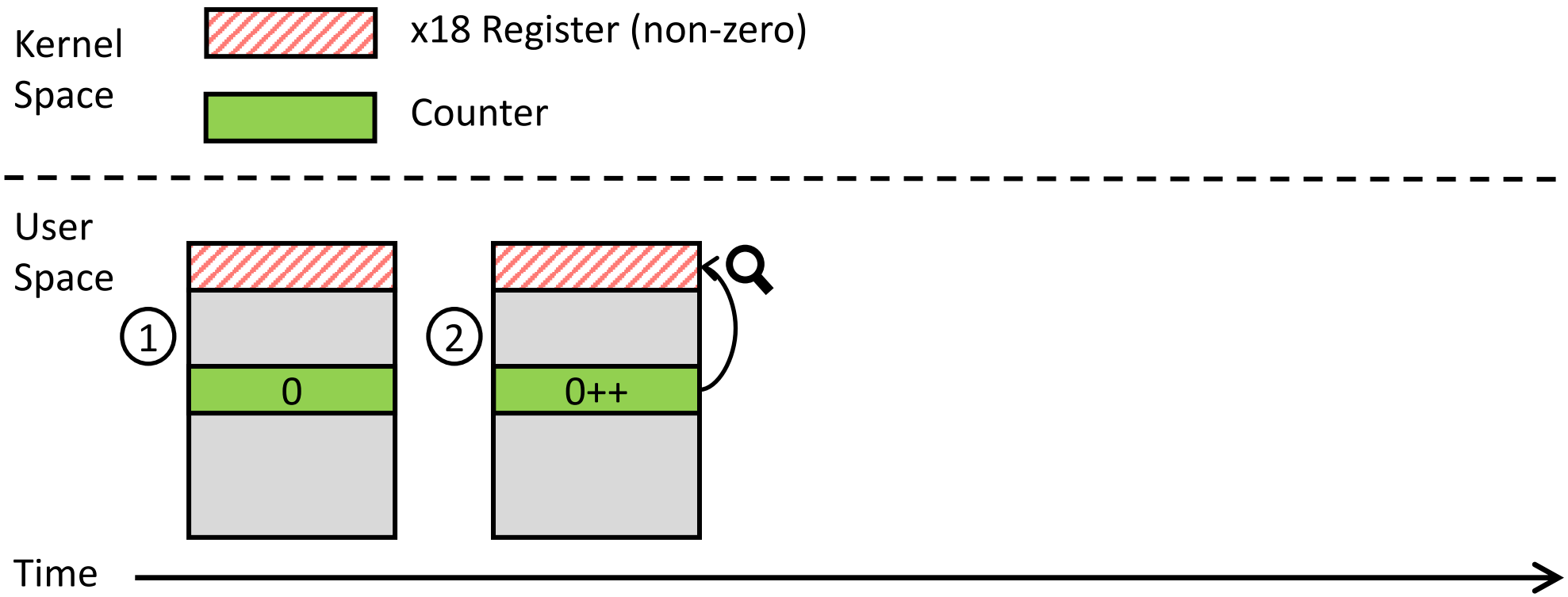
```
void set_x18(int value) {
    __asm__ volatile("mov x18, %x0" : : "r"((uint64_t)value));
}

static int get_x18() {
    uint64_t value;
    __asm__ volatile("mov %x0, x18" : "=r"(value));
    return (int)value;
}
```

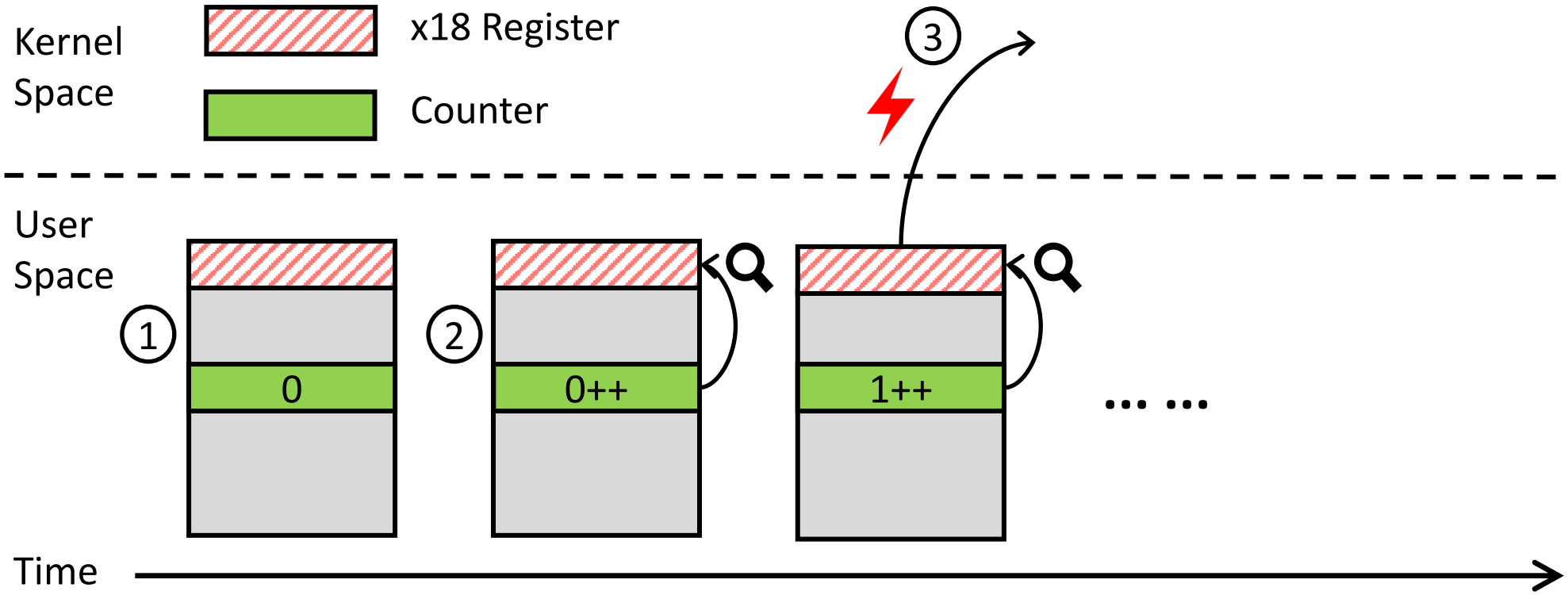
Apply TIDE to Detect Interrupt Timing



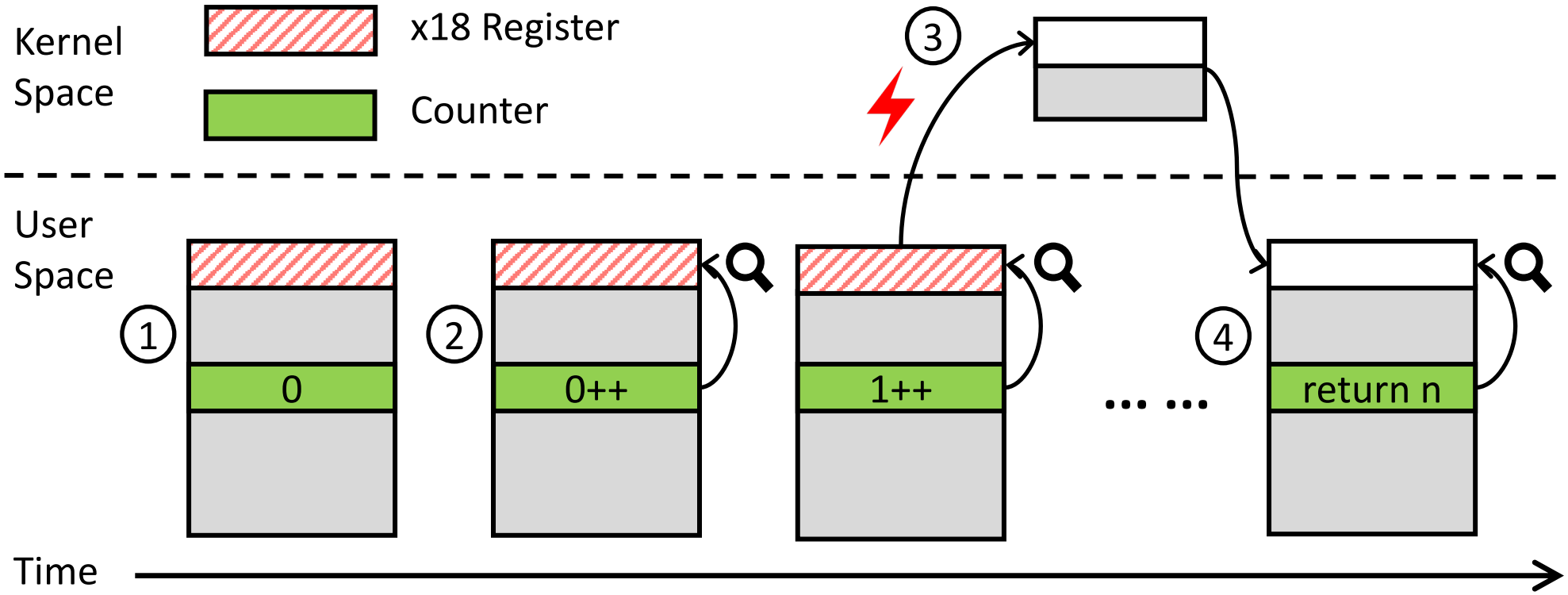
Apply TIDE to Detect Interrupt Timing



Apply TIDE to Detect Interrupt Timing

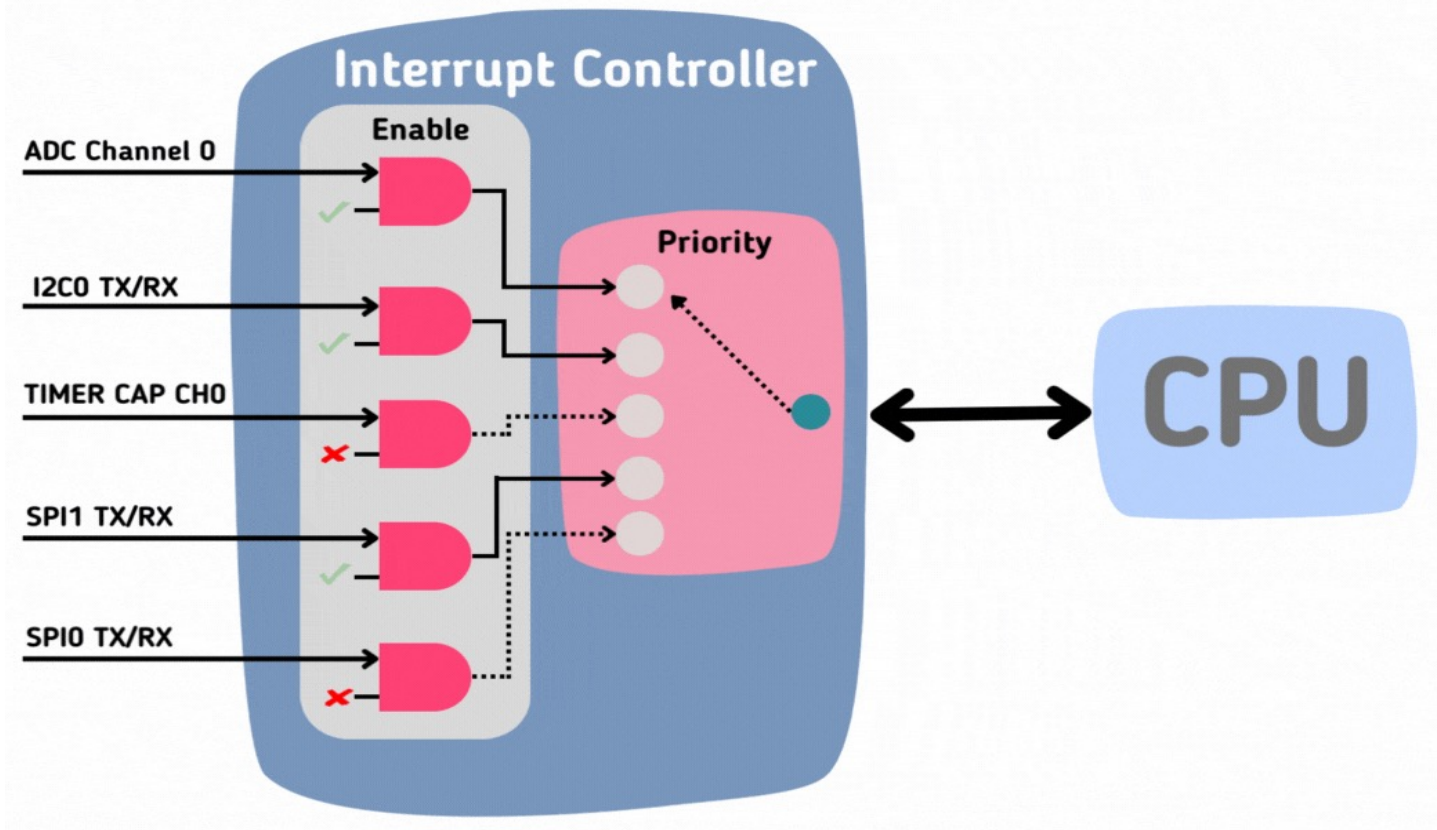


Apply TIDE to Detect Interrupt Timing



Apple Interrupt Controller

TIDE only detects interrupts on its operating core, extending it to a multi-core attack requires understanding the Apple interrupt controller.



Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

Efficiency: To quantify how often interrupts are received under a certain setting, we defined the efficiency metric as the ratio between the number of SPI requests generated and the number of interrupts successfully received.

Apple Interrupt Controller

Goals:

- Q1: How does AIC deliver shared peripheral interrupts?
- Q2: What factors can affect interrupt delivery decisions?

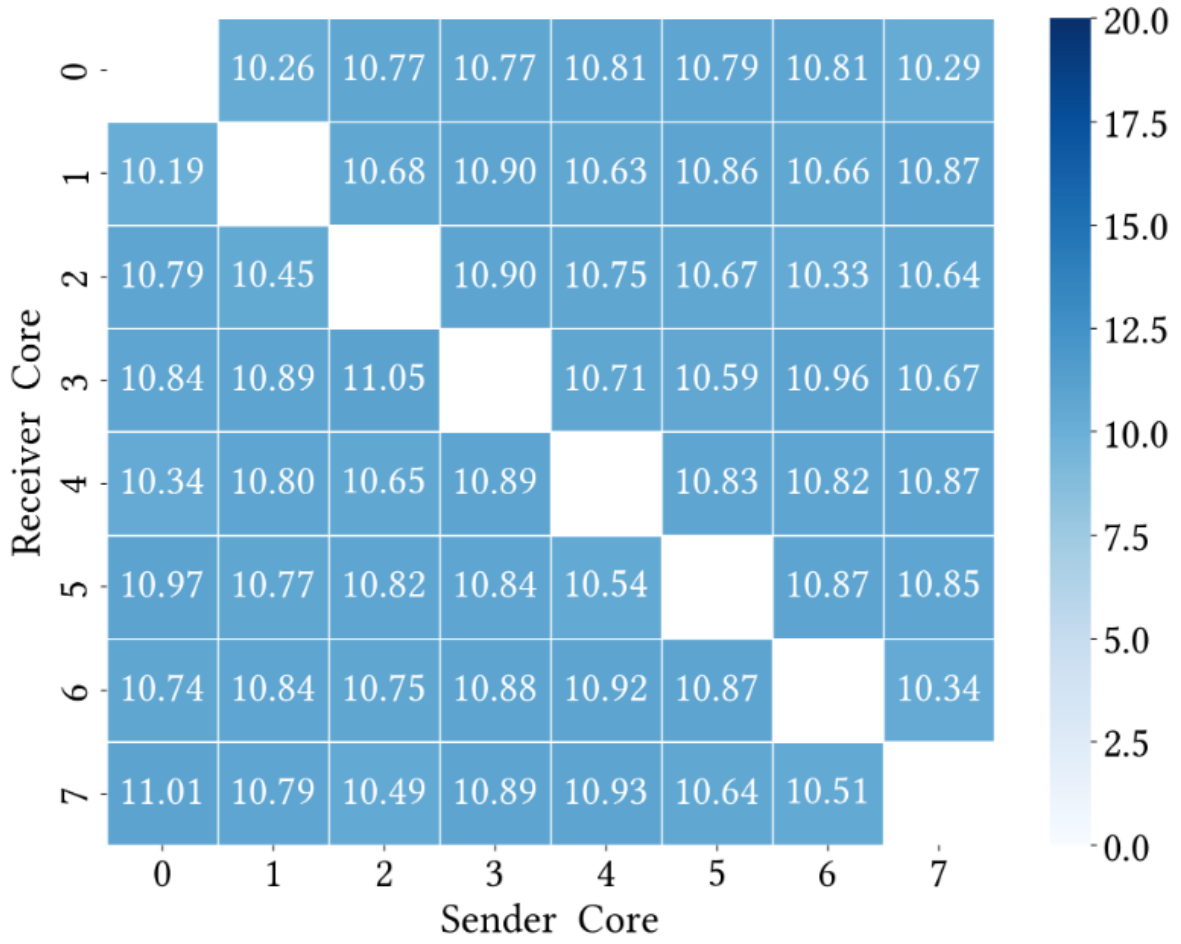
Efficiency: To quantify how often interrupts are received under a certain setting, we defined the efficiency metric as the ratio between the number of SPI requests generated and the number of interrupts successfully received.

We employ CoreBinder, a kernel extension that allows pinning threads to specific CPU cores on Apple silicon.

https://github.com/junjie1475/MacOS_CoreBinder

Apple Interrupt Controller

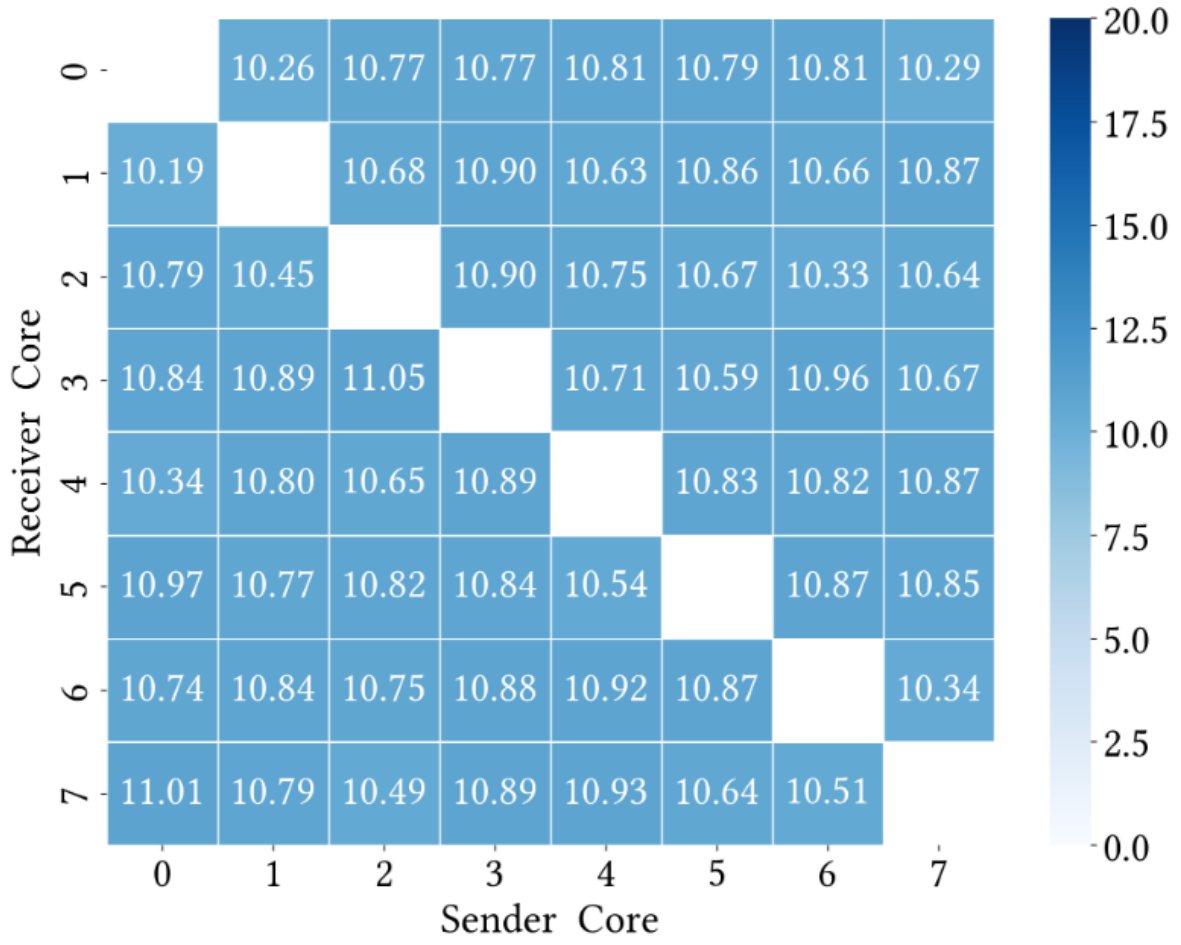
The figure shows the efficiency matrix when we bind sender and receiver to specific cores. Each cell corresponds to one pair of cores.



Apple Interrupt Controller

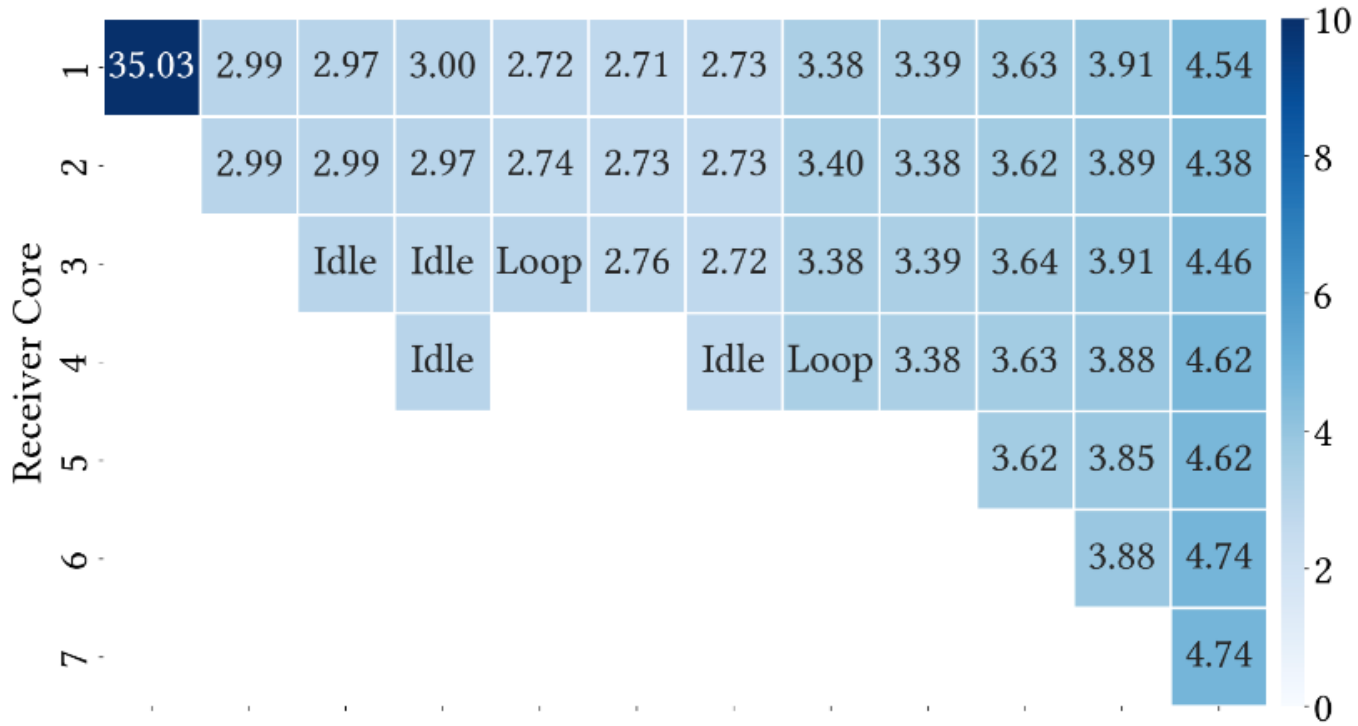
The figure shows the efficiency matrix when we bind sender and receiver to specific cores. Each cell corresponds to one pair of cores.

Unlike Linux-based systems, the combination of Apple silicon and macOS does not adopt an interrupt affinity mechanism, and deliver shared peripheral interrupts to a fixed core.



Apple Interrupt Controller

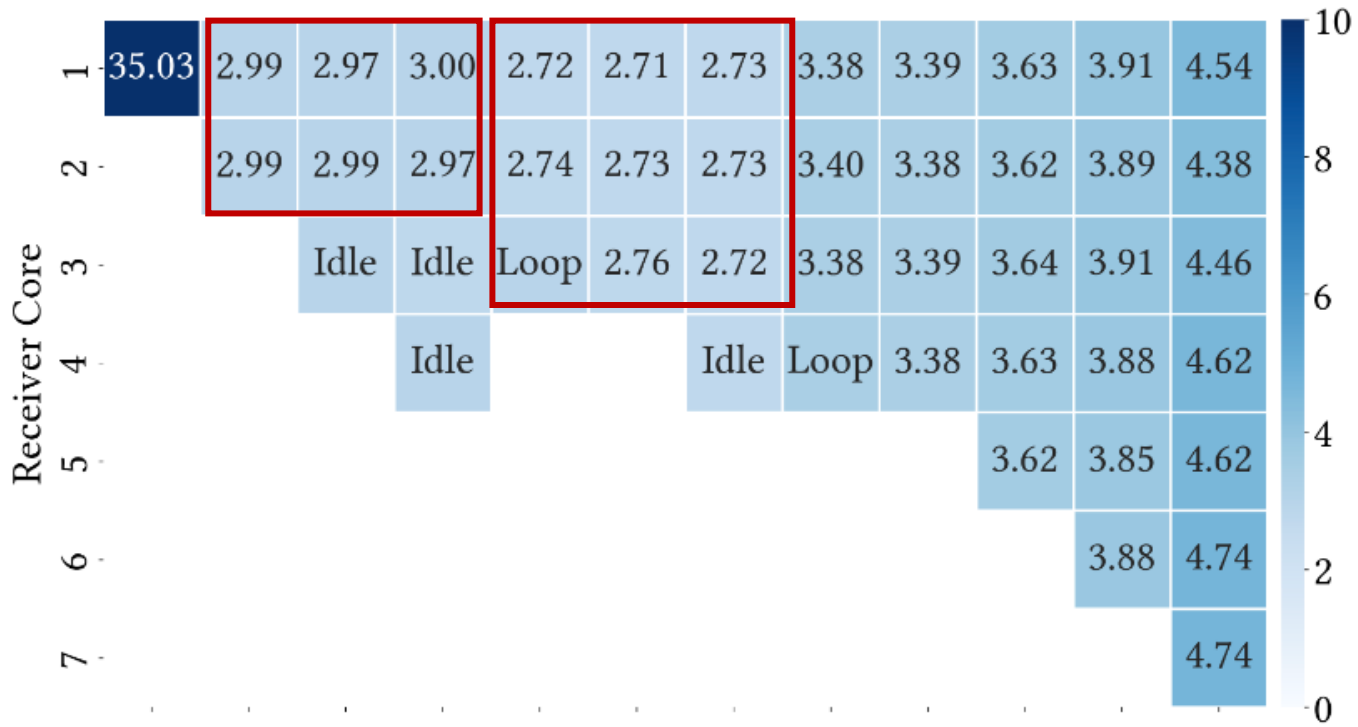
The heatmap shows the interrupt efficiency under different configurations.



Apple Interrupt Controller

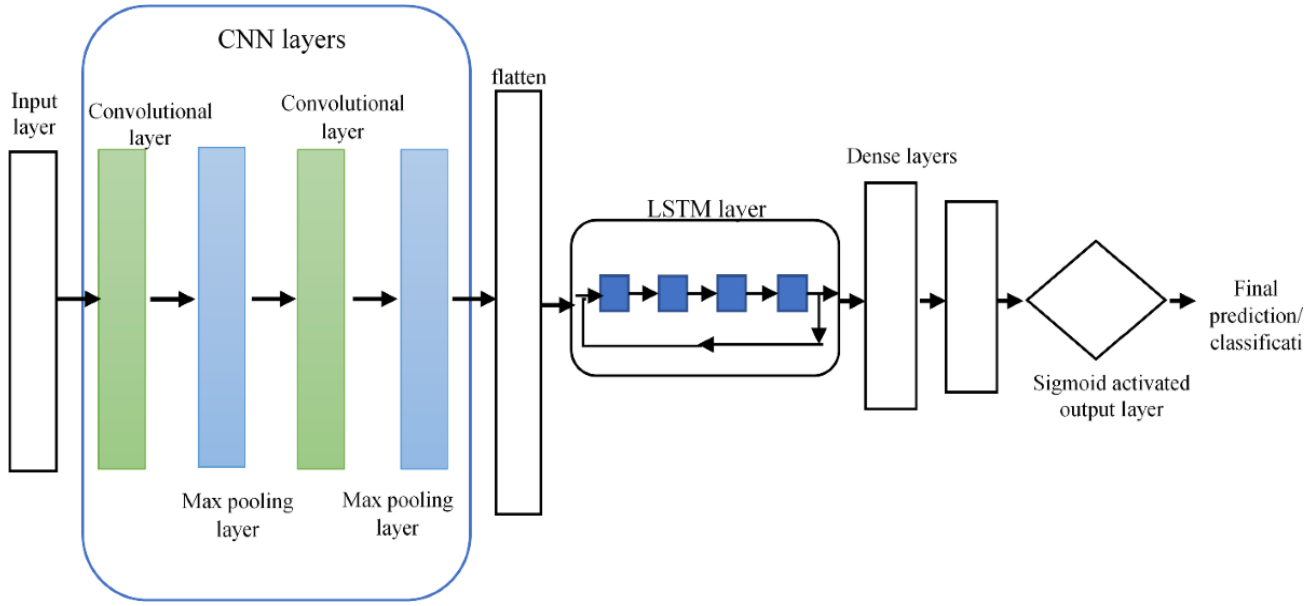
The heatmap shows the interrupt efficiency under different configurations.

Apple silicon with macOS uniformly delivers SPIs to all active cores and ignores the idle cores.



Case Studies——Website and Video Fingerprinting

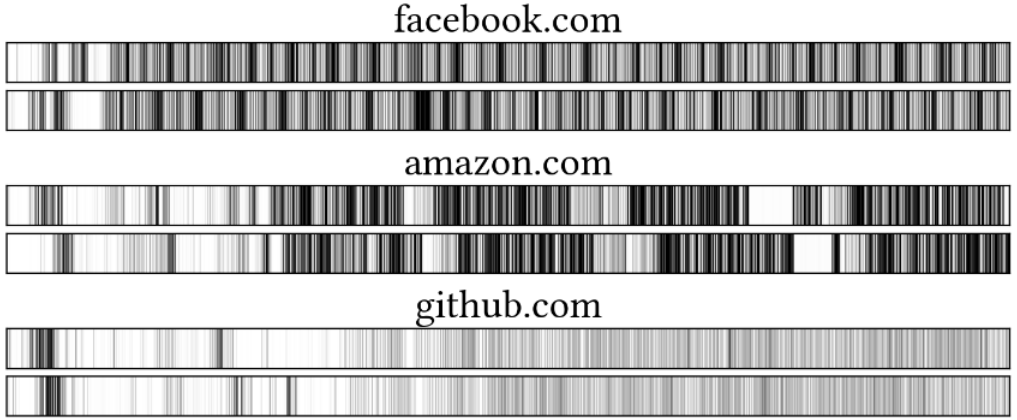
- We use TIDE to collect interrupt traces while Safari loads webpages, then use an ML classifier to infer the visited website/video.
- Model: LSTM with 32 units, trained and evaluated using 10-fold cross validation.



https://github.com/jackcook/bigger-fish/tree/main

Case Studies——Website and Video Fingerprinting

- When accessing a website, distinctive network packets will be sent and received, which generates distinctive network interrupts.



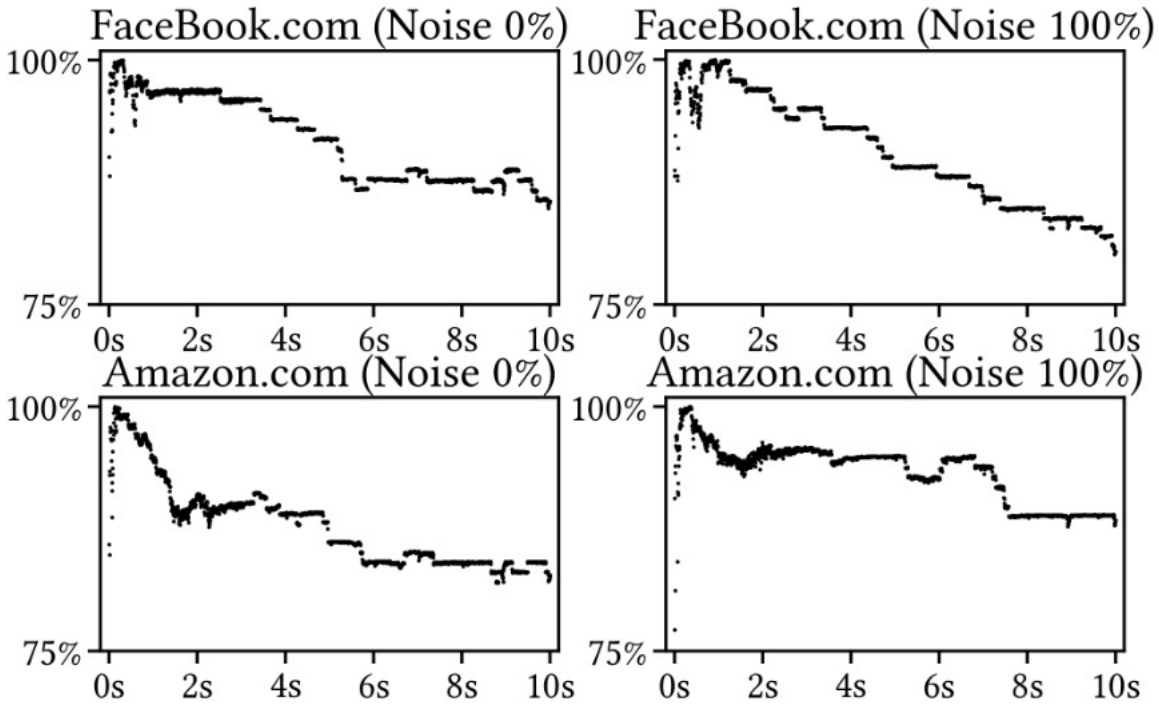
- Closed-world: 100 target websites or 20 target videos

- Open-world: 100 target websites plus an other class of non-target websites.

Attack Scenario	Default		Fixed Frequency	
	Top-1 Acc	Top-5 Acc	Top-1 Acc	Top-5 Acc
Closed-world Website FP	93.8%	98.7%	93.1%	98.5%
Open-world Website FP	91.5%	98.5%	91.1%	98.3%
Video FP	78.1%	97.9%	87.2%	98.3%

Case Studies——Enhanced Website Fingerprinting

- The *number of active cores* changes the interrupt signal observed by the attacker, affecting all interrupt side channels on Apple silicon.
- For loop-counting attacks [1], training under one noise level fails to generalize to another.
- By augmenting training with active-core information, we recover robust accuracy under both clean and noisy settings.



Test \ Train	Noise (0%)	Noise (20%)	Noise (100%)	Enhanced
Noise (0%)	94.8%	89.4%	39.3%	93.6%
Noise (100%)	40.7%	66.4%	92.7%	92.8%

[1] There's Always a Bigger Fish: A Clarifying Analysis of a Machine-Learning-Assisted Side-Channel Attack, published in International Symposium on Computer Architecture (ISCA) 2022.

Case Studies——Denoising Counting-Thread Timers

- We use TIDE to denoise SysBumps[1] attack under heavy interrupt noise, improving the success rate of KASLR derandomization from 54% to 81%.

- We apply TIDE to a covert channel, achieving an average bandwidth of 111.65 bps with an average error rate of 4.45%.

- We use TIDE to distinguish key-bit guesses in the CIRCL cryptographic library (32,119,284 when $M_i=M_{i-1}$; 31,362,263 when $M_i!=M_{i-1}$).

[1] SysBumps: Exploiting Speculative Execution in System Calls for Breaking KASLR in macOS for Apple Silicon, published in ACM SIGSAC Conference on Computer and Communications Security (CCS) 2024.

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.
- On 15-March-2025, the macOS security team requested a copy of our manuscript

Vulnerability Disclosure

- On 22-January-2025, we disclosed our findings to Apple’s hardware security team.
- On 2-February-2025, we disclosed our findings to Apple’s macOS security team.
- On 2-February-2025, the hardware security team responded on and stated: We have forwarded your report to the appropriate team to investigate as a potential future hardening effort.
- On 15-March-2025, the macOS security team requested a copy of our manuscript
- On 9-April-2025, the macOS security team commented: Our engineers found your paper intriguing, and it helped shed more light on the side-channel techniques you describe. This helps with the team's considerations for possible platform enhancements.

Mitigation

- Fully mitigating TIDE can be achieved by preserving the user's original x18 value.

93.8% -> 0%

- Injecting artificial jitter can reduce the signal strength, which serves as a general mitigation strategy.

93.8% -> 61.8%

Takeaway

- TIDE can be used to probe fine-grained interrupts *without any (external) timers* on Apple silicon. It reveals an *unintended side effect of Apple's Double Map* implementation on the Arm architecture
- Based on TIDE, we *reverse-engineer* the Apple interrupt delivery mechanism and show that many x86 interrupt side-channel conclusions *do not carry over* to Apple silicon.
- Artifact (Available, Functional, and Reproducible): https://github/zhangxin00/tide

Towards Practical Interrupt Side-Channel Attacks on macOS for Apple Silicon

Q & A