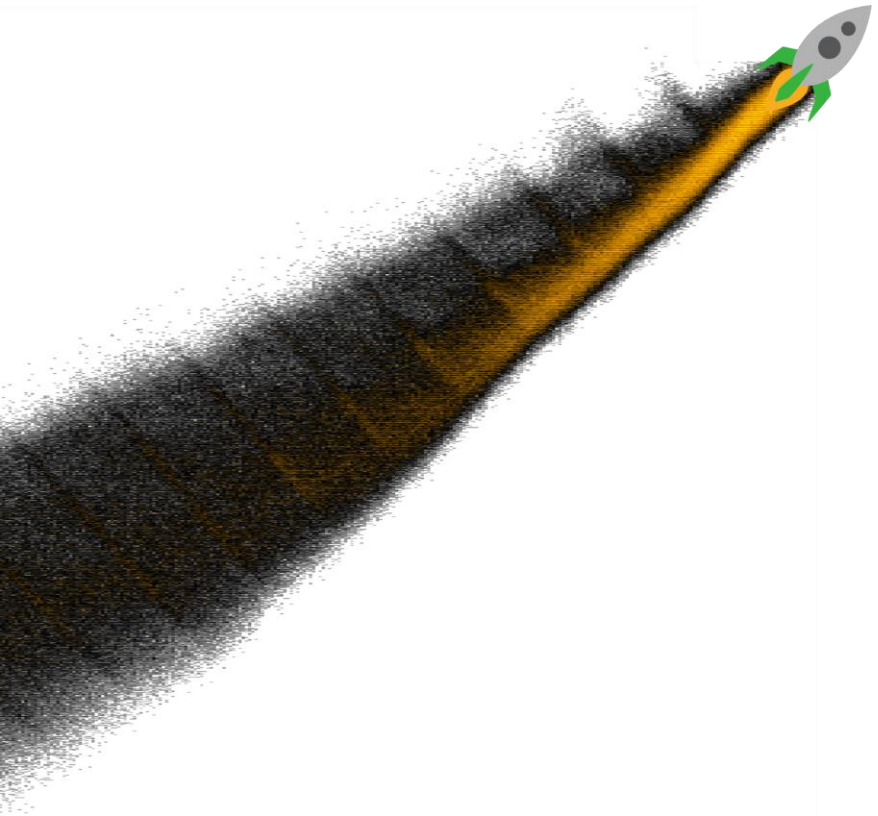
ETH zürich

Prevention of Microarchitectural Covert Channels on an Open-Source 64-bit RISC-V Core

Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020)



May 29th, 2020

Nils Wistoff

Moritz Schneider

Frank K. Gürkaynak

Luca Benini

Gernot Heiser

RWTH AACHEN UNIVERSITY

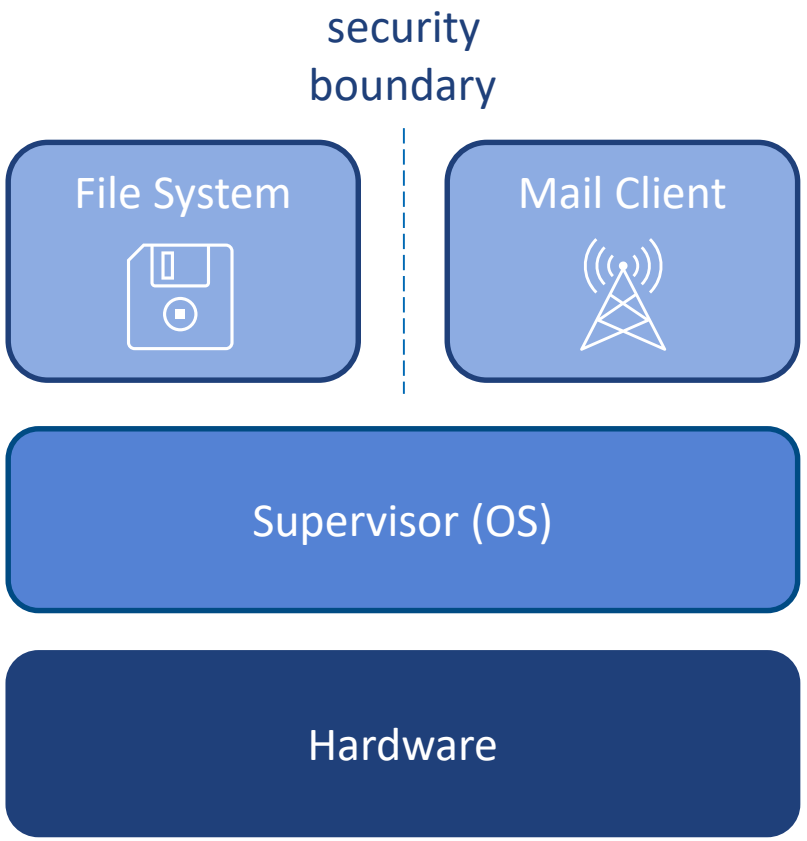
CSIRO DATA 61

HENSOLDT Detect and Protect.

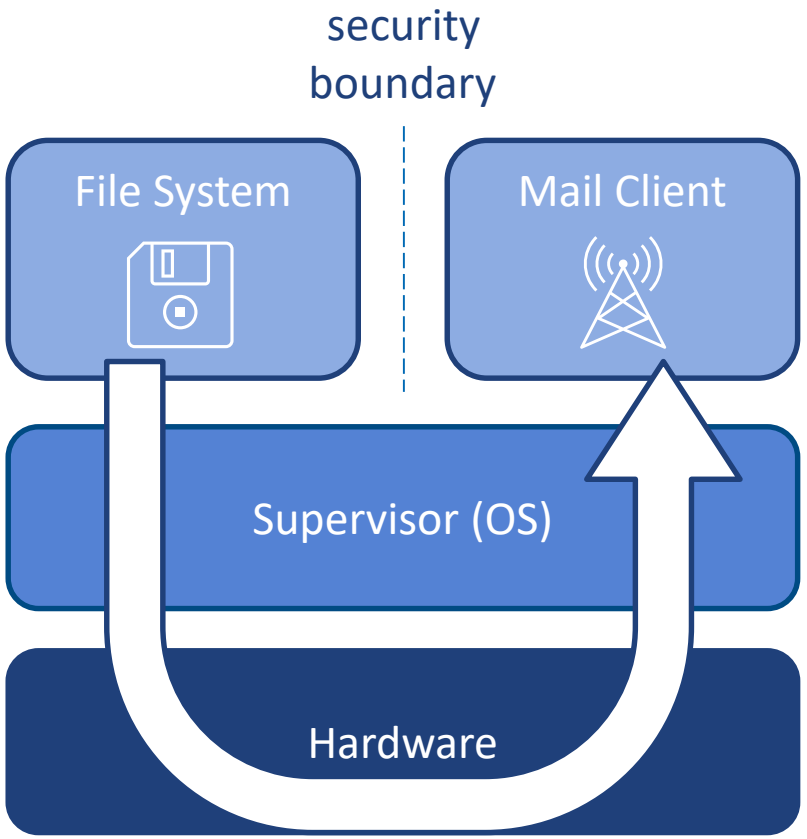
Outline

1. Covert channels?
2. Measure
3. Mitigate
4. Costs
5. Conclusion

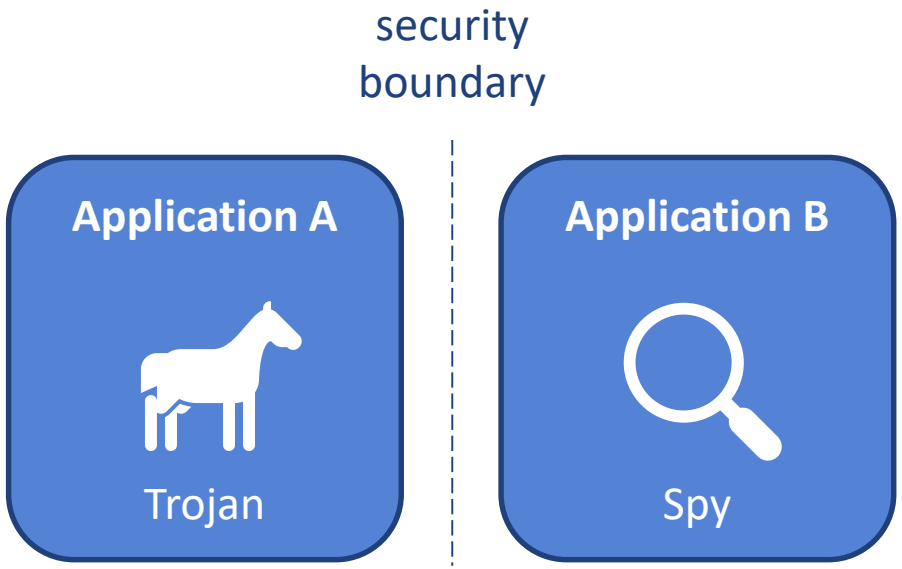
Covert Channel



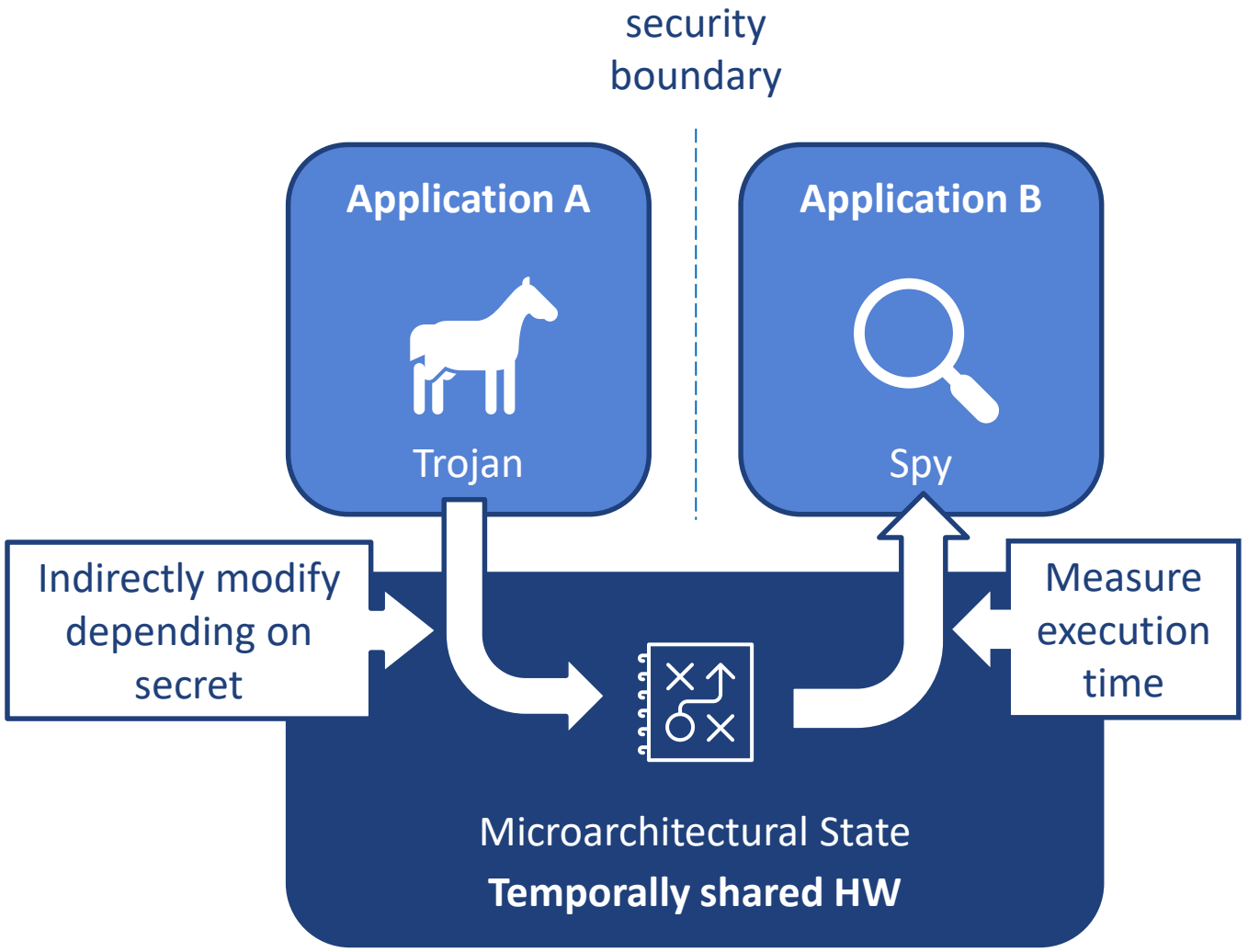
Covert Channel



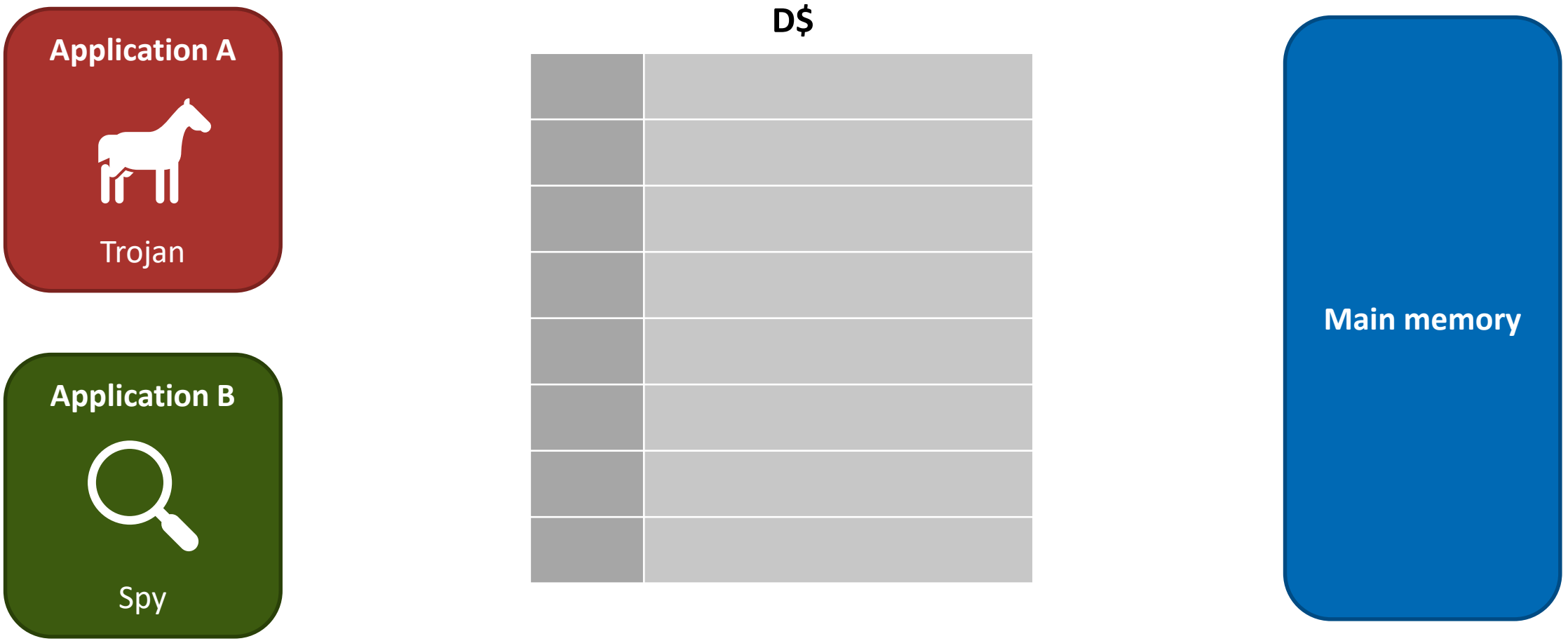
Microarchitectural Timing Channel



Microarchitectural Timing Channel

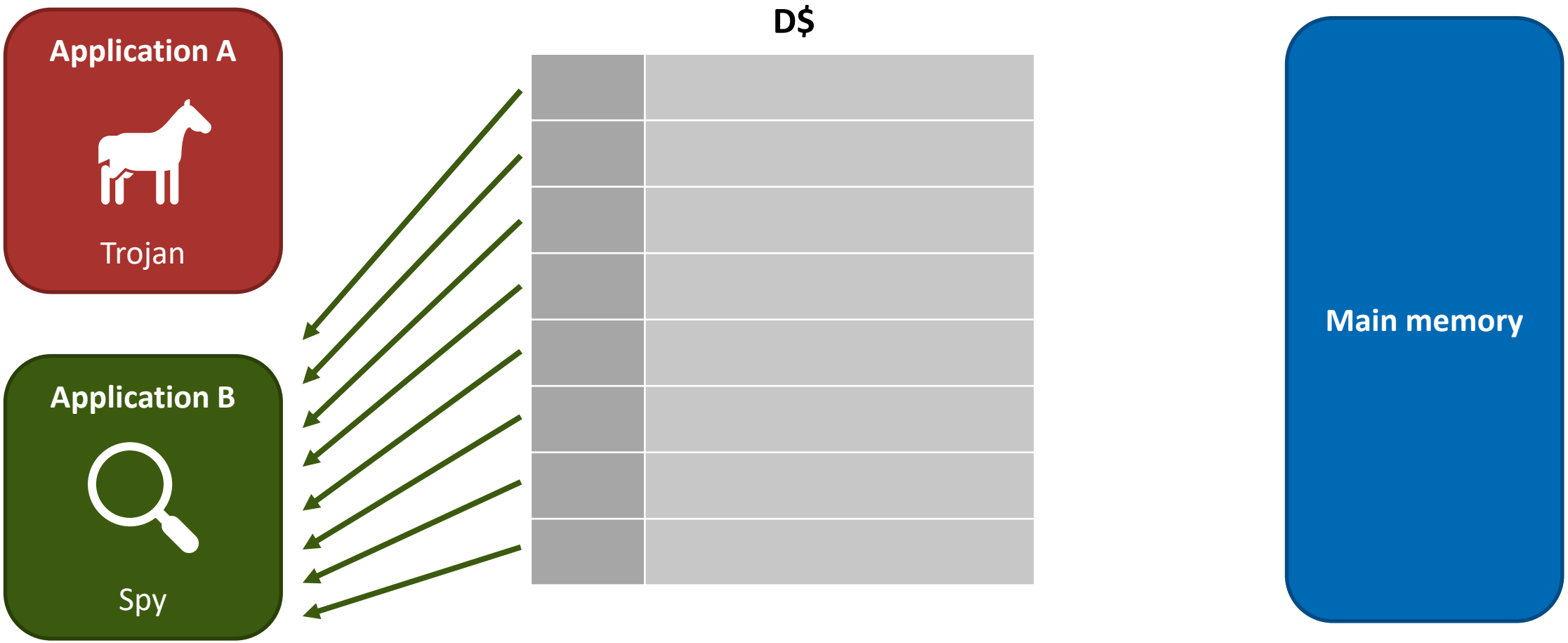


Example: D$ Timing Channel



(1) Spy: **Prime** (2) OS: **Cont. sw.** (3) Trojan: **Encode *s*** (4) OS: **Cont. sw.** (5) Spy: **Probe**

Example: D$ Timing Channel – Prime



(1) Spy: **Prime**

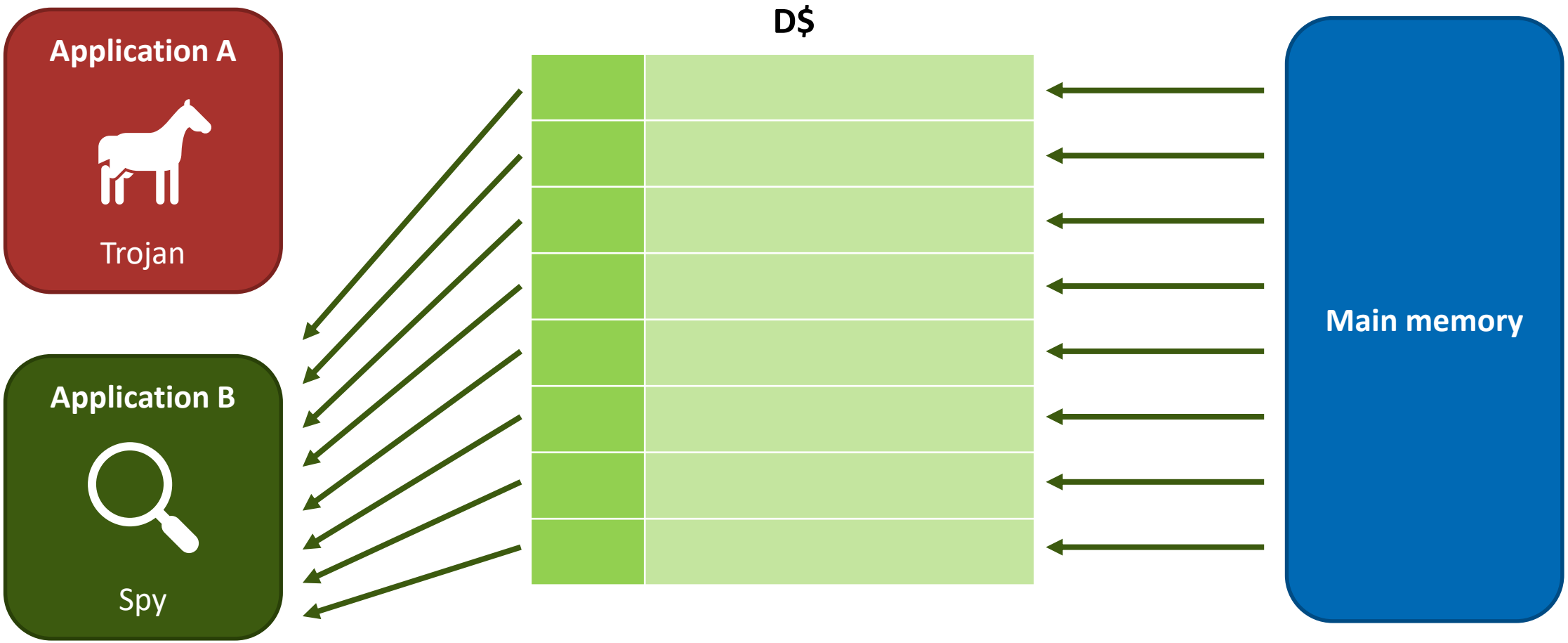
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

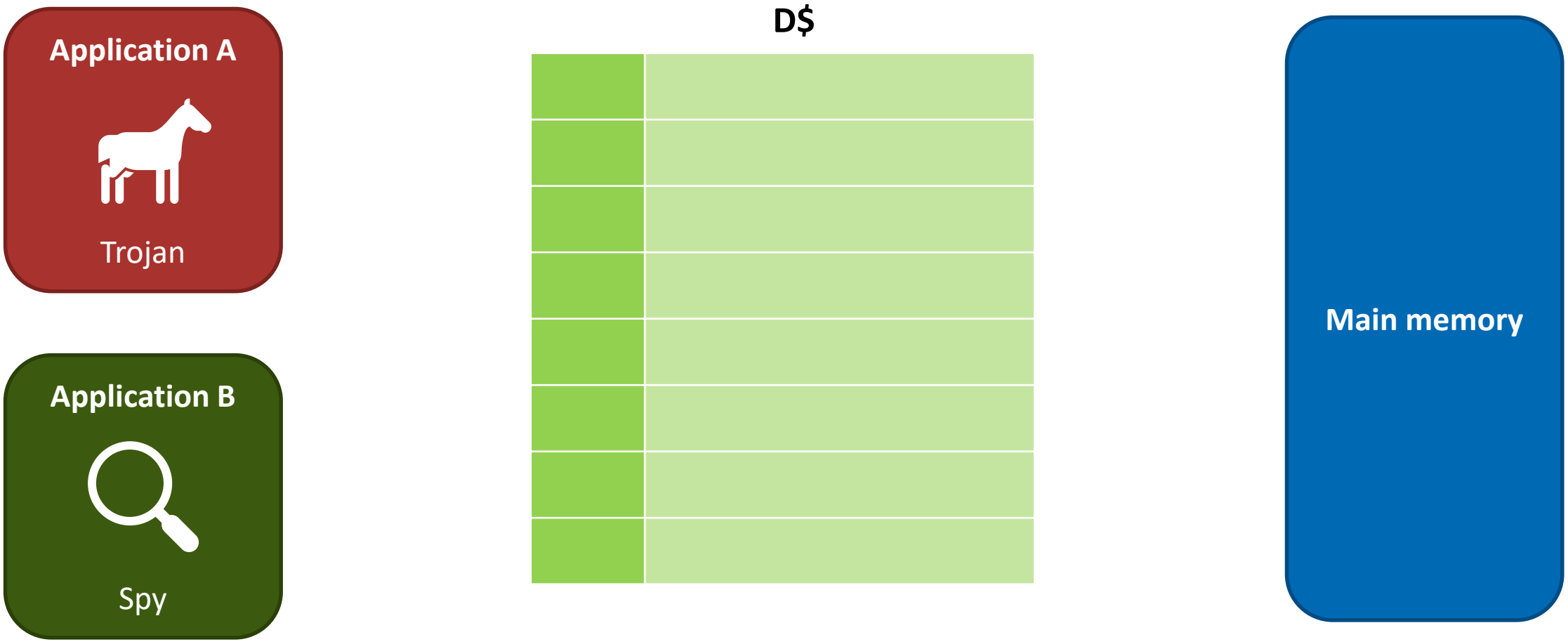
(4) OS: **Cont. sw.**

(5) Spy: **Probe**

Example: D$ Timing Channel – Prime



Example: D$ Timing Channel – Context switch



(1) Spy: **Prime**

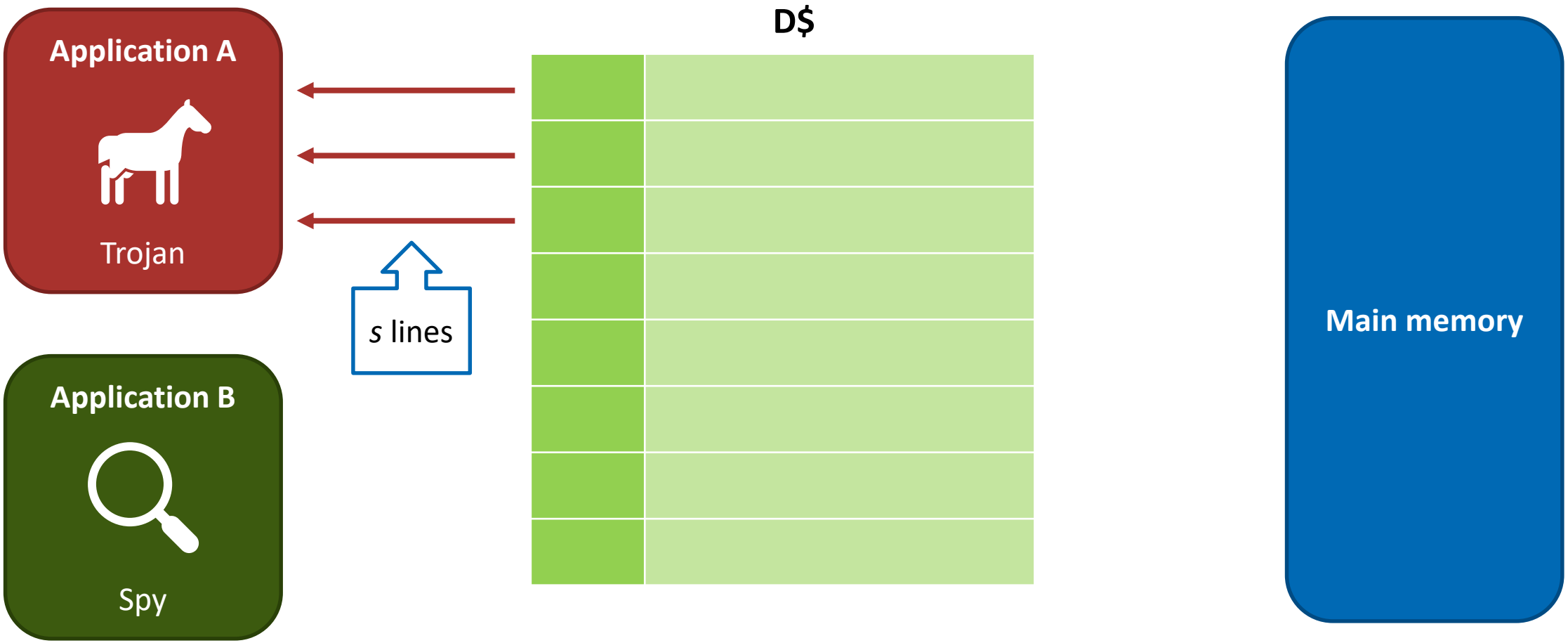
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

(4) OS: **Cont. sw.**

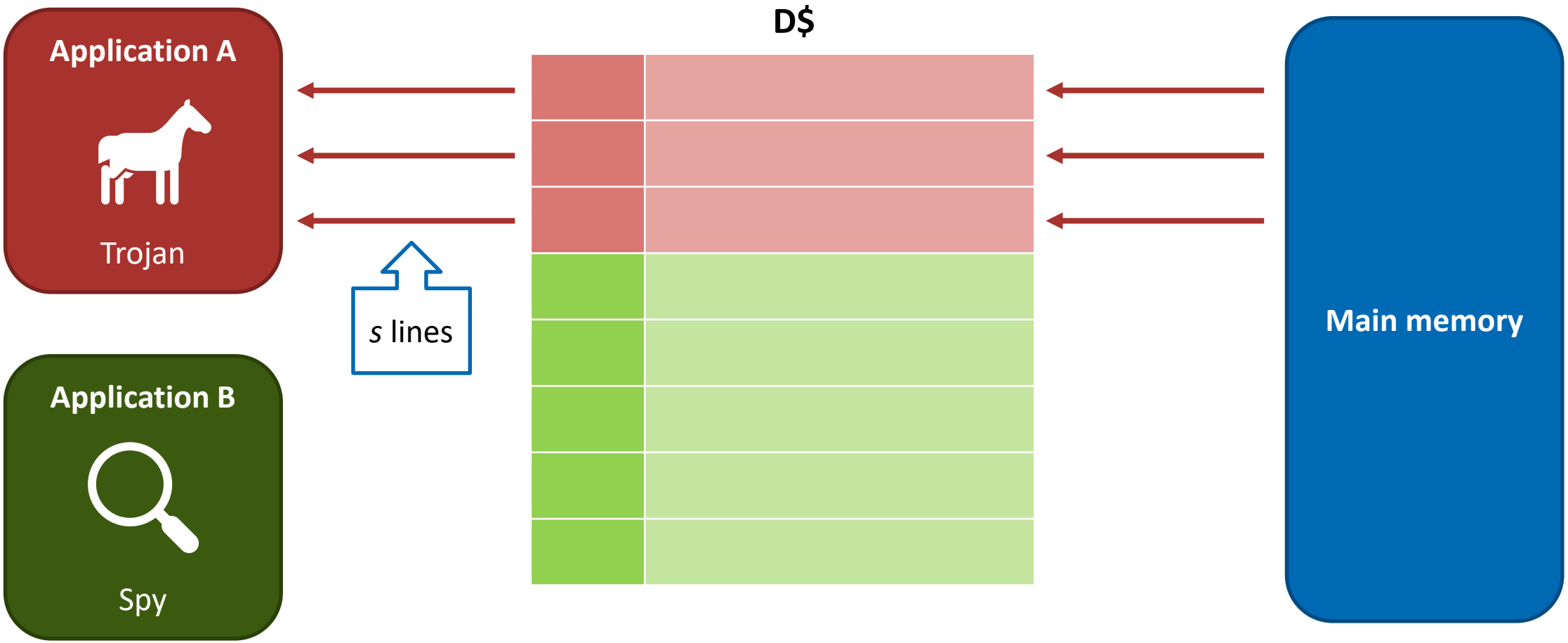
(5) Spy: **Probe**

Example: D$ Timing Channel – Encode *s*



(1) Spy: **Prime** (2) OS: **Cont. sw.** (3) Trojan: **Encode *s*** (4) OS: **Cont. sw.** (5) Spy: **Probe**

Example: D$ Timing Channel – Encode *s*



(1) Spy: **Prime**

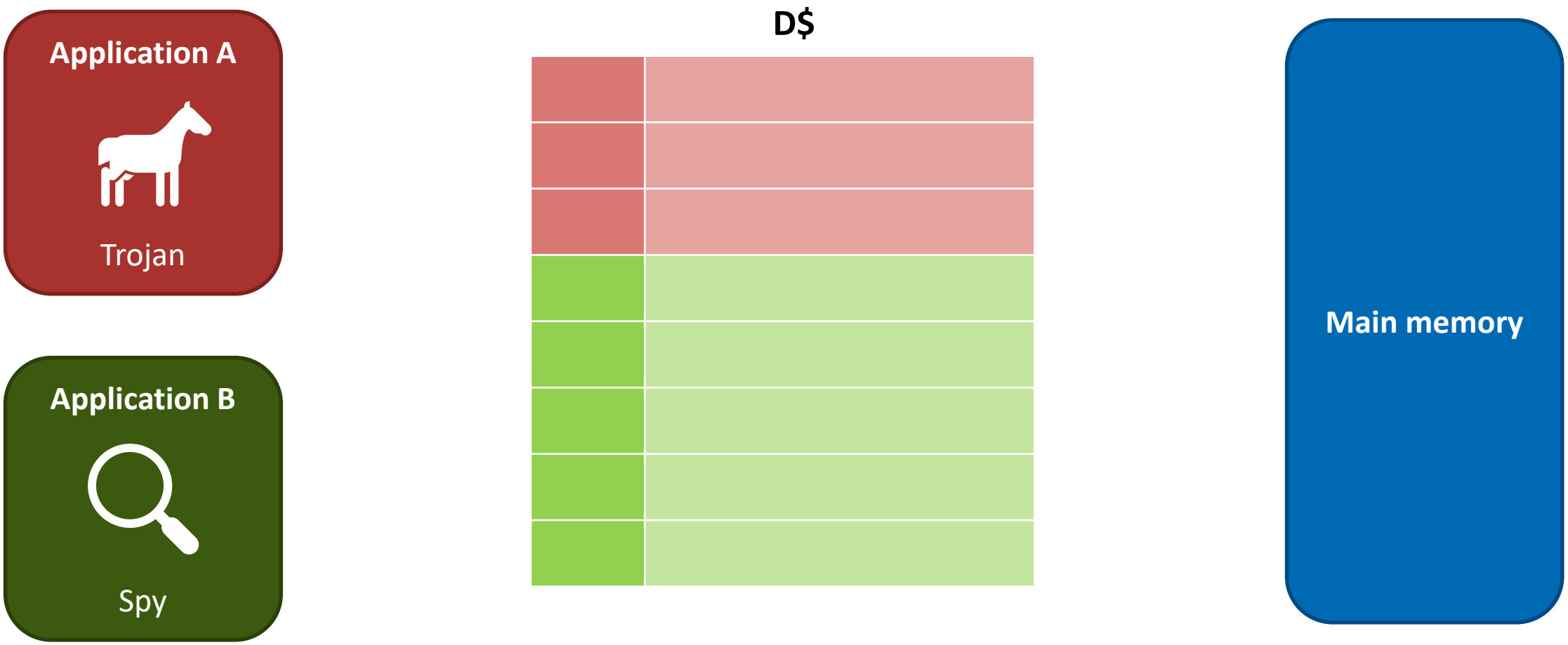
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

(4) OS: **Cont. sw.**

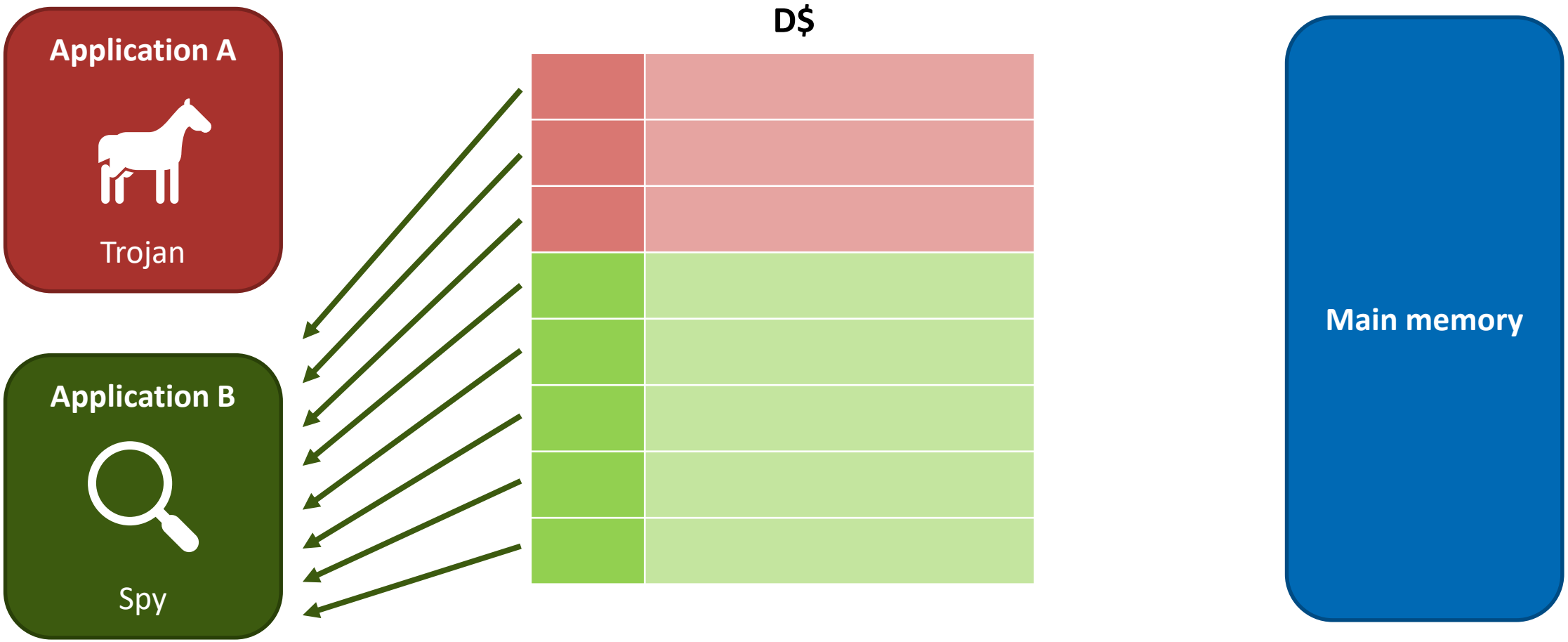
(5) Spy: **Probe**

Example: D$ Timing Channel – Context Switch

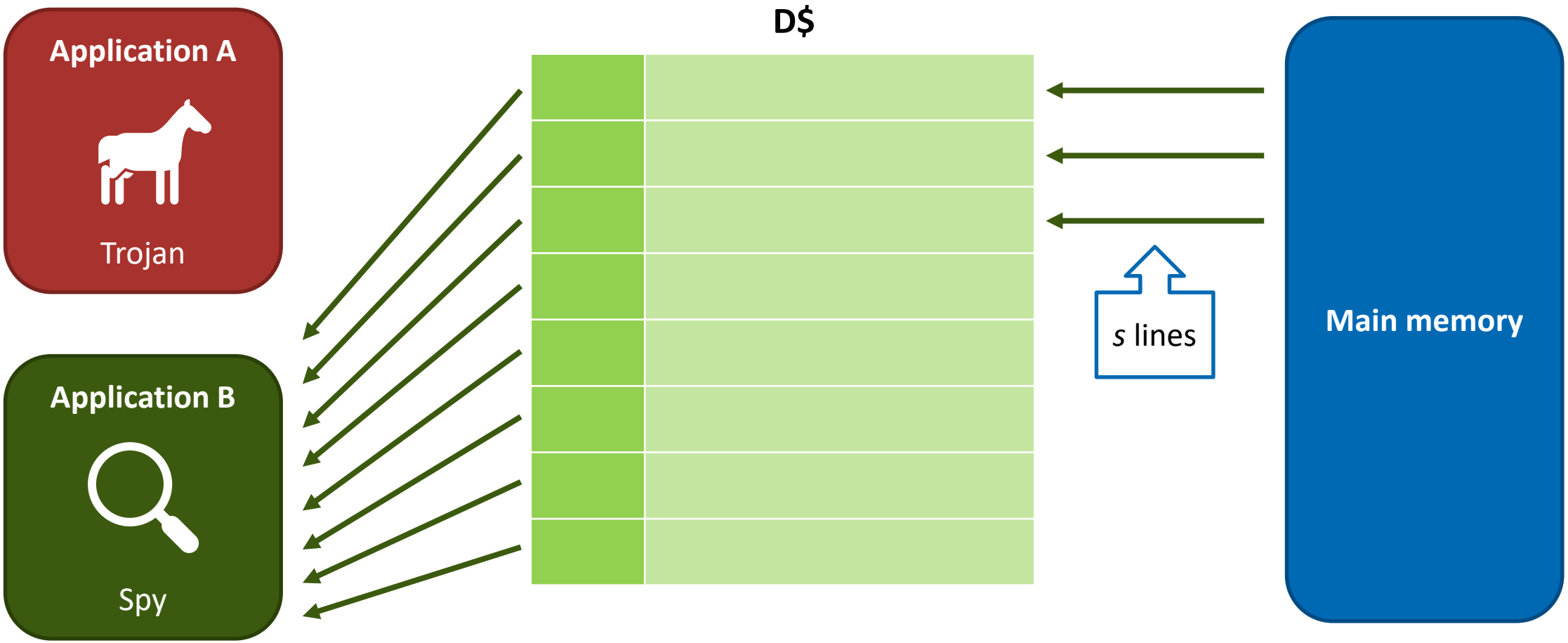


(1) Spy: **Prime** (2) OS: **Cont. sw.** (3) Trojan: **Encode *s*** (4) OS: **Cont. sw.** (5) Spy: **Probe**

Example: D$ Timing Channel – Probe



Example: D$ Timing Channel – Probe



(1) Spy: **Prime**

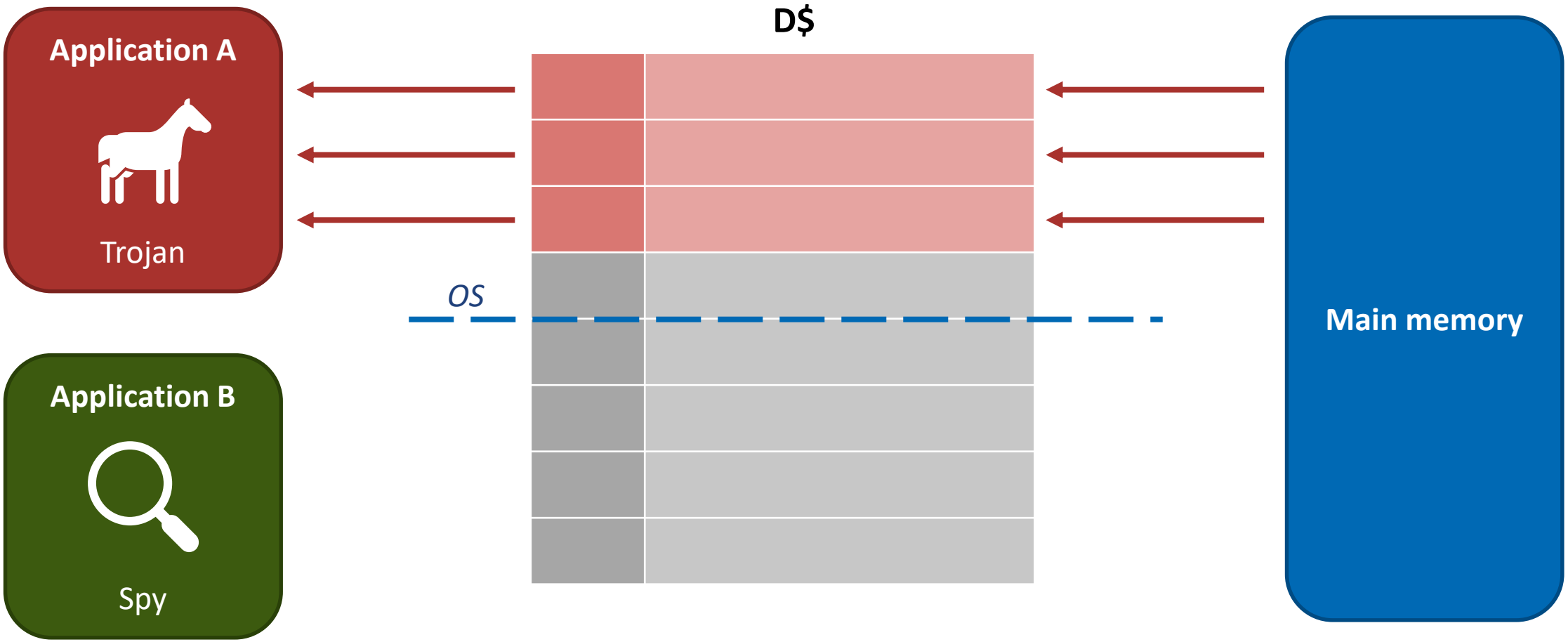
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

(4) OS: **Cont. sw.**

(5) Spy: **Probe**

Spatial Partitioning



(1) Spy: **Prime**

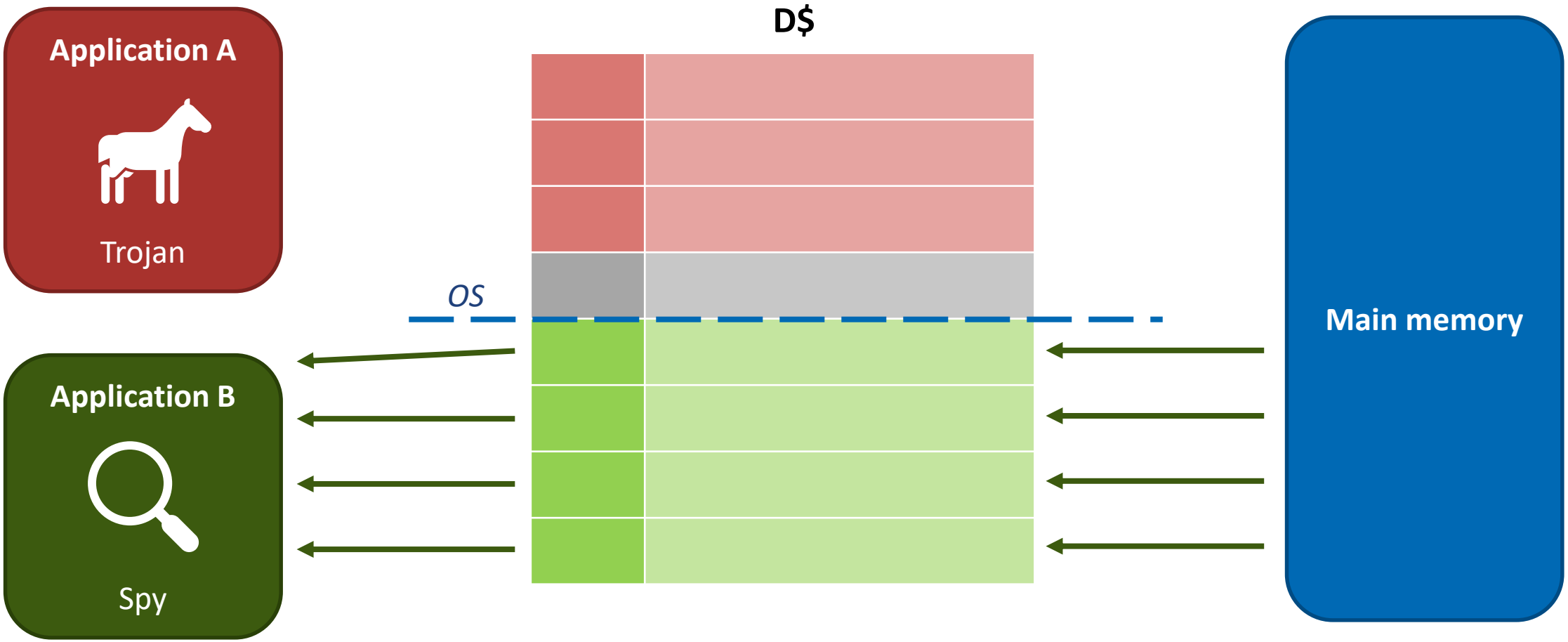
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

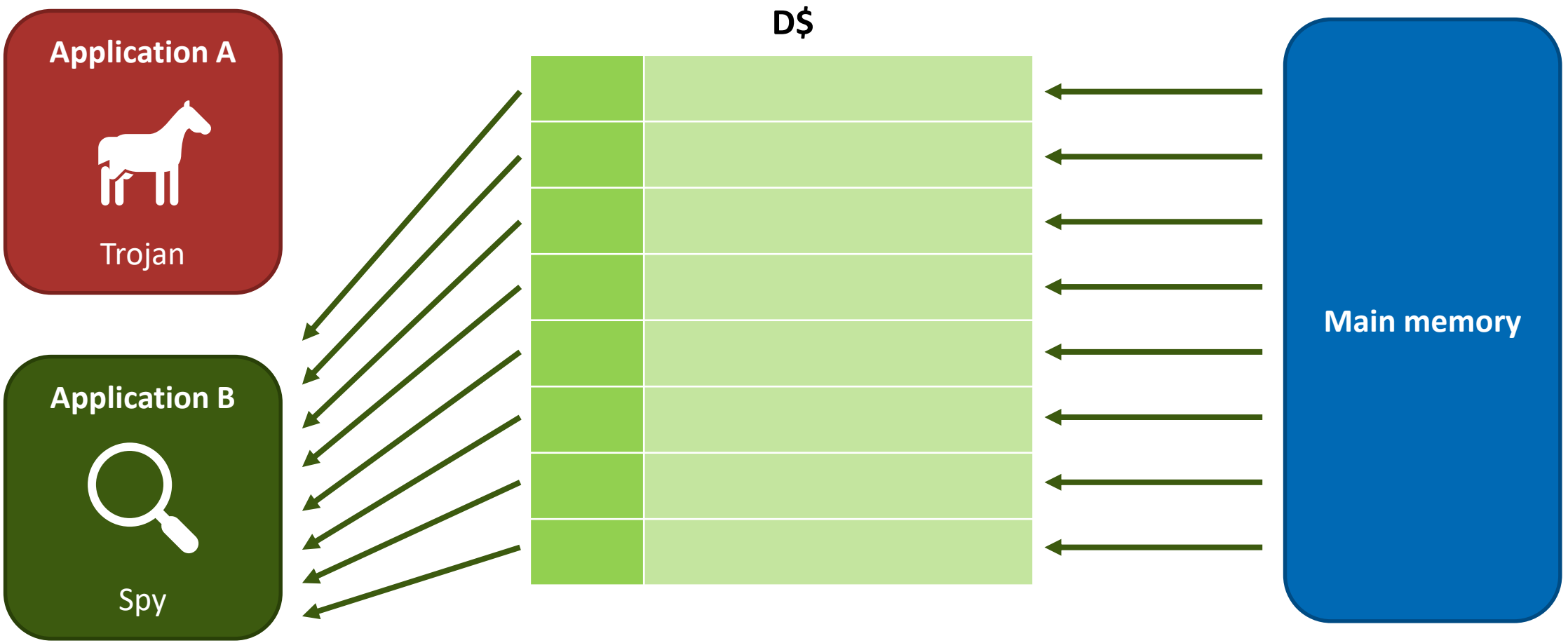
(4) OS: **Cont. sw.**

(5) Spy: **Probe**

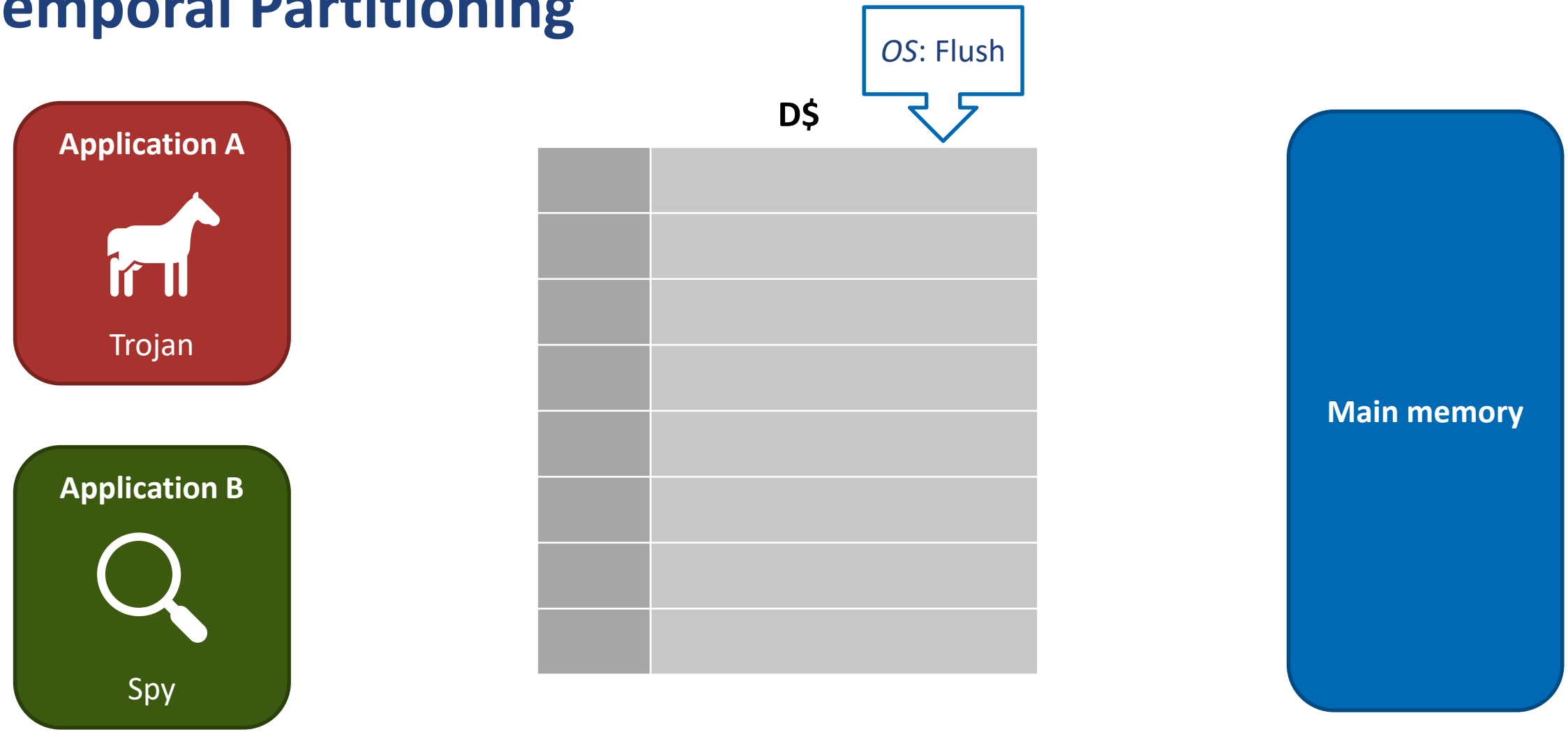
Spatial Partitioning



Temporal Partitioning



Temporal Partitioning



(1) Spy: **Prime**

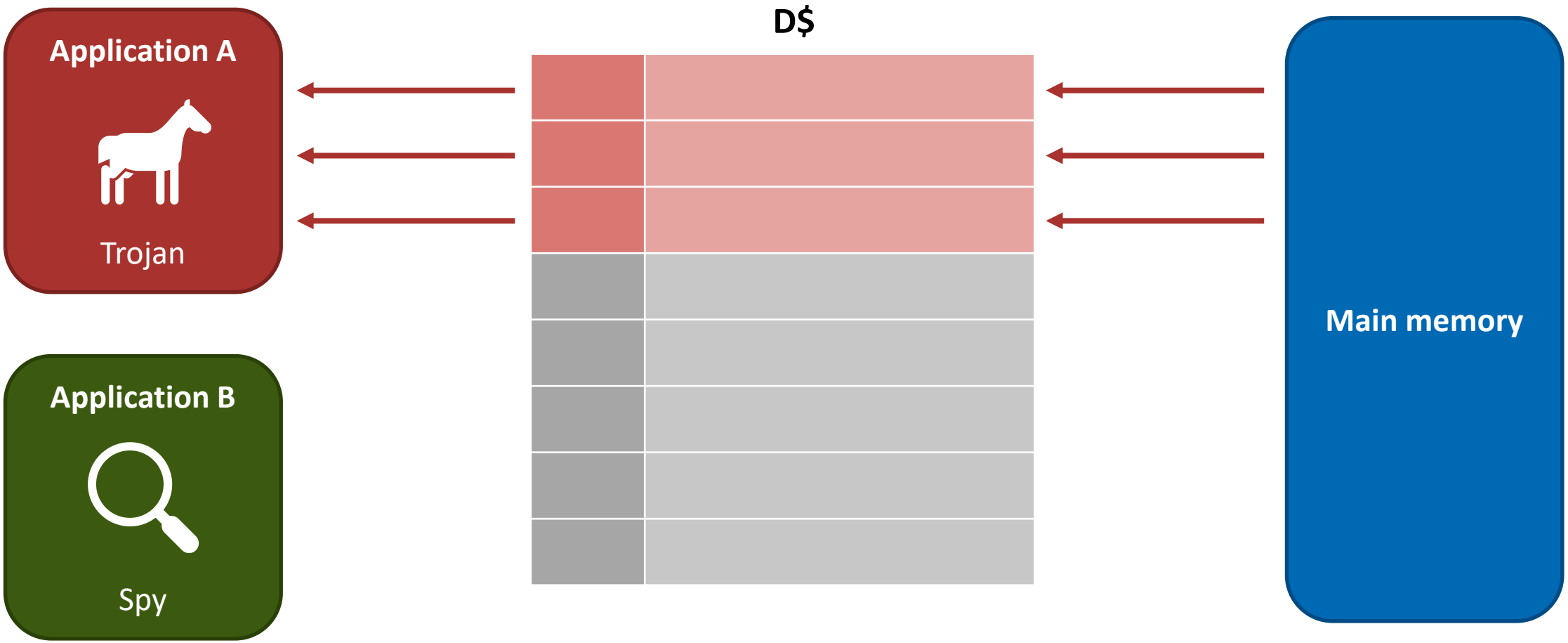
(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

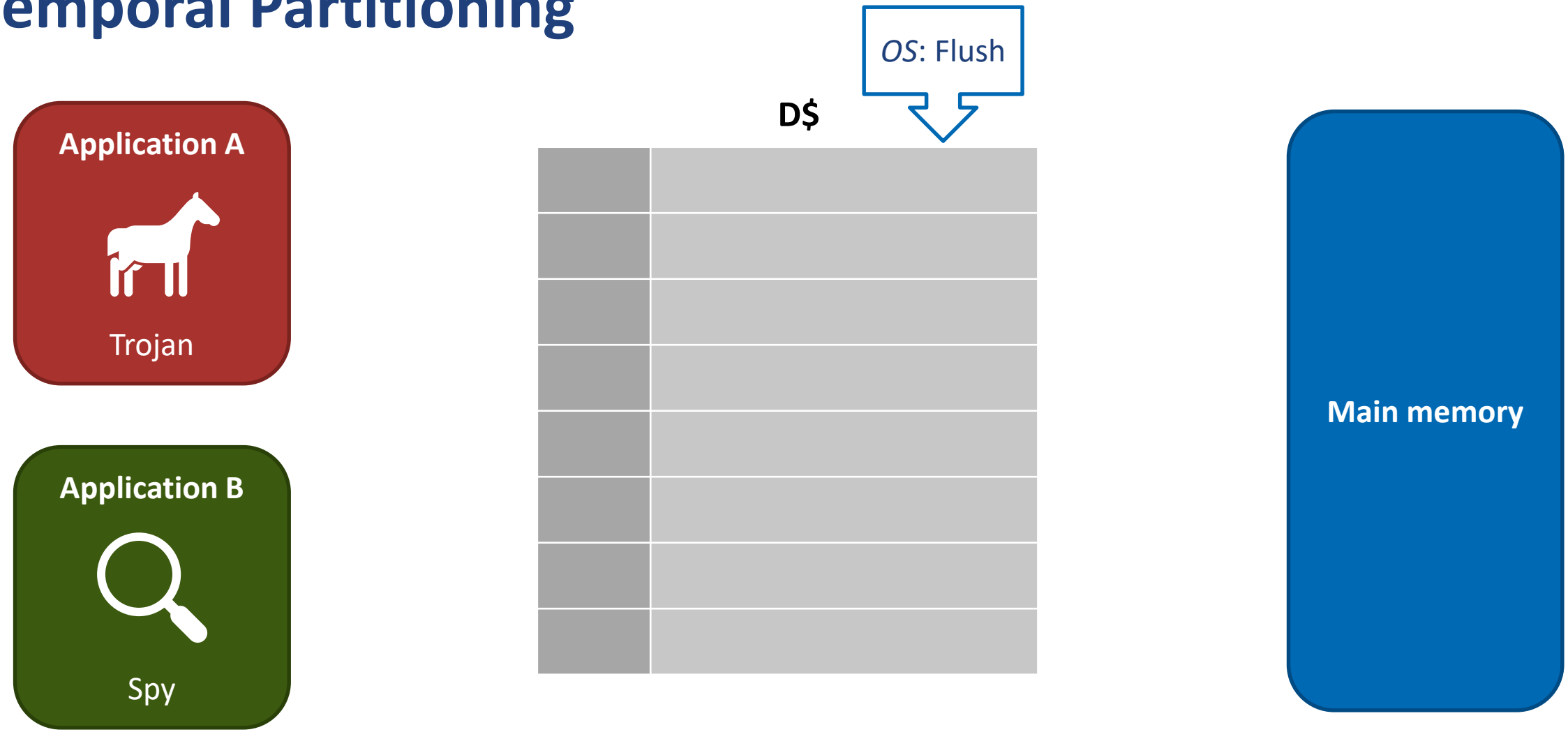
(4) OS: **Cont. sw.**

(5) Spy: **Probe**

Temporal Partitioning

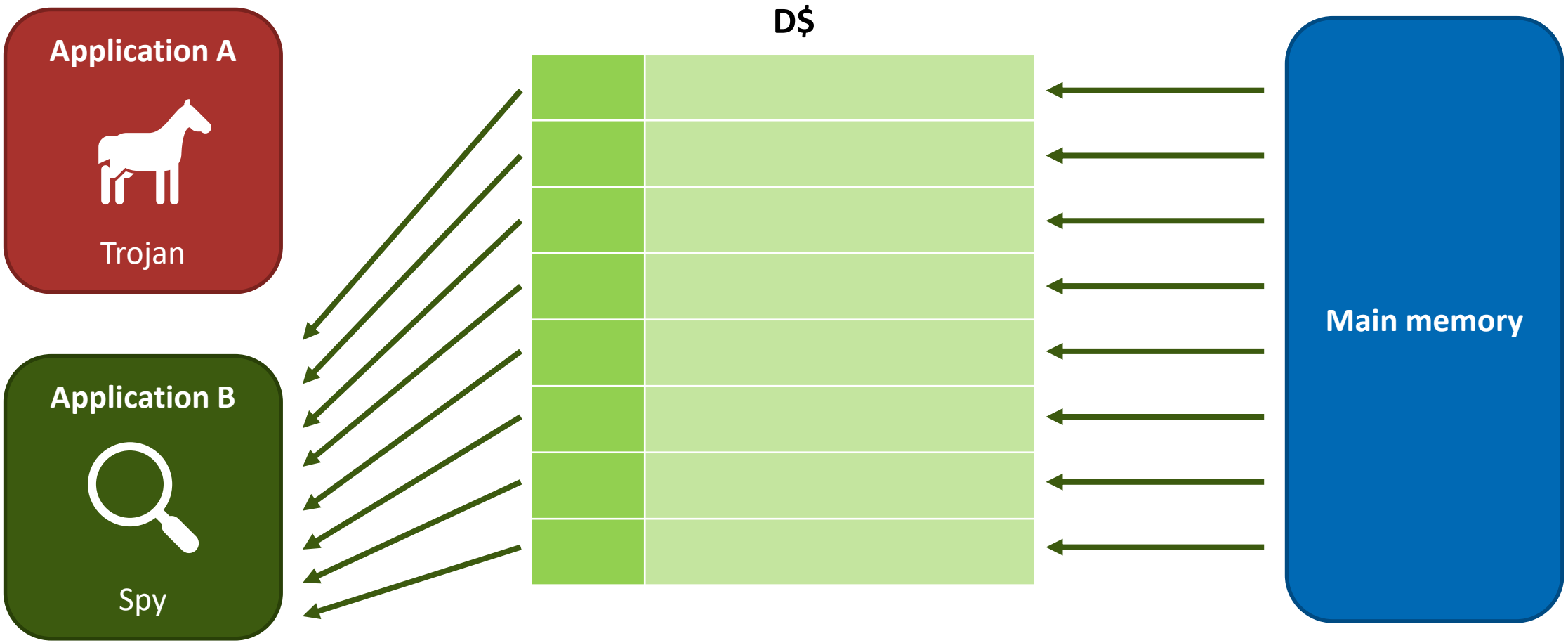


Temporal Partitioning

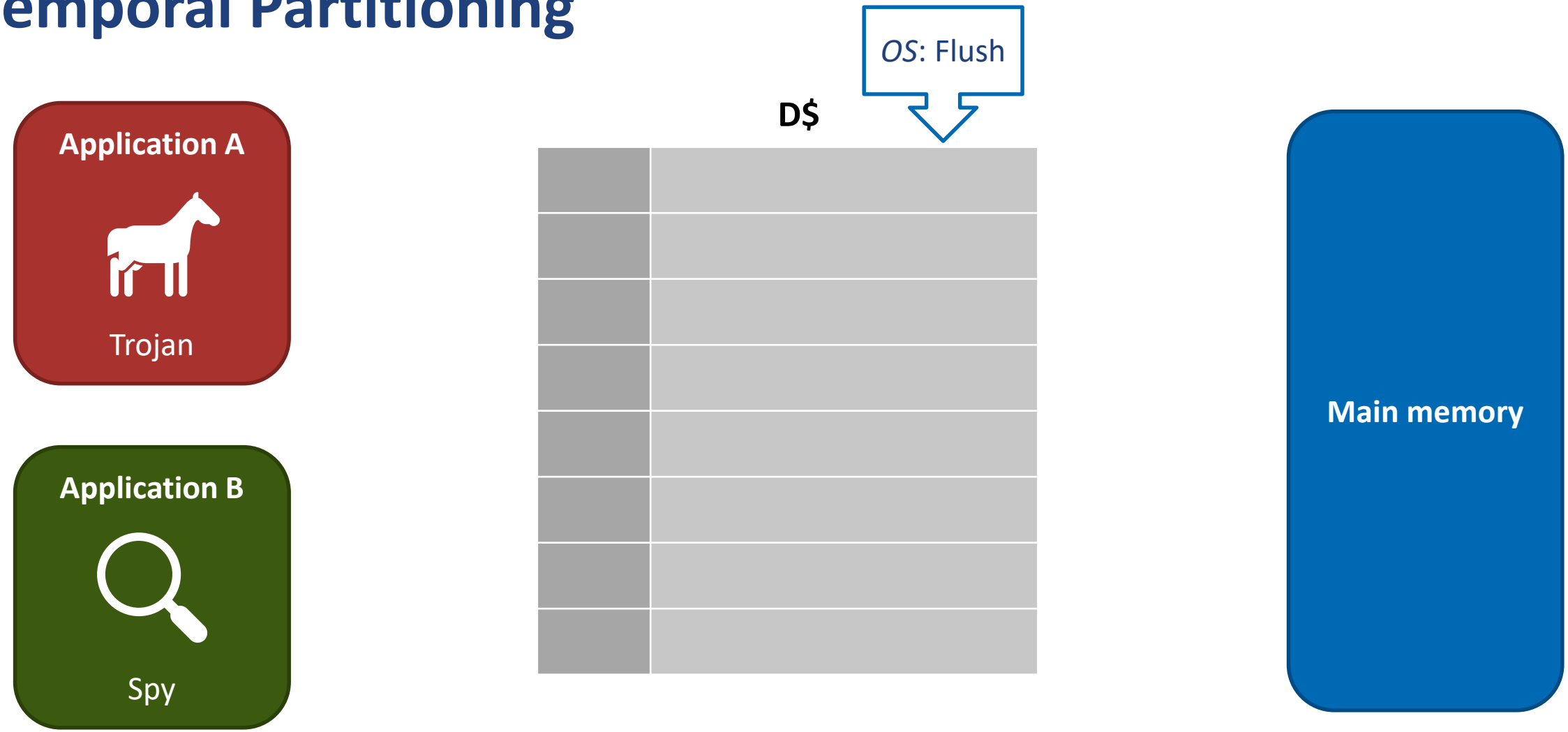


(1) Spy: **Prime** (2) OS: **Cont. sw.** (3) Trojan: **Encode *s*** (4) OS: **Cont. sw.** (5) Spy: **Probe**

Temporal Partitioning

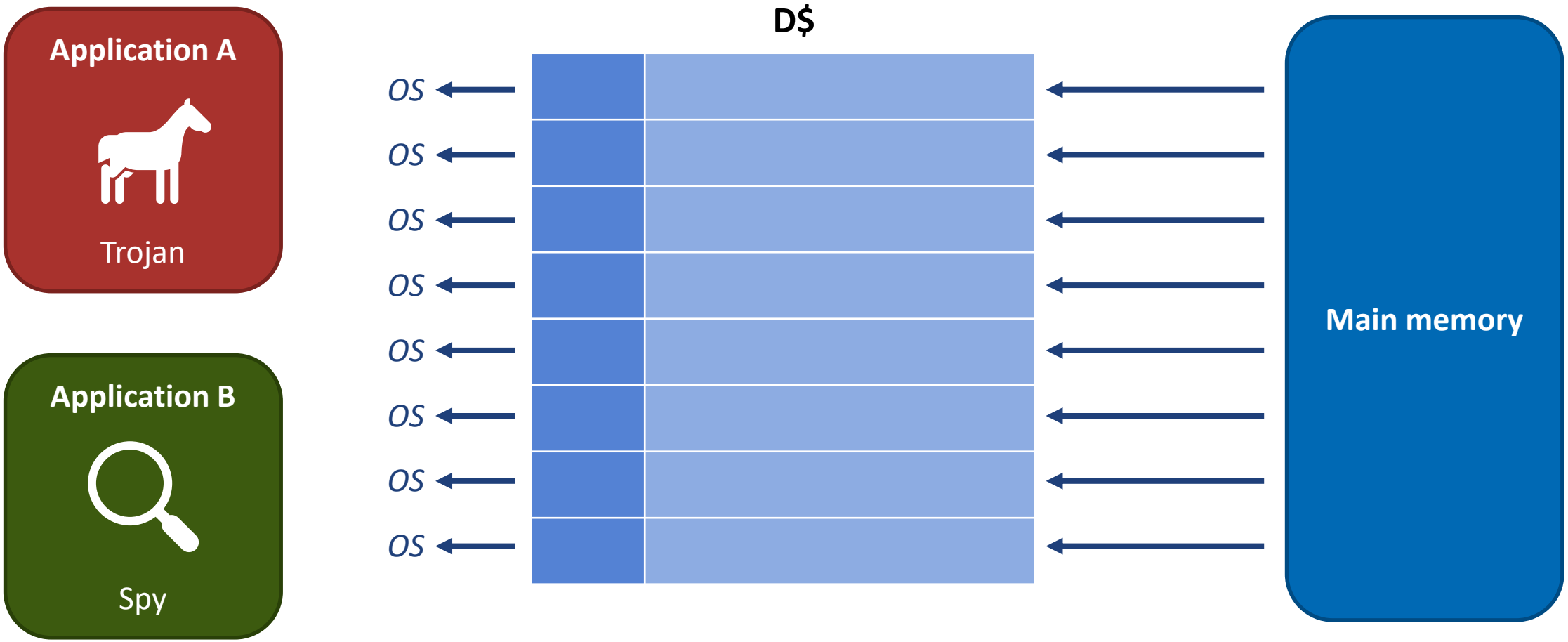


Temporal Partitioning



(1) Spy: **Prime** (2) OS: **Cont. sw.** (3) Trojan: **Encode *s*** (4) OS: **Cont. sw.** (5) Spy: **Probe**

Flush: SW Approach



(1) Spy: **Prime**

(2) OS: **Cont. sw.**

(3) Trojan: **Encode *s***

(4) OS: **Cont. sw.**

(5) Spy: **Probe**

Evaluation Platform

Hardware platform

Ariane RV64GC core [4]

- FPGA (Genesys 2) @50MHz
- Add timer peripheral and 512KiB LLC [3]
- Write-through 32KiB L1D$ and 16KiB L1I$
- 16-entry DTLB, 16-entry BTB, 64-entry BHT

Evaluation Platform

Supervisor

seL4 microkernel [5]

- Formally verified μKernel by Data61
- Focus on security
- Port to Ariane
- Enable cache colouring of LLC

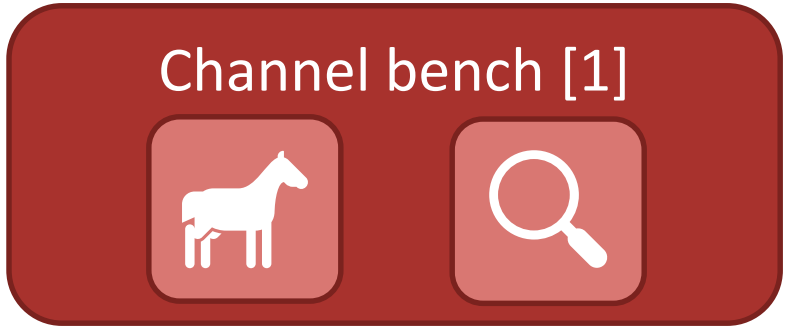
Hardware platform

Ariane RV64GC core [4]

- FPGA (Genesys 2) @50MHz
- Add timer peripheral and 512KiB LLC [3]
- Write-through 32KiB L1D$ and 16KiB L1I$
- 16-entry DTLB, 16-entry BTB, 64-entry BHT

Evaluation Platform

Application



- Measure covert channels on ARM/x86
- Port to RISC-V
- Tailor attacks to Ariane‘s µArch

Supervisor

- Formally verified µKernel by Data61
- Focus on security
- Port to Ariane
- Enable cache colouring of LLC

Hardware platform

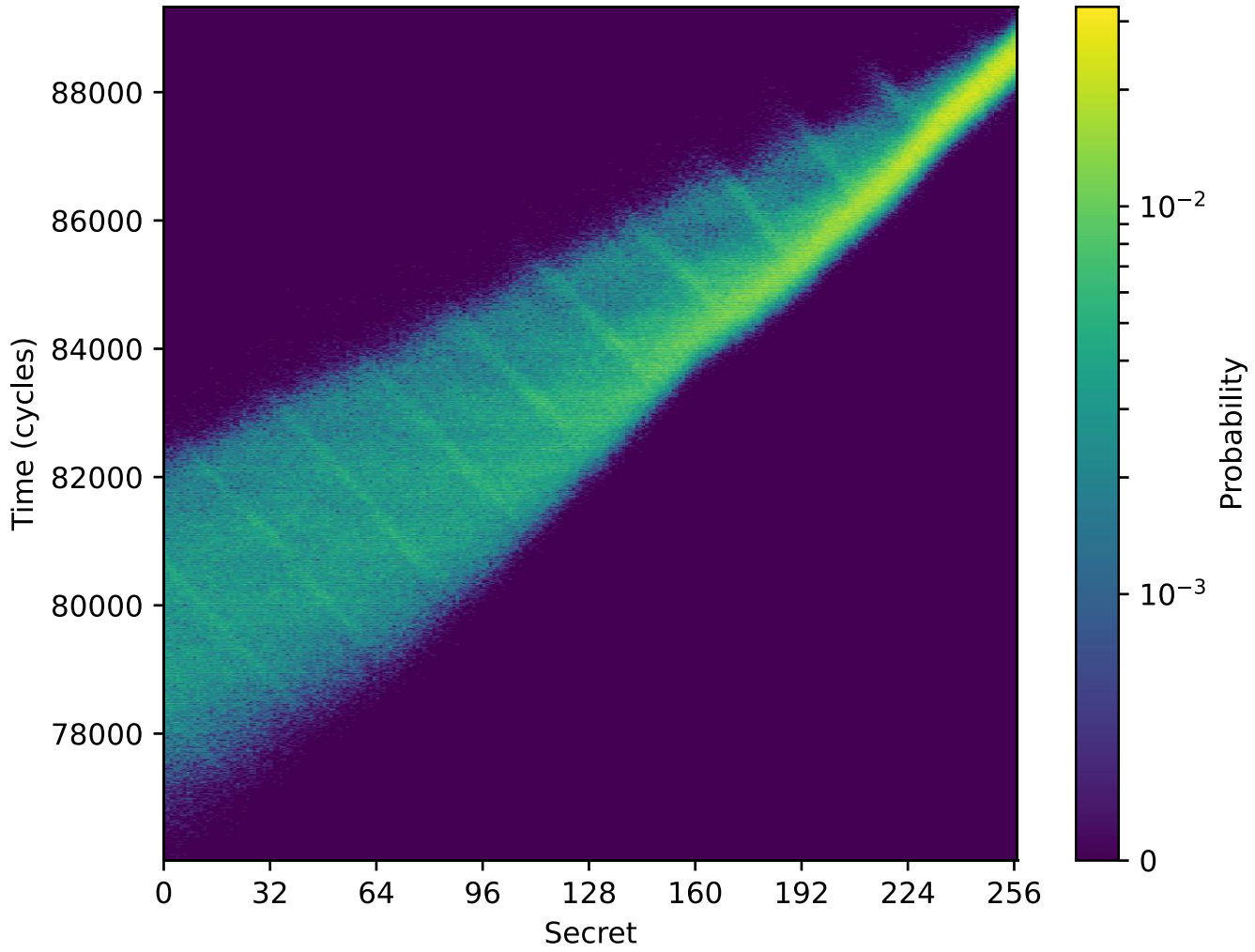
- FPGA (Genesys 2) @50MHz
- Add timer peripheral and 512KiB LLC [3]
- Write-through 32KiB L1D$ and 16KiB L1I$
- 16-entry DTLB, 16-entry BTB, 64-entry BHT

Channel Bench Output: L1 D$

s_0	107	t_0	83316
s_1	11	t_1	80209
s_2	112	t_2	82069
s_3	235	t_3	88152
s_4	246	t_4	88856
s_5	152	t_5	86627

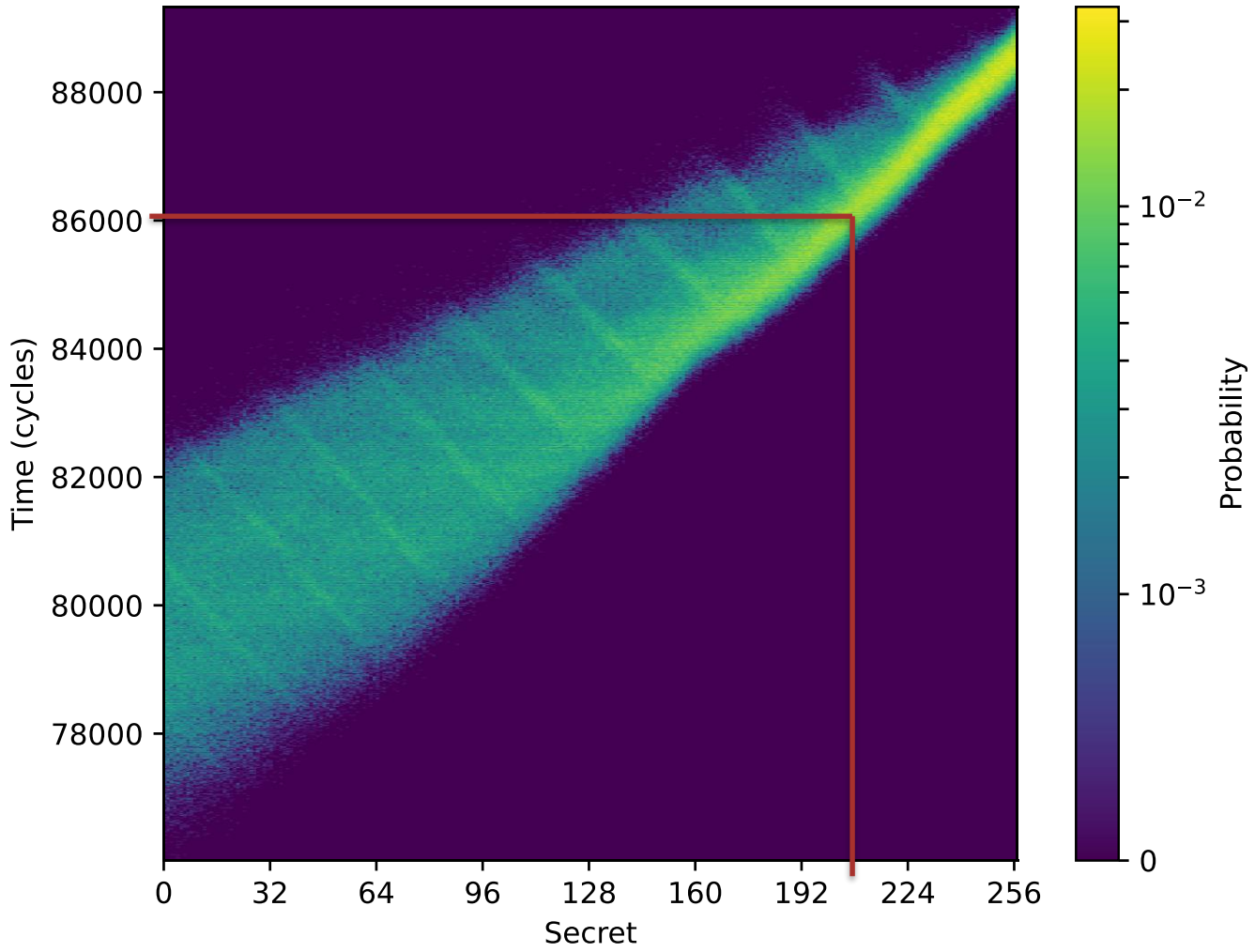
Channel Matrix: L1 D$

$N = 10^6$



Channel Matrix: L1 D$

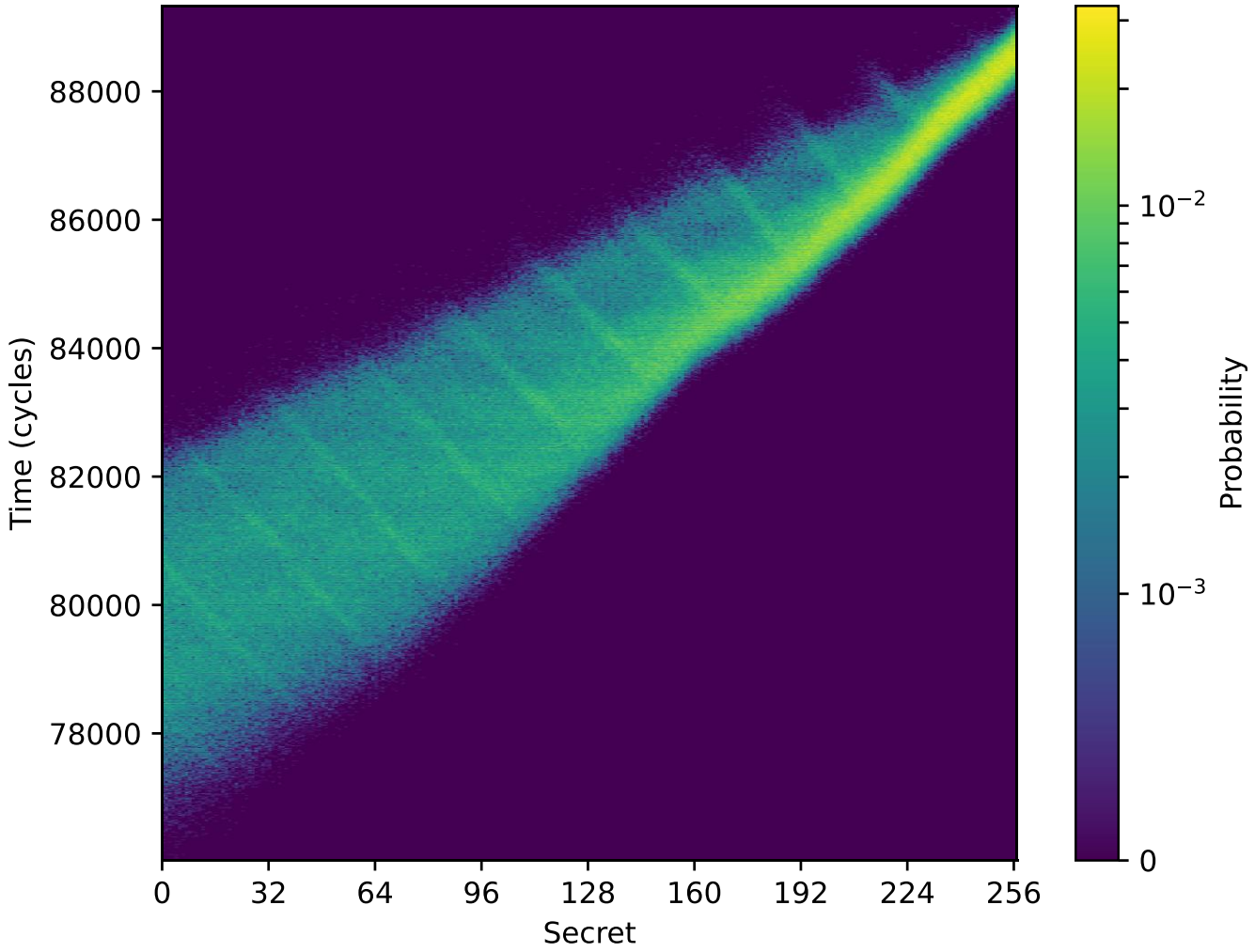
$N = 10^6$



Channel Matrix: L1 D$

$N = 10^6$

$M = 1667.3$ mb



Channel Bench Output: L1 D$

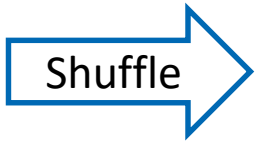
s_0	107	t_0	83316
s_1	11	t_1	80209
s_2	112	t_2	82069
s_3	235	t_3	88152
s_4	246	t_4	88856
s_5	152	t_5	86627

M

Channel Bench Output: L1 D$

s_0	107	t_0	83316
s_1	11	t_1	80209
s_2	112	t_2	82069
s_3	235	t_3	88152
s_4	246	t_4	88856
s_5	152	t_5	86627

M



s_0	107	t_3	88152
s_1	11	t_5	86627
s_2	112	t_1	80209
s_3	235	t_4	88856
s_4	246	t_0	83316
s_5	152	t_2	82069

M_0^0

Channel Bench Output: L1 D$

s_0	107	t_0	83316
s_1	11	t_1	80209
s_2	112	t_2	82069
s_3	235	t_3	88152
s_4	246	t_4	88856
s_5	152	t_5	86627

M

Shuffle

s_0	107	t_3	88152
s_1	11	t_5	86627
s_2	112	t_1	80209
s_3	235	t_4	88856
s_4	246	t_0	83316
s_5	152	t_2	82069

M_0^0

Repeat

s_0	t_2
s_1	t_1
s_2	t_0
s_3	t_4
s_4	t_3
s_5	t_5

M_0^1

s_0	t_1
s_1	t_2
s_2	t_0
s_3	t_3
s_4	t_4
s_5	t_5

M_0^2

s_0	t_5
s_1	t_2
s_2	t_0
s_3	t_1
s_4	t_3
s_5	t_4

M_0^3

s_0	t_5
s_1	t_4
s_2	t_0
s_3	t_3
s_4	t_1
s_5	t_2

M_0^4

M_0: 95% confidence interval of M_0^*

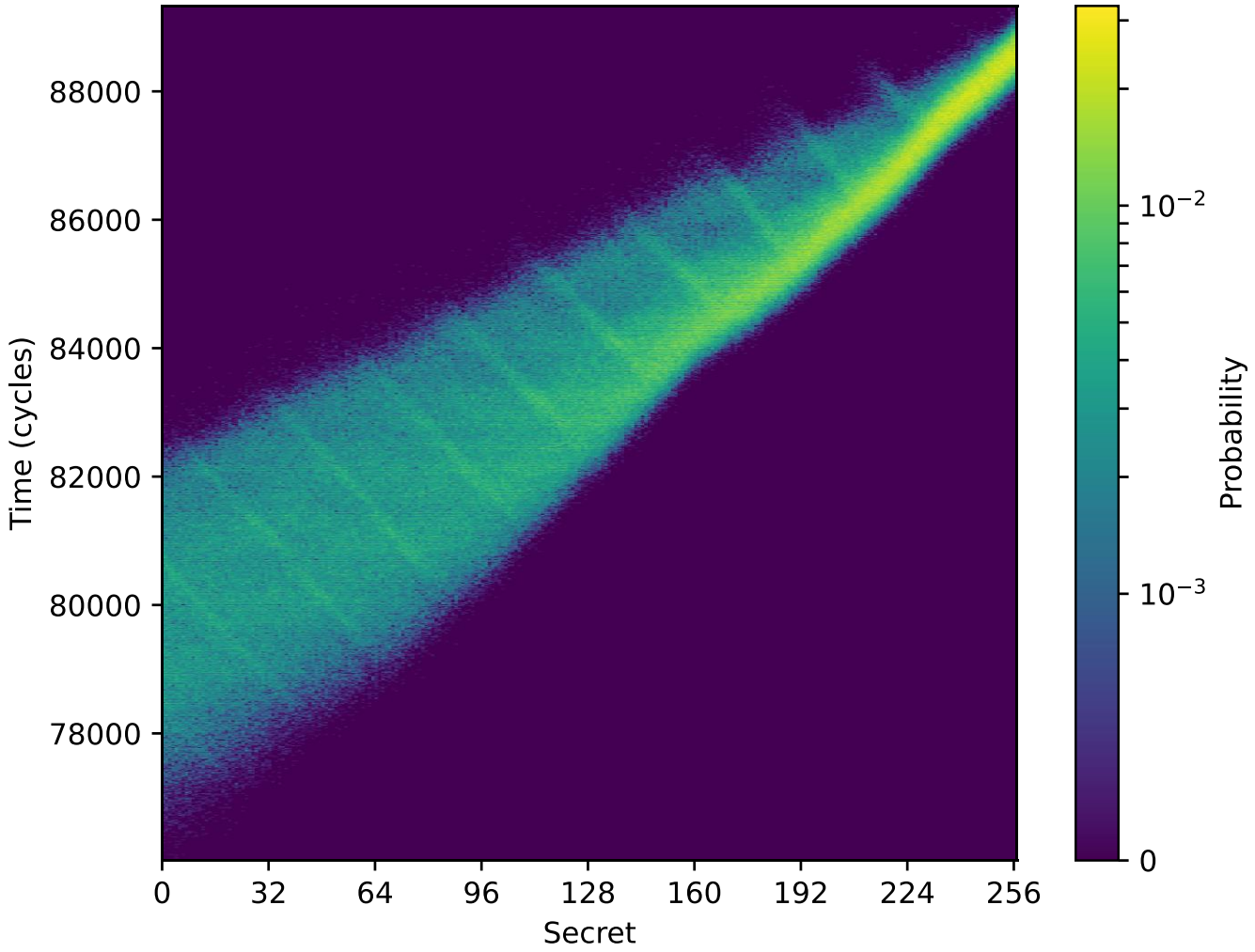
$M > M_0 \Rightarrow$ covert channel!

Channel Matrix: L1 D$

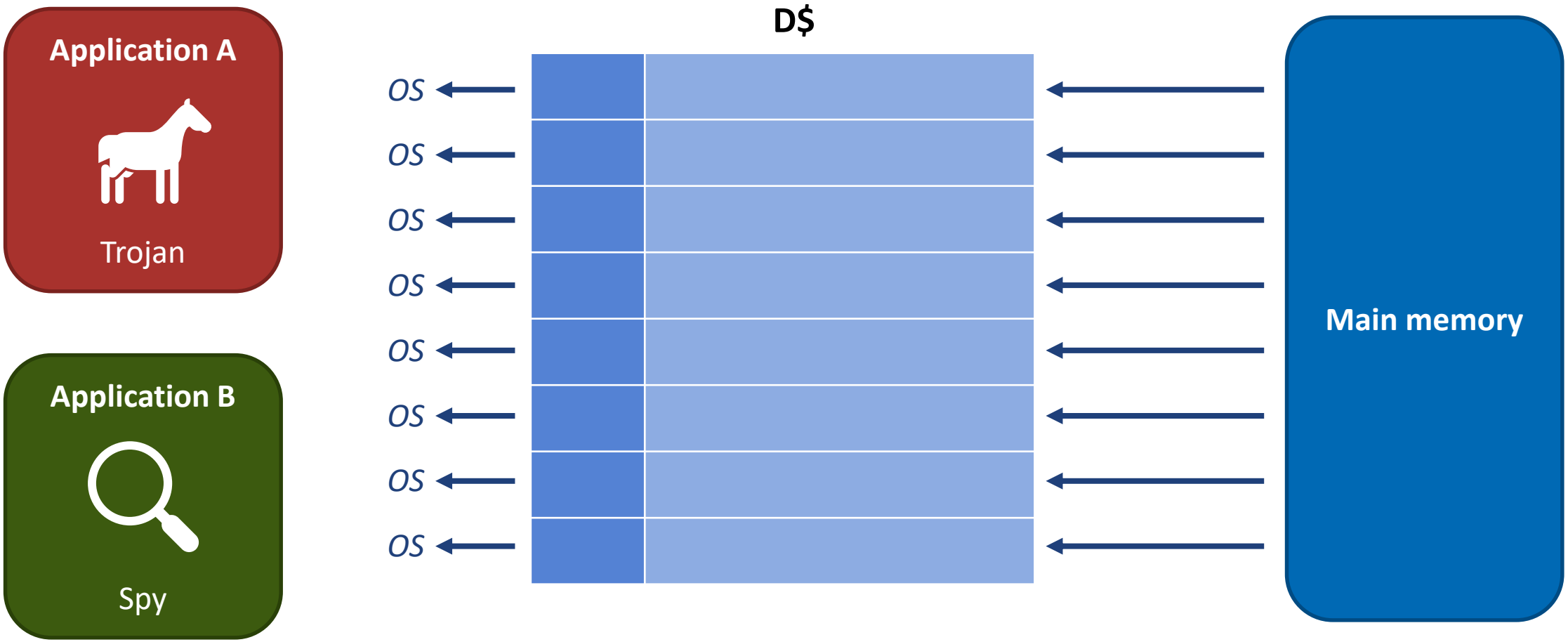
$N = 10^6$

$M = 1667.3$ mb

$M_0 = 0.5$ mb



Flush: SW Approach



(1) Spy: **Prime**

(2) OS: **Cont. sw.**

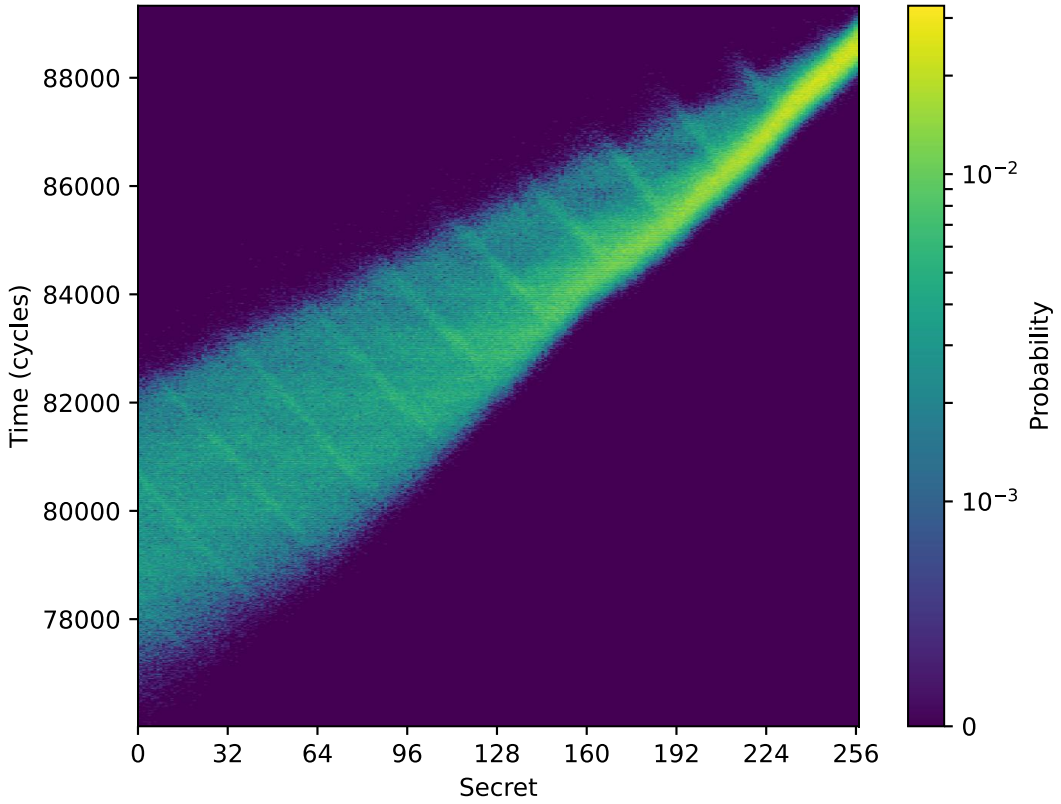
(3) Trojan: **Encode *s***

(4) OS: **Cont. sw.**

(5) Spy: **Probe**

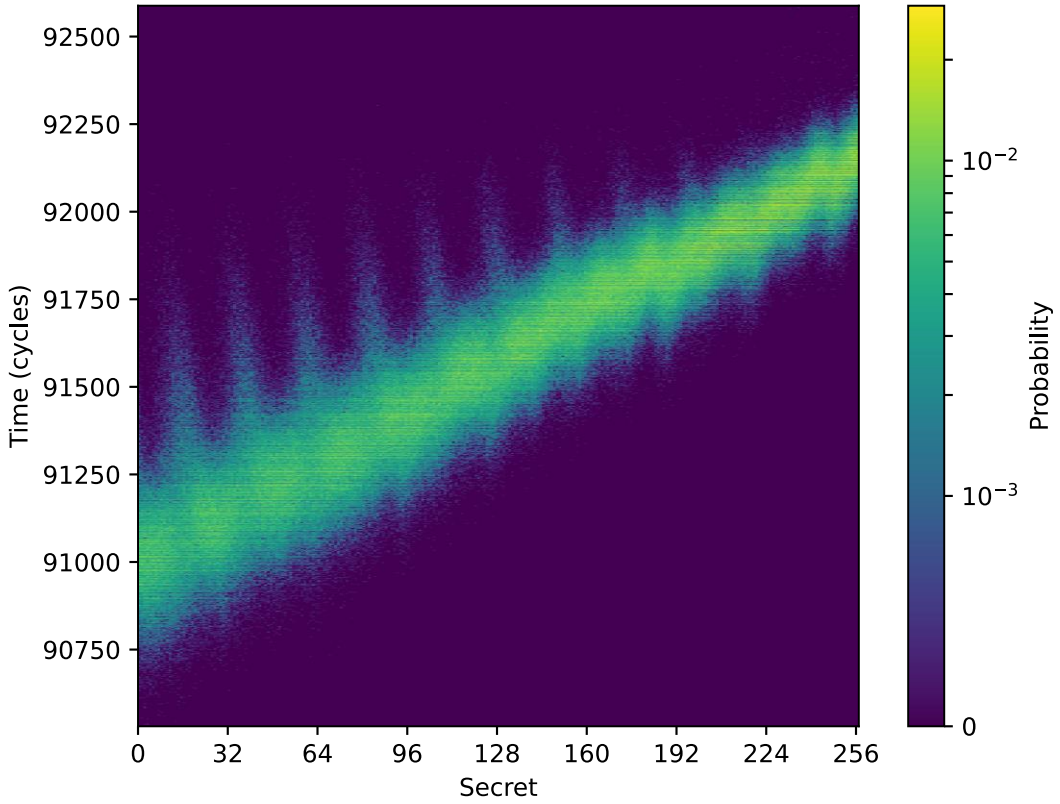
Software Mitigation: L1 D$ Channel

Unmitigated



$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

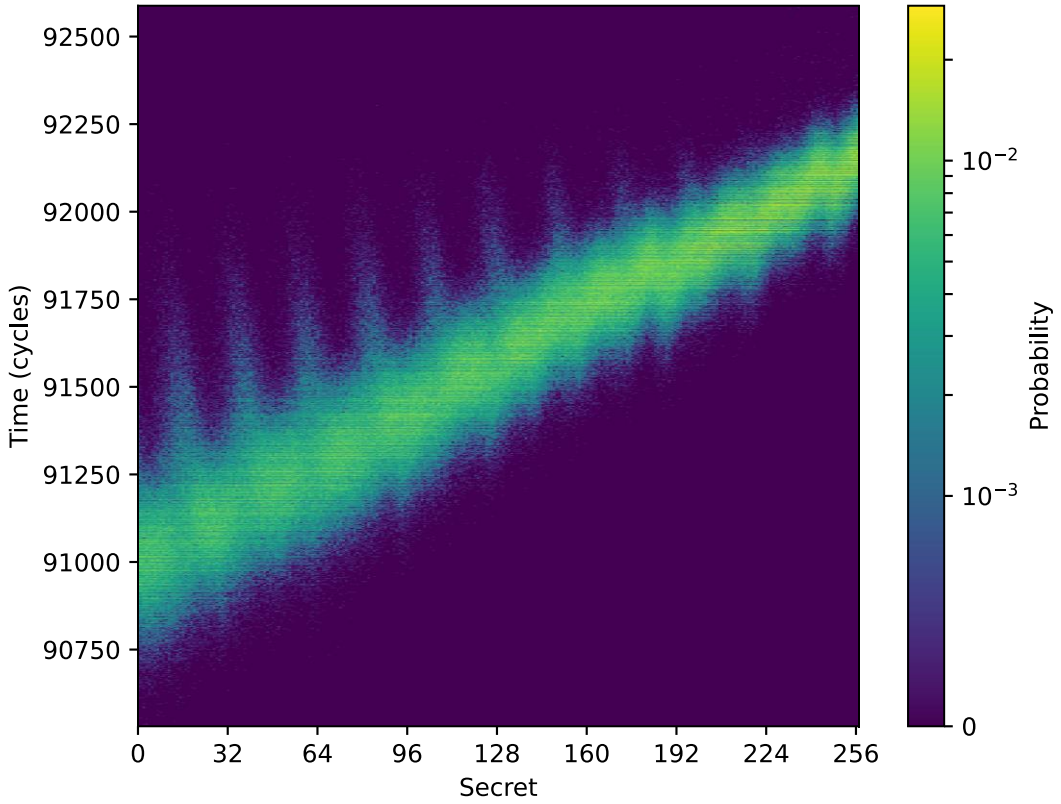
L1 D$ prime on context switch



$N = 10^6$, $M = 1471.5$ mb, $M_0 = 0.6$ mb

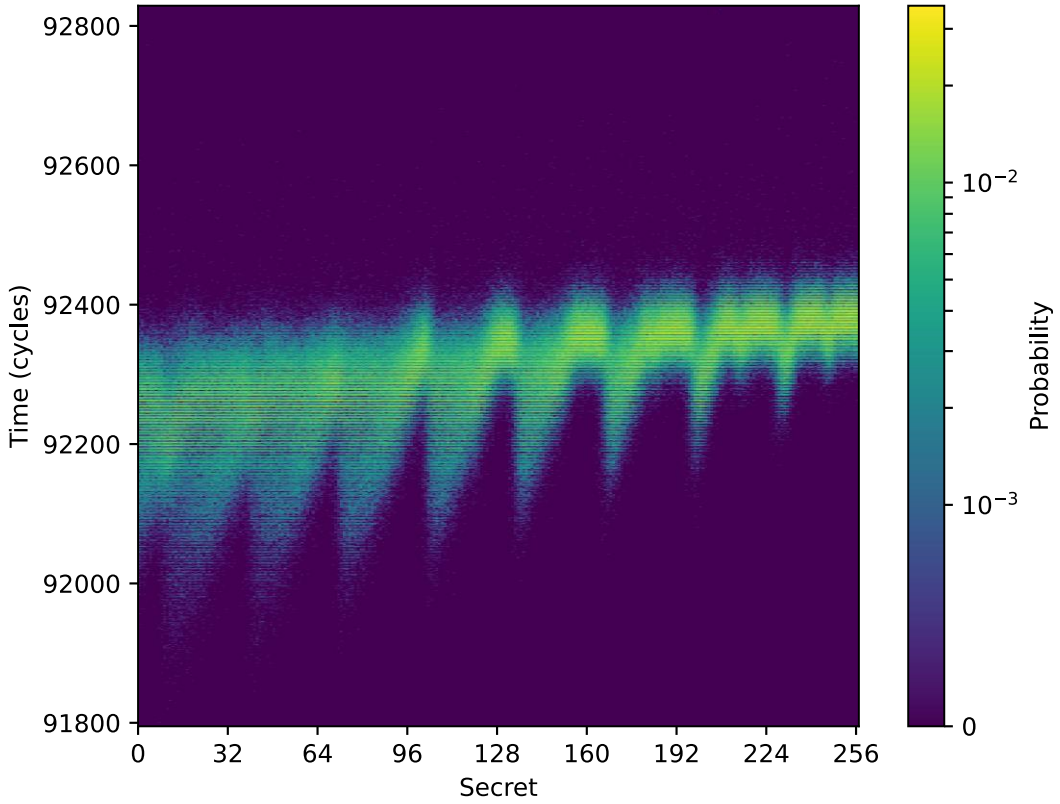
Software Mitigation: L1 D$ Channel

Single L1 D$ prime on context switch



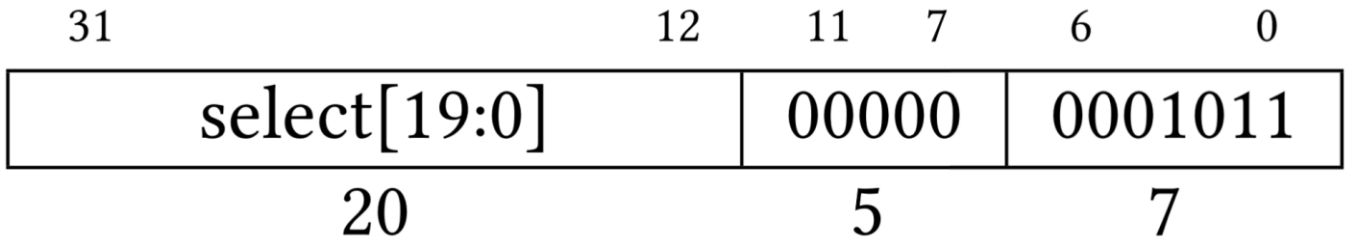
$N = 10^6$, $M = 1471.5$ mb, $M_0 = 0.6$ mb

Double L1 D$ prime on context switch

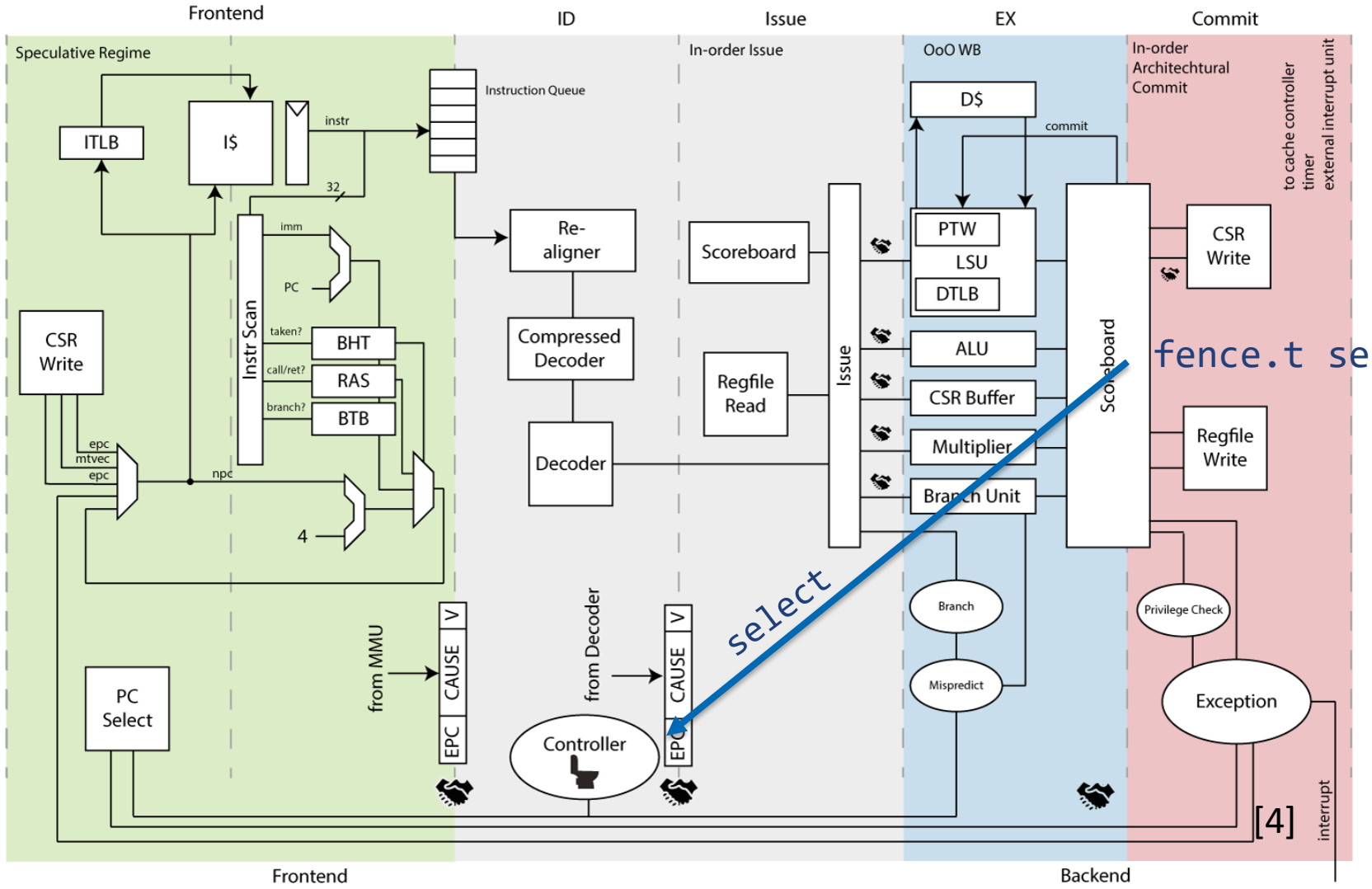


$N = 10^6$, $M = 515.7$ mb, $M_0 = 1.1$ mb

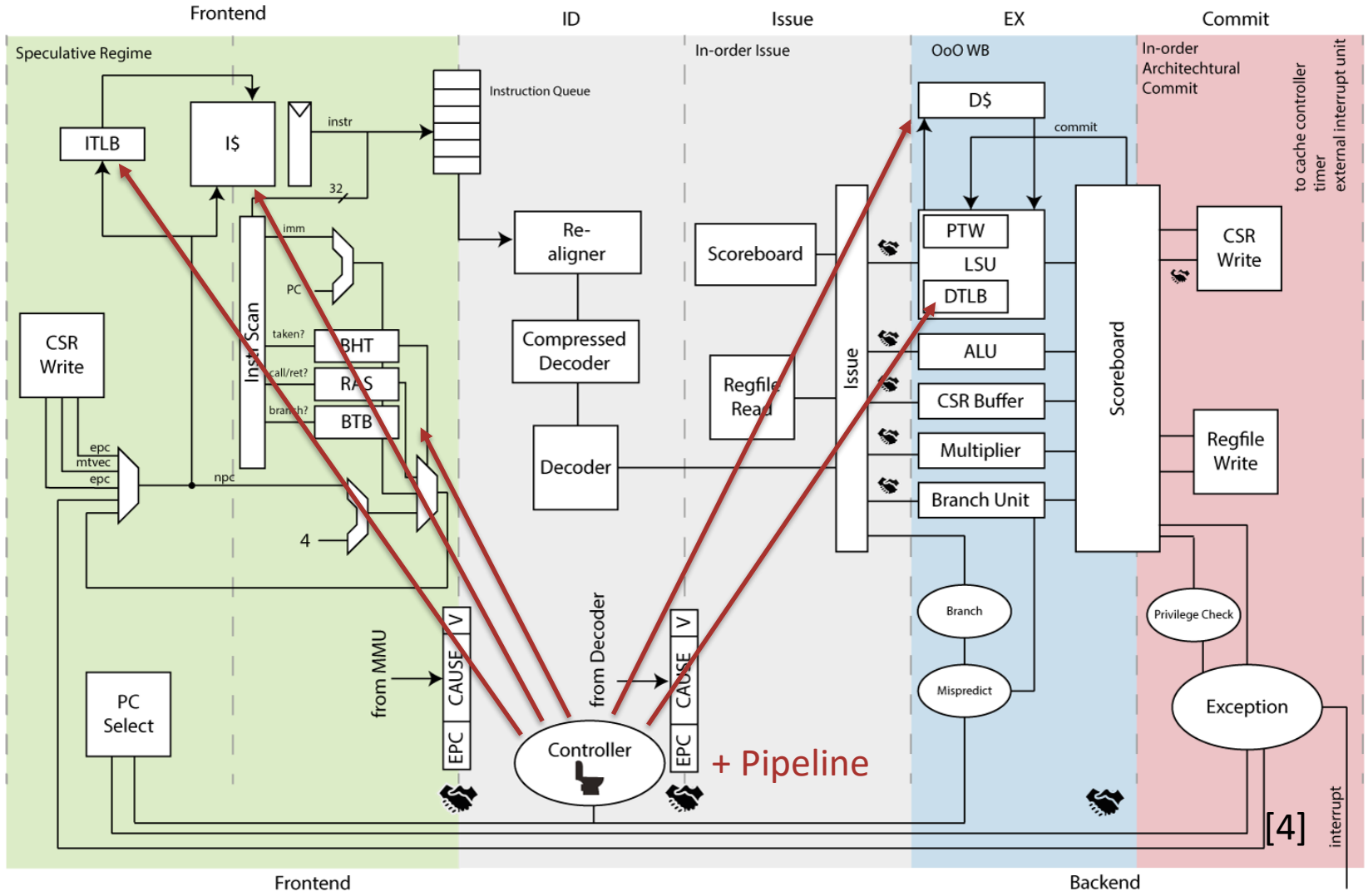
Temporal Fence Instruction (`fence.t`)



Temporal Fence Instruction (`fence.t`)

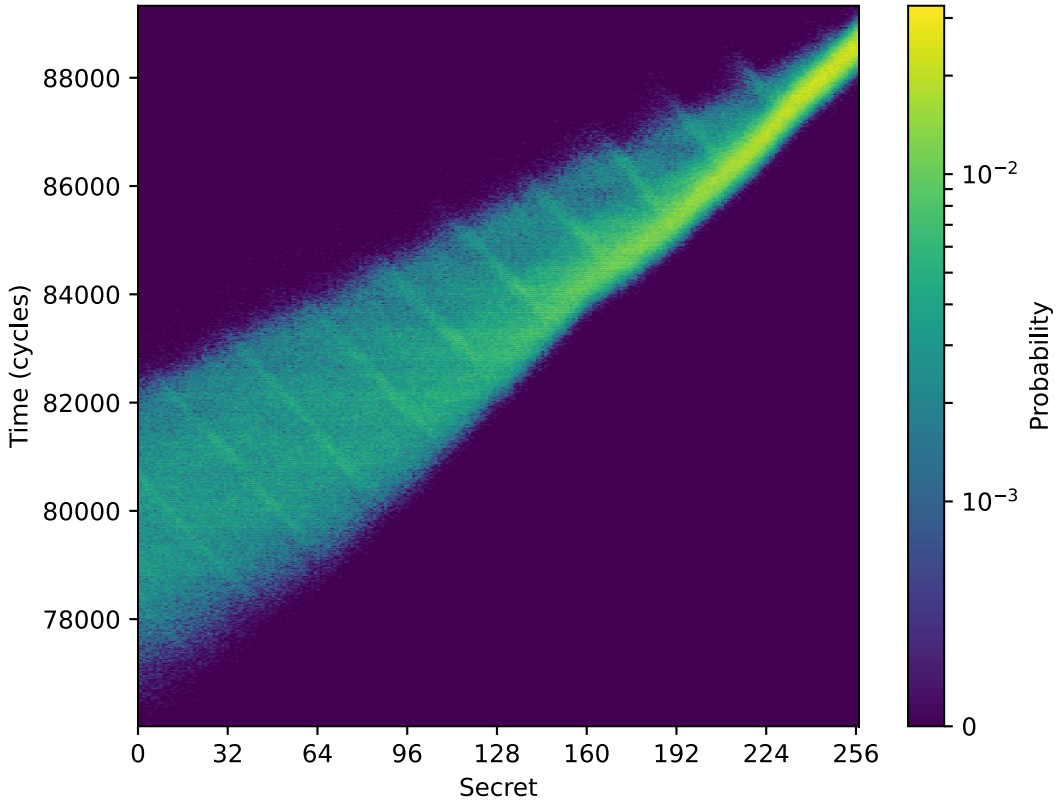


Temporal Fence Instruction (`fence.t`)



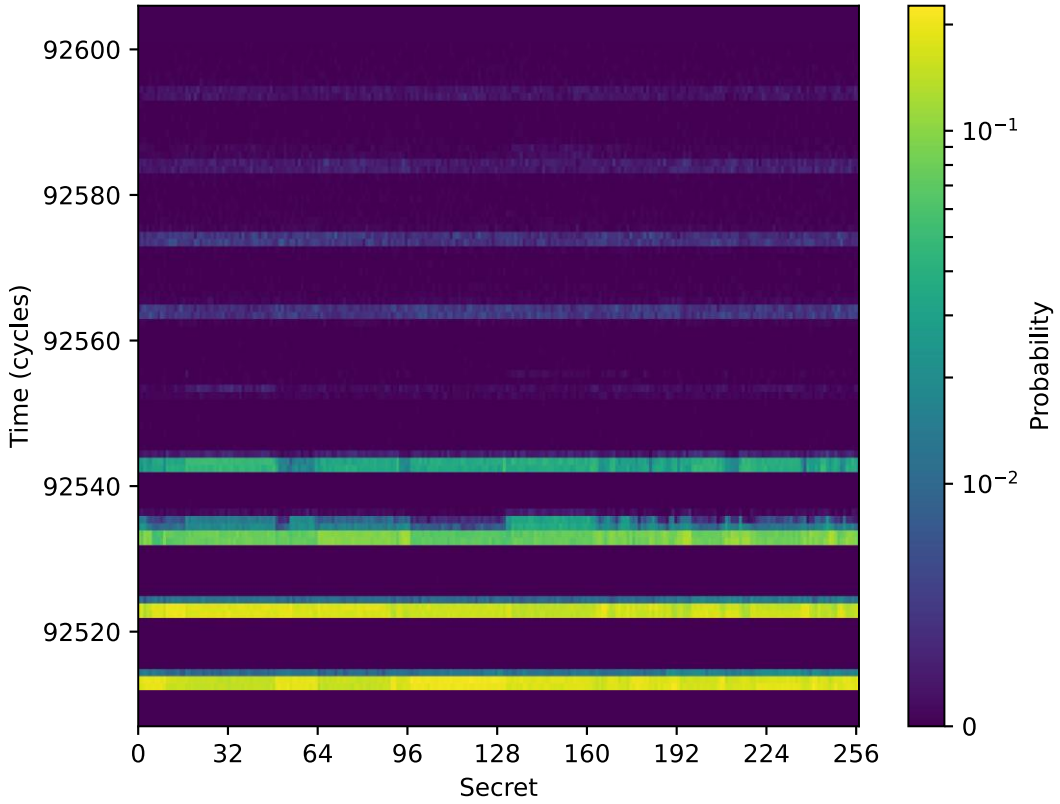
`fence.t`: L1 D$ Channel

Unmitigated



$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

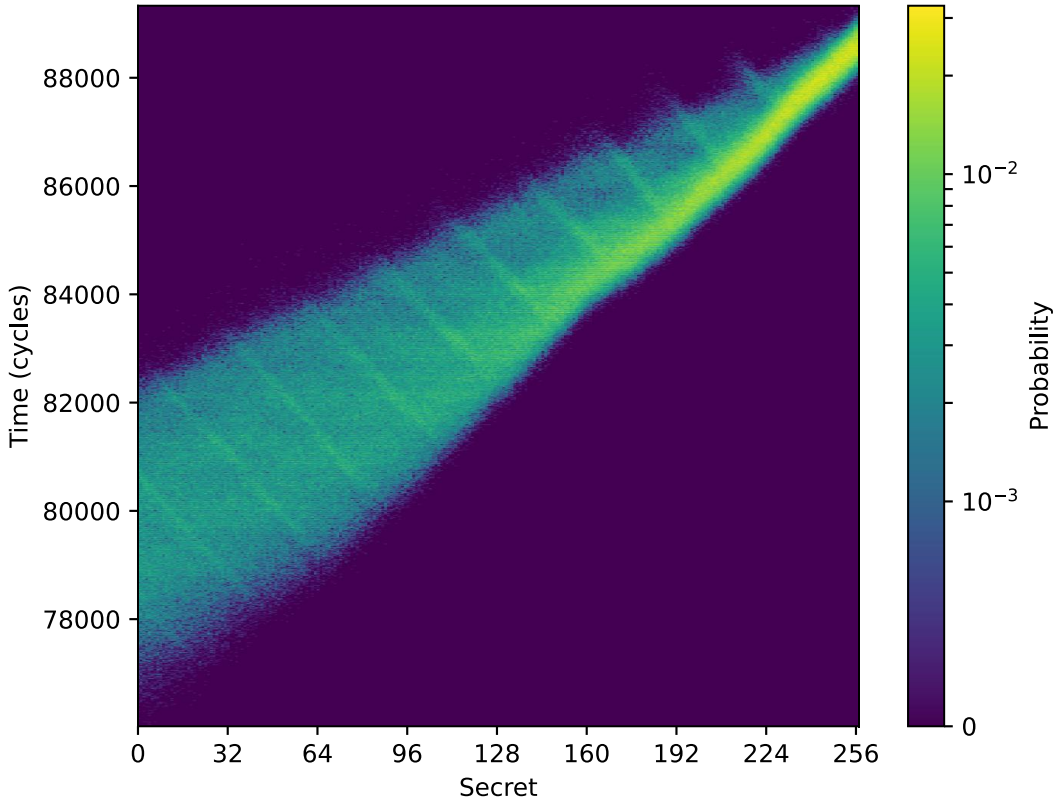
Flush targeted components on context switch



$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

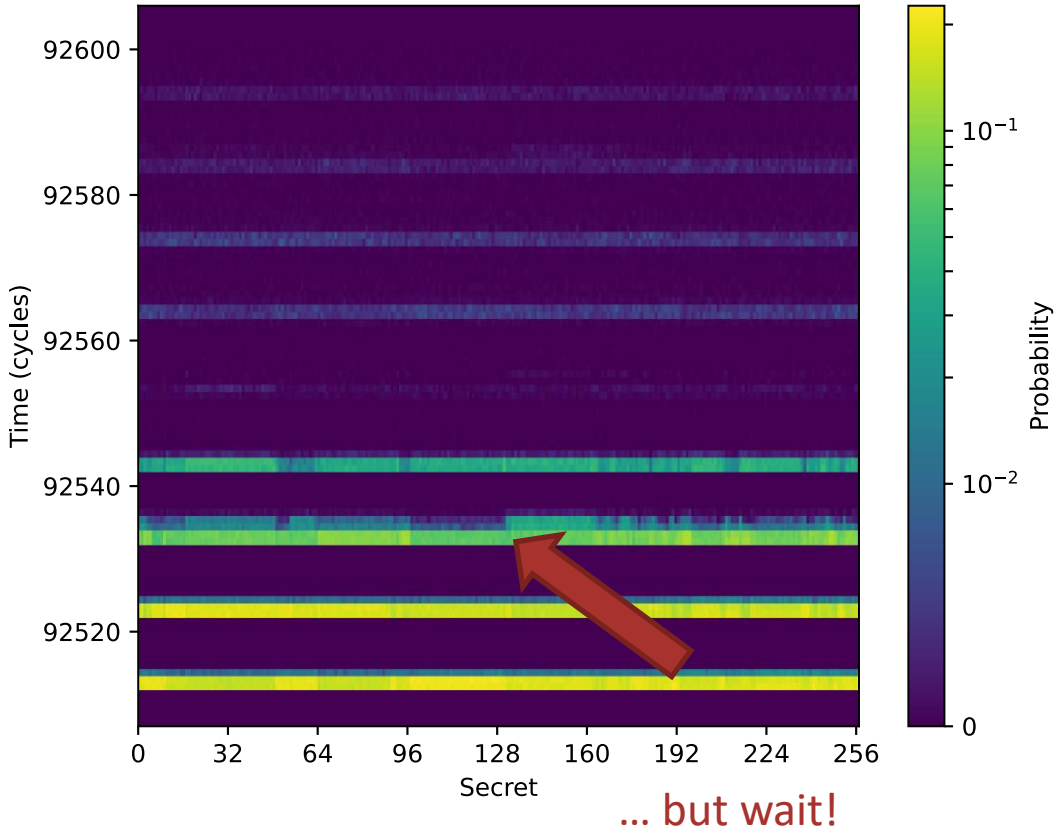
`fence.t`: L1 D$ Channel

Unmitigated



$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

Flush targeted components on context switch



… but wait!

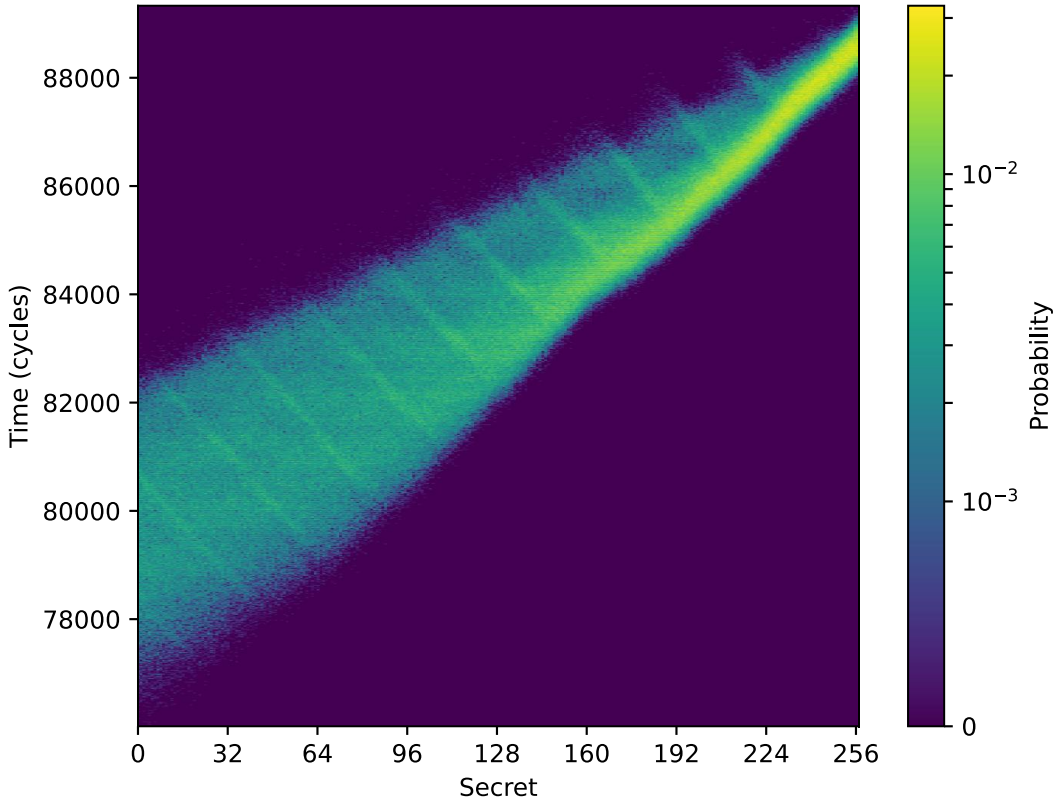
$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

Vulnerable 2nd Order State-Holding Components

- **L1 D$:**
  - LFSR for pseudo-random replacement policy
  - Memory arbiter
  - TX FIFO
  - Write-buffer arbiters
- **L1 I$:**
  - LFSR for pseudo-random replacement policy
- **TLBs:**
  - Pseudo-LRU tree for replacement policy

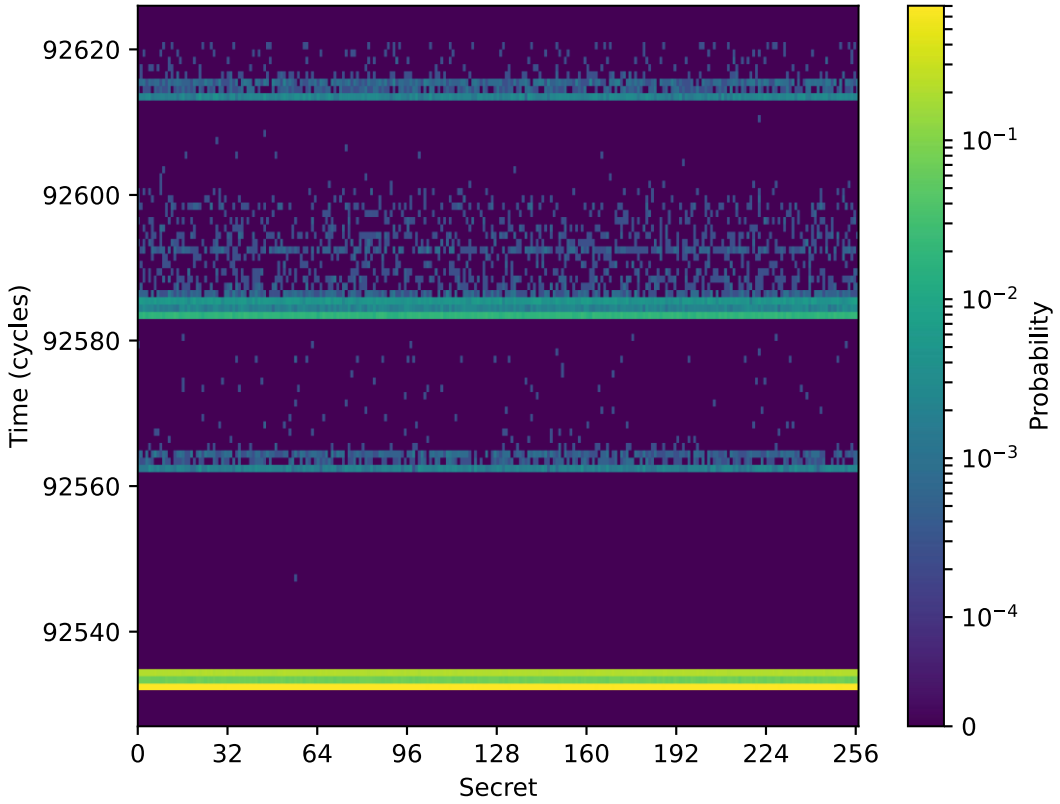
Full `fence.t`: L1 D$ Channel

Unmitigated



$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

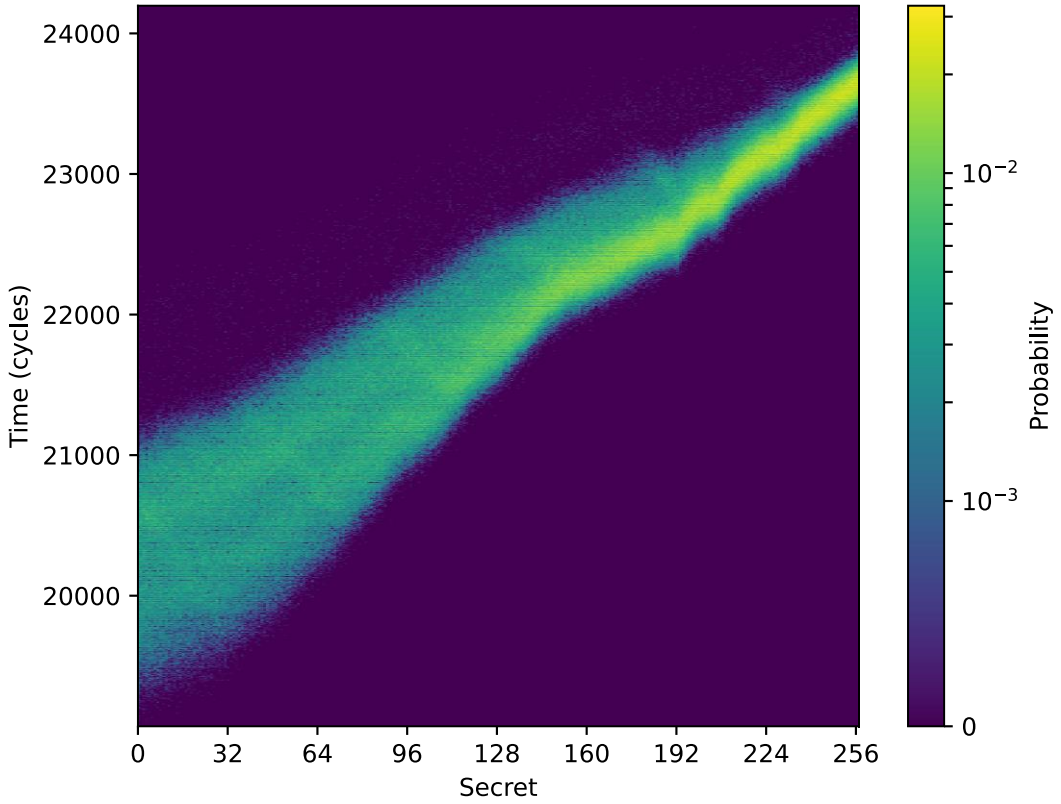
Flush all vulnerable components on context switch



$N = 10^6$, $M = 8.4$ mb, $M_0 = 9.6$ mb

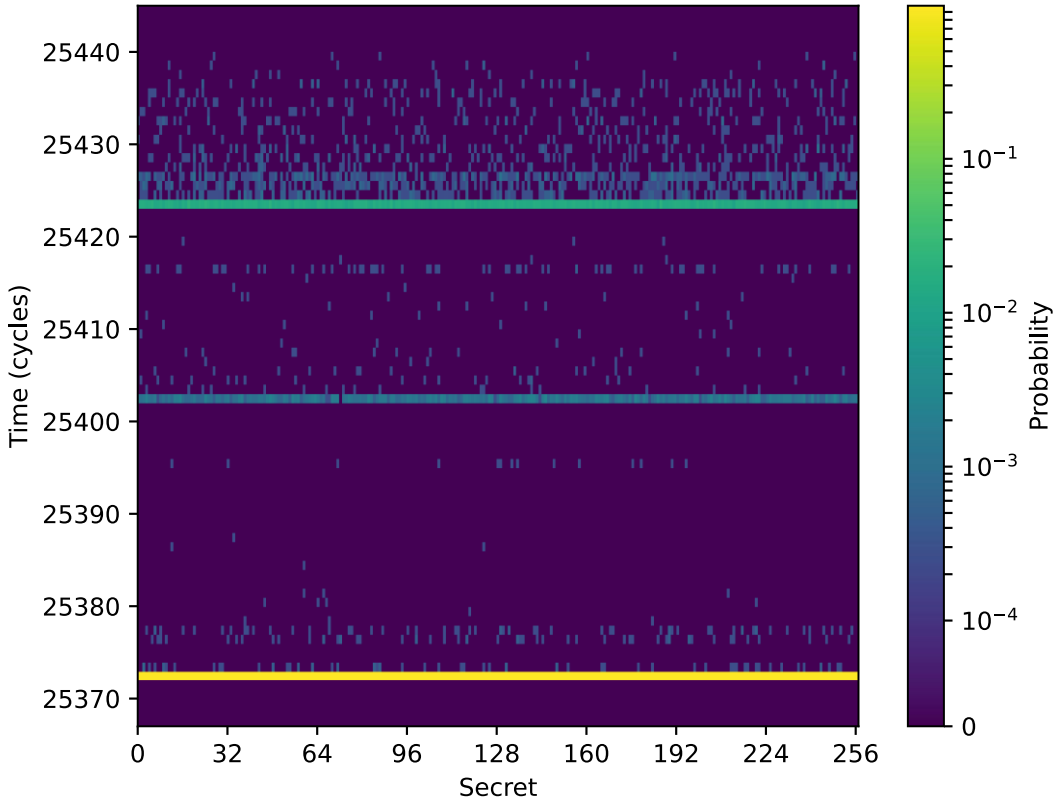
L1 I$ Channel

Unmitigated



$N = 10^6$, $M = 1905.0$ mb, $M_0 = 0.5$ mb

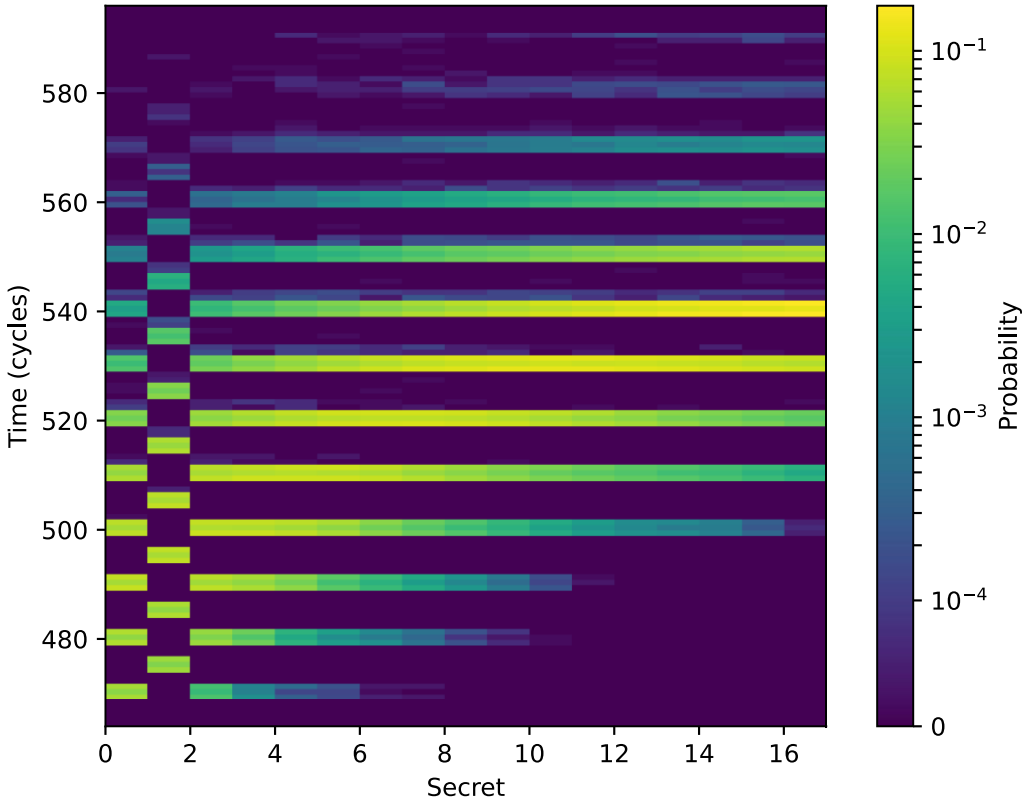
Flush all vulnerable components on context switch



$N = 10^6$, $M = 19.5$ mb, $M_0 = 20.5$ mb

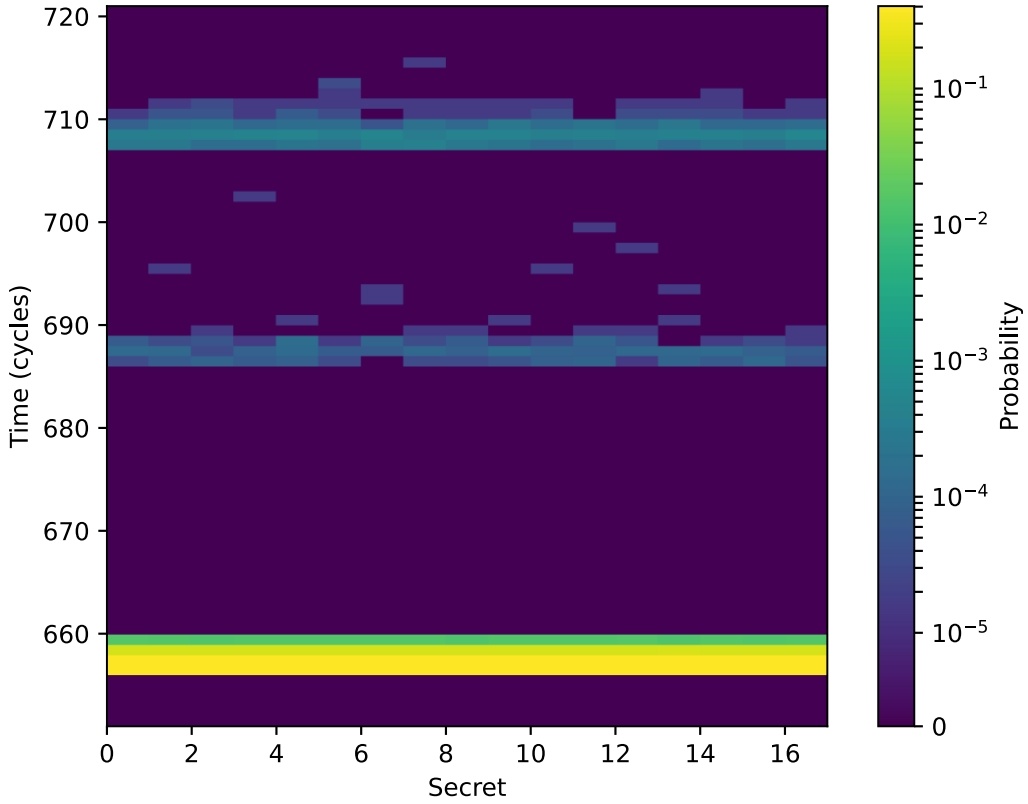
TLB Channel

Unmitigated



$N = 10^6$, $M = 409.2$ mb, $M_0 = 0.1$ mb

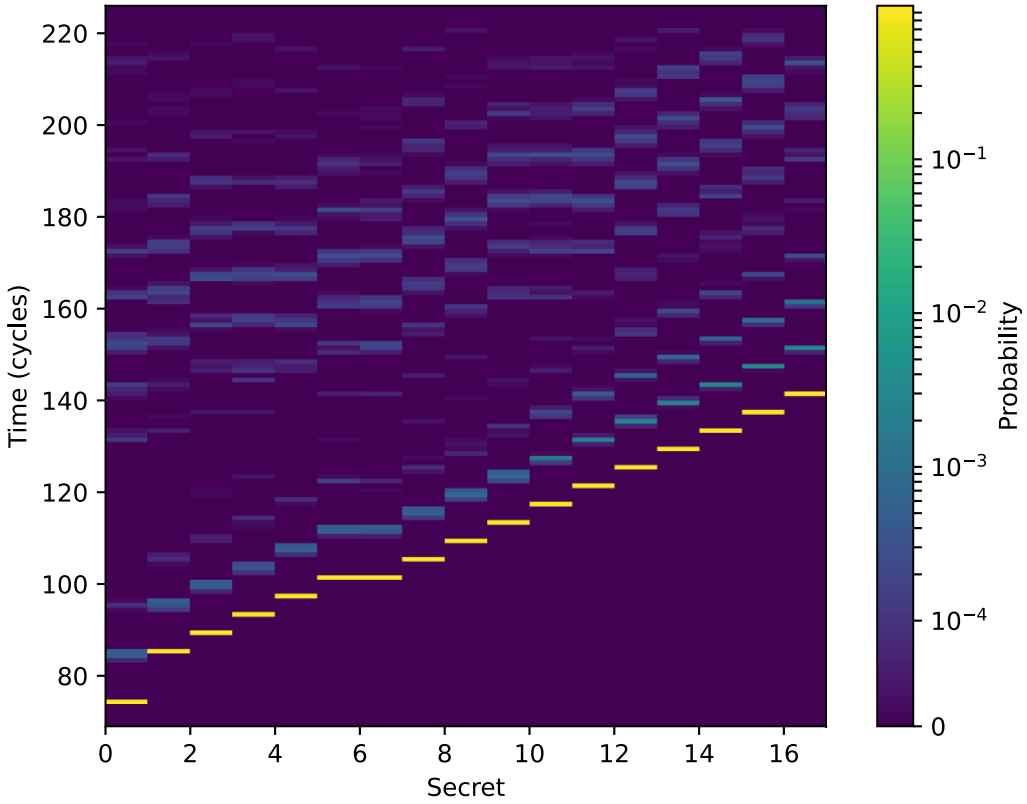
Flush all vulnerable components on context switch



$N = 10^6$, $M = 2.7$ mb, $M_0 = 5.4$ mb

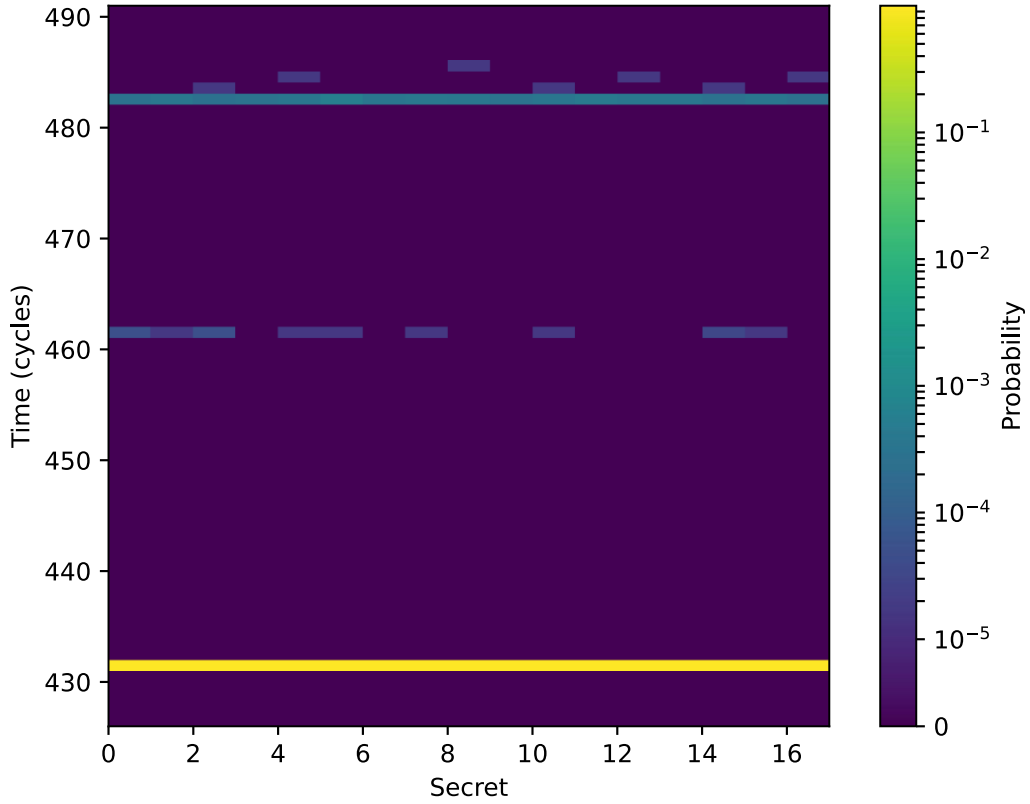
BTB Channel

Unmitigated



$N = 10^6$, $M = 3481.3$ mb, $M_0 = 0.1$ mb

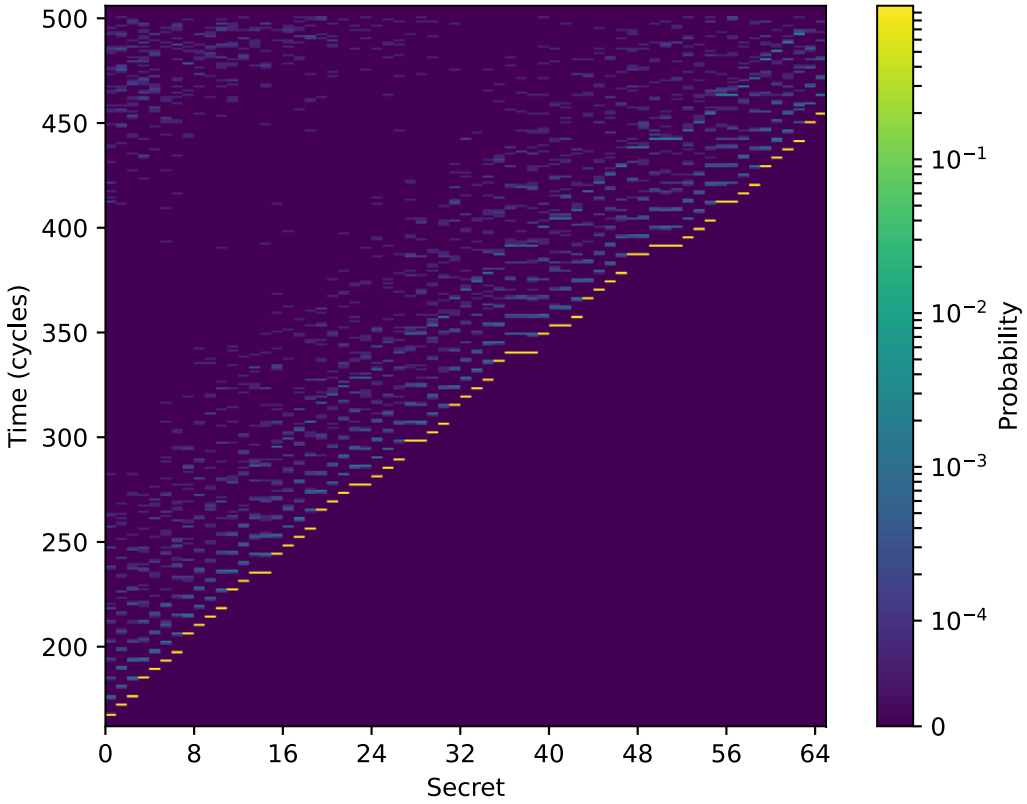
Flush all vulnerable components on context switch



$N = 10^6$, $M = 33.0$ mb, $M_0 = 57.6$ mb

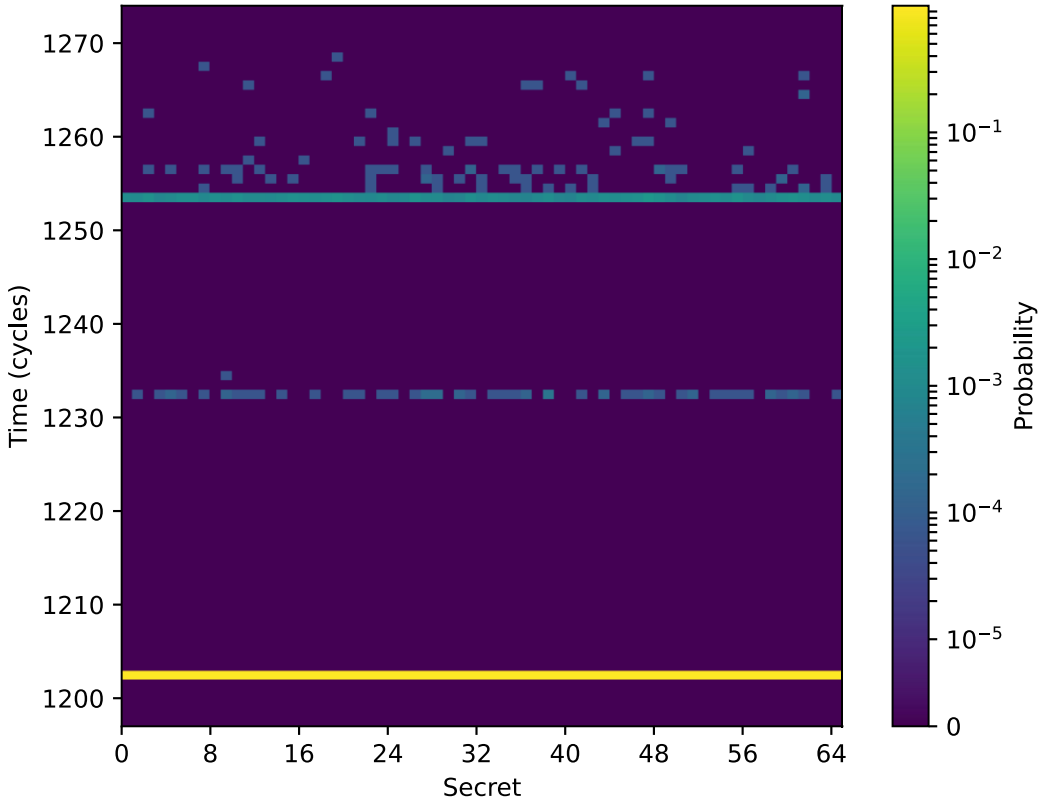
BHT Channel

Unmitigated



$N = 10^6$, $M = 4873.3$ mb, $M_0 = 0.1$ mb

Flush all vulnerable components on context switch



$N = 10^6$, $M = 44.1$ mb, $M_0 = 58.8$ mb

Context Switch Latency

seL4 one-way inter-address-space IPC microbenchmark

Unmitigated	
Hot	Cold
430 (±7.0)	1,180 (±1.0)

Context Switch Latency

seL4 one-way inter-address-space IPC microbenchmark

Unmitigated		D$ Software Flush	
Hot	Cold	Single	Double
430 (±7.0)	1,180 (±1.0)	12,099 (±52)	51,876 (±256)

Context Switch Latency

seL4 one-way inter-address-space IPC microbenchmark

Unmitigated		D$ Software Flush		HW Flush
Hot	Cold	Single	Double	
430 (±7.0)	1,180 (±1.0)	12,099 (±52)	51,876 (±256)	1,502 (±0.9)

Hardware Costs: FPGA

	LUTs	Registers	Muxes
Unmodified	102,796 (±10)	58,957 (±208)	13,590 (±38)
w/ `fence.t`	102,792 (±57)	60,607 (±5)	15,038 (±2)
	±0%	**+2.8%**	**+10.6%**

Conclusion

- **We measure five distinct covert channels on Ariane**
- **Confirmed: OS needs HW-support for time protection [1]**
- **HW-mechanism must flush *all* μArch state**
  - Identifying μArch state not always straight-forward
  - Systematic approach for HW / Security codesign needed
- **Further, off-core covert channels still need to be addressed**
  - e.g. DRAM, thermal controller, etc.

Sources

[1] Qian Ge, Yuval Yarom, Tom Chothia, and Gernot Heiser: “Time Protection: The Missing OS Abstraction”, EuroSys, 2019

[2] R. E. Kessler and Mark D. Hill: “Page Placement Algorithm for Large Real-Indexed Caches”, ACM Trans. Comp. Syst. 19, 1992

[3] Wolfgang Rönninger: “Memory Subsystem for the First Fully Open-Source RISC-V Heterogeneous SoC”, Master’s thesis, ETH Zurich, 2019

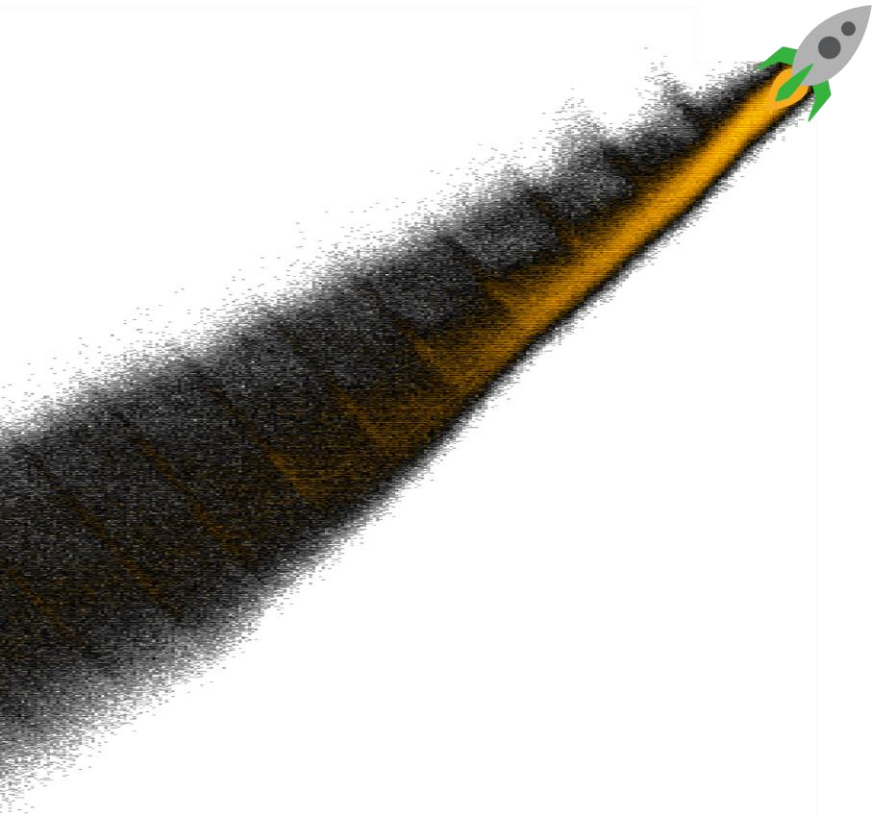
[4] Florian Zaruba and Luca Benini: “The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology”, IEEE Trans. on VLSI Systems 27, 2019

[5] Gerwin Klein, June Andronick, Kevin Elphistone, Toby Murray, Thomas Sewell, Rafal Kolanski, and Gernot Heiser: “Comprehensive Formal Verification of an OS Microkernel”, ACM Trans. Comp. Syst. 32, 2014

ETH zürich

Prevention of Microarchitectural Covert Channels on an Open-Source 64-bit RISC-V Core

Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020)



May 29th, 2020

Nils Wistoff

Moritz Schneider

Frank K. Gürkaynak

Luca Benini

Gernot Heiser

RWTH AACHEN UNIVERSITY

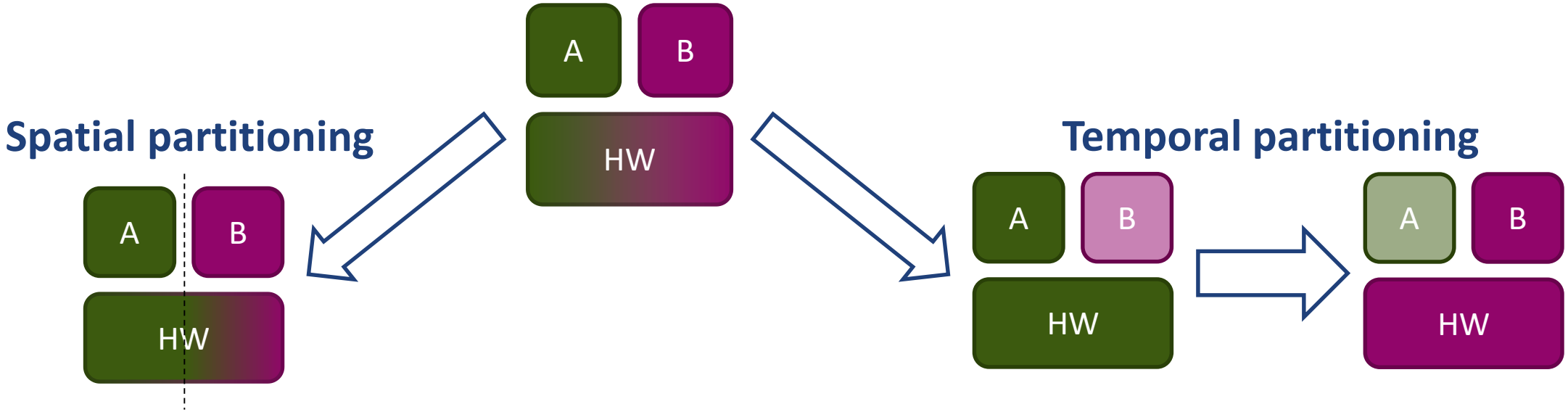
CSIRO DATA 61

HENSOLDT Detect and Protect.

Hardware Costs: FPGA

	LUTs	Registers	Muxes
Unmodified	102,796 (±10)	58,957 (±208)	13,590 (±38)
w/ `fence.t`	102,792 (±57) 50.4%	60,607 (±5) 14.9%	15,038 (±2) 9.8%
	±0%	**+2.8%**	**+10.6%**

Time Protection [1]



- Off-core components
- e.g. cache colouring (LLC) [2]
- Not a solution for on-core components!

- On-core components
- e.g. L1 caches, TLBs, branch predictors
- Reset μArch state on context switch