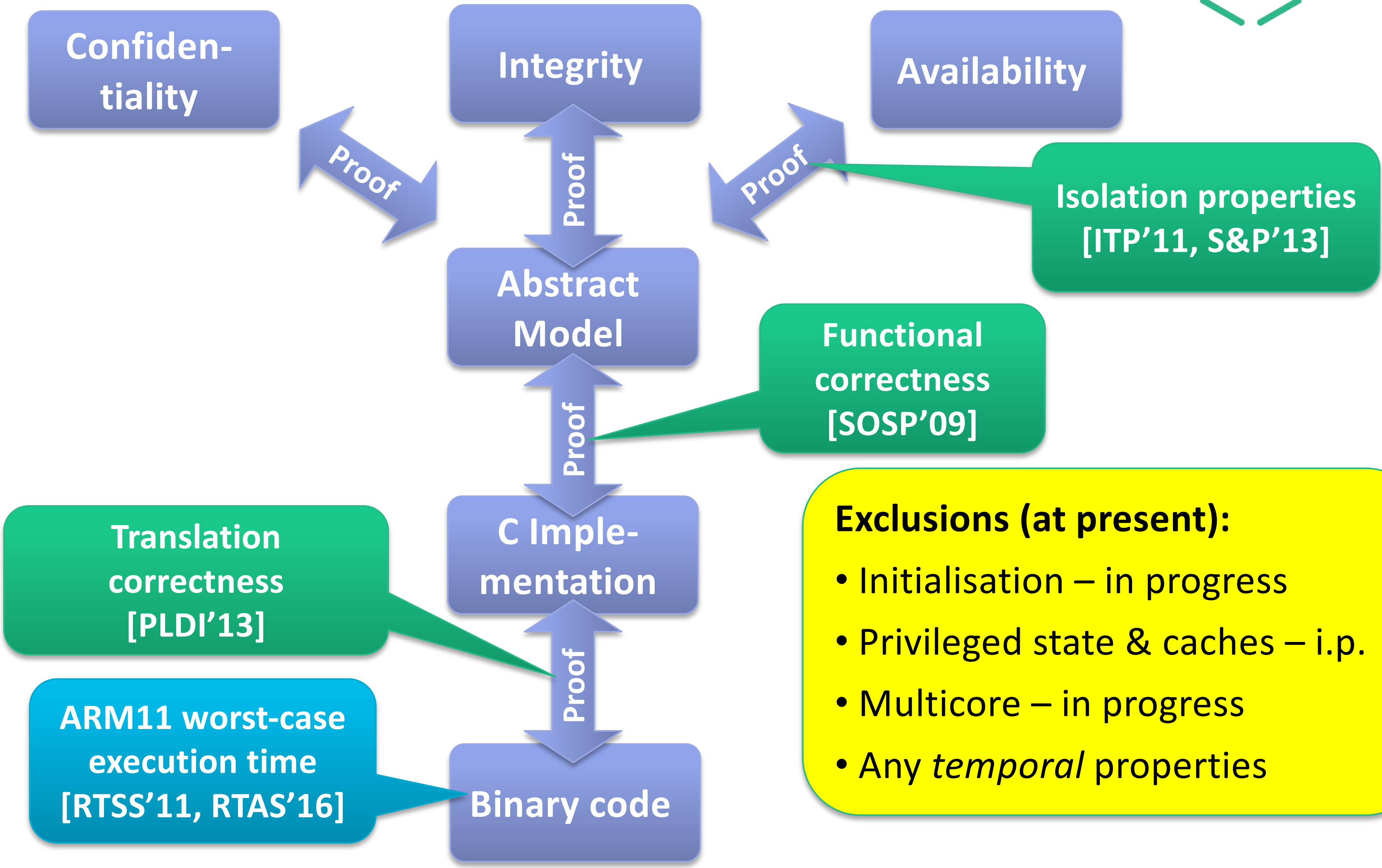
Challenges of Temporal Isolation

Gernot Heiser
gernot.heiser@data61.csiro.au | @GernotHeiser
Dagstuhl, October 2016

https://trustworthy.systems

Spatial Isolation: Solved



Exclusions (at present):

- Initialisation – in progress
- Privileged state & caches – i.p.
- Multicore – in progress
- Any *temporal* properties

Critical Systems: DARPA HACMS

Boeing Unmanned Little Bird

Retrofit existing system!

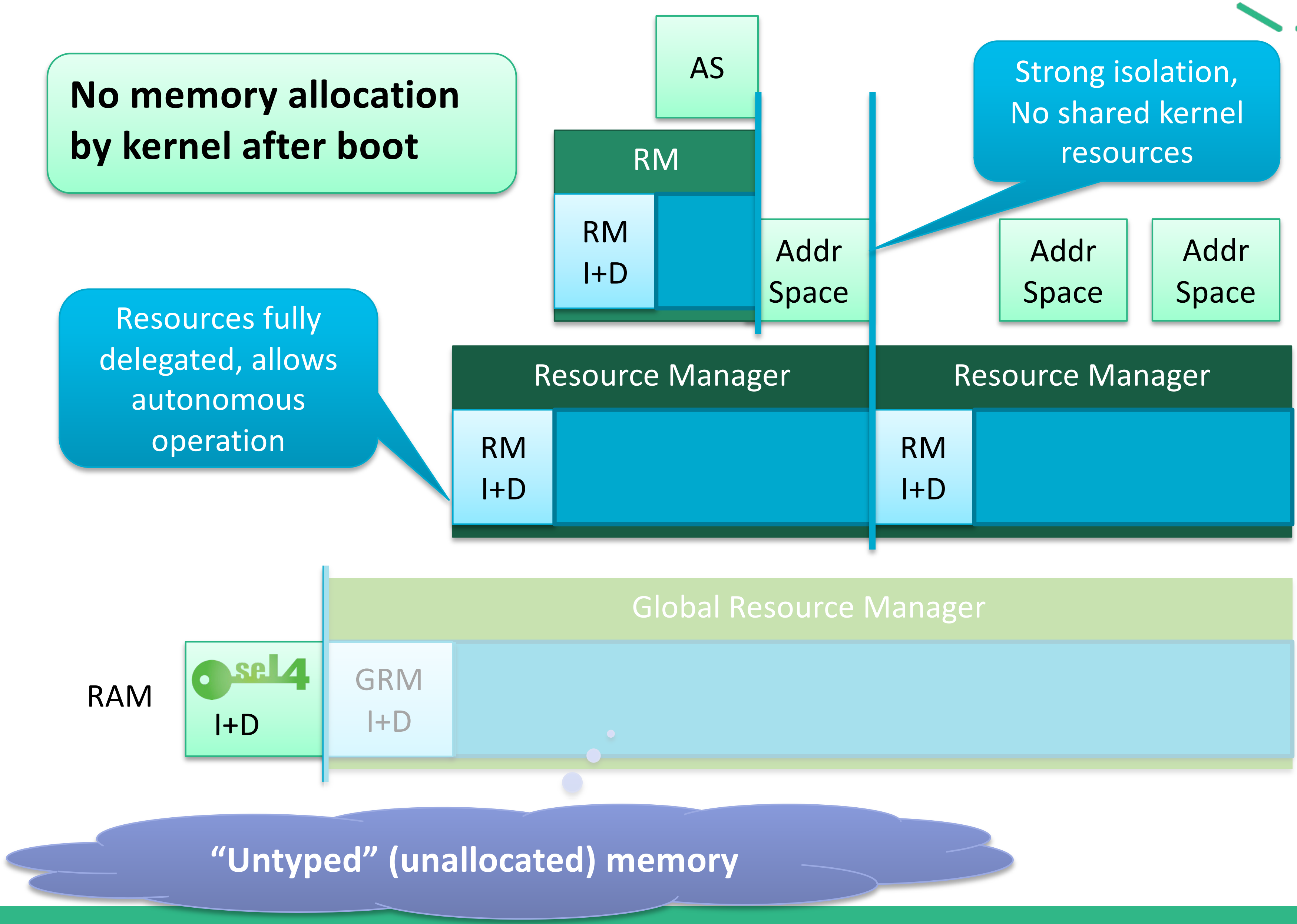
US Army Autonomous Trucks

SMACCMcopter Research Vehicle

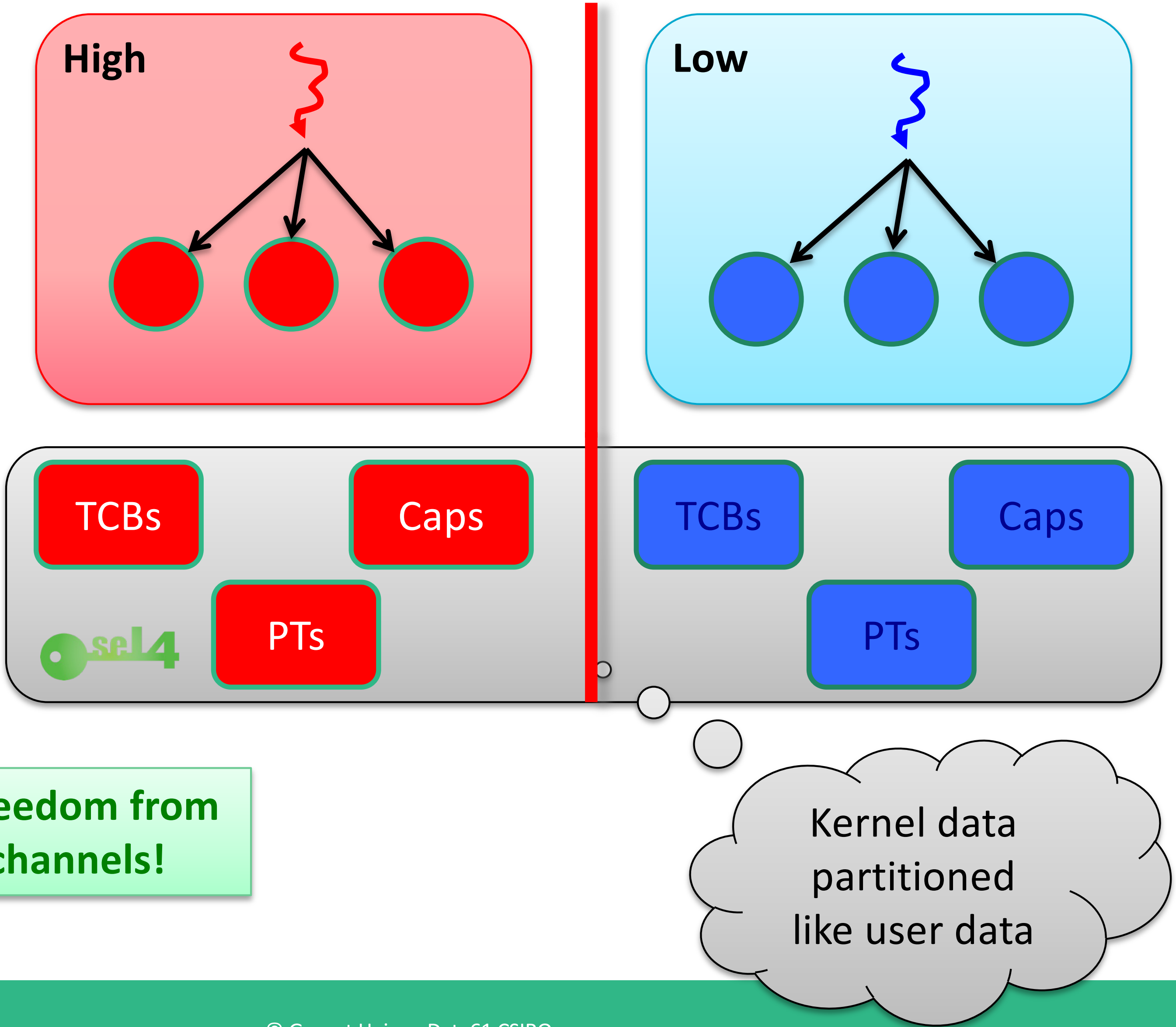
Develop technology

TARDEC GVRbot

Design for Isolation



Isolation Goes Deep



Provable freedom from storage channels!

Enabler of Spatial Isolation: ISA

ISA: instruction-set architecture

- Defines *functional* interface between hardware and software
- Contract on which the software-designer can rely for creating functionally-correct software
  - Modulo hardware bugs – e.g. rowhammer
- No information on timing, which is affected by
  - Pipelining
  - Multi-cycle instructions
  - Caches of various sorts:
    - Instruction cache (I-cache)
    - Data cache (D-cache)
    - Write buffers
    - Translation-lookaside buffer (TLB)
    - Branch-prediction unit (BPU)
  - Shared busses, memory controllers

Time – The Final Frontier

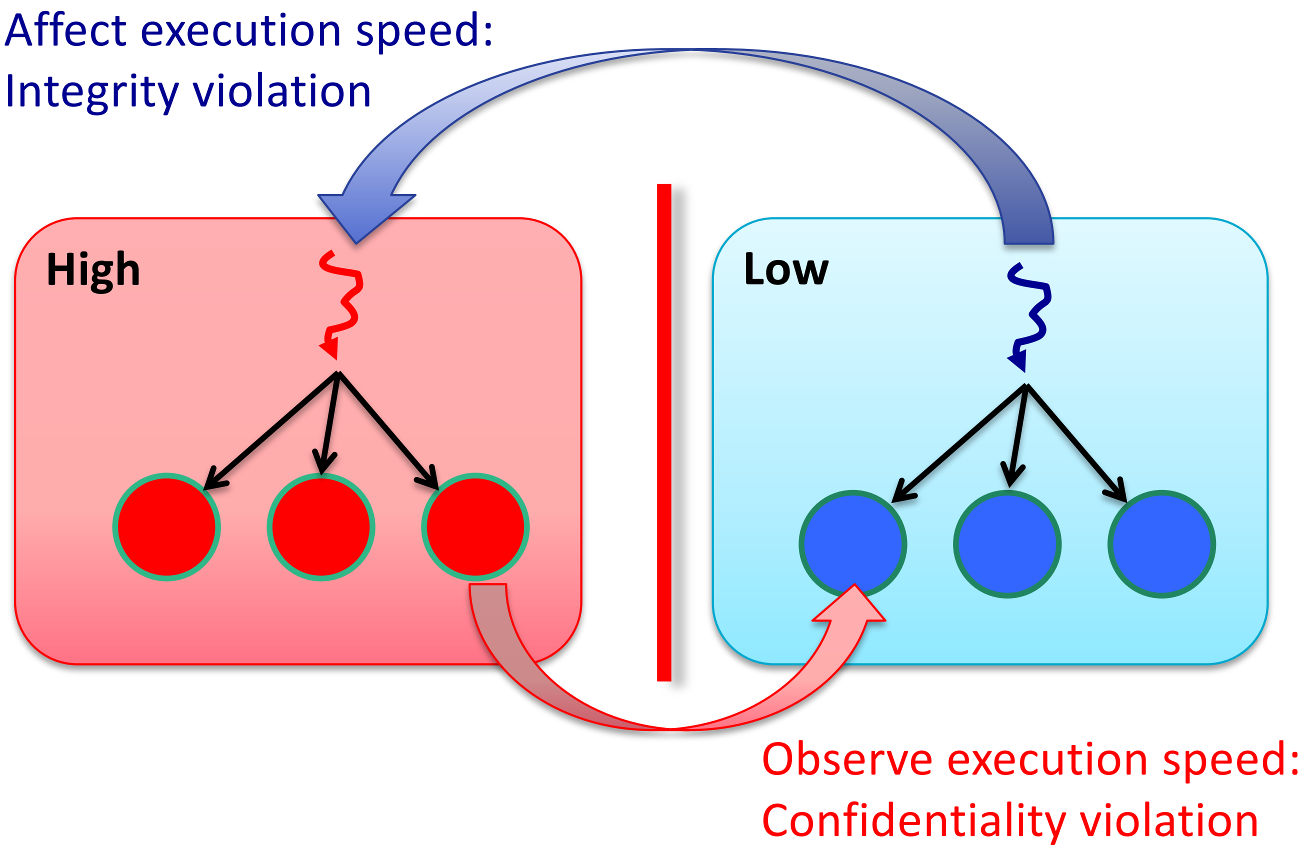
Two Aspects of Temporal Isolation

Safety: Timeliness

- *Bound* execution interference

Security: Confidentiality

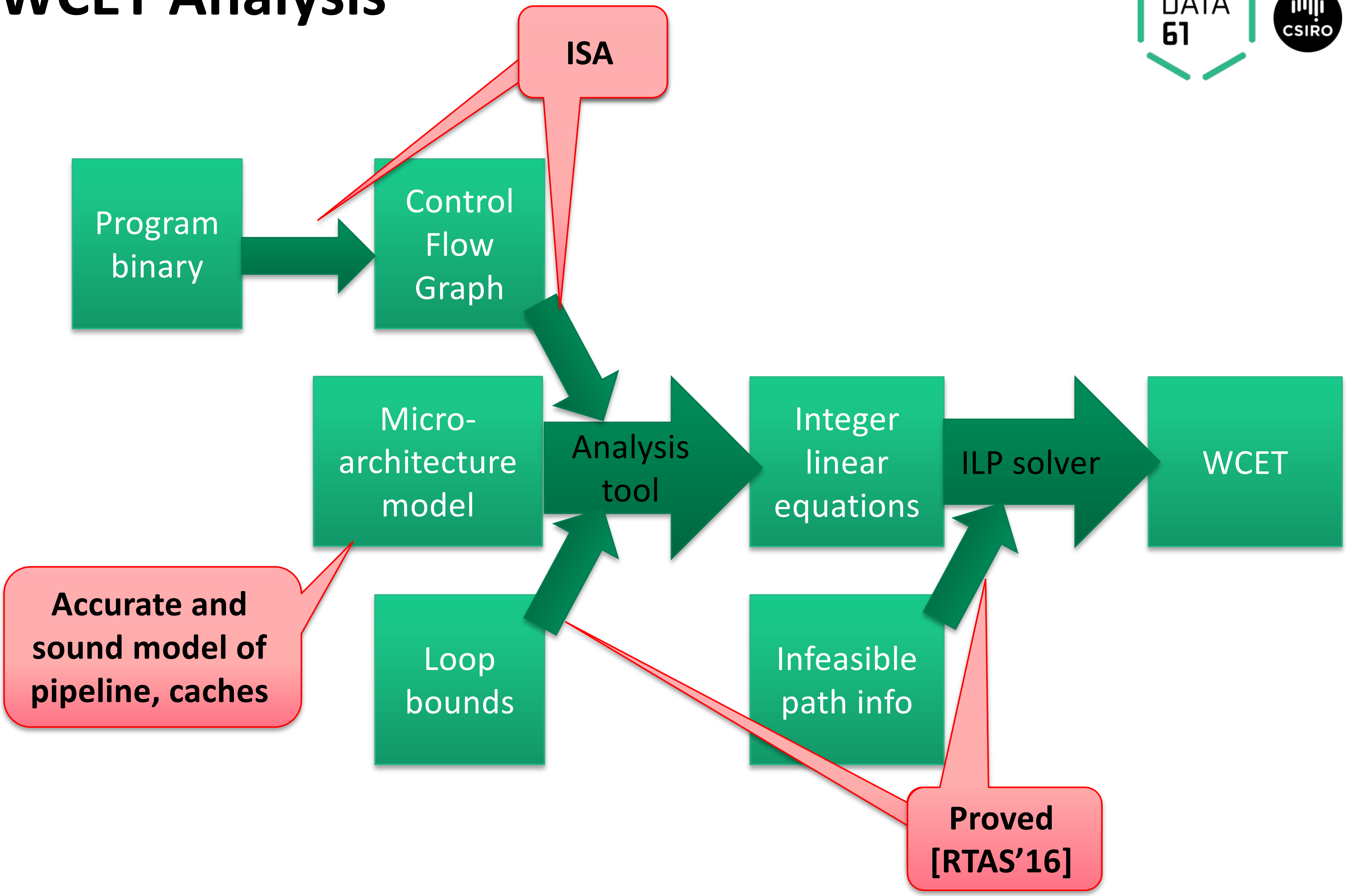
- *Prevent* leakage via timing channels



Temporal Integrity Requirements

1. Must be able to enforce limits on CPU time consumption
   - … using priorities, time budgets
   - No problem for a decent real-time OS
     - Mostly an issue of OS API design, using appropriate scheduling theory
2. Must be able to determine worst-case execution times (WCET)
   - … of application code
   - ... of the OS
   - Pessimism is the least worry, can use slack – *mixed-criticality systems*

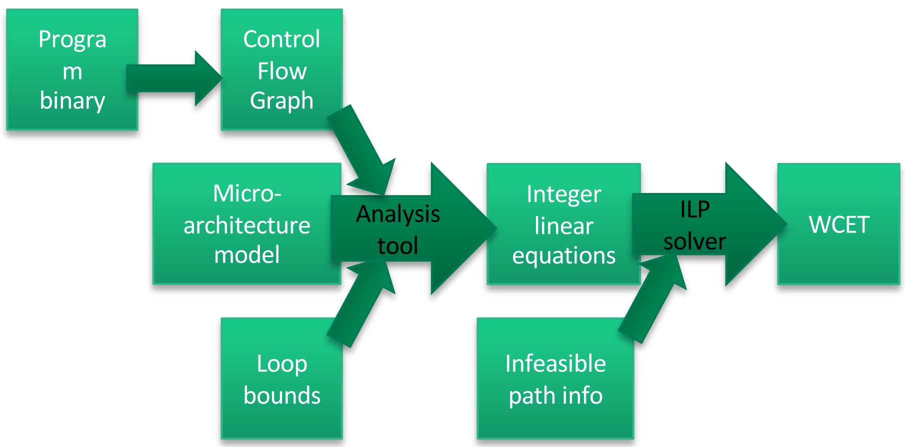
WCET Analysis



WCET Analysis

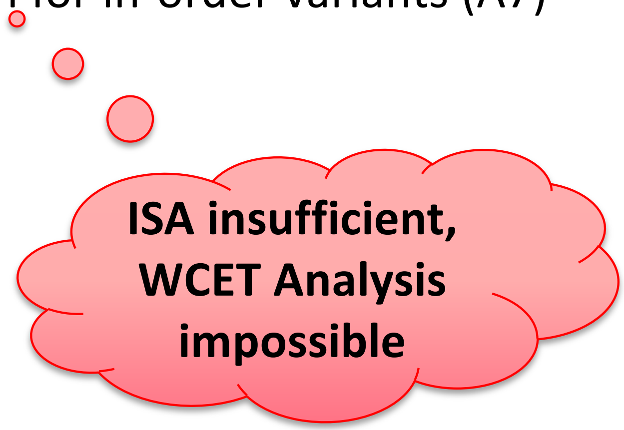
Works fine on ARM11 core

- ARM-provided list of instruction latencies
- Well-defined cache behaviour



Does not work on recent ARM, x86!

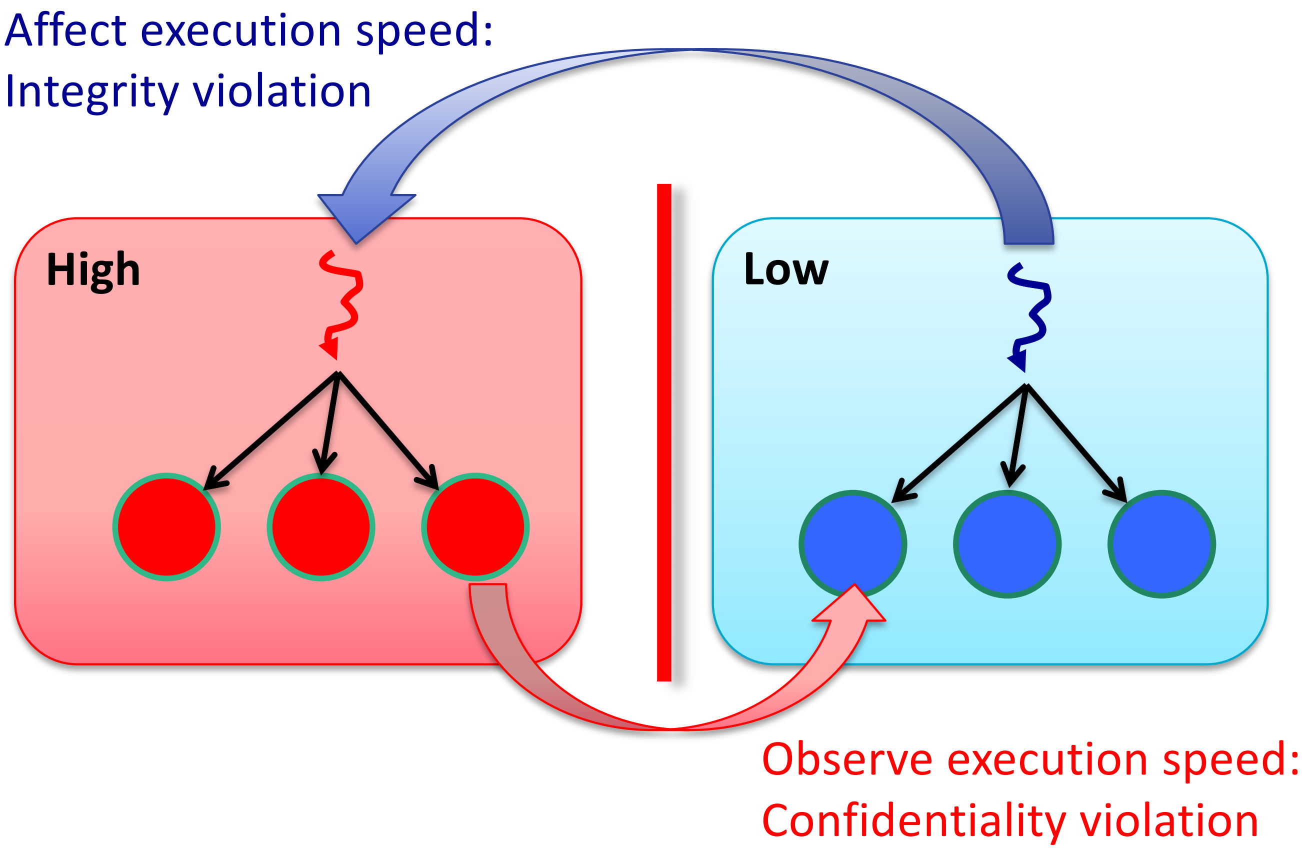
- Undefined cache behaviour
  - "random" replacement
    ⇒ high pessimism
- Out-of-order cores (A9, Core)
  - No published latency bounds
  - … even for in-order variants (A7)



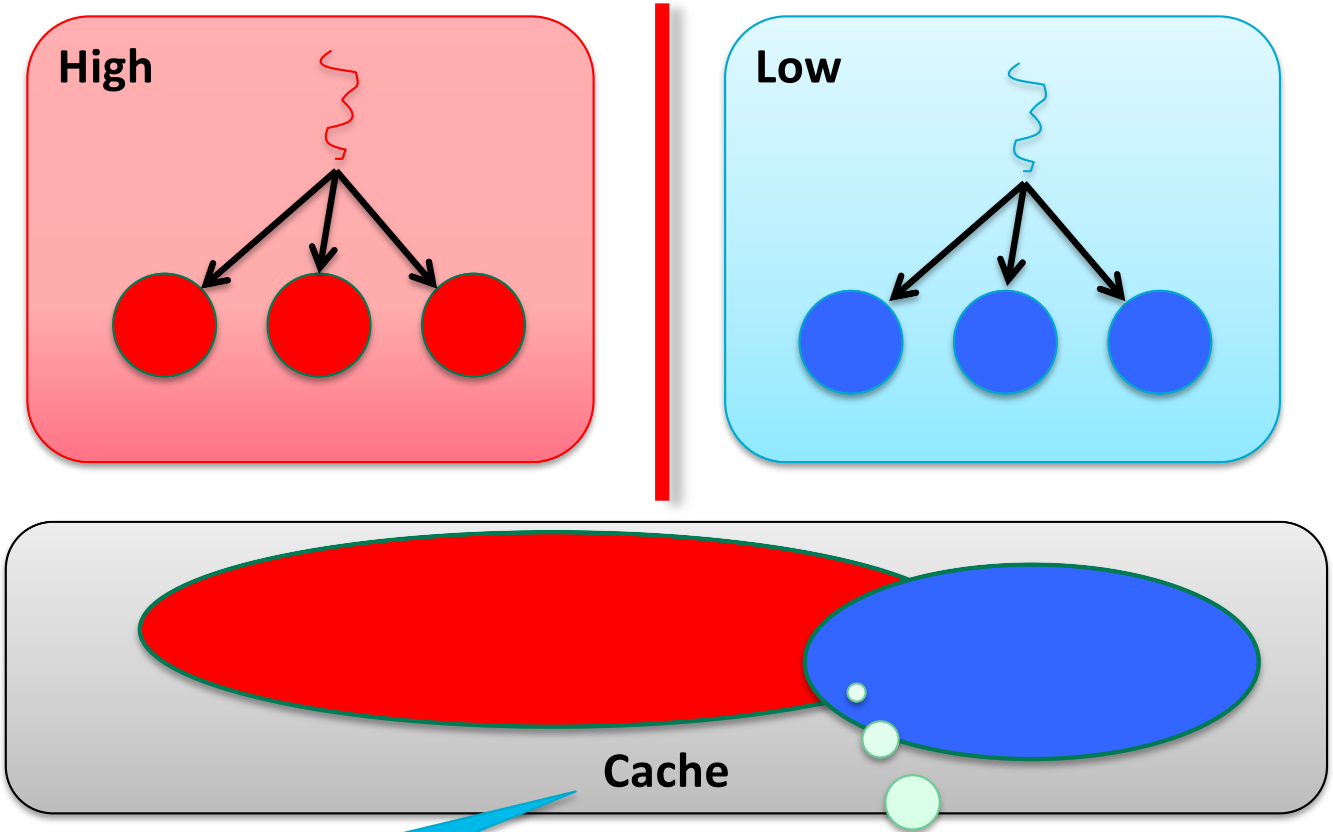
Two Aspects of Temporal Isolation

Safety: Timeliness
- *Bound* execution interference

Security: Confidentiality
- *Prevent* leakage via timing channels



Timing Channels

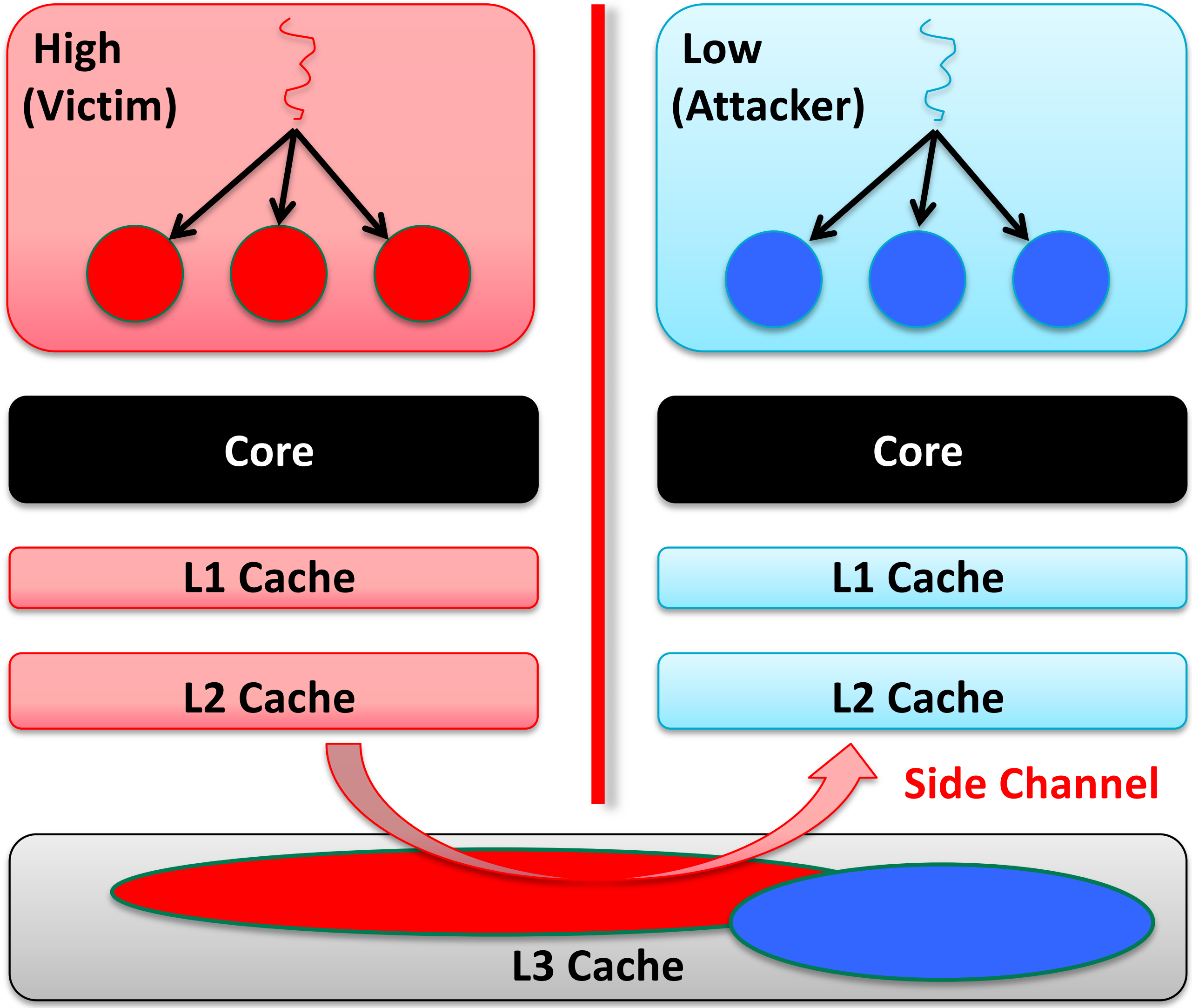


“Cache” might be:

- I-, D-cache
- TLB
- BPU…

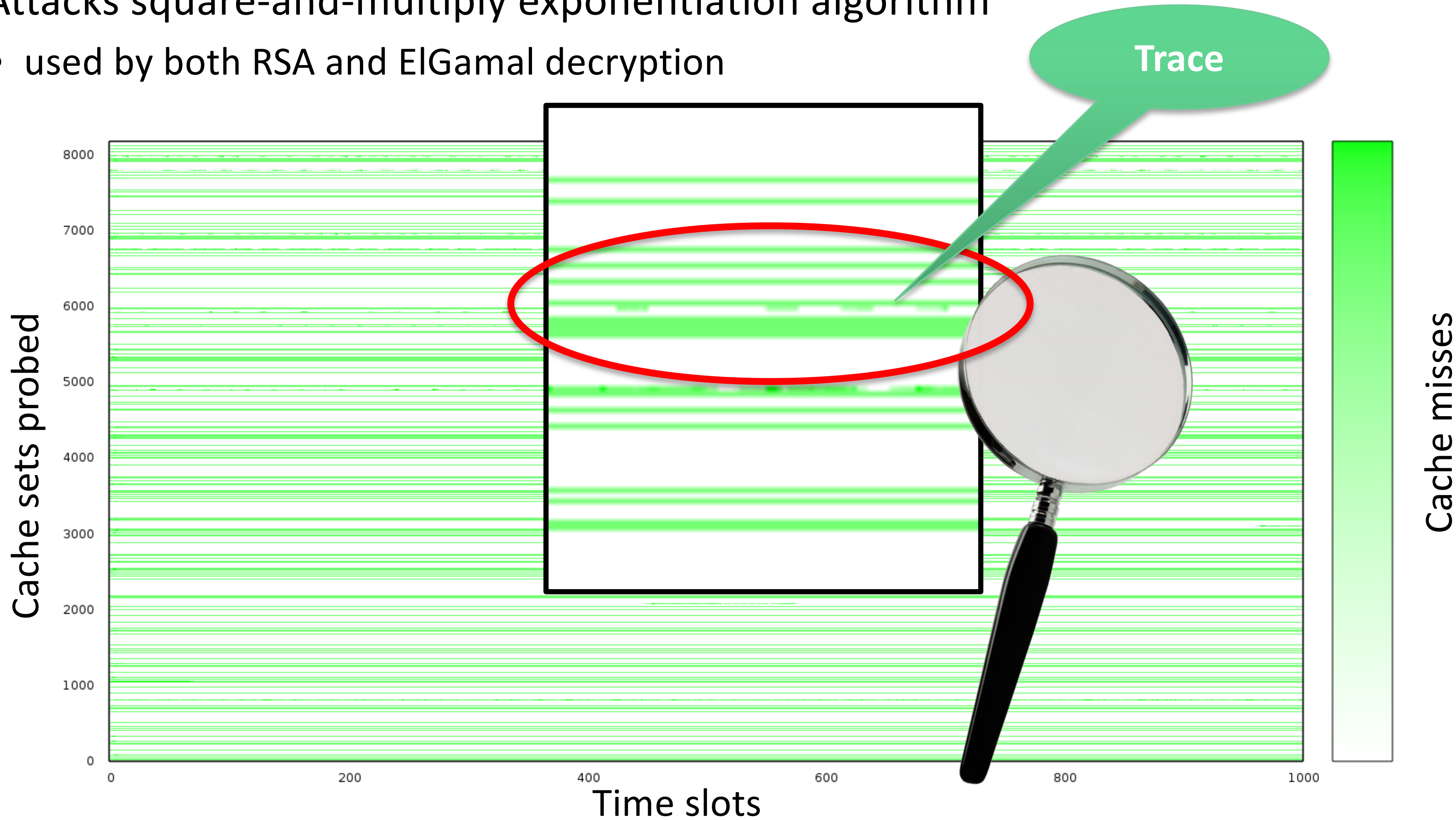
Cache footprint of one process affects progress of others!

Cloud Scenario: Cross-Core LLC Attack
Side-channel attack through last-level cache

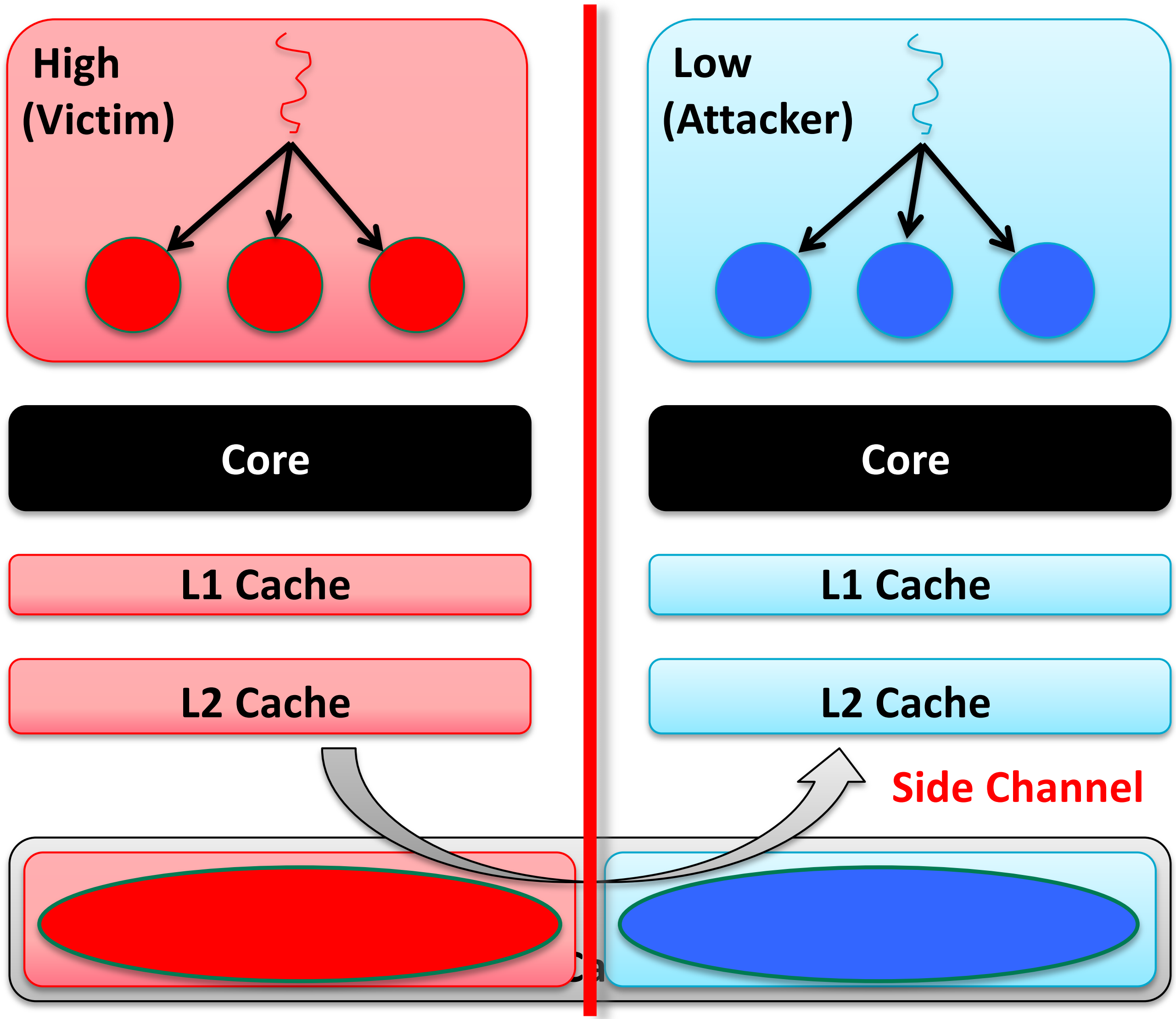


Cross-Core LLC Timing Side Channel
[Liu et al, Oakland'15]

- Prime + probe technique, cross-core, cross-VM
- Attacks square-and-multiply exponentiation algorithm
  - used by both RSA and ElGamal decryption

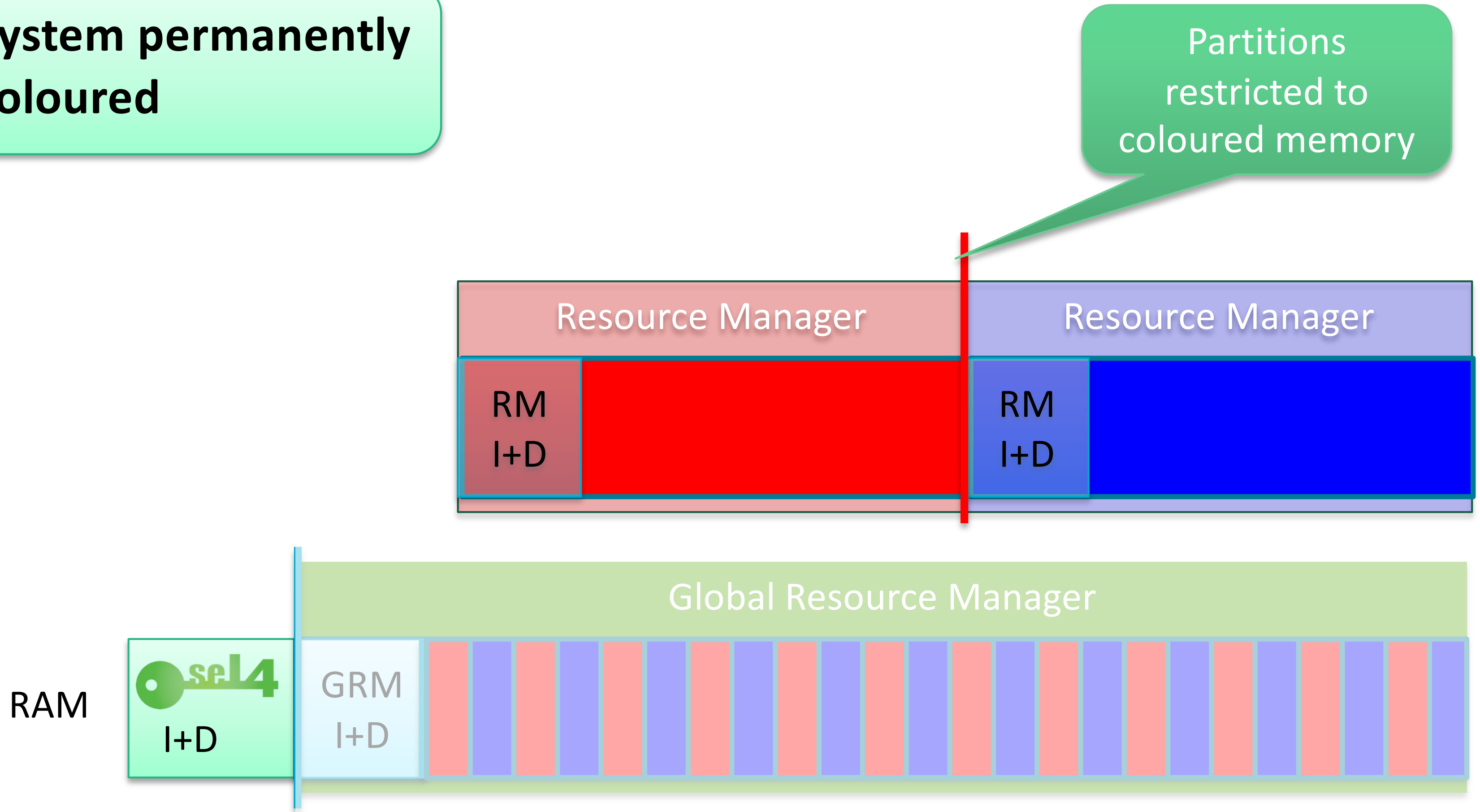


Mitigation: Partition Cache (Colouring)



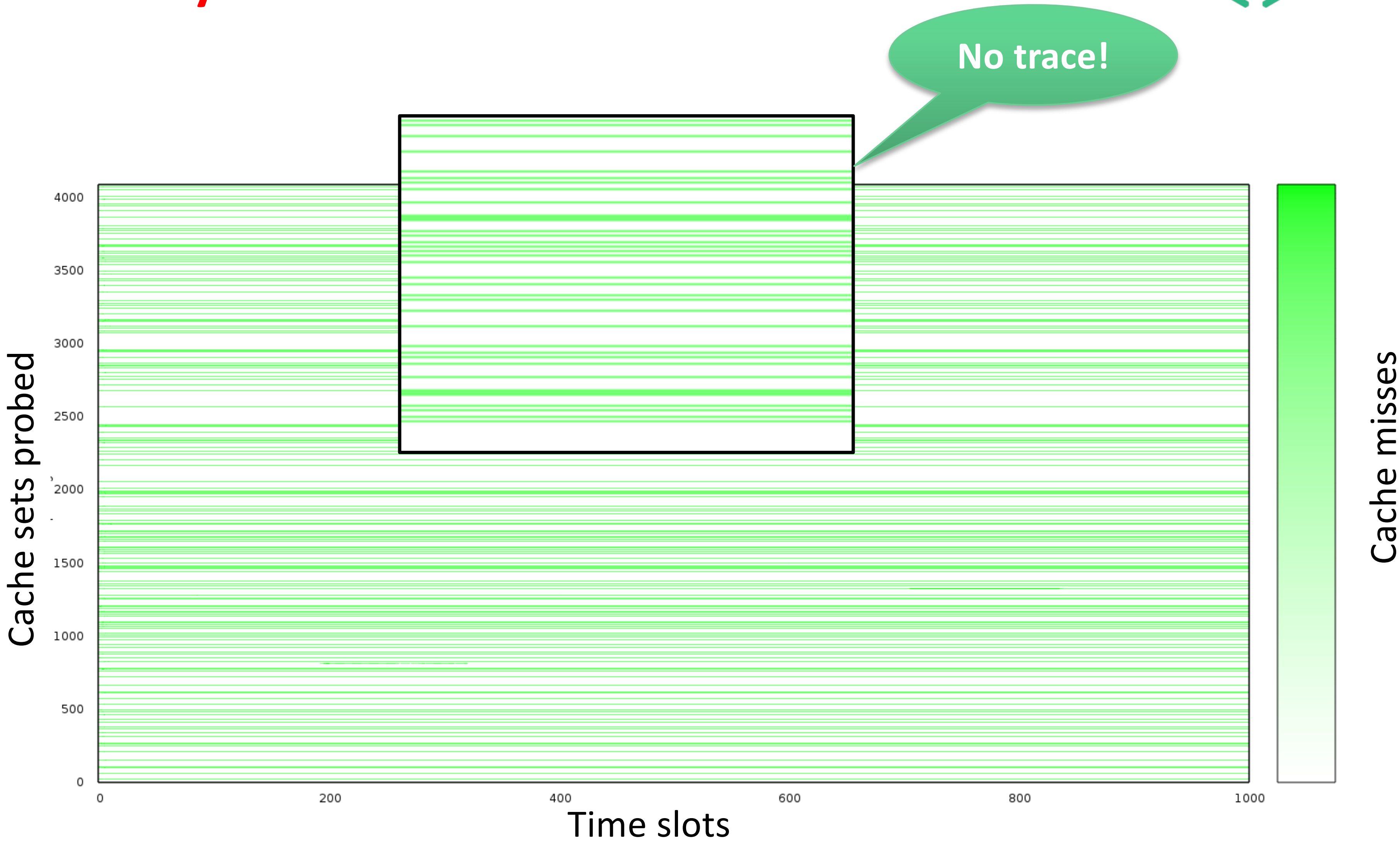
Colouring the System is Easy

System permanently coloured



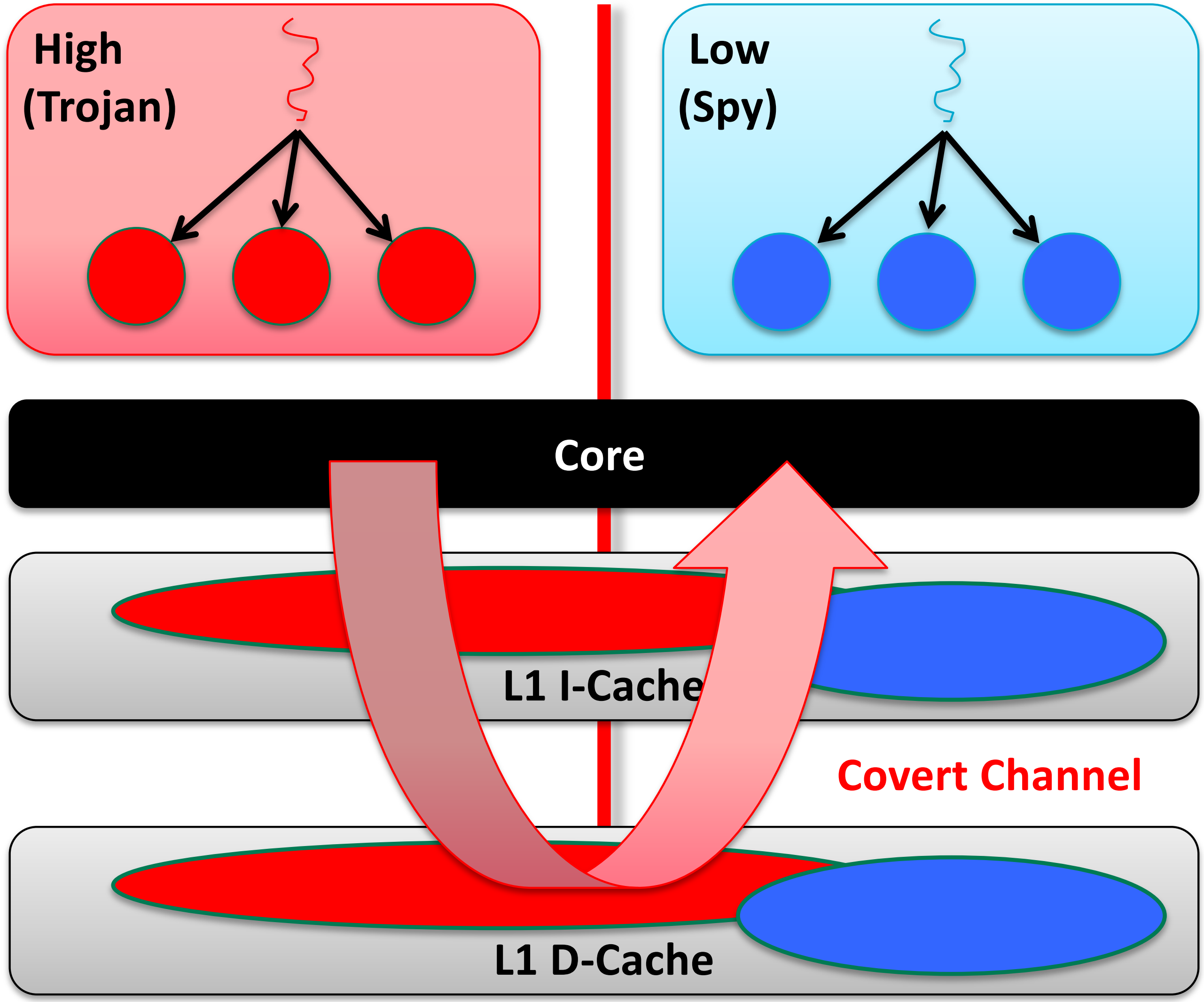
LLC Timing Side Channel Attack on seL4

Coloured System



Covert-Channel Scenario: Intra-Core L1 Attack

Covert-channel attack on time-shared core



L1 D-Cache Covert Channel

High (Trojan)

```
int count = 0;
for( ; ;) {
    wait_for_new_system_tick( );
    if (count & 4)
        access(L1_cache_buffer)
    count++;
```

Patterns of using L1 D

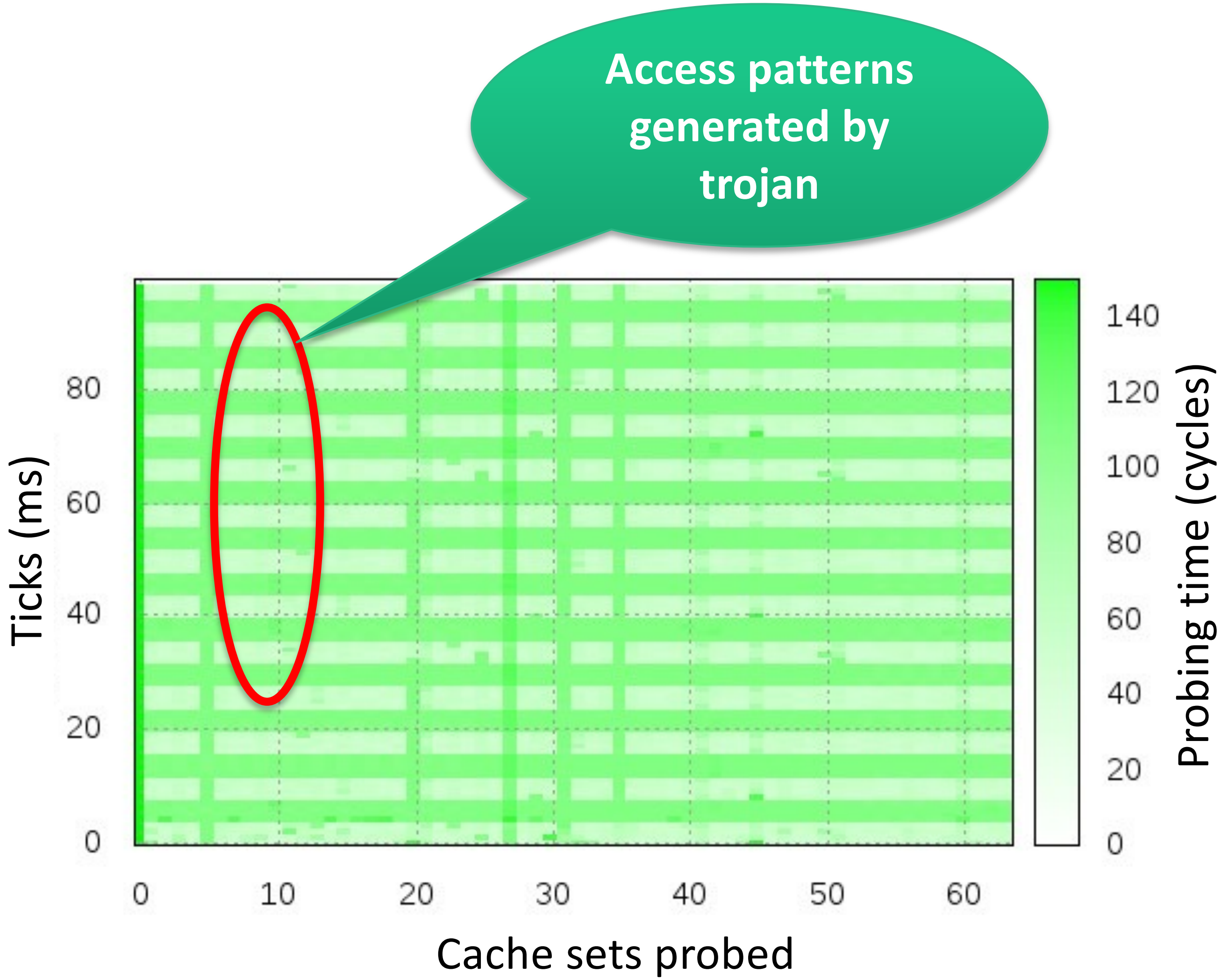
Low (Spy)

```
for( t = 0; t < 100 ; t++) {
    wait_for_new_system_tick( );
    probe(L1_cache_buffer)
}
```

Probing for 100 ticks

L1-D Cache Covert Channel

Spy observations on ARM A9



Challenge: L1 Cache Cannot Be Coloured

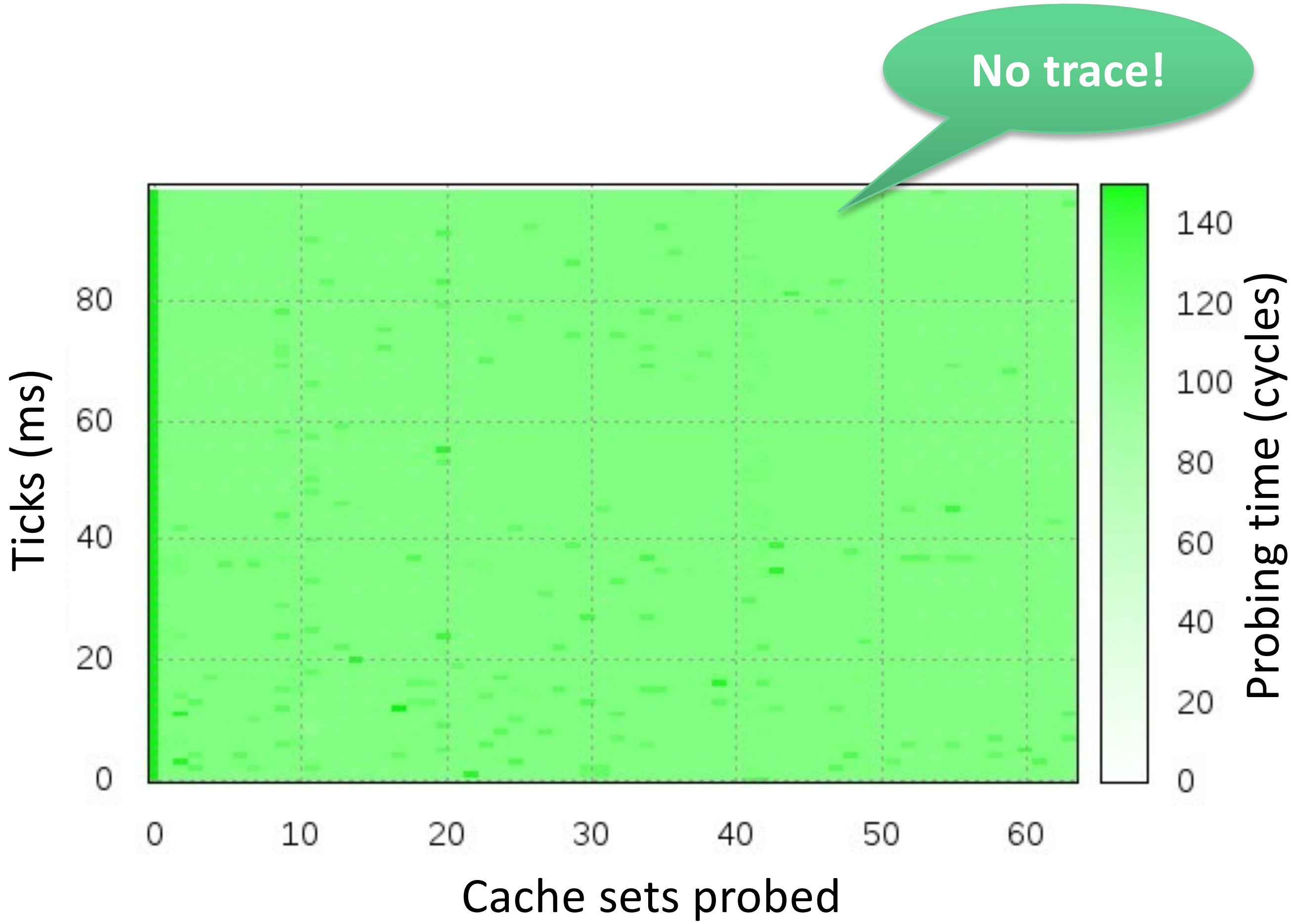
- L1 is virtually addressed ⇒ layout is not under OS control
- On most processors, L1 is too small (single colour)
- Even if it could be partitioned, performance cost would be high

Solution: Flush L1 on context switch

- Direct cost low (1–2 µs)
- Indirect cost negligible:
  - Done only done on partition switch (≥ 1ms)
  - No hot data in L1 anyway after two partition switches
- Pain on x86: no selective L1 flush instruction
  - Need to flush by walking a buffer

Unsafe: makes assumptions on line-replacement policy of cache

L1-D Cache Covert Channel
Spy observations with L1 flush on ARM A9



L1 I-Cache Covert Channel

Probing for 100 ticks

High (Trojan)

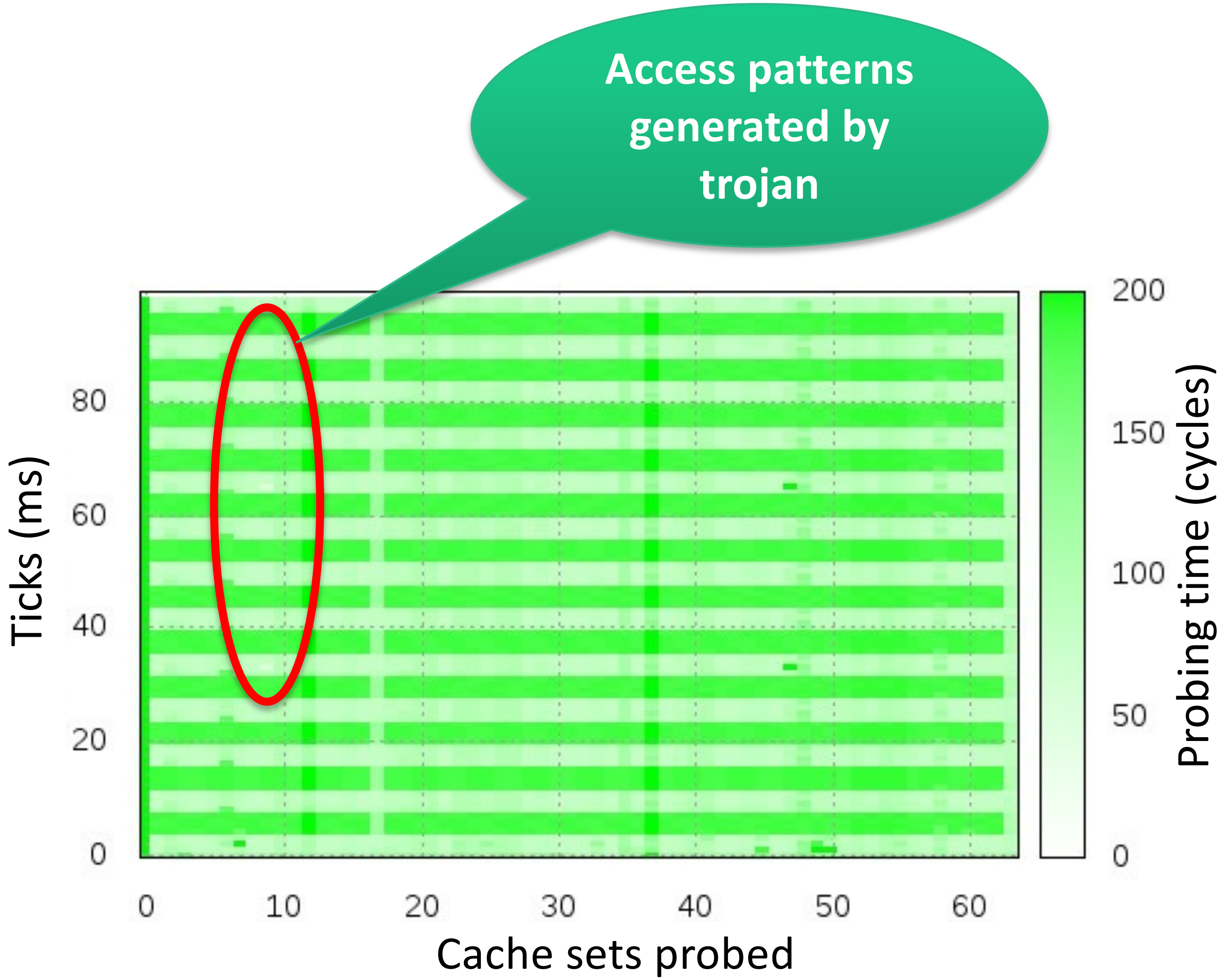
```
int count = 0;
for( ; ;) {
    wait_for_new_system_tick( );
    if (count & 4)
        jump(L1_cache_buffer)
    count++;
```

Low (Spy)

```
for( t = 0; t < 100 ; t++) {
    wait_for_new_system_tick( );
    jump_probe(L1_cache_buffer)
}
```

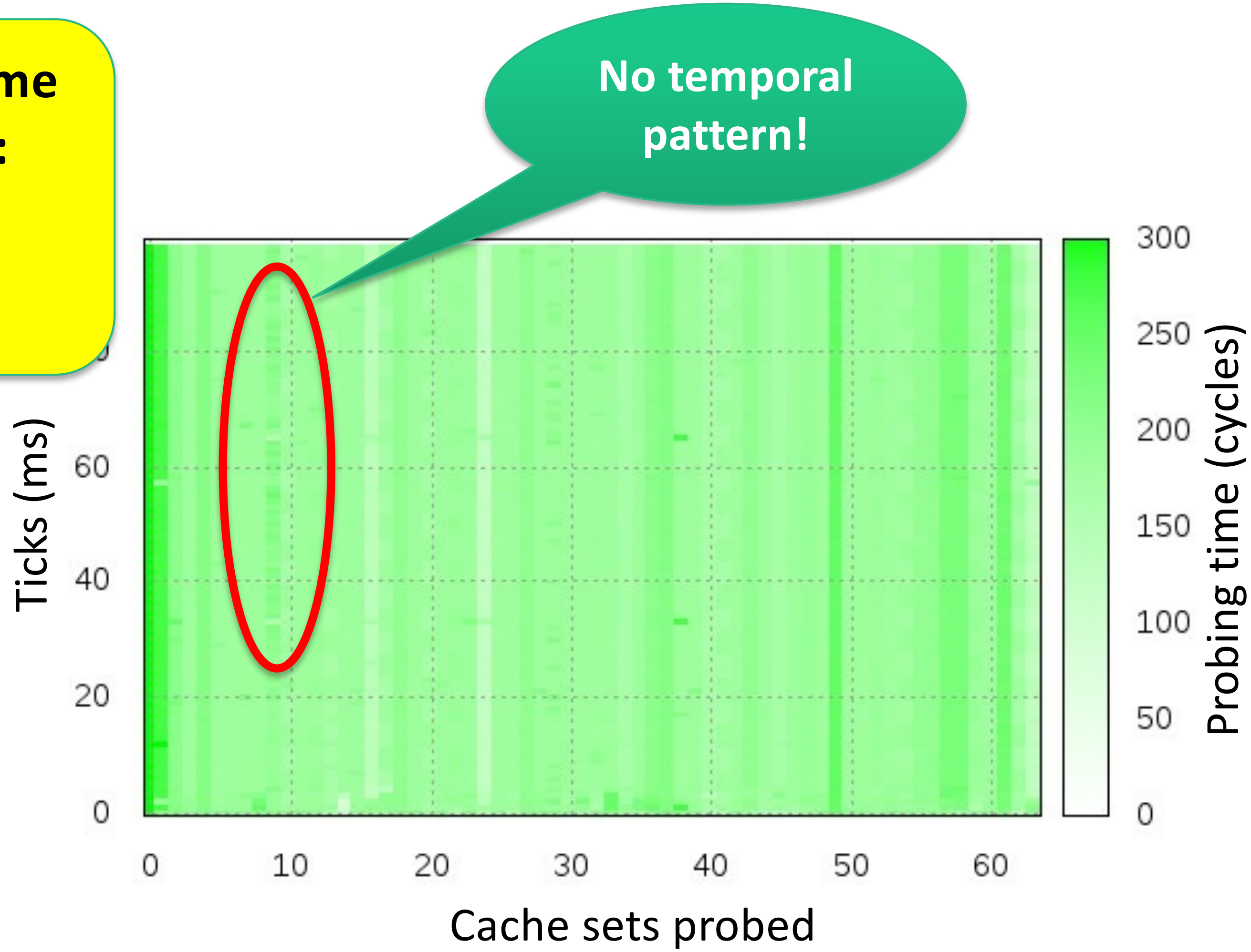
Patterns of using L1 I

L1-I Cache Covert Channel
Spy observations on ARM A9



L1-I Cache Covert Channel
Spy observations with flush on ARM A9

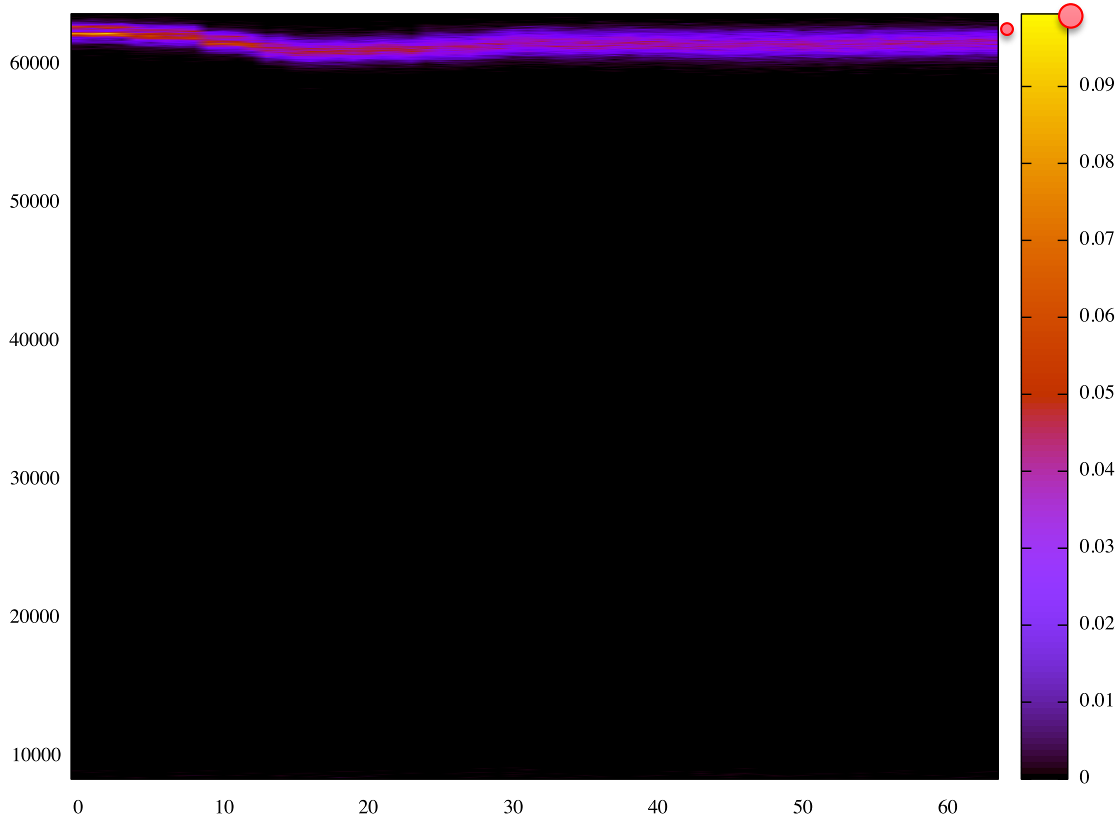
Can play same games with:
- TLB
- BPU



Recent Intel Processors

- Approach: complete (very expensive) cache flush,
- Flush everything the hardware lets you flush

Channel remains, no way to close!



Discussion

Security & safety are a losing game!

- The ISA is no longer a sufficient contact for safety **or** security
  - Software designers are left to second-guess micro-architecture
  - Failures are predictable (and demonstrable)
  - The ISA over-abstracts hardware
- Safety and security requires an extended ISA

Need new hardware-software contract!

- For security:
  - Must explicitly identify all shared hardware state
  - All shared state must be either partitionable or flushable
  - Architecture must provide explicit flush of non-partition able state
- For safety, need sufficient information to bound execution latencies
  - Well-defined cache behaviour
  - **Usable** model of instruction latencies
    - Just upper bounds are too pesimistic
  - Bus access latencies bounded or software-manageable

Thank you

Gernot Heiser | gernot.heiser@data61.csiro.au| @GernotHeiser

Dagstuhl, October 2016

https://trustworthy.systems

Machine Specifications for Experiments

Processor Model	i7-2600
Microarchitecture	Sandy Bridge
Clock Frequency	3.4 GHZ
# of Cores	4
LLC	8,192 KiB (16 ways)
Cache line size	64 B
L1 Caches	2 × 32 KiB (I & D)
OS	seL4