



DATA  
61

# Security vs Manufacturers

## A Losing Battle

Gernot Heiser

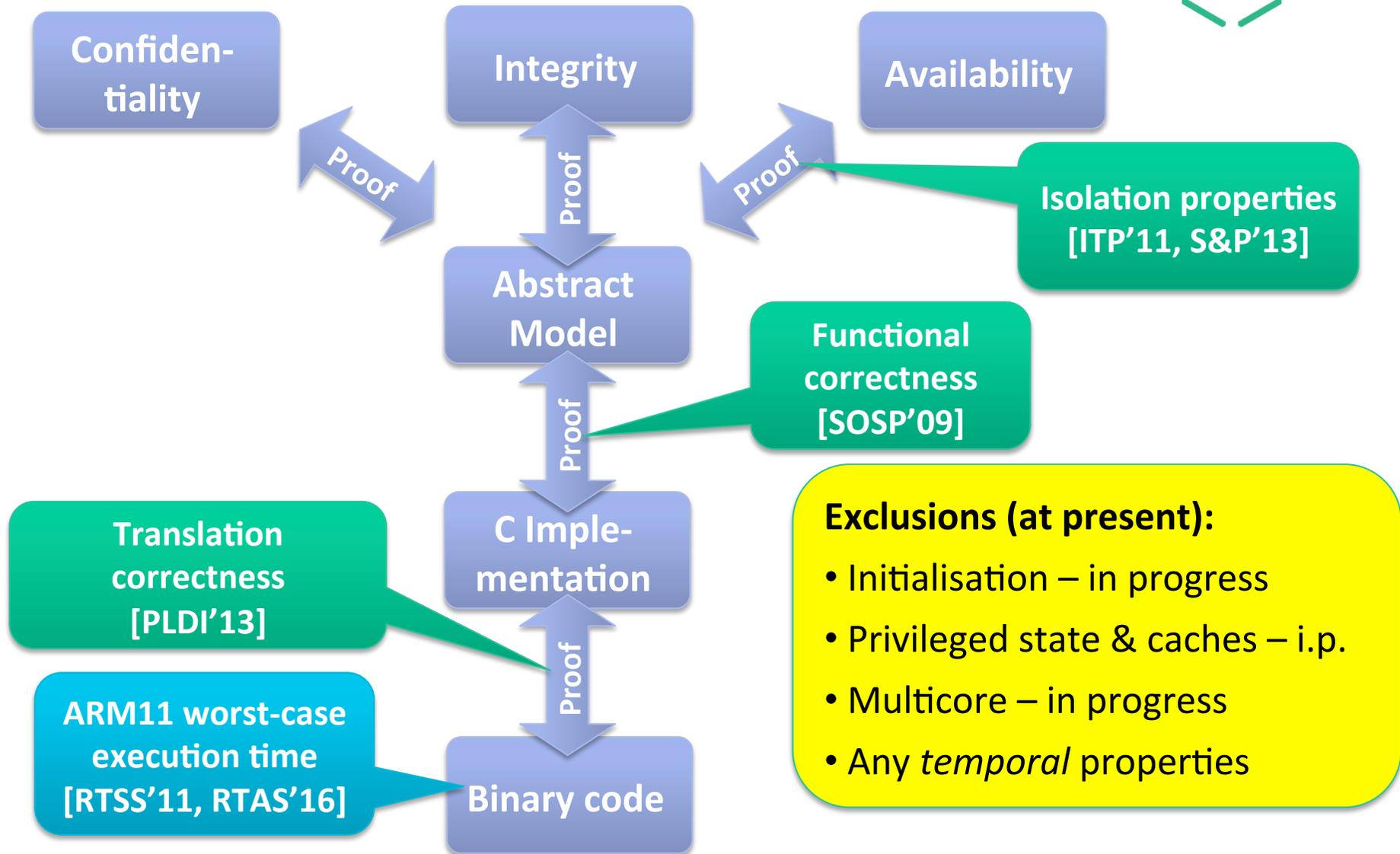
gernot.heiser@data61.csiro.au | @GernotHeiser

IFIP WG10.4, January 2017

<https://trustworthy.systems>



# seL4 Spatial Isolation: Solved



# se14 Critical Systems: DARPA HACMS



Boeing Unmanned Little Bird

Retrofit existing system!



US Army Autonomous Trucks



SMACCMcopter Research Vehicle

Develop technology



TARDEC GVRbot

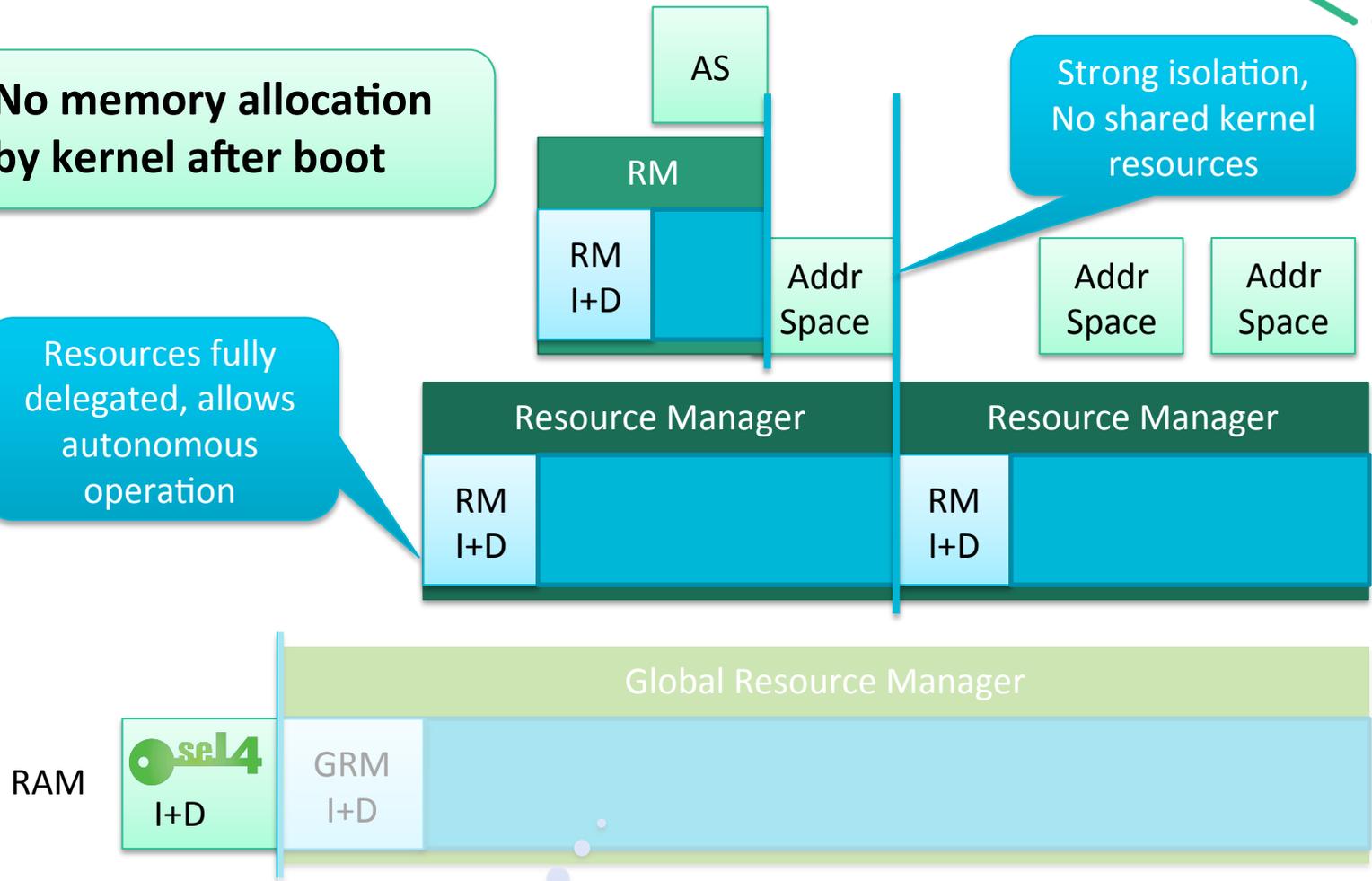
# sel4 Design for Isolation



No memory allocation by kernel after boot

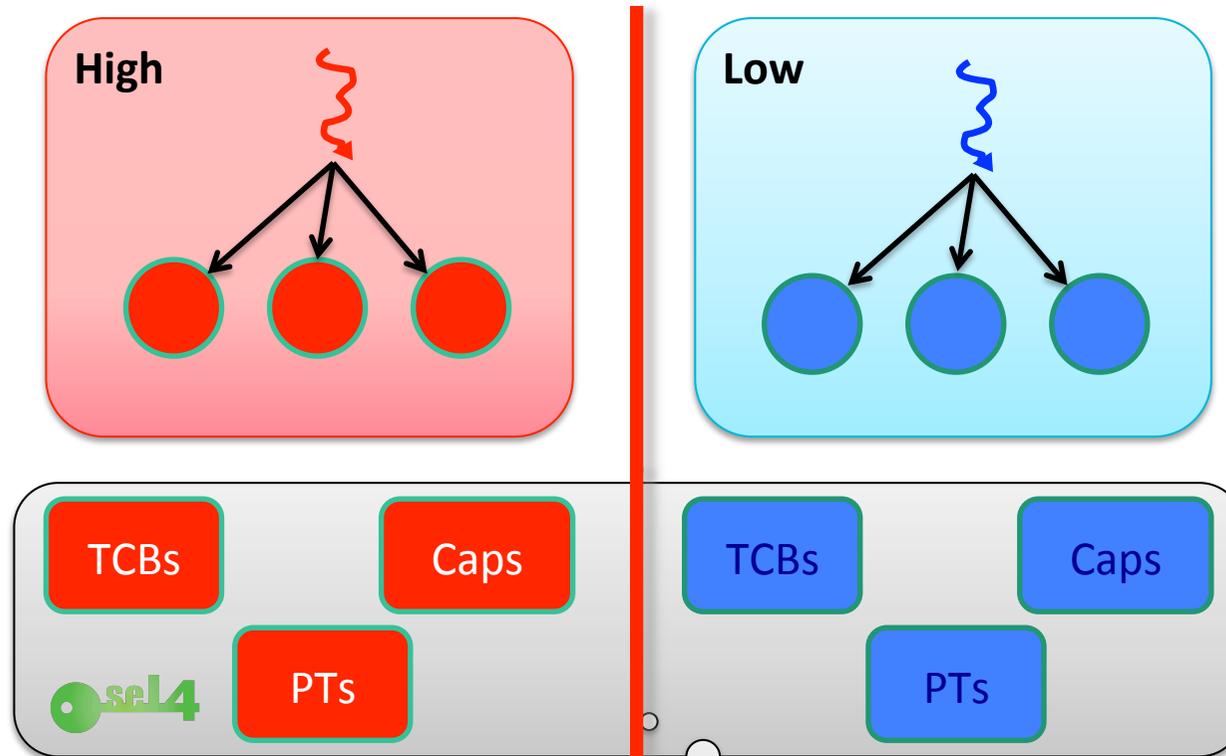
Resources fully delegated, allows autonomous operation

Strong isolation, No shared kernel resources



“Untyped” (unallocated) memory

# sel4 Isolation Goes Deep



Provable freedom from storage channels!

Kernel data partitioned like user data

DATA  
61



# Time – The Final Frontier



# Two Aspects of Temporal Isolation

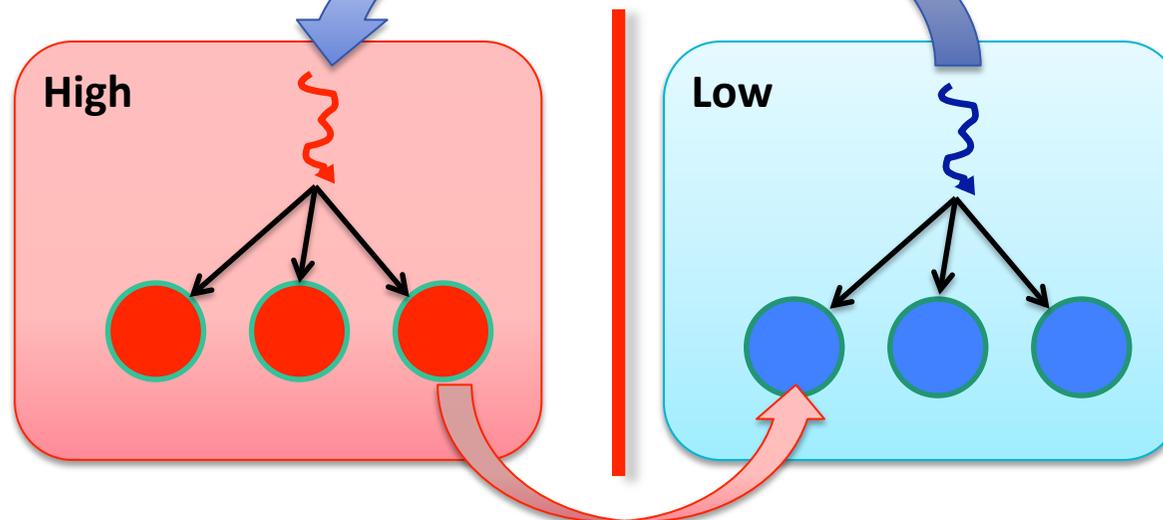
## Safety: Timeliness

- *Bound* execution interference

## Security: Confidentiality

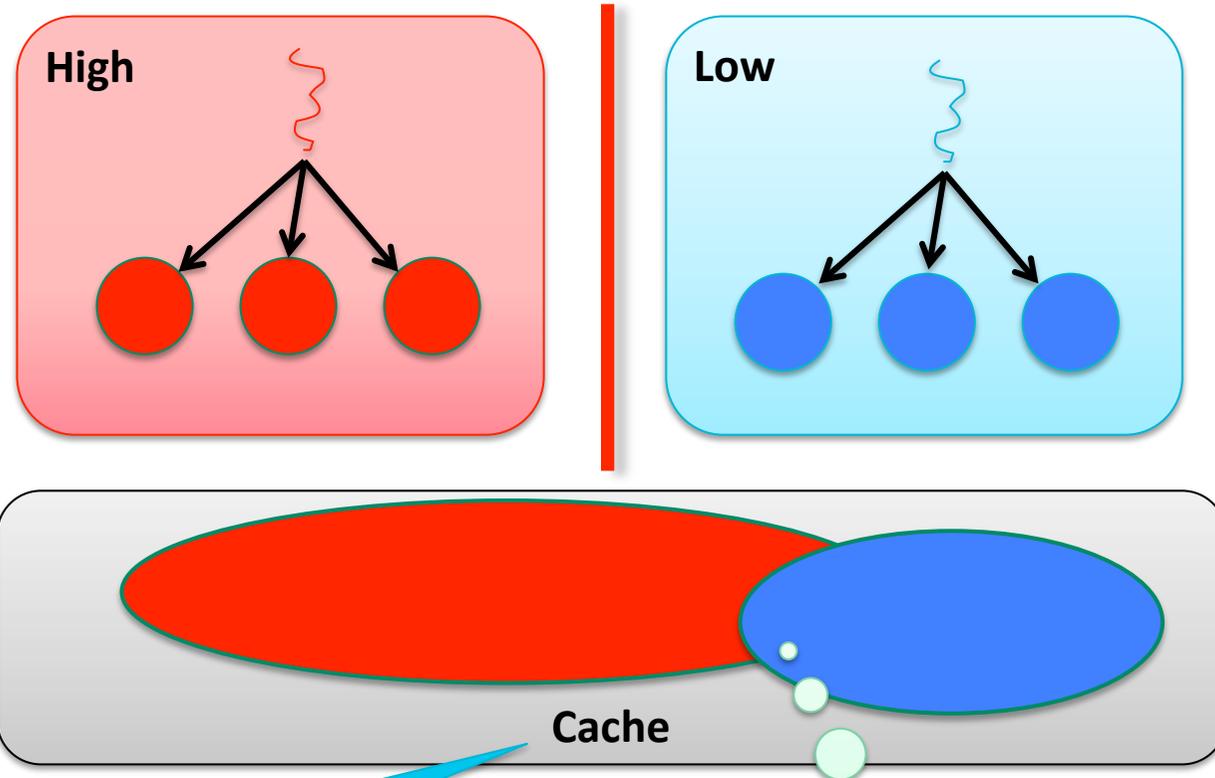
- *Prevent* leakage via timing channels

Affect execution speed:  
Integrity violation



Observe execution speed:  
Confidentiality violation

# Timing Channels



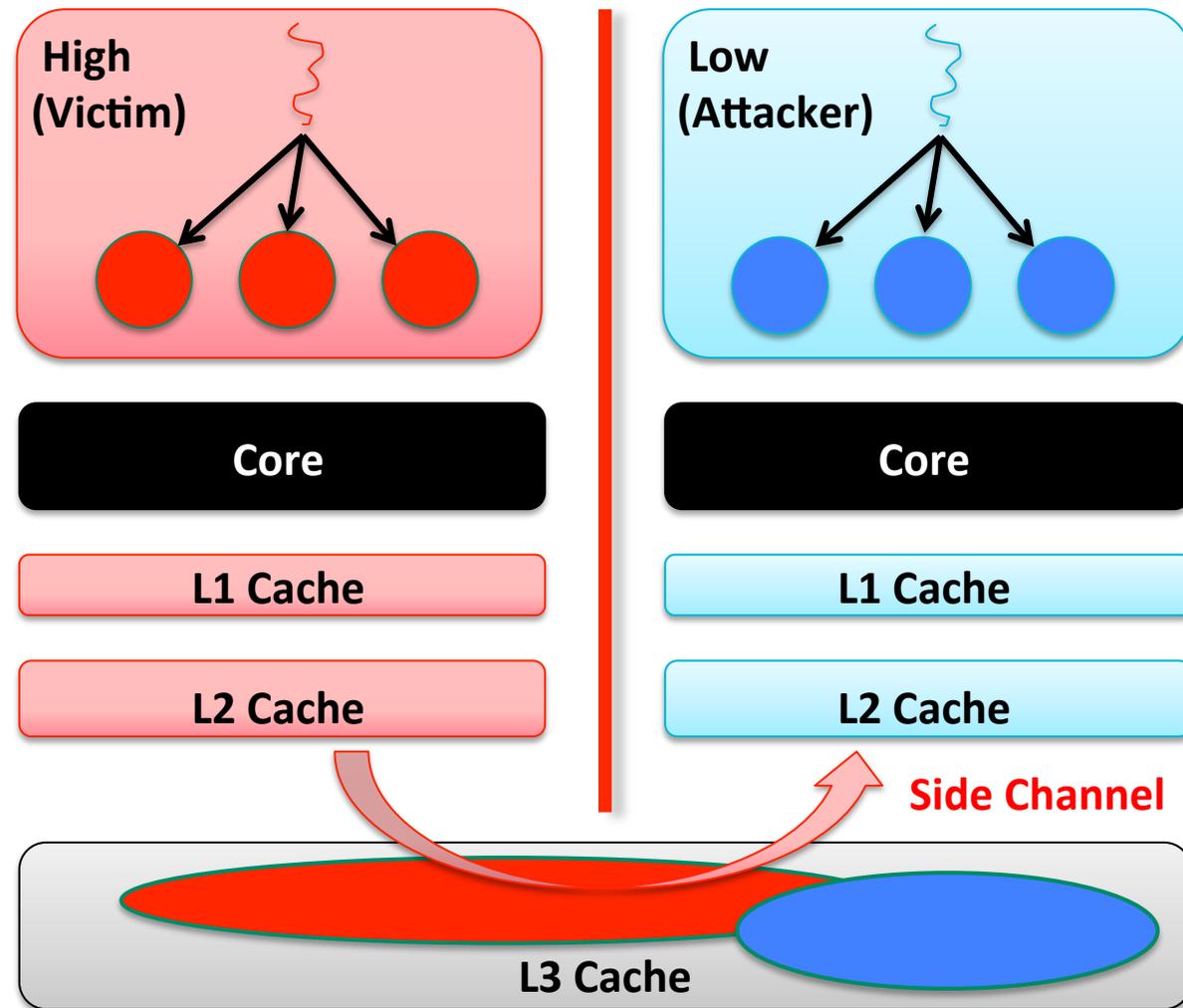
“Cache” might be:

- 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

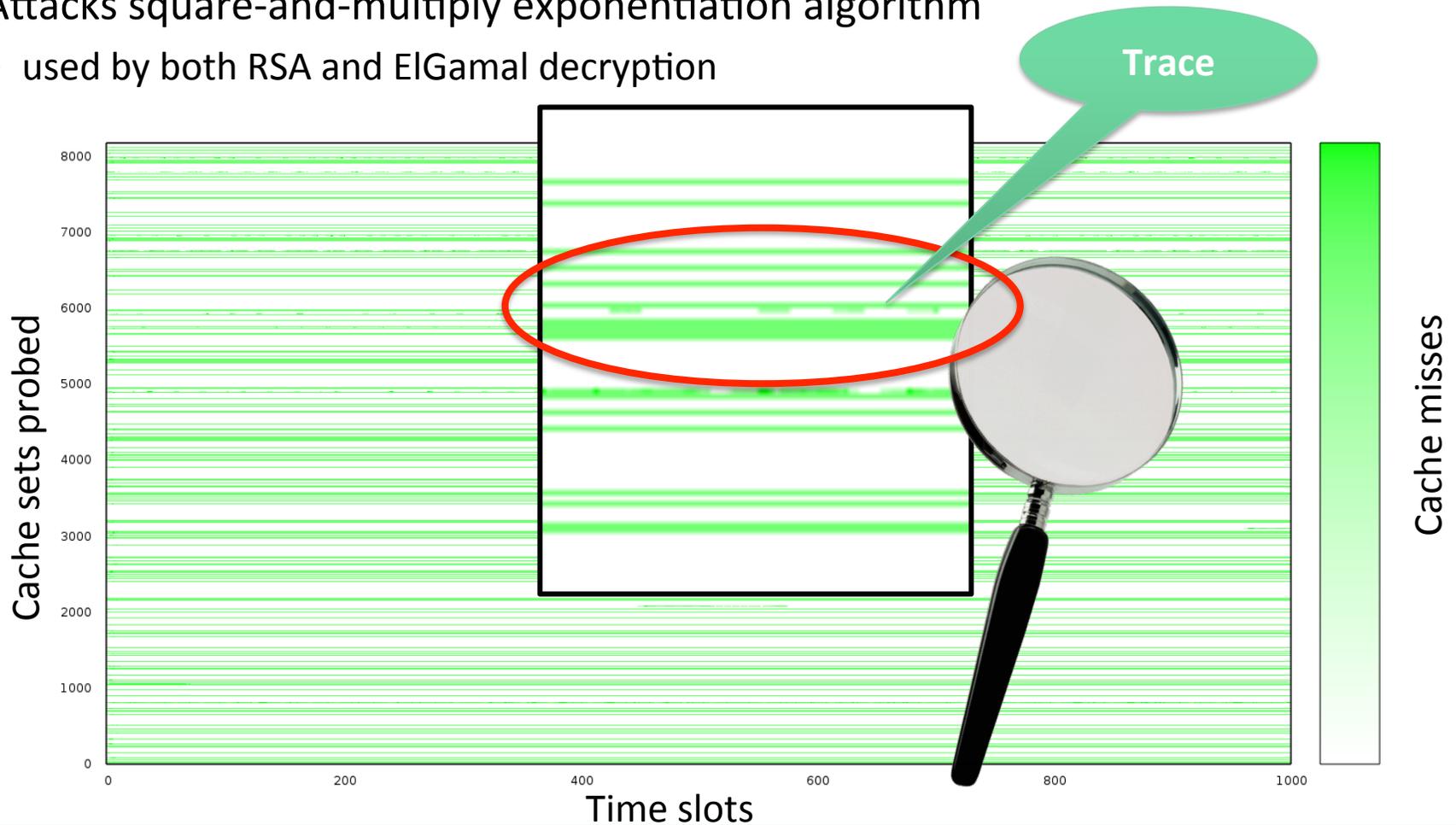


# Cross-Core LLC Timing Side Channel

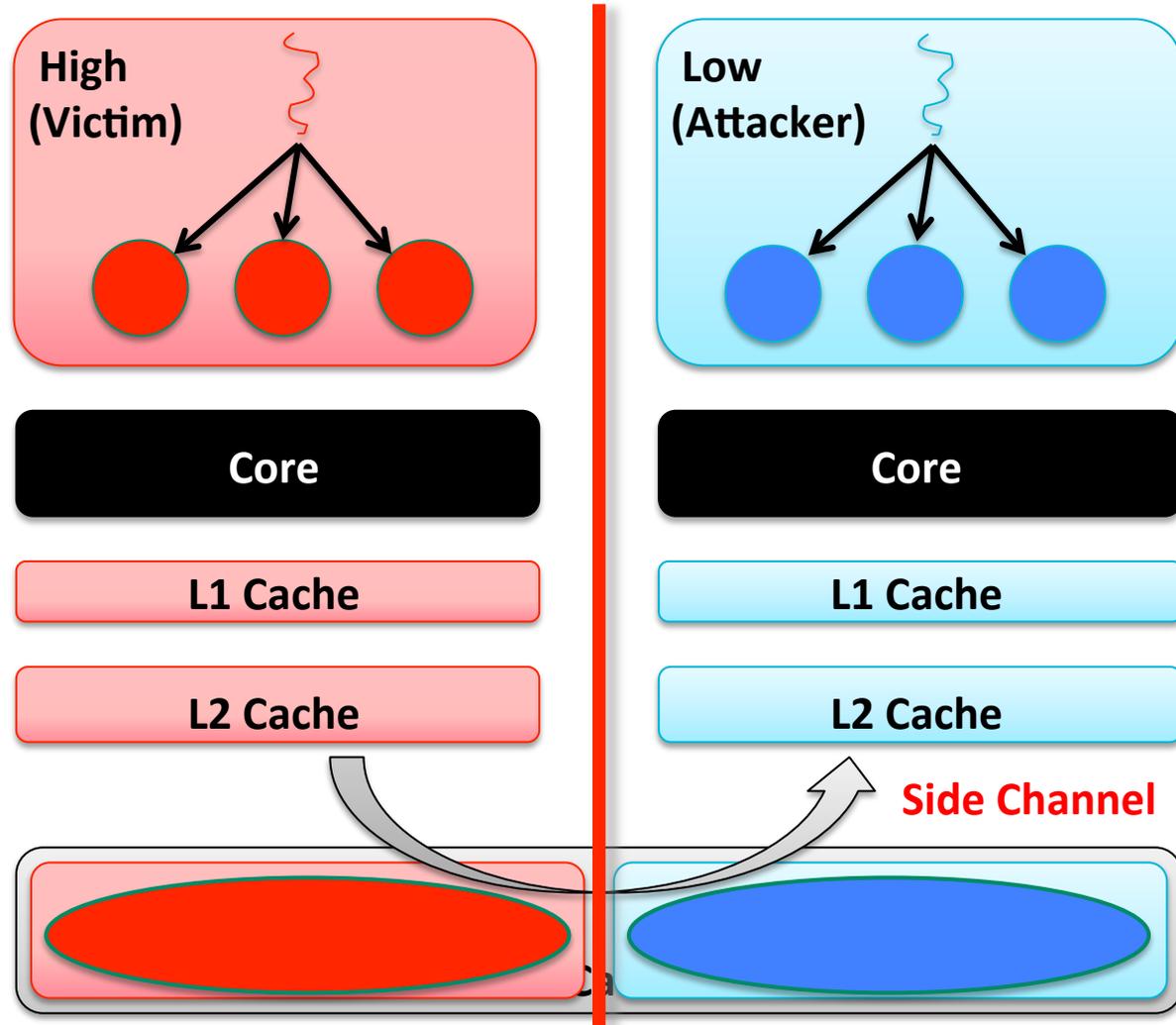
[Liu et al, Oakland'15]



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



# Mitigation: Partition Cache (Colouring)

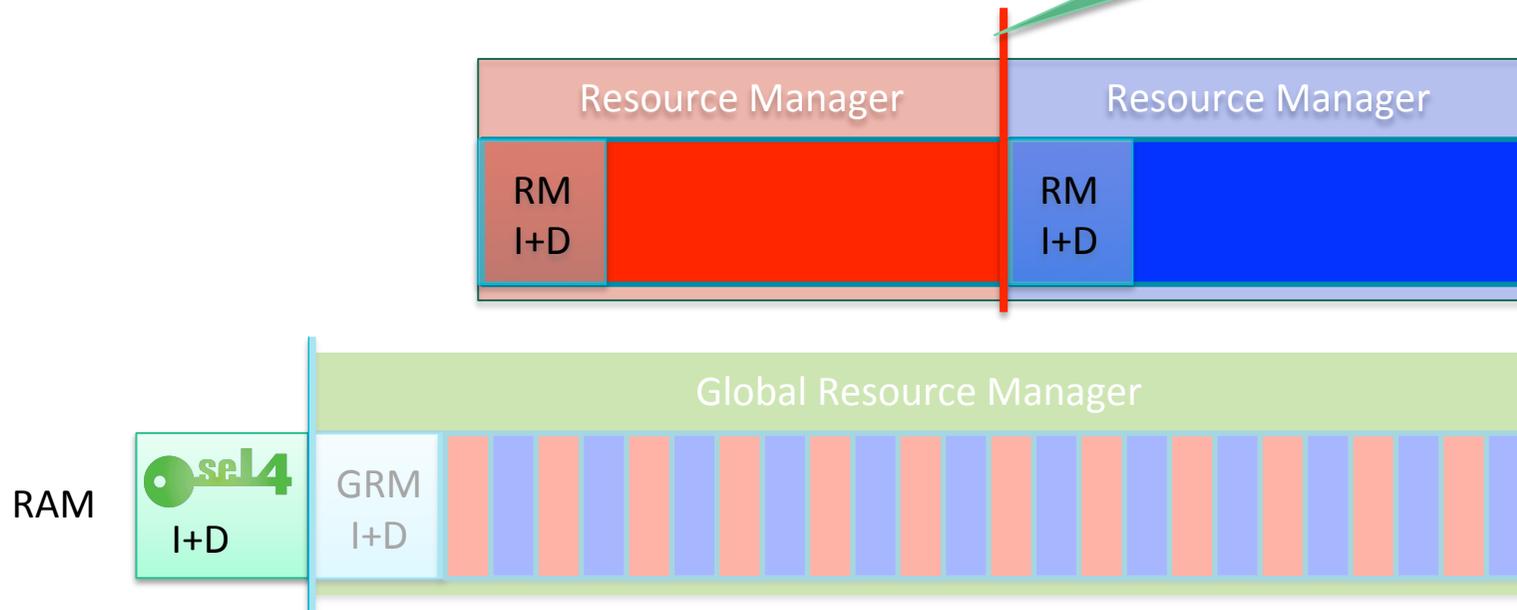


# sel4 Colouring the System is Easy



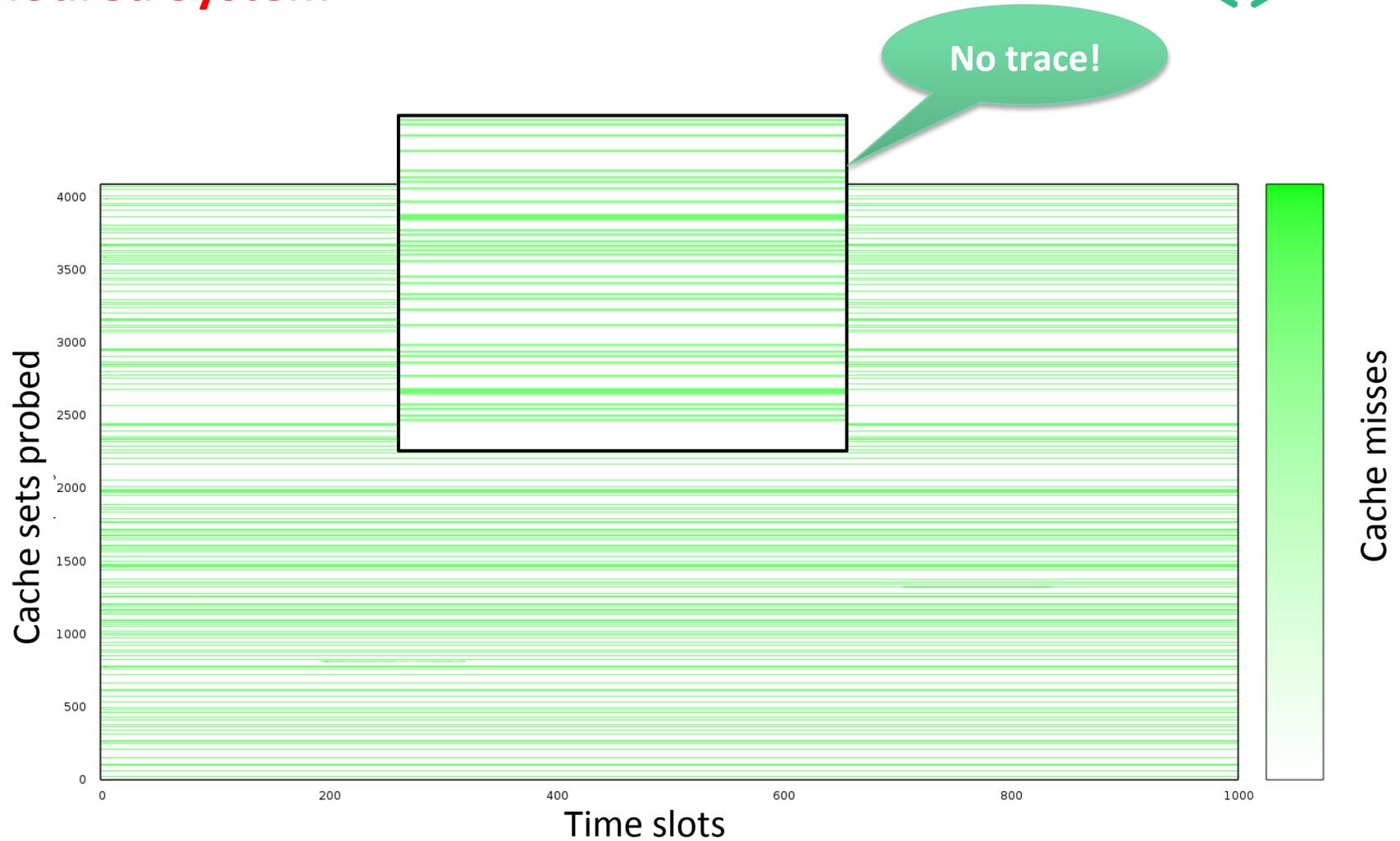
System permanently coloured

Partitions restricted to coloured memory



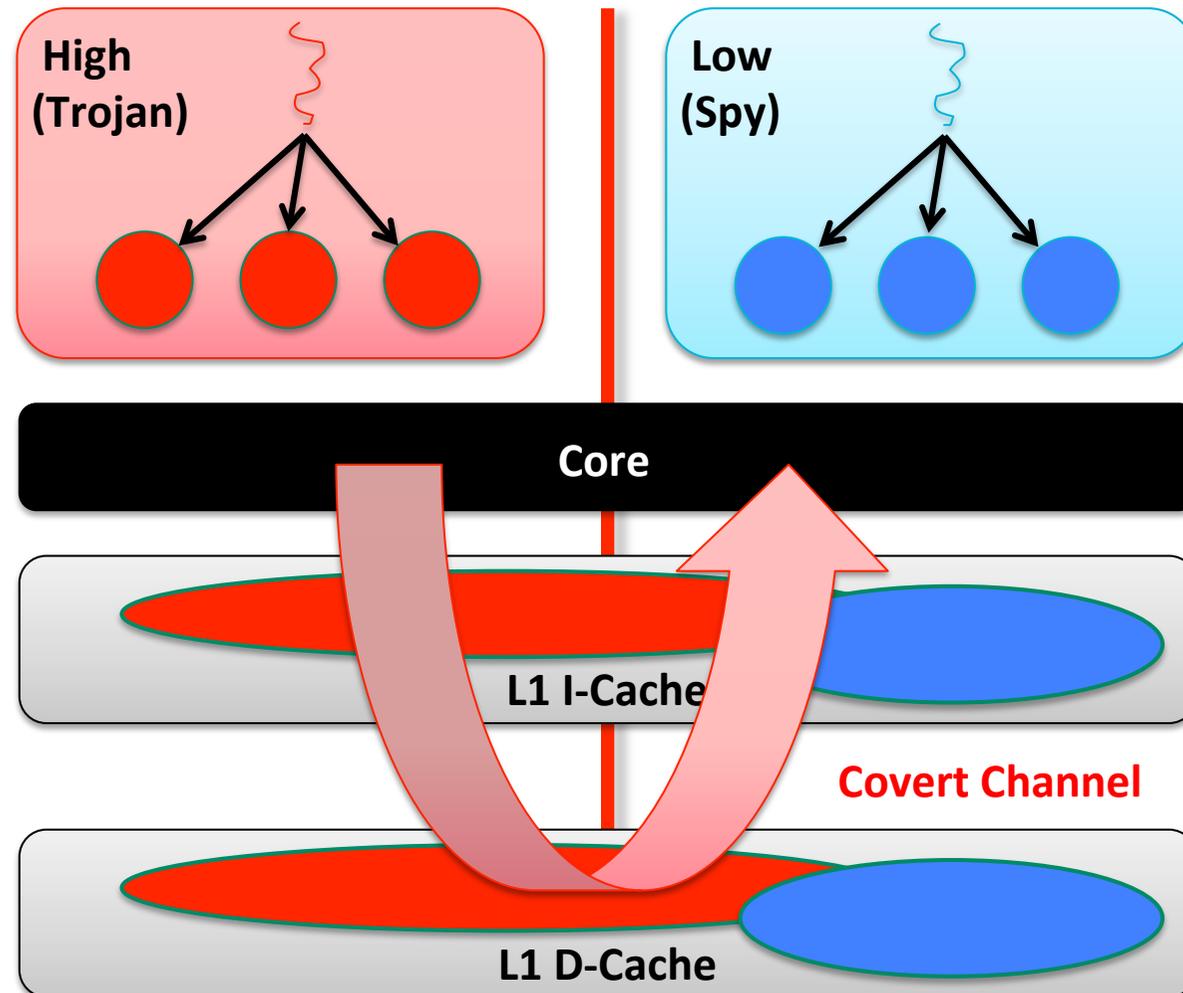
# LLC Timing Side Channel Attack on seL4

## 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

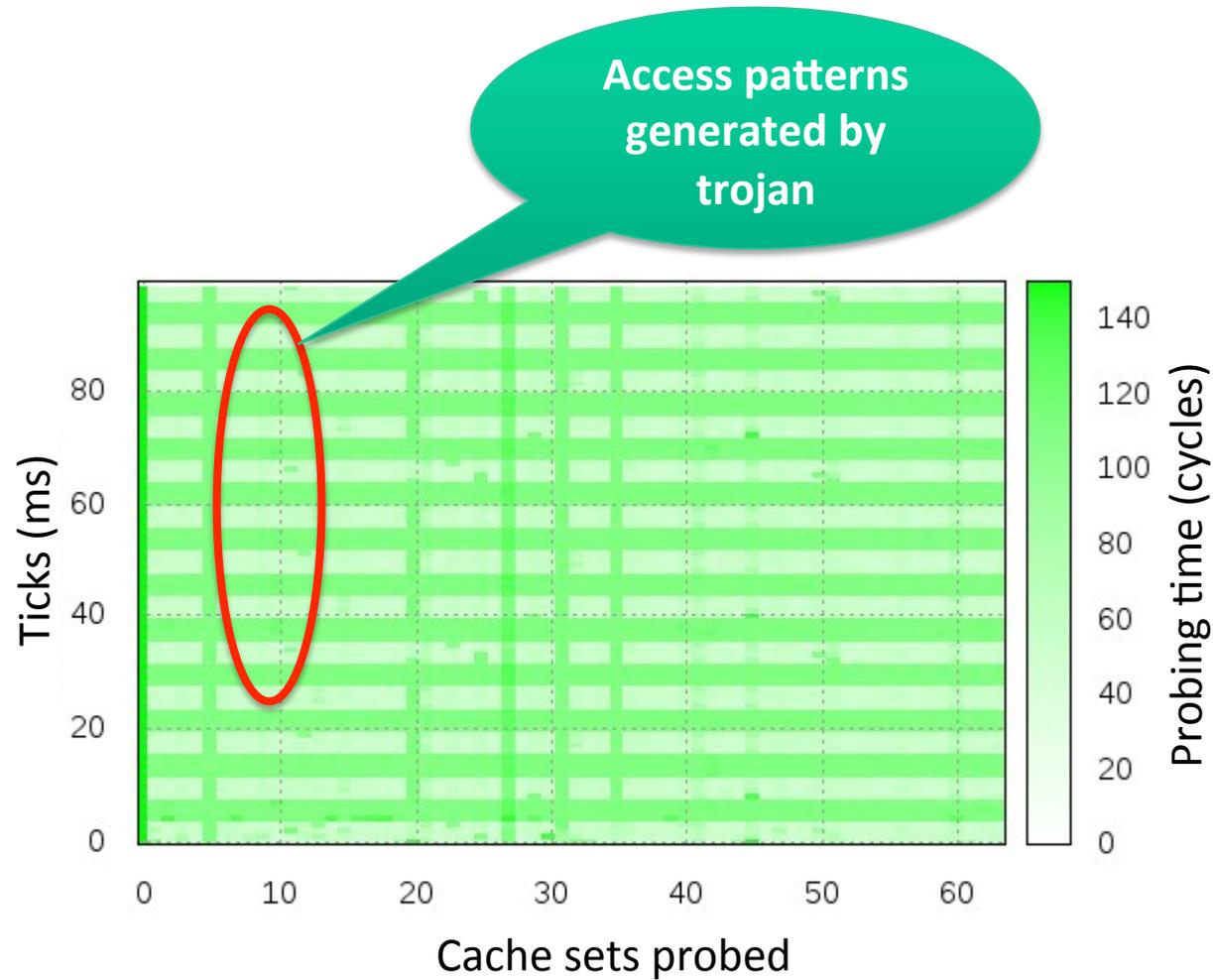
## Low (Spy)

```
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  $\Rightarrow$  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

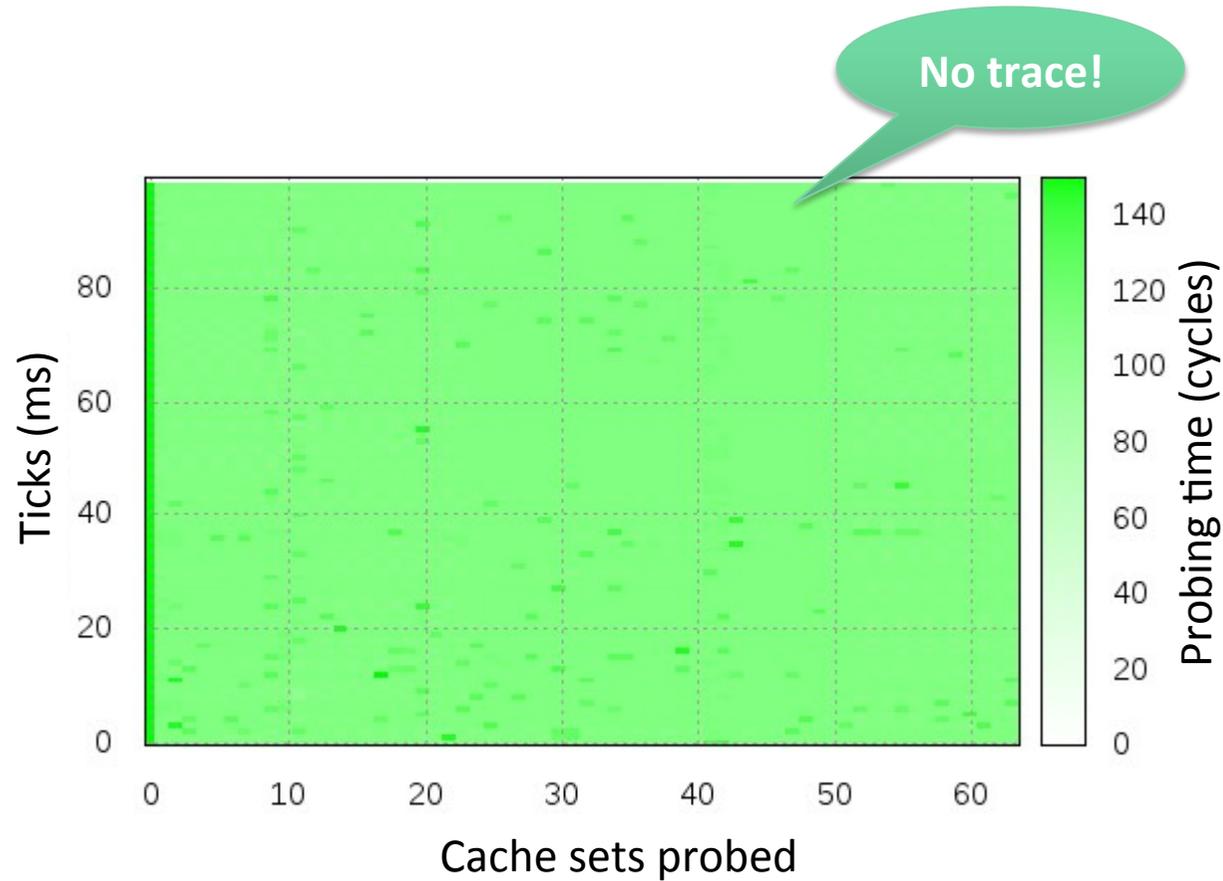
- Direct cost low (1–2  $\mu$ s)
- Indirect cost negligible:
  - Done only done on partition switch ( $\geq 1$ ms)
  - No hot data in L1 anyway after two partition switches
- Pain on x86: no selective L1 flush instruction
  - Need to flush by walking a buffer

Unsafe: makes assumptions on line-replacement policy of cache

A pink thought bubble with a red outline and three smaller circles leading to it from the left. The text inside is bold and black.

# L1-D Cache Covert Channel

## Spy observations with L1 flush on ARM A9



# L1 I-Cache Covert Channel



## High (Trojan)

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

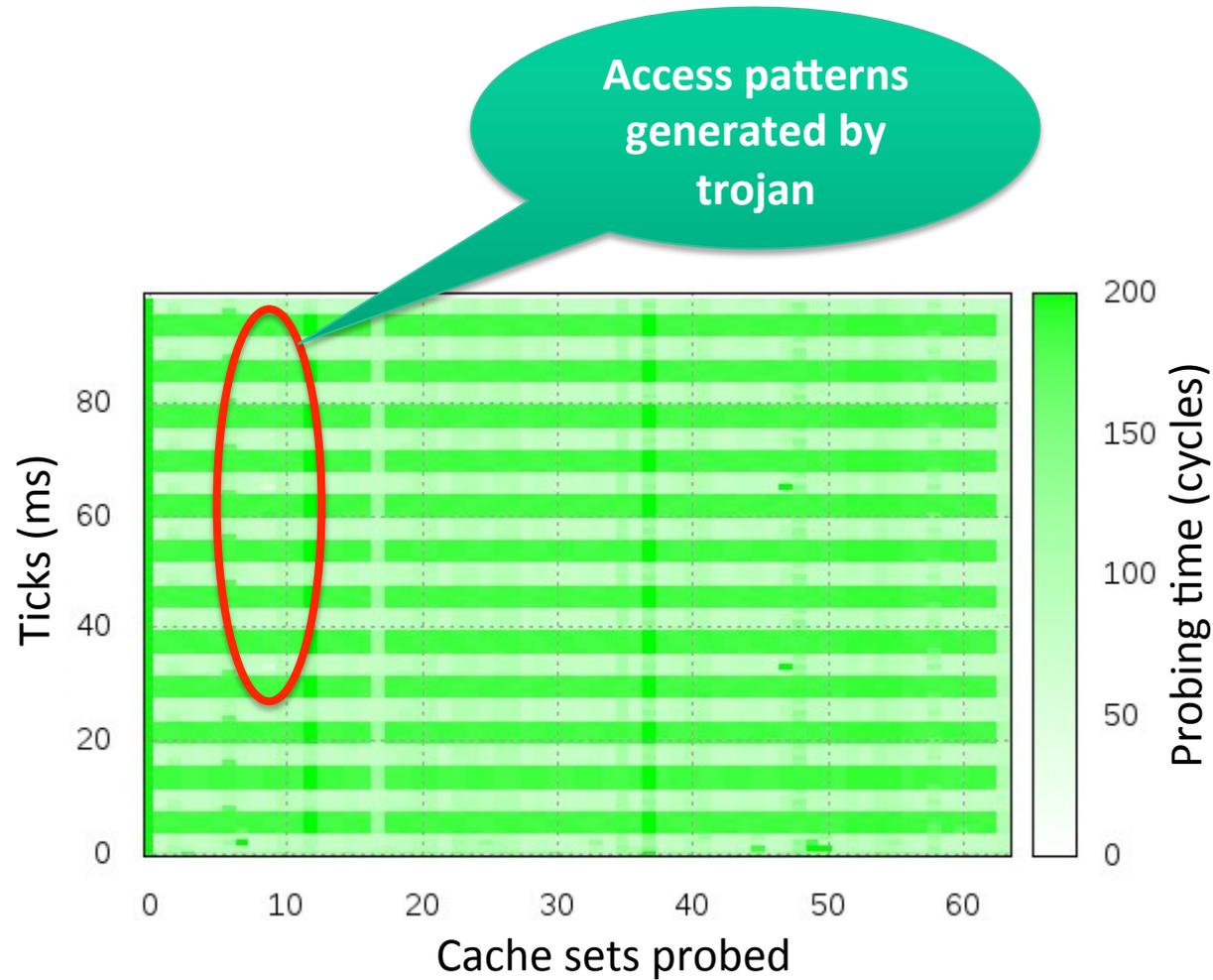
Patterns of using L1 I

## Low (Spy)

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

Probing for 100 ticks

# L1-I Cache Covert Channel Spy observations on ARM A9



# L1 I-Cache Covert Channel

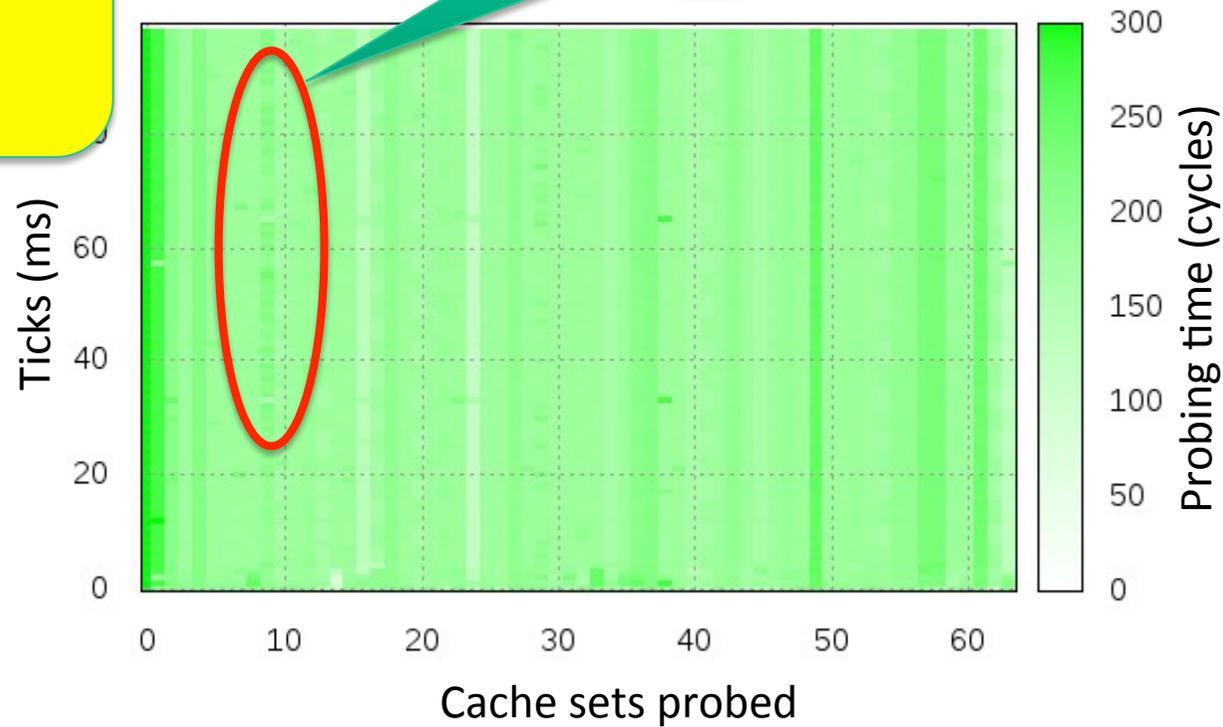
## Spy observations with flush on ARM A9



Can play same games with:

- TLB
- BPU

No temporal pattern?



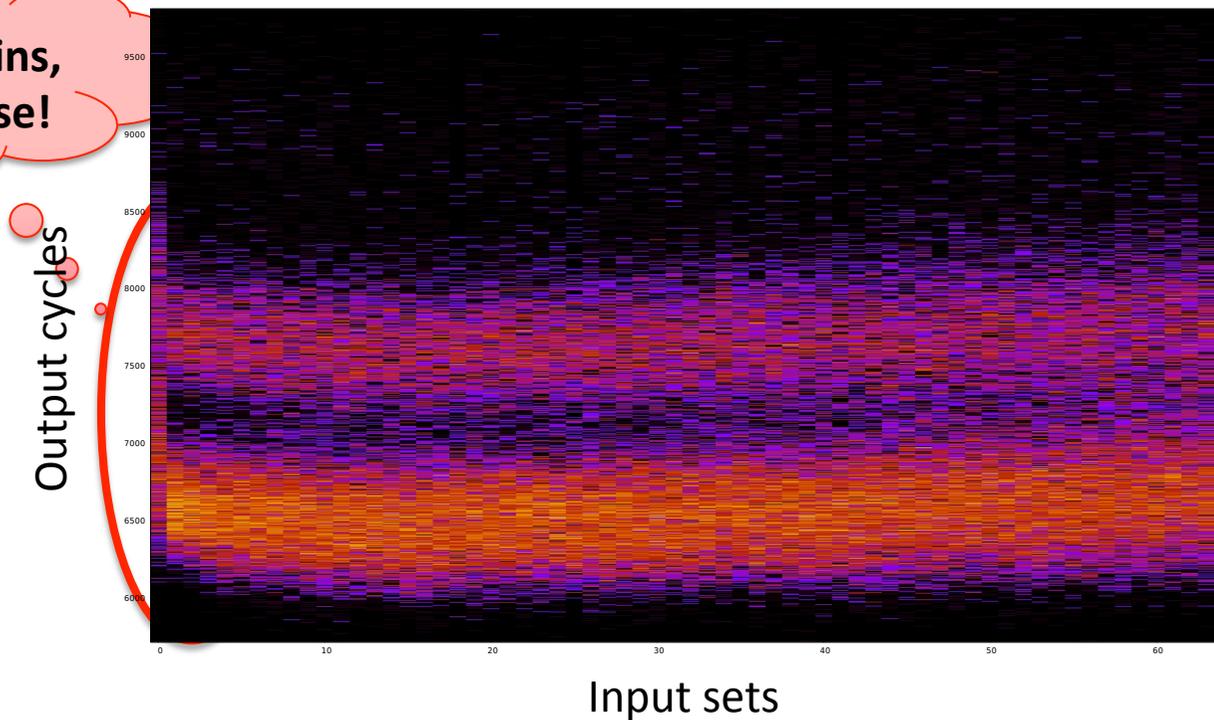
# Recent Intel Processor: Skylake I-Cache



## Approach:

- Do everything the hardware lets you (expensive!)
- Flush L1, L2, L3, TLB, BPU
- Disable prefetcher

Channel remains,  
no way to close!



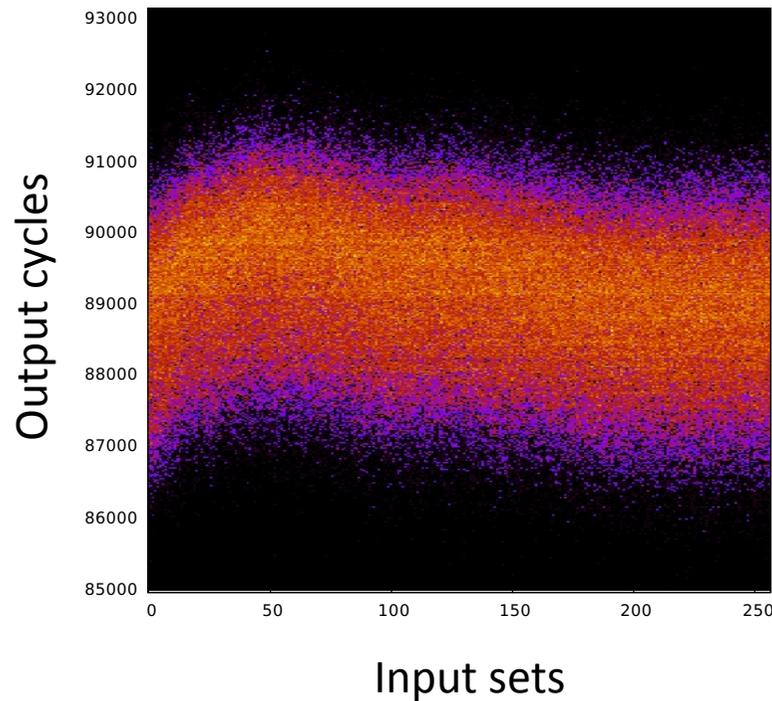
# Recent ARM Processor: A53 I-Cache



## Approach:

- Do everything the hardware lets you (expensive!)
- Flush L1, L2, L3, TLB, BPU
- Disable prefetcher

Channel remains,  
no way to close!



# Discussion



## Security seems a losing game!

- The ISA is no longer a sufficient contract for security
  - Software designers are left to second-guess micro-architecture
  - Failures are predictable (and demonstrable)
  - The ISA over-abstracts hardware
- Security requires an extended ISA:
  - Must explicitly identify all shared hardware state
  - All shared state must be either partitionable or flushable
  - Architecture must provide explicit flush of all non-partitionable state

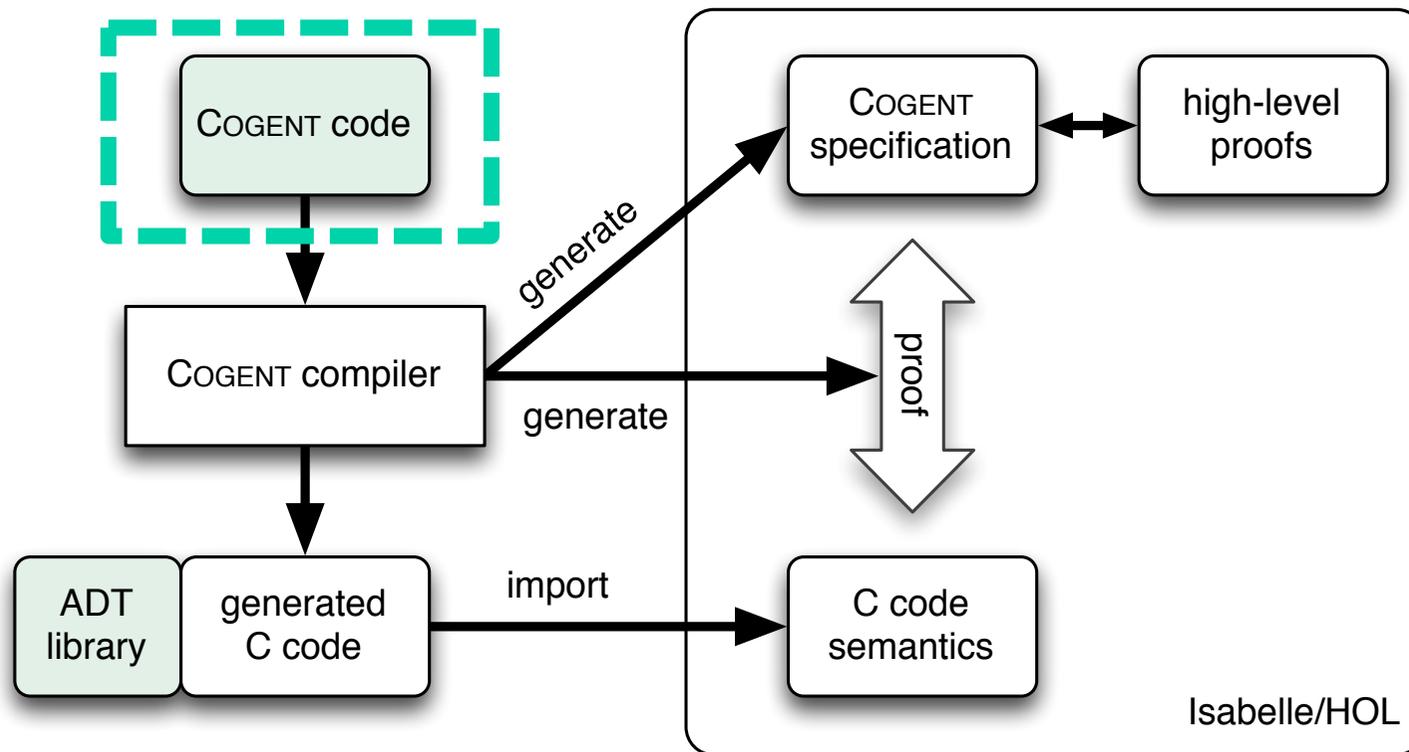
Need new hardware-  
software contract!



# On a Different Note

# Cogent workflow

- Cogent: purely functional memory-safe language





DATA  
61

Thank you

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

<https://trustworthy.systems>

