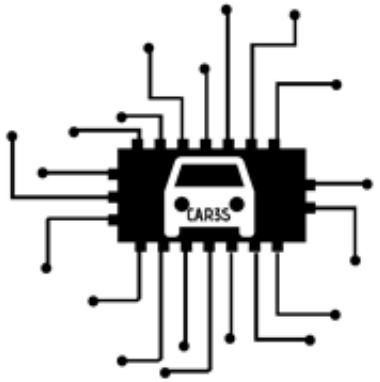
@WINTER SCHOOL

@CSE-IIT Kanpur*

22nd December 2020

*Joining IIT Bombay in May 2021



MICROARCHITECTURE-SECURITY@WINTER SCHOOL

@CSE-IIT Kanpur*

22nd December 2020

*Joining IIT Bombay in May 2021

Disclaimer

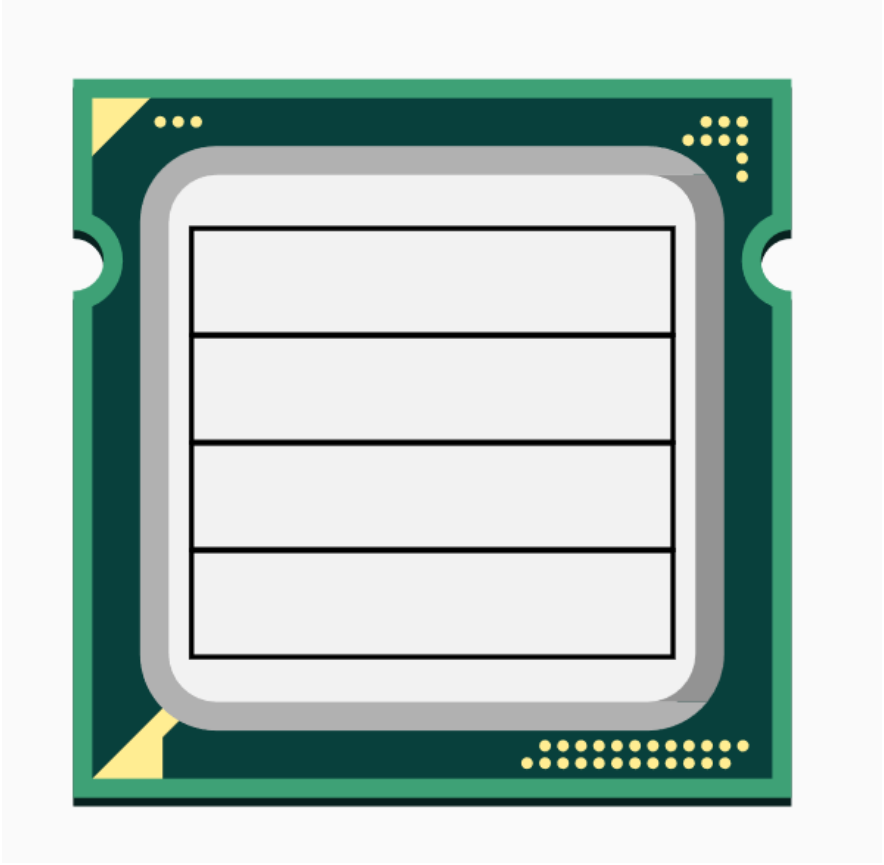
Talk is not about leaking information from your machines through MS teams.

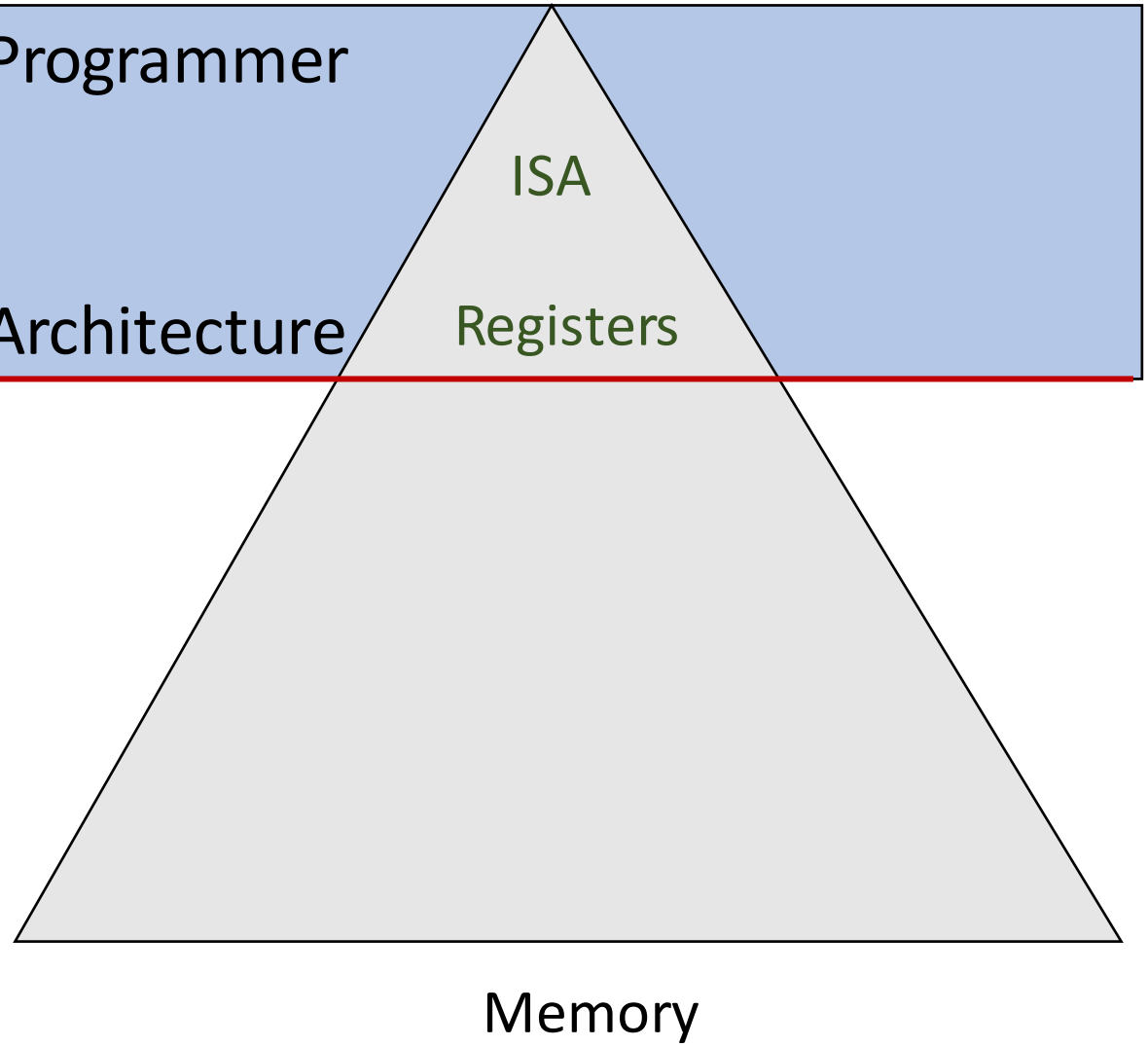
I am not trying for the same. Trust me ☺

After the talk, if your password is compromised, It is not me ☺

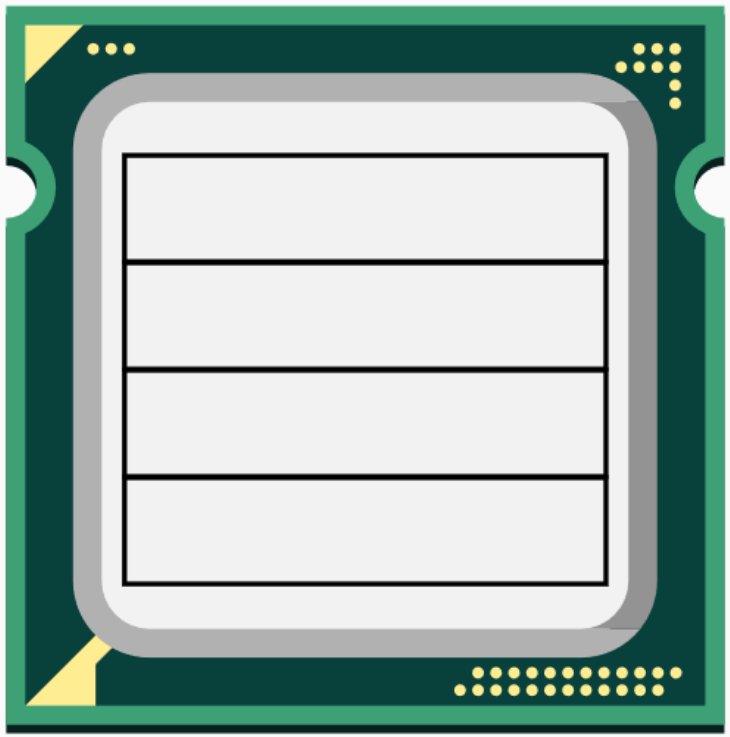
Let’s start the talk then …

Architecture:101





Microarchitecture:101



Programmer

ISA

Architecture

Registers

Caches, TLBs,

Branch predictors,

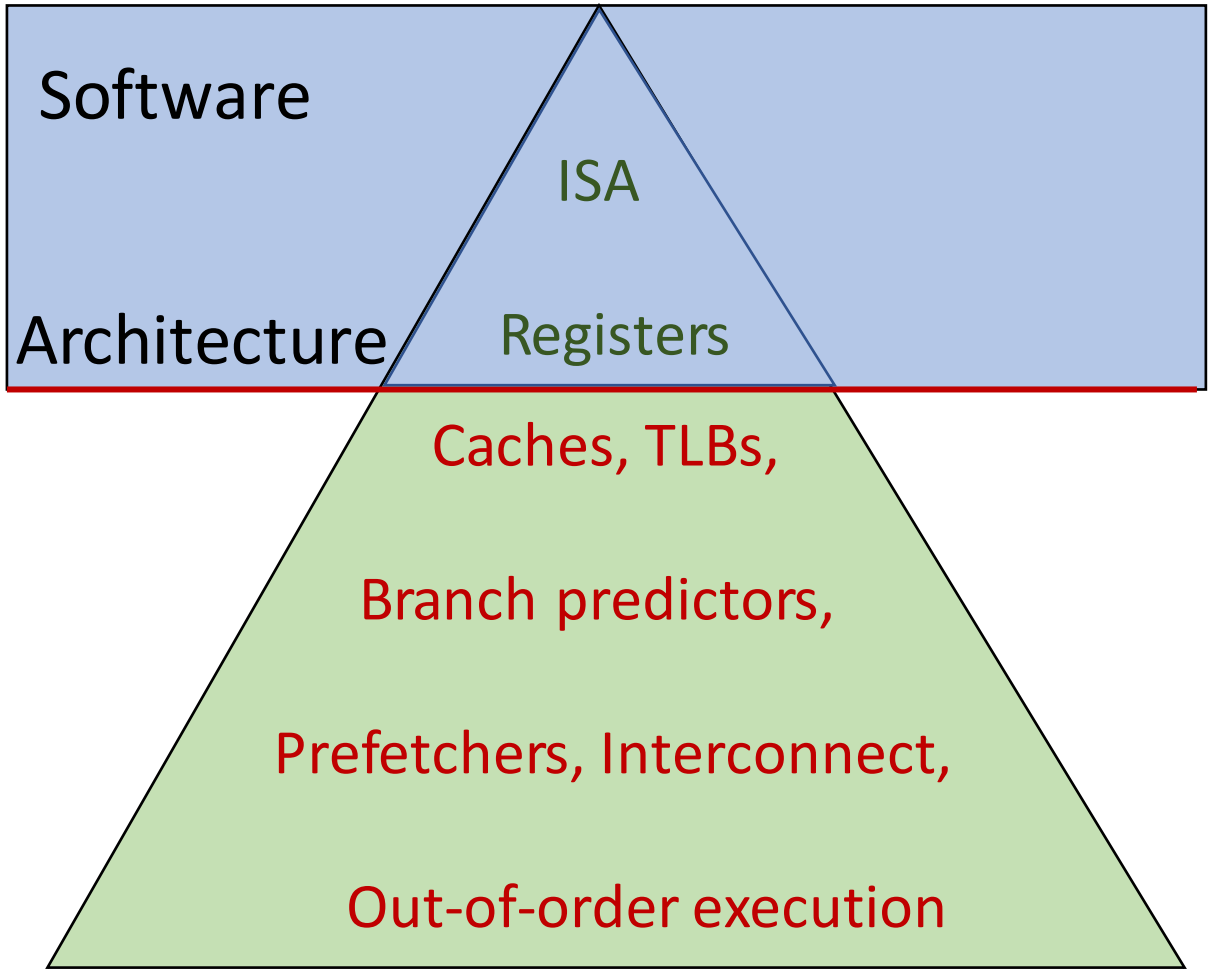
Prefetchers, Interconnect,

Out-of-order execution

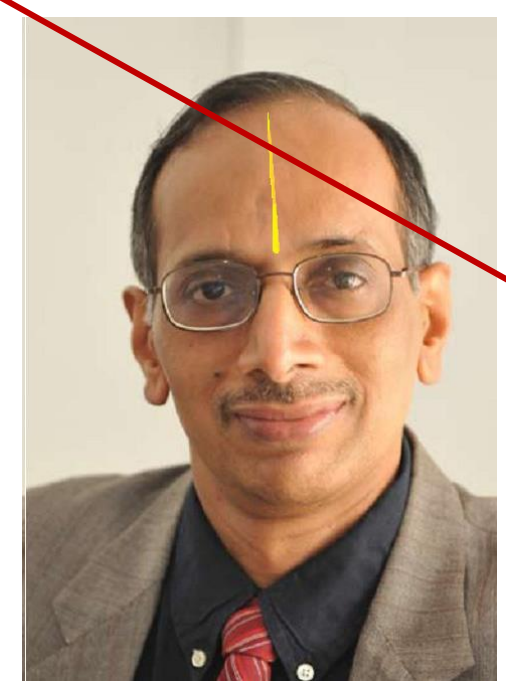
Memory

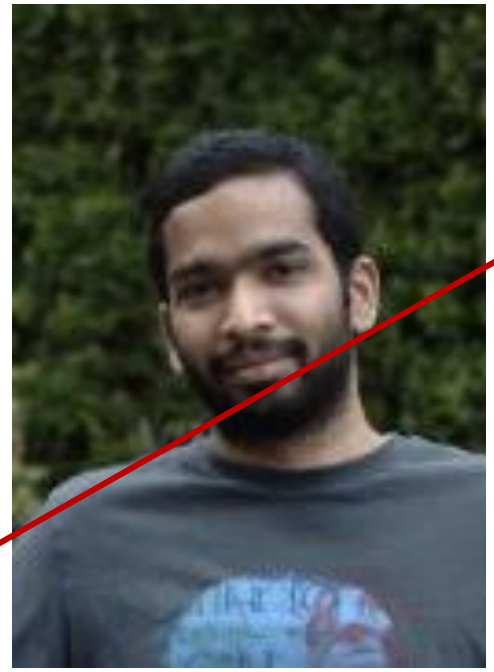
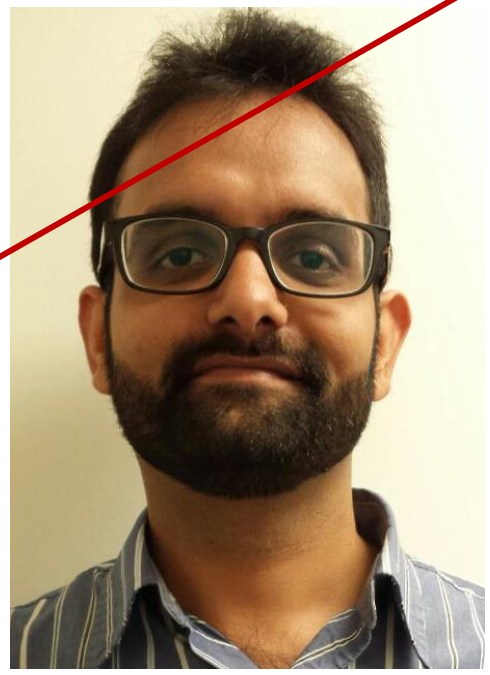
Not exposed to programmer

From Performance to Security: 10K Feet View



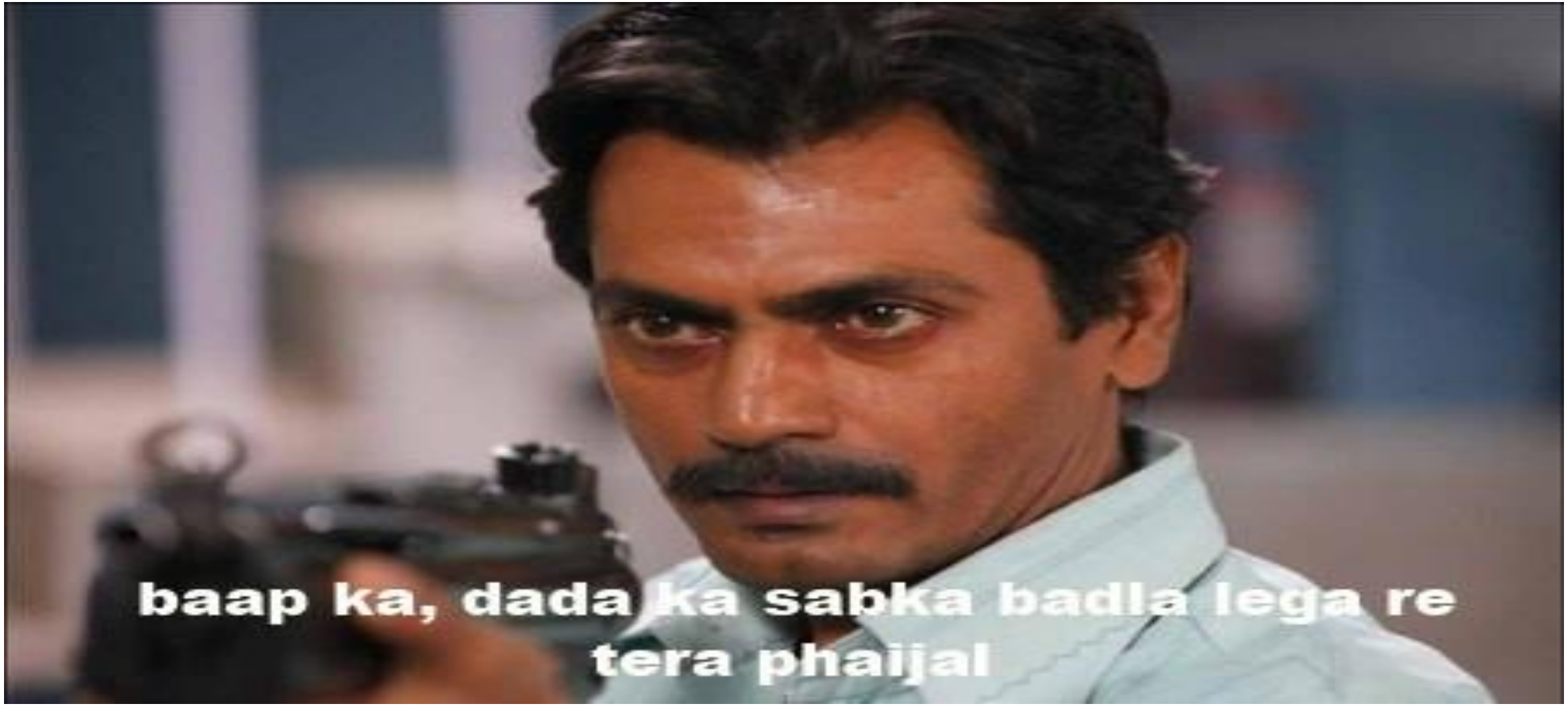
CAWS Talks and Ideas: Secure?



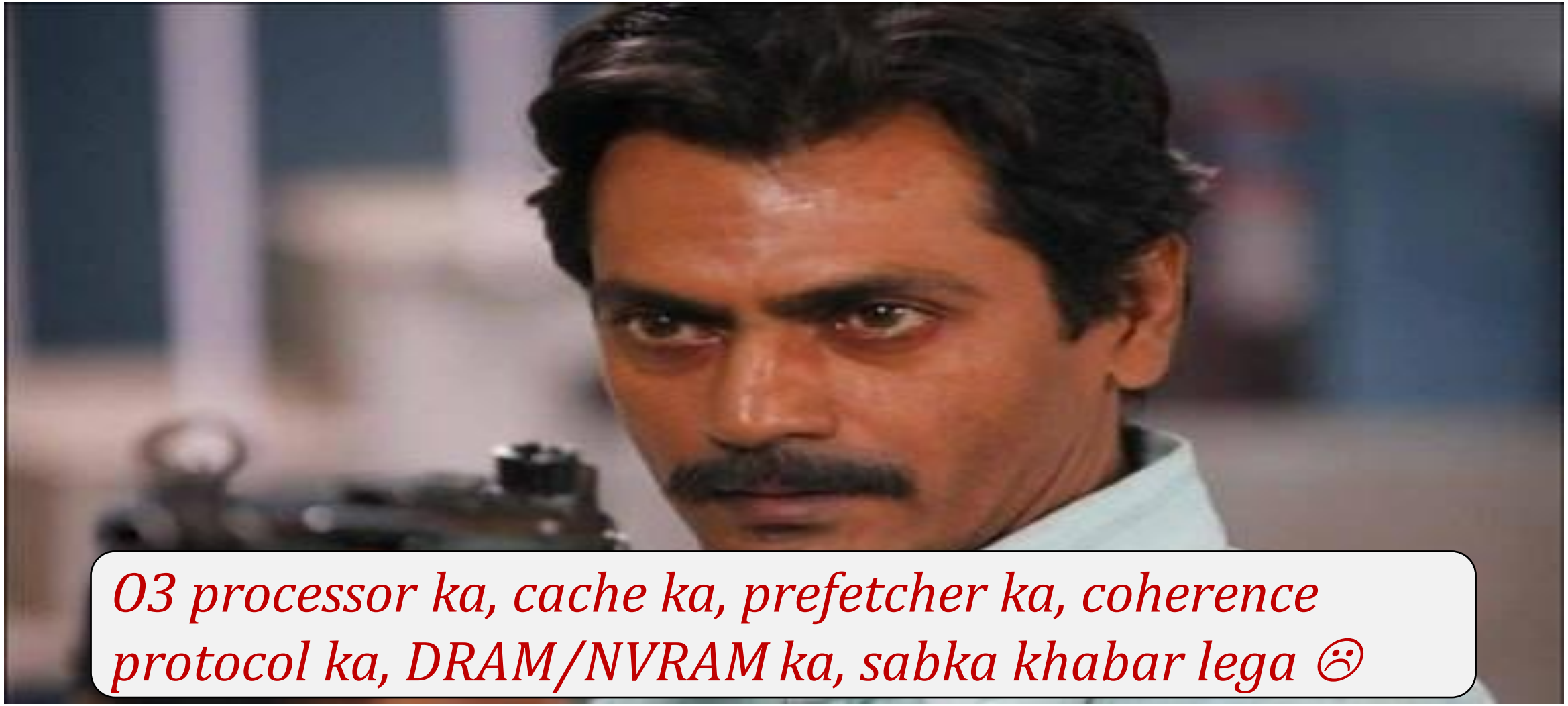




CAWS Talks and Ideas: Secure?

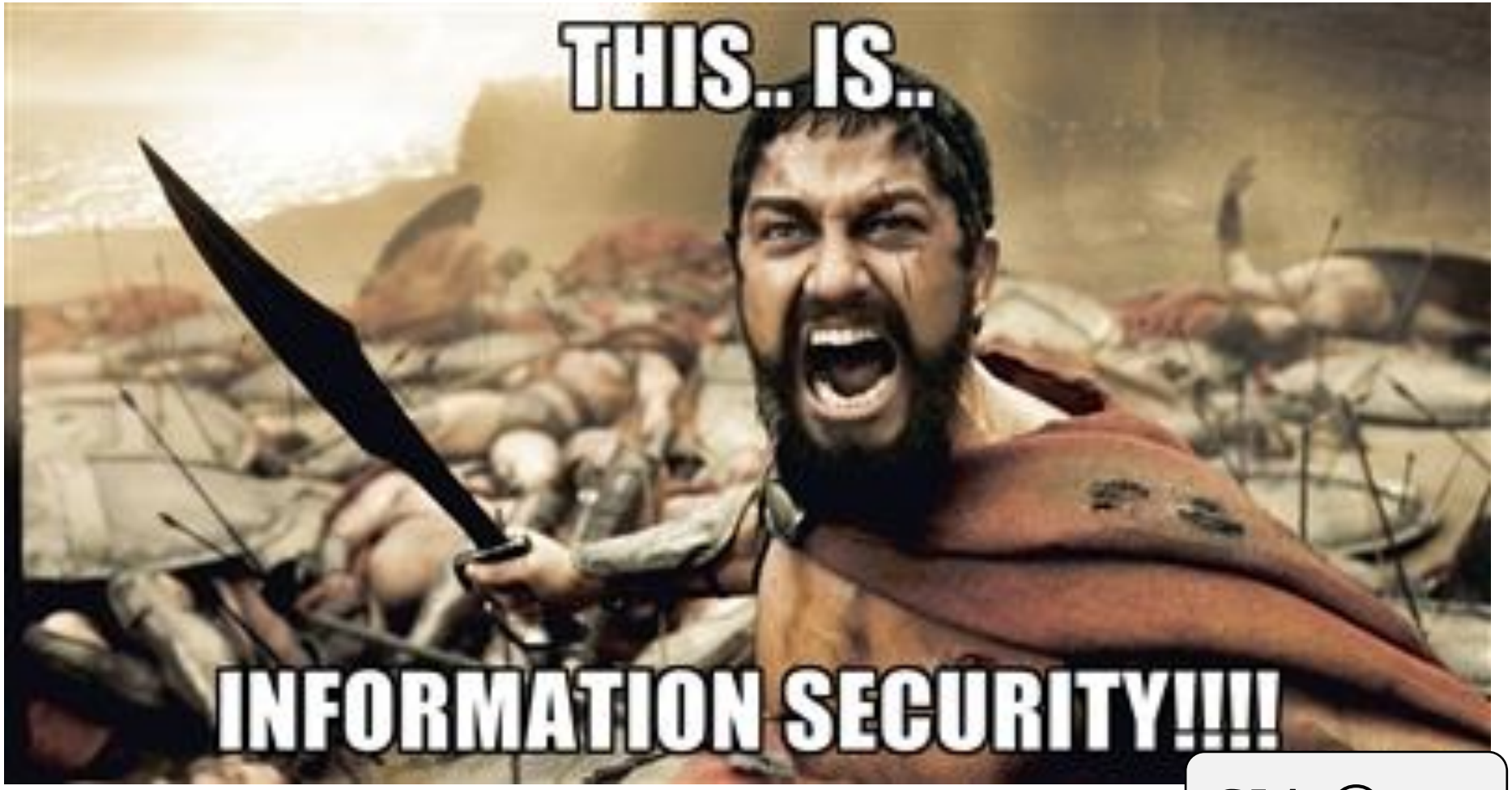
Attacker



Attacker


O3 processor ka, cache ka, prefetcher ka, coherence protocol ka, DRAM/NVRAM ka, sabka khabar lega ☹

What Is Security?



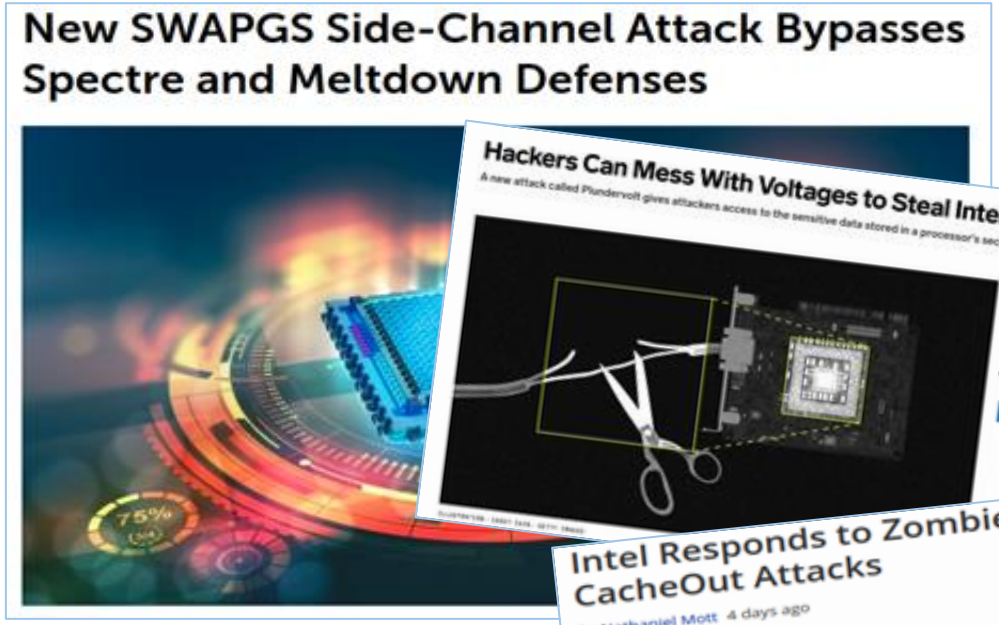
CIA ☹

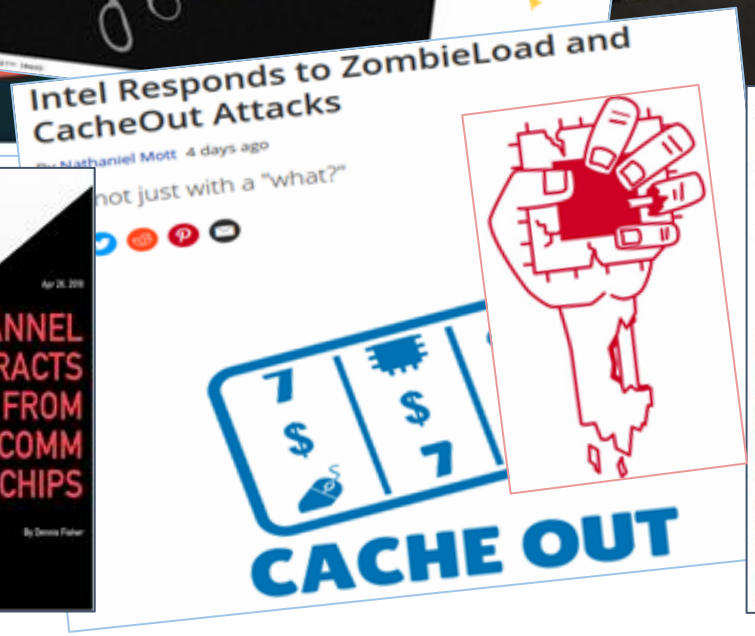
Security: A bit Subtle

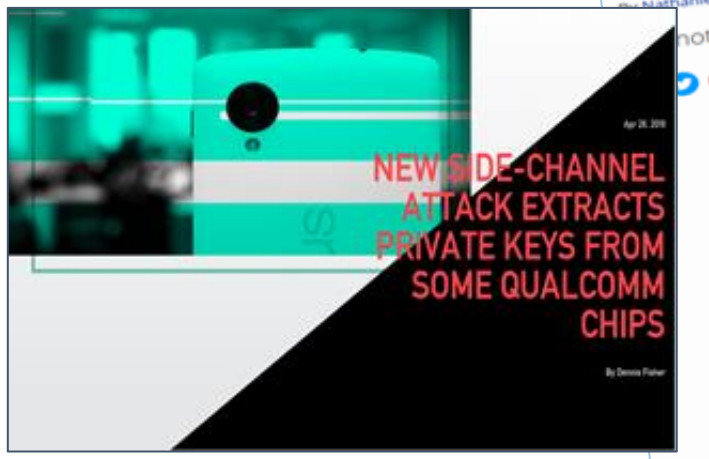
Confidentiality	*You do not **see (READ)** what you are not supposed to see*
Integrity	*You do not **change (WRITE)** what you are not supposed to see*
Availability	*You do not **affect (DELAY)** others (un)intentionally*

Attacks Inside

Media Articles ☹







Brushing-up: Information Leakage

```
x ⟵ 1
for i ⟵ |e|-1 downto 0 do
    x ⟵ x^2 mod n
    if (e_i = 1) then
        x = xb mod n
    endif
done
return x
```

square

reduce

multiply

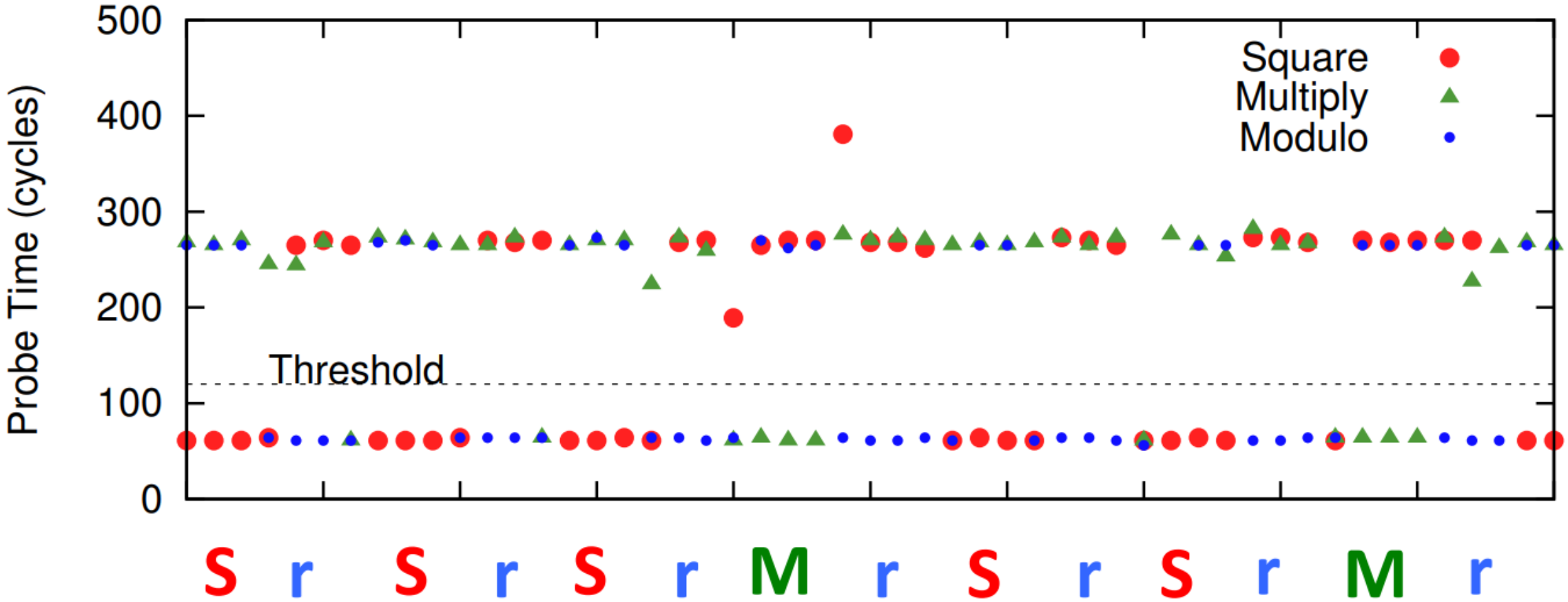
Modular exponentiation, b^e mod n

Exponent e is used for decryption

$e_i = 0$, *Square Reduce (SR)*
$e_i = 1$, *SRMR*

Attacker tries to get the e

Timing Channel



Adpated from flush+reload attack [Usenix Security '14]

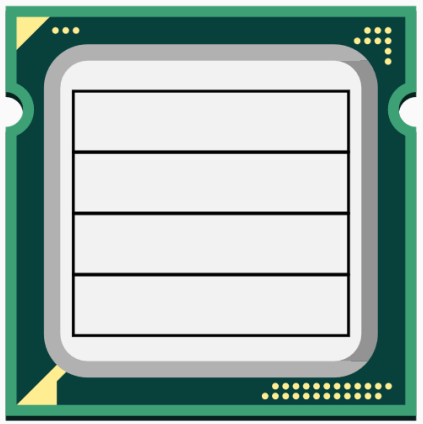
Toy Example: Flush Based Attacks

```
If secret=1 do
    access(&a)
else // secret=0
    no-access
```

Victim

```
    flush(&a)
t1=start_timer
    access(&a)
t2=end_timer
```

Attacker

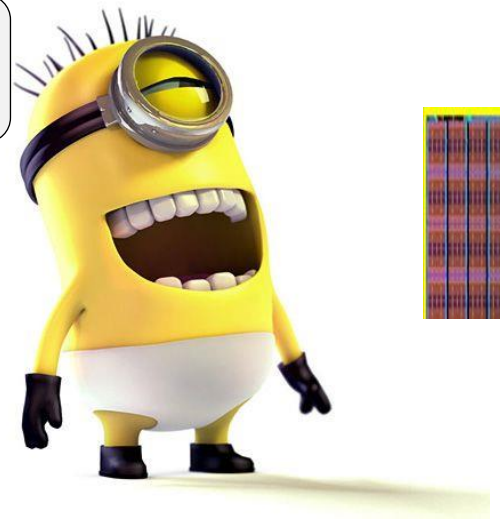


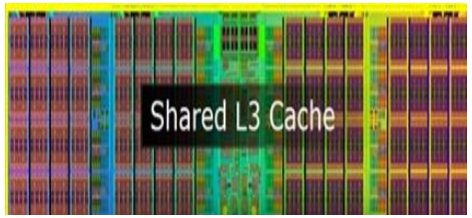
Fast – 1

Slow – 0

Side and Covert Channels

Spy

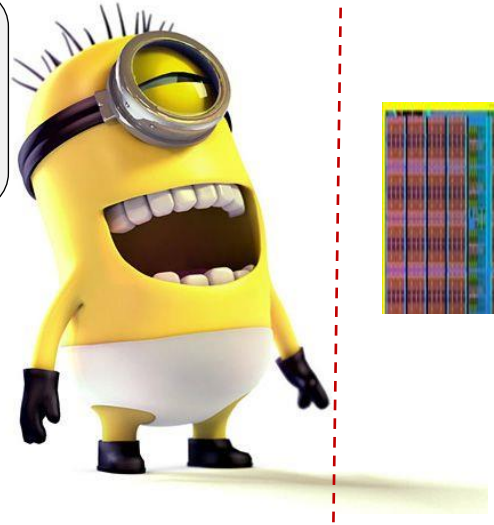


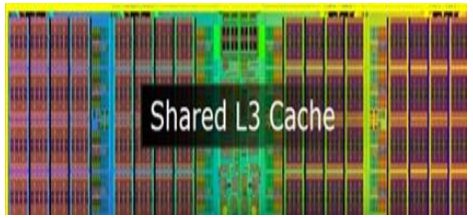


Victim

Side-channel attacks

Let's play



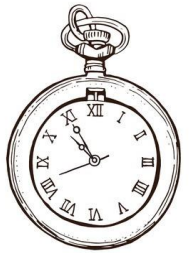


Oh Yes!!

Covert-channel attacks

Shared LLC Attacks

Attacks at the LLC exploit timing channels:
LLC miss > LLC hit



Flush + Reload

Evict + Reload

Prime + Probe

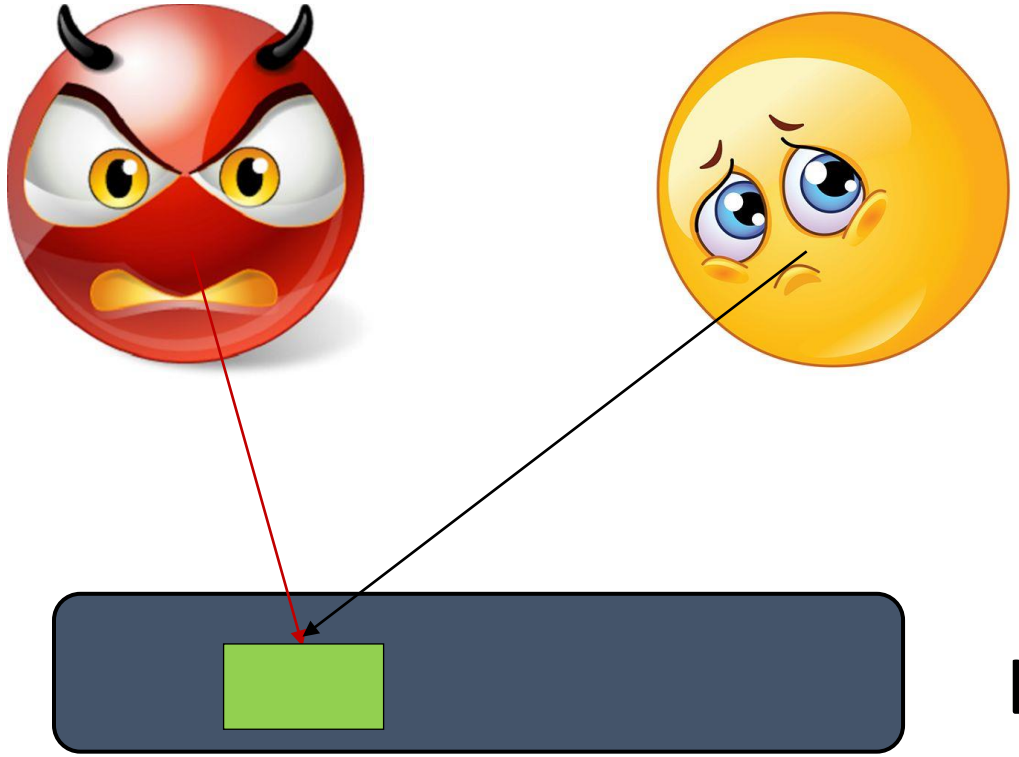
clflush

Eviction based attacks

Threat Model

Knowing the victim *has accessed a cache set (line)* can be considered as a *successful* attack

Flush+Reload Attack



LLC

Step 0:Spy *maps* the shared library, shared in the cache

Flush+Reload Attack

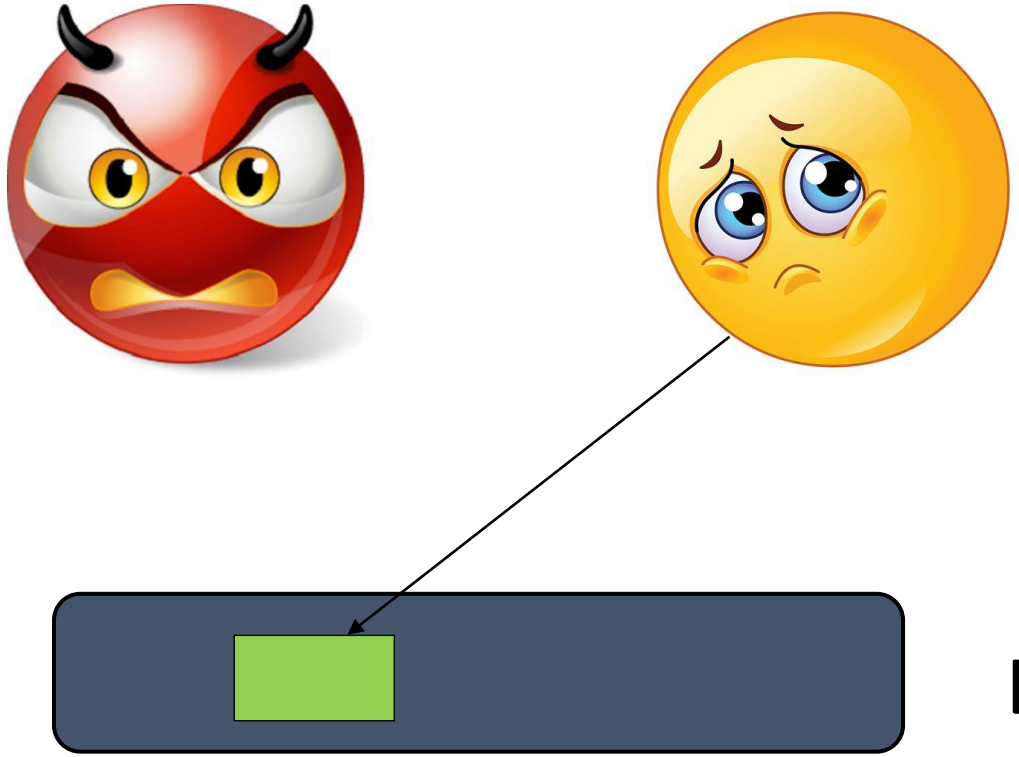
Clflush

LLC

Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Flush+Reload Attack

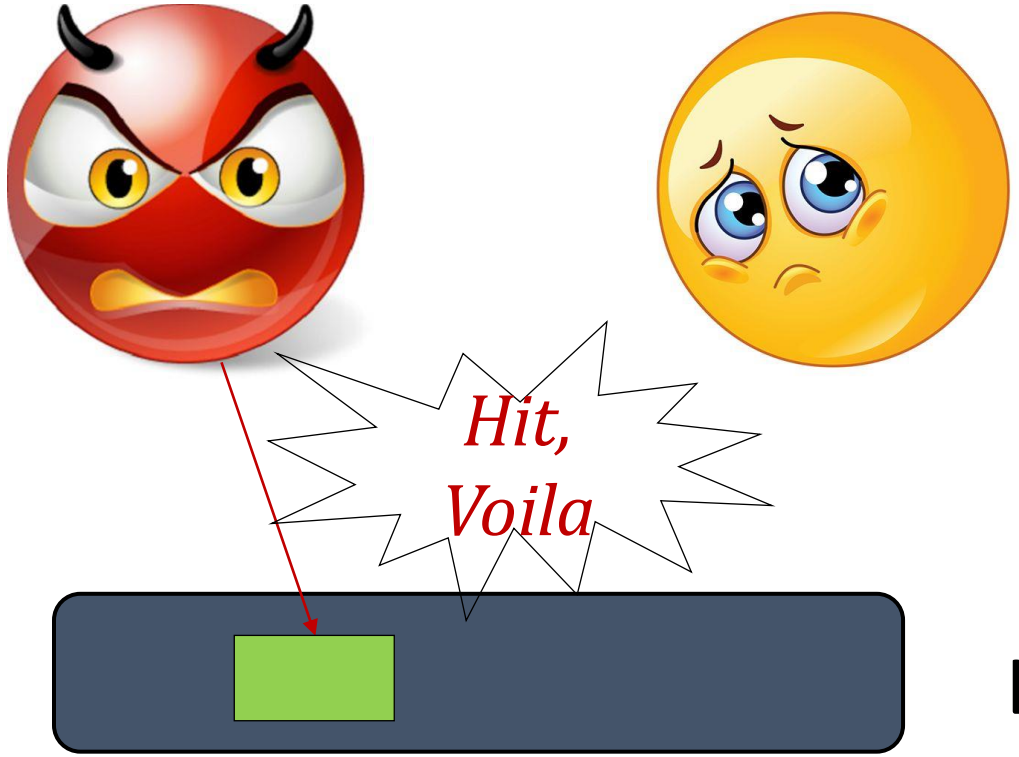


Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Flush+Reload Attack



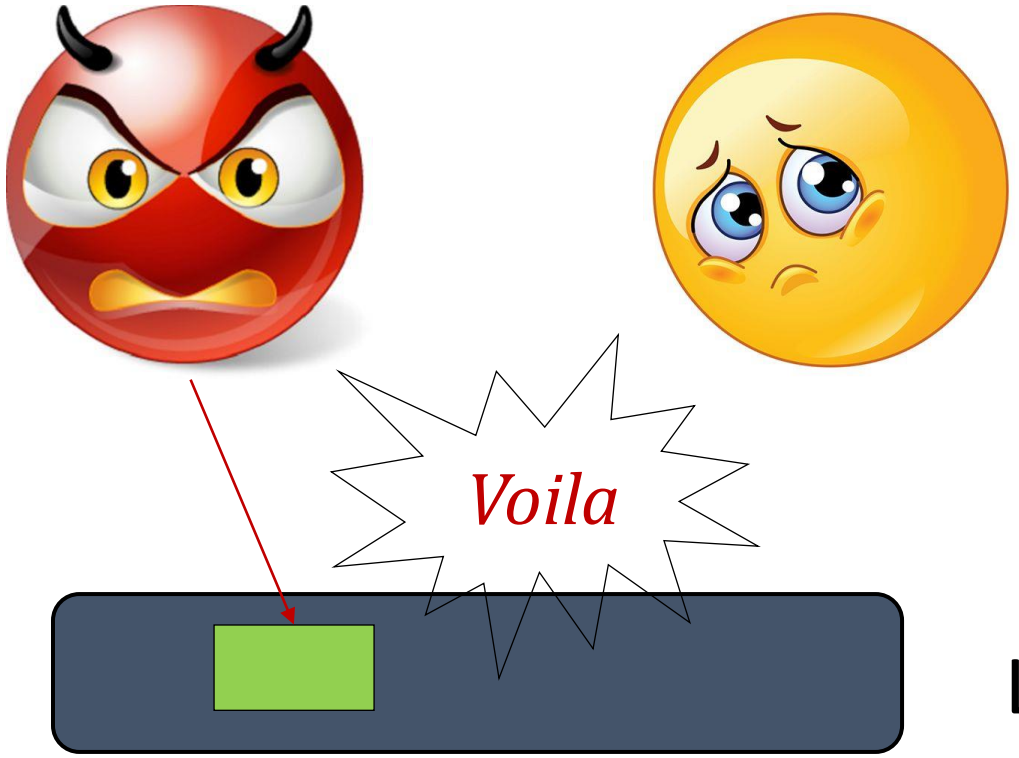
Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Step 3: Spy *reloads* the cache block (hit/miss)

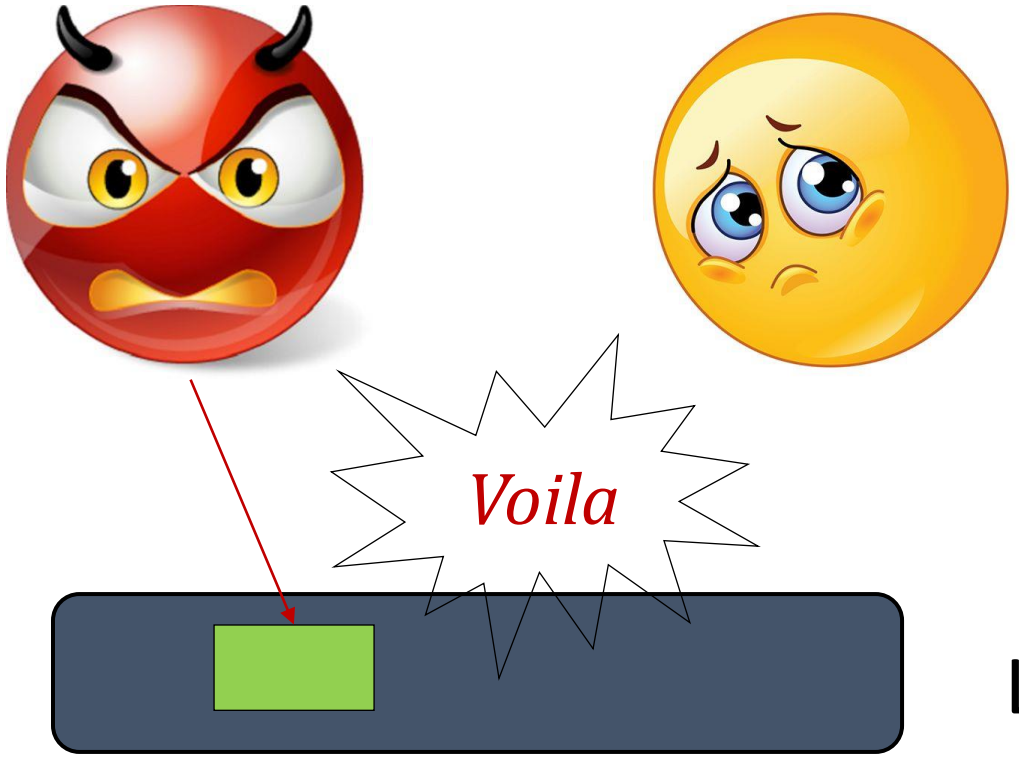
Flush + Flush



Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Flush + Flush

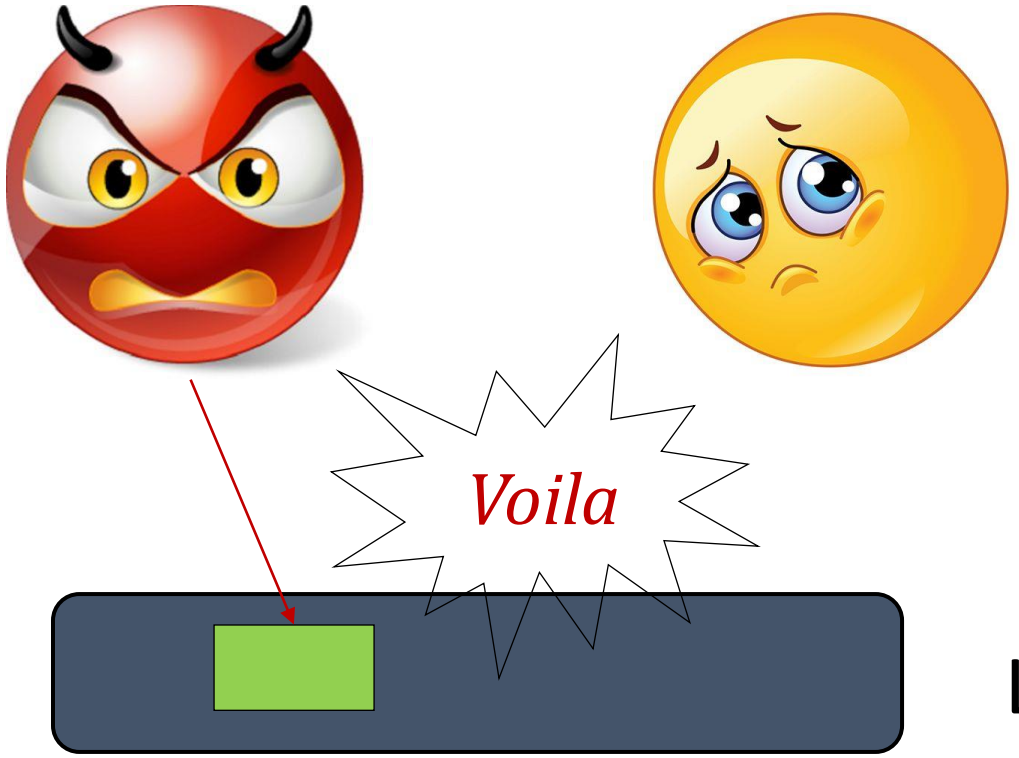


Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Flush + Flush



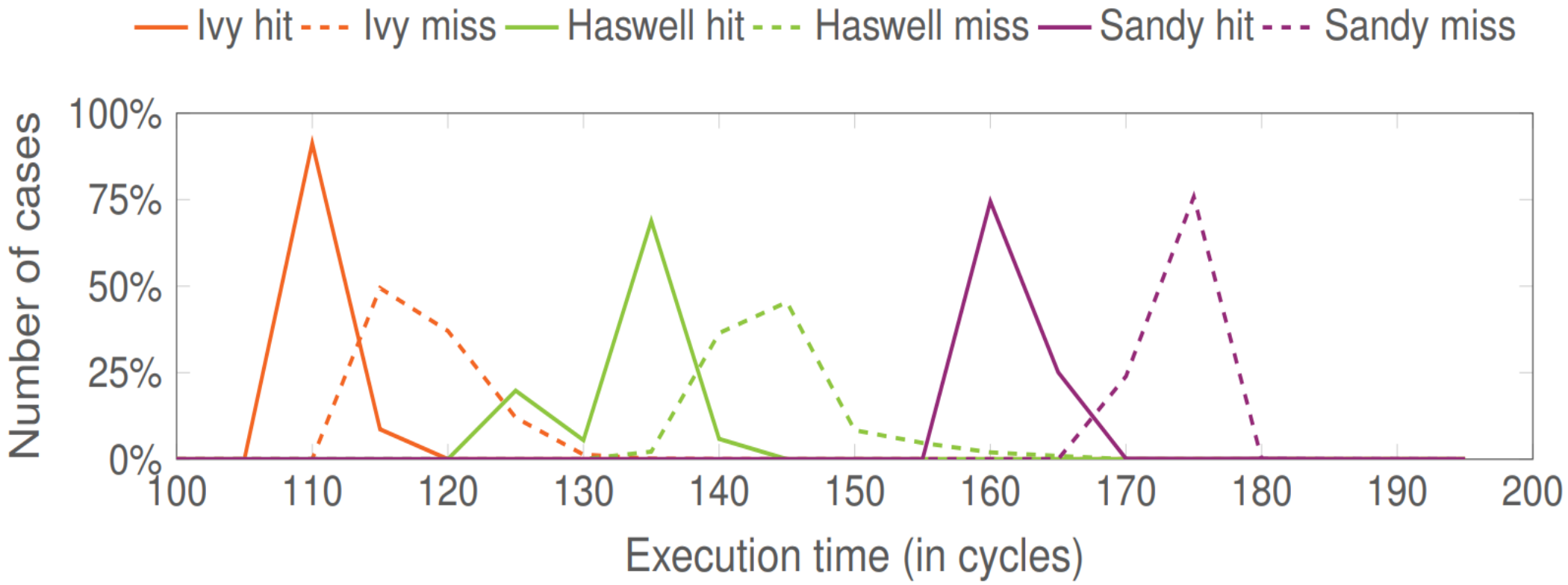
Step 0:Spy *maps* the shared library, shared in the cache

Step 1:Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Step 3: Spy *flushes* the cache block again

Confused?



Adpated from flush+flush attack [DIMVA '16]

No sharing?

What If I do not share anything with you ??

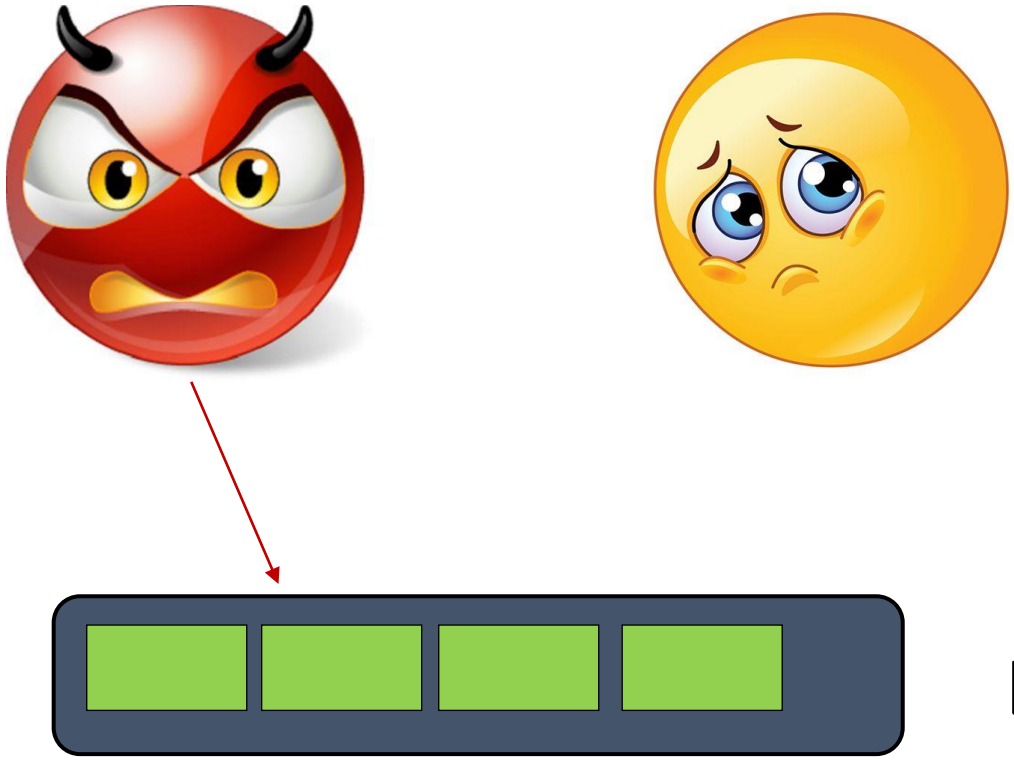
Do not worry, I have Amazon Prime

amazon.com Prime

Sorry:

amazon.com Prime+Probe

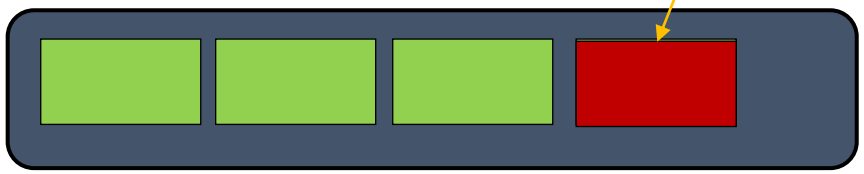
Prime+Probe



Step 0:Spy *fills* the entire shared cache

LLC

Prime+Probe

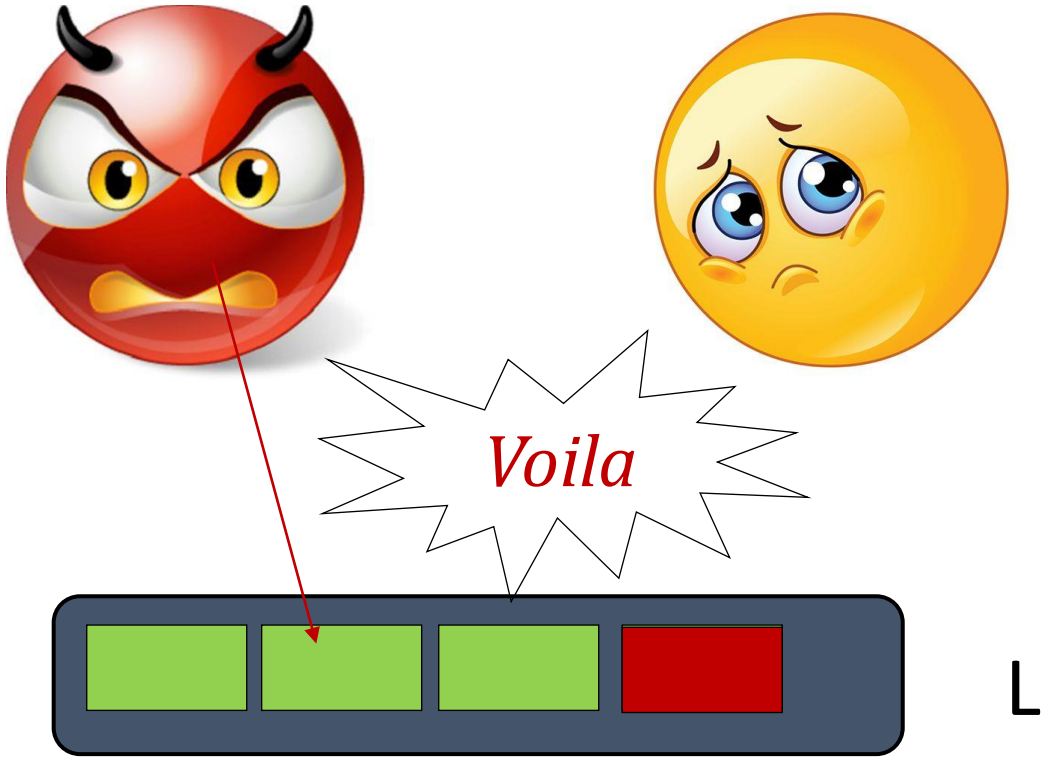


LLC

Step 0:Spy *fills* the entire shared cache

Step 1: Victim *evicts* cache blocks while running

Prime+Probe



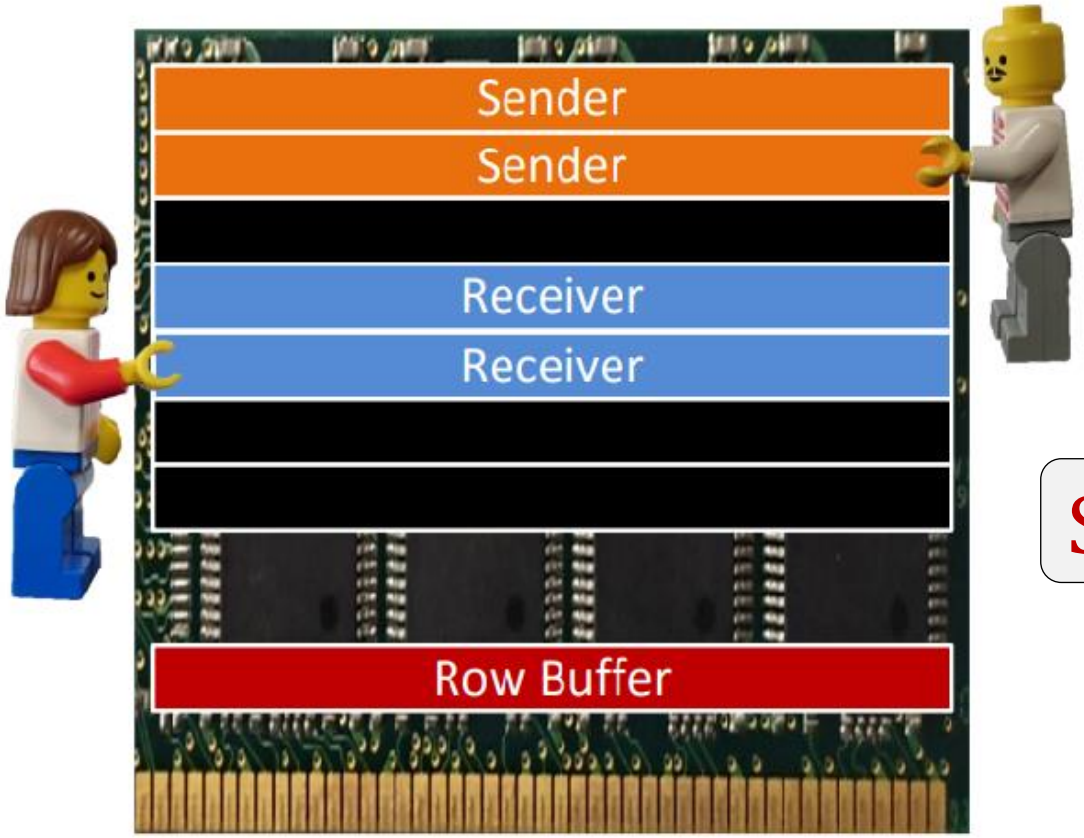
Step 0:Spy *fills* the entire shared cache

Step 1: Victim *evicts* cache blocks while running

Step 2: Spy *probes* the cache set

If misses then victim has accessed the set

Same at the DRAM

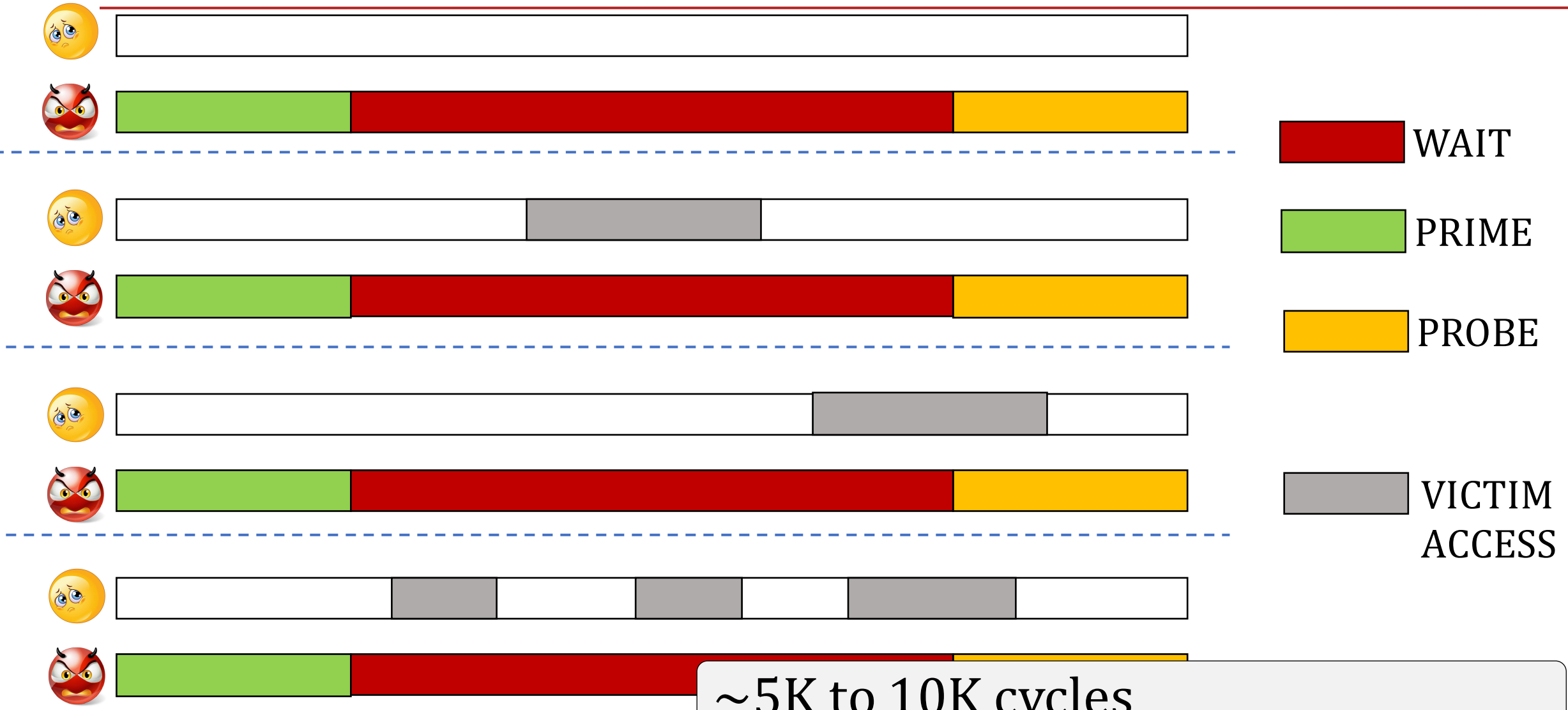


Row buffer hit/miss

Same at all the shared resources

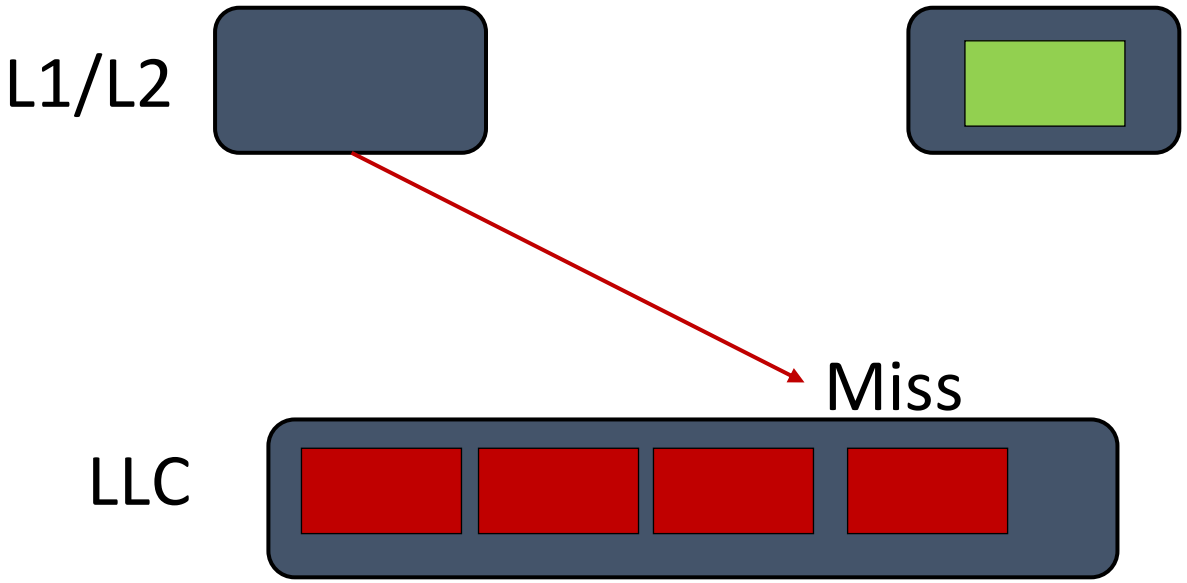
Is it that Simple?

Notion of Time Gap

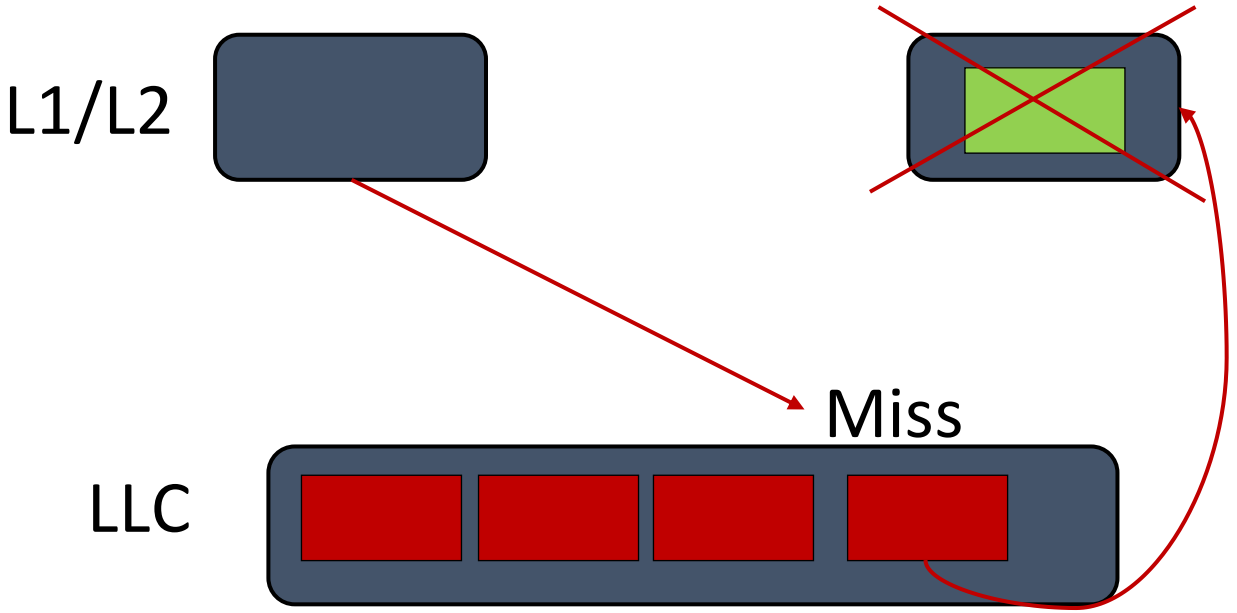


~5K to 10K cycles

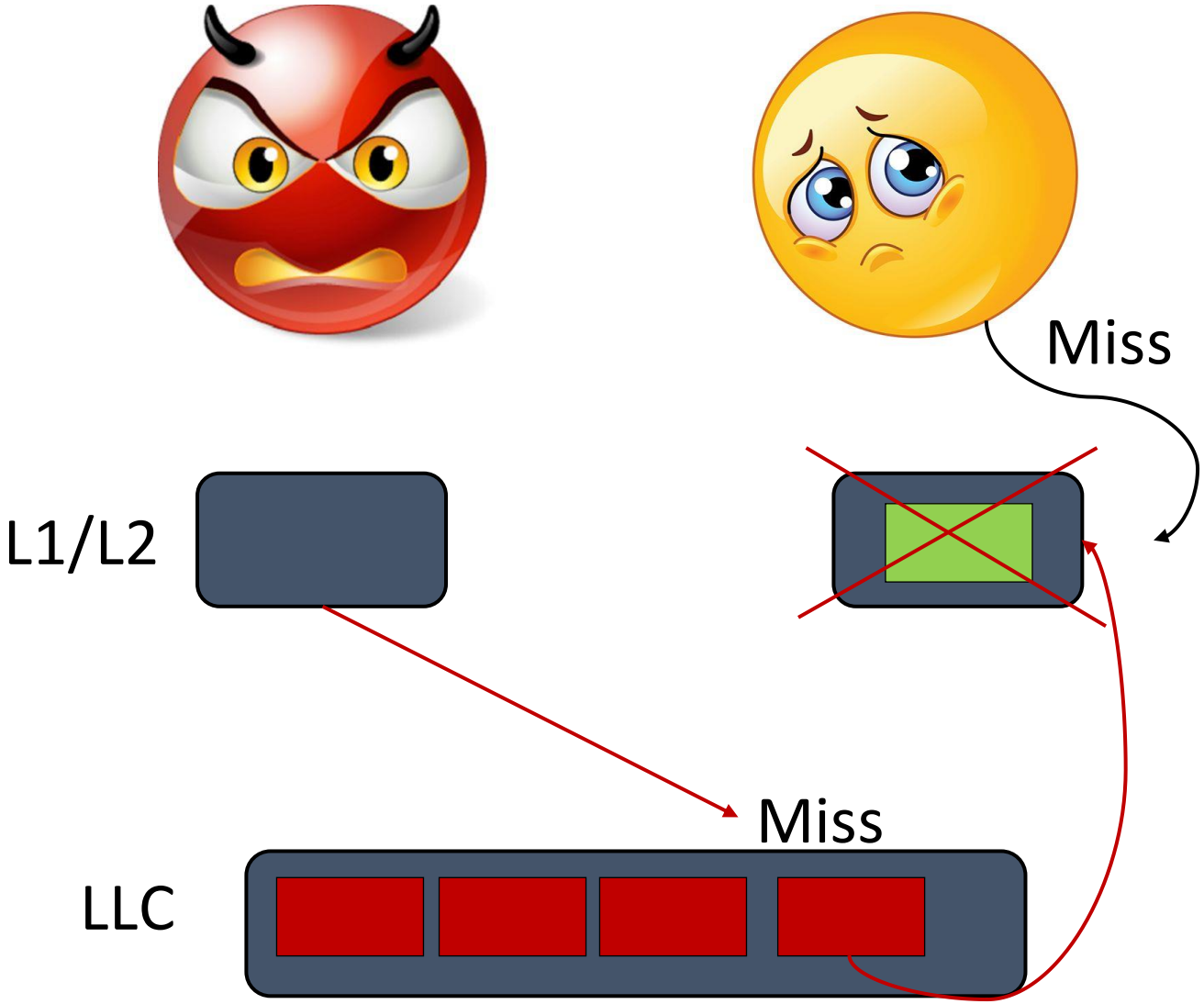
Inclusiveness



Inclusiveness



Inclusiveness



Attacker knows whether victim has accessed a set or not

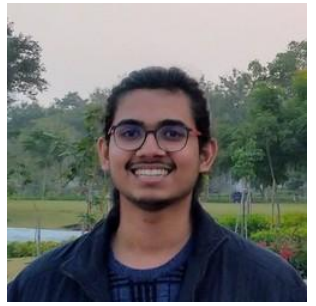
System Noise

DVFS aware

Aware of co-location of victim

Fast and Stealthier

High accuracy even on noisy systems



https://car3s.github.io/dabangg/

The Hacker News

New Noise-Resilient Attack On Intel and AMD CPUs Makes Flush-based Attacks Effective

New Technique Improves Effectiveness of Timing Channel Attacks

By Ionut Arghire on June 01, 2020

Share Tweet Recommend 0 RSS

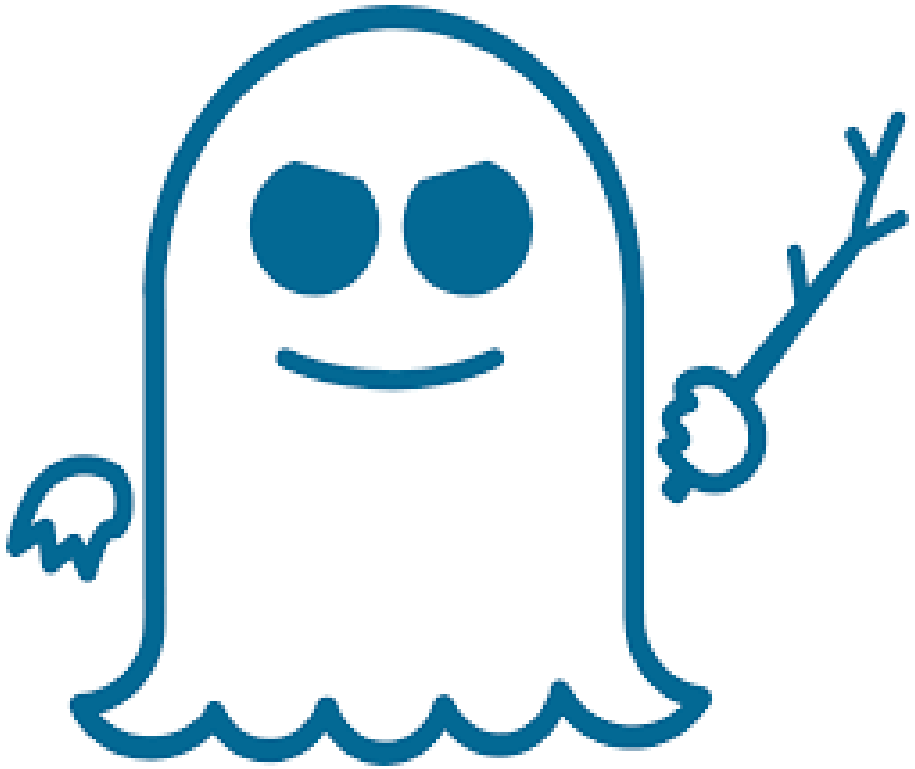
Two researchers have discovered a new timing channel attack technique that remains effective even if multiple processes are running on a system.

Called DABANGG (the Hindi word for fearless), the newly proposed technique improves the effectiveness of flush-based attacks such as Flush+Reload and Flush+Flush, researchers Anish Saxena and Biswabandan Panda from the Indian Institute of Technology Kanpur claim in a research paper.

How Practical?

Future is uncertain, if we do not take care of present attacks, future may be worse ☹

Spectre and Meltdown

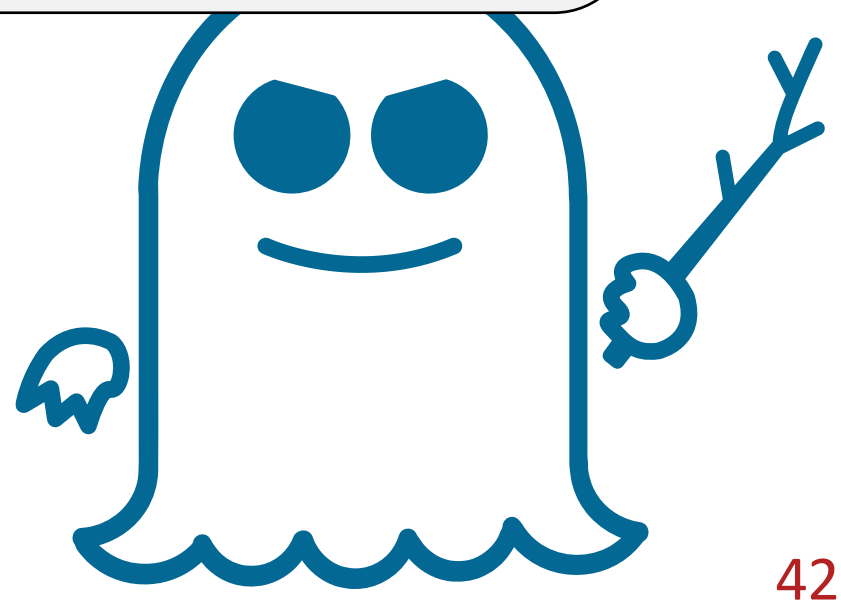


Spectre in Action: Fasten Your Seat Belts

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

DRAM LOAD

DRAM LOAD



Branch Predictor and Speculative Execution

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

Branch predictor returns TRUE ☹

Branch Predictor and Speculative Execution

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

Branch predictor returns TRUE ☹

T T T T T T T T T T T T

Attacker has mis-trained it ☹ ☹

How? By using values less than 3 always ☹ ☹

Branch Predictor and Speculative Execution

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch Predictor and Speculative Execution

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch resolution latency 200 cycles ☹ ☹ ☹ ☹

Within these 200 cycles ☺

```
int CAWSArray = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CAWSArray))
        y = MyArray[CAWSArray[attacker]*512]
```

CAWSArray[4] is in L1/L2/L3 ☹

The address is in the cache ☹ ☹

Yes, you guessed it right: F+R, P+P cache attacks ☹ ☹ ☹

Picture Abhi Baki Hai ☺ After 200 cycles

Processor realized it was a mistake and *flushed* all wrong path instructions

But cache has the data ☹

y = MyArray[CAWSArray[attacker]*512]

LOAD MyArray[0] 60 ns
LOAD MyArray[512] 60 ns
LOAD MyArray[1024] 5 ns Bingo !! *CAWSArray[attacker] = 2*

Meltdown: The O3 Curse!!

1. raise_exception();
2. // line below is never reached
3. secret=KernelArray[data*4096];

Kernel Trap

Out-of-order (O3) as it has no dependency

1. secret=KernelArray[data*4096];
2. raise_exception();

What about page-fault?

Mitigations: Read it On your Own !!

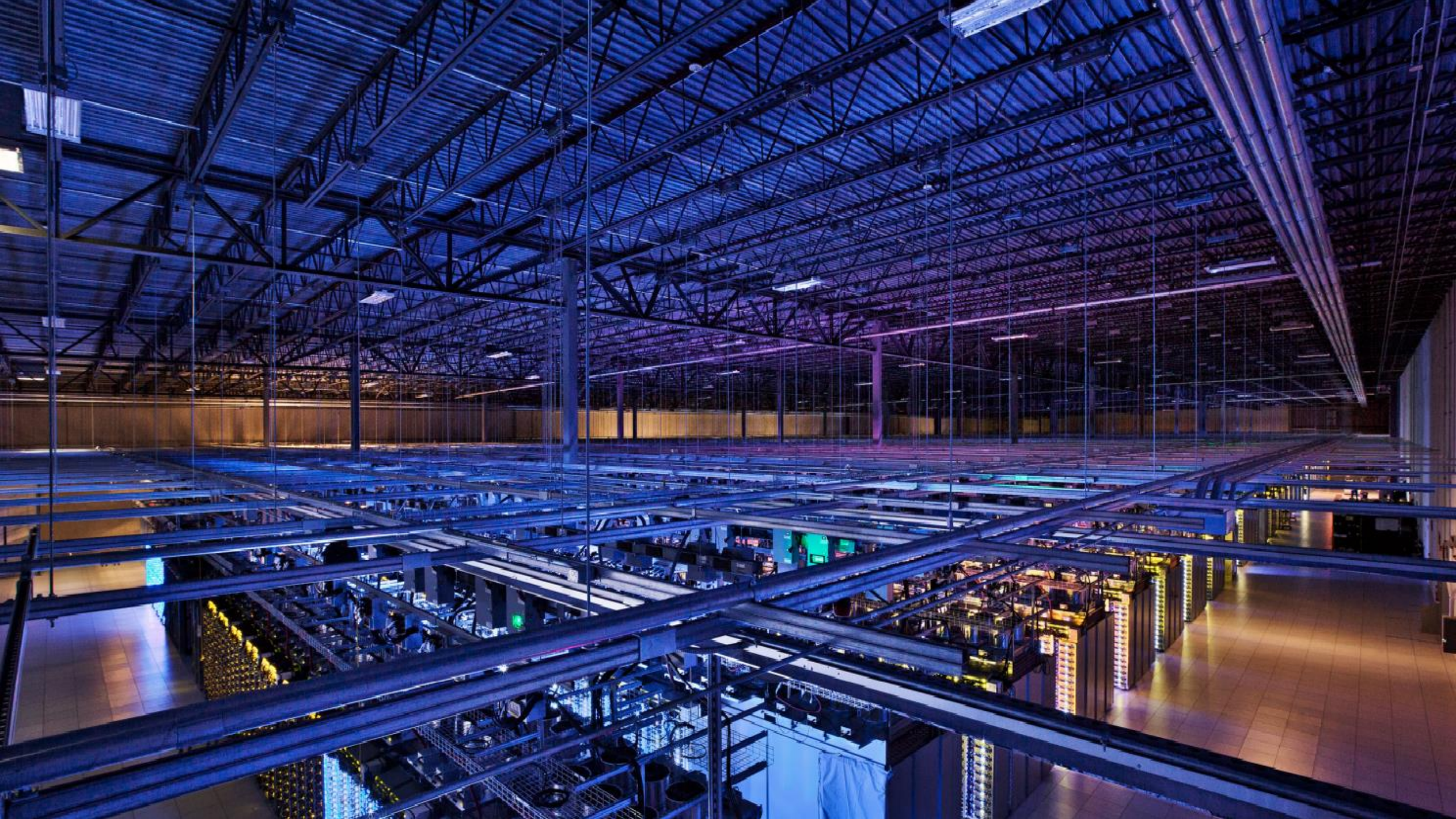
Attacks in the Cloud

Oh no ☹

Attacks in the Cloud

I use cloud

No fear of information leakage ??




server

CPU #1

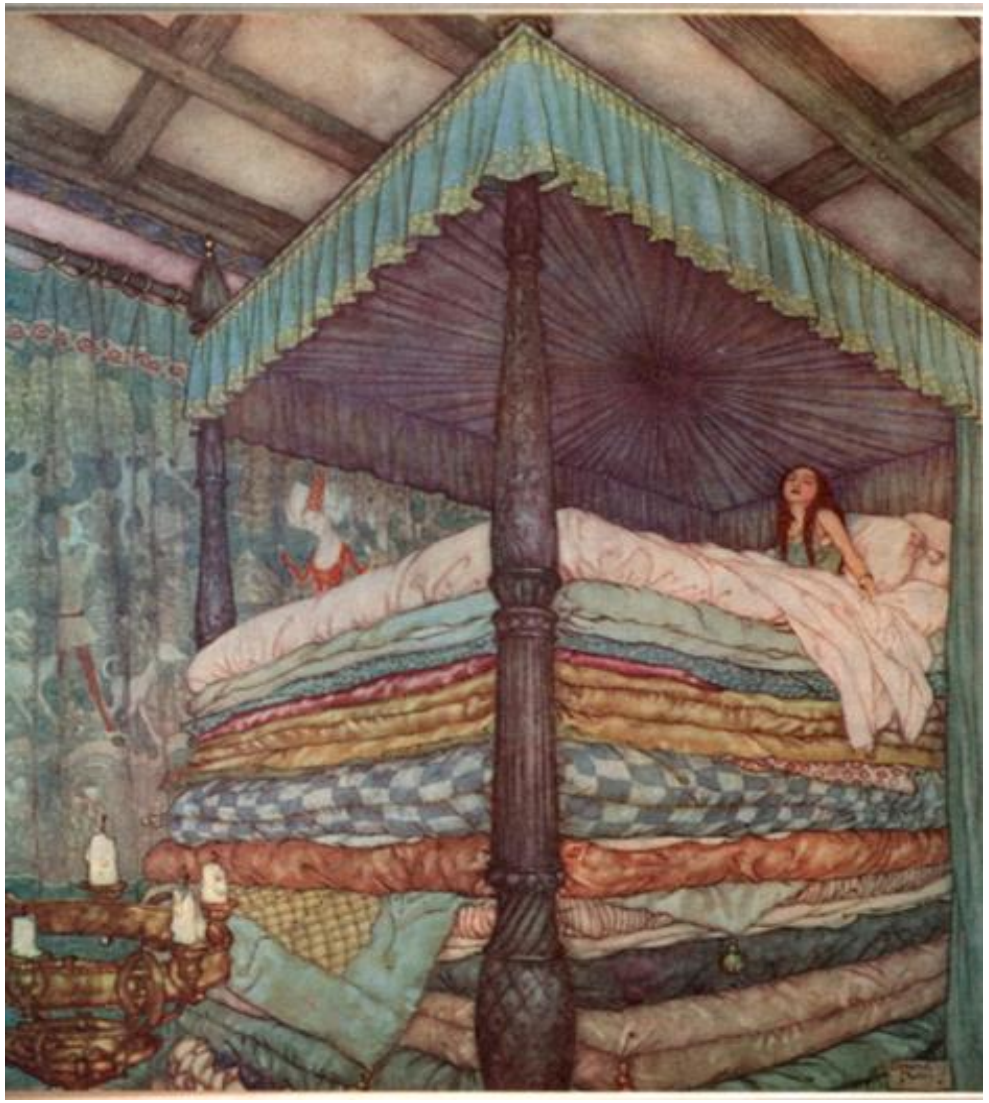
CPU #2

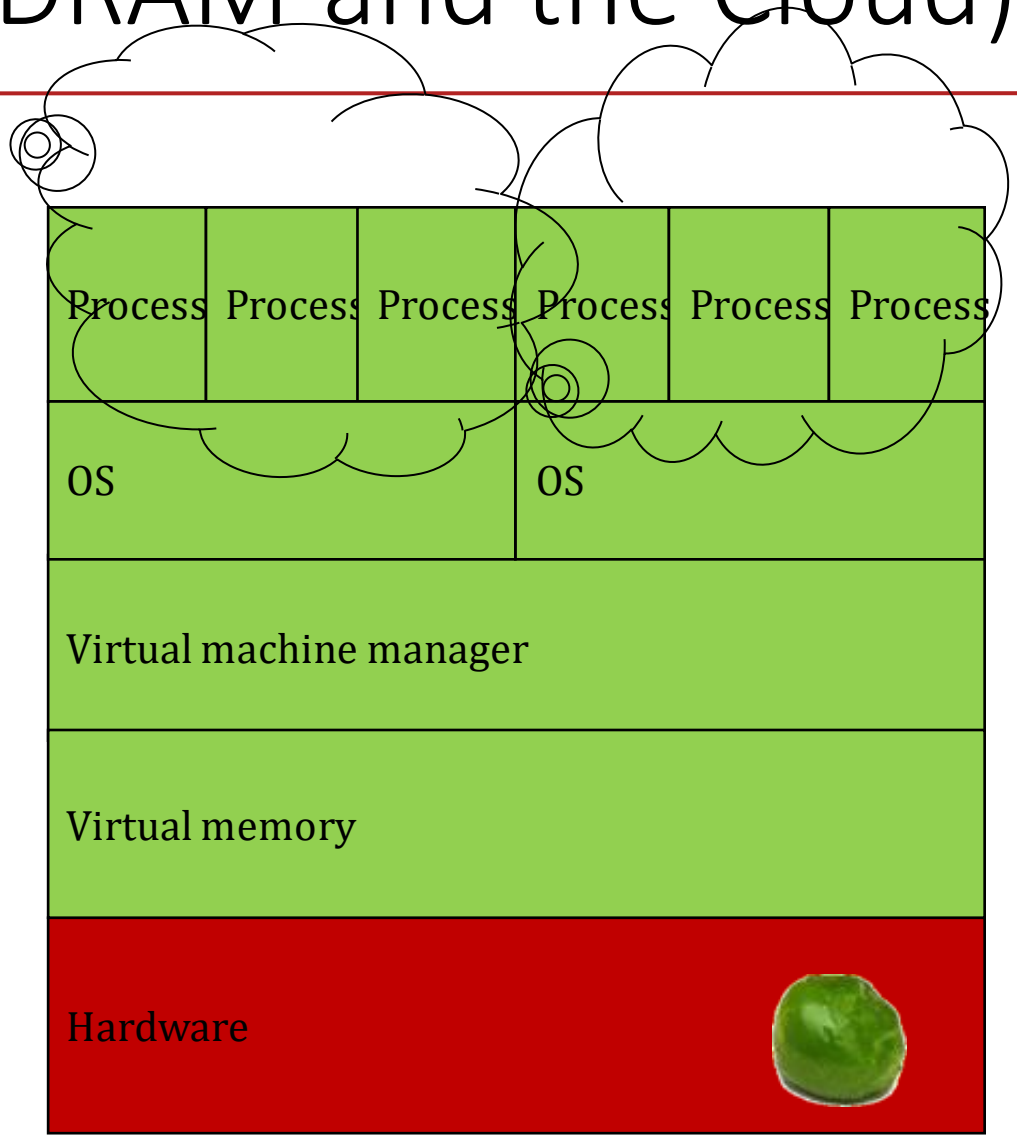
DRAM

Gotcha!!

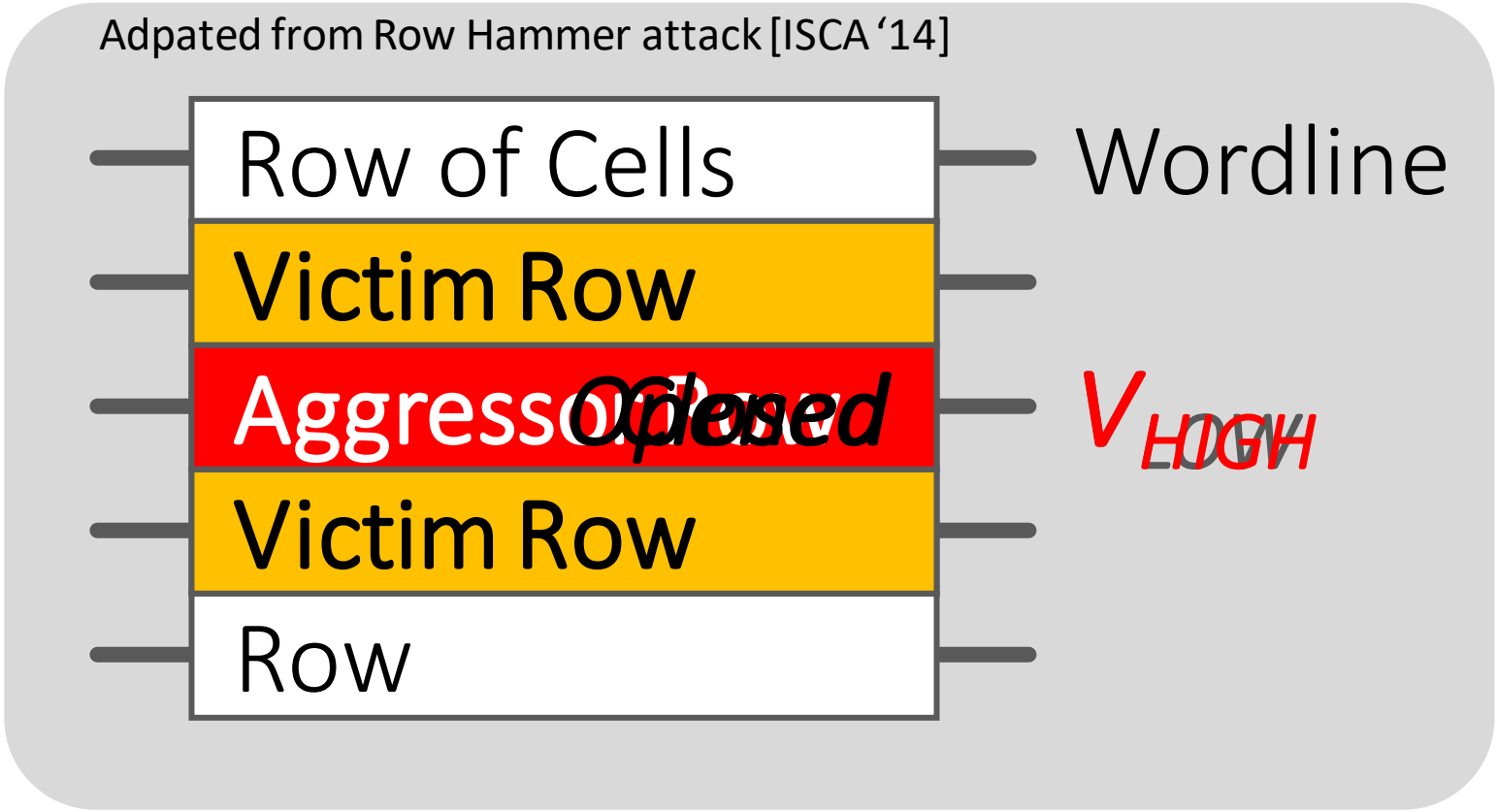
[Courtesy: Eran Tromer]

Pea and the Princess (LLC/DRAM and the Cloud)

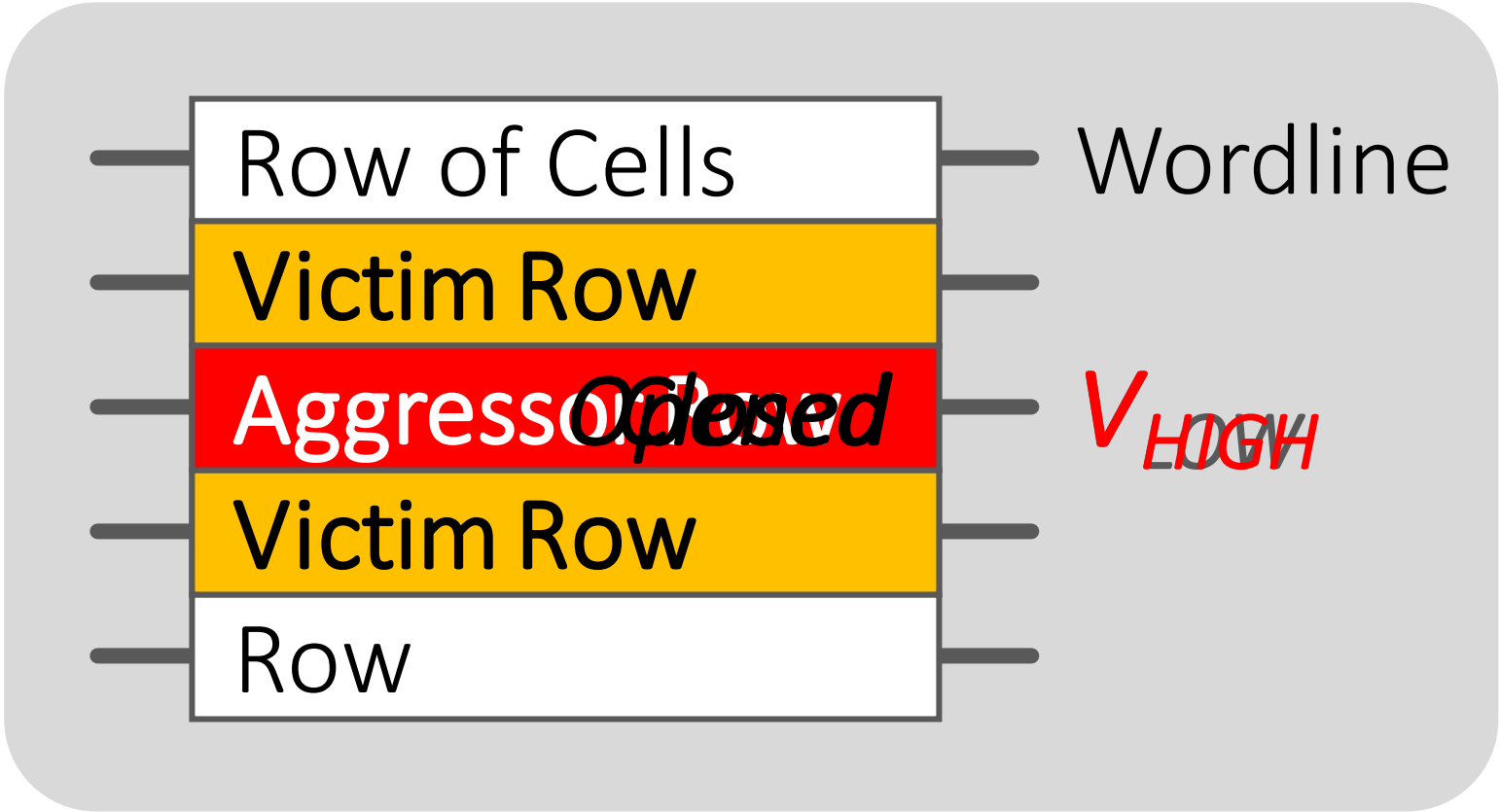




RowHammer (Time for Integrity based attacks)

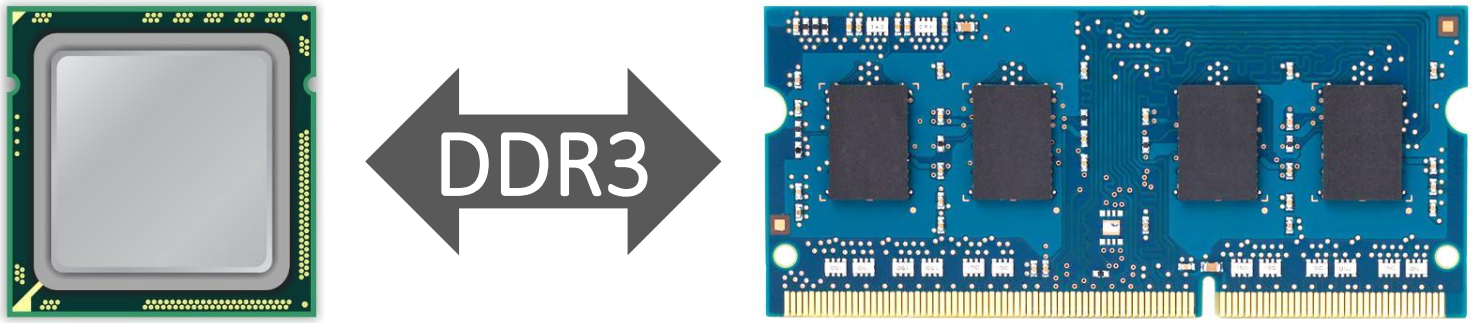


*Repeatedly opening and closing a row induces **disturbance errors** in adjacent rows*



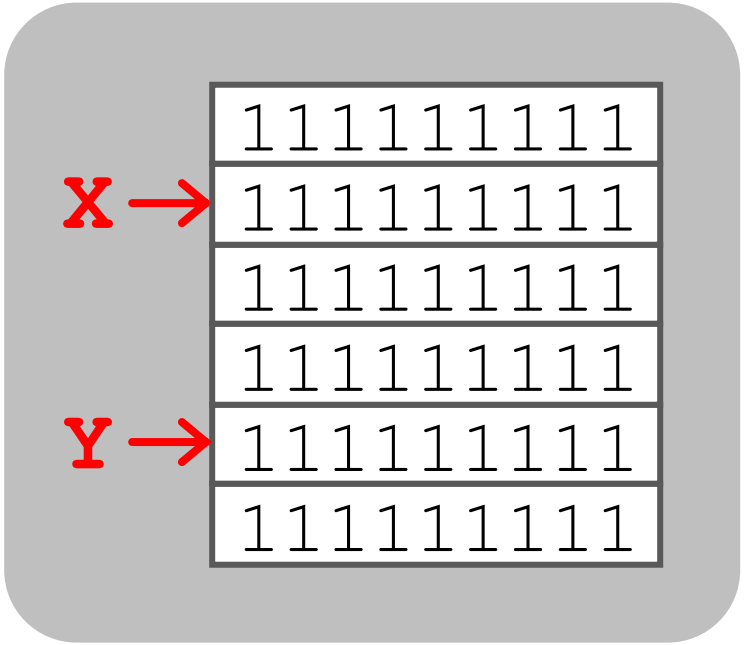
"It's like breaking into an apartment by repeatedly slamming a neighbor's door until the vibrations open the door you were after" – Motherboard Vice

RowHammer



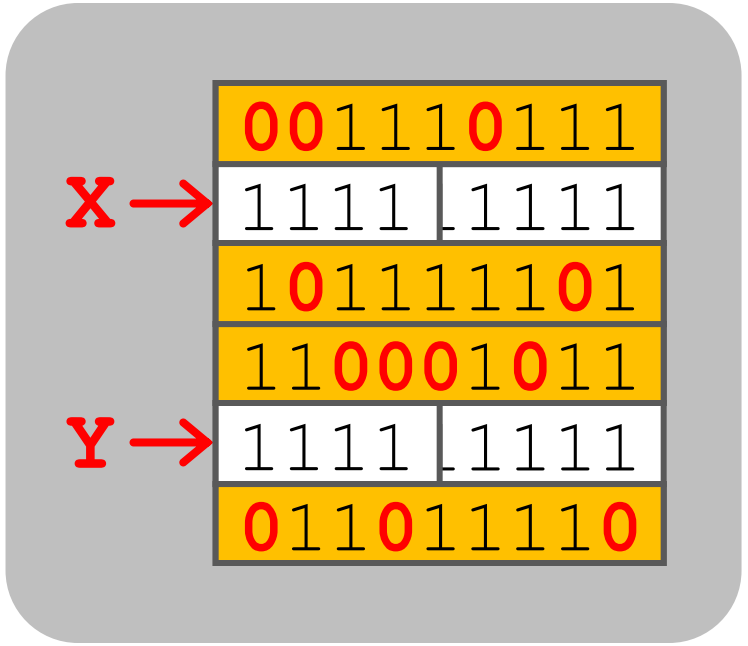
1. Avoid ***cache hits***
   – Flush **X** from cache

2. Avoid ***row hits*** to **X**
   – Read **Y** in another row



RowHammer (But Why? Read it Later)

```
loop:
   mov (X), %eax
   mov (Y), %ebx
   clflush (X)
   clflush (Y)
   mfence
   jmp loop
```

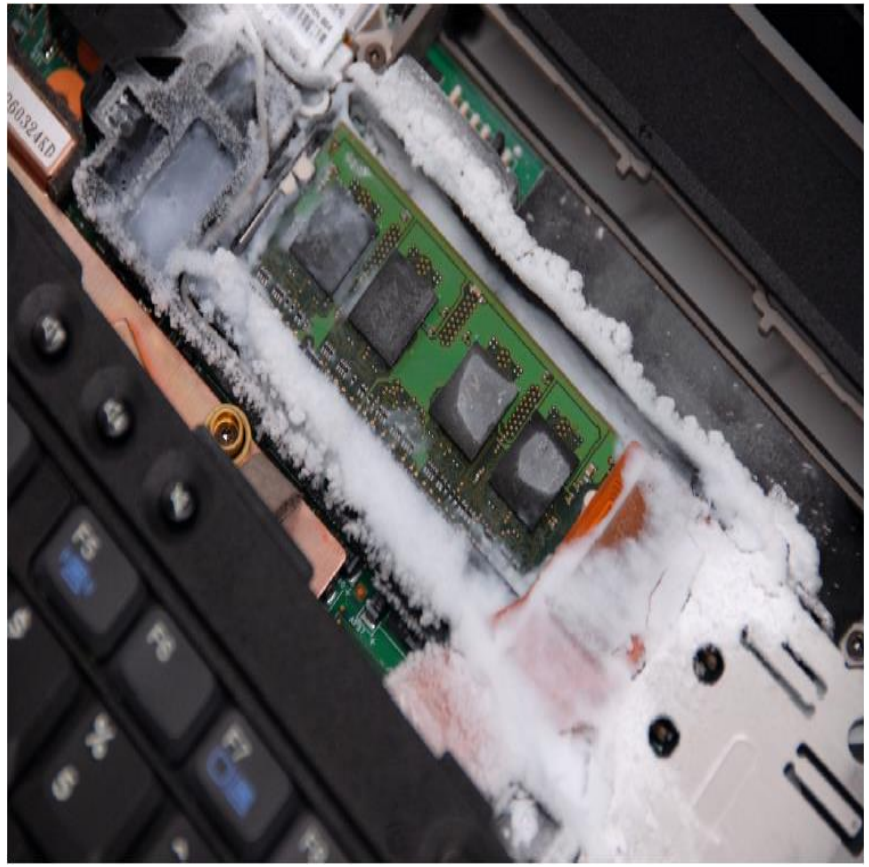


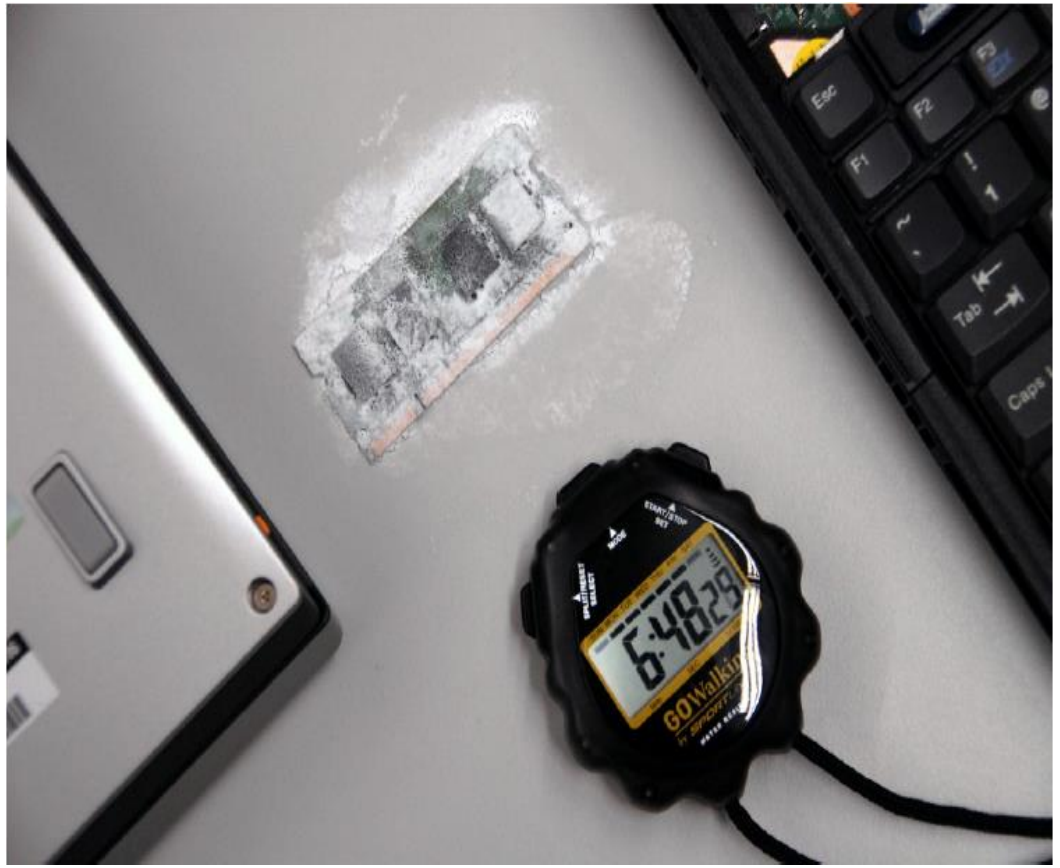
Some Cool Attacks

Before powering off

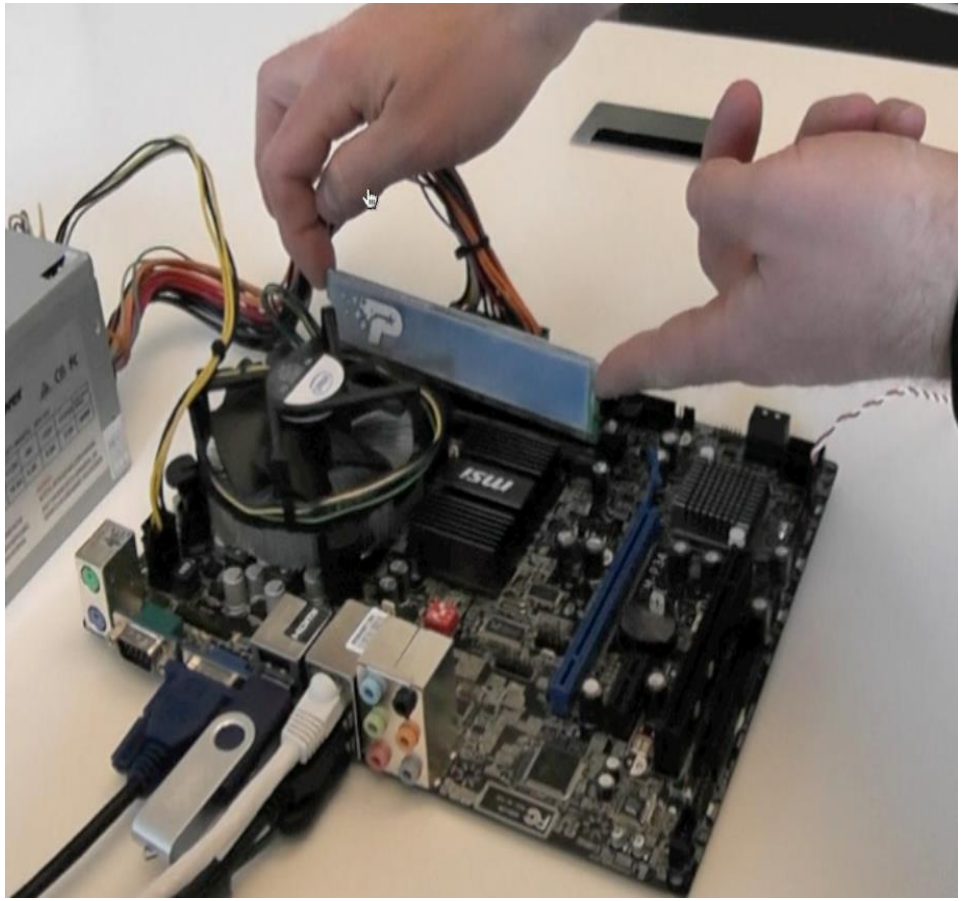
Freeze it to -50 ° C

Cool It Down (Data remanence, cold-boot attack)



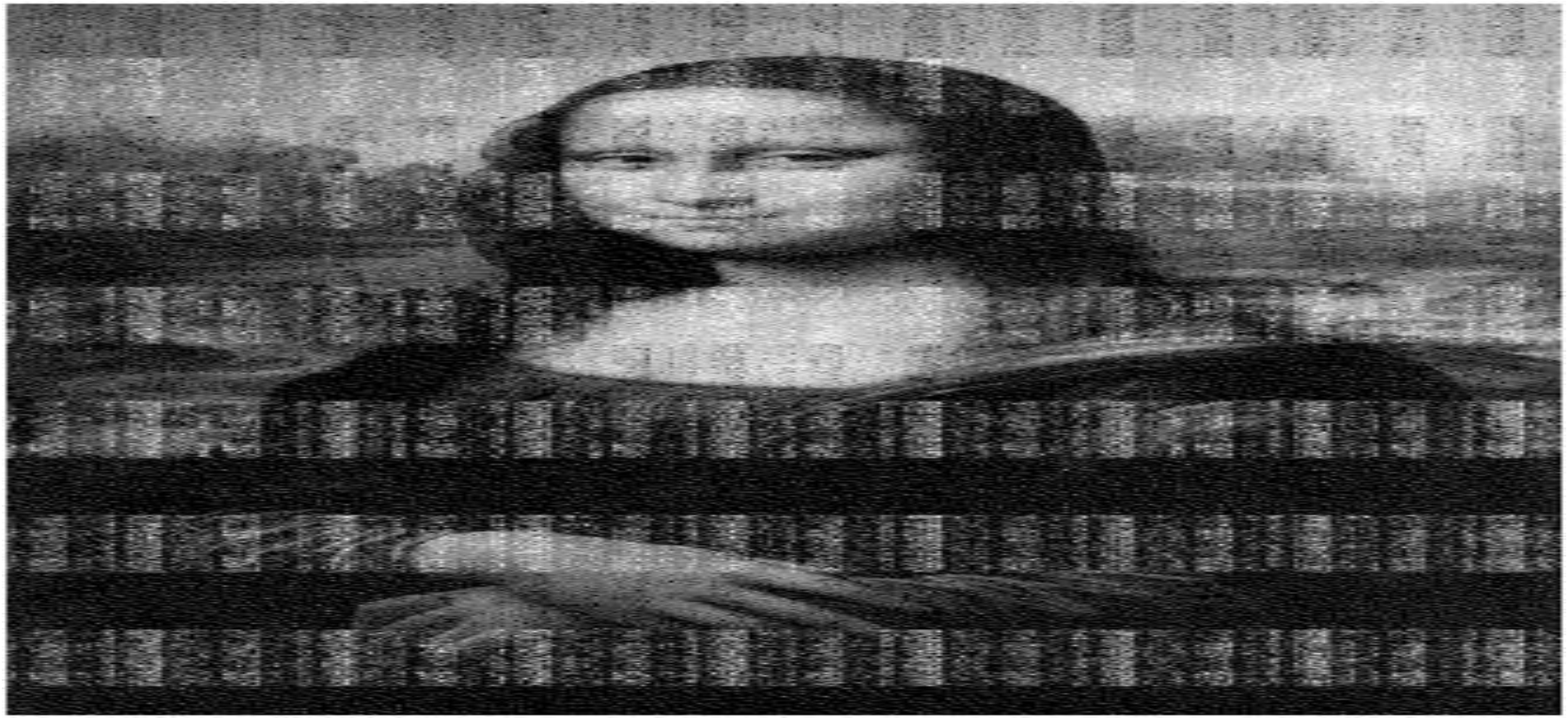


Put It Back

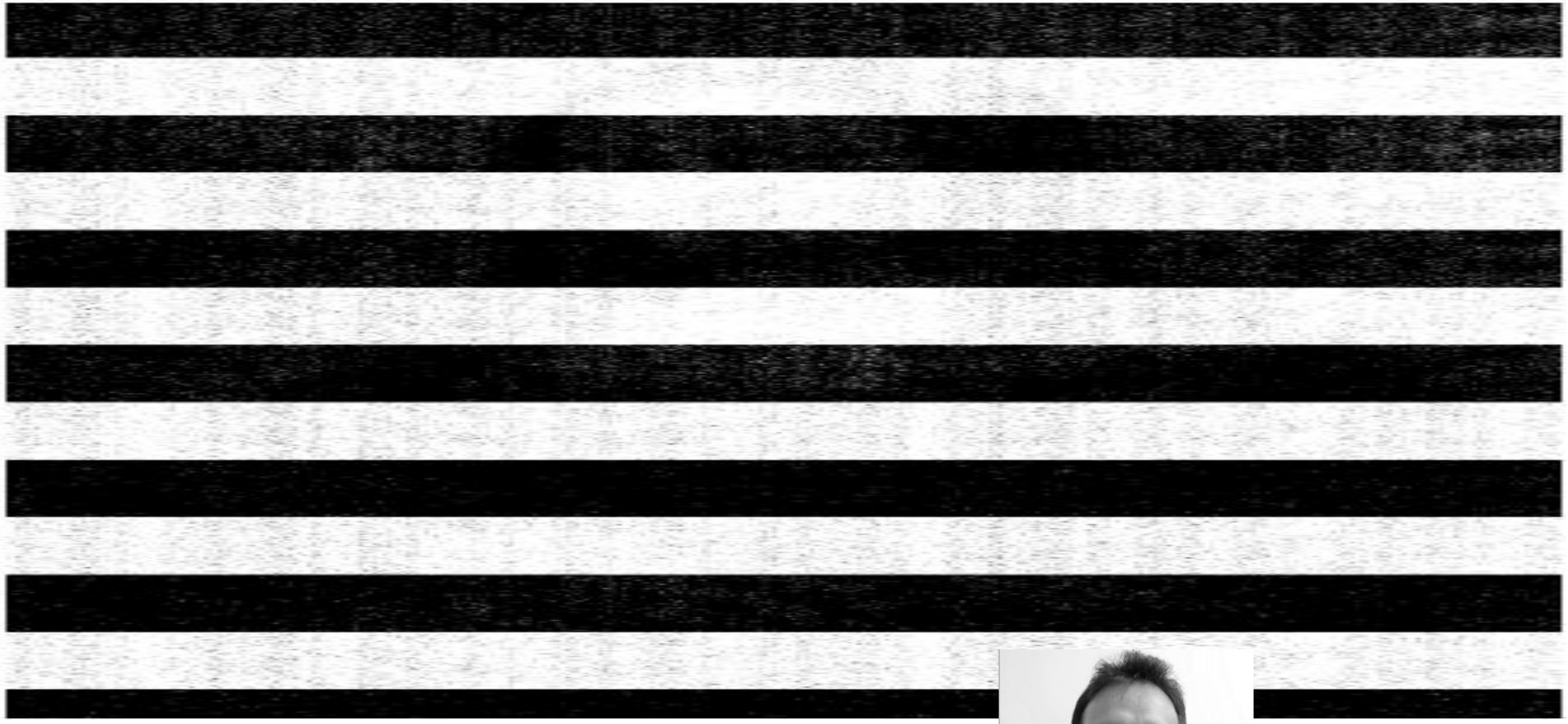


After Five Seconds

After 30 Seconds

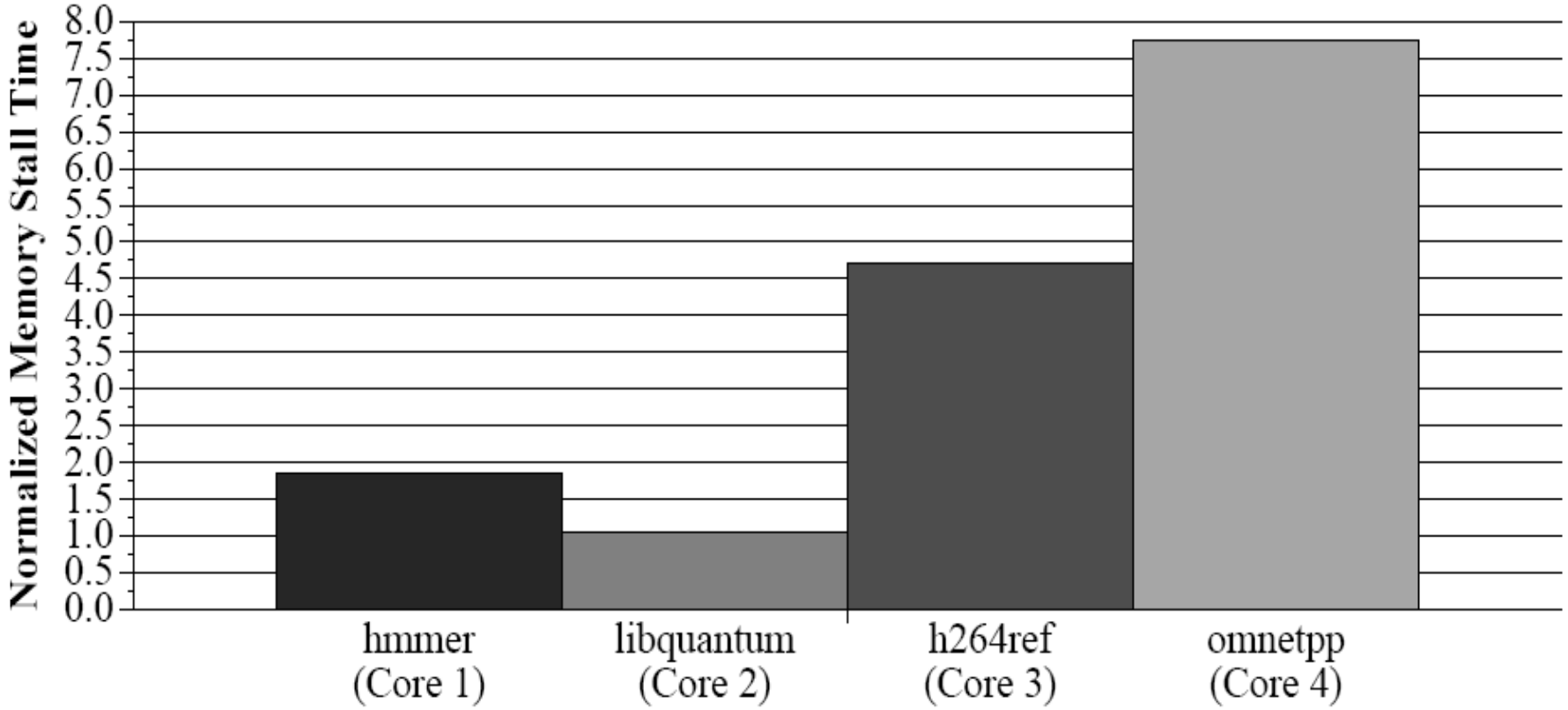


After 300 Seconds



“Think about non-volatile memory”

DOS Attacks: At the DRAM (Availability)



Usenix Security '07

Non-deterministic Slowdown

How?

```
// initialize large arrays A, B
   streaming
for (j=0; j<N; j++) {
    index = j*linesize;
    A[index] = B[index];
    ...
}
```

STREAM

```
// initialize large arrays A, B
    random
for (j=0; j<N; j++) {
    index = rand();
    A[index] = B[index];
    ...
}
```

RANDOM

Sequential memory access	Random
high row buffer locality (96% hit rate)	Low row buffer locality (3% hit rate)
Memory Intensive	Memory Intensive

https://github.com/CMU-SAFARI/Cache-Memory-Hog

Trusted Execution Environments

arm TRUSTZONE

Not resistant to timing channels

Sky is Certainly Not Falling

Offensive side

Media articles *sensationalize* ☹

All the attacks are POCs

Can we make these attacks real?

Defensive side

Microarchitecture need to be fixed to avoid surprises

Can we design secure microarchitecture?

Research Questions

- Relatively new field of research.
- Top forums: USENIX SECURITY, S&P, CCS, NDSS, WOOT along with top computer architecture conferences.
- Performance-security trade-offs
- New attacks, new threat models
- Cat-mouse game ☺

Secure Thanks

Ack

- Some of the slides are adapted and modified from Clementine Maurice