Covert and Side Channel Attacks and Defenses

Mengjia Yan

Fall 2020

Based on slides from Christopher W. Fletcher

Reminder

- Lab assignment will be released 09/21 Monday
  - Recommend to read ”Cache missing for fun and profit.” (2005).

- Check out the presentation schedule on course website
  - 7 slots empty, volunteer or invited speaker or Mengjia/Miles

Resources

- Side channel tutorial website
  - https://sites.google.com/view/arch-sec/home

- External resources
  - Mastik, a toolkit for uarch side channels: https://cs.adelaide.edu.au/~yval/Mastik/
  - Survey on microarchitectural timing attacks: https://eprint.iacr.org/2016/613.pdf
  - Survey on transient execution attacks: https://arxiv.org/abs/1811.05441

What is Covert and Side Channel?

Covert channel:

- **Intended** communication between two or more security parties

Side channel:

- **Unintended** communication between two or more security parties

What is Covert and Side Channel?

Covert channel:

- **Intended** communication between two or more security parties

Side channel:

- **Unintended** communication between two or more security parties

In both cases:

- Communication should not be possible, following system semantics
- The communication medium is not designed to be a communication channel

What is Covert and Side Channel?

Covert channel:

- **Intended** communication between two or more security parties

Side channel:

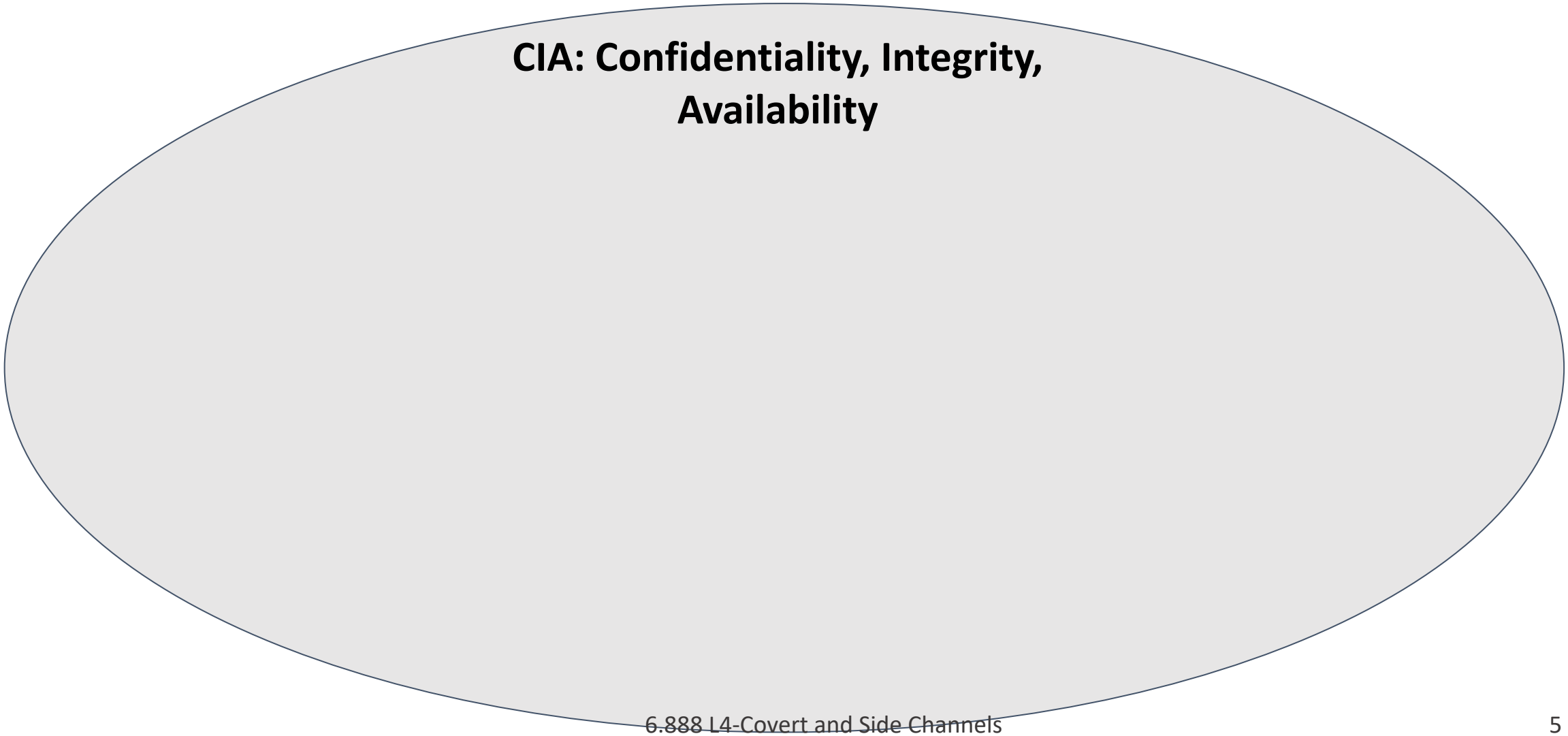
- **Unintended** communication between two or more security parties

In both cases:

- Communication should not be possible, following system semantics
- The communication medium is not designed to be a communication channel

Covert channel can show “best case” leakage

Scope



Scope

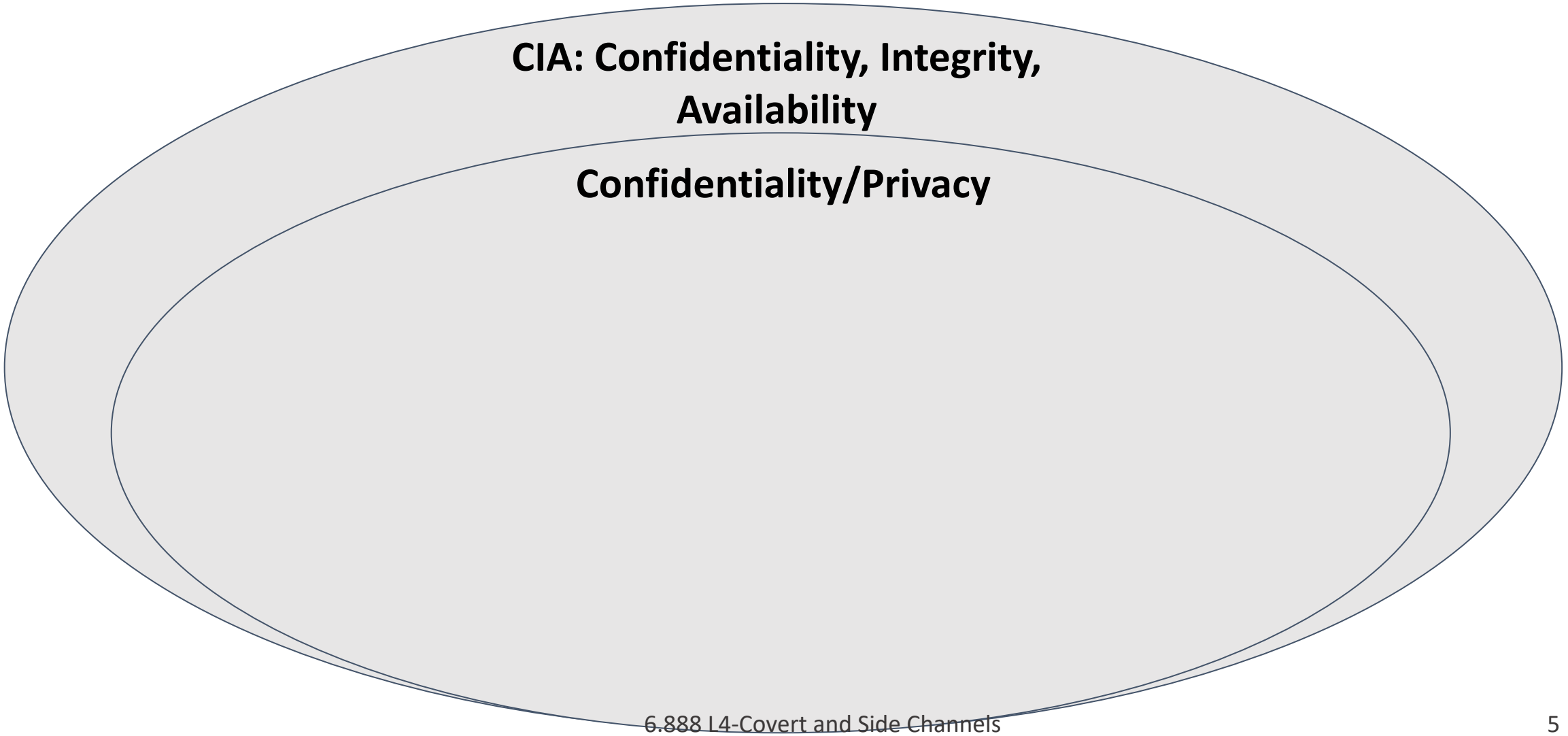
CIA: Confidentiality, Integrity, Availability

Confidentiality: was data being computed upon not revealed to an un-permitted party?

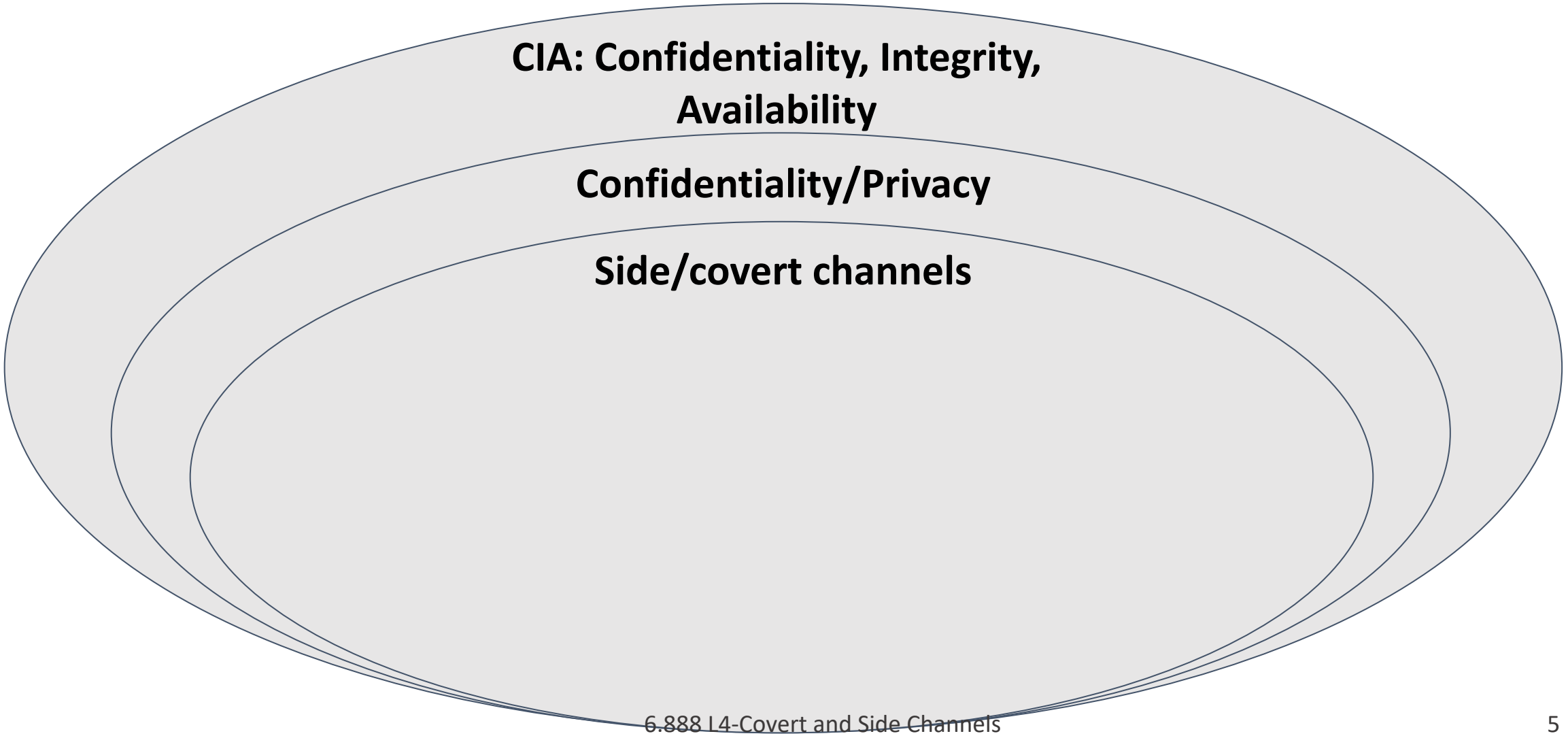
Integrity: was the computation performed correctly, returning the correct result?

Availability: did the computational resource carry out the task at all?

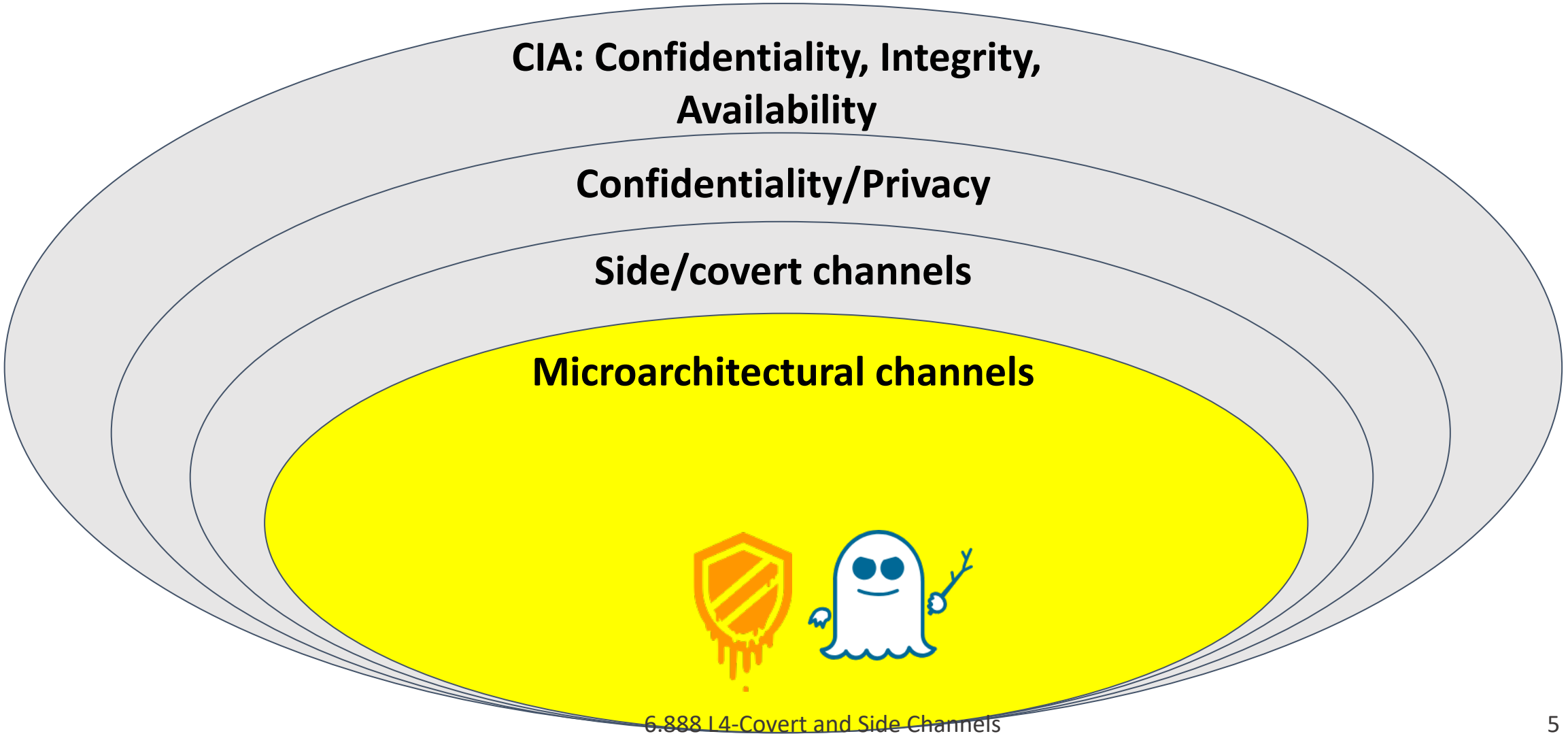
Scope


CIA: Confidentiality, Integrity, Availability
Confidentiality/Privacy

Scope


CIA: Confidentiality, Integrity, Availability
Confidentiality/Privacy
Side/covert channels

Scope


CIA: Confidentiality, Integrity, Availability
Confidentiality/Privacy
Side/covert channels
Microarchitectural channels

Side Channels Are Almost Everywhere

Daily Life Examples

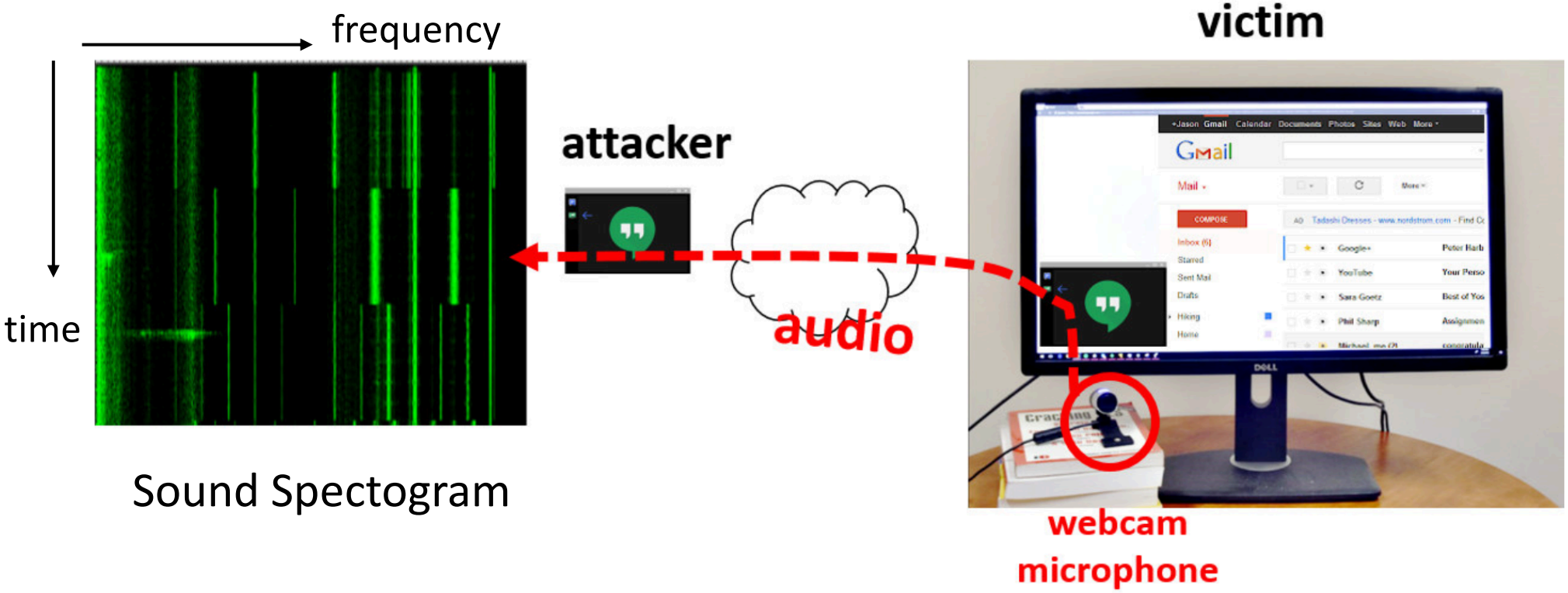
- Acoustic side channels
  - Monitor keystrokes
  - You only need: a cheap microphone + an ML model

Daily Life Examples

- Acoustic side channels
  - Monitor keystrokes
  - You only need: a cheap microphone + an ML model

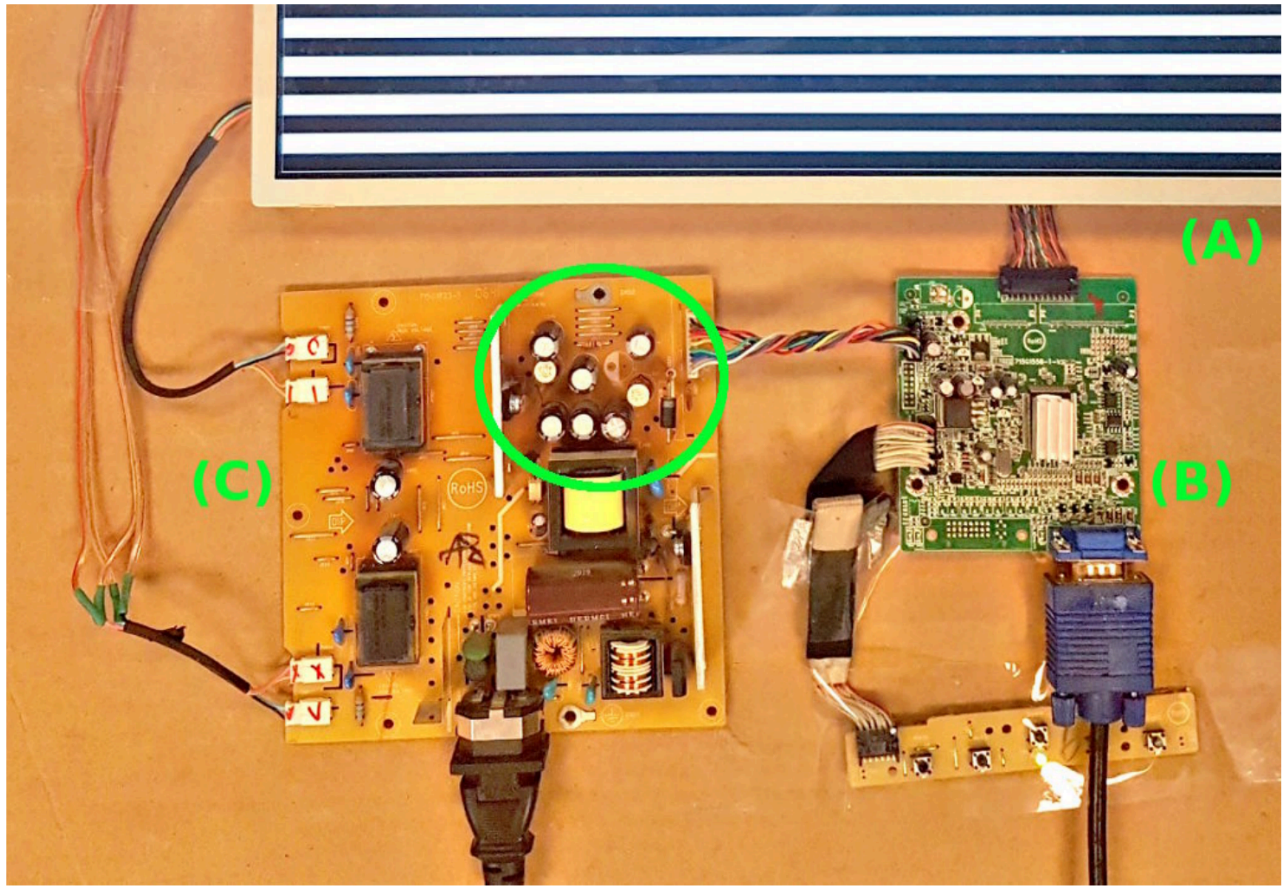
- Network traffic contention side channel
  - If you want to be an active attacker, try stress test

“Hear” The Screen



Genkin et. al. Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels. S&P’19

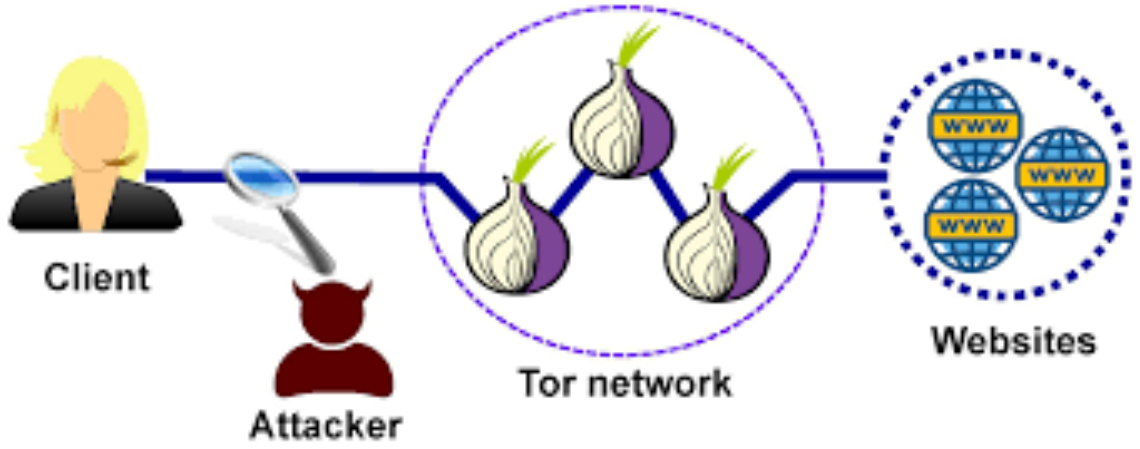
“Hear” The Screen



(A) is the LCD panel, (B) is the screen’s digital logic and image rendering board and, (C) is the screen’s power supply board.

Network Side Channels

- Website Fingerprinting

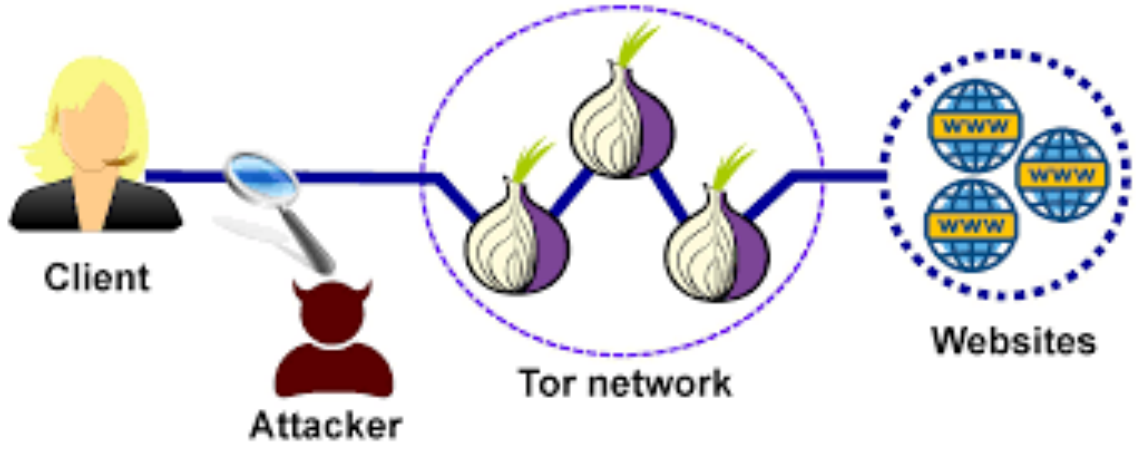


Lescisin et. al. Tools for Active and Passive Network Side-Channel Detection for Web Applications. WOOT'18
Cai et. al. Touching from a distance: Website fingerprinting attacks and defenses. CCS'12.

Network Side Channels

- Website Fingerprinting

- Response dependent:
  - iSideWith.com
- Real-time feedback:
  - Google Search auto-complete



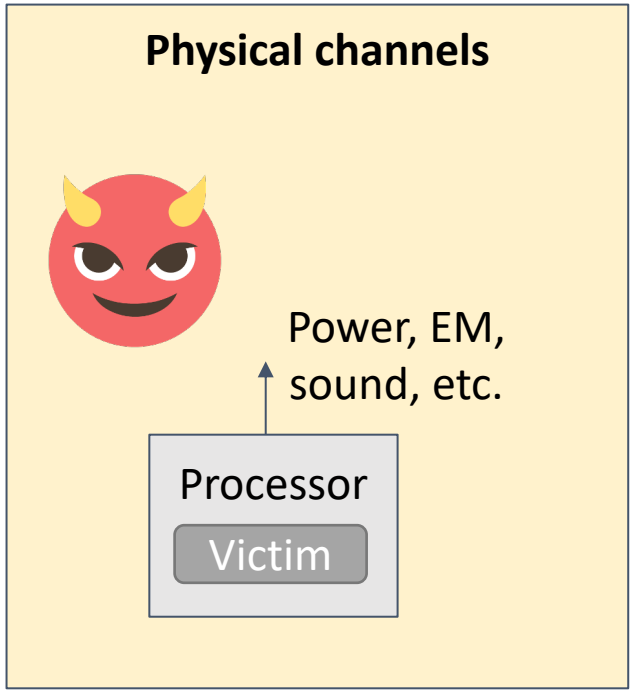
Lescisin et. al. Tools for Active and Passive Network Side-Channel Detection for Web Applications. WOOT'18
Cai et. al. Touching from a distance: Website fingerprinting attacks and defenses. CCS'12.

Physical v.s. Timing v.s. uArch Channel

- What can the adversary observe?

Physical v.s. Timing v.s. uArch Channel

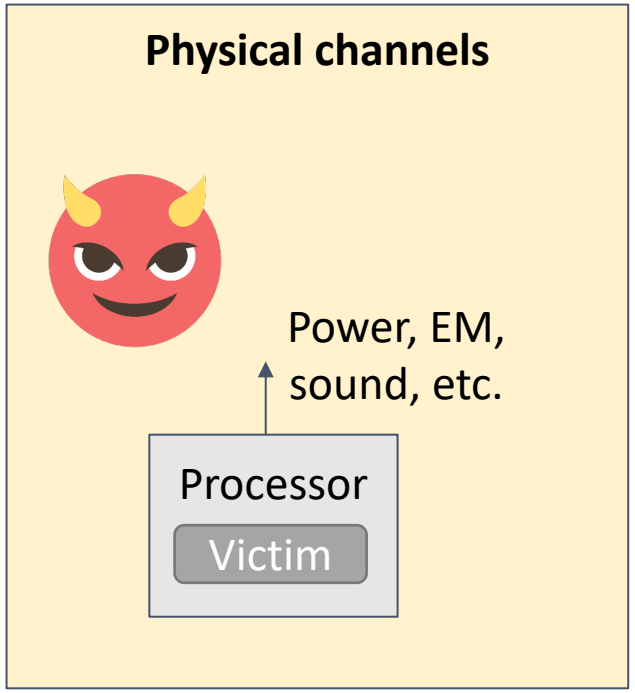
- What can the adversary observe?



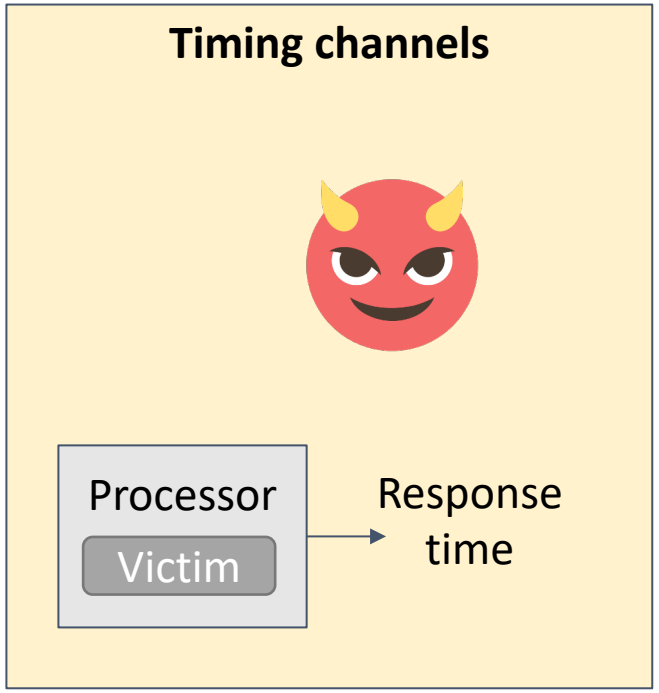
Attacker requires measurement equipment → physical access

Physical v.s. Timing v.s. uArch Channel

- What can the adversary observe?



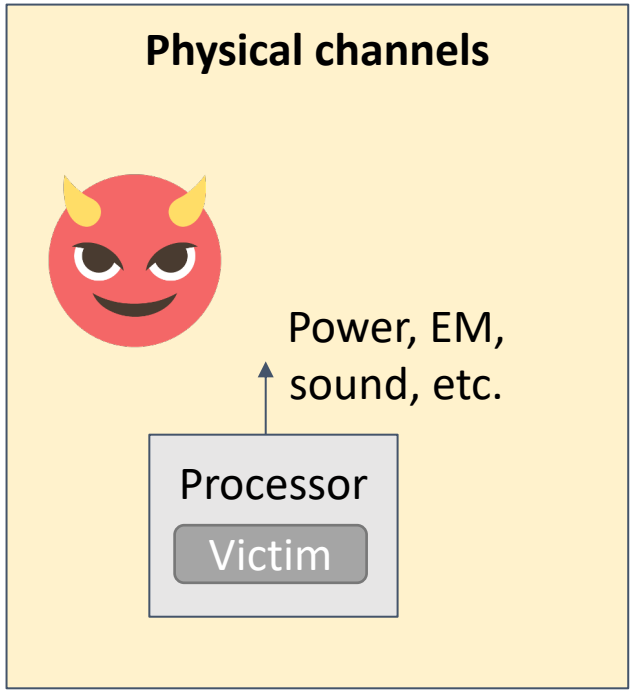
Attacker requires measurement equipment → physical access



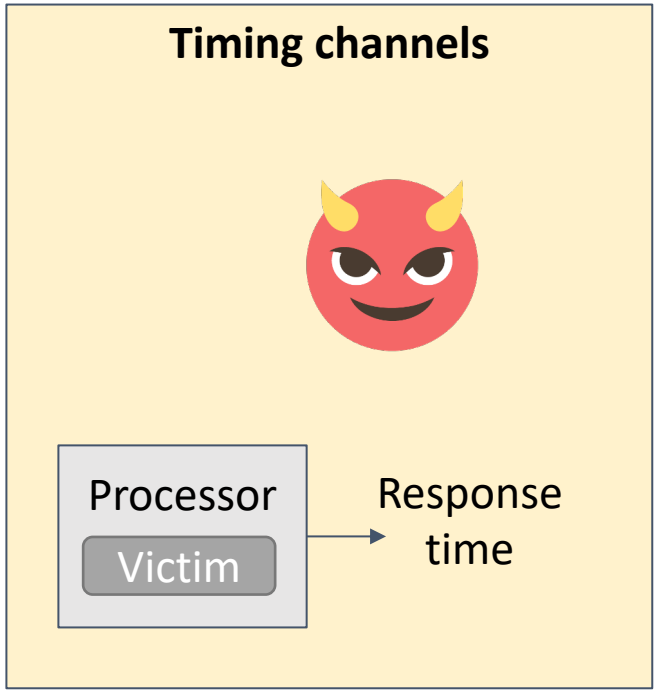
Attacker may be remote (e.g., over an internet connection)

Physical v.s. Timing v.s. uArch Channel

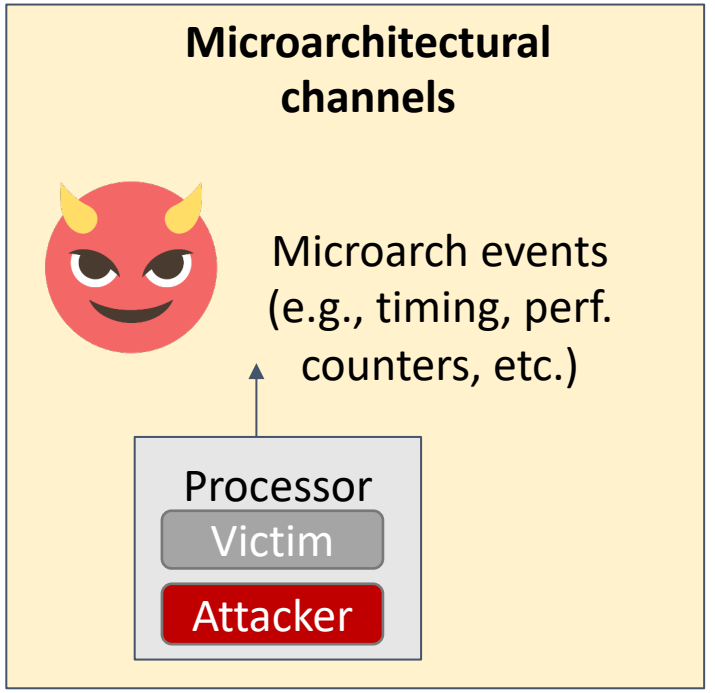
- What can the adversary observe?



Attacker requires measurement equipment → physical access

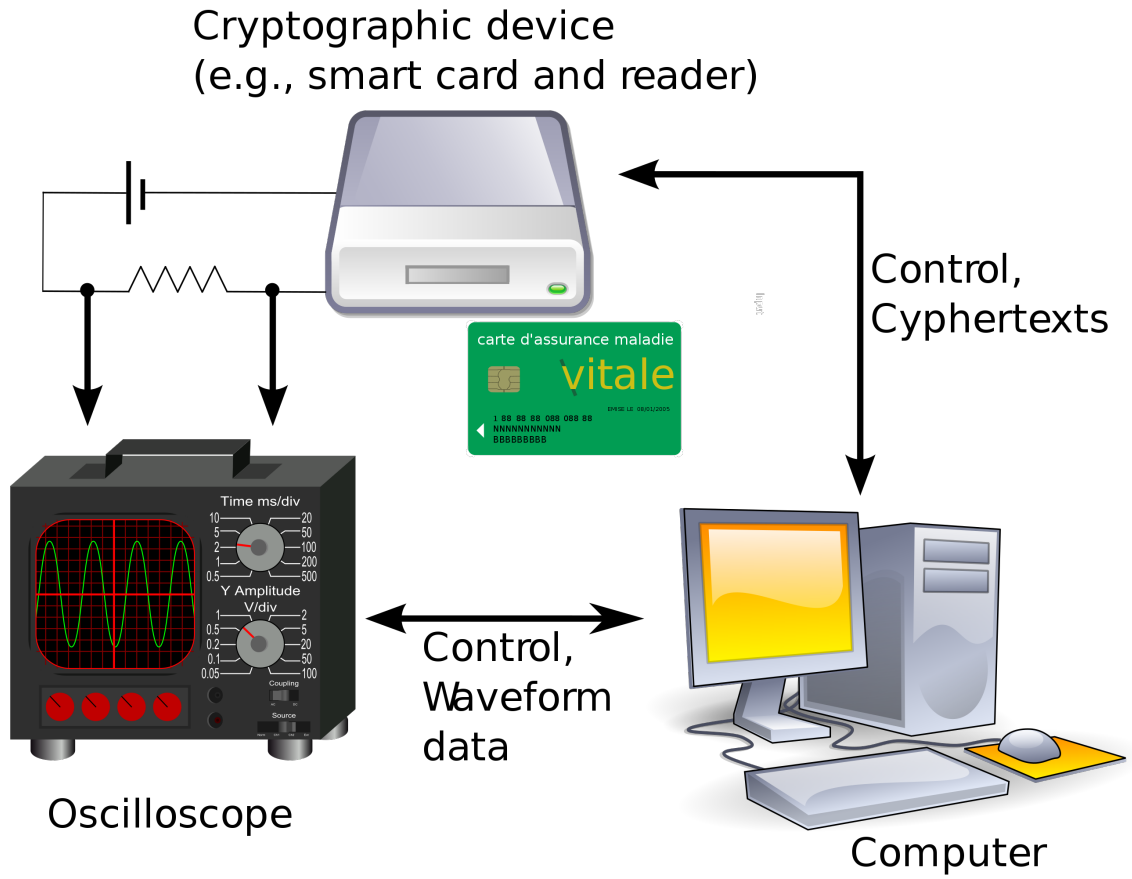


Attacker may be remote (e.g., over an internet connection)



Attacker may be remote, or be co-located

Power Analysis



from https://en.wikipedia.org/wiki/Power_analysis

Victim Application: RSA

- Square-and-multiply based exponentiation

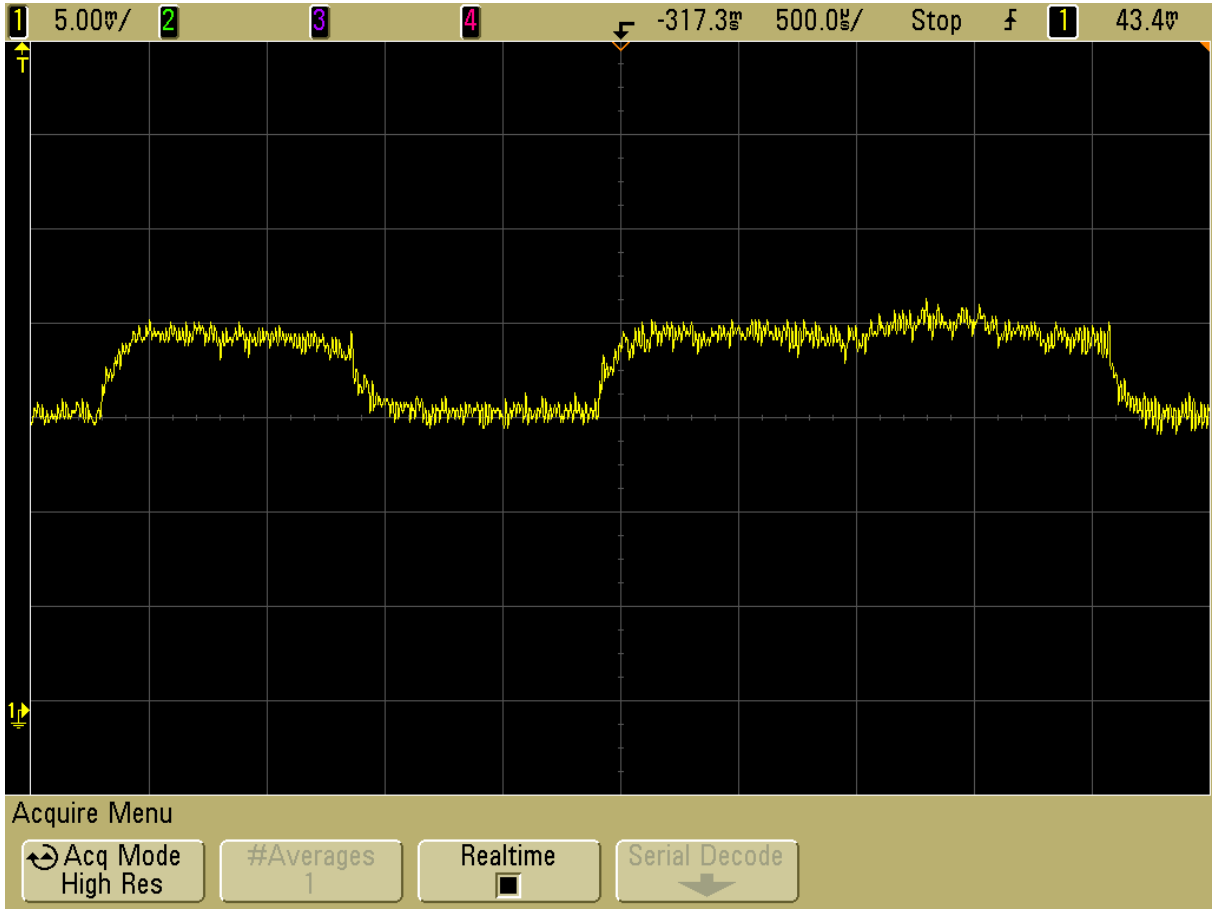
Input : base b, modulo m, exponent $e = (e_{n-1} \ldots e_0)_2$
Output: $b^e \bmod m$
$r = 1$
for $i = n-1$ *down to* 0 **do**
    $r = sqr(r)$
    $r = mod(r,m)$
    if $e_i == 1$ **then**
        $r = mul(r,b)$
        $r = mod(r,m)$
    end
end
return r

Victim Application: RSA

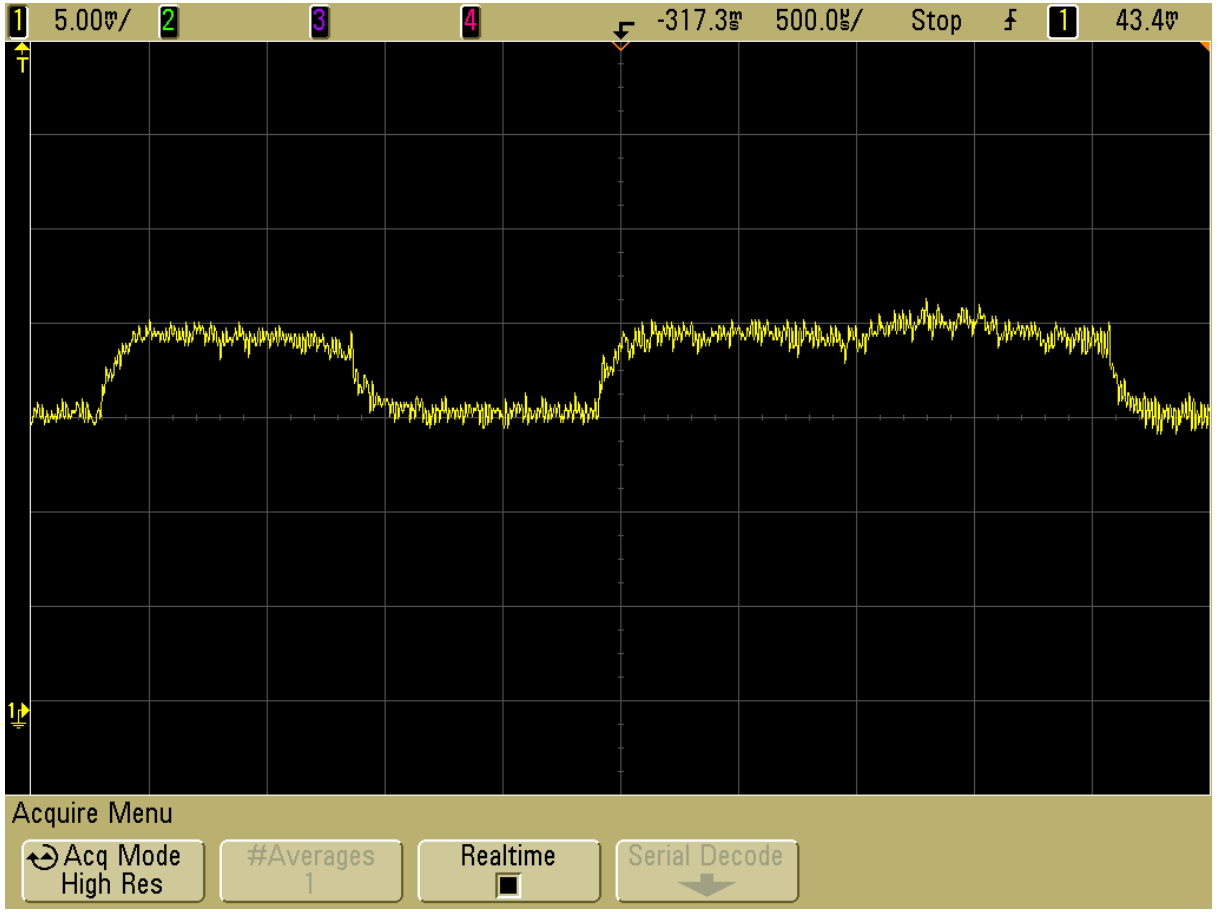
- Square-and-multiply based exponentiation

Input : base b, modulo m, exponent $e = (e_{n-1} \ldots e_0)_2$
Output: $b^e \bmod m$
$r = 1$
for $i = n-1$ *down to* 0 **do**
    $r = sqr(r)$
    $r = mod(r,m)$
    if $e_i == 1$ **then**
        $r = mul(r,b)$
        $r = mod(r,m)$
    end
end
return r

Power Analysis



Power Analysis



- Various signal processing techniques to de-noise.
- More advanced: differential power analysis (DPA)

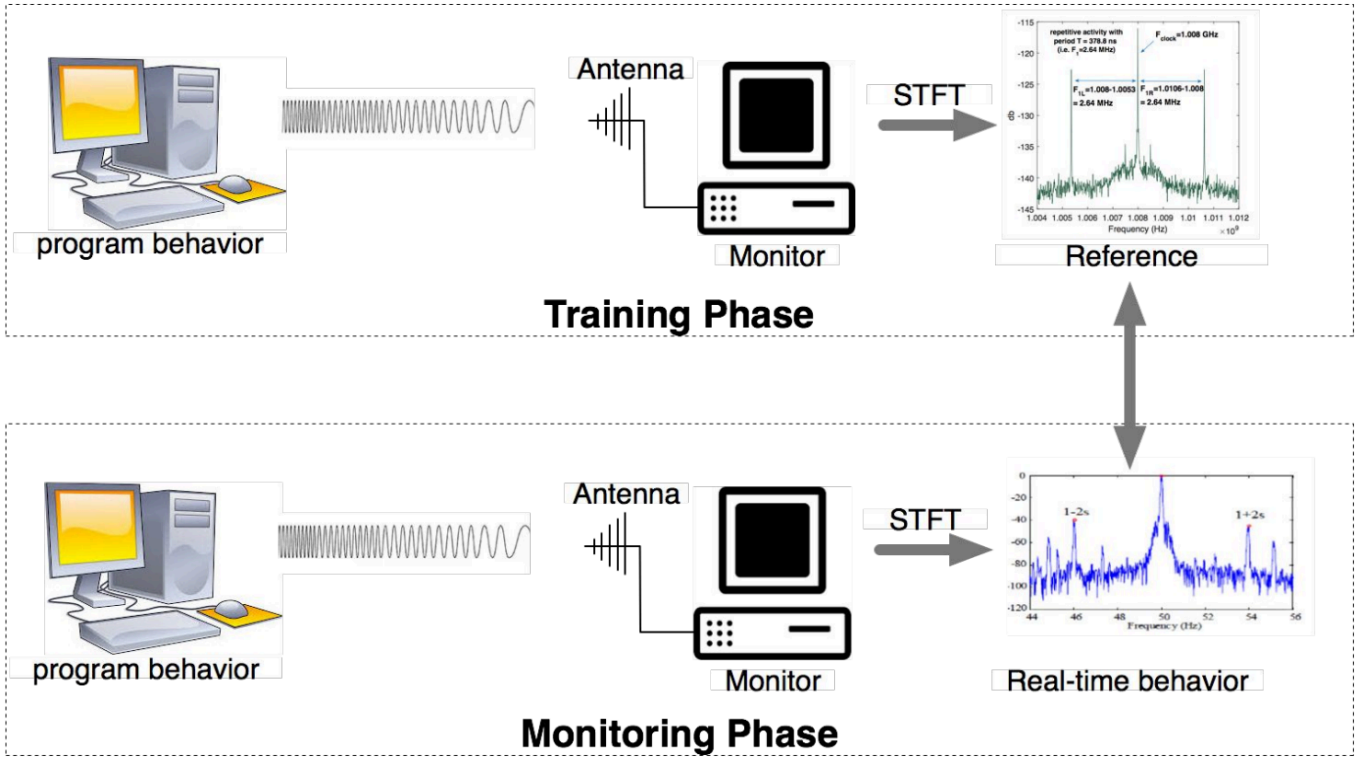
Benign Usage: Non-intrusive Software Monitoring

- How to efficiently monitor application for anomaly detection?

Sehatbakhsh et al. Spectral Profiling: Observer-Effect-Free Profiling by Monitoring EM Emanations. MICRO'16

Benign Usage: Non-intrusive Software Monitoring

- How to efficiently monitor application for anomaly detection?



Sehatbakhsh et al. Spectral Profiling: Observer-Effect-Free Profiling by Monitoring EM Emanations. MICRO'16

What can you do with these channels?

- Violate privilege boundaries
  - Inter-process communication
  - Infer an application's secret
- (Semi-Invasive) application profiling

What can you do with these channels?

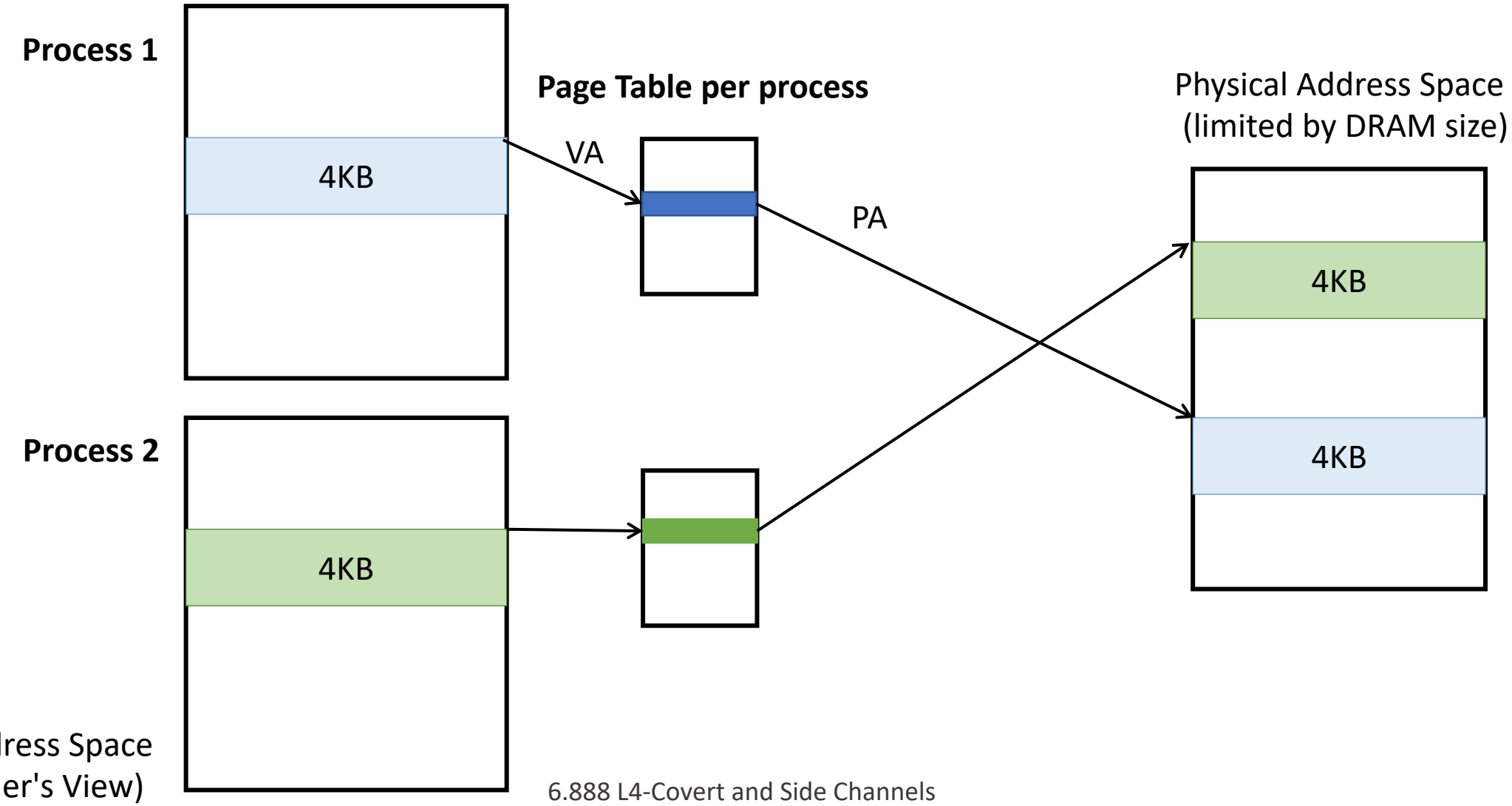
- Violate privilege boundaries
  - Inter-process communication
  - Infer an application's secret
- (Semi-Invasive) application profiling

Different from traditional software or physical attacks:

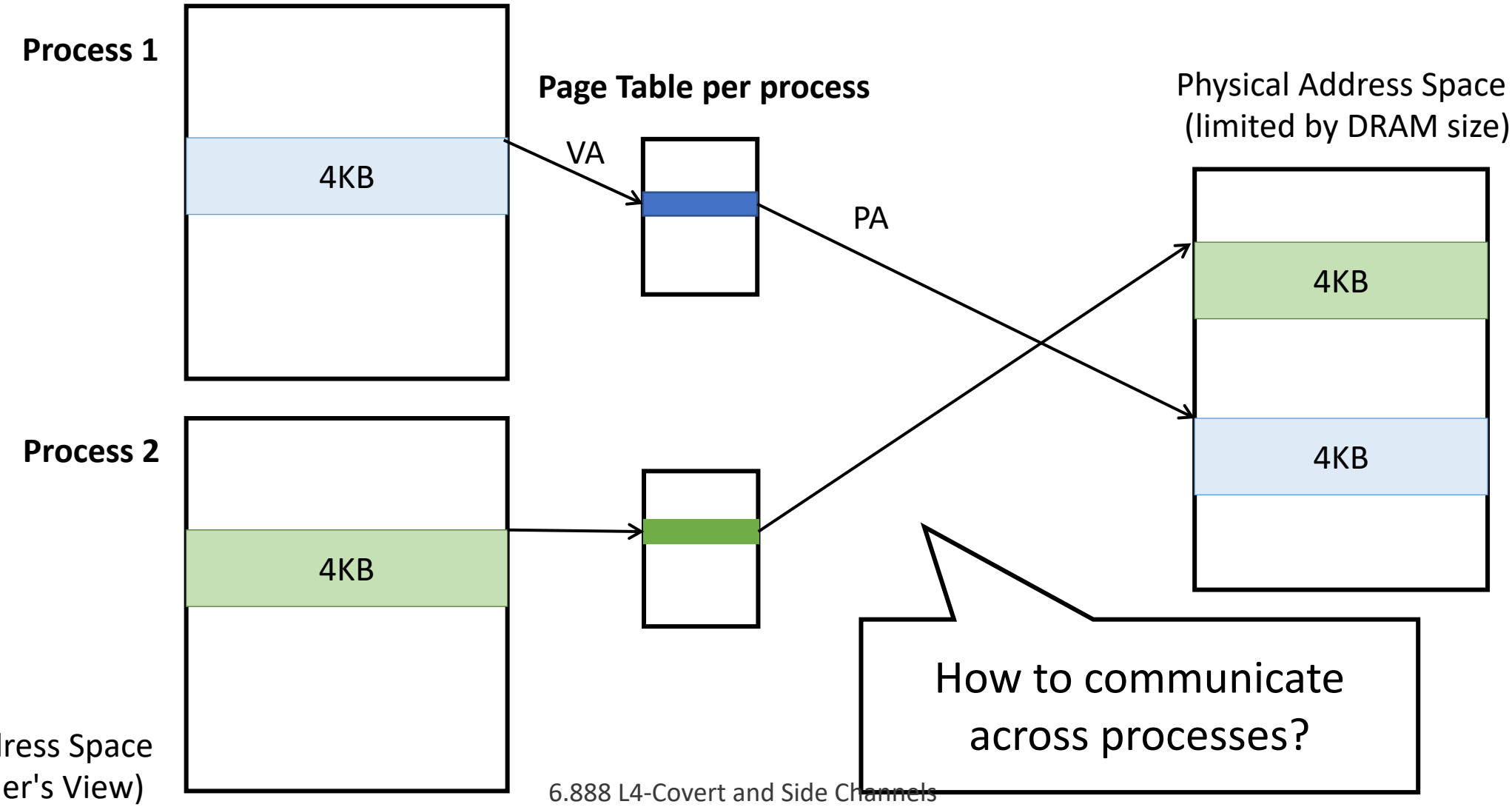
- Stealthy. Sophisticated mechanisms needed to detect channel
- Usually no permanent indication one has been exploited

uArch Side Channels

Recap: Process Isolation



Recap: Process Isolation



Normal Cross-process Communication

```
include <socket.h>

void send(bit msg) {
  socket.send(msg);
}

bit recv() {
  return socket.recv(msg);
}
```

Normal Cross-process Communication

```
include <socket.h>

void send(bit msg) {
  socket.send(msg);
}

bit recv() {
  return socket.recv(msg);
}
```

How to communication without letting OS know?

Normal Cross-process Communication

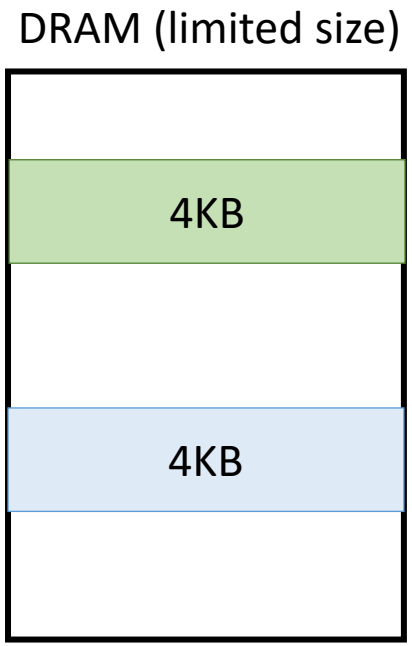
```
include <socket.h>

void send(bit msg) {
  socket.send(msg);
}

bit recv() {
  return socket.recv(msg);
}
```

How to communication without letting OS know?

--> Use HW contention

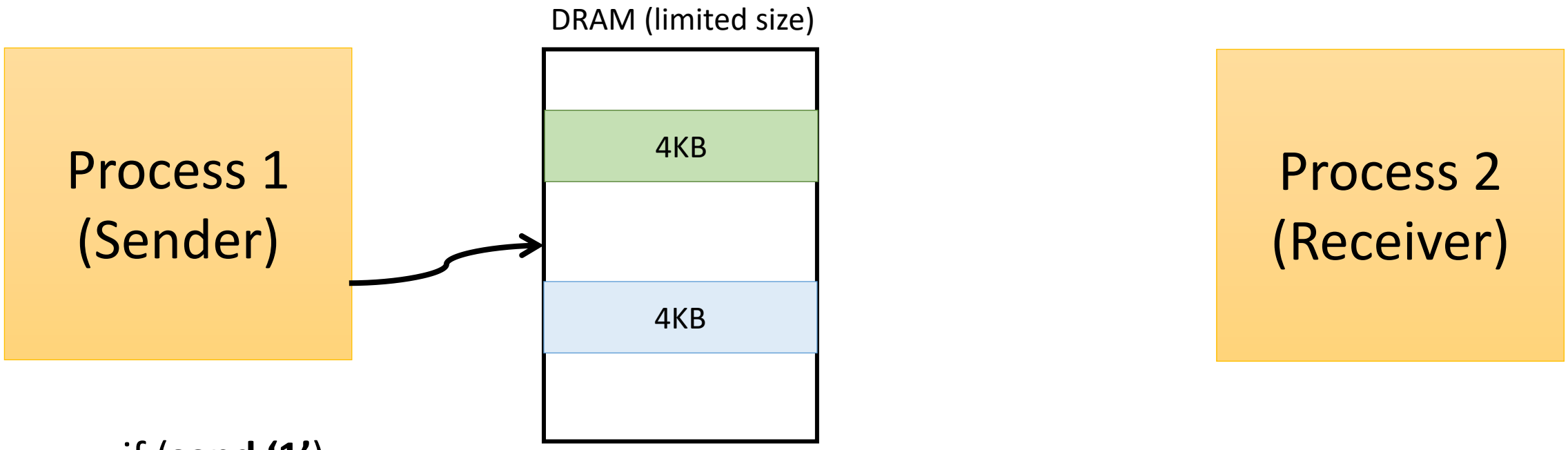
Covert Channels 101: Through the Page Fault

Process 1
(Sender)



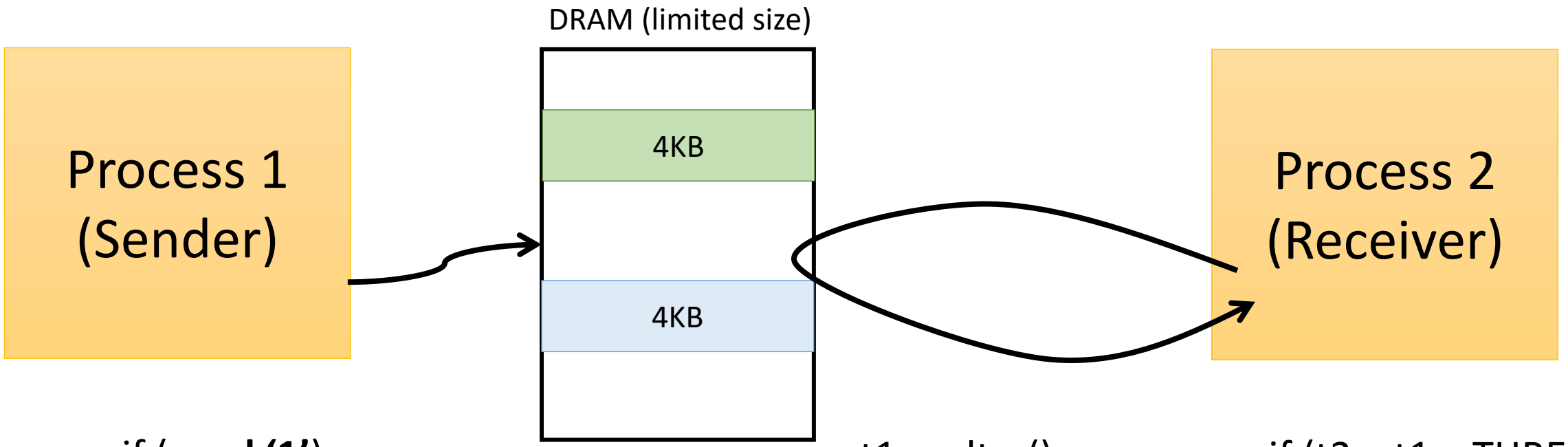
Process 2
(Receiver)

Covert Channels 101: Through the Page Fault



if (**send '1'**)
  accesses many pages
else
  idle

Covert Channels 101: Through the Page Fault

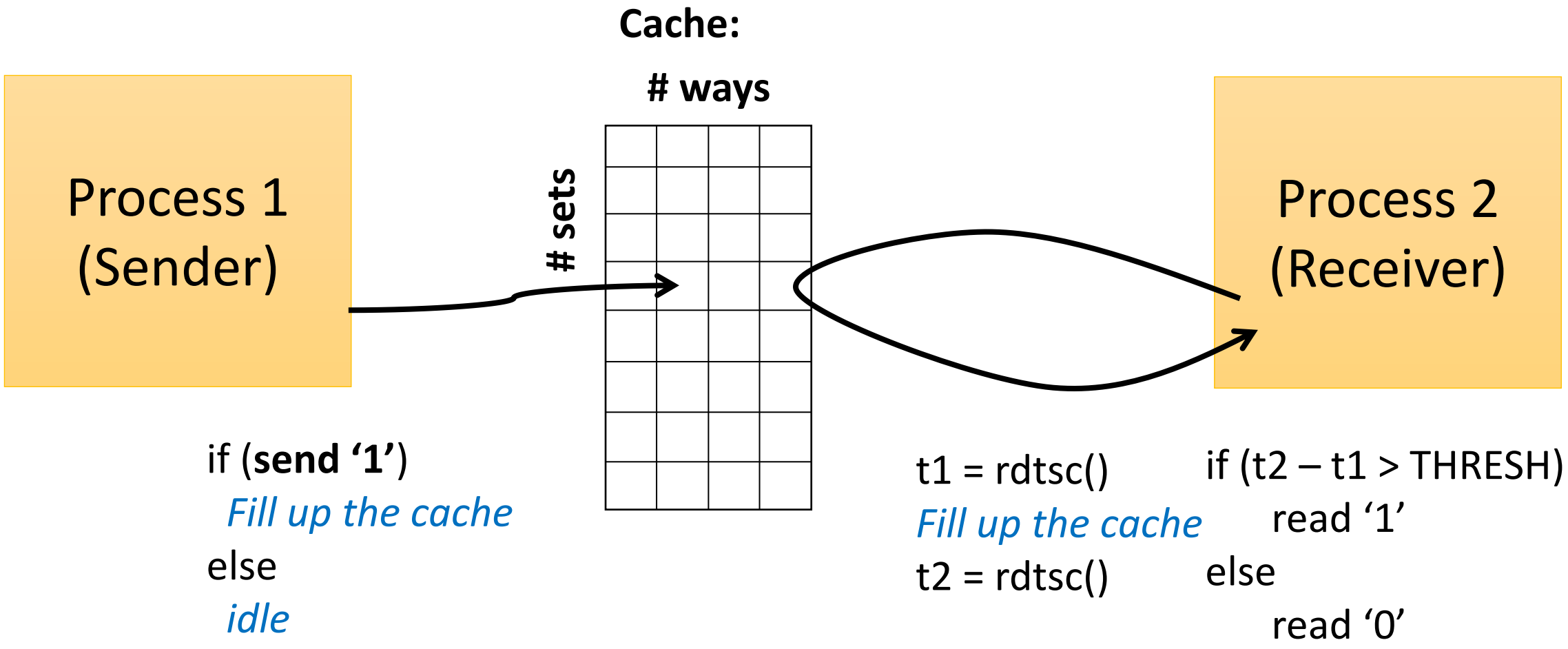


if (**send '1'**)
  accesses many pages
else
  idle

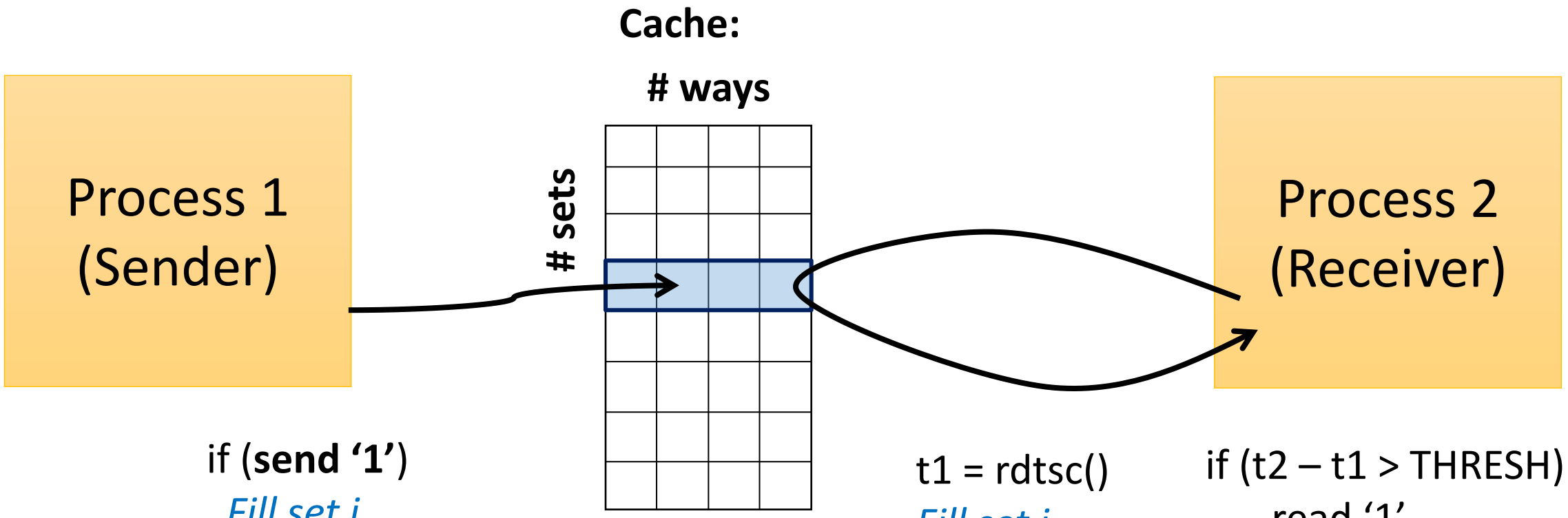
t1 = rdtsc()
Accesses many pages
t2 = rdtsc()

if (t2 – t1 > THRESH)
    read '1'
else
    read '0'

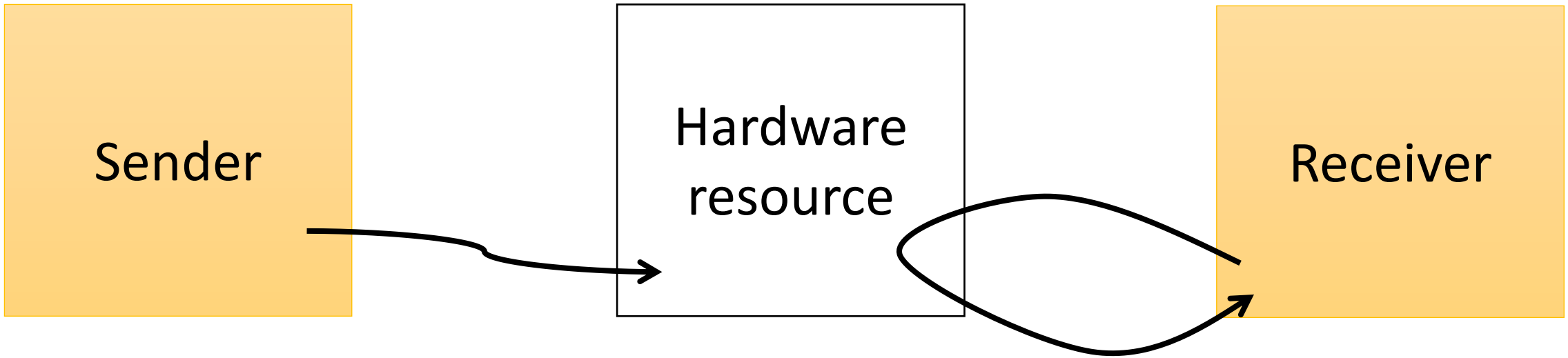
Another Example of Using Caches



Faster Communication



Generalizes to Channels Beyond Caches

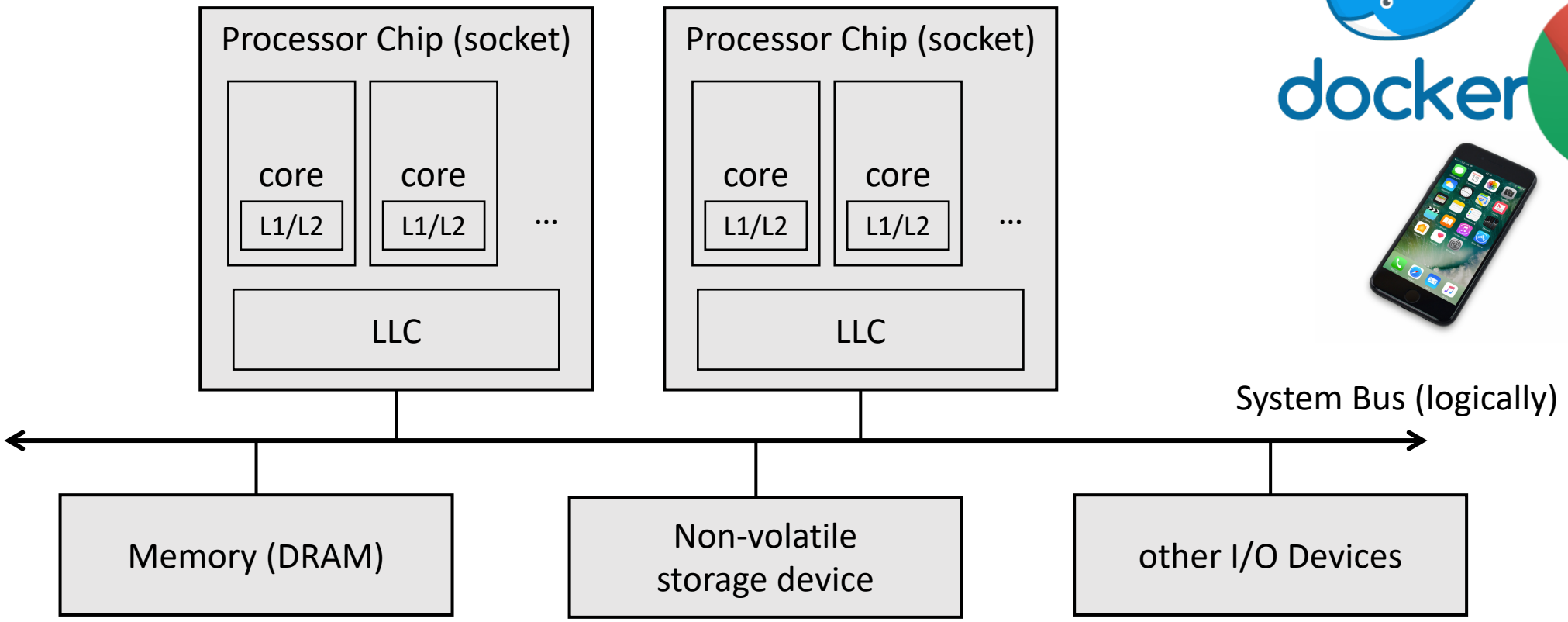


if (**send '1'**)
  Use resource
else
  idle

t1 = rdtsc()
Use resource
t2 = rdtsc()

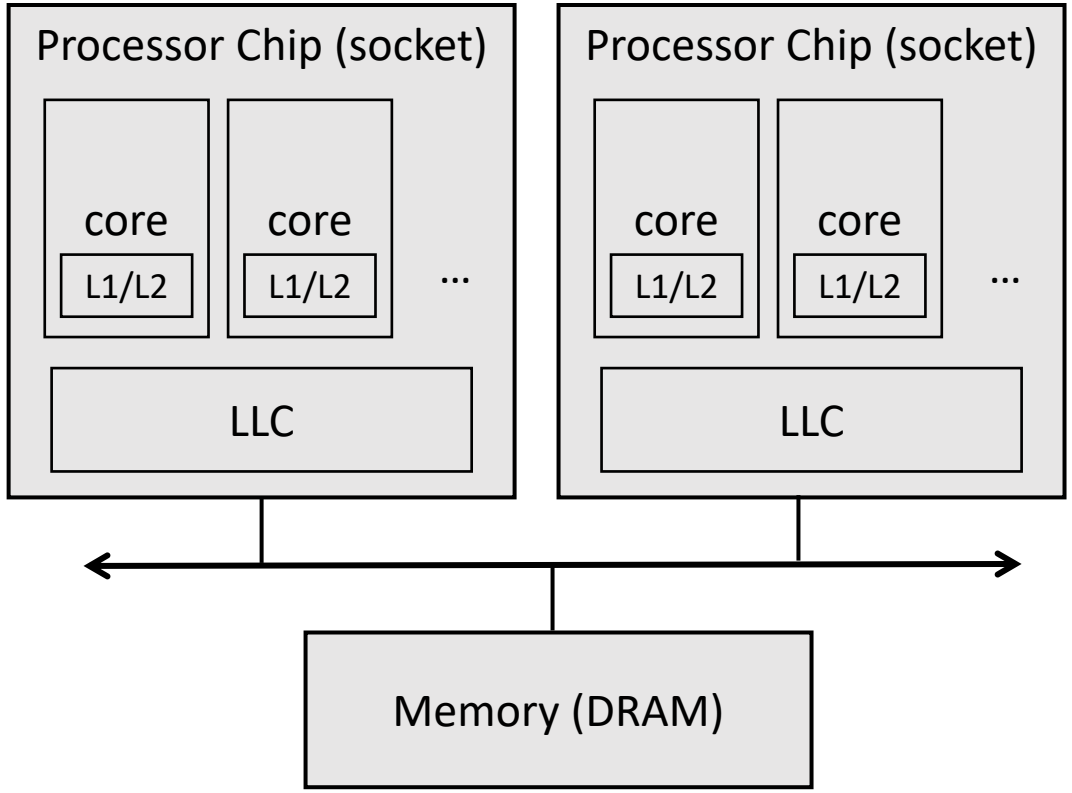
if (t2 – t1 > THRESH)
    read '1'
else
    read '0'

HW Resource Contention



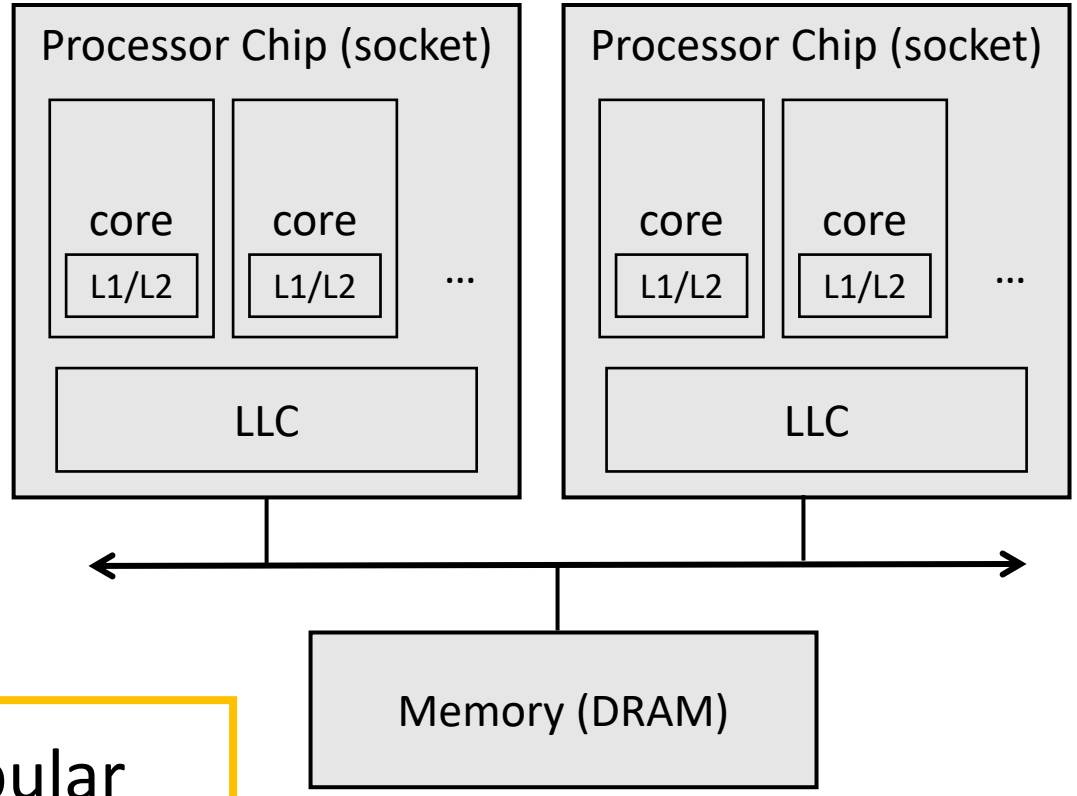
The Memory Hierarchy

- L1, L2
  - Shared by threads on the same core
- LLC:
  - Shared by threads on different cores
- Directory:
  - Shared by threads on different sockets
- DRAM row buffer:
  - Shared by …..



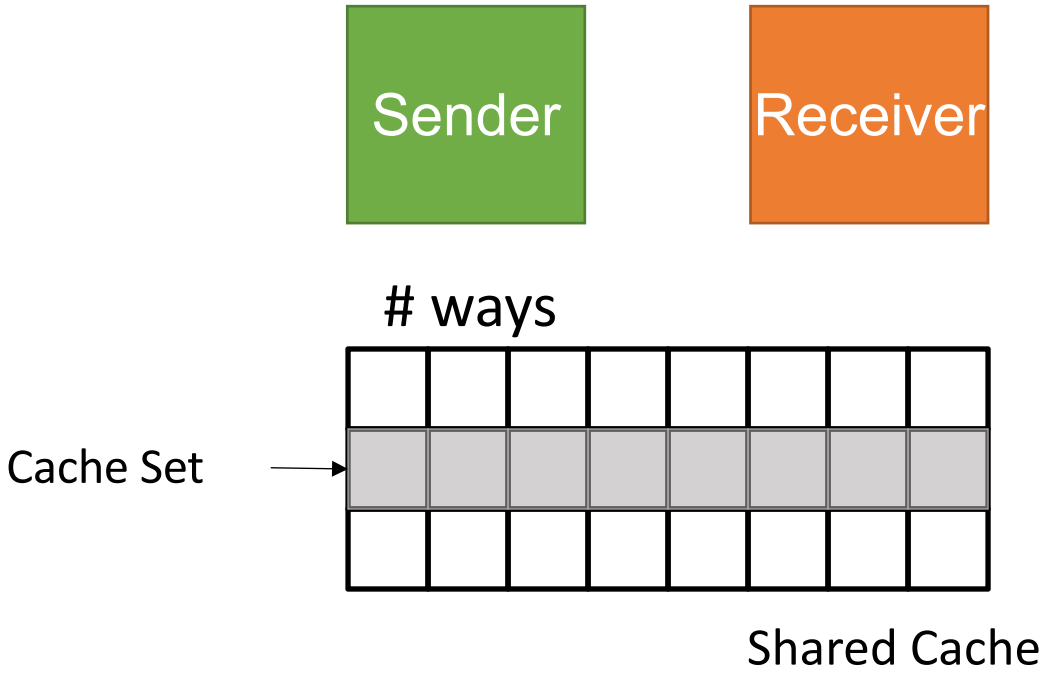
The Memory Hierarchy

- L1, L2
  - Shared by threads on the same core
- LLC:
  - Shared by threads on different cores
- Directory:
  - Shared by threads on different sockets
- DRAM row buffer:
  - Shared by …..

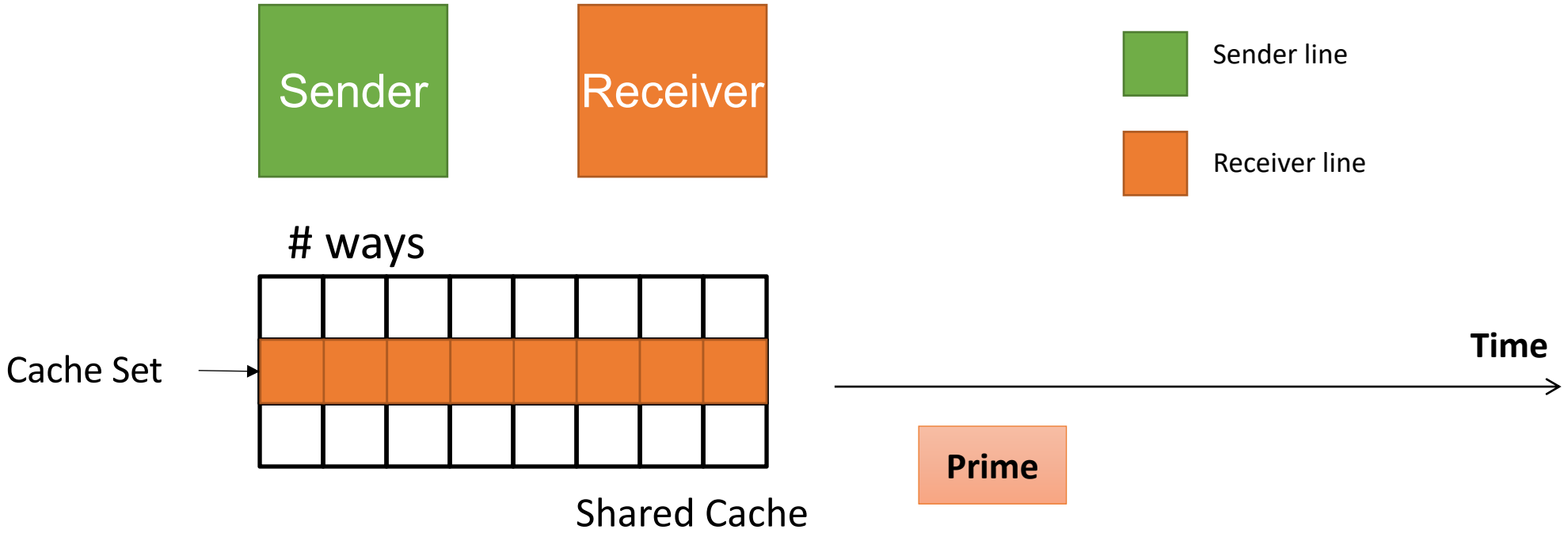


Cache is a popular attack target. Why?

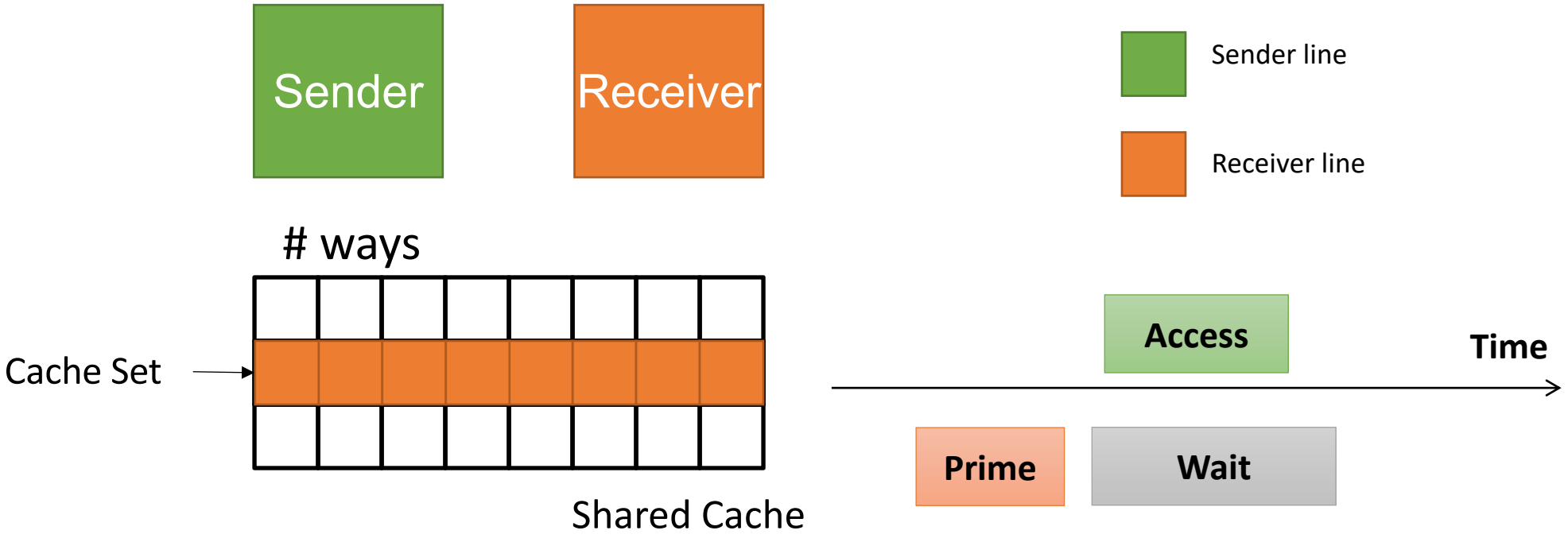
Protocol 101: Prime+Probe in the Cache



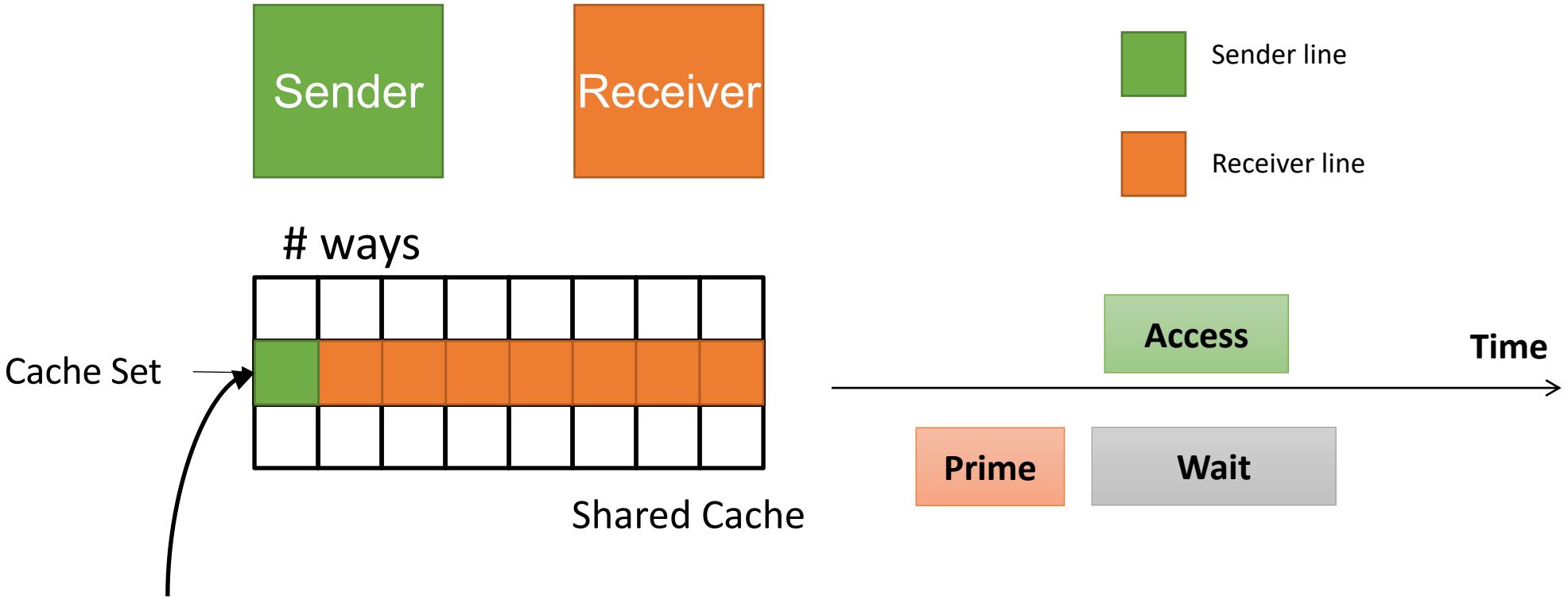
Prime+Probe


Sender
Receiver
Sender line
Receiver line
ways
Cache Set
Time
Prime
Shared Cache

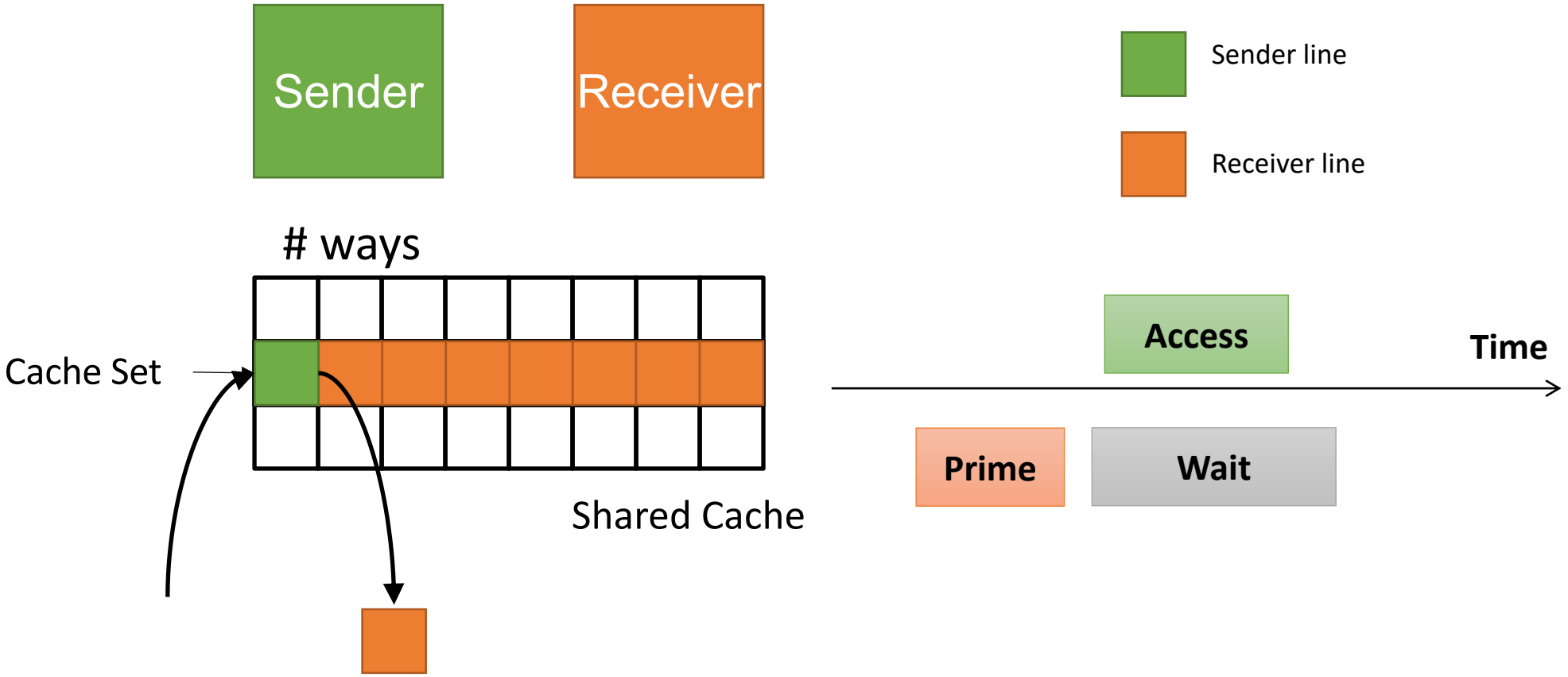
Prime+Probe – Send “1”



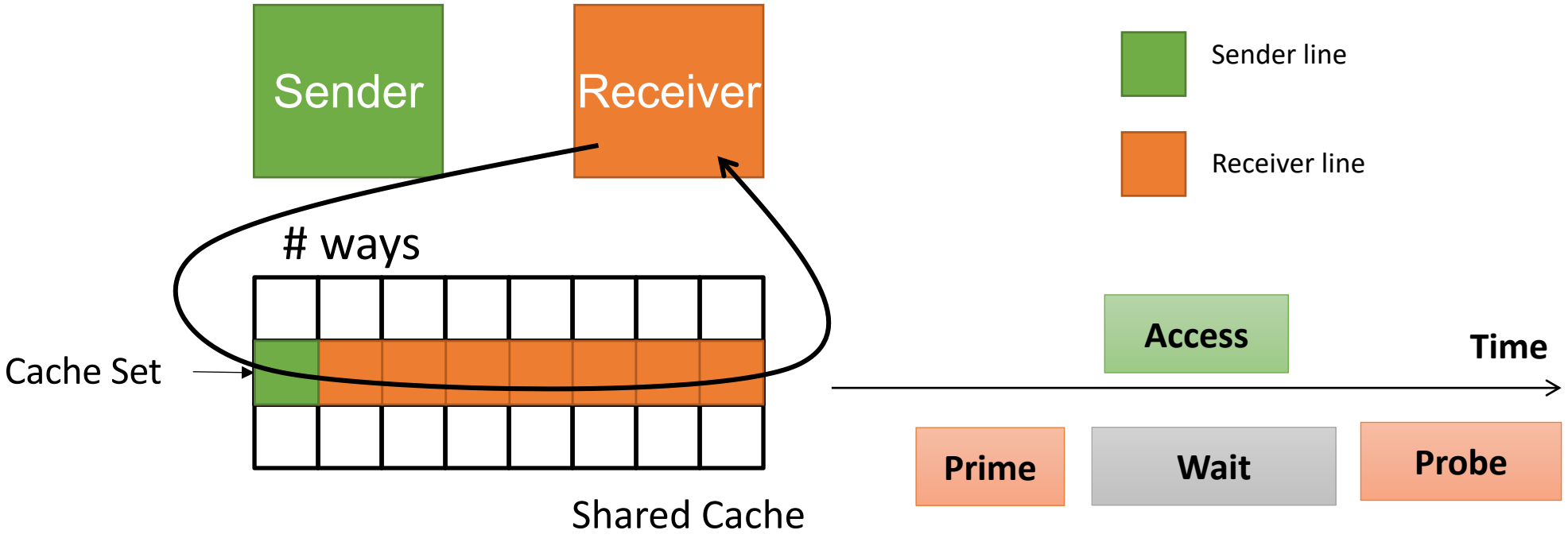
Prime+Probe – Send “1”



Prime+Probe – Send “1”

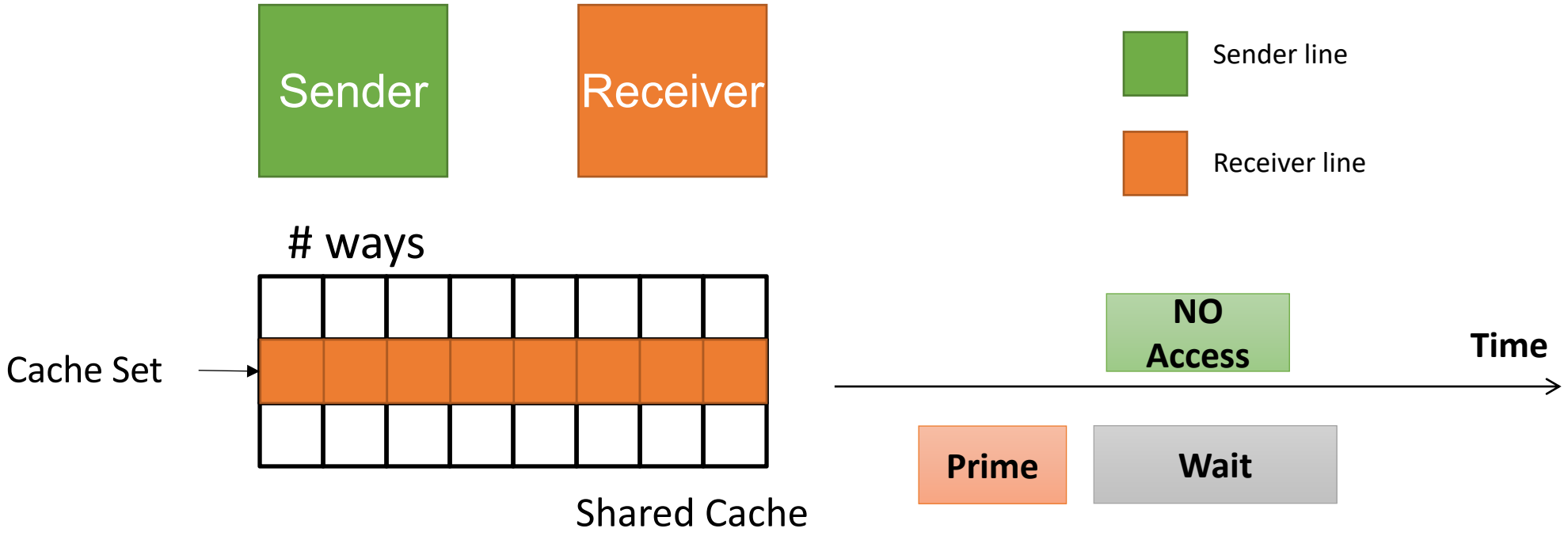


Prime+Probe – Receive “1”

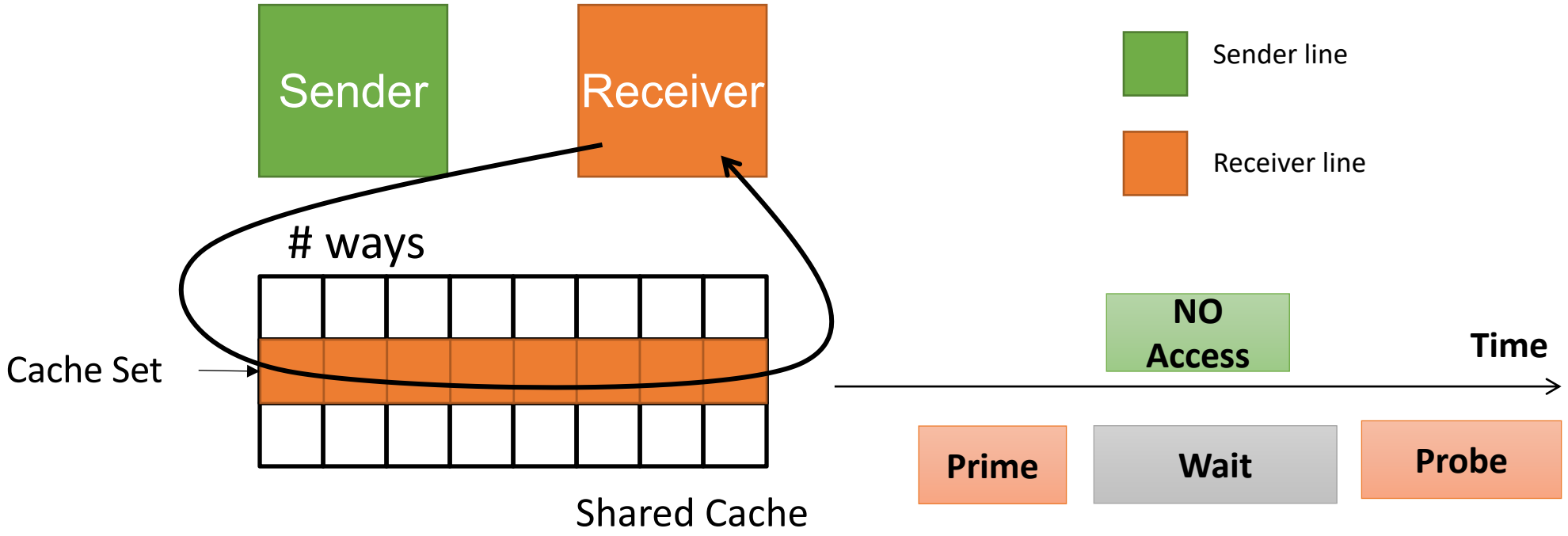


Receive “1” = 16 accesses → 1 miss

Prime+Probe – Send “0”

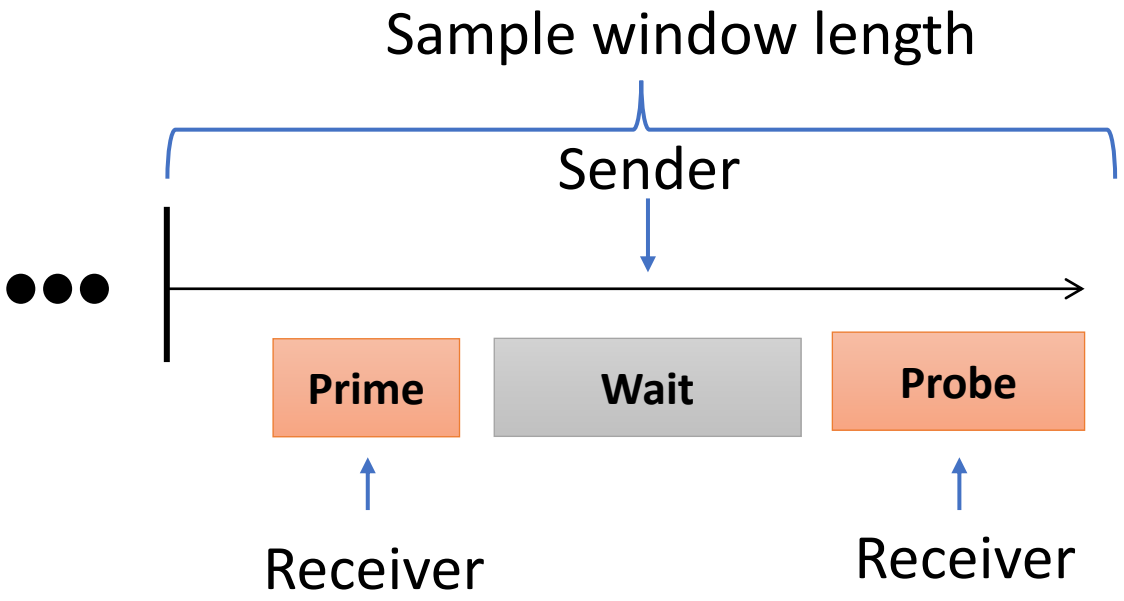


Prime+Probe – Receive “0”



Receive “0” = 16 accesses → 0 miss

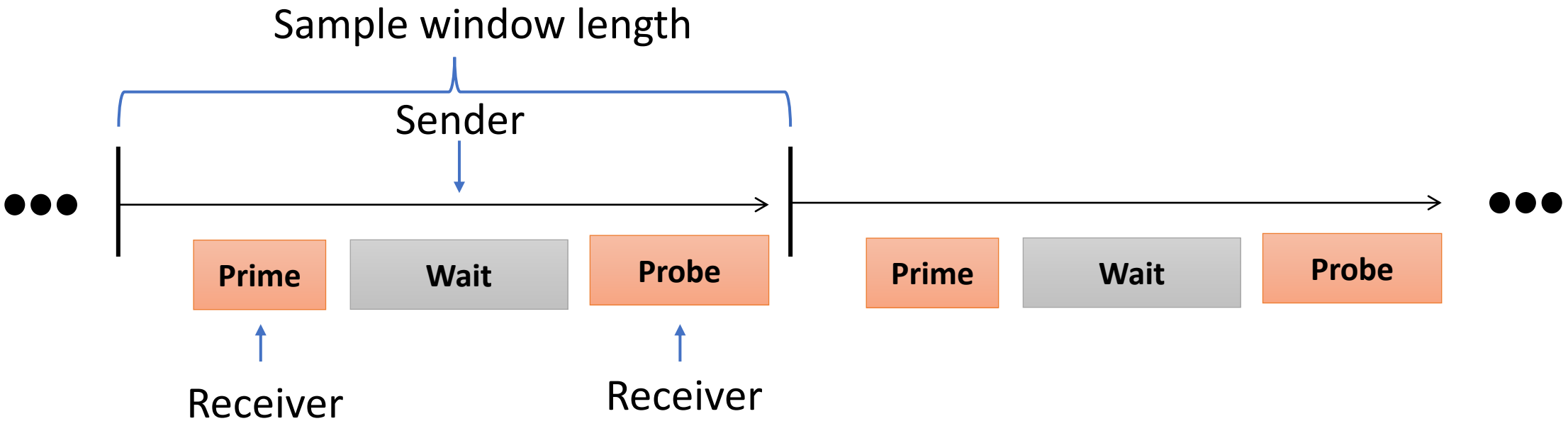
A Complete Protocol -- Synchronization



- Window size agreed on by sender and receiver
- Each window transmits some bits

- Sender & receiver need to perform an window alignment at the start

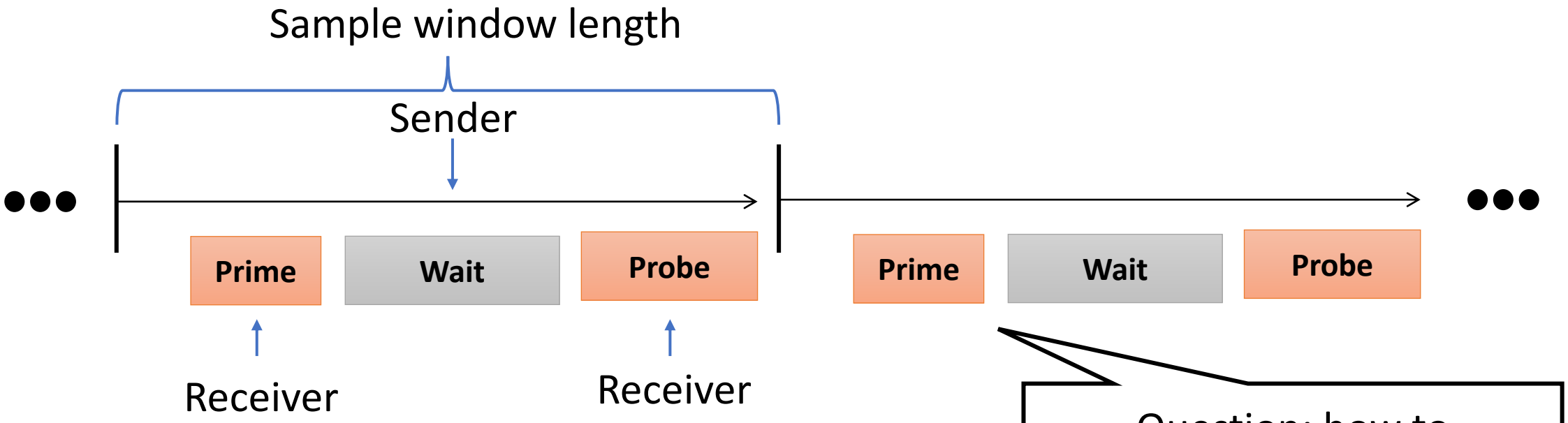
A Complete Protocol -- Synchronization



- Window size agreed on by sender and receiver
- Each window transmits some bits

- Sender & receiver need to perform an window alignment at the start

A Complete Protocol -- Synchronization

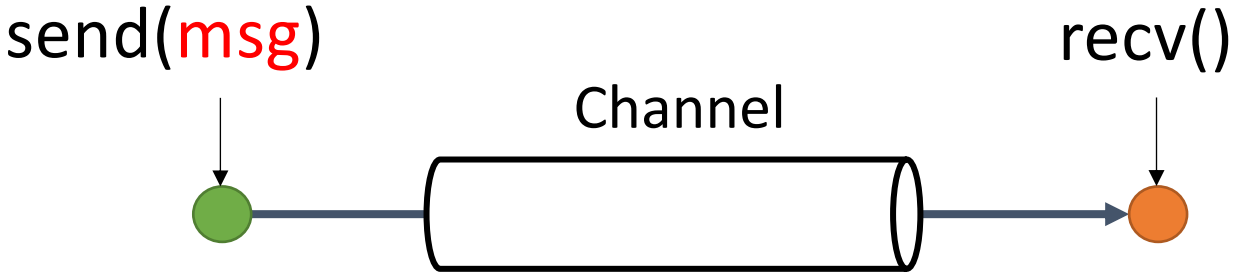


- Window size agreed on by sender and receiver
- Each window transmits some bits

- Sender & receiver need to perform an window alignment at the start

Bandwidth

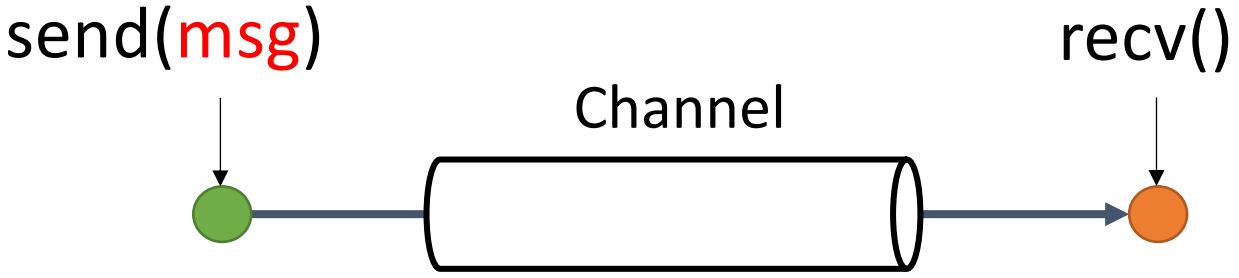
Error-free bitrate of send() → recv()



Depends on what hardware structure is used to build the channel.

Bandwidth

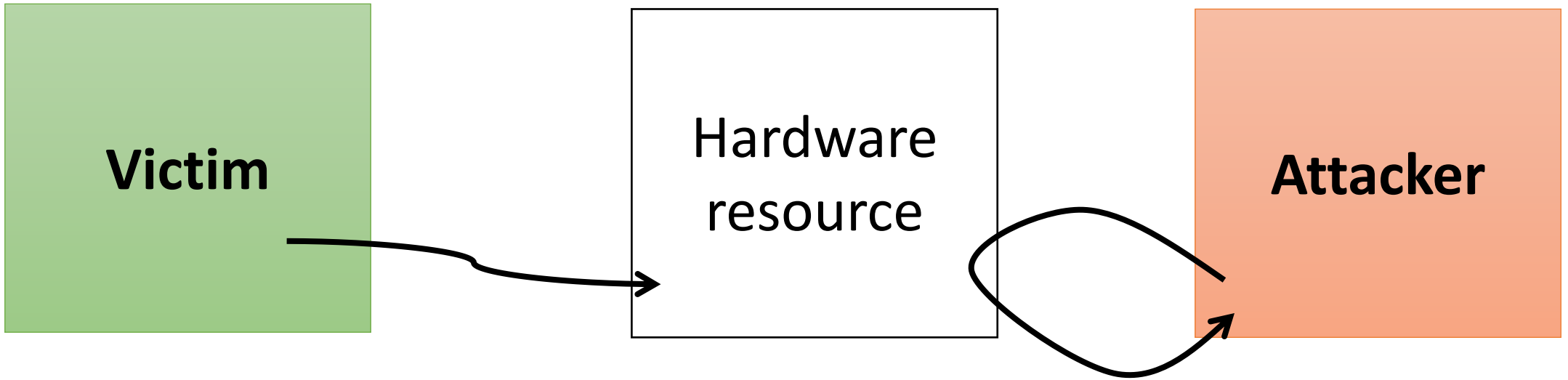
Error-free bitrate of send() → recv()



Depends on what hardware structure is used to build the channel.

- RDRAND unit: 7-200 Kbps [EP'16]
- Ld/st performance counters: ~75-150 Kbps [HKRVDT'15]
- MemBus/AES-NI contention: ~550-650 Kbps [HKRVDT'15]
- LLC: 1.2 Mbps [MNHF'15]
- Various structures on GPGPU: up to 4 Mbps [NKG'17]

From Covert → Side Channels



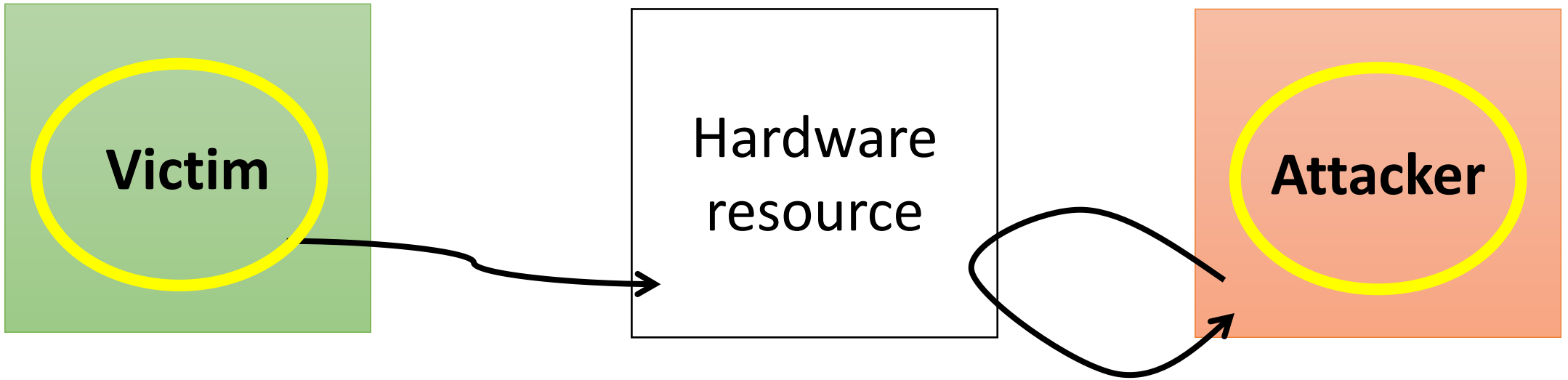
Covert channel:

if (**send '1'**)
  Use resource
else
  idle

t1 = rdtsc()
Use resource
t2 = rdtsc()

if (t2 – t1 > THRESH)
  read '1'
else
  read '0'

From Covert → Side Channels



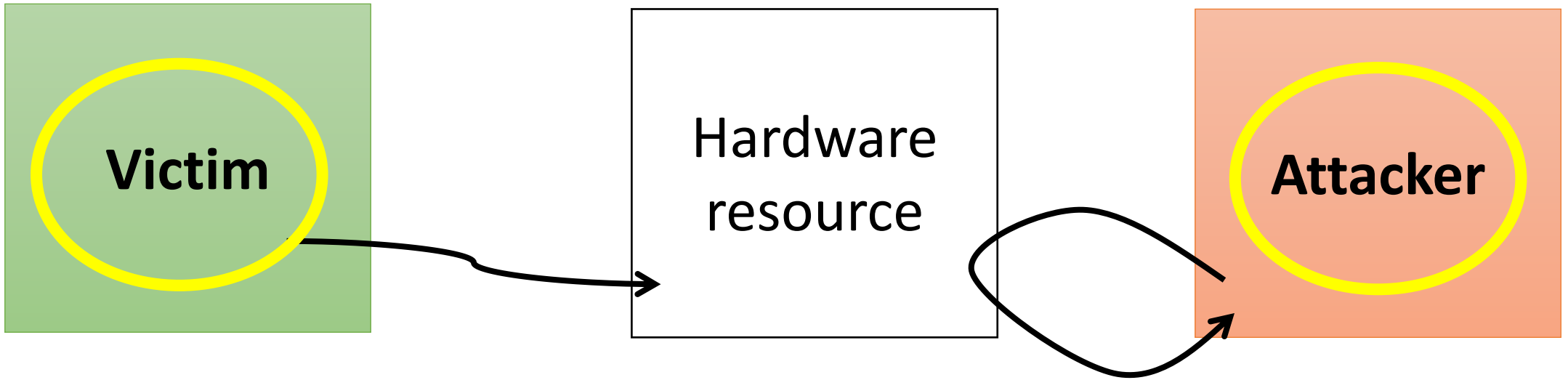
Covert channel:

if (**send '1'**)
  Use resource
else
  idle

t1 = rdtsc()
Use resource
t2 = rdtsc()

if (t2 – t1 > THRESH)
  read '1'
else
  read '0'

From Covert → Side Channels



Covert channel:

if (**send '1'**)
  Use resource
else
  idle

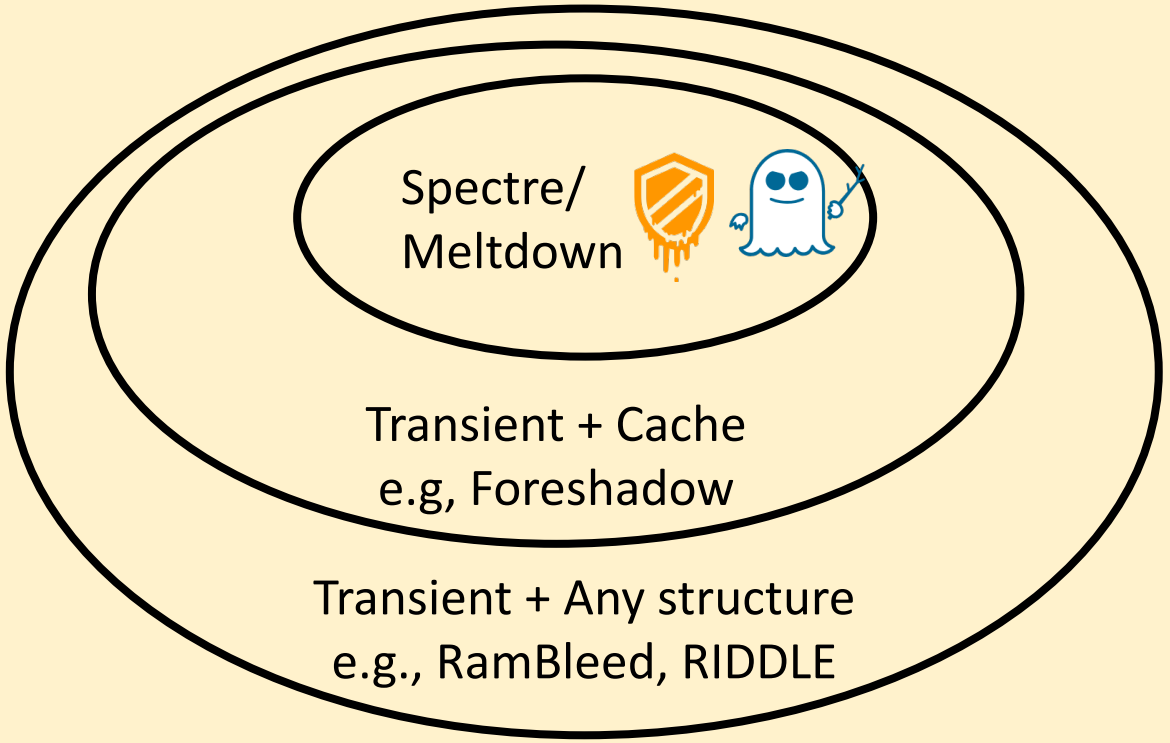
Side channel:

if (**secret**)
  Use resource
else
  idle

t1 = rdtsc()
Use resource
t2 = rdtsc()

if (t2 – t1 > THRESH)
  read '1'
else
  read '0'

uArch Side Channels



Non-transient + Any structure

Micro-architecture Side Channels

Side Channels Targeting Different Structures

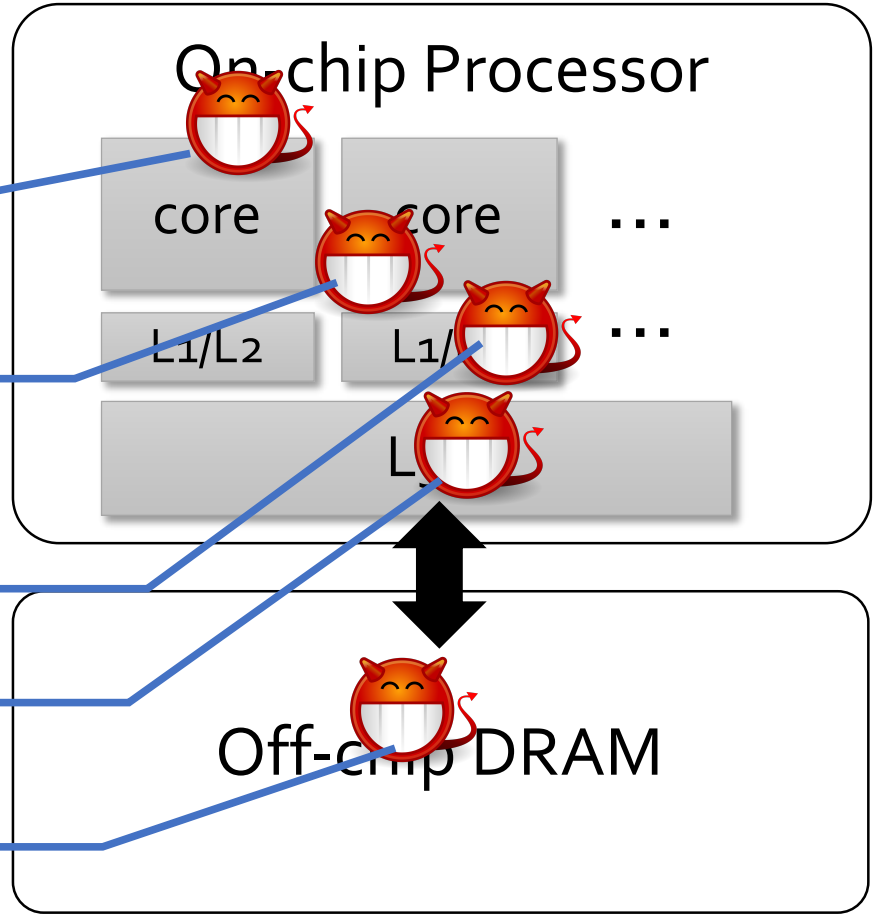
Spectre, Meltdown
Foreshadow,
Arithmetic timing

CacheOut, RIDL, Fallout
Port contention,
cache banking, 4K Alias

RNG Unit Covert Chanel

Flush+Reload, Prime+Probe
Directory Attacks

RowHammer, DRAMA
RAMBleed



Next Lecture:
Non-transient µArch Side Channels

Hard Disk Drive (HDD)