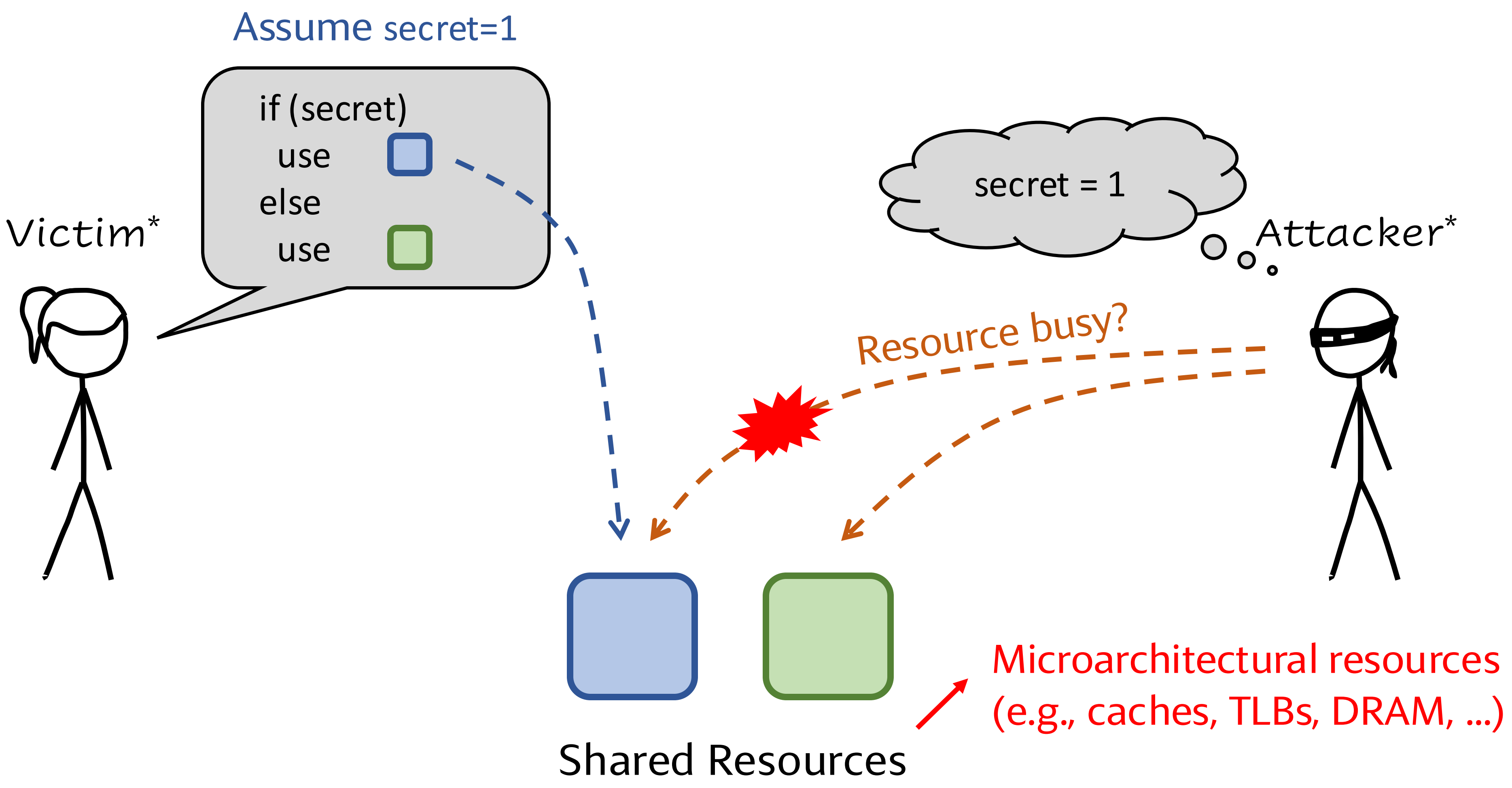
ECE 382N-Sec (FA25):

# L2: Side Channels in Public Clouds

Neil Zhao

neil.zhao@utexas.edu

# Previously on ECE 382N: Sharing Resource => Side Channel

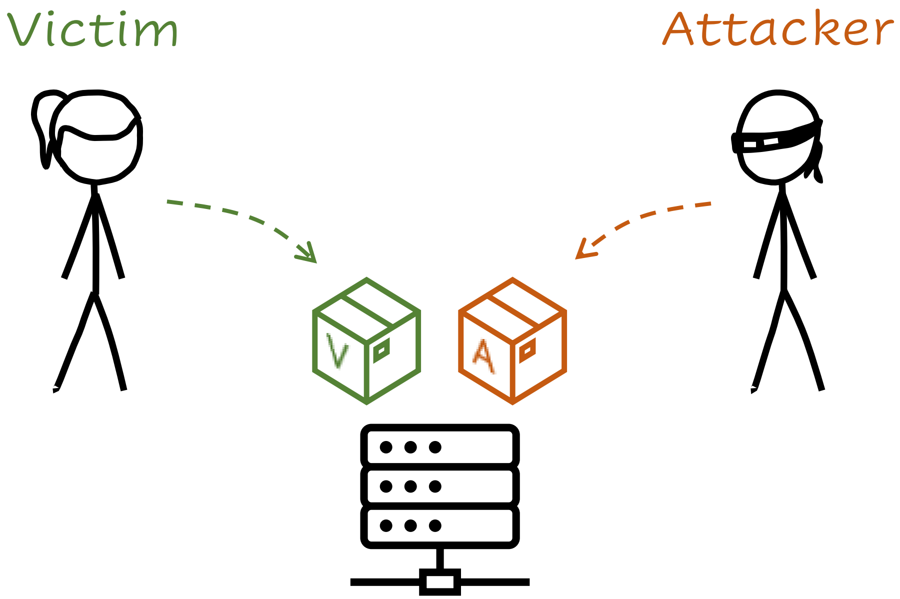


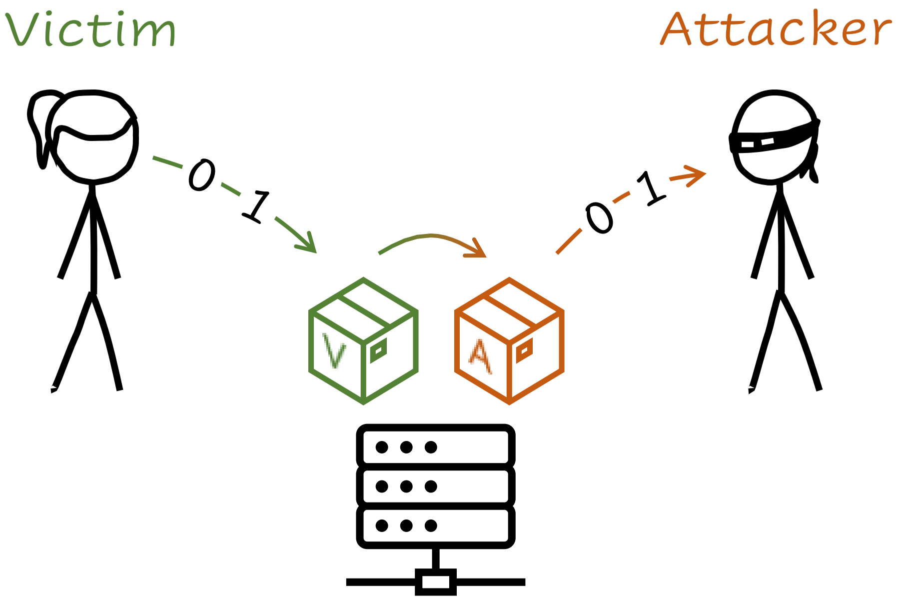

*Characters are based on https://xkcd.com/2176 and https://xkcd.com/1808 (under a CC Attribution-NonCommercial 2.5 License)

# Two Steps of a Side-Channel Attack

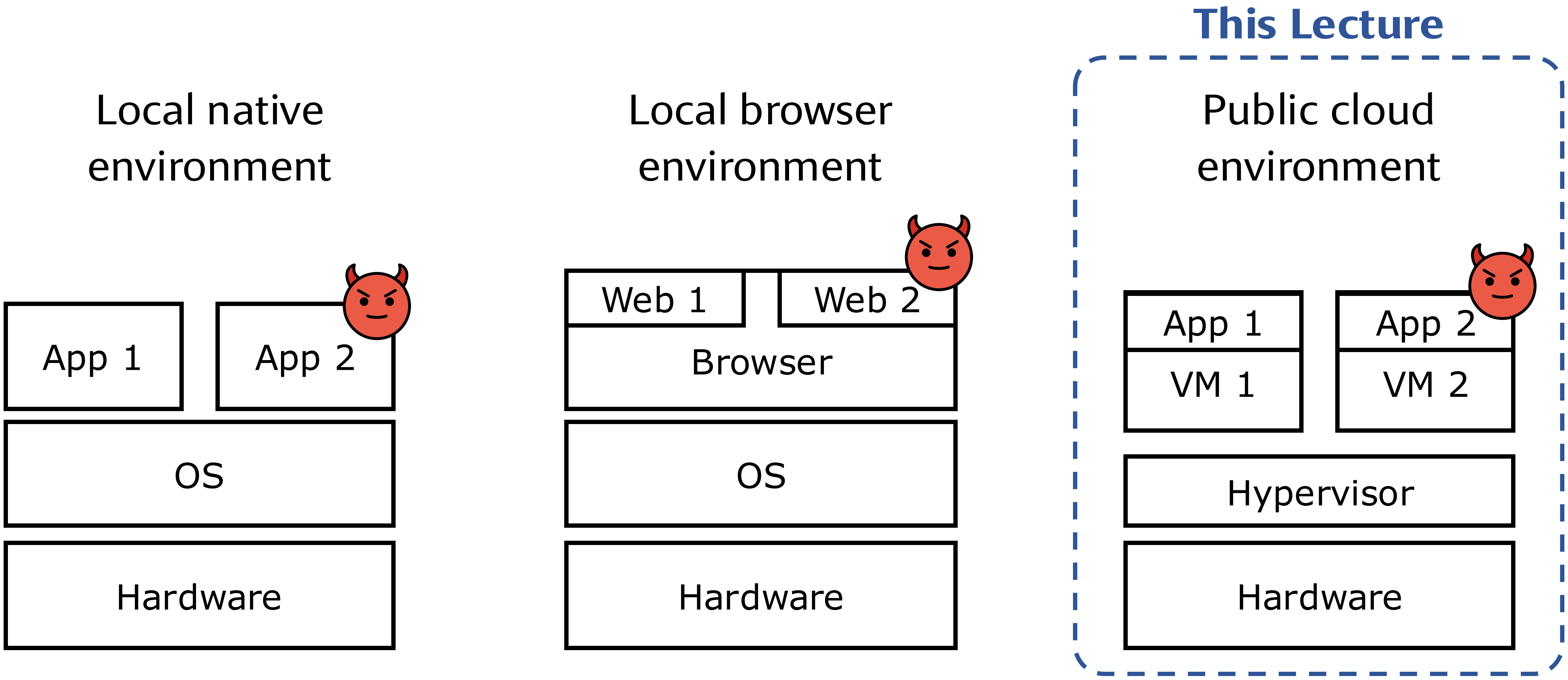
**Step 1:** Co-location

**Step 2:** Extraction



# Three Common Co-location Scenarios



# Why Public Cloud?

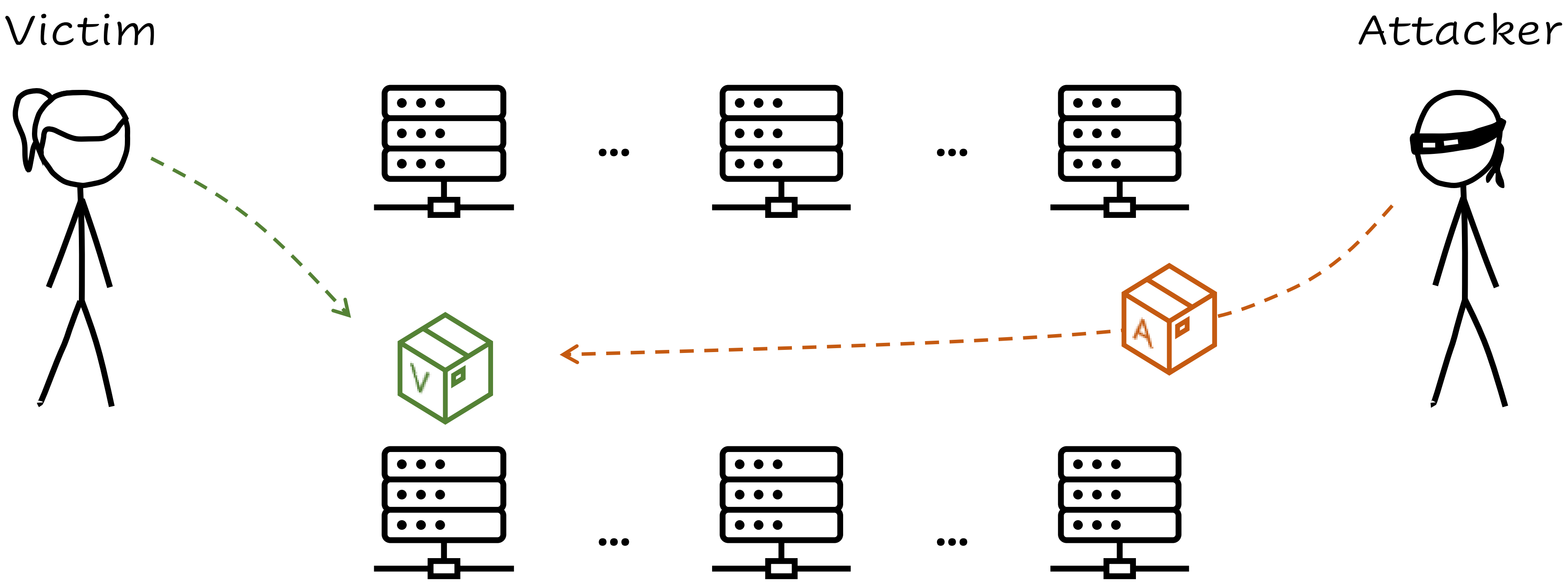
- Public cloud fundamentally relies sharing hardware resources among tenants

- Your VMs are (very likely) always co-located with someone else's VMs
  - Unlike the local setting, the attacker doesn't need to trick you to visit a malicious website or download a malicious app

- Moreover,
  - You cannot control who your "neighbors" are
  - You don't even know who your "neighbors" are

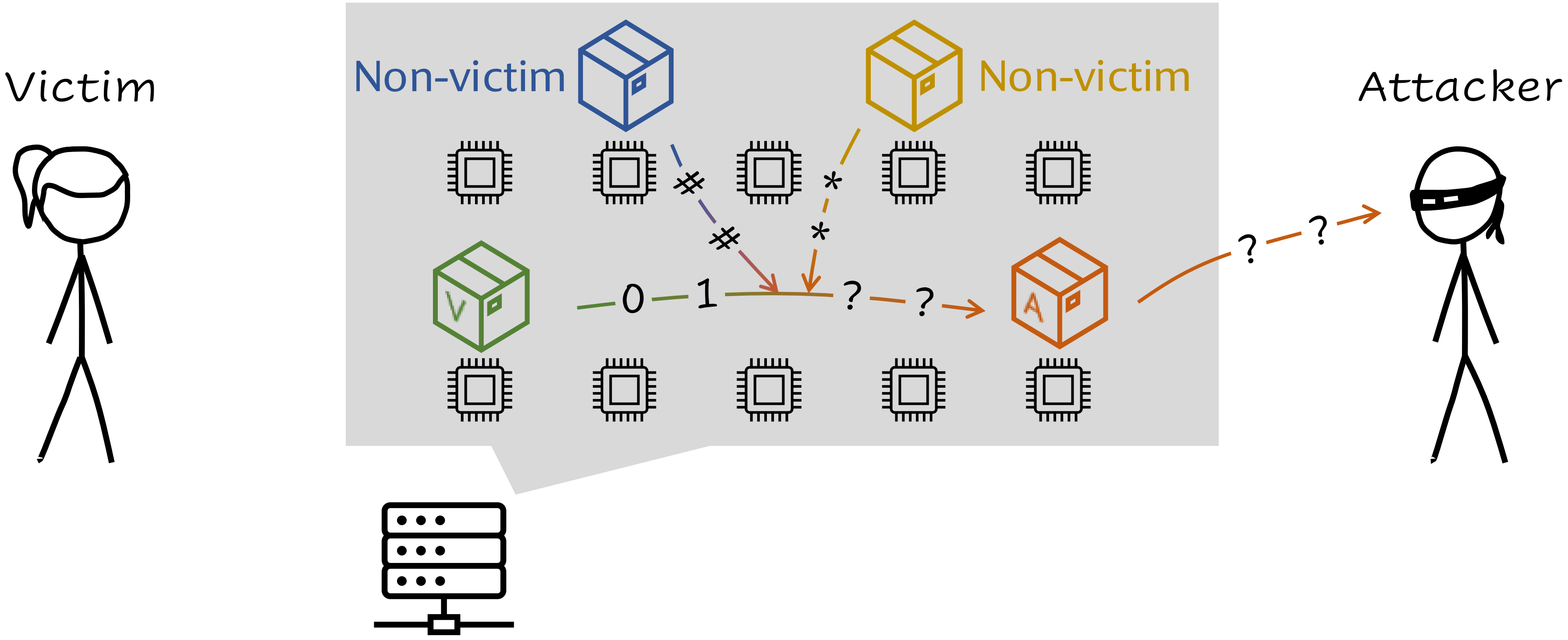
# Why Side Channels in Public Cloud is Hard?

## Challenge 1: Co-location with the victim in a vast datacenter



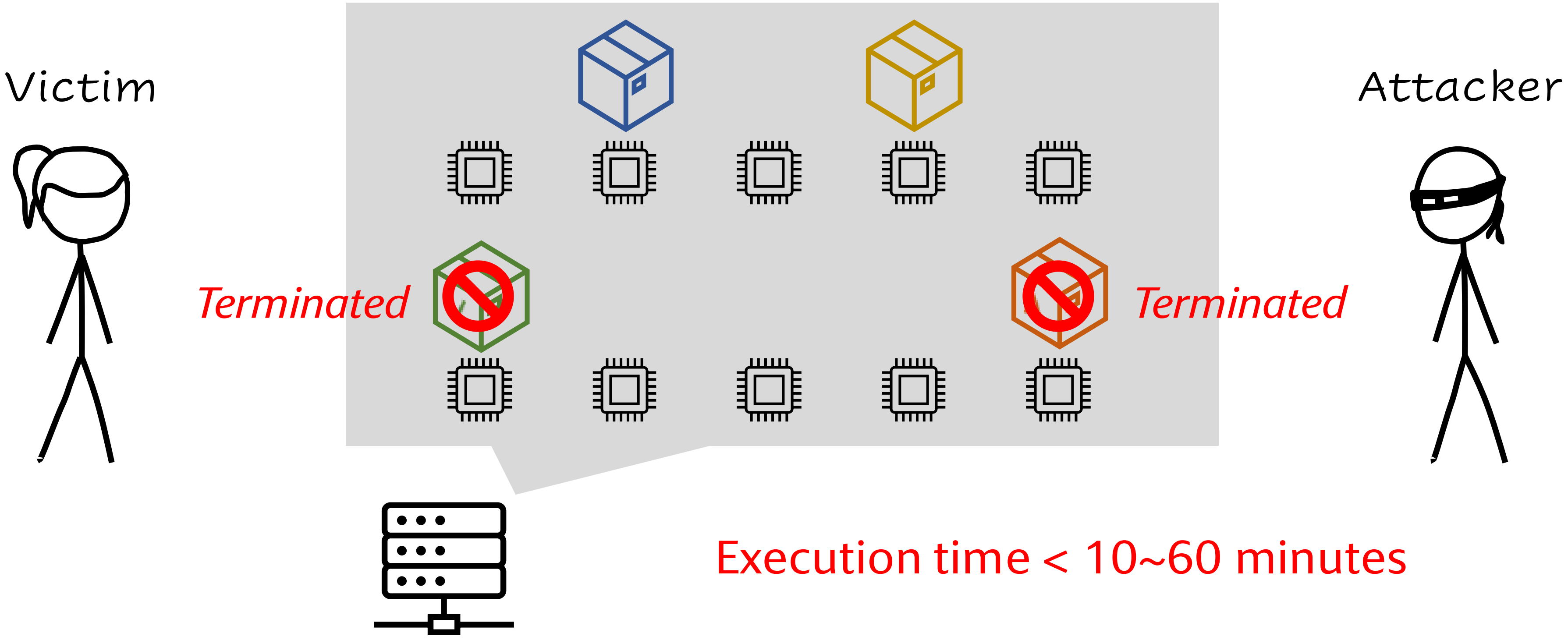
# Why Side Channels in Public Cloud is Hard?

Challenge 2: Production cloud is noisy



# Why Side Channels in Public Cloud is Hard?

Challenge 3: Modern clouds (e.g., FaaS) are dynamic




Execution time < 10~60 minutes

# Cloud Vendors Claim Side-Channel Attacks are Impractical

**Example:** AWS Whitepaper – The Security Design of the AWS Nitro System (Nov. 2022)

aws

*Paraphrased:*

"Last-level cache (LLC) Prime+Probe is impractical due to the noise; therefore, our side-channel mitigations are very strong even if we do not protect VMs against LLC Prime+Probe"

# Let's Start with Co-location

CCS '09

## Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds

Thomas Ristenpart* Eran Tromer† Hovav Shacham* Stefan Savage*

*Dept. of Computer Science and Engineering
University of California, San Diego, USA
{tristenp,hovav,savage}@cs.ucsd.edu

†Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, USA
tromer@csail.mit.edu

3085 citations to date

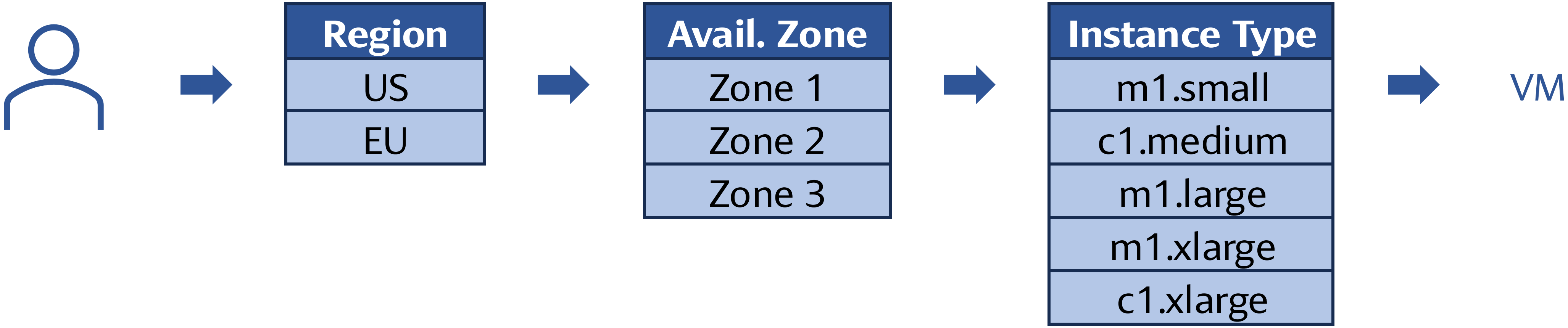
Tips:

- Be the first
- Find a catchy title
- Form a strong team

# Threat Model

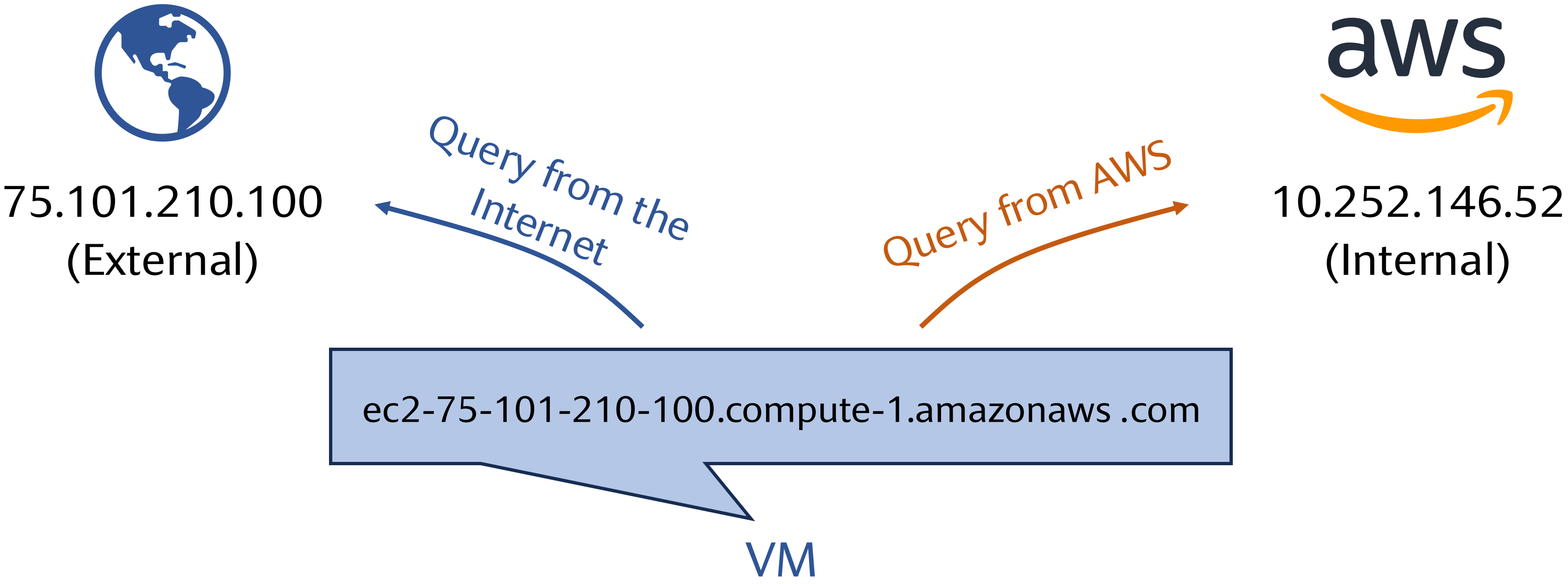
- Attacker-victim co-location is possible (i.e., not sole-tenancy)
- Cloud vendors are trusted
- The attacker is a normal cloud user
  - Has access to typical cloud interfaces available to any customer
  - Can launch many VM instances (20 concurrent VMs)
  - Can run arbitrary code within these VM instances
- The attacker can interact with the victim (e.g., trigger the victim's execution) and has prior knowledge about the victim
- May target a group of users or a specific victim

# Launching an EC2 VM on AWS (in 2009)




These three parameters determine where a VM is placed

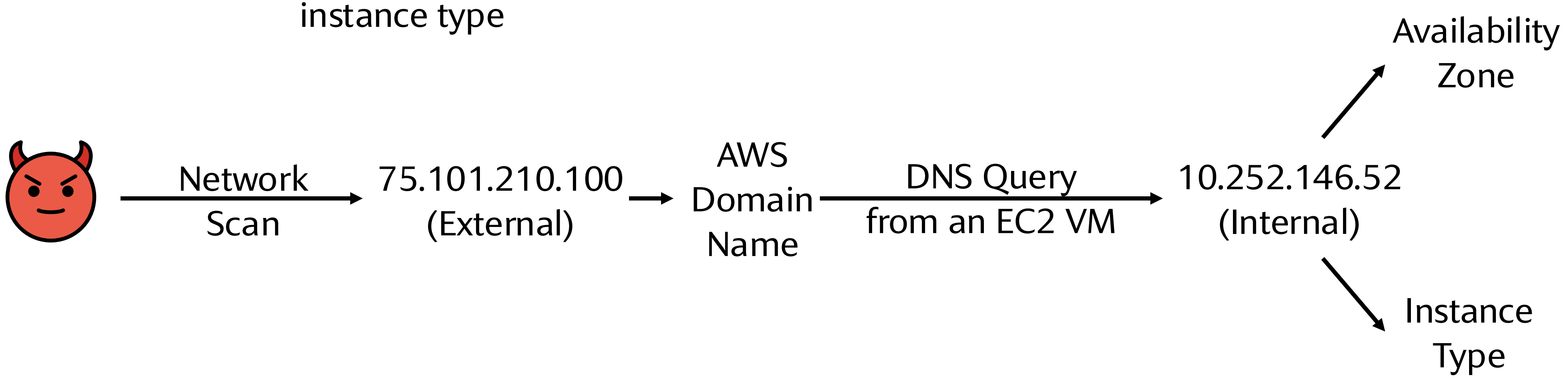
# Networking of VMs in AWS (in 2009)




Hypothesis: a VM's internal IP depends on its where the VM is placed

# Cloud Cartography

- **Method:** Launch VMs with different instance types in different zones, record their internal IPs
- **Overall findings:**
  - Different availability zones use distinct ranges of internal IP addresses
  - An IP /24 prefix (e.g., 10.252.10.*) mostly corresponds to a specific instance type



# Network-Based Co-location Checks

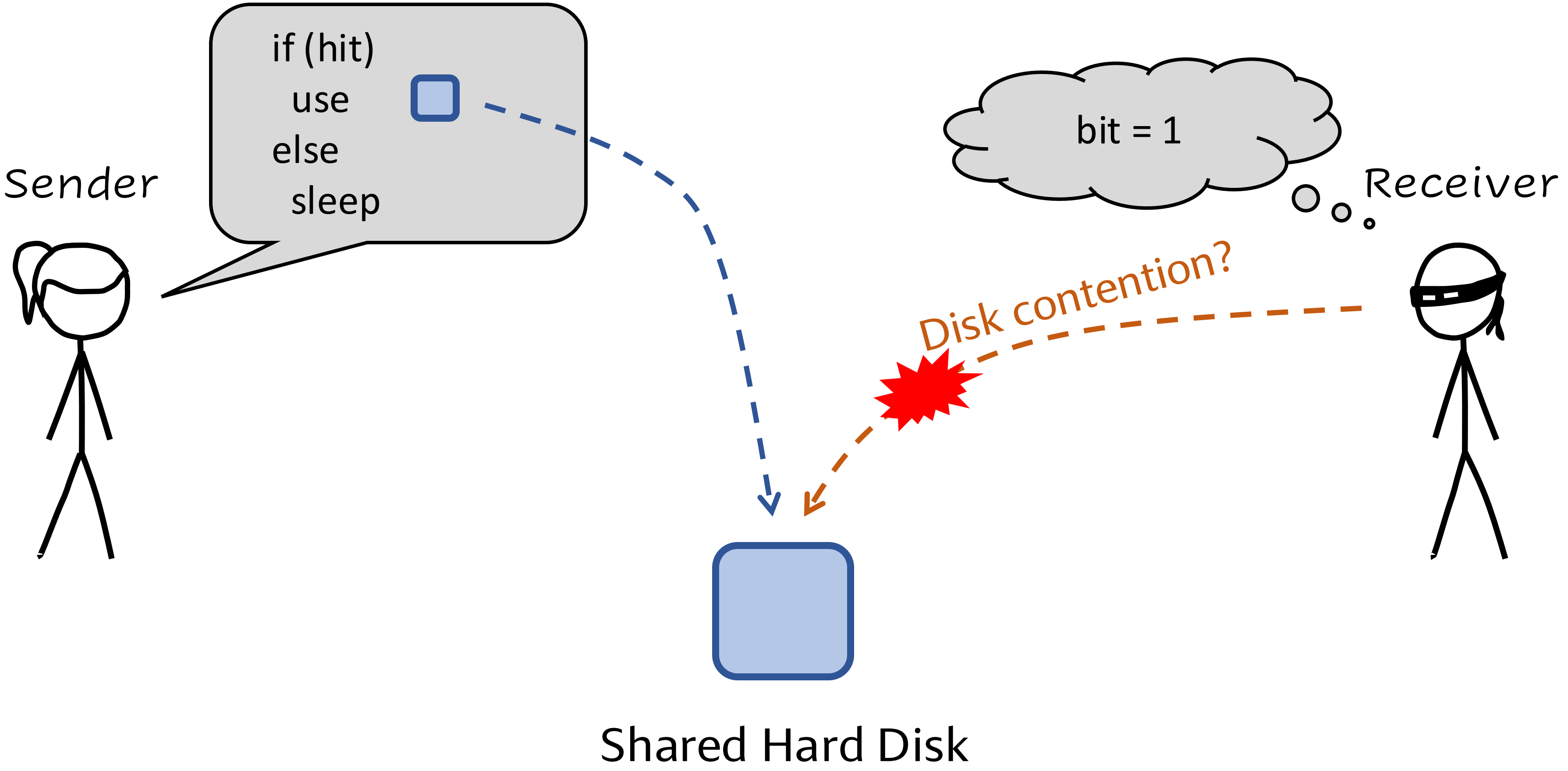
**VMs 1 and 2 are likely co-located if they have:**

- The same IP address of the management Xen VM (i.e., Dom0)
  - By tracerouting from VM1 to VM2
- Small network round-trip time
  - By pinging VM2 from VM1
- Numerically-close internal IP addresses (within 7)
  - By resolving both VMs’ domain names within EC2

Minimum victim VM cooperation. All we need is an open port in victim VM

# Verifying Co-location With a Covert Channel

Send a random 5-bit message



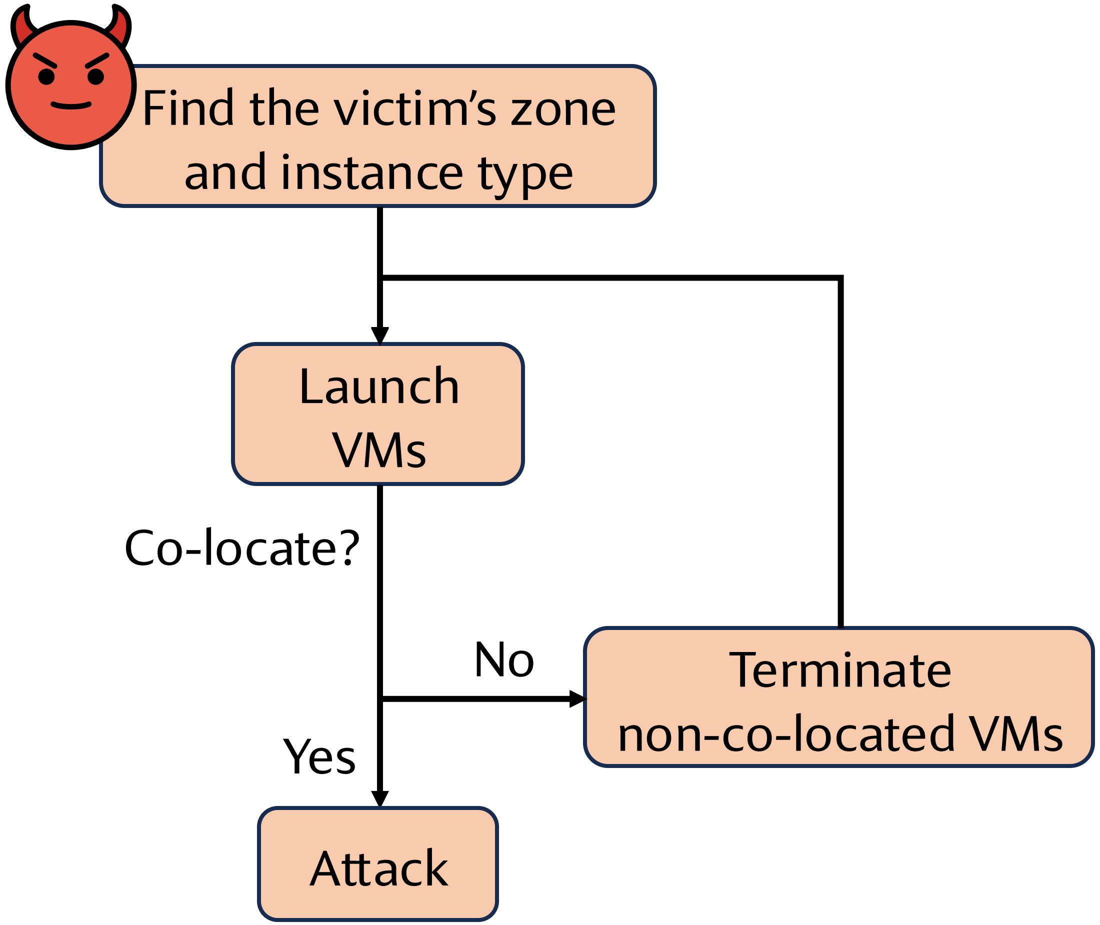
# EC2 VM Placement (in 2009)

## Empirical observations

- VMs from the same account never co-locate
  - Good for reliability
  - Also good for the attacker!

- Placement locality:
  - Sequential locality: A new VM prefers the host of a recently terminated VM
  - Parallel locality: Two VMs launched around the same time prefer the same host

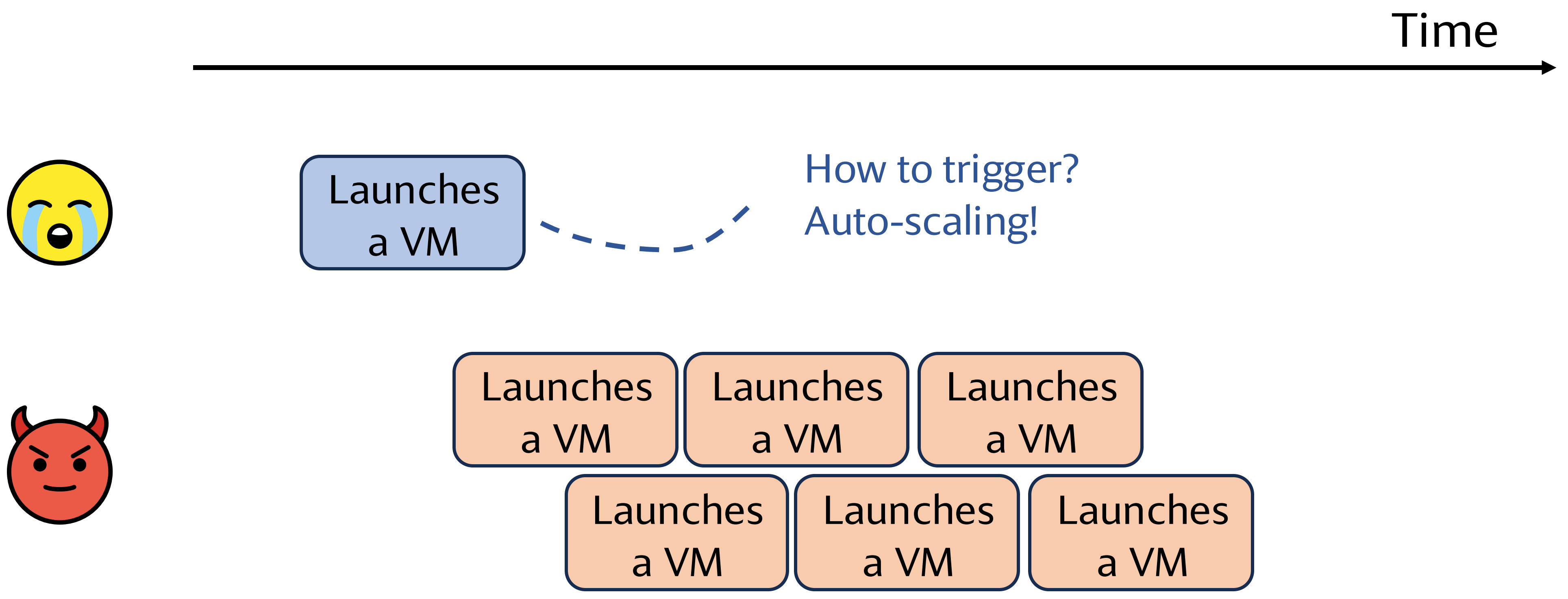
# Brute-Force Placement




8.4% victim coverage
(over 18 days)

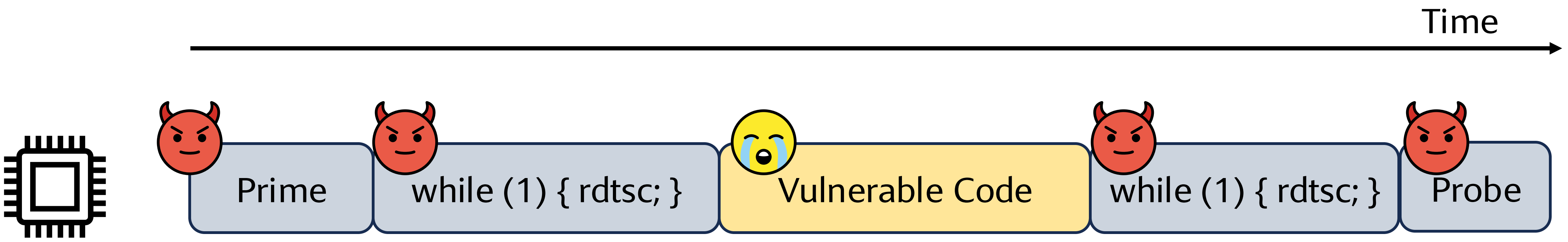
# Abusing Placement Locality

Parallel locality: Two VMs launched around the same time prefer the same host



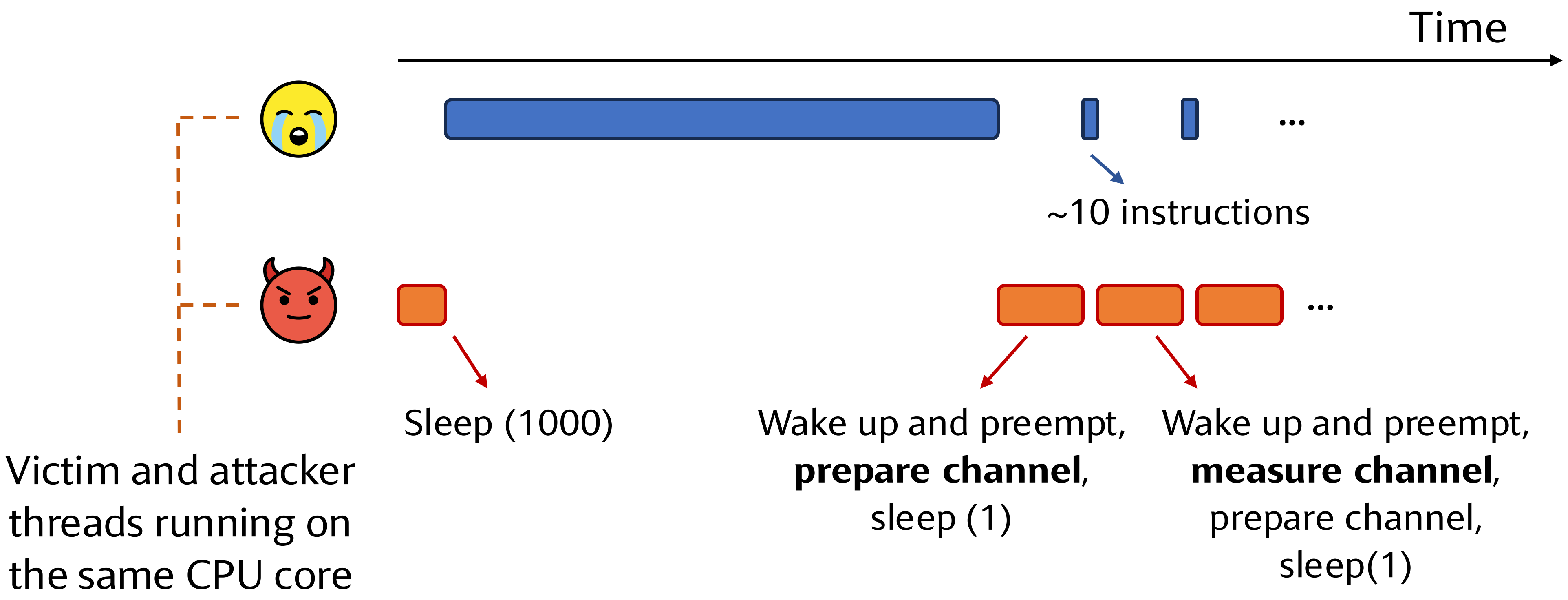

40% of the time the attacker succeeds

# Prime+Trigger+Probe




- Coarse-grained information leakage
- Can detect co-location without network-based techniques
- Can detect keystrokes (in a local environment) with a time resolution of 13ms

# Gaming the OS Thread Scheduler




Focused mostly on Linux Completely Fair Scheduler (CFS)

# Some Follow-Up Work (Around 2015)

**Seriously, get off my cloud!**
**Cross-VM RSA Key Recovery in a Public Cloud**

Mehmet Sinan İnci, Berk Gülmezoğlu, Gorka Irazoqui, Thomas Eisenbarth, Berk Sunar
Worcester Polytechnic Institute, Worcester, MA, USA
Email:{msinci, bgulmezoglu, girazoki, teisenbarth, sunar}@wpi.edu

Demonstrated LLC Prime+Probe on AWS

# Some Follow-Up Work (Around 2015)

## A Placement Vulnerability Study in Multi-Tenant Public Clouds

Venkatanathan Varadarajan, *University of Wisconsin—Madison;* Yinqian Zhang, *The Ohio State University;* Thomas Ristenpart, *Cornell Tech;* Michael Swift, *University of Wisconsin—Madison*

https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/varadarajan

High-speed covert channel -> Fast co-location detection

## A Measurement Study on Co-residence Threat inside the Cloud

Zhang Xu, *College of William and Mary;* Haining Wang, *University of Delaware;* Zhenyu Wu, *NEC Laboratories America*

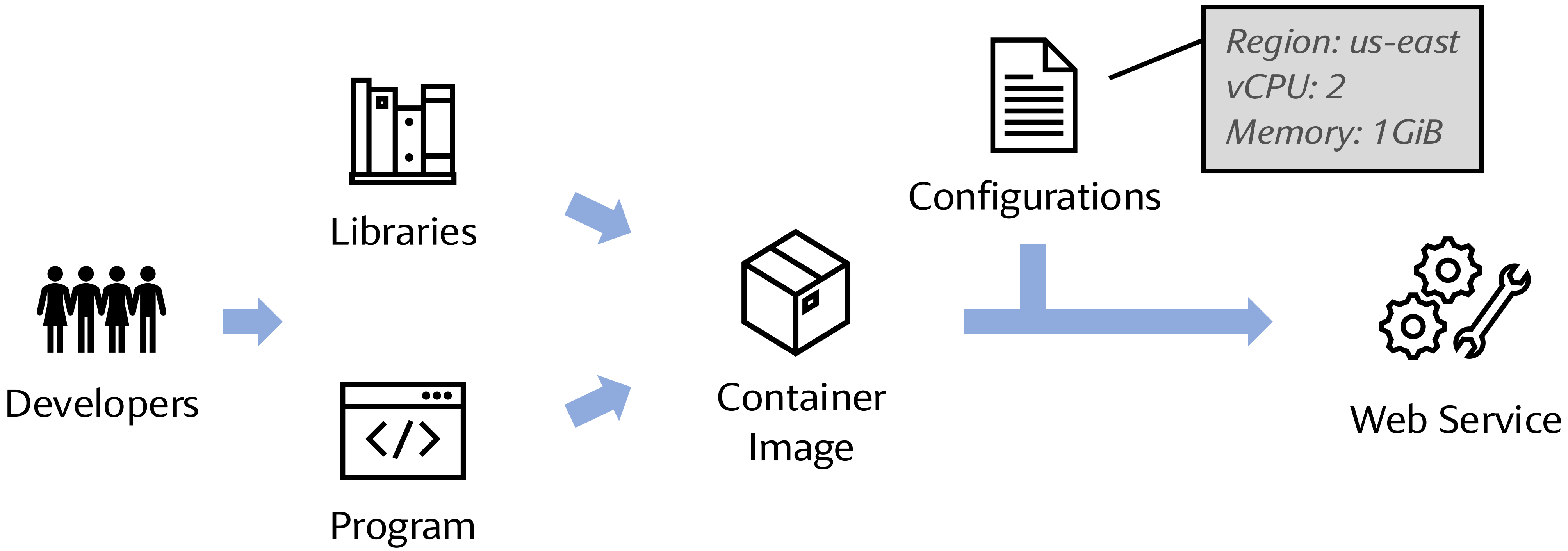
https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/xu

Revisited network-based co-location tests

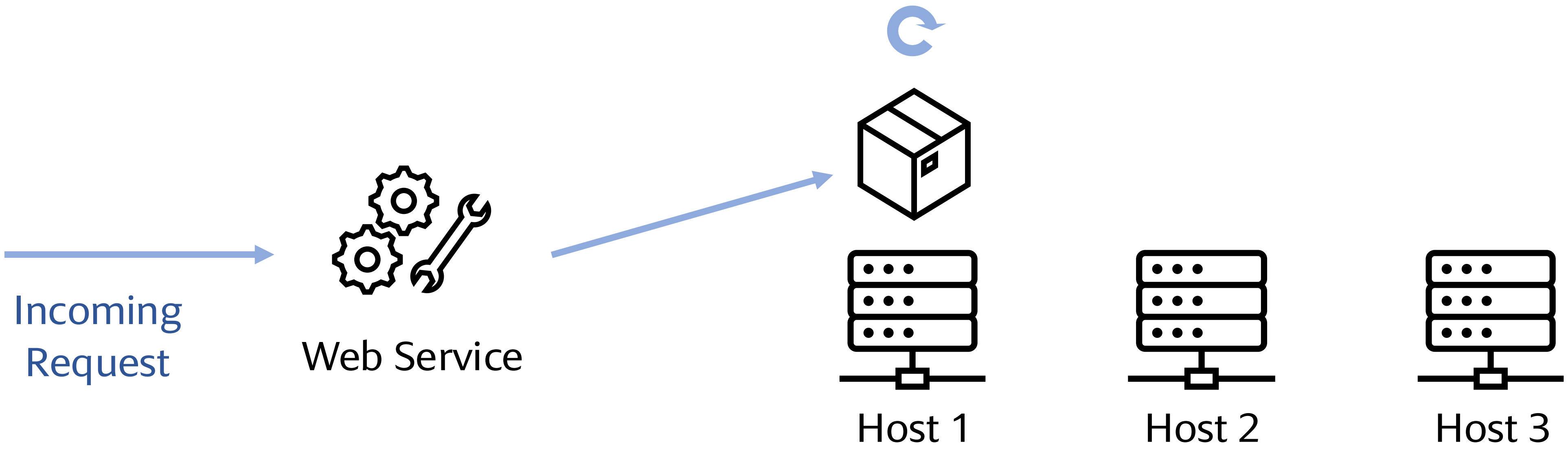
# What Changed from 2009 and 2015

- Virtual private cloud => network isolation between tenants, cloud vendors hide traceroute information
- More servers => Harder to co-located
- More powerful hardware => More tenants on the same node and thus more noise
- The widespread of non-inclusive LLC
- Container + Function-as-a-Service (FaaS)
  - Short lifetime
  - High background noise

# Background: Fully-Managed Containerized Environment



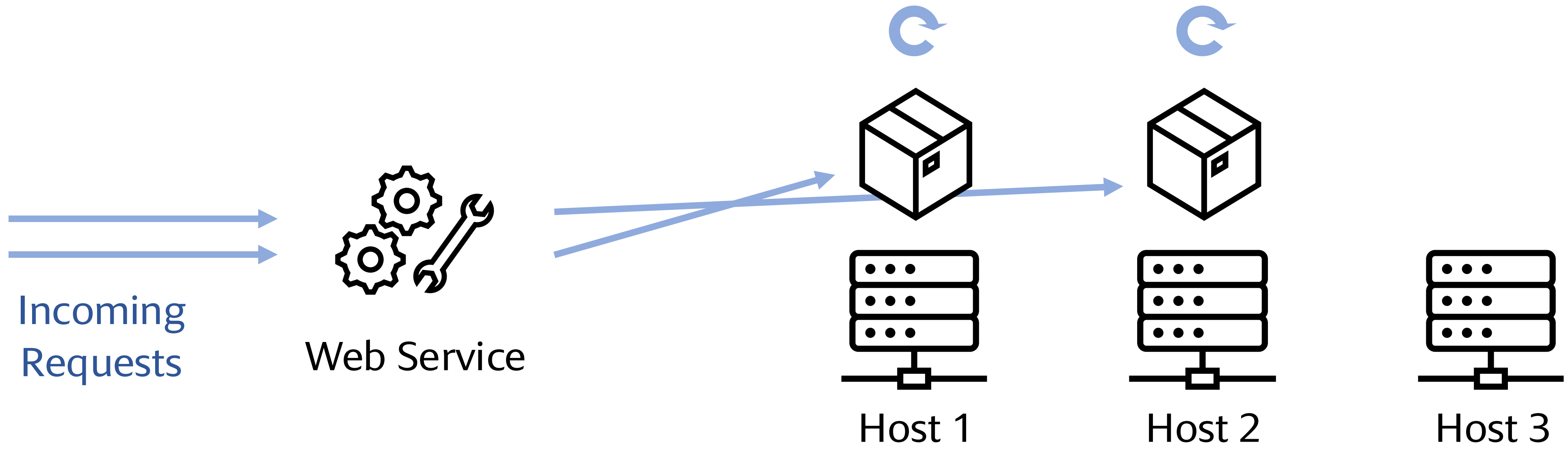
# Background: Fully-Managed Containerized Environment




Cloud vendor automatically launches a container instance

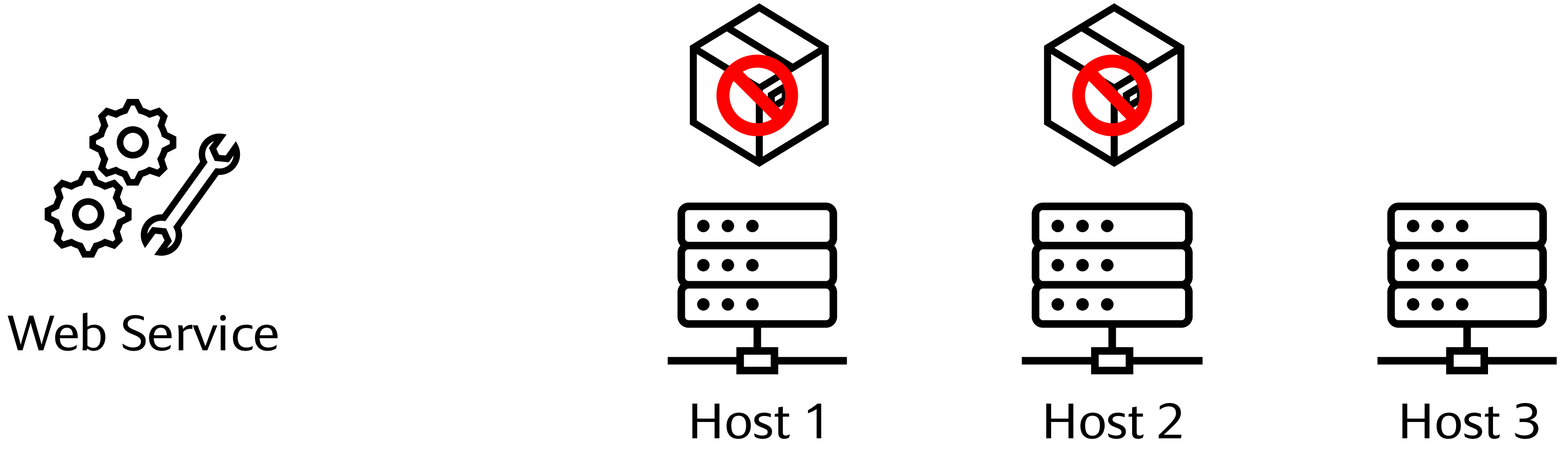
(The instance placement is managed by the vendor)

# Background: Fully-Managed Containerized Environment



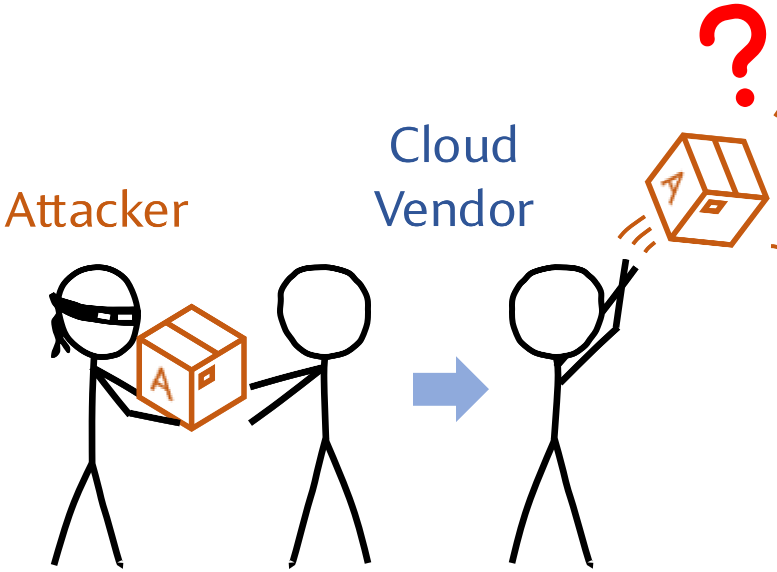

Cloud vendor launches more instances to handle traffic increases

# Background: Fully-Managed Containerized Environment



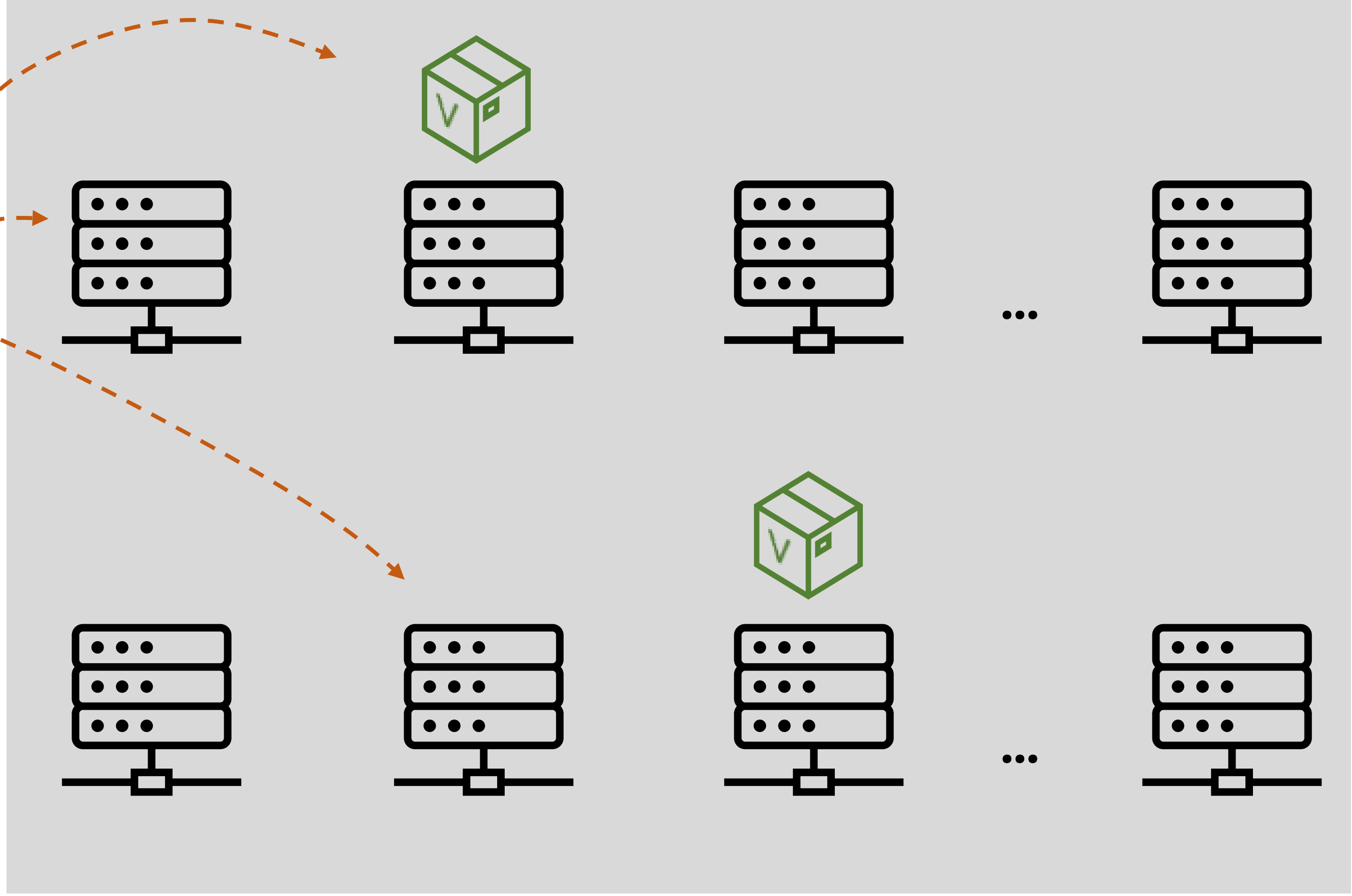

Cloud vendor automatically terminates idle instances

# Fog of War: Container Instance Placement



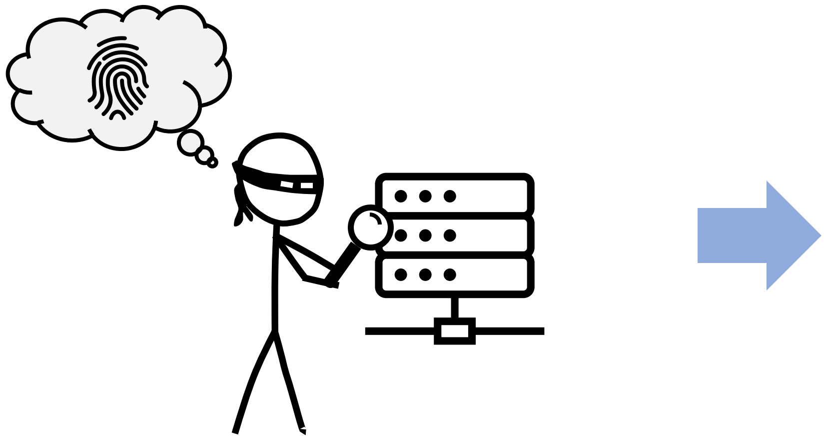

Cloud vendors use **unknown** instance placement policies

⇒ Naively launching instances often has zero co-location



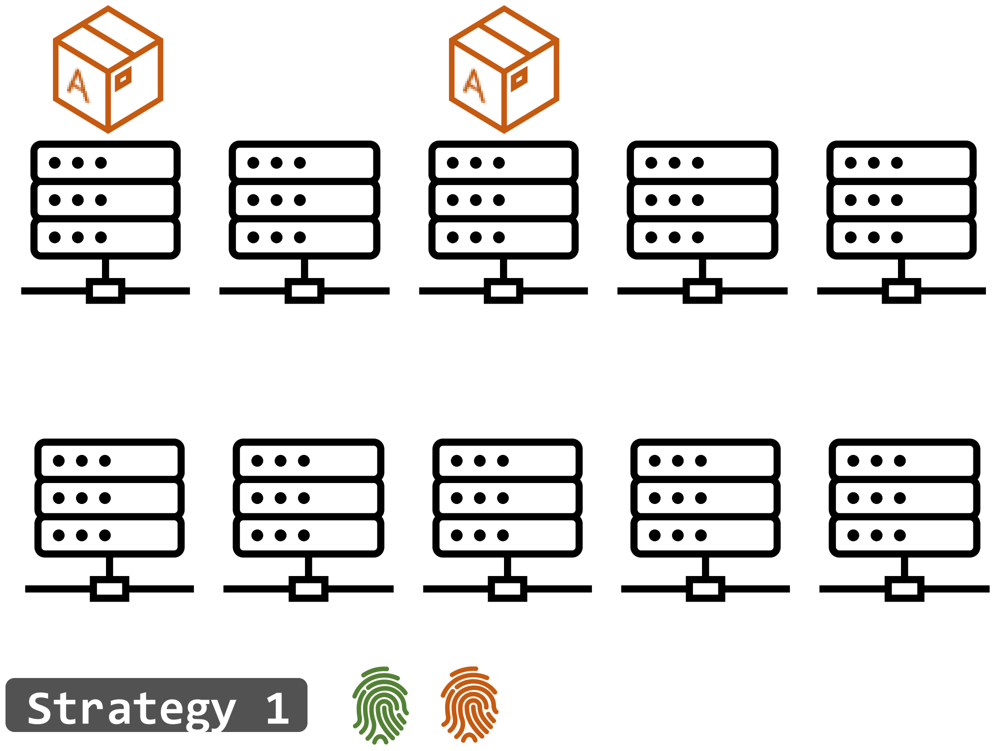

Data Center

# Idea: Fingerprint Host → Reverse Engineer Placement Behavior

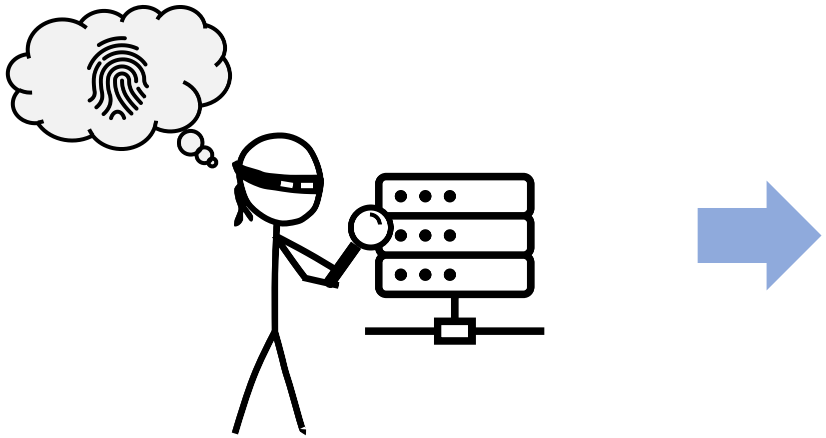


Accurate host fingerprinting

Understand container placement

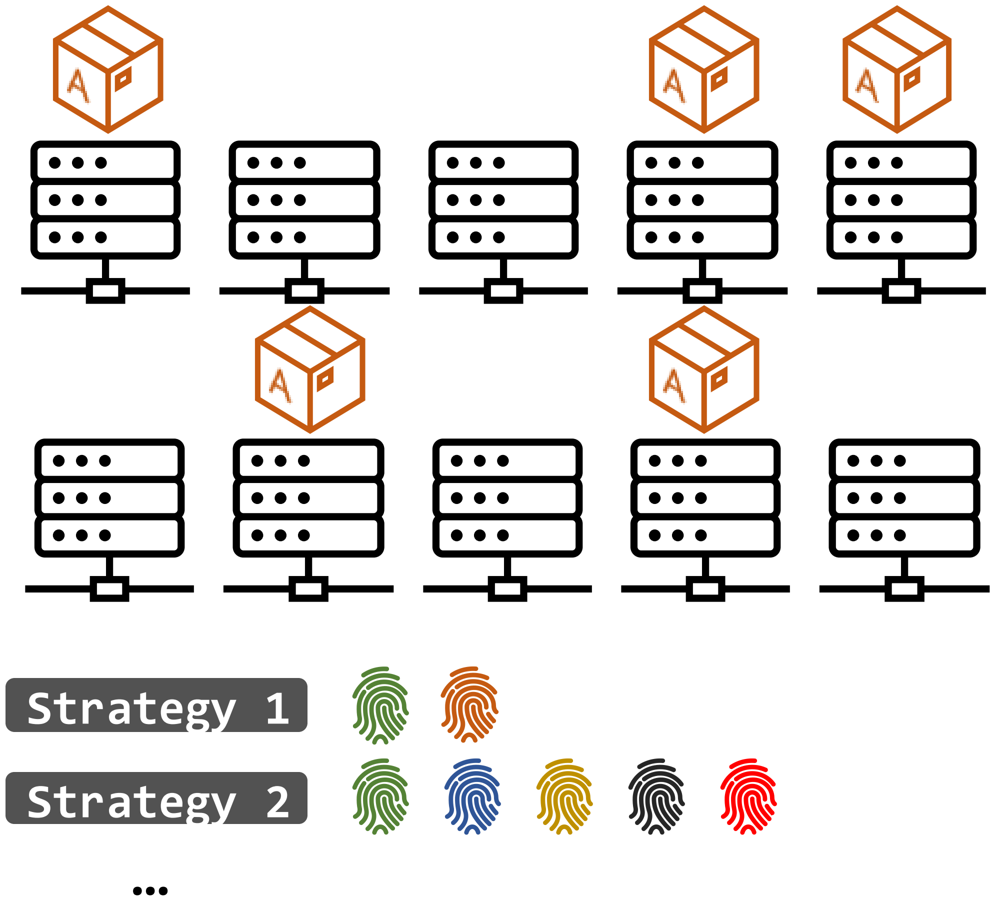


# Idea: Fingerprint Host → Reverse Engineer Placement Behavior

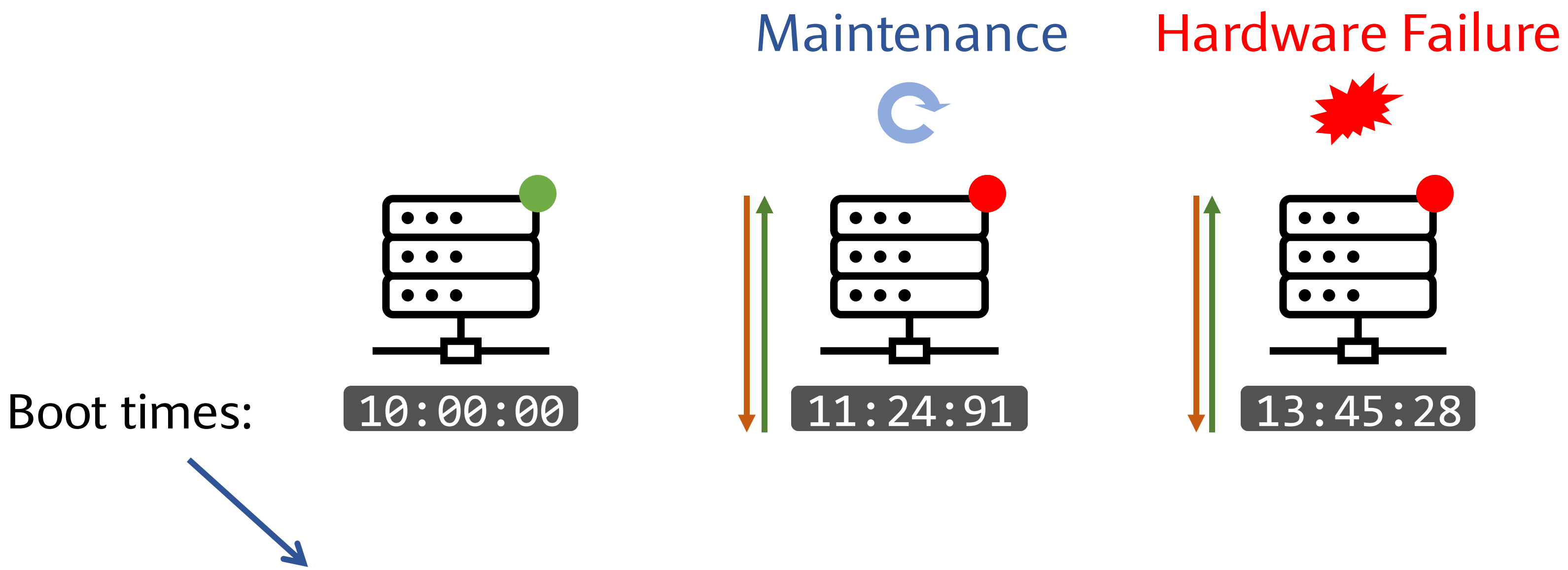


Accurate host fingerprinting

Understand container placement

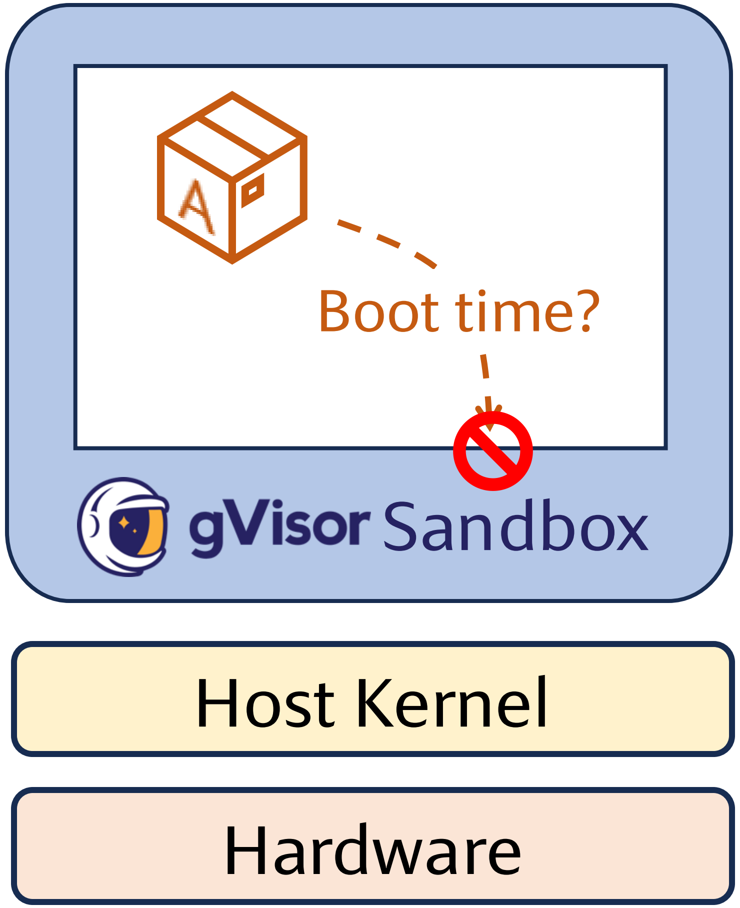


# Insight 1: Physical Host's Boot Time as Fingerprint




Likely unique within a data center

# Challenge: Host Information is Hidden Due to Sandboxing

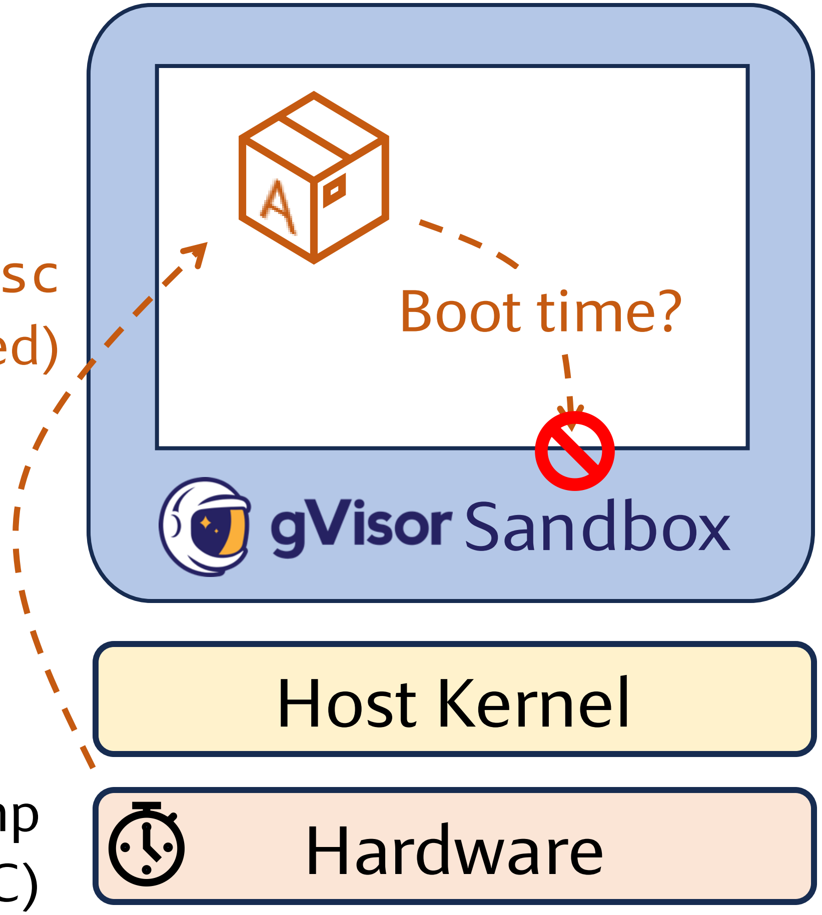


# Insight 2: Bypassing Software Protection by Asking the Hardware

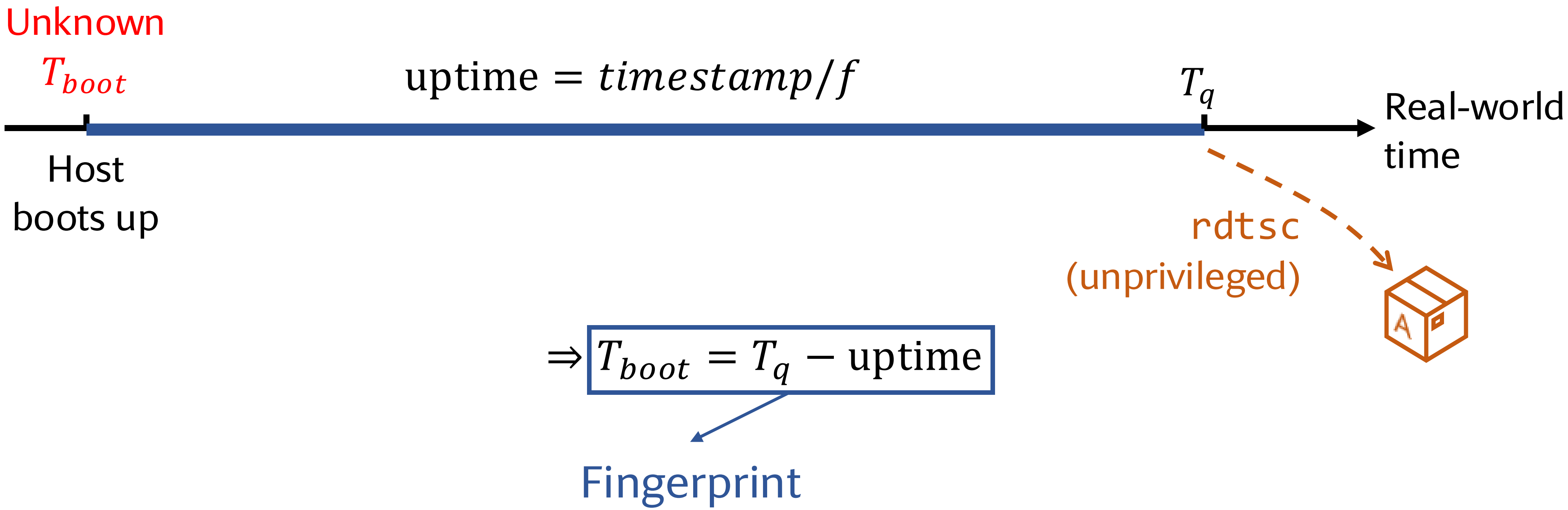
**Useful properties:**
1. Reset to **0** when the physical host boots
2. Increments at a **known** fixed rate $f$

Not affected by CPU frequency scaling and power state

x86 timestamp counter (TSC)

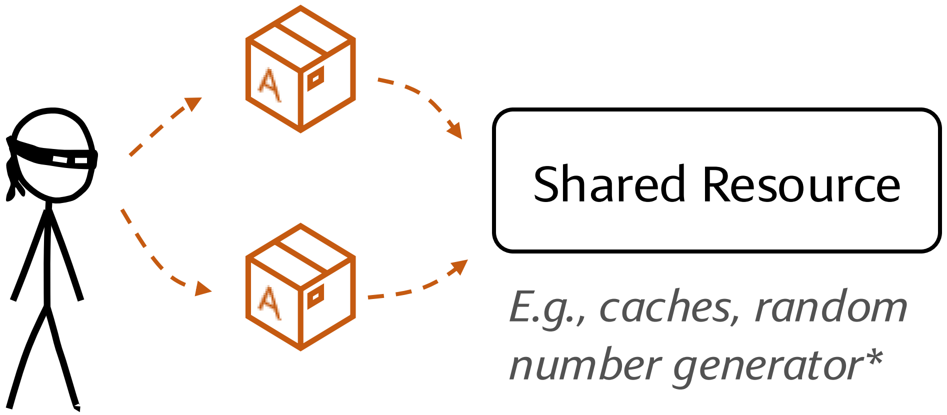


# Derive Boot Time From Timestamp Counter

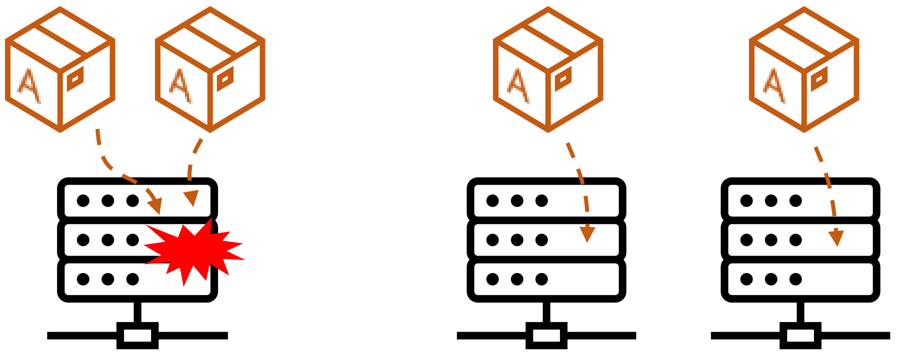


# Verifying Co-Location

Instance Pair




*E.g., caches, random number generator**



Contention!
⇒ Co-Located

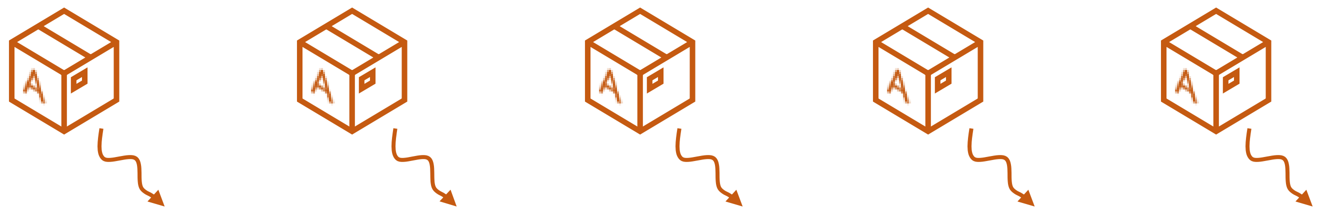
No Contention
⇒ Not Co-Located

**Scalability issue:** it requires $O(N^2)$ pairwise tests to verify $N$ containers

* Evtyushkin et al., Covert Channels through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations (CCS ’16)

# Go Beyond Pairwise Testing – Batch Testing

$A_1$ $A_2$ $A_3$ $A_4$ $A_5$

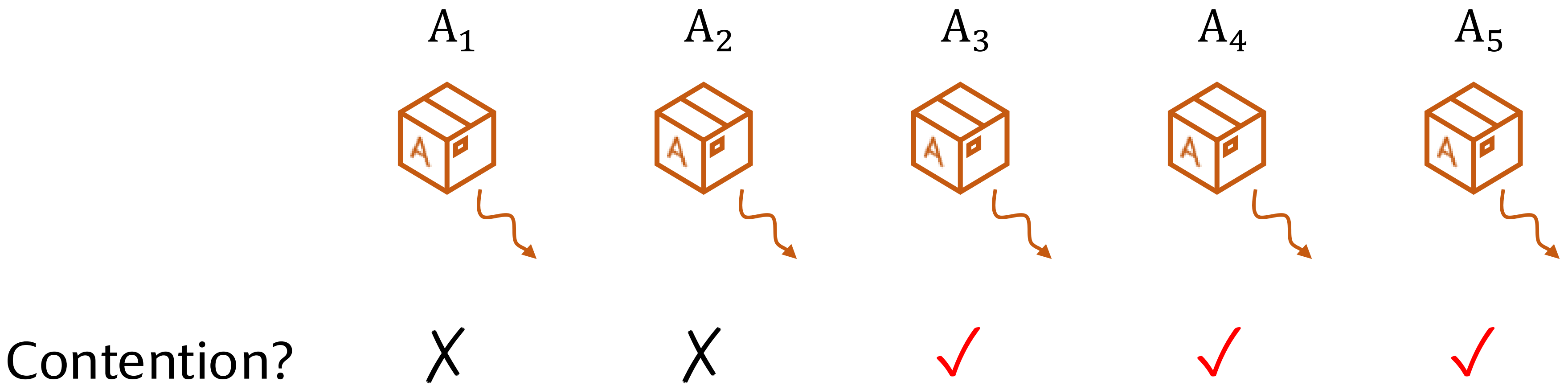


Contention? ✗ ✗ ✗ ✓ ✓

$A_1, A_2, A_3$ do not co-locate with any other instance
(i.e., they are *single instances*)

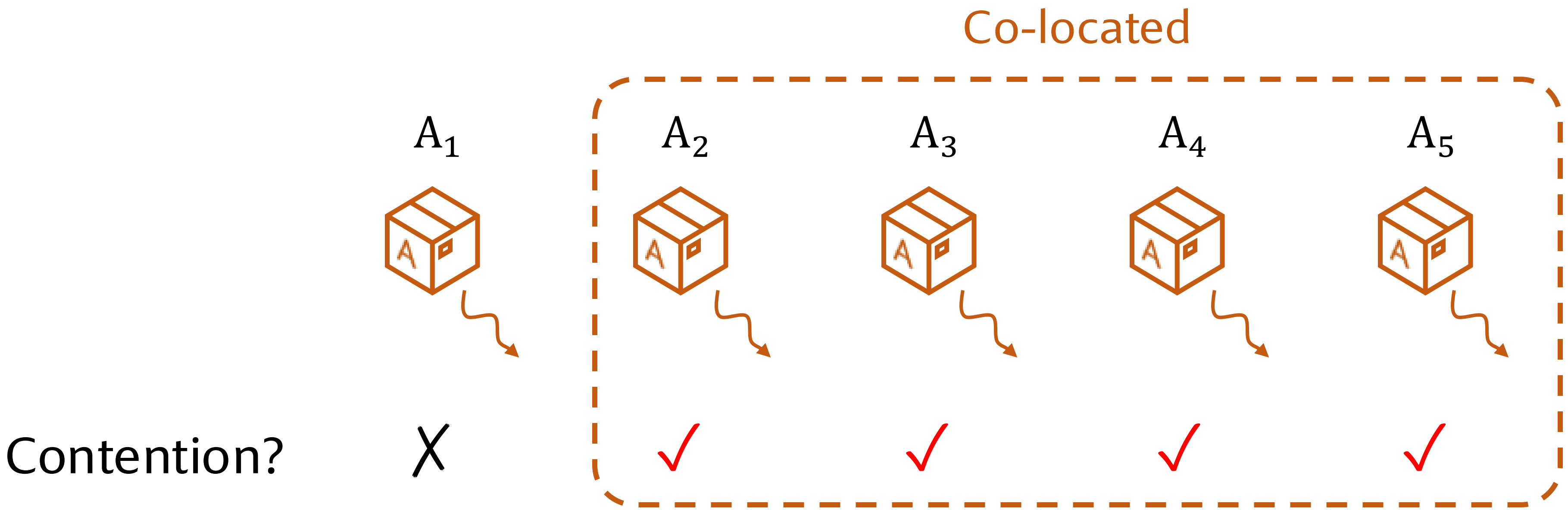
$A_4$ and $A_5$ are co-located

# Go Beyond Pairwise Testing – Batch Testing



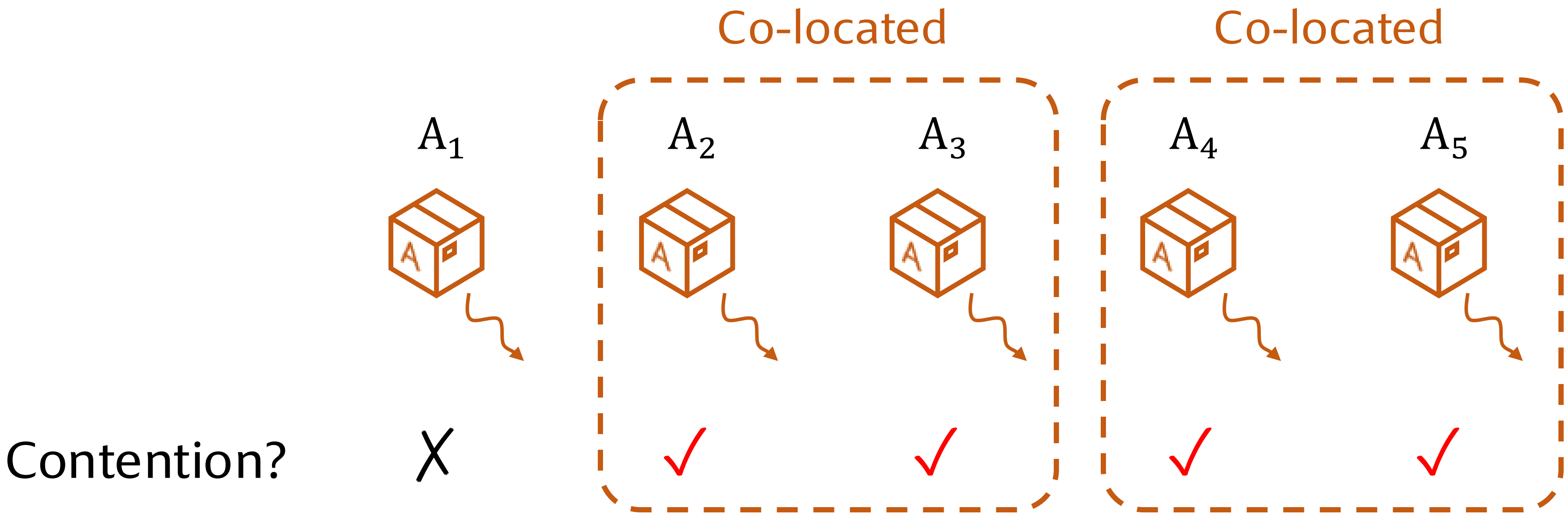

$A_3, A_4, A_5$ are co-located

# Go Beyond Pairwise Testing – Batch Testing




$A_2, A_3, A_4, A_5$ are co-located? Not sure!

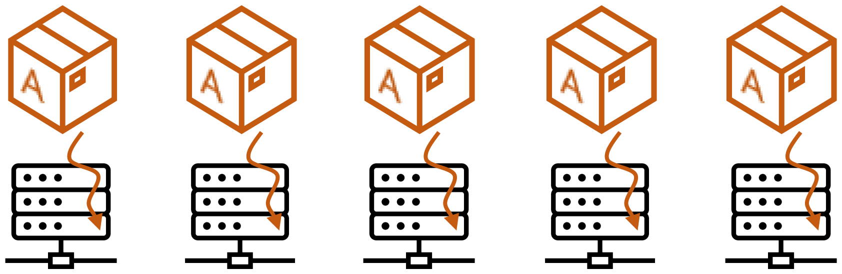
# Go Beyond Pairwise Testing – Batch Testing




$A_2, A_3, A_4, A_5$ are co-located? Not sure!

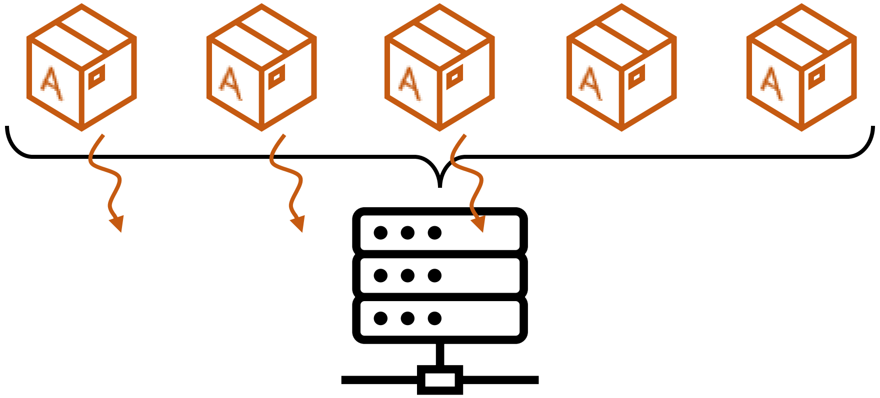
# Batch Testing Strategies

Truly not co-located



Batch test all instances at once

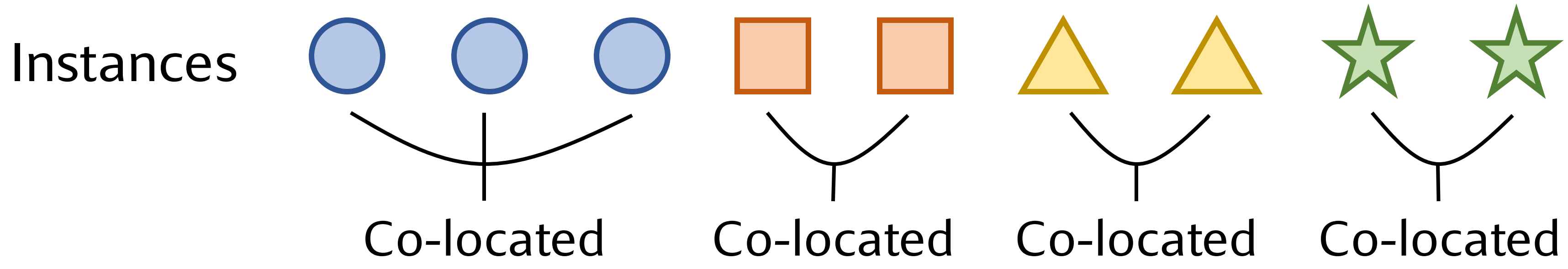
Truly co-located



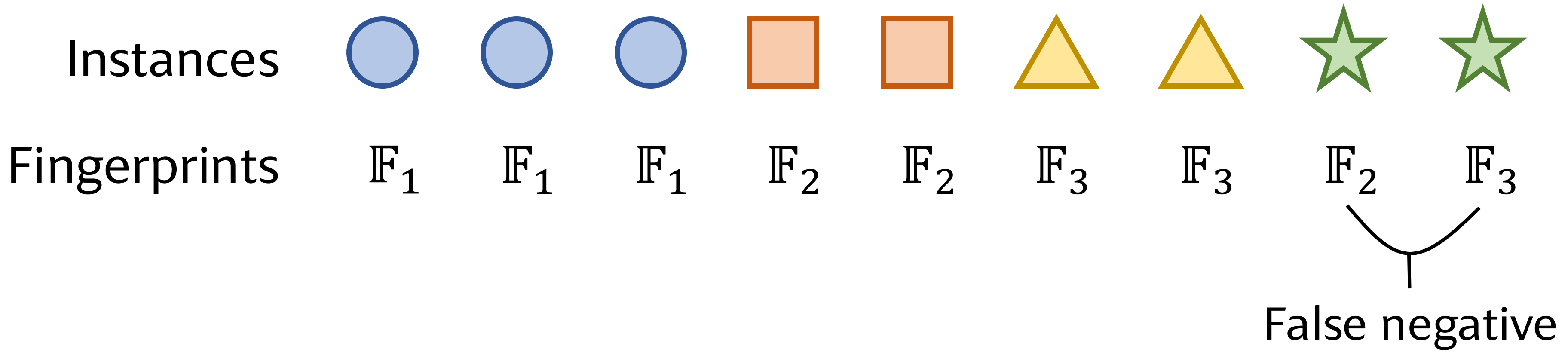
Batch test 3 instances at once

If fingerprints are accurate, they can provide hints on which instances are likely co-located

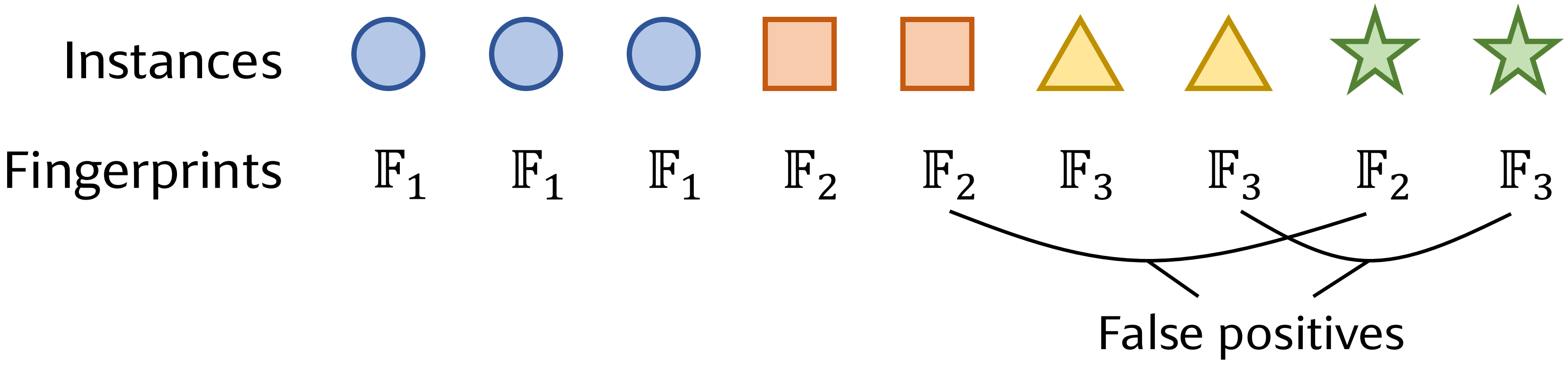
# Fingerprint-Assisted Co-Location Verification



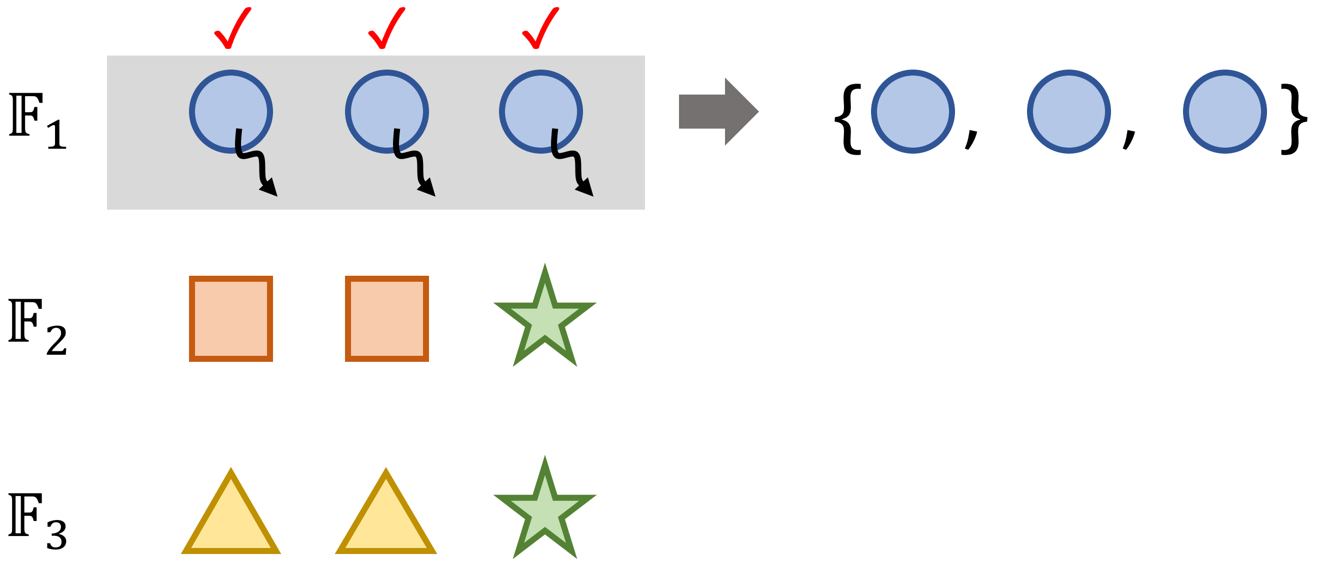
# Fingerprint-Assisted Co-Location Verification



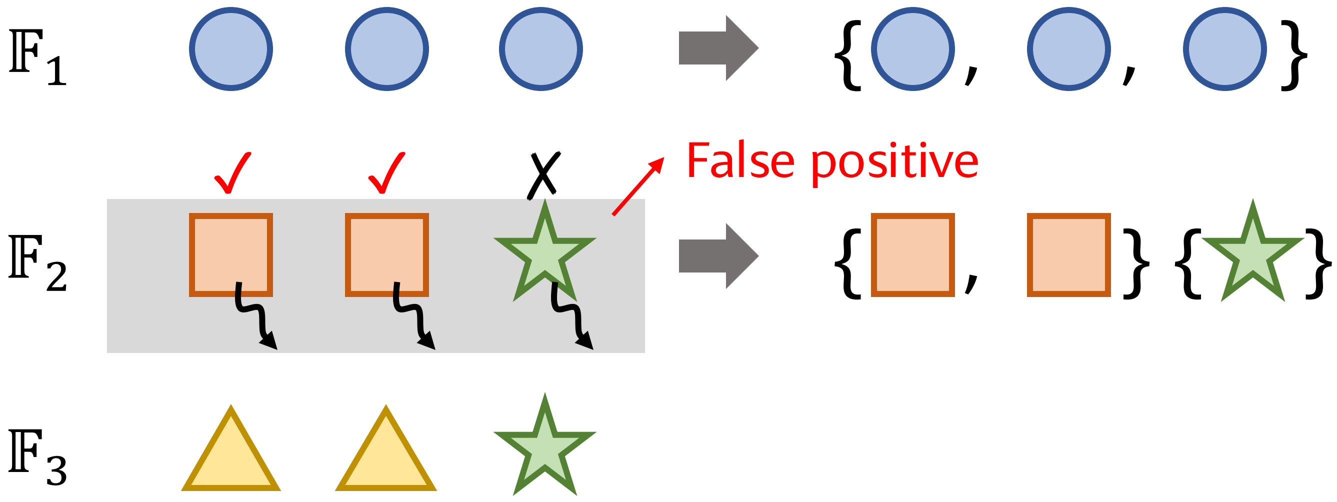
# Fingerprint-Assisted Co-Location Verification



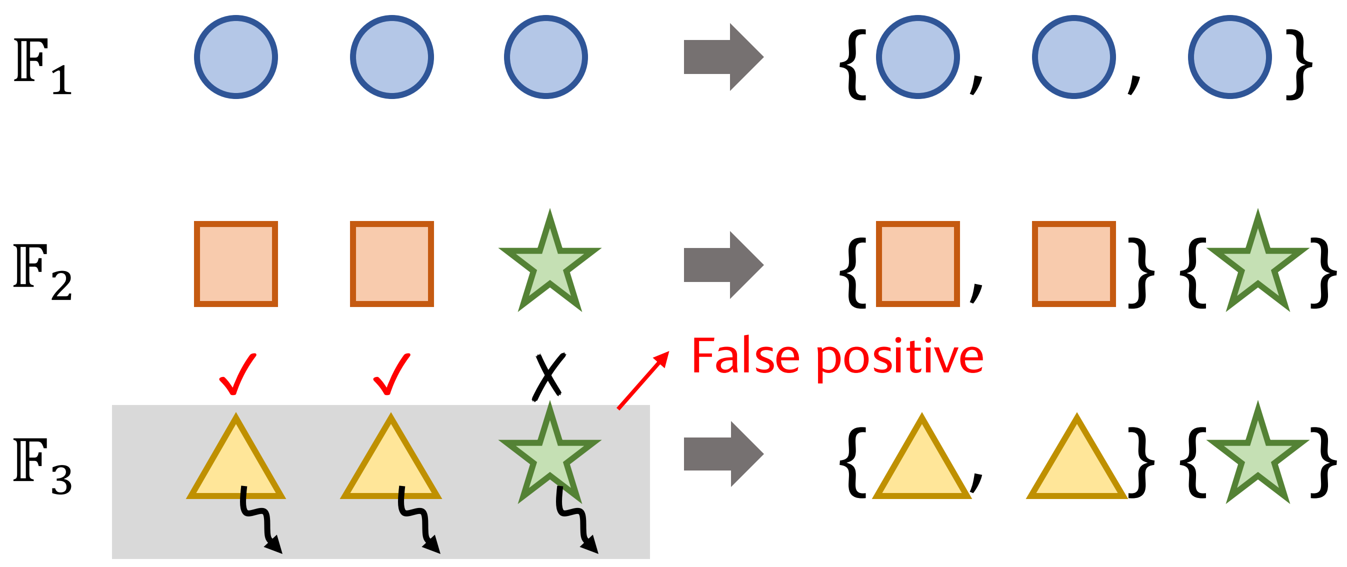
# Fingerprint-Assisted Co-Location Verification



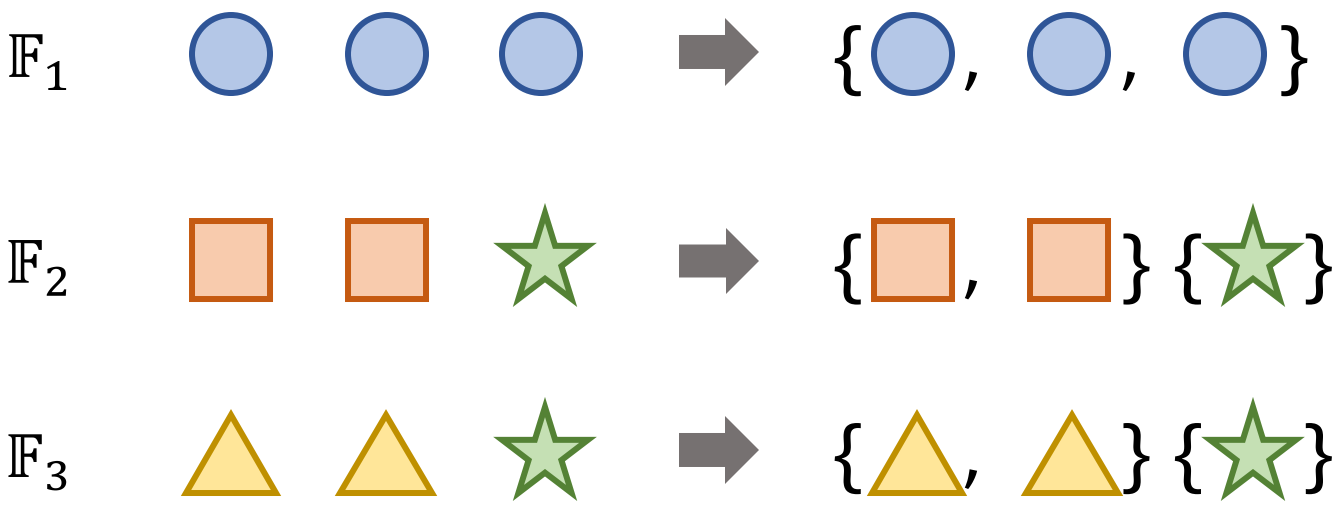
# Fingerprint-Assisted Co-Location Verification



# Fingerprint-Assisted Co-Location Verification

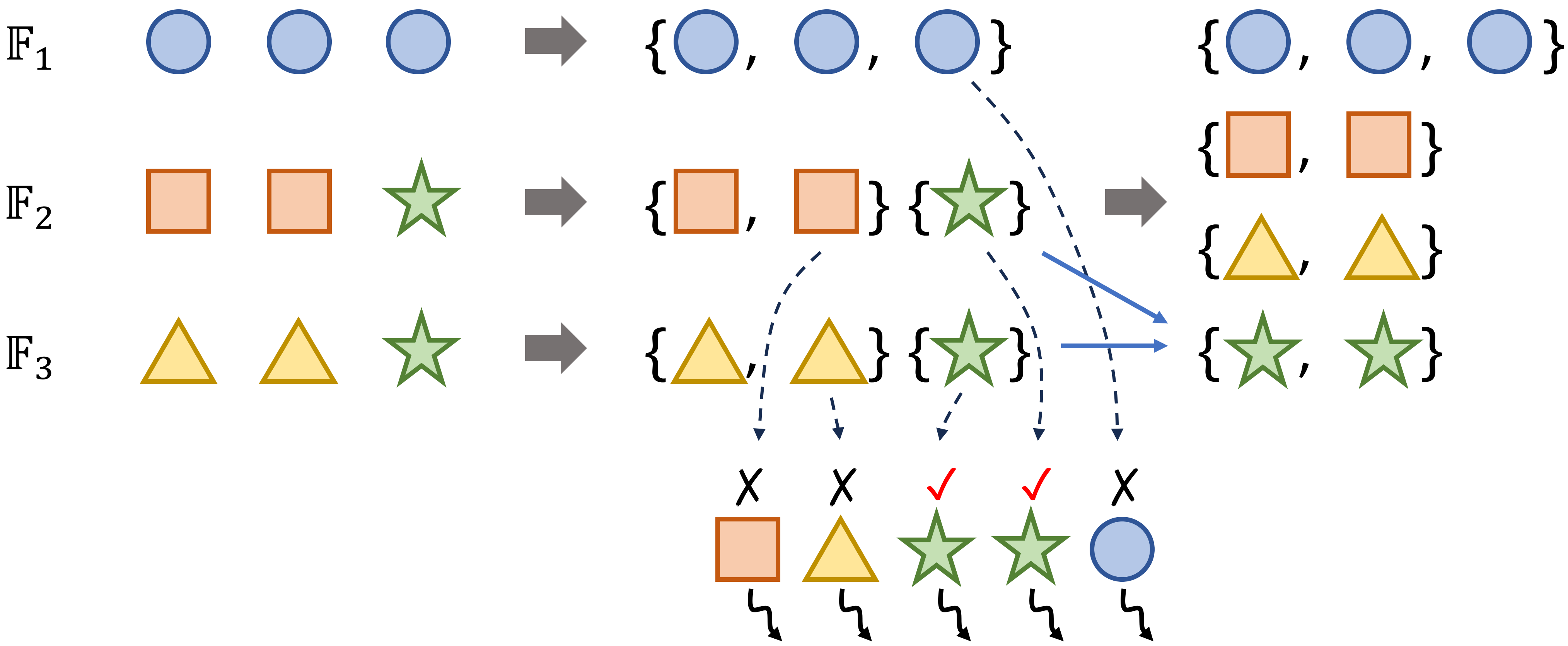


# Fingerprint-Assisted Co-Location Verification

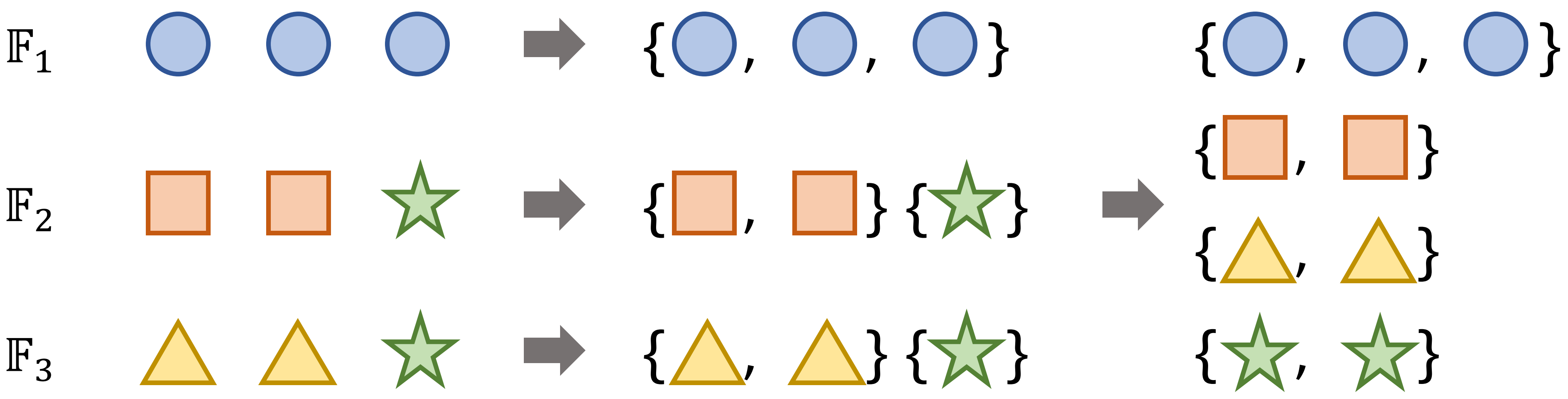



All false positives are identified, proceed to find false negatives

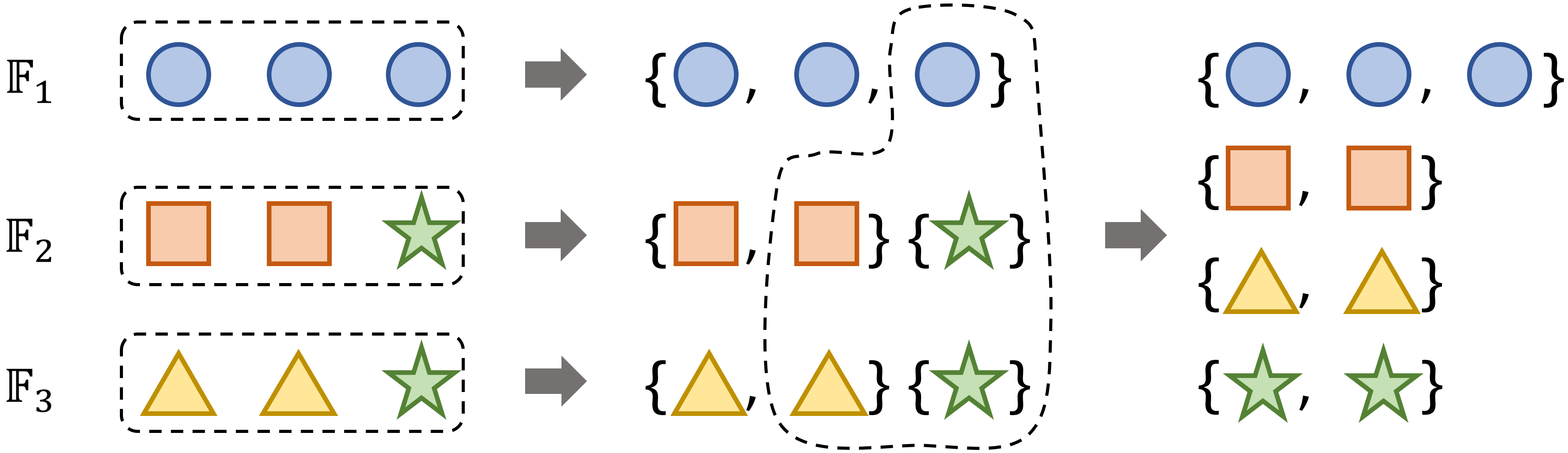
# Fingerprint-Assisted Co-Location Verification



# Fingerprint-Assisted Co-Location Verification



# Fingerprint-Assisted Co-Location Verification




4 batch tests instead of $9 \times 8/2 = 36$ pairwise tests

<u>More discussion in the post-lecture reading</u>

# Host Fingerprints are Highly Accurate

**For each pair of container instances**

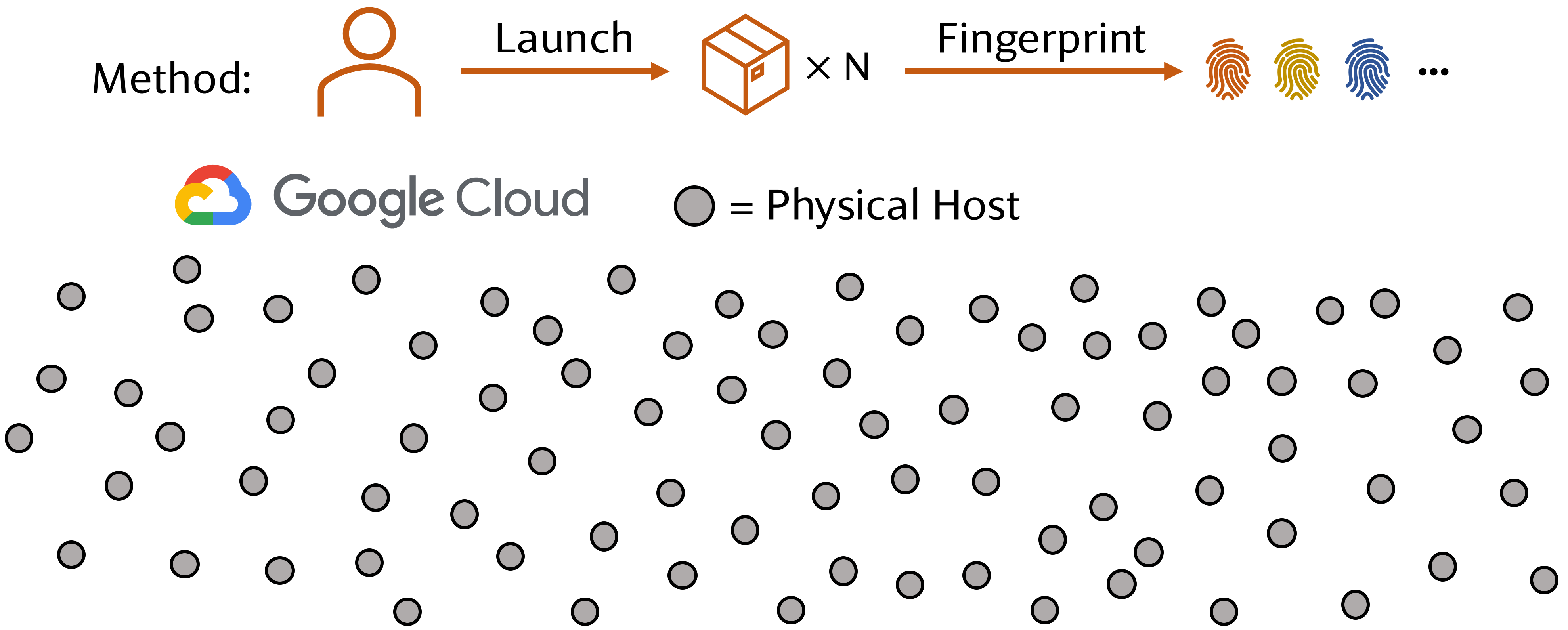
- **False positive (FP):** same fingerprints but not co-located
- **False negative (FN):** different fingerprints but co-located

- Measure accuracy in three data center regions (us-central1/east1/west1)
- Repeat measurements five times in each data center region

Average FN rate: 0.00%
Average FP rate: 0.02%

☺ 14 out of 15 measurements generate perfect fingerprints (no FP nor FN)

# Understanding Instance Placement Policy

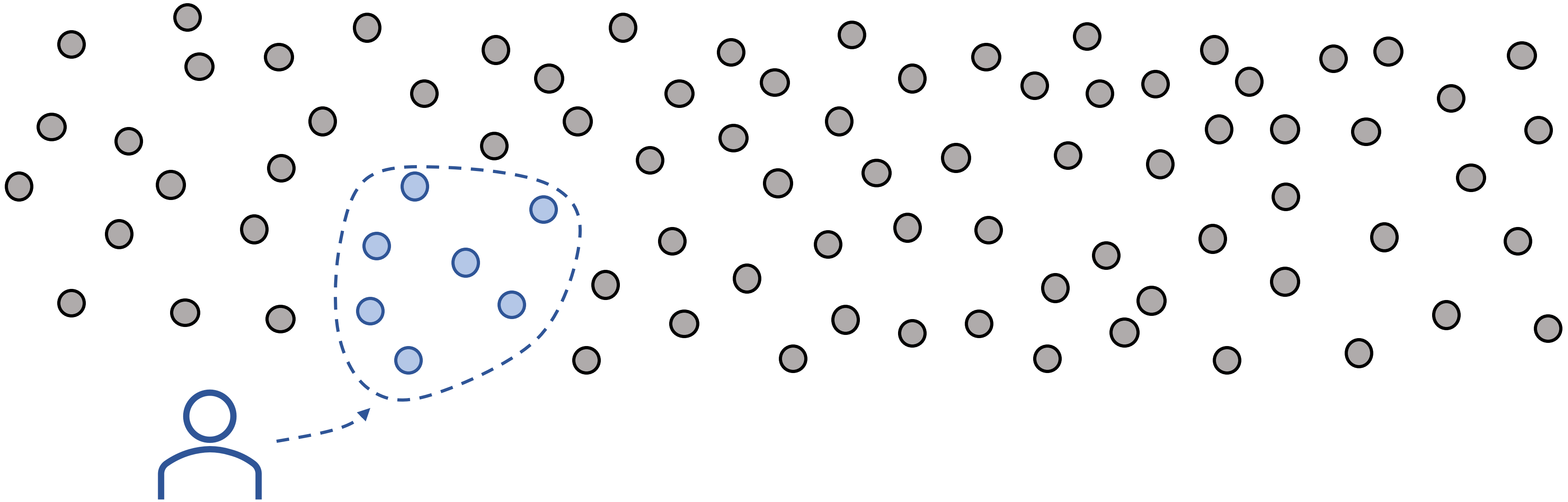


# Observation 1: An Account Has a Preferred Set of Hosts

**Why:** Affinity scheduling to reduce communication overhead

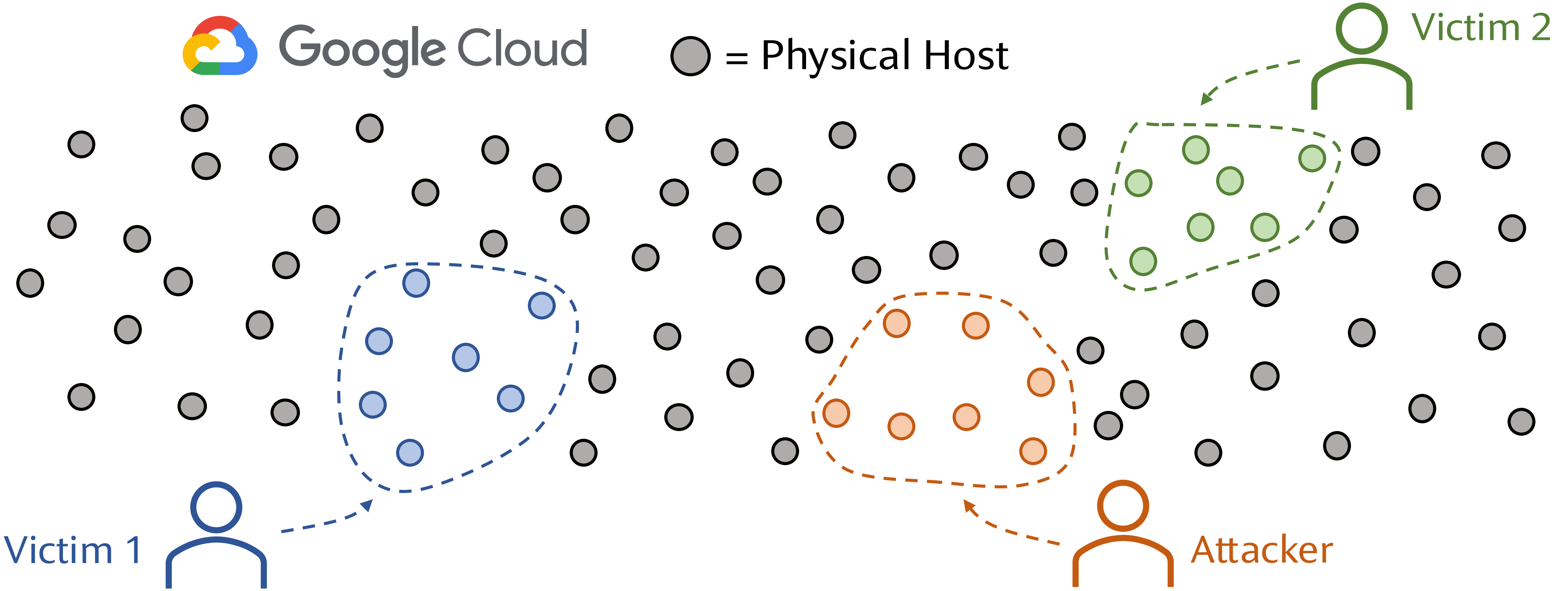
Google Cloud

= Physical Host

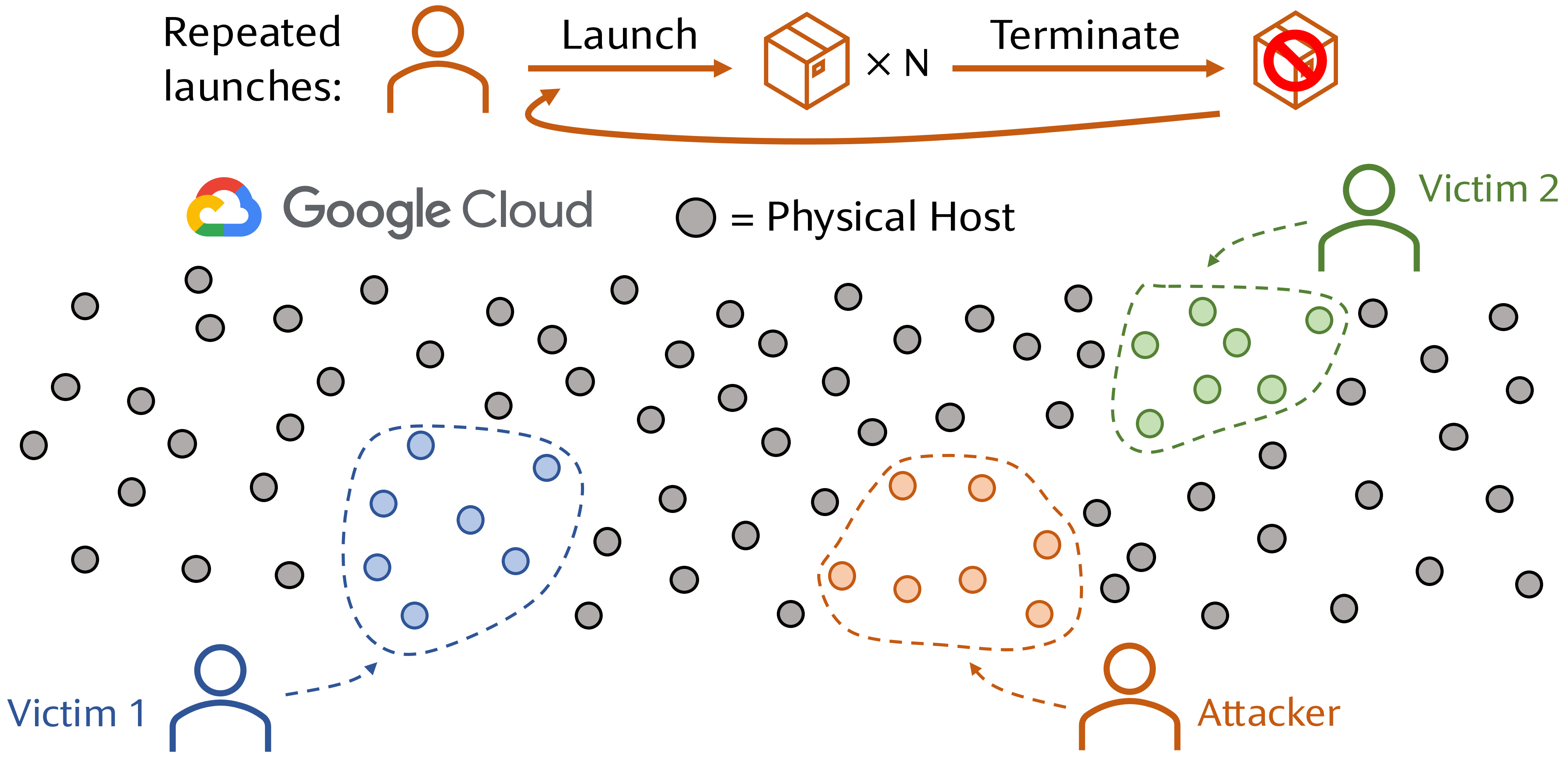


# Observation 2: Different Accounts Have Different Preferred Hosts

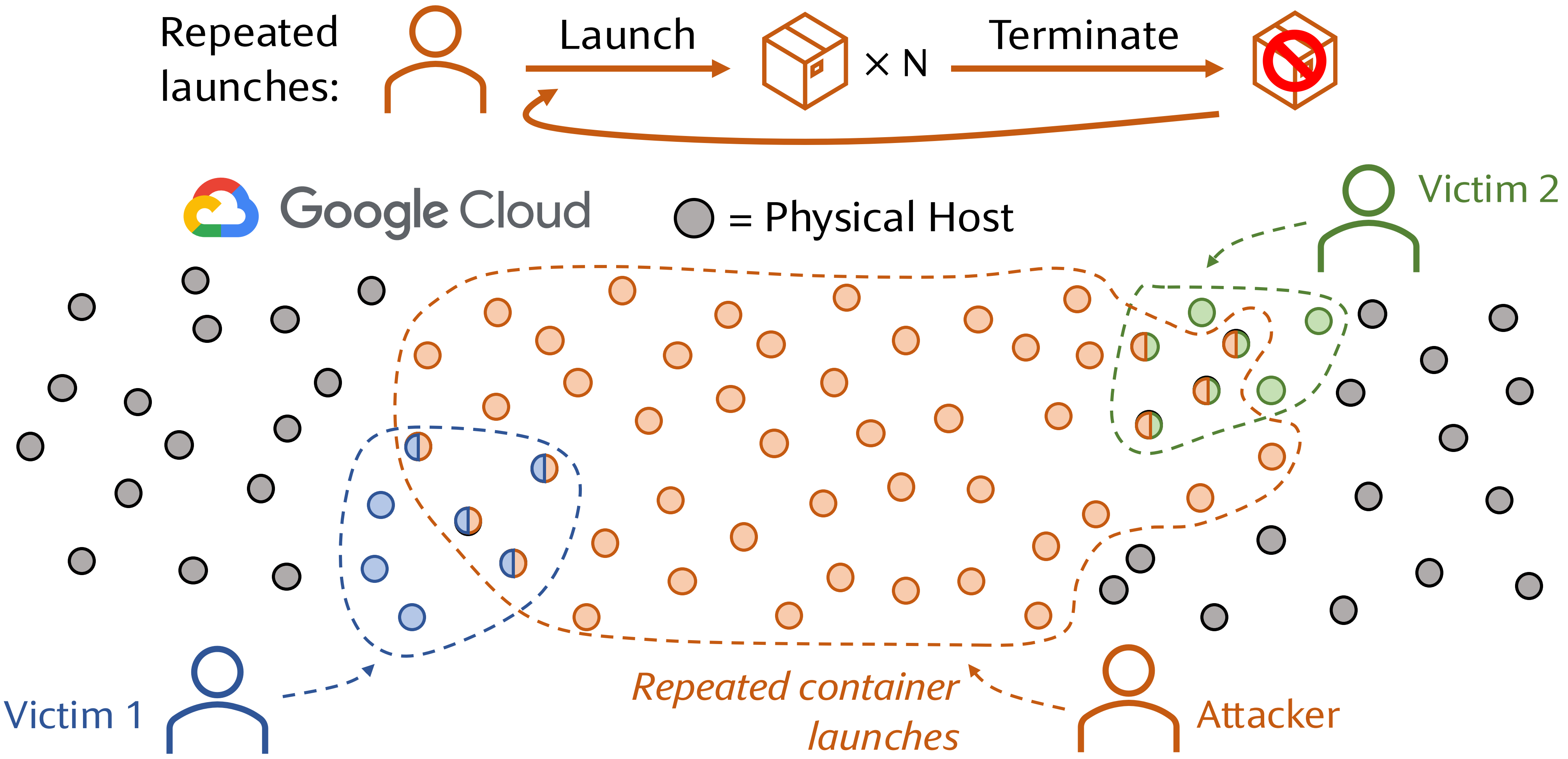
**Implication:** Low chance of co-location with a target user



# Observation 3: Repeated Launches Spread Instances

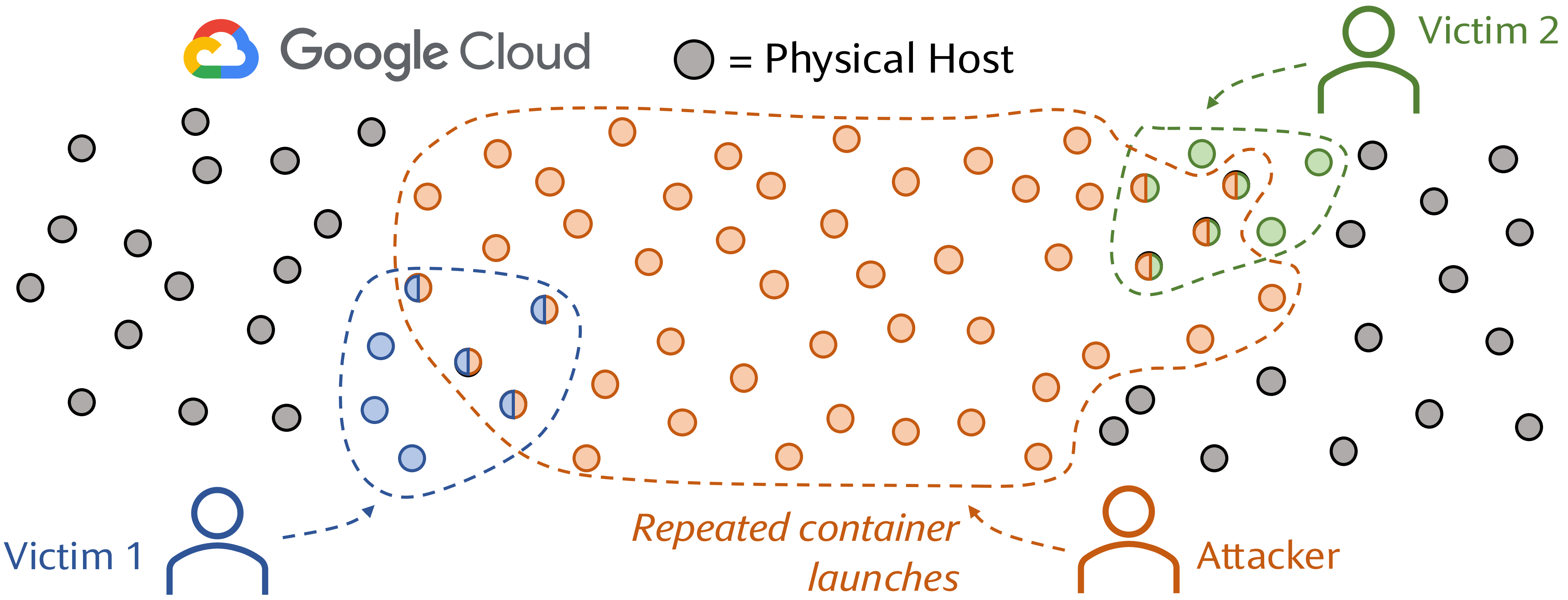


# Observation 3: Repeated Launches Spread Instances



# Observation 3: Repeated Launches Spread Instances

**Why:** Repeated launches ⇒ User has high demand ⇒ Load balance

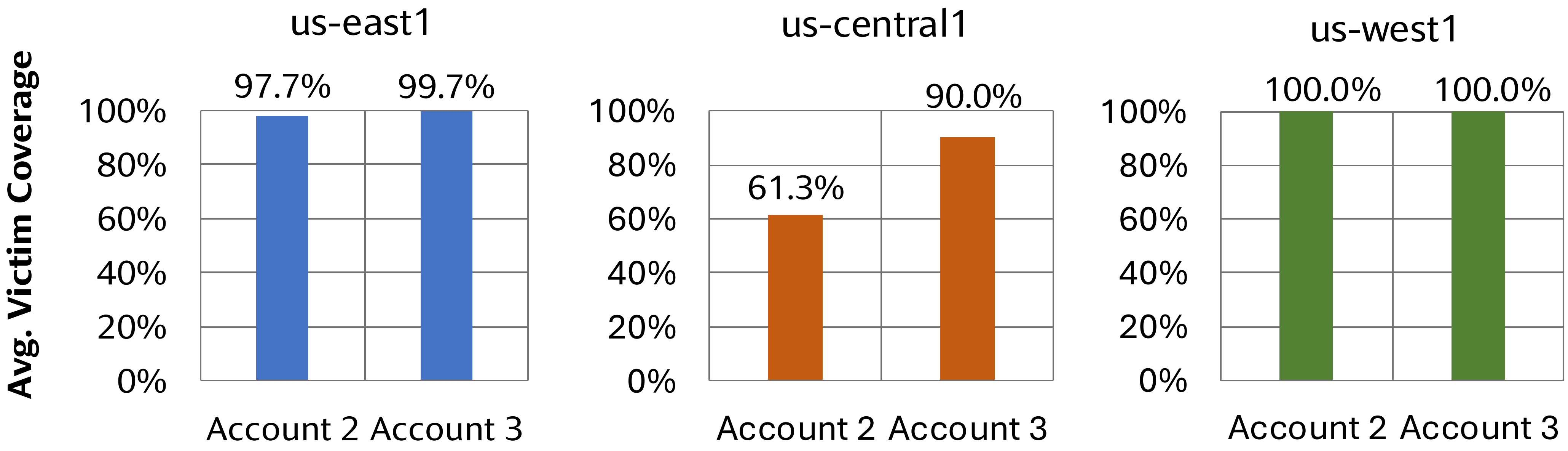


# Evaluation: Co-Location with Victims


**Victim coverage**: Percentage of victim instances that are co-located with the attacker

# High Victim Coverage and Low Attack Cost

**Average Victim Instance Coverage** (3 repetitions in each region)

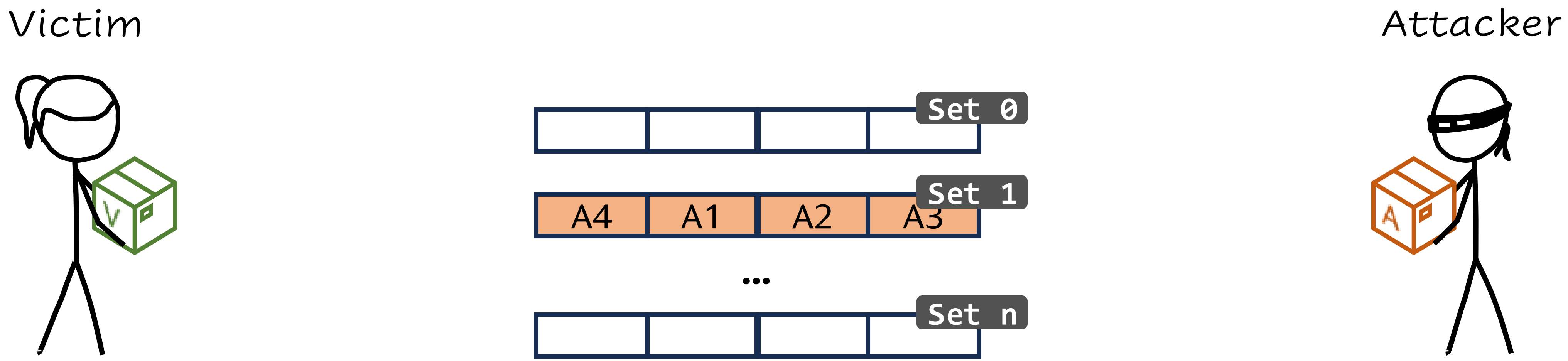



**Avg. Attack Cost:** 24 USD 23 USD 27 USD

**Takeaway:** High victim coverage and low attack cost

# LLC Prime+Probe Attack with an Eviction Set

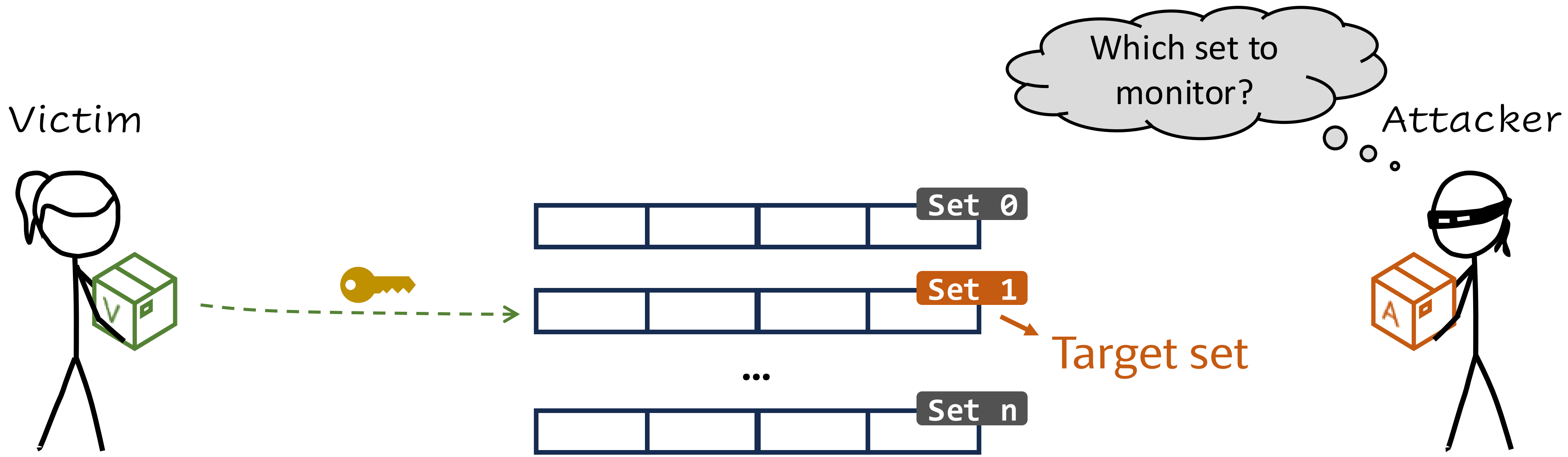
Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**




4-way set-associative
last-level cache (LLC)

# An Unprivileged Attacker Does Not Know the Target Set

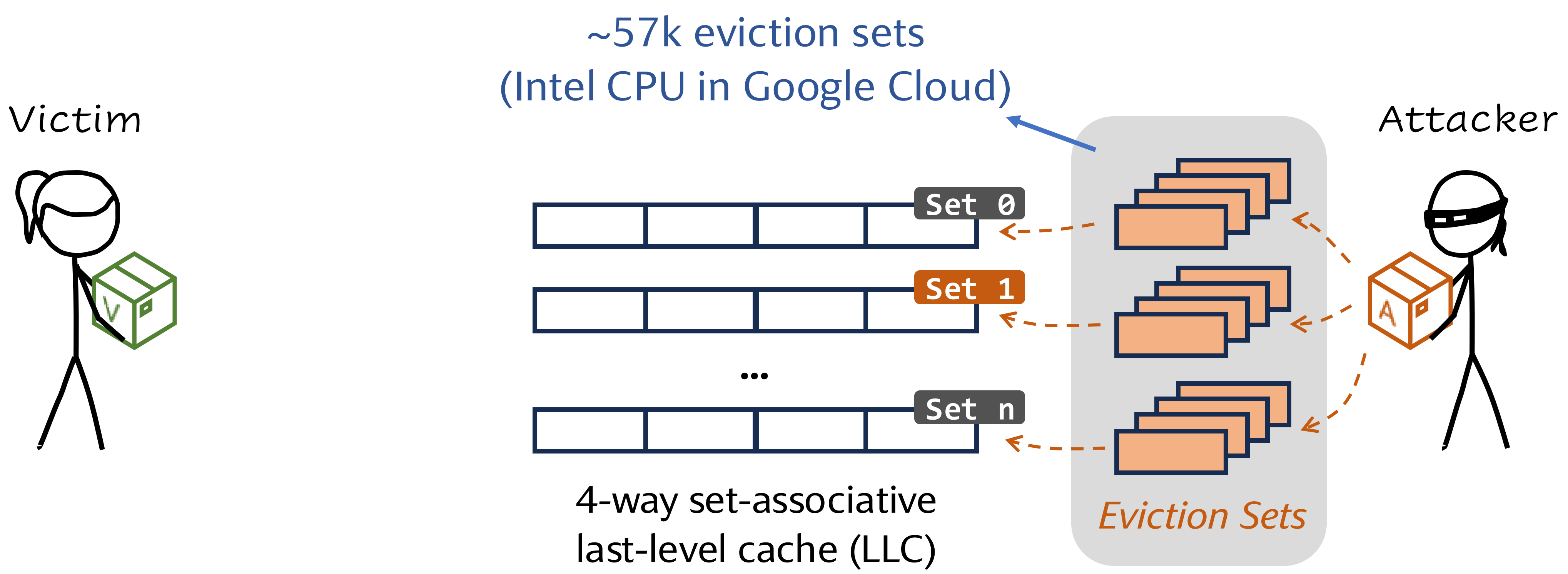
**Target set:** An LLC set accessed by the victim in a secret-dependent manner




4-way set-associative
last-level cache (LLC)

# Step 1: Build Many Eviction Sets

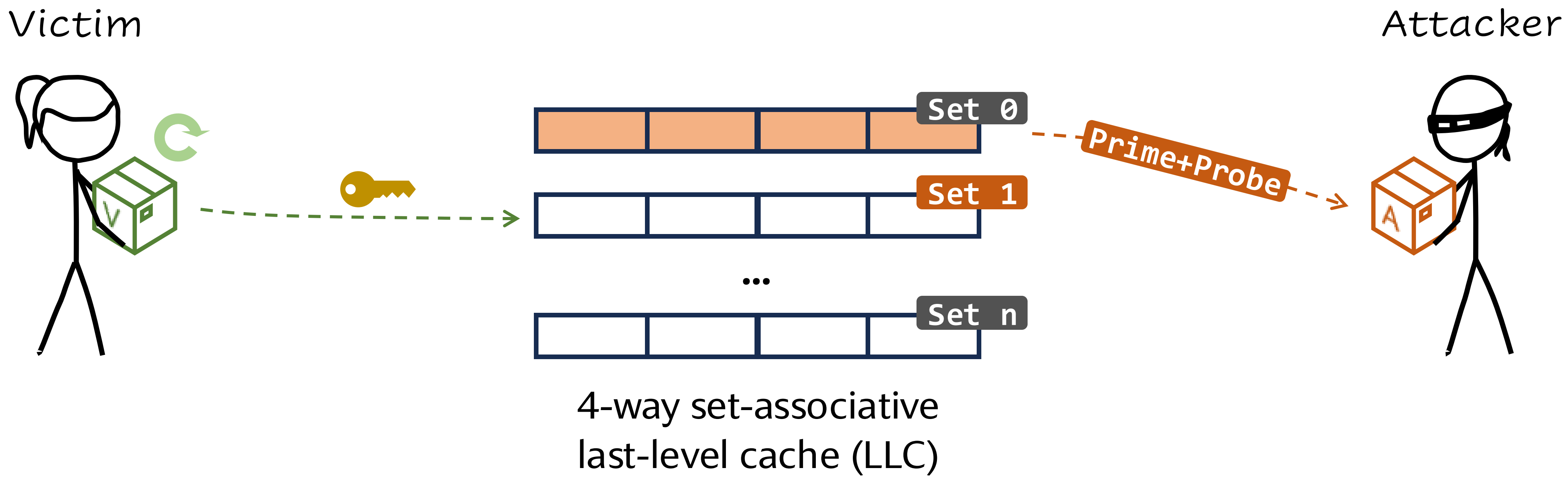
Attacker needs an eviction set for every LLC set in the system



# Step 2: Identify Target LLC Set to Monitor

Attacker collects an access trace from *each* LLC set

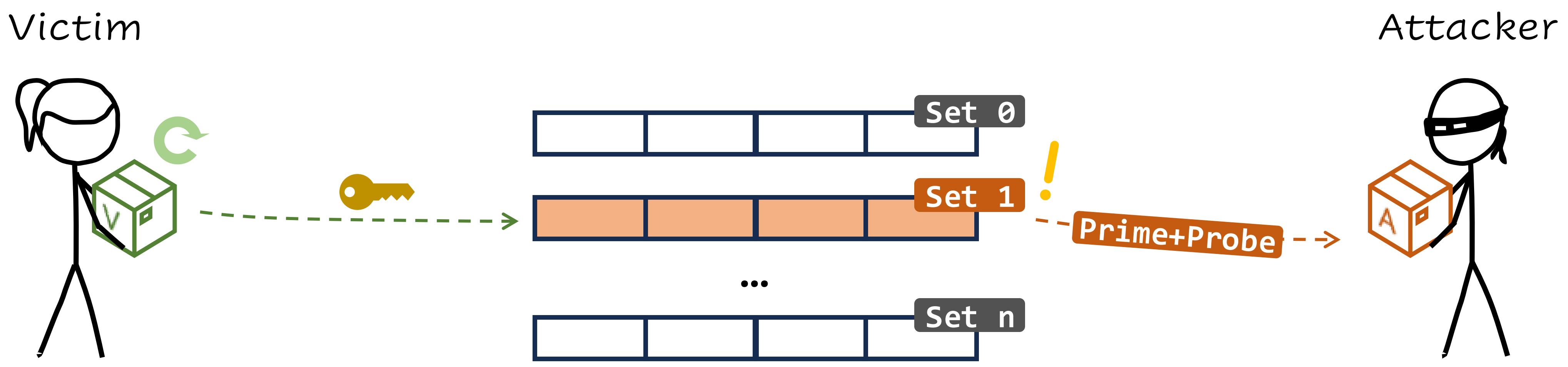
⇒ Checks whether the access trace matches victim's access behavior



# Step 2: Identify Target LLC Set to Monitor

Attacker collects an access trace from *each* LLC set

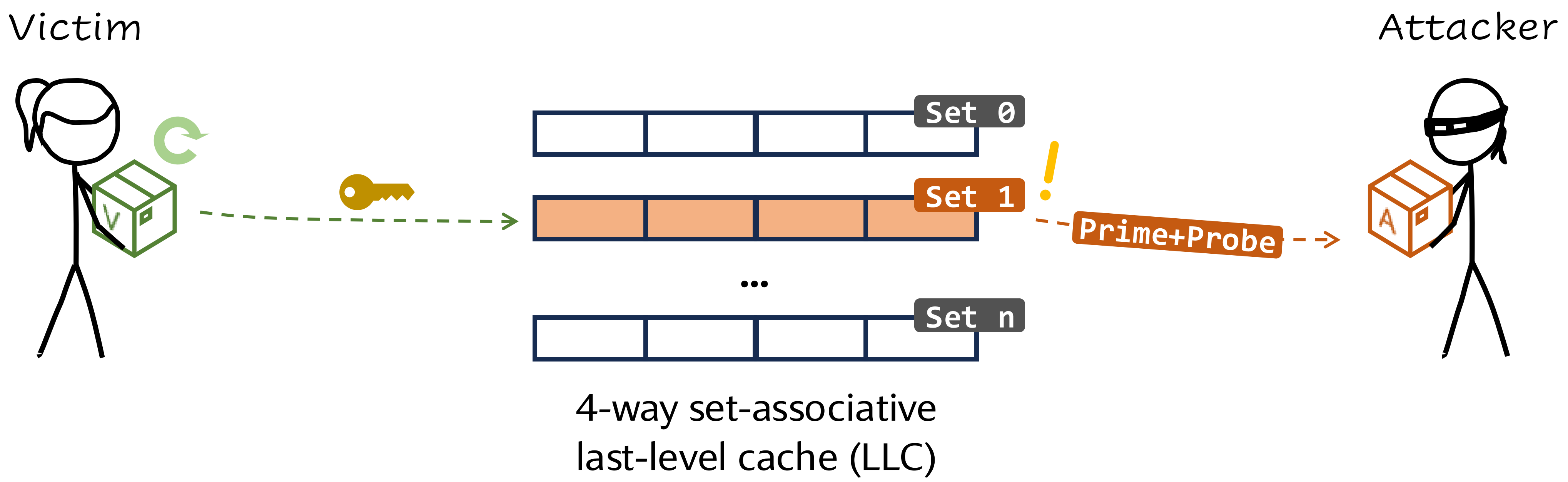
⇒ Checks whether the access trace matches victim's access behavior




4-way set-associative
last-level cache (LLC)

# Step 3: Extract Information from the Victim

Attacker monitors the target set and extracts the sensitive information



# Victim: Elliptic Curve Digital Signature Algorithm (ECDSA)

**Target victim:** A vulnerable ECDSA program from OpenSSL 1.0.1e

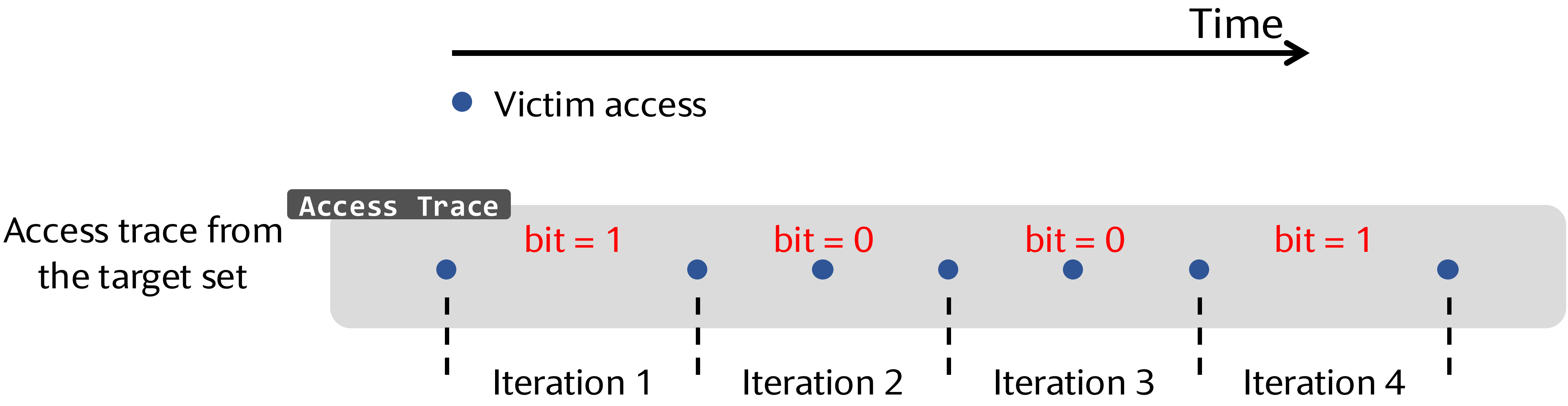
**Setup:** The victim runs in a container owned by a different Google account

**Simplified victim code**

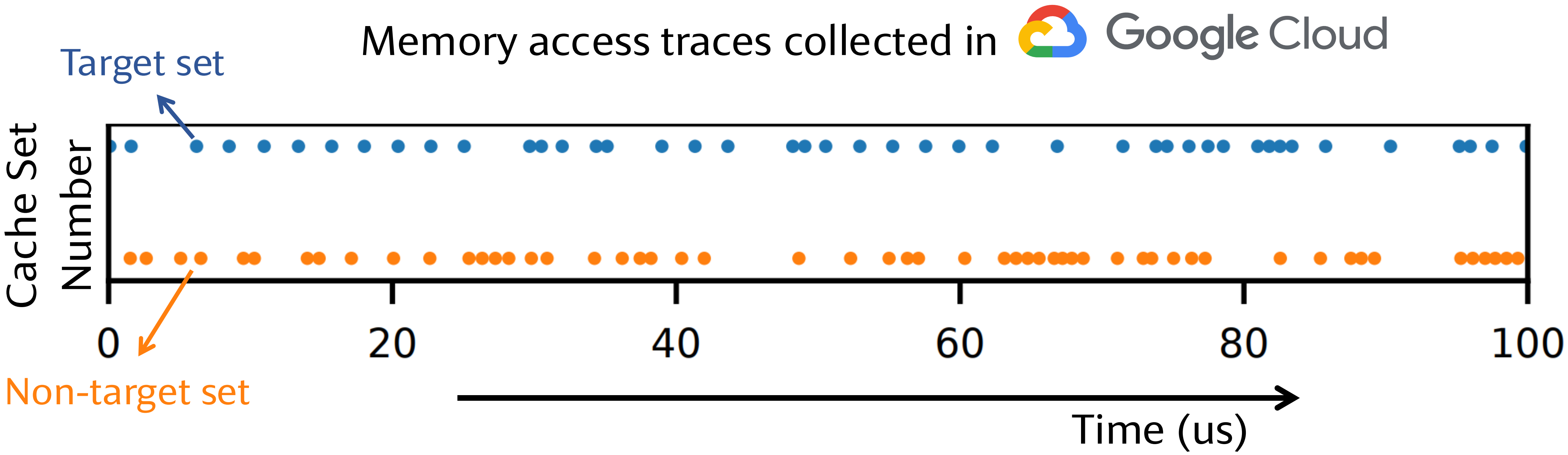
```
for bit in secret_nonce {
  // ...
  if (!bit) {
    Fetch Victim Address;
  }
  // ...
  Fetch Victim Address;
}
```

Can be derive the private key

# Expected Access Trace from the Target Set

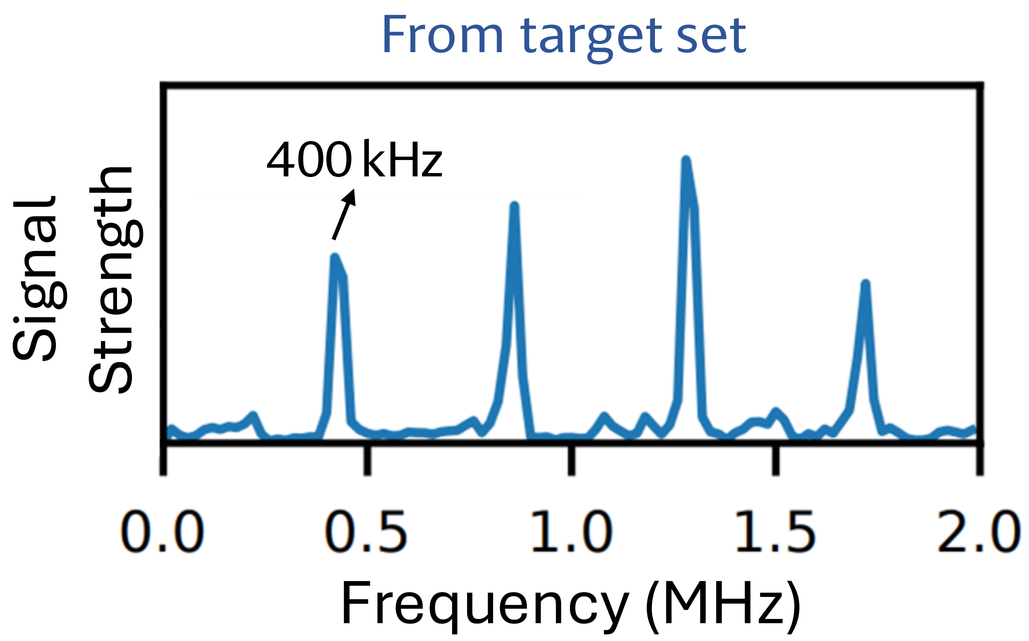


# Actual Access Traces Collected in Google Cloud



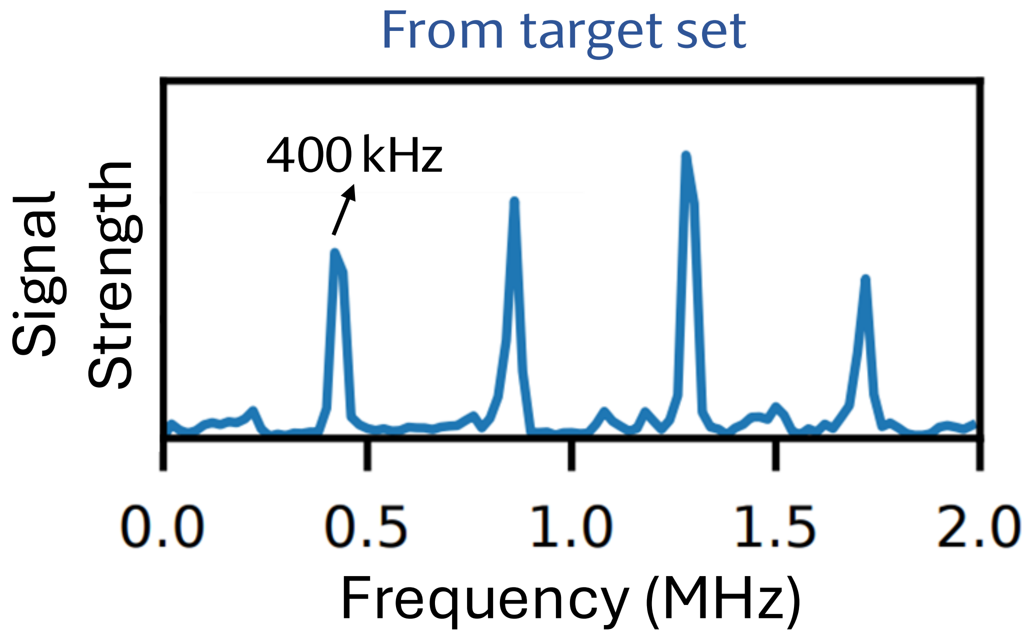
# New Technique: Identify Target Set via Spectral Analysis

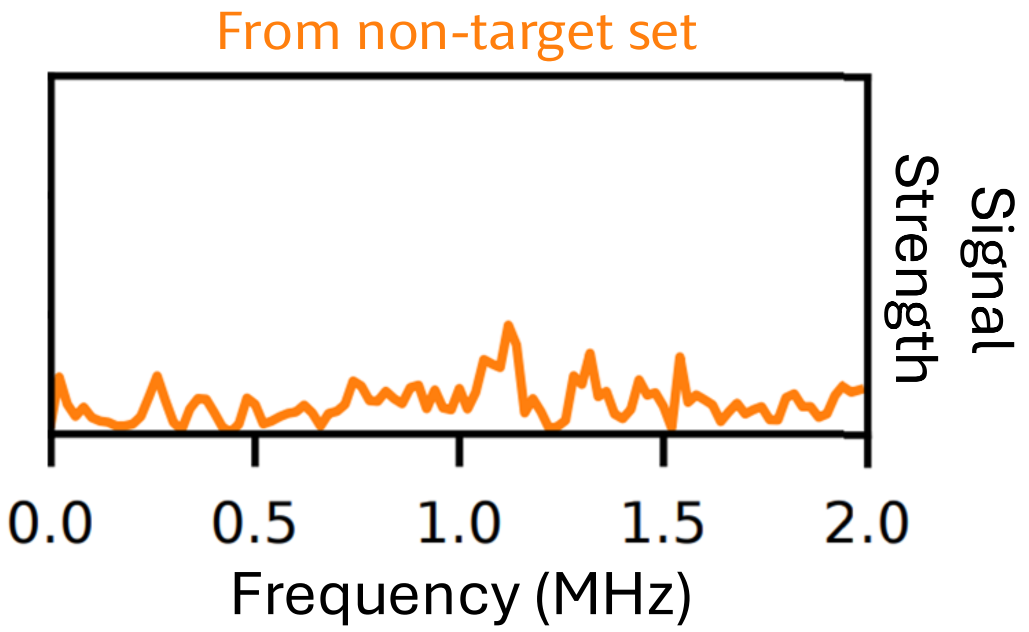
**Intuition:** Victim accesses are periodic ⇒ Spectral analysis



# New Technique: Identify Target Set via Spectral Analysis

**Intuition:** Victim accesses are periodic ⇒ Spectral analysis




**Result:** Find the target set in ~3 minutes, 74% success rate

Google Cloud

**Legends**

- Attacker Container
- Victim Container
- Task Running
- Task Completed
- Task Failed
- Victim Found

**Attack stage:**

Not Started

Preflight | Next | Repeat | Reset

**Running Time:** 0.0s

**Attacker Count:** 0

**Server Count:** 0

**Cost:** $0

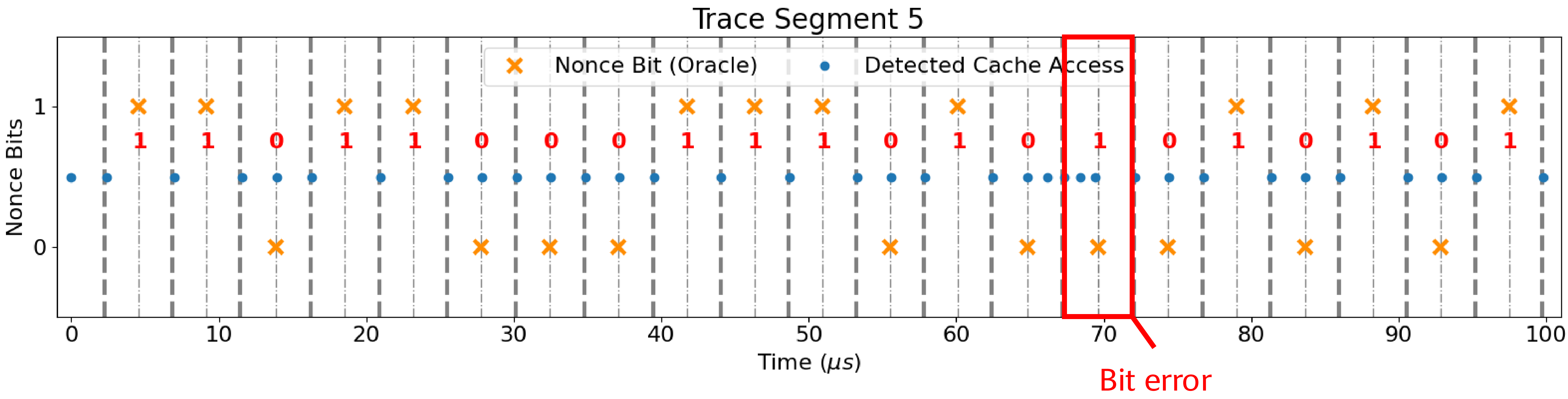
**Attack Traces:**

WebSocket Status: Connected

Mode: Live Switch

# Extracted Traces

Percentage of Secret Bits Extracted: 88.4%; Error rate: 2.8%;



# Next Two Lectures: Defenses to Side Channel

- Partitioning => Limit the sharing
- Randomization => Obfuscate the usage
- Detection => Catch the offender

# A Few Announcements and Reminders

- Please sign up for paper discussion if you haven't
  - Will randomly map students to papers after Weds

- I will go over some of the paper reviews next lecture and answer the discussion questions. I'll grade them later this week

- May further reduce the reading load
  - 2 pre-lecture + 1 review / week or
  - 2 reviews / week