



Understanding and Mitigating Covert Channel and Side Channel Vulnerabilities Introduced by RowHammer Defenses

F. Nisa Bostancı

O. Canpolat A. Olgun I. E. Yüksel K. Kanellopoulos
M. Sadrosadati A. G. Yağlıkçı O. Mutlu

Talk Outline

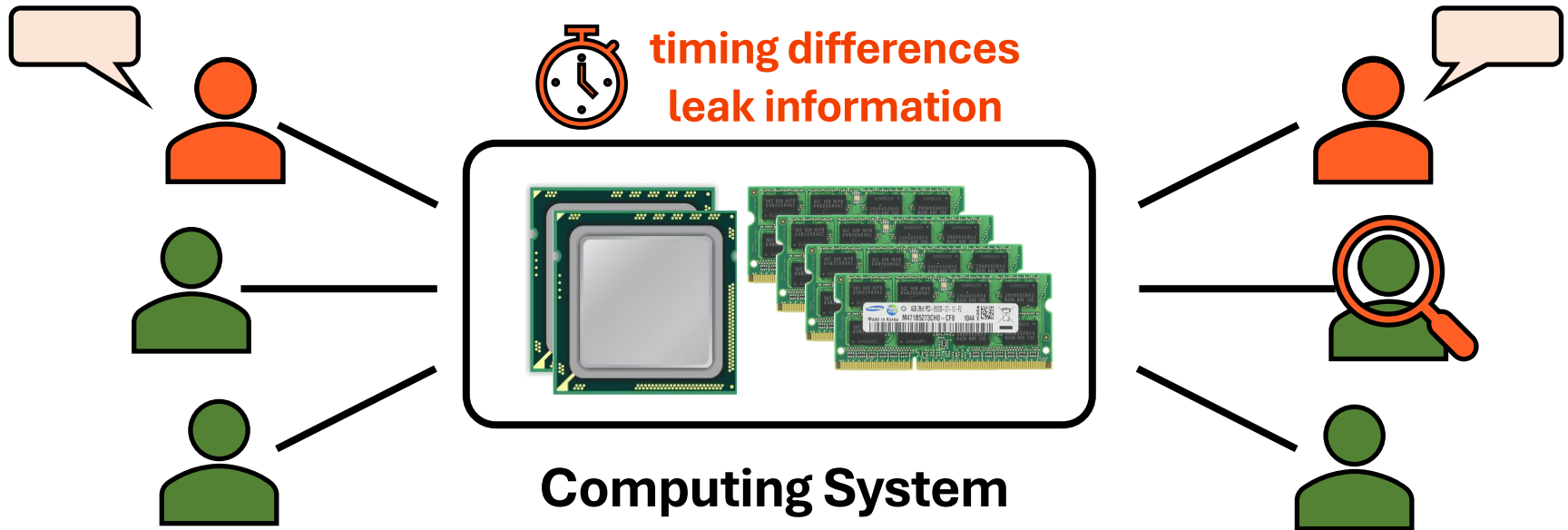
- I. Motivation and Problem
- II. Key Observation
- III. LeakyHammer and Case Studies
- IV. Mitigating LeakyHammer
- V. Conclusion

Talk Outline

- I. Motivation and Problem**
- II. Key Observation
- III. LeakyHammer and Case Studies
- IV. Mitigating LeakyHammer
- V. Conclusion

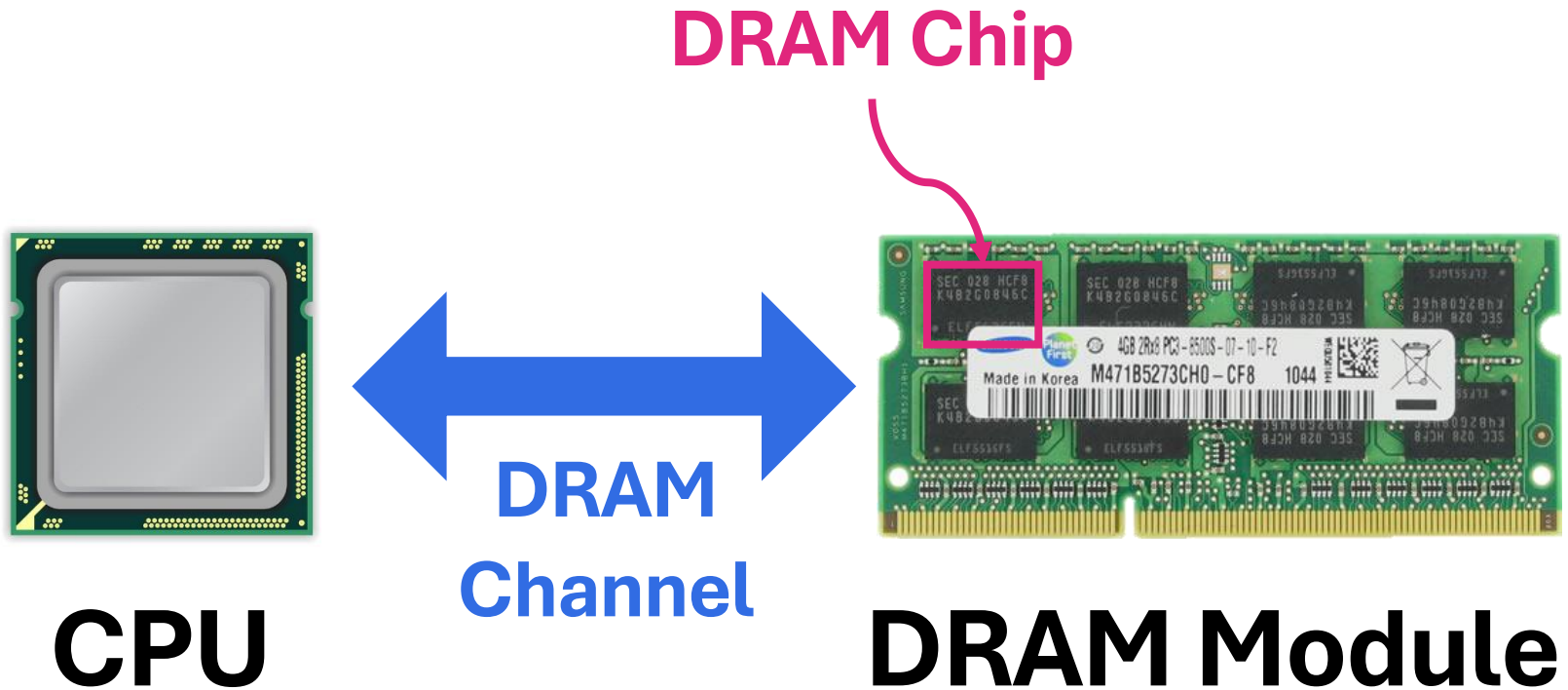
A Typical DRAM-Based Computing System

- Modern systems are vulnerable to **security attacks** such as **covert channel and side channel attacks**

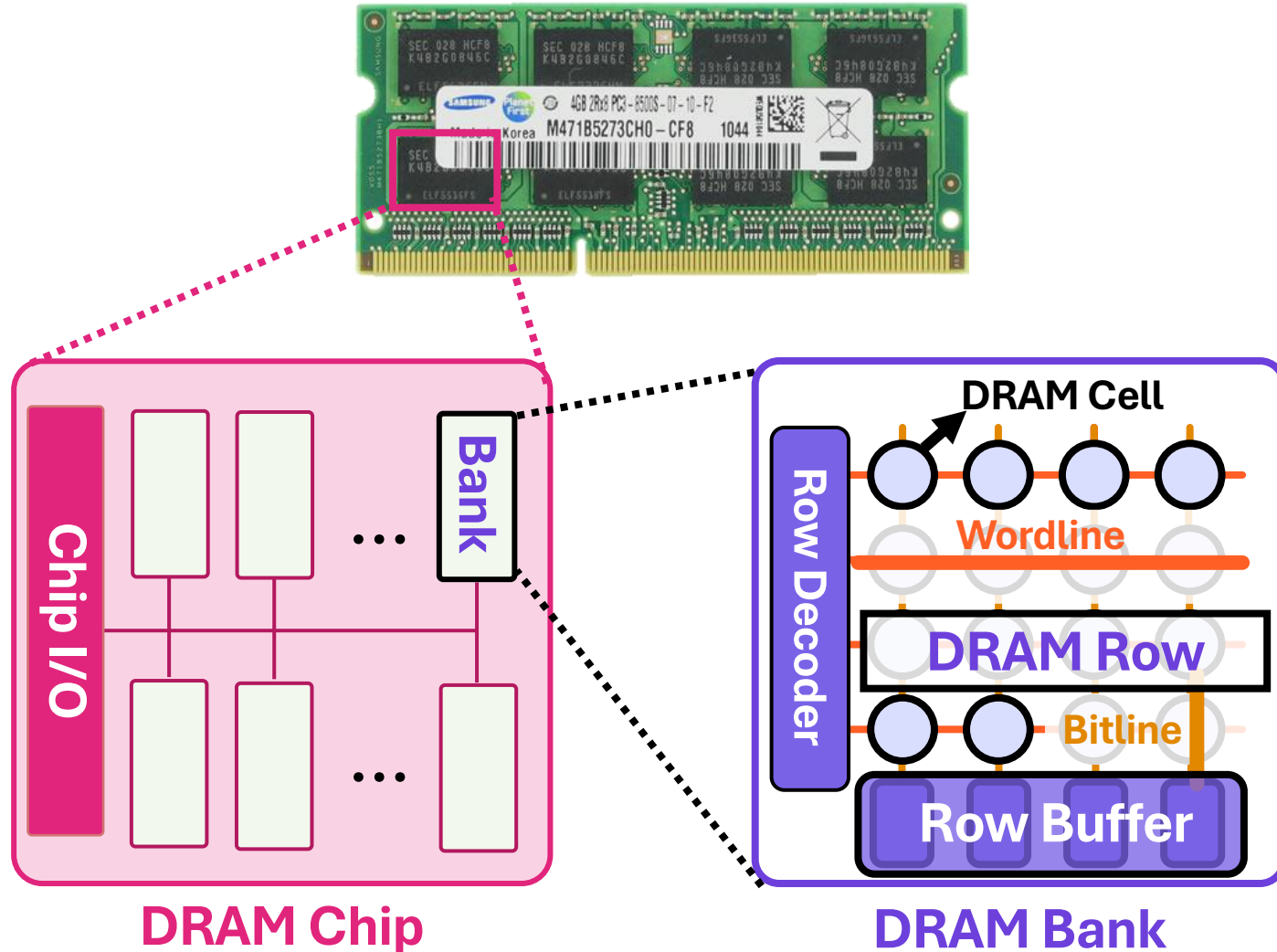


- A set of attacks **exploit hardware structures** that result in **different execution time for a type of operation** based on the input or the current state of the structure

A Typical DRAM-Based Computing System

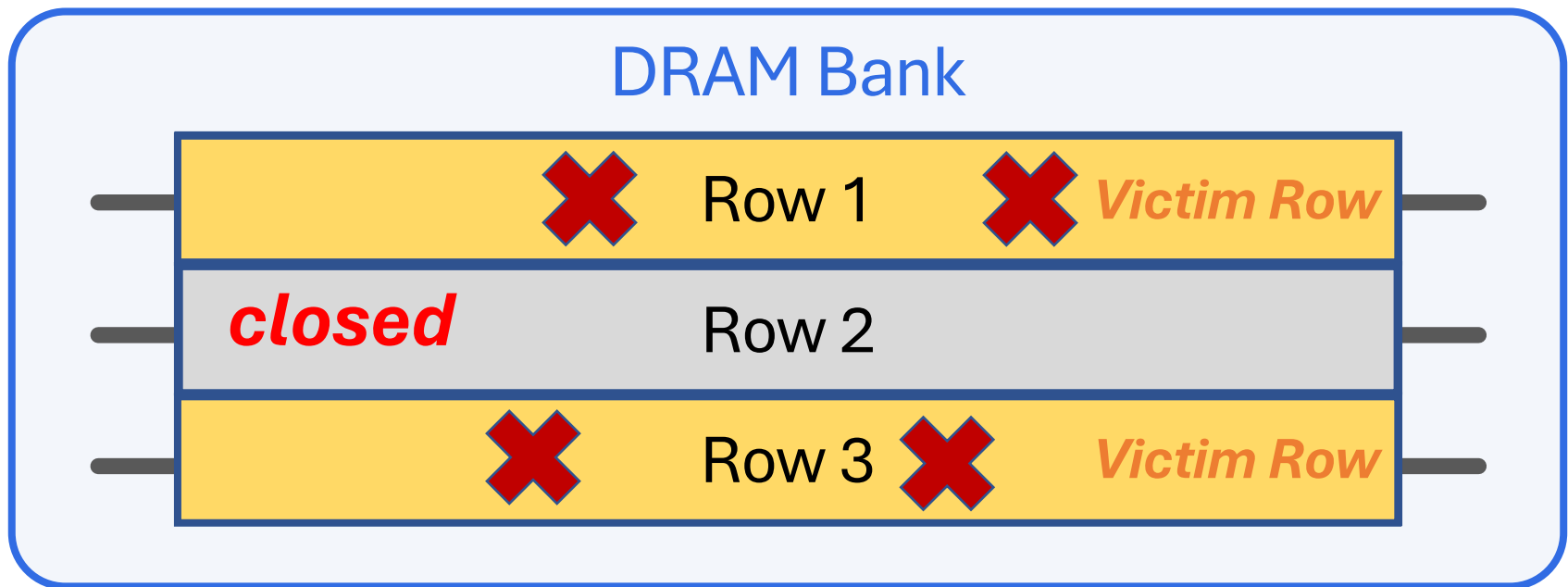


DRAM Organization



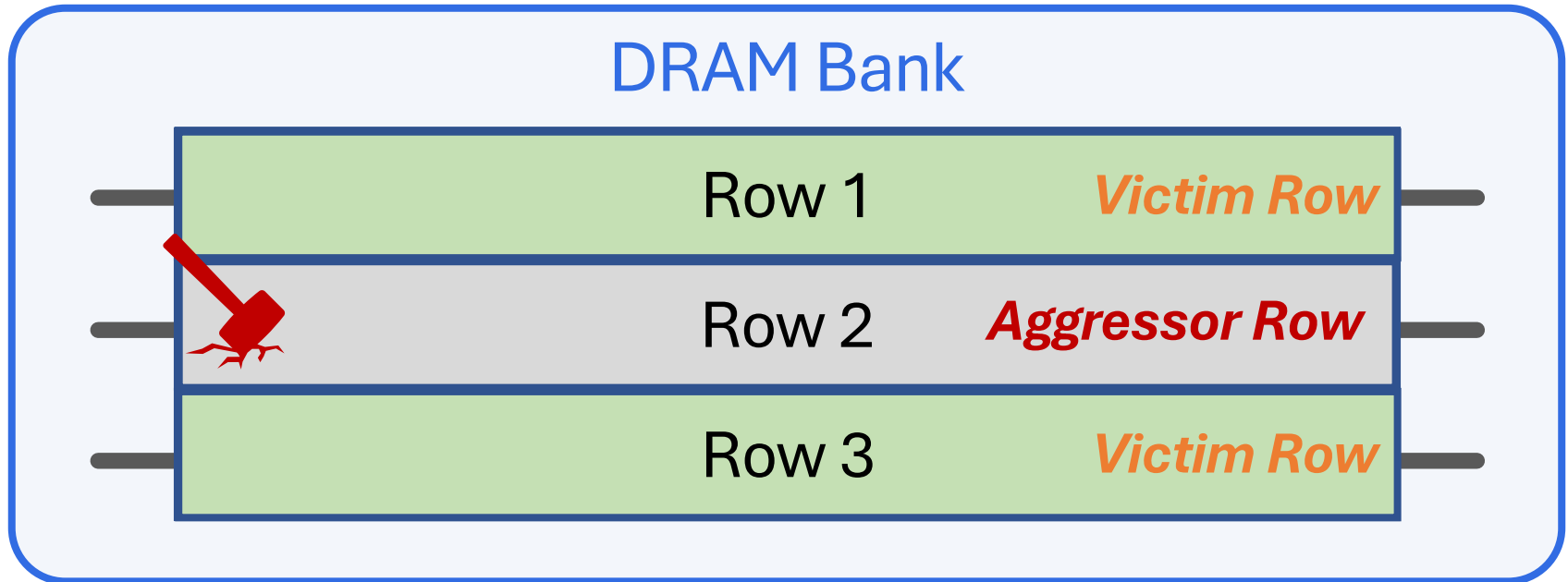
DRAM Read Disturbance

- Read disturbance in DRAM **breaks memory isolation**
- Prominent example: **RowHammer**



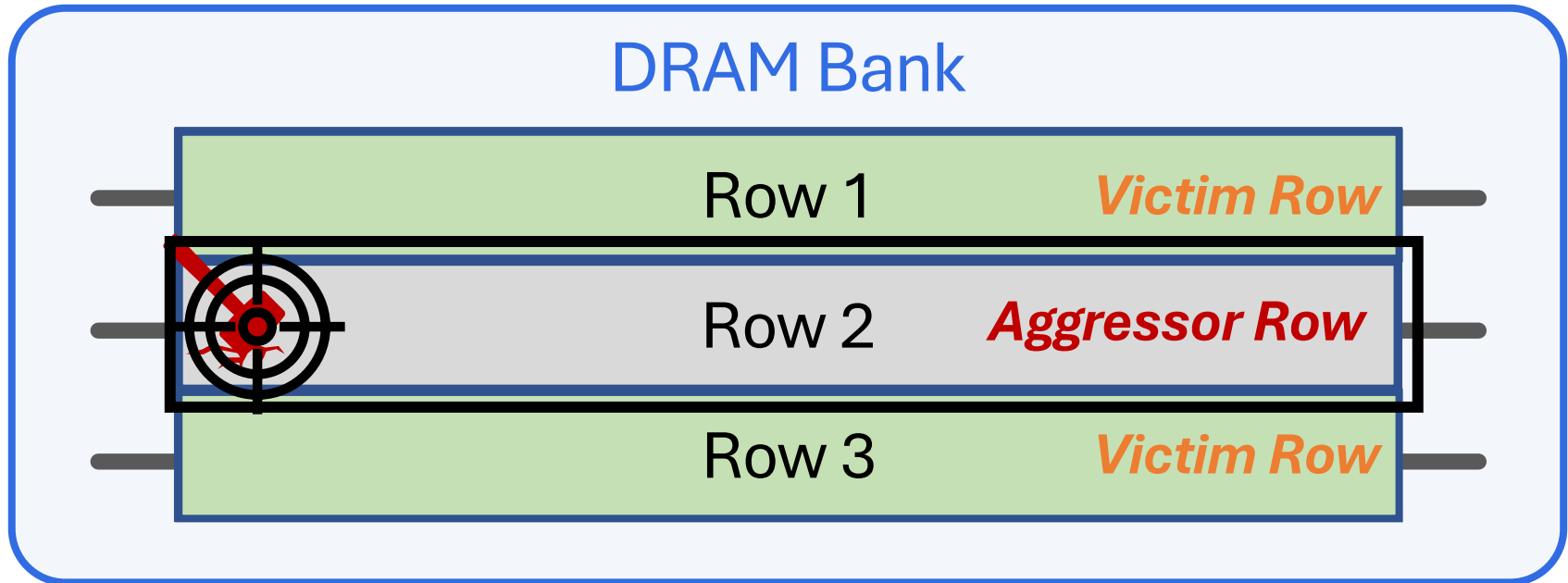
Repeatedly **opening** (activating) and **closing** (precharging) a DRAM row causes **RowHammer bitflips** in nearby cells

Mitigating Read Disturbance (I): Preventive Refresh



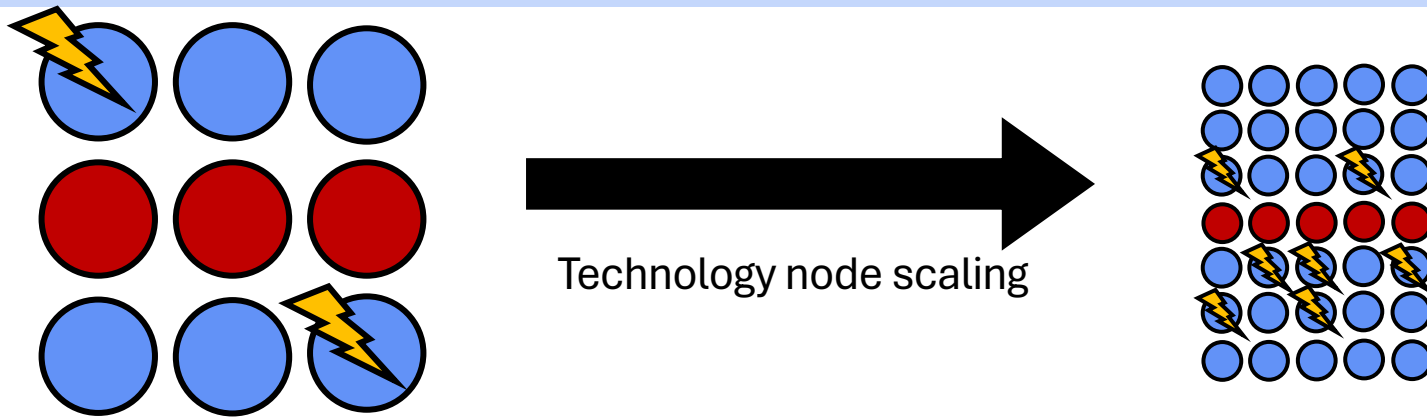
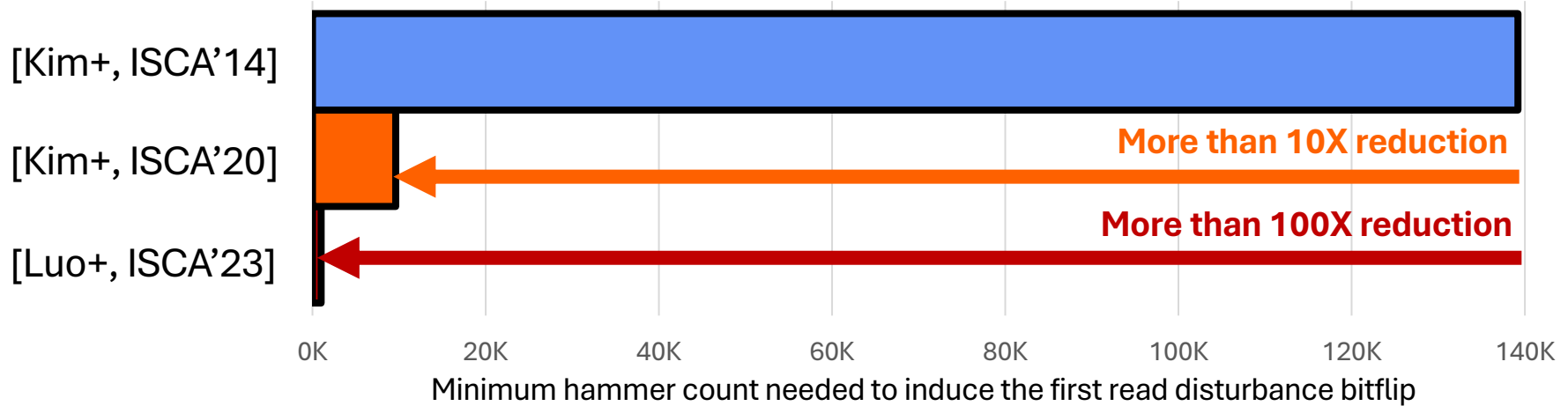
**Refreshing potential victim rows
mitigates read disturbance bitflips**

Mitigating Read Disturbance (II): Preventive Refresh



Defense techniques **track row activation counts** (of aggressor rows) to **preventively refresh** potential victim rows

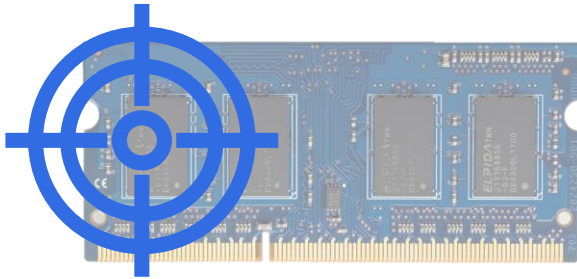
Read Disturbance is Getting Worse



DRAM cells become **increasingly more vulnerable to read disturbance**, requiring **low-cost defenses**

Latest Industry Defenses for RowHammer

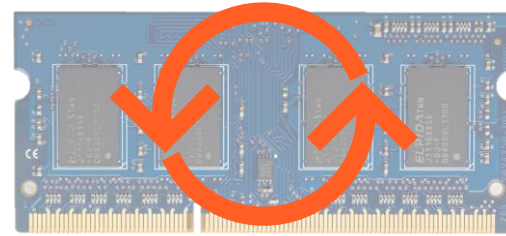
Recent DDR5 specifications introduce **two new defense frameworks** to **mitigate RowHammer bitflips**:



Per-row activation counters
to accurately **track** aggressor rows'
activation counts

**Per Row Activation Counting
(PRAC)**

as of April 2024



A command to
provide DRAM with the time
for **performing preventive actions**

Refresh Management (RFM)
before 2024

Do industrial and academic
RowHammer defenses introduce
**new covert and side channel
vulnerabilities?**

Unfortunately, yes.

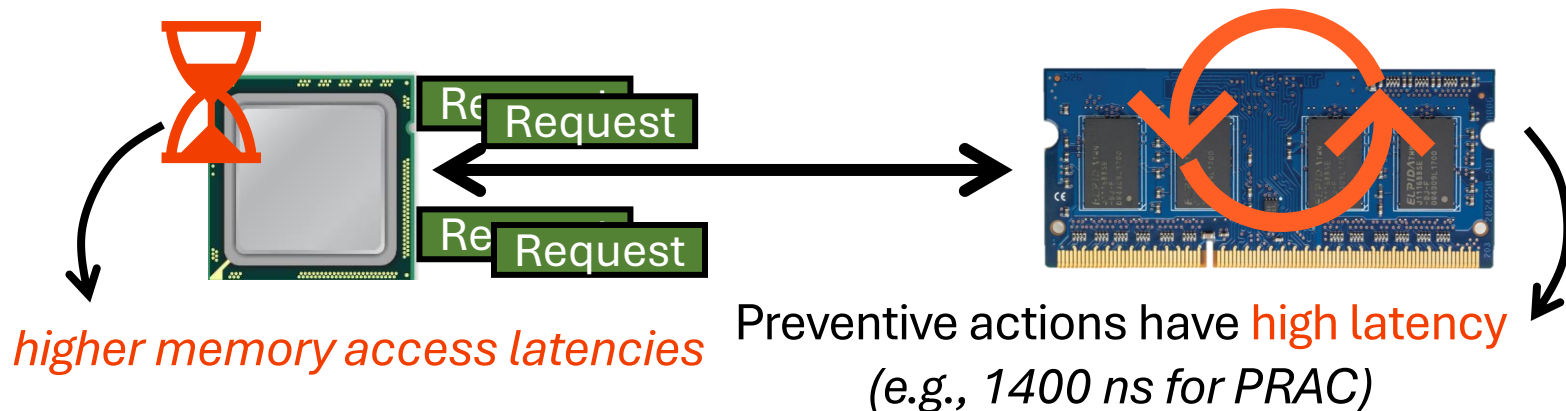
Talk Outline

- I. Motivation and Problem
- II. Key Observation**
- III. LeakyHammer and Case Studies
- IV. Mitigating LeakyHammer
- V. Conclusion

Key Observation (I)

- Preventive actions have **two fundamental features** that **enable covert channels and side channels**:

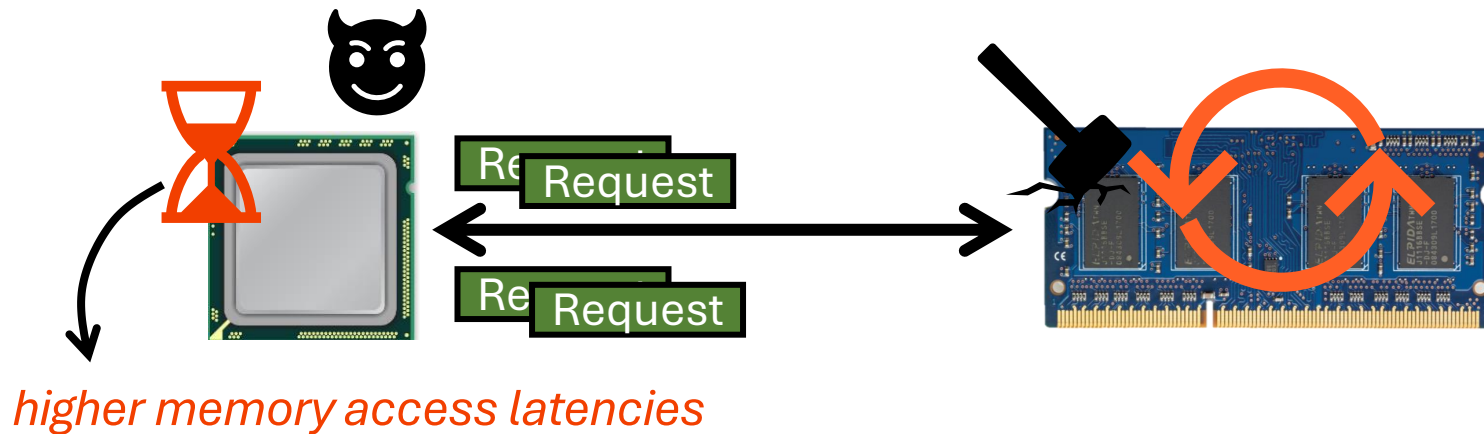
1 Preventive actions often result in **high memory access latencies** for regular memory requests



Key Observation (II)

- Preventive actions have **two fundamental features** that **enable covert channels and side channels**:

2 A user can leverage memory access patterns to **intentionally trigger a preventive action**



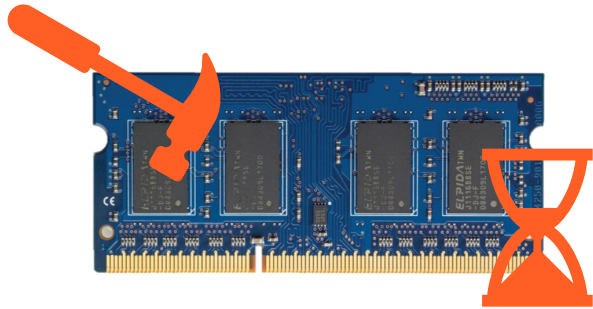
Talk Outline

- I. Motivation and Problem
- II. Key Observation
- III. LeakyHammer and Case Studies**
- IV. Mitigating LeakyHammer
- V. Conclusion



LeakyHammer

A new class of attacks that exploit RowHammer defense-induced memory latency differences



Deterministically and intentionally
impose observable latency overheads
on requests via preventive actions



Accurately **infer victims'**
memory access patterns
that result in preventive actions

LeakyHammer Case Studies

CS1: PRAC-Based Covert Channel Attack

CS2: RFM-Based Covert Channel Attack

CS3: PRAC-Based Website Fingerprinting

Evaluation Methodology

- **Environment:** System simulation using [gem5](#) [Lowe-Power+ '20] integrated with cycle-level DRAM simulator [Ramulator 2.0](#) [Kim+ CAL'15, Luo+ CAL'24]
- **System Configuration:**

Processor	1-, 2-, and 4-core, out-of-order, 3GHz clock frequency
DRAM	DDR5, 1 channel, 2 rank/channel, 8 bank groups, 4 banks/bank group, 128K rows/bank
Memory Ctrl.	64-entry read and write requests queues, Scheduling policy: FR-FCFS with a column cap of 16 Last-level cache 4 MiB (per-core), 16-way
PRAC	$N_{BO}=128$ activations, 4 RFMs per back-off
PRFM	$T_{RFM}=40$ activations

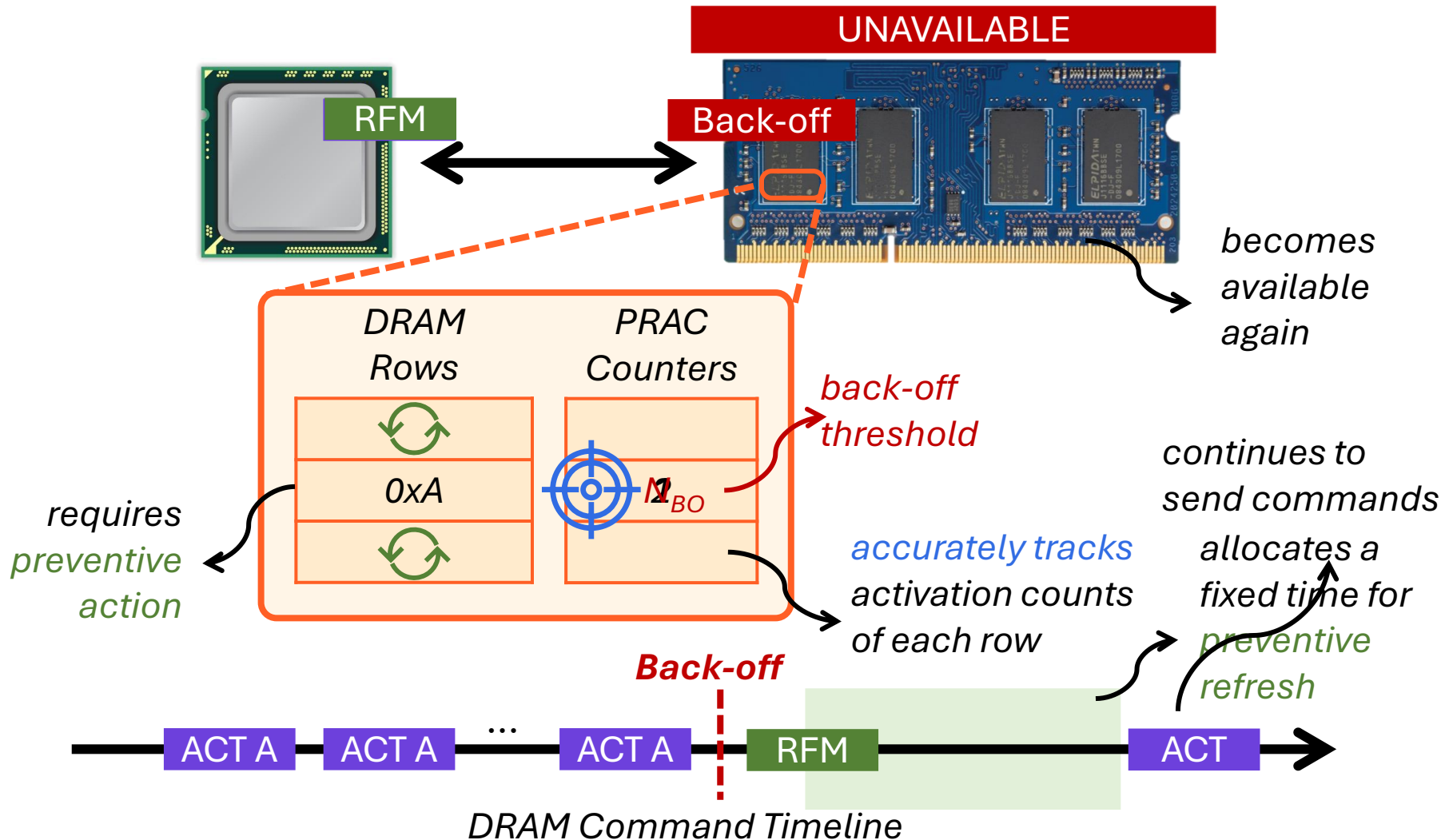
LeakyHammer Case Studies

CS1: PRAC-Based Covert Channel Attack

CS2: RFM-Based Covert Channel Attack

CS3: PRAC-Based Website Fingerprinting

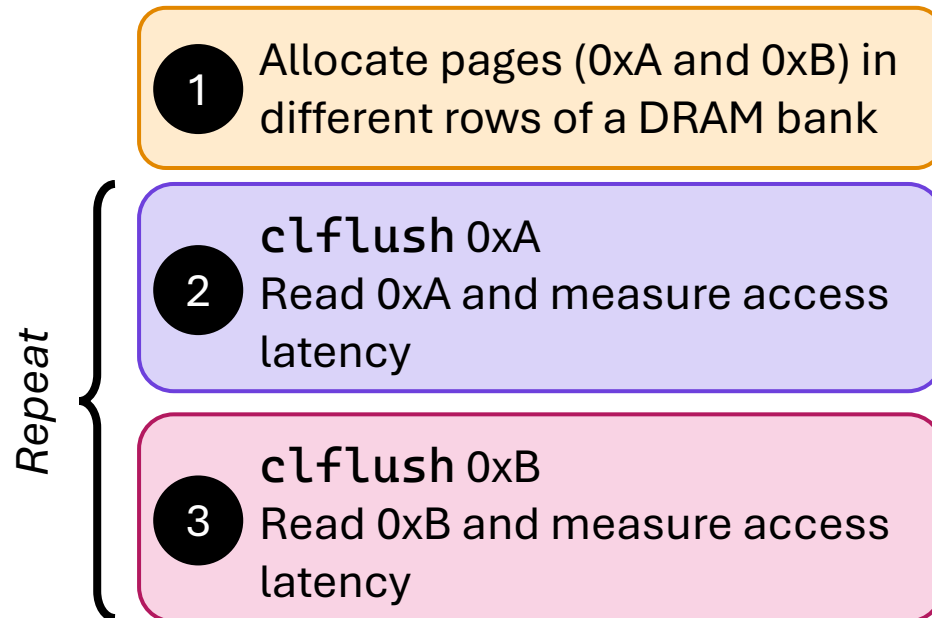
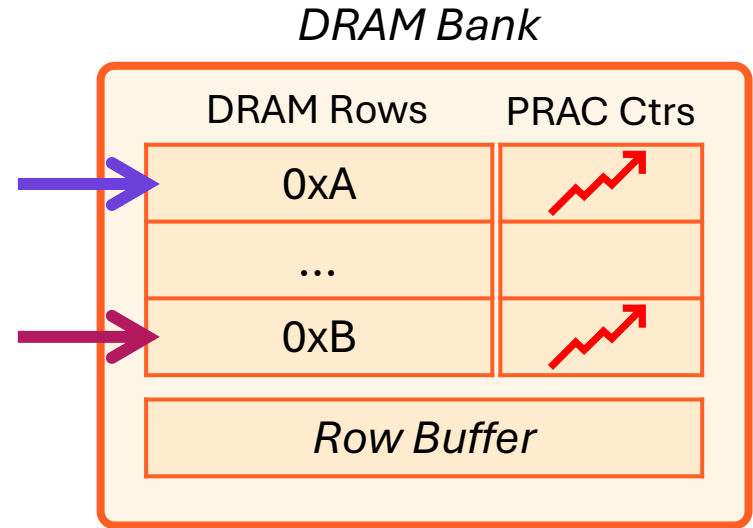
Per Row Activation Counting in DDR5



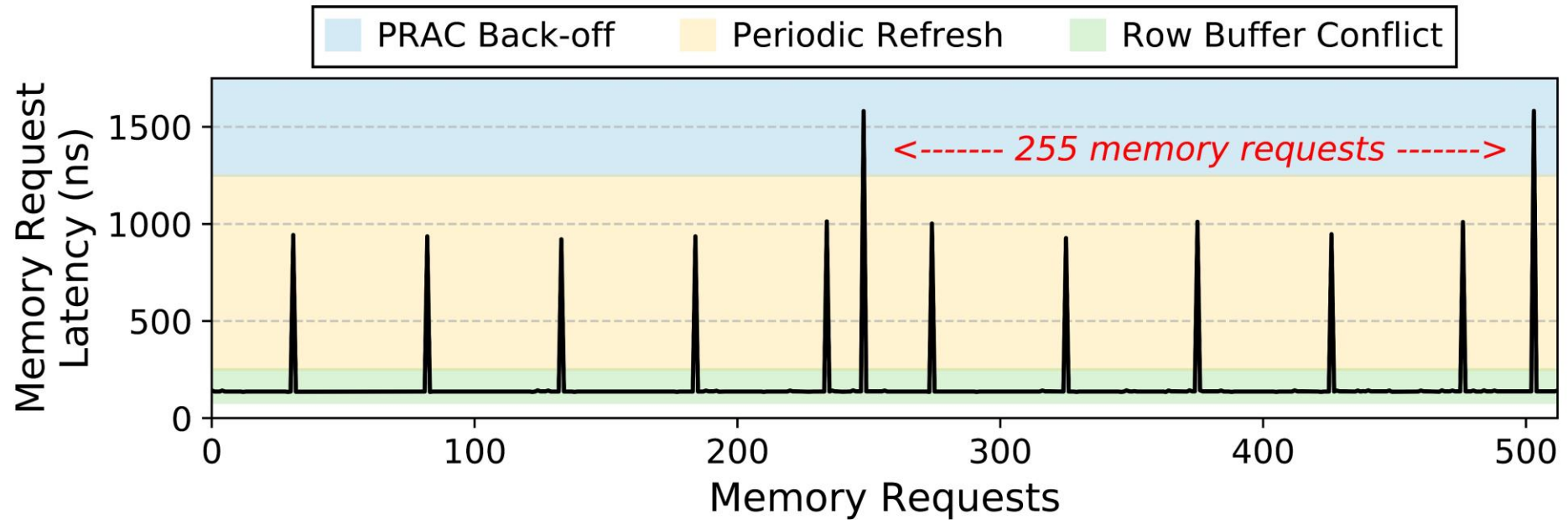
Observing PRAC-Induced Latencies (I)

We test if an application can:

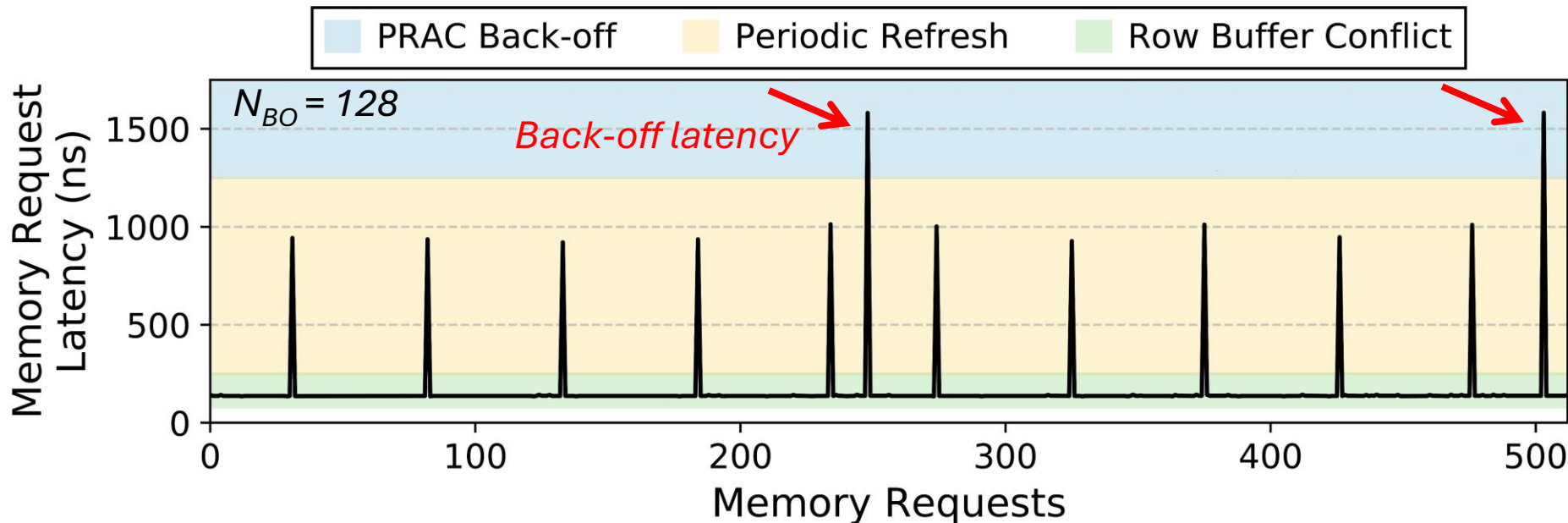
- intentionally trigger a PRAC back-off
- observe PRAC-induced higher memory access latencies



Observing PRAC-Induced Latencies (II)



Observing PRAC-Induced Latencies (II)



The observed back-off latency is **consistent with the specification** and **can be distinguished** from regular memory operations

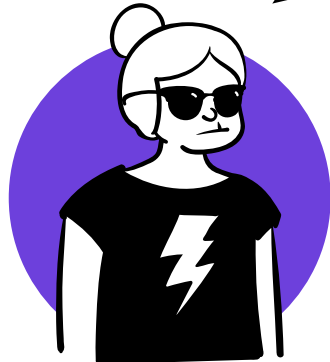
PRAC-Based Covert Channel: Attack Overview

10101001

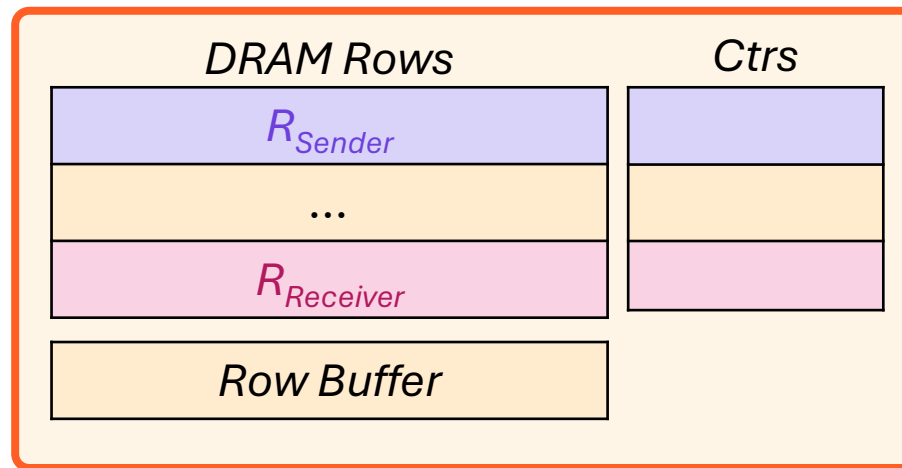
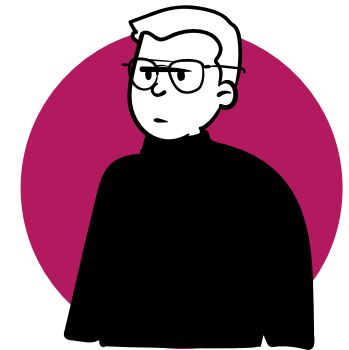
Encoding

logic-1: back-off
logic-0: no-back-off

DRAM Bank



Sender



Receiver

Only 1 bit will be transmitted



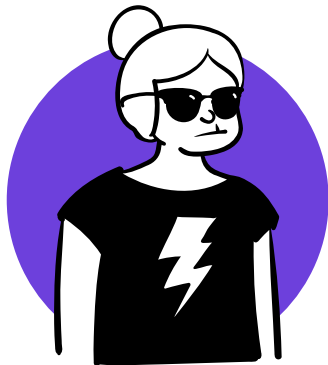
PRAC-Based Covert Channel: Attack Overview

Encoding

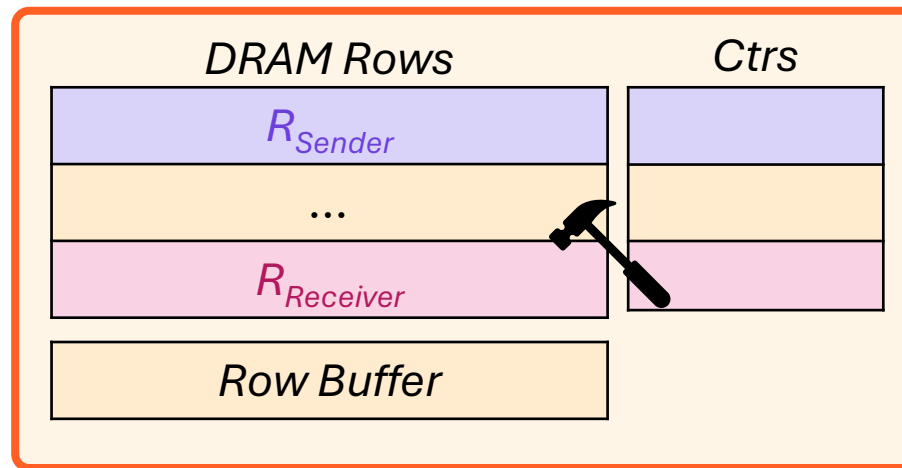
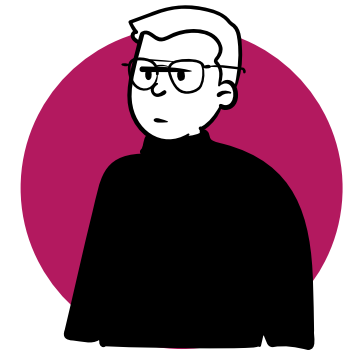
logic-1: back-off
logic-0: no-back-off

DRAM Bank

Loop:

 $\text{time}(\text{access } R_{\text{Receiver}})$ 

Sender



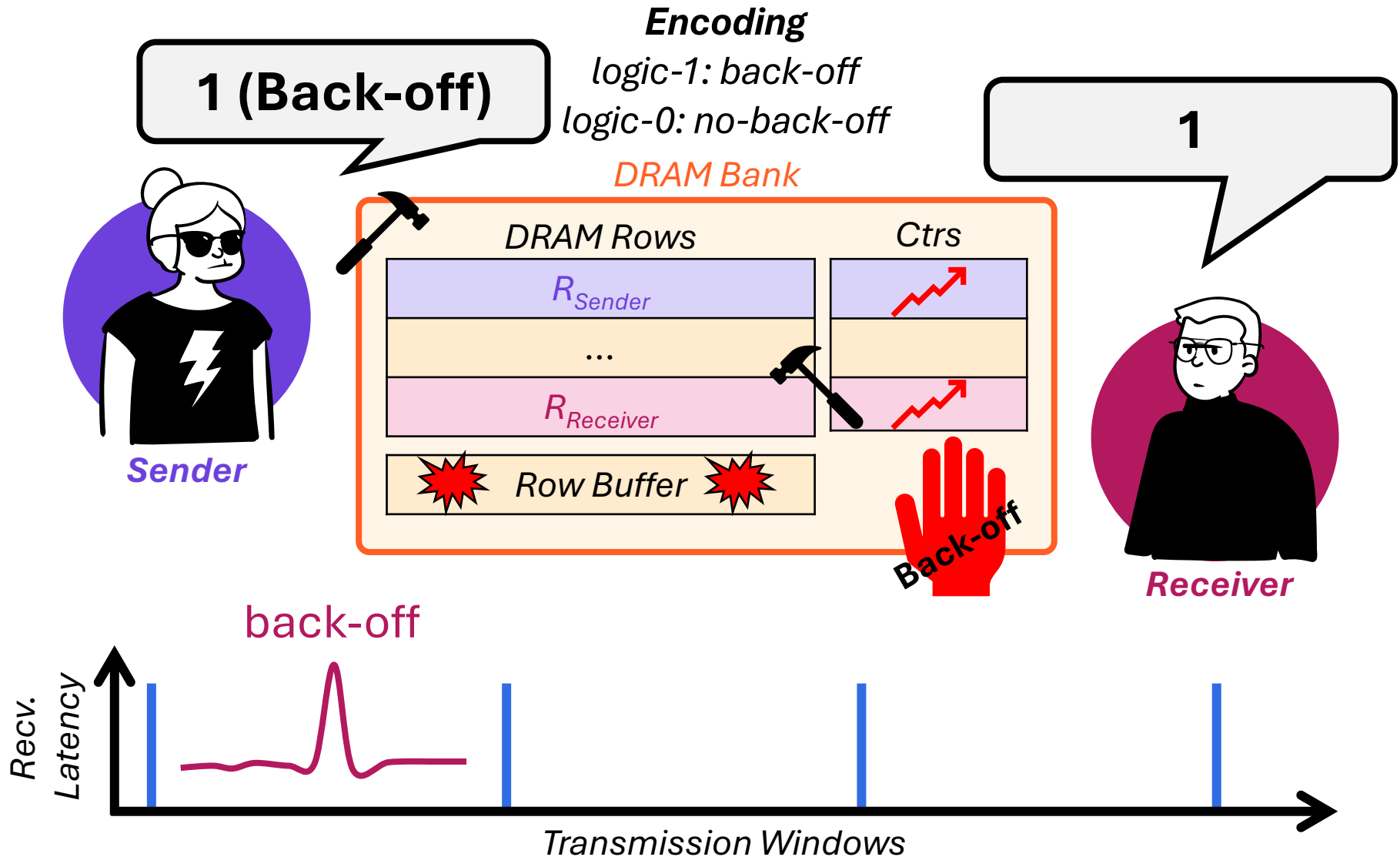
Receiver

Only 1 bit will be transmitted

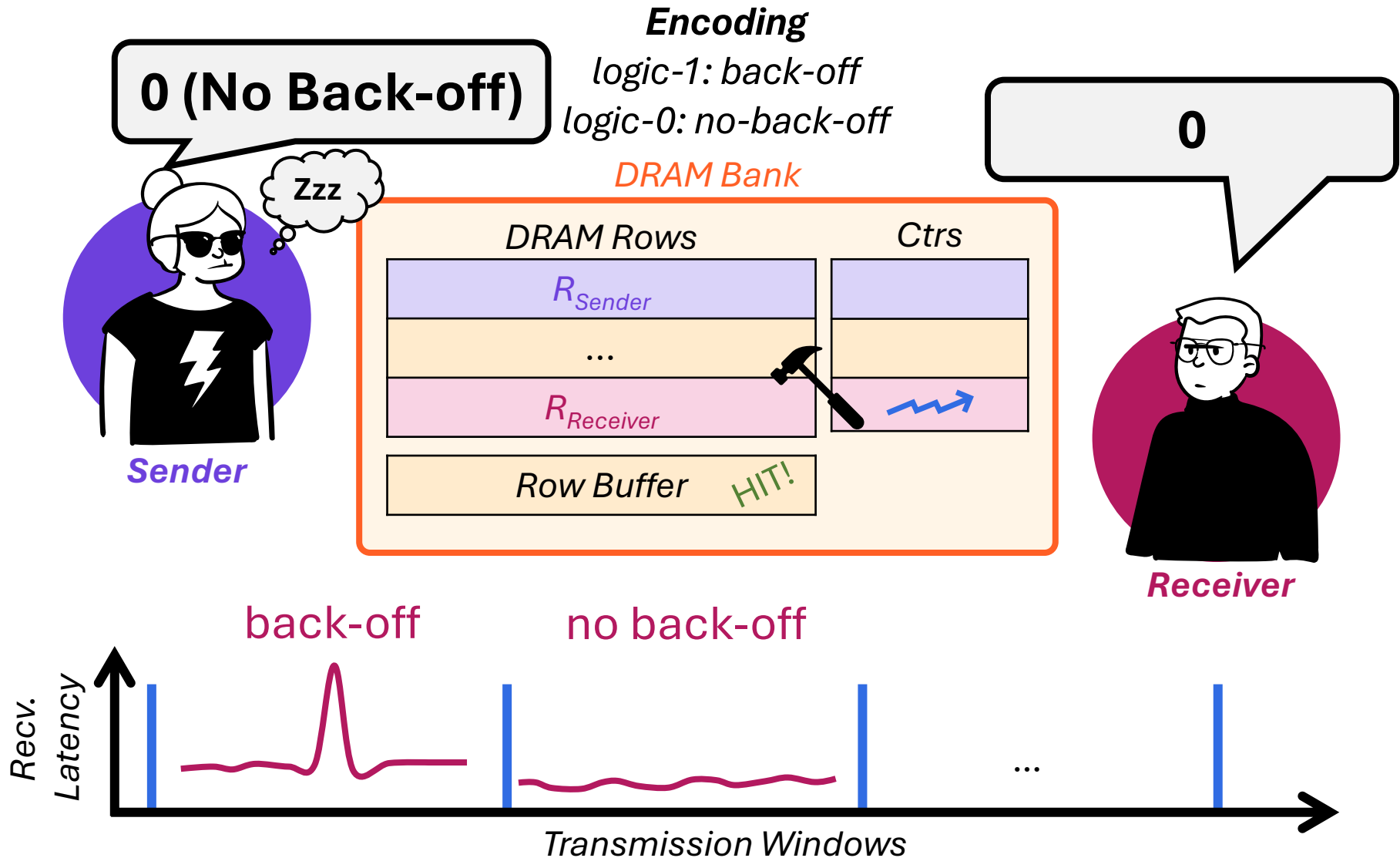
used to determine the bit value



PRAC-Based Covert Channel: Attack Overview

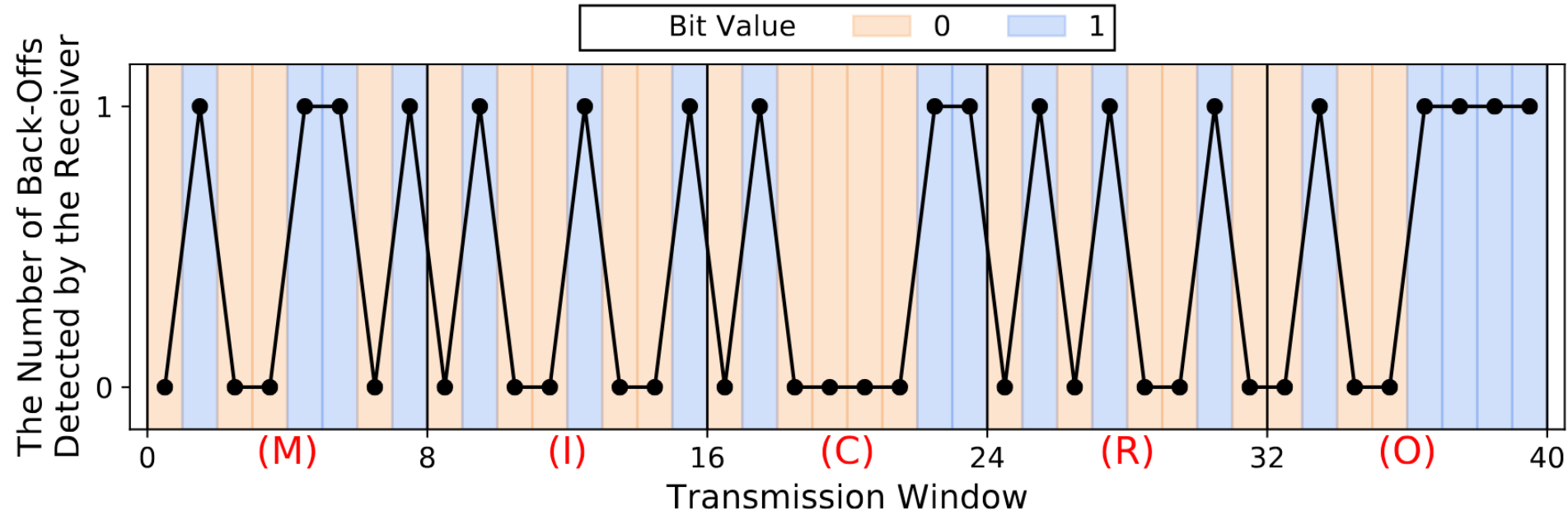


PRAC-Based Covert Channel: Attack Overview



PRAC-Based Covert Channel: Proof-of-Concept and Channel Capacity

- Validation with transmitting a 40-bit message

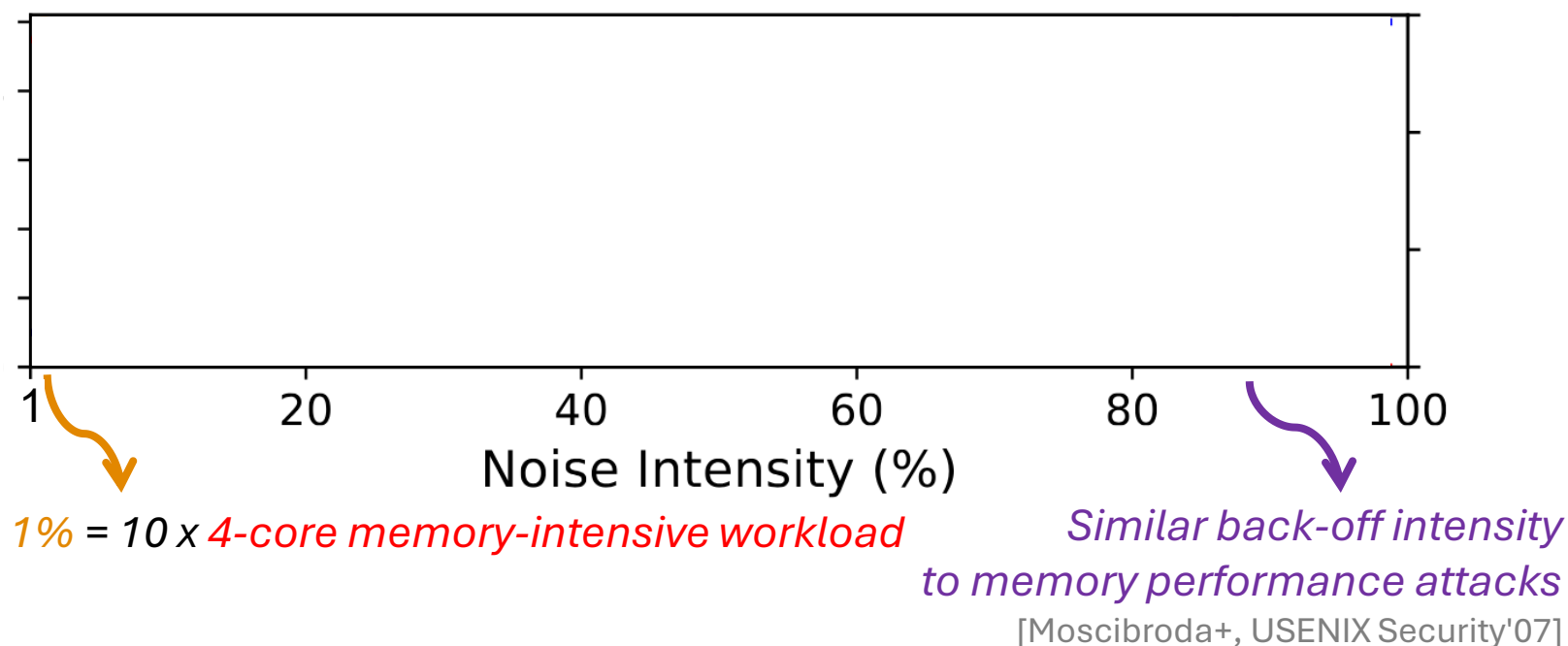


The receiver **successfully decodes** the sender's message

The attack achieves **39.0 Kbps raw bit rate**

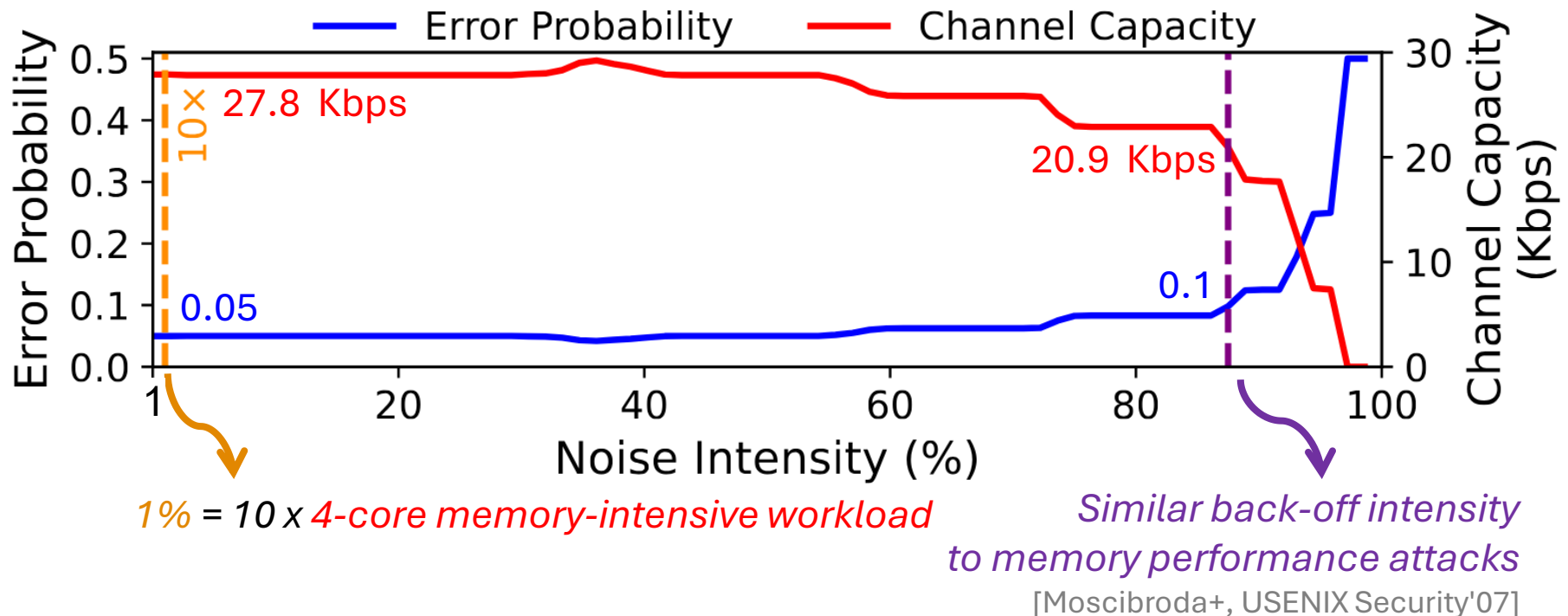
PRAC-Based Covert Channel: Noise Analysis

- We use a **microbenchmark** that causes back-offs with varying frequencies to simulate system noise



PRAC-Based Covert Channel: Noise Analysis

- We use a **microbenchmark** that causes back-offs with varying frequencies to simulate system noise



The PRAC-based covert channel attack **maintains its capacity** until **very high noise intensity values**

LeakyHammer Case Studies

CS1: PRAC-Based Covert Channel Attack

CS2: RFM-Based Covert Channel Attack

CS3: PRAC-Based Website Fingerprinting

RFM-Based Covert Channel Summary

We exploit an RFM-based RowHammer defense that tracks **the number of row activations per DRAM bank** as described in the DDR5 standard

We observe that applications **can intentionally trigger an RFM command** and can detect it by **observing the high memory latency**

The attack successfully transmits the message

Due to per DRAM bank activation counting, applications can trigger RFM more easily, making the channel **more susceptible to noise**

RFM-Based Covert Channel Summary



Understanding and Mitigating Covert Channel and Side Channel Vulnerabilities Introduced by RowHammer Defenses

F. Nisa Bostancı¹ Oğuzhan Canpolat^{1,2} Ataberk Olgun¹ İsmail Emir Yüksel¹
Konstantinos Kanellopoulos¹ Mohammad Sadrosadati¹ A. Giray Yağlıkçı^{1,3} Onur Mutlu¹
¹ETH Zürich ²TOBB ETÜ ³CISPA

DRAM chips are increasingly vulnerable to read disturbance phenomena (e.g., RowHammer and RowPress), where repeatedly accessing or keeping open a DRAM row causes bitflips in nearby rows, due to DRAM density scaling. Attackers can exploit RowHammer bitflips in real systems to compromise security, which has motivated many prior works on RowHammer defenses. To enable such defenses, recent DDR specifications introduce new defense frameworks (e.g., PRAC and RFM). For robust (i.e., secure, safe, and reliable) operation, it is critical to analyze security implications of widely-adopted RowHam-

rows) [1–14]. RowHammer [1, 2, 4, 14] is a type of read disturbance phenomenon, where a victim row can experience bitflips when at least one nearby row is hammered more times than a threshold, called the RowHammer threshold (N_{RH}). Modern DRAM chips become more vulnerable to RowHammer as DRAM technology node size becomes smaller [1, 2, 4, 14–19]. RowPress [13] is another example DRAM read disturbance phenomenon that amplifies bitflips by keeping the aggressor row open for longer, thereby causing more disturbance with each activation. Prior works show that attackers can leverage

<https://arxiv.org/abs/2503.17891>

Oct 2025

LeakyHammer Case Studies

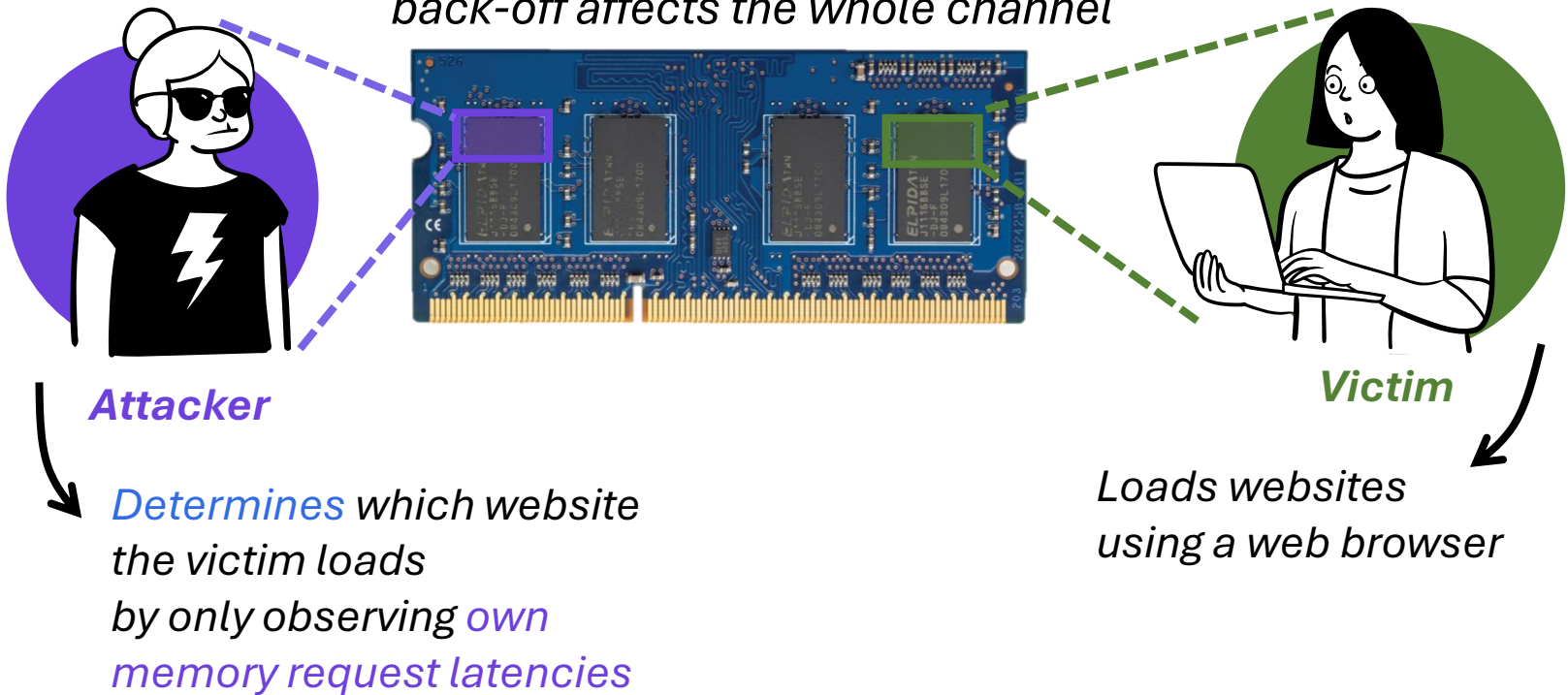
CS1: PRAC-Based Covert Channel Attack

CS2: RFM-Based Covert Channel Attack

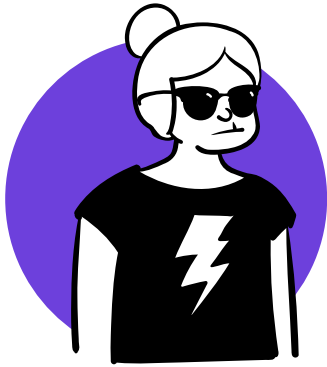
CS3: PRAC-Based Website Fingerprinting

PRAC-Based Website Fingerprinting Attack

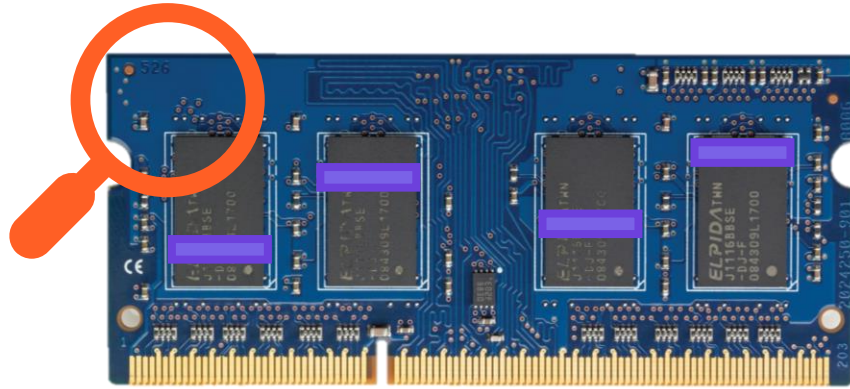
*No need to colocate in the same bank as
back-off affects the whole channel*



How to Fingerprint

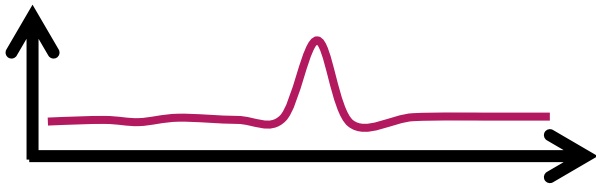


Attacker



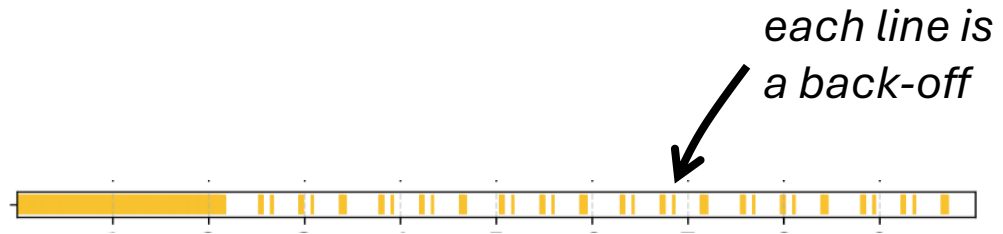
1

allocate multiple *test rows* to avoid triggering back-off while observing



2

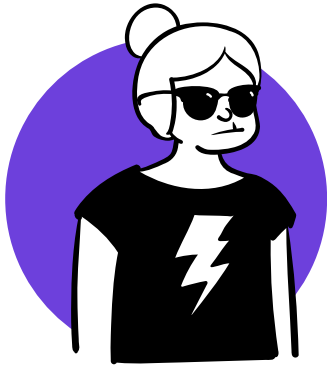
access *test rows* continuously and *measure access latencies* to *detect back-offs*



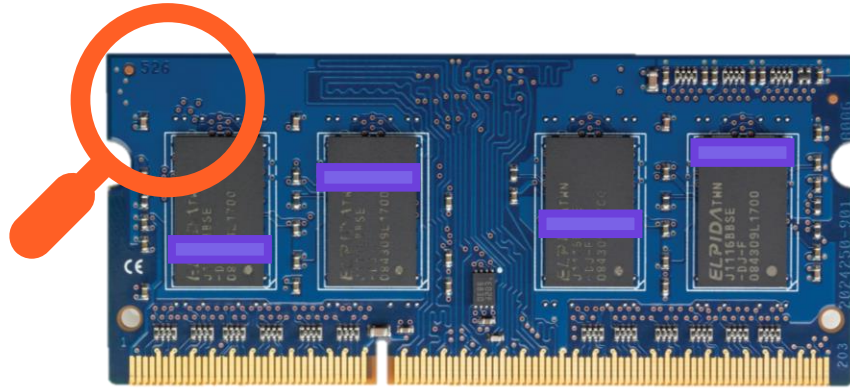
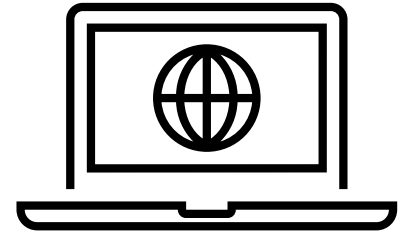
3

create *fingerprint* by only using the timestamps of *back-offs*

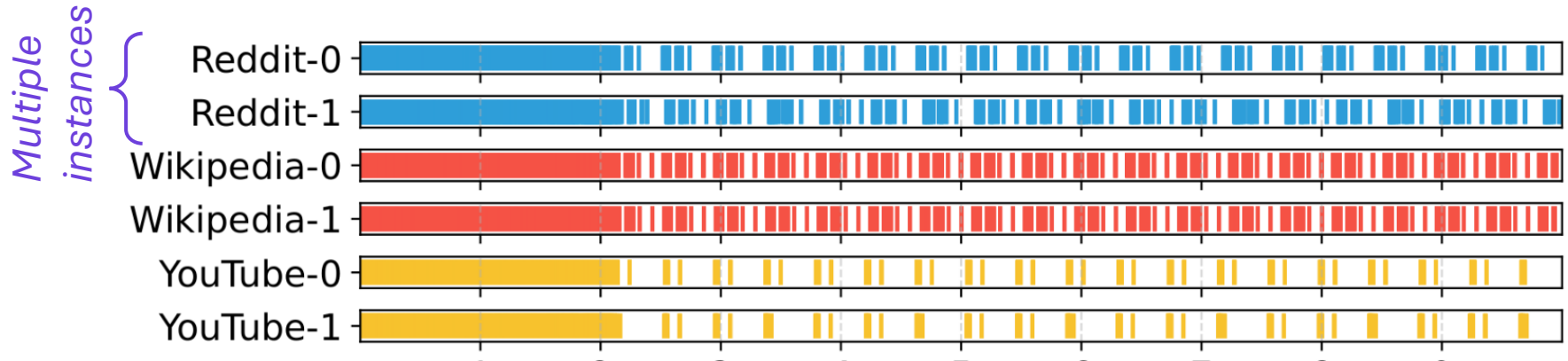
Preparing for the Attack: Creating a Database



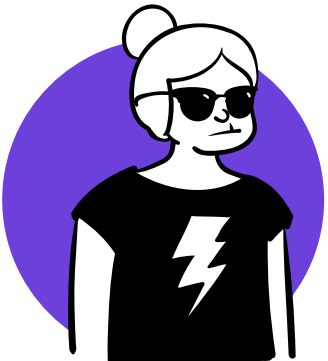
Attacker



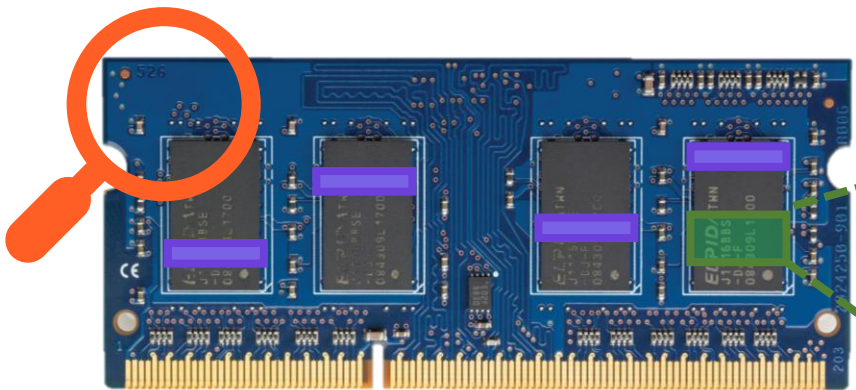
*fingerprint many websites
and create a fingerprint database*



Launching the Attack

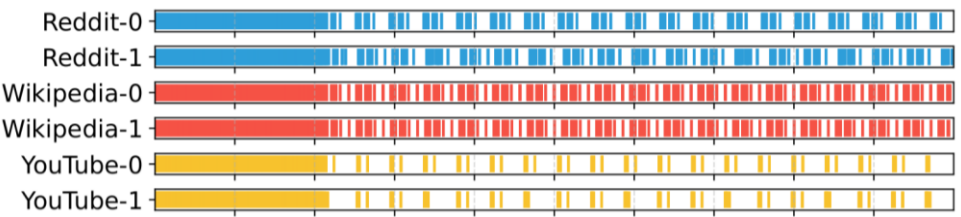


Attacker



Victim

1 Get a fingerprint when the victim loads a website



2 compare with the database and determine the website

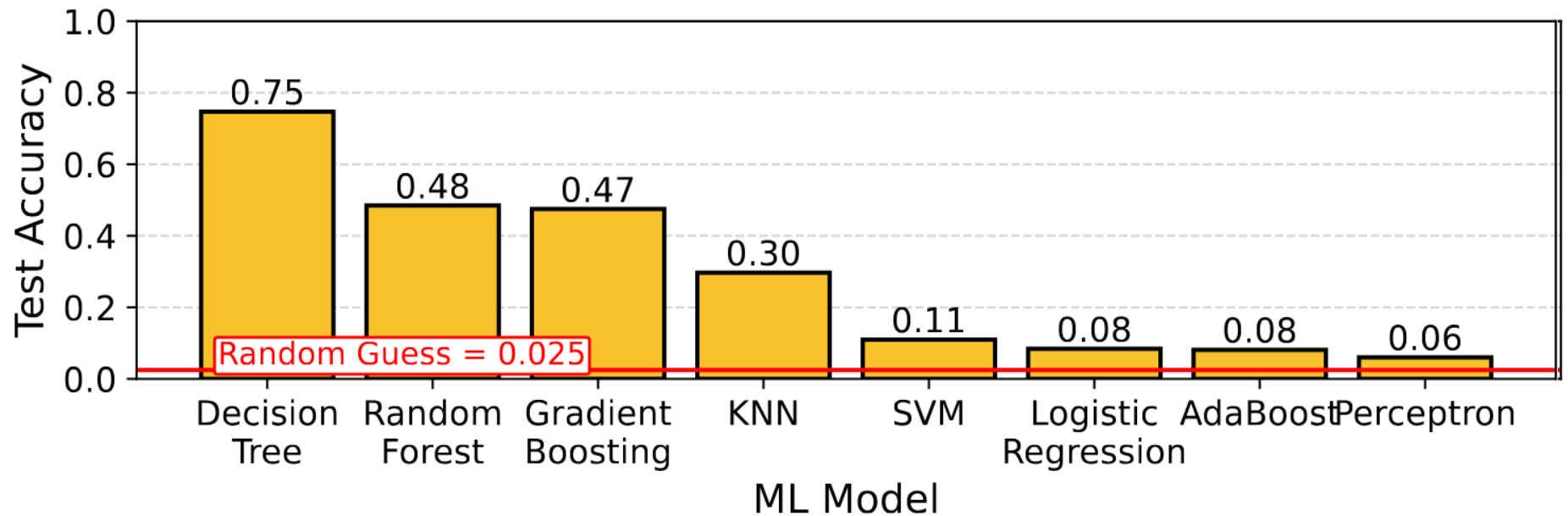


Website: YouTube

PRAC-Based Website Fingerprinting: Evaluation

- We collect **back-off fingerprints** of a website by loading it with a browser and getting a back-off trace with the attacker routine
 - **40 most-visited websites** from Semrush and Similarweb top sites
- **A machine learning approach** to classify websites:
 - Simple and lightweight machine learning models (e.g., decision tree, random forest, gradient boosting, k-nearest neighbors)

PRAC-Based Website Fingerprinting: Test Accuracy



Using back-off-based fingerprints, **classical machine learning models** can **classify websites with up to 75% accuracy**

Talk Outline

- I. Motivation and Problem
- II. Key Observation
- III. LeakyHammer and Case Studies
- IV. Mitigating LeakyHammer**
- V. Conclusion

Mitigating LeakyHammer

- We propose **three countermeasures** against LeakyHammer:

Fixed-Rate RFM (FR-RFM)

eliminates LeakyHammer by **decoupling preventive actions** from **memory access patterns** of running applications

Randomly Initialized Activation Counters (RIAC)

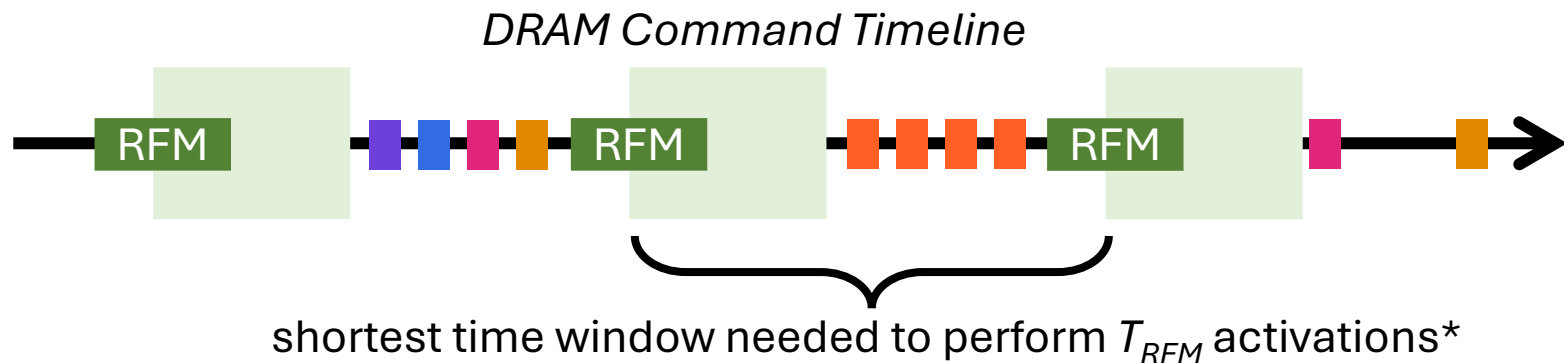
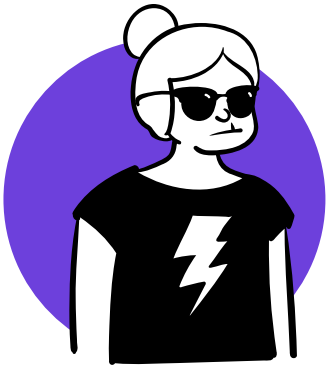
reduces channel capacity **with a randomization method**

Bank-Level PRAC (PRAC-Bank)

performs bank-level preventive actions to limit the attack scope to that of existing DRAM-based attacks

Fixed-Rate RFM (FR-RFM)

- Periodically performs **a preventive action**,
 - independently** from **the memory access patterns**

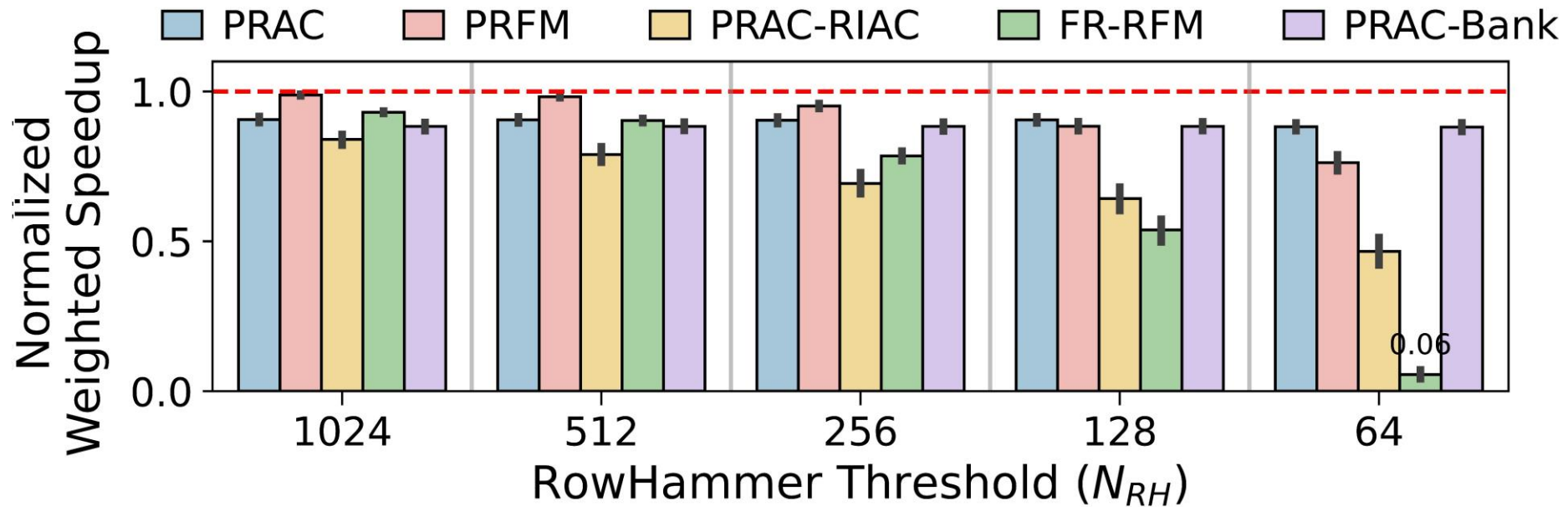


- **A sender** cannot cause preventive actions **intentionally** to send messages
- **An attacker** cannot observe **a victim's memory access patterns** that **lead to a preventive action**

* Security analysis available in the paper

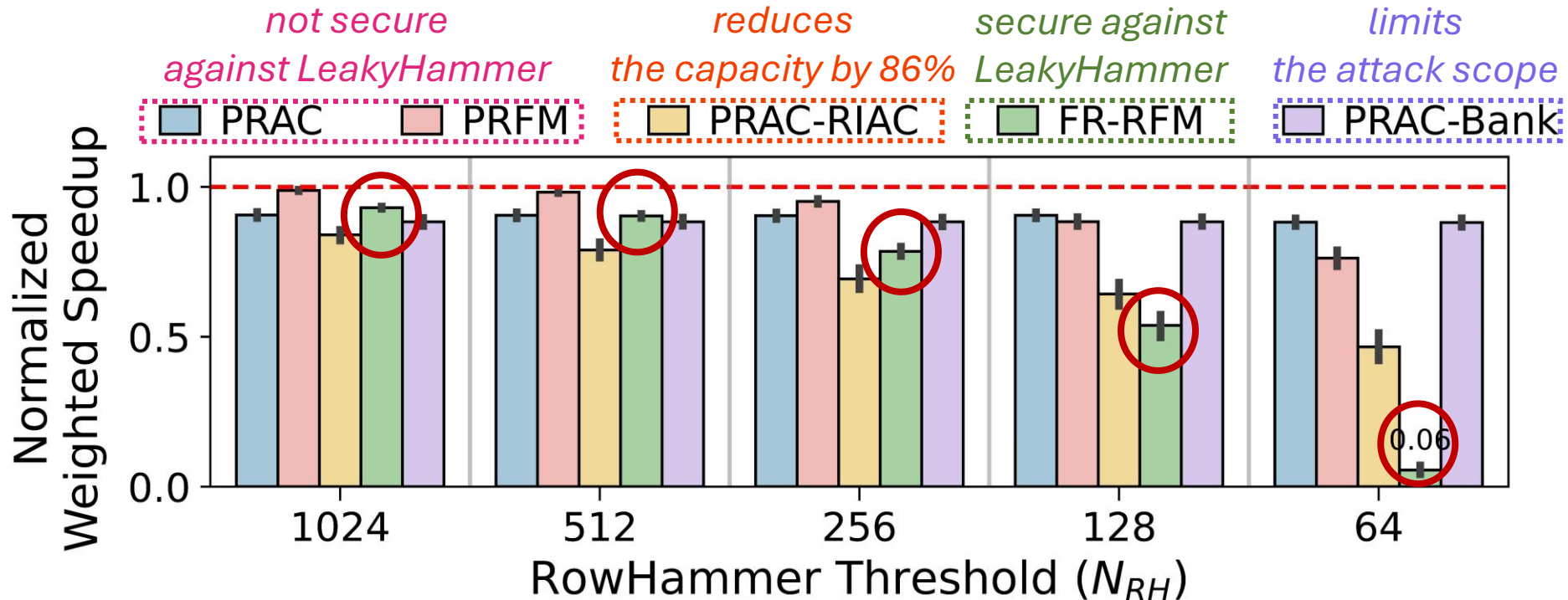
Evaluating LeakyHammer Countermeasures

- 60 four-core workloads: SPEC2017 and SPEC2006



Evaluating LeakyHammer Countermeasures

- 60 four-core workloads: SPEC2017 and SPEC2006



FR-RFM mitigates LeakyHammer and performs similar to PRAC and PRFM for near-future N_{RH} values (e.g., 1024)

As N_{RH} decreases, other countermeasures have lower performance overheads

More in the Paper

- More details of LeakyHammer
 - Operational details on case studies (e.g., RFM-based covert channel)
 - Multibit LeakyHammer covert channels
 - LeakyHammer countermeasures
- More results and analyses
 - LeakyHammer with other RowHammer mitigations
 - Impact of real workloads on LeakyHammer
 - Sensitivity of PRAC-based covert channel
 - Impact of RFM commands per back-off, RowHammer defense latency, larger cache hierarchy and data prefetchers
 - Comparison against existing cache and memory-based timing channels



Understanding and Mitigating Covert Channel and Side Channel Vulnerabilities Introduced by RowHammer Defenses

F. Nisa Bostancı¹ Oğuzhan Canpolat^{1,2} Ataberk Olgun¹ İsmail Emir Yüksel¹
Konstantinos Kanellopoulos¹ Mohammad Sadrosadati¹ A. Giray Yağlıkçı^{1,3} Onur Mutlu¹
¹ETH Zürich ²TOBB ETÜ ³CISPA

DRAM chips are increasingly vulnerable to read disturbance phenomena (e.g., RowHammer and RowPress), where repeatedly accessing or keeping open a DRAM row causes bitflips in nearby rows, due to DRAM density scaling. Attackers can exploit RowHammer bitflips in real systems to compromise security, which has motivated many prior works on RowHammer defenses. To enable such defenses, recent DDR specifications introduce new defense frameworks (e.g., PRAC and RFM). For robust (i.e., secure, safe, and reliable) operation, it is critical to analyze security implications of widely-adopted RowHam-

rows) [1–14]. RowHammer [1, 2, 4, 14] is a type of read disturbance phenomenon, where a victim row can experience bitflips when at least one nearby row is hammered more times than a threshold, called the RowHammer threshold (N_{RH}). Modern DRAM chips become more vulnerable to RowHammer as DRAM technology node size becomes smaller [1, 2, 4, 14–19]. RowPress [13] is another example DRAM read disturbance phenomenon that amplifies bitflips by keeping the aggressor row open for longer, thereby causing more disturbance with each activation. Prior works show that attackers can leverage

Oct 2025

<https://arxiv.org/abs/2503.17891>

LeakyHammer is Open Source and Artifact Evaluated



The screenshot shows the GitHub repository page for LeakyHammer. At the top, there are three badges: "Artifact Available V1.1" (green), "Artifact Evaluated Functional V1.1" (red), and "Results Reproduced V1.1" (blue). The repository name "LeakyHammer" is displayed with a "Public" label. Below the repository name, there are buttons for "Edit Pins", "Watch", "Fork", and "Star". The main content area shows the file list for the "master" branch, including files like "gem5", ".gitignore", "Dockerfile", "README.md", "ae_install_requirements.sh", "container_setup.sh", "native_setup.sh", and "requirements.txt". The right sidebar contains an "About" section with a description of the repository and a "Releases" section.

LeakyHammer Public

master 1 Branch 0 Tags

Go to file angular Snap Add file Code

nisabostanci Initial commit b2ef9ad · now 1 Commit

File	Commit	Time
gem5	Initial commit	now
.gitignore	Initial commit	now
Dockerfile	Initial commit	now
README.md	Initial commit	now
ae_install_requirements.sh	Initial commit	now
container_setup.sh	Initial commit	now
native_setup.sh	Initial commit	now
requirements.txt	Initial commit	now

About

This repository contains the source code of LeakyHammer, our MICRO'25 paper. LeakyHammer is a new class of attacks that leverage the RowHammer mitigation-induced memory latency differences to establish communication channels and leak secrets.

Readme Activity Custom properties 0 stars 0 watching 0 forks Report repository

Releases

<https://github.com/CMU-SAFARI/LeakyHammer>

Talk Outline

- I. Motivation and Problem
- II. Key Observation
- III. LeakyHammer and Case Studies
- IV. Mitigating LeakyHammer
- V. Conclusion**

Conclusion

We introduce **LeakyHammer**, a new class of timing attacks that **exploit RowHammer defense-induced memory latency differences**

We systematically analyze **two latest industry defenses to RowHammer**

- Build **two covert communication channels**
- Demonstrate **a website fingerprinting attack** that **identifies visited websites** with high accuracy

We propose and evaluate **three countermeasures**

- **Completely mitigating LeakyHammer with low performance overhead in near-future DRAM-based systems**
- **In systems that are highly RowHammer-vulnerable, countermeasures that reduce channel capacity have lower performance overhead**



arXiv



GitHub



Understanding and Mitigating Covert Channel and Side Channel Vulnerabilities Introduced by RowHammer Defenses

F. Nisa Bostancı

O. Canpolat A. Olgun I. E. Yüksel K. Kanellopoulos
M. Sadrosadati A. G. Yağlıkçı O. Mutlu

ETH zürich

SAFARI

 **CISPA**
HELMHOLTZ CENTER FOR
INFORMATION SECURITY