June 2023

# DramaQueen: Revisiting Side Channels in DRAM

**Ben Gras (Intel Corporation)**
**Victor van der Veen (Qualcomm Technologies Inc.)**

**DRAMSec23**

# Legal disclaimers

As to the opinions and positions in this document that the authors express or to which the authors contributed, they are those of the authors and do not represent the views of any current or previous employer, including Intel Corporation or its affiliates.

No product or component can be absolutely secure.

# Can we forget about classical side channels?

- Classical: they exploit secret-dependent accesses, so.. “insecure” and “wrong”

```
void
mul_point_scalar (point result,
   scalar scalar, point point)
{
  for (j=nbits-1; j >= 0; j--) {
   ec_duplicate_point (result);
   if (bit_on (scalar, j))
    ec_add_point(result,point);
  }
}
```

- But: sometimes we want to use an “insecure” implementation
- And: sometimes we want defense in depth (e.g.: Memjam)

# Overview

Straw-man Cache Attack Immunity Framework

DRAMA Side Channel Signal

Evaluation of DRAMA Side Channel Signal potential

# Overview

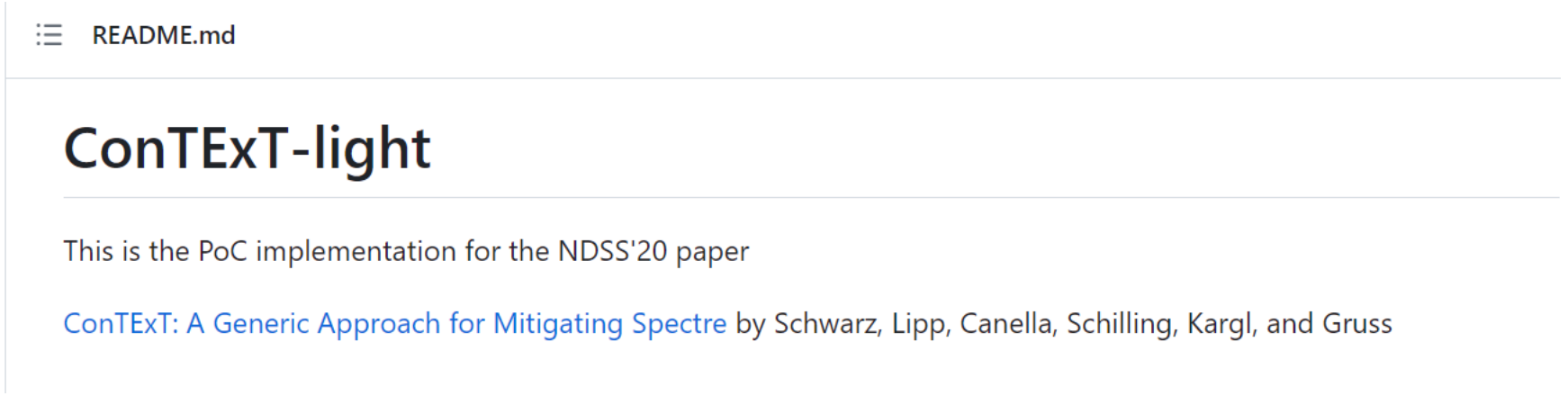
**Straw-man Cache Attack Immunity Framework**

DRAMA Side Channel Signal

Evaluation of DRAMA Side Channel Signal potential

# Can we forget about classical side channels?

- Idea: Let security-critical code bypass caches by mapping the code and data uncacheable: no LLC, etc. exposure. To do this, we can re-use the ConTExT spectre mitigation framework:


README.md

ConTExT-light

This is the PoC implementation for the NDSS'20 paper

ConTExT: A Generic Approach for Mitigating Spectre by Schwarz, Lipp, Canella, Schilling, Kargl, and Gruss

- And: Disable SMT (no L1, L2, TLB, etc. exposure)
- Can we forget about classical side channels now?
- This is the **straw-man cache attack immunity framework**

# Overview

Straw-man Cache Attack Immunity Framework

**DRAMA Side Channel Signal**

Evaluation of DRAMA Side Channel Signal potential

# Reading from a bank

# Reading from a bank

1. Activate row N (ACT)

   Data is moved to row buffer

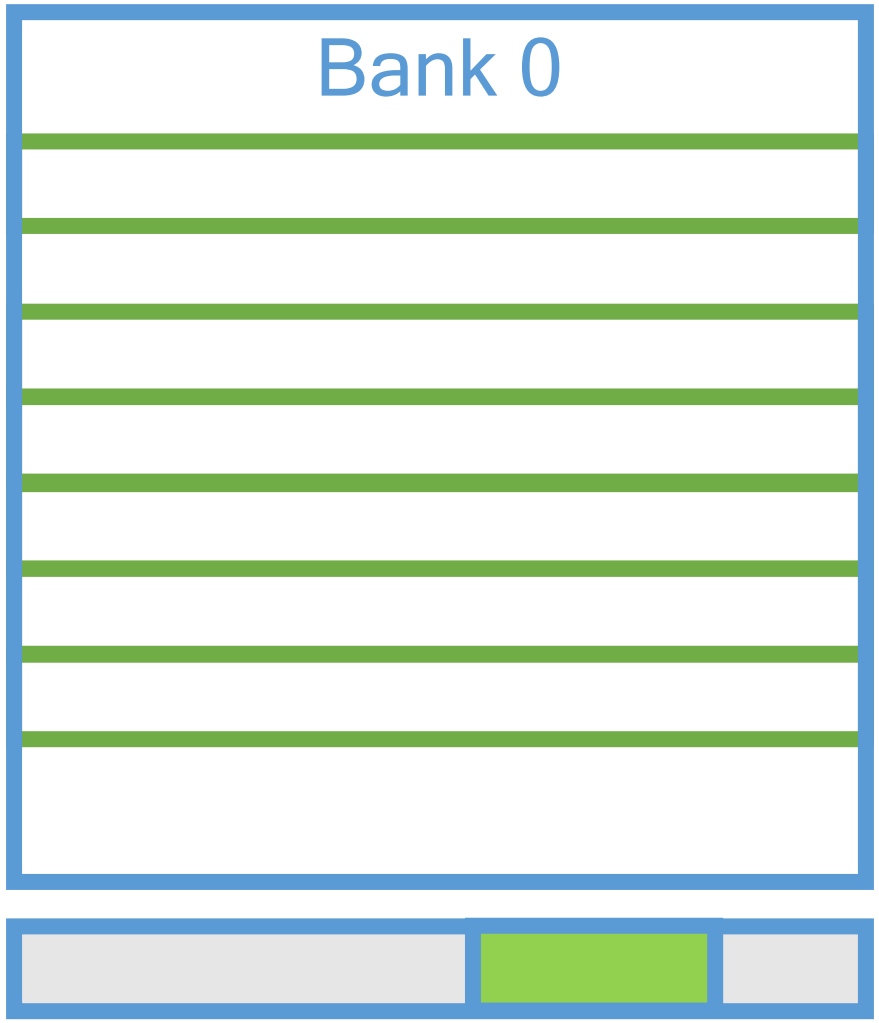
# Reading from a bank

1. Activate row N (ACT)

   Data is moved to row buffer

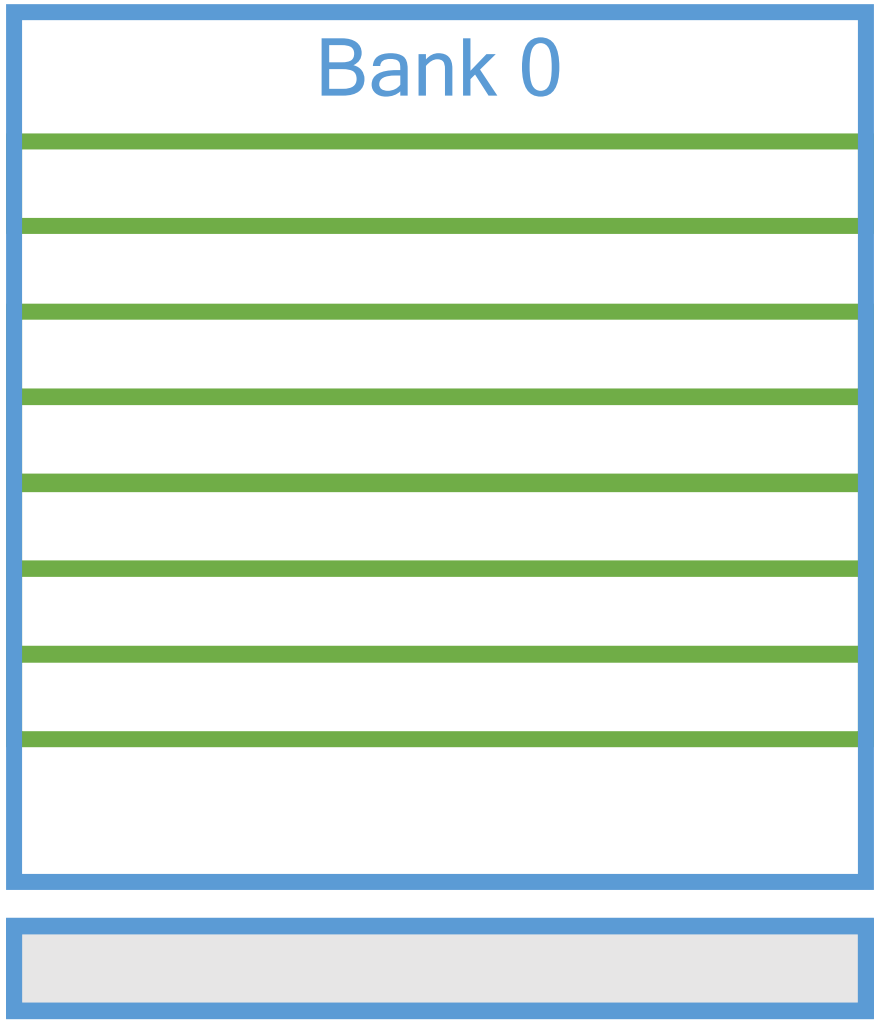
2. Read col Y (RD)

   Bytes are sent over the bus



# Reading from a bank

1. Activate row N (ACT)
   Data is moved to row buffer
2. Read col Y (RD)
   Bytes are sent over the bus
3. (auto) Precharge (PRE)
   Write row buffer back to row

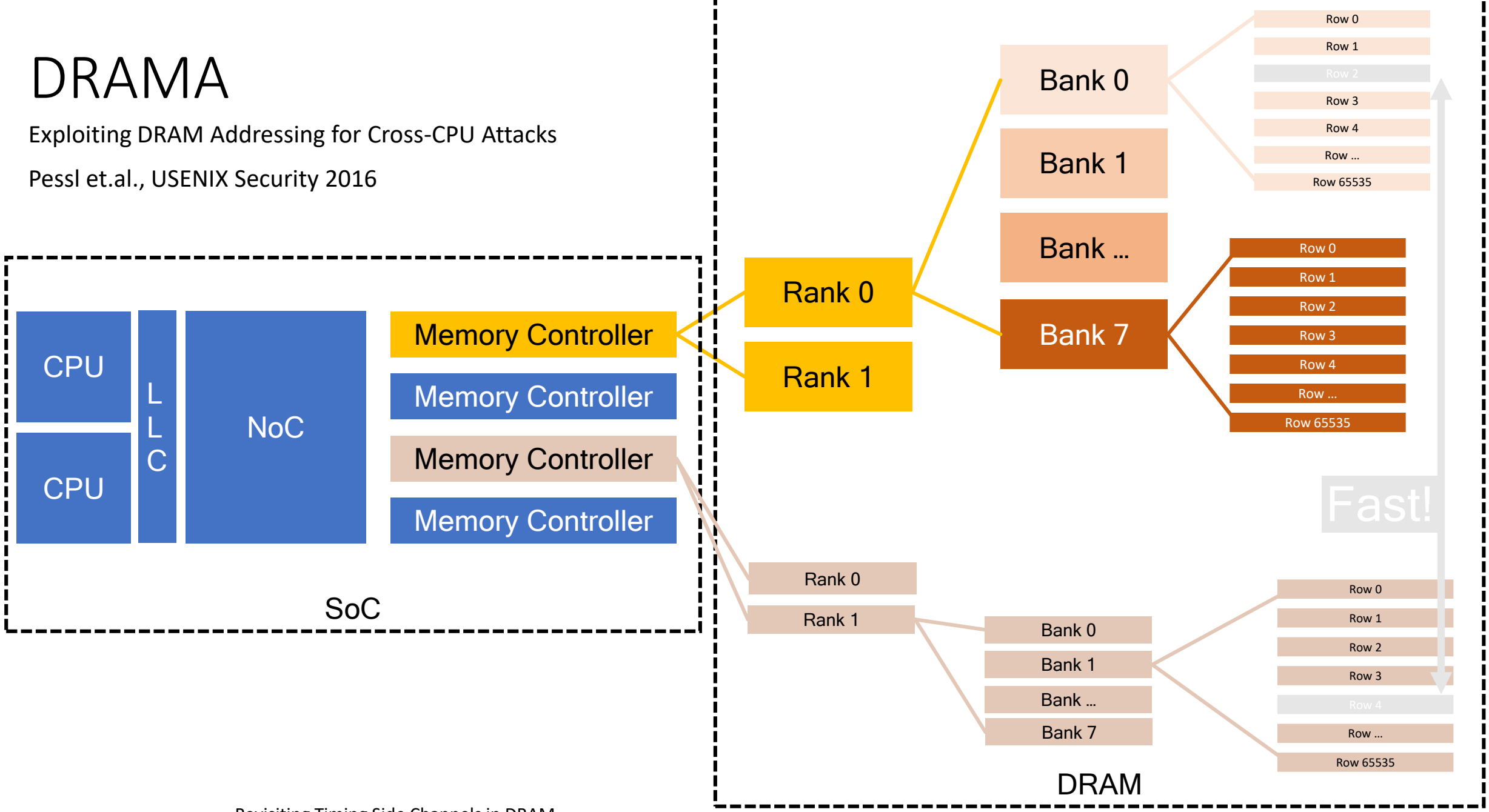



This takes time. Next access is preferably in a different bank.

# DRAMA

Exploiting DRAM Addressing for Cross-CPU Attacks

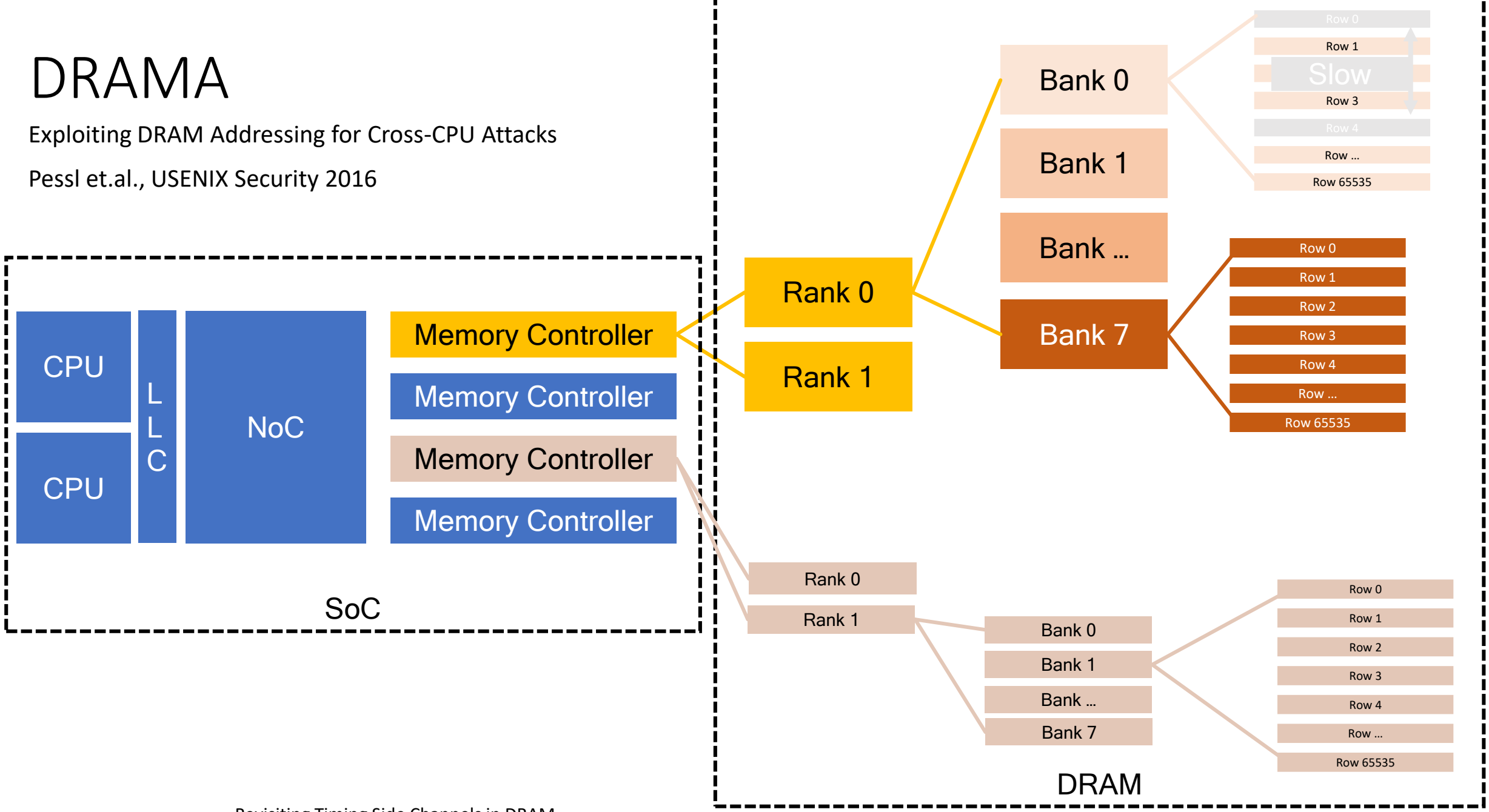
Pessl et.al., USENIX Security 2016



# DRAMA

Exploiting DRAM Addressing for Cross-CPU Attacks

Pessl et.al., USENIX Security 2016

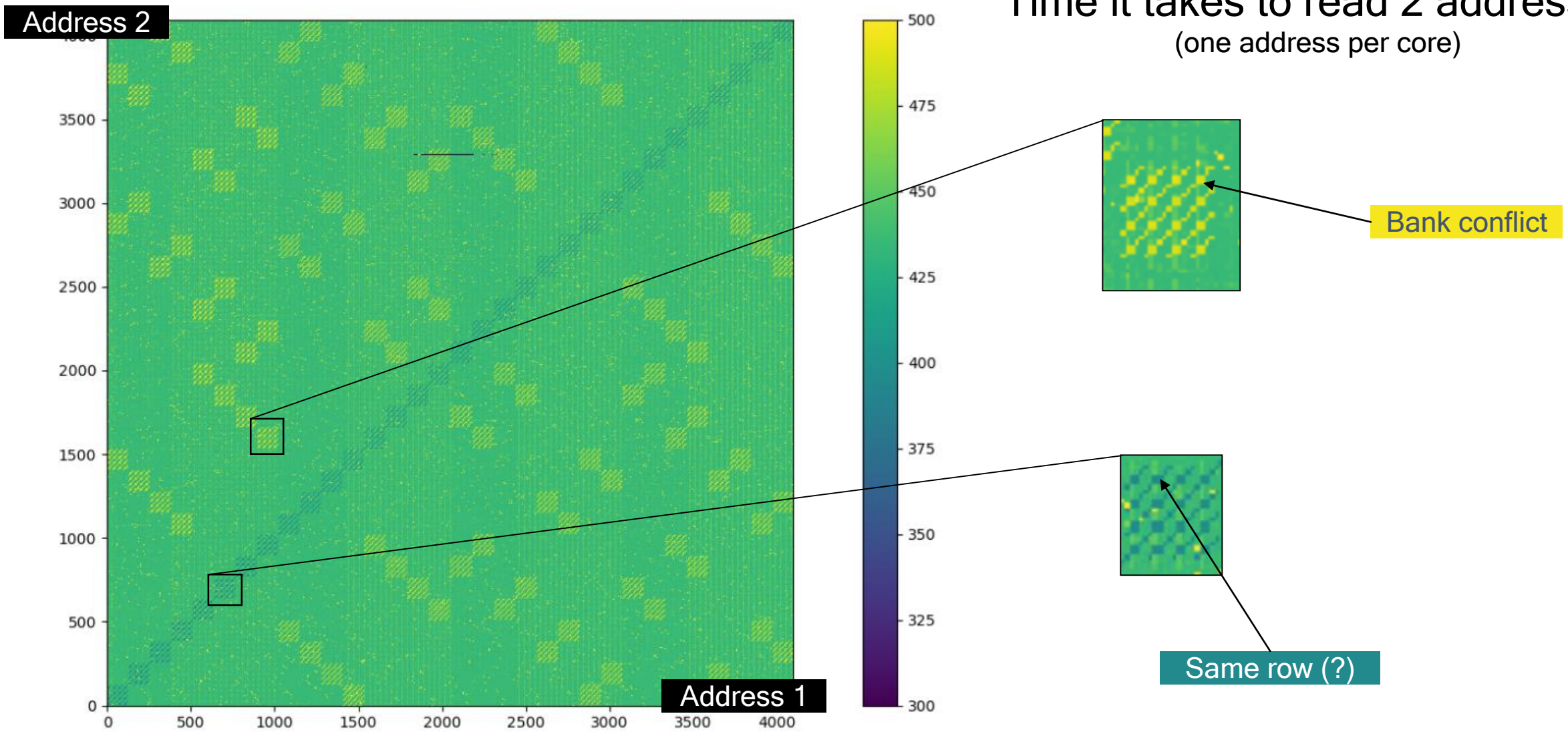


# Page Hits and Misses (bank / row conflicts)

| | Same bank, different rows | | Different banks | |
|---|---|---|---|---|
| | Address 0 (channel 0) | Address 1 (channel 0) | Address 0 (channel 0) | Address 1 (channel 1) |
| T0 | ACT bank p, row x | | ACT bank p, row x | ACT bank q, row y |
| T1 | | | | |
| T2 | Read | | Read | Read |
| T3 | | | | |
| T4 | PRE bank p | | PRE bank p | PRE bank q |
| T5 | | | | |
| T6 | | ACT bank p, row y | | |
| T7 | | | | |
| T8 | | Read | | |
| T9 | | | | |
| T10 | | PRE bank p | | |

# Time it takes to read 2 addresses

(one address per core)




Most pairs are

1. Different channel; or
2. same channel, different bank

Pixel 4XL from 2019 with SM8150 / Hana

# Overview

Straw-man Cache Attack Immunity Framework

DRAMA Side Channel Signal

**Evaluation of DRAMA Side Channel Signal potential**

# Why is this Relevant?

Microarchitectural attacks

- Like CPU caches, DRAM is a shared resource
- Attacker can monitor access time
- Victim leaks behavior about code and/or data accesses
- This is a problem for secret-dependent memory accesses

```
void
mul_point_scalar (point result,
   scalar scalar, point point)
{
  for (j=nbits-1; j >= 0; j--) {
   ec_duplicate_point (result);
   if (bit_on (scalar, j))
    ec_add_point(result,point);
  }
}
```

# DRAMA Queen

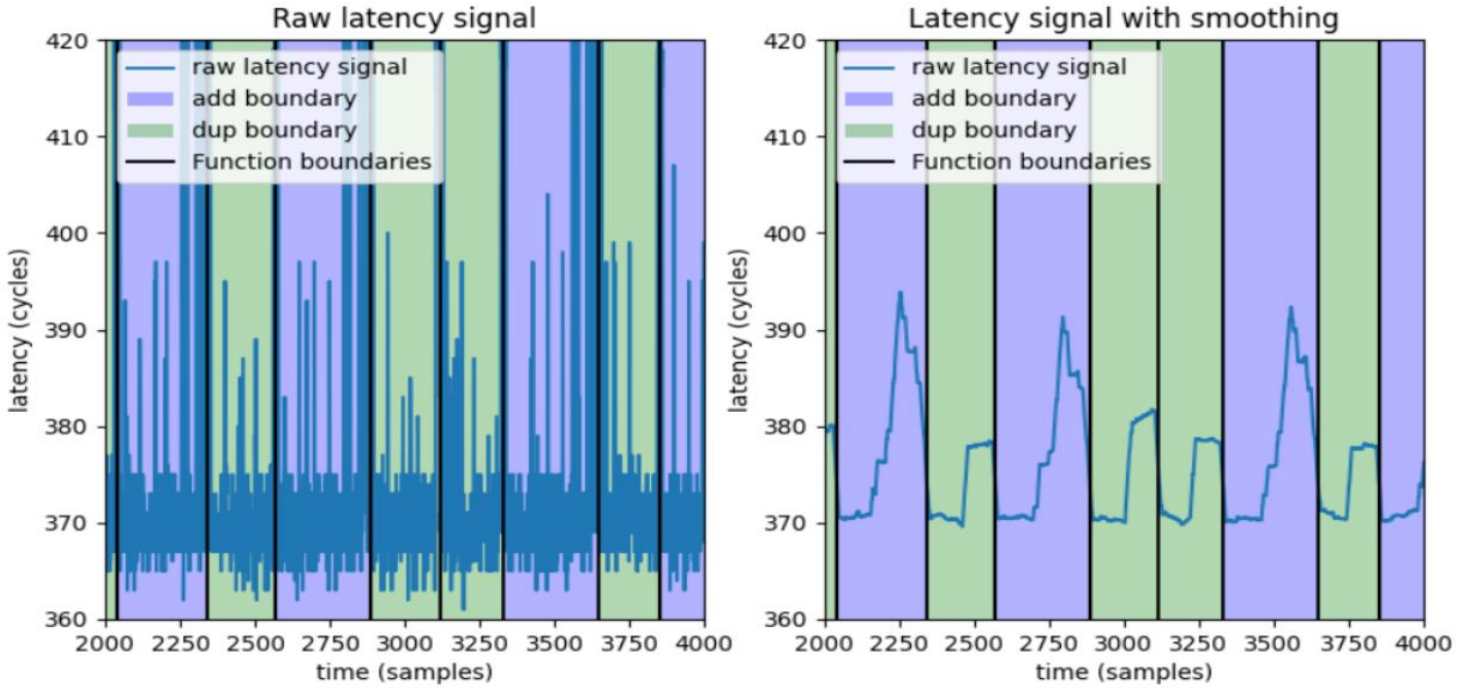
Revisiting Timing Attacks on DRAM

- We apply the cache attack immunization framework:
  victim disables caching for security-sensitive code & data pages
  - Accesses to such page go to DRAM – stops all cache attacks!

- Attacker monitors a single address that is in the same DRAM bank
  - Address must fall in the same bank as the cryptographic code function
  - Time every access to construct an execution trace
  - High latency suggests a bank conflict or page miss – victim must have done an access!

# DRAMA Queen

Revisiting Timing Attacks on DRAM

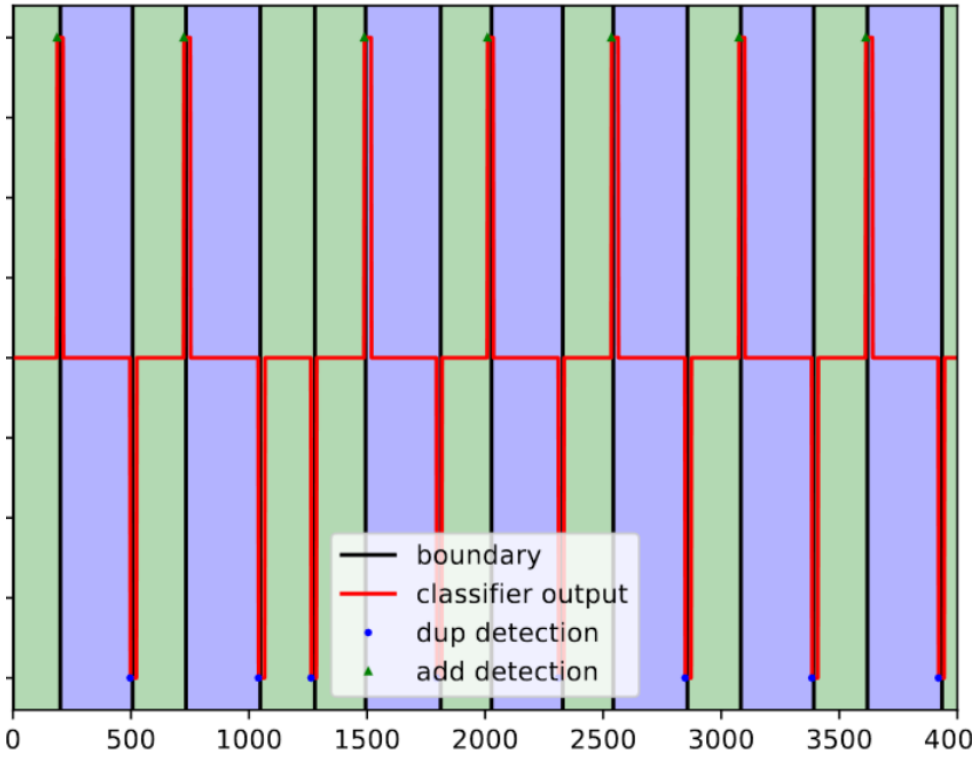


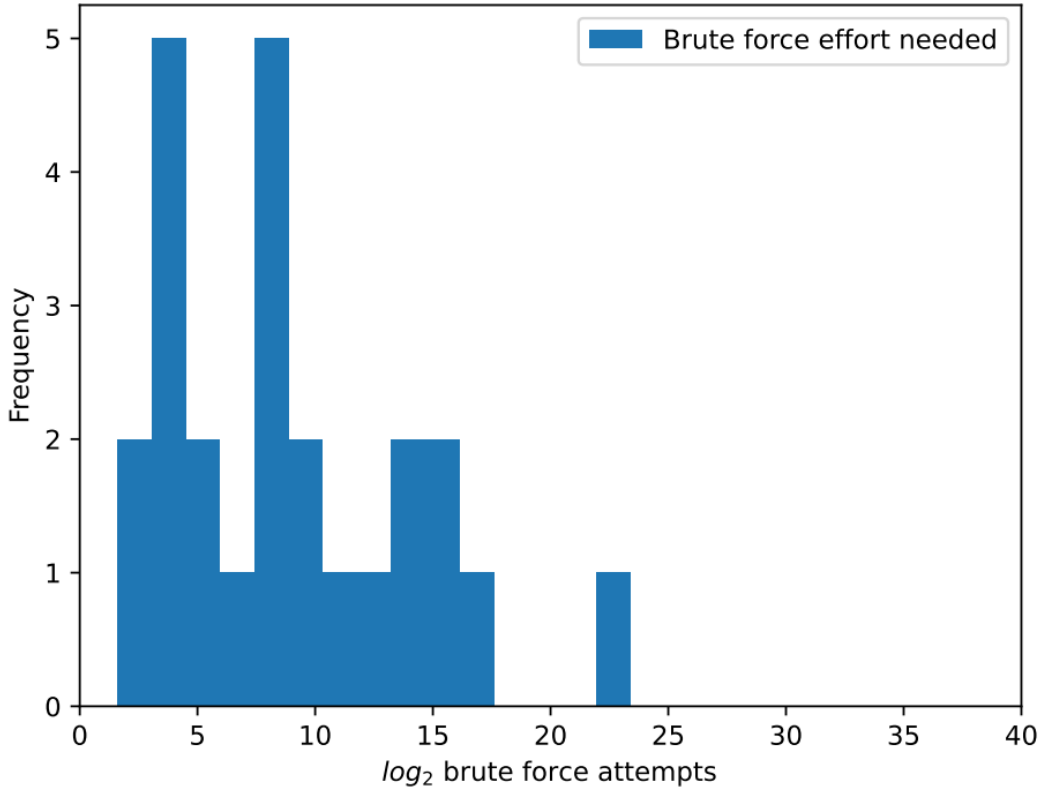

- This is what an attacker can “see” using this side channel
- Looks like the signal is clear enough to distinguish secret phases
- Can we recover secret key bits?

# DRAMA Queen

Revisiting Timing Attacks on DRAM

Yes, automatic classification using an SVM classifier is reliable and required brute force effort for key recovery is modest



# Conclusion

Revisiting Timing Attacks on DRAM

- The side channel immunity framework probably will work against cache attacks, but should consider more shared resources, e.g. DRAM
- In its current state, not a complete defense against side channels
- OS may be able to complete DRAM bank isolation

# Thank You

Revisiting Timing Attacks on DRAM

- Thank you for your attention