DRAMA: Exploiting DRAM Buffers for Fun and Profit

Master Defense Presentation

Michael Schwarz

October 13, 2016

Graz University of Technology

Introduction

Overview

If cache attacks are not possible, is the system secure against microarchitectural side-channel attacks?

Motivation

- We know “normal” Cache Attacks
  - Flush+Reload
  - Prime+Probe
  - Flush+Flush

Motivation

- We know “normal” Cache Attacks
    - Flush+Reload
    - Prime+Probe
    - Flush+Flush
- As these attacks became known, countermeasures were developed
    - Deactivate Memory Deduplication
    - Use multiple CPUs that do not share a cache

Overview

- Identify DRAM as a new attack target across CPUs

Overview

- Identify DRAM as a new attack target across CPUs
- First fully automated method to reverse engineer DRAM

Overview

- Identify DRAM as a new attack target across CPUs
- First fully automated method to reverse engineer DRAM
- Demonstrate DRAM-based attacks

Overview

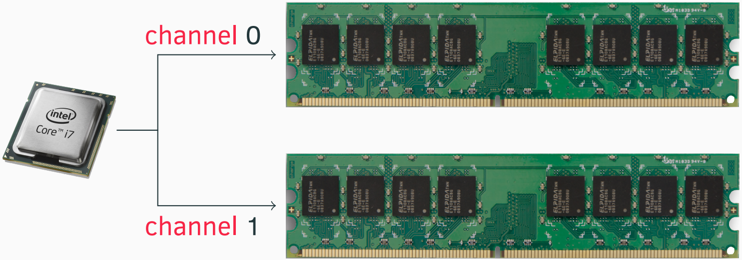
- Identify DRAM as a new attack target across CPUs
- First fully automated method to reverse engineer DRAM
- Demonstrate DRAM-based attacks
    - DRAM-based template attacks

Overview

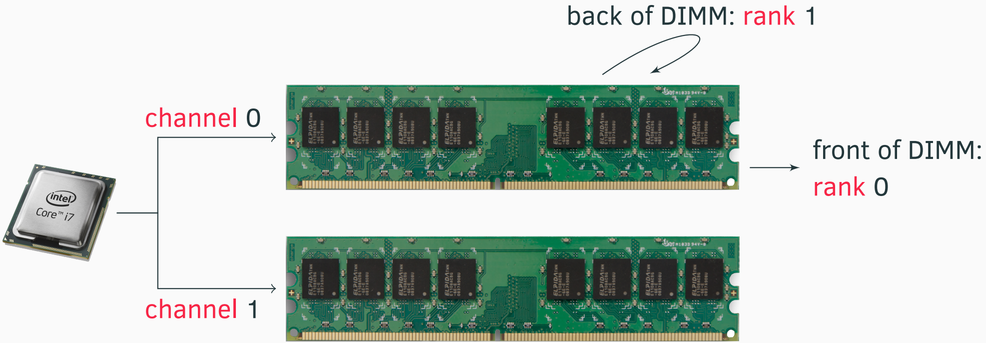
- Identify DRAM as a new attack target across CPUs
- First fully automated method to reverse engineer DRAM
- Demonstrate DRAM-based attacks
    - DRAM-based template attacks
    - Access the internet from a VM without network hardware using a JavaScript covert channel

How is DRAM organized?

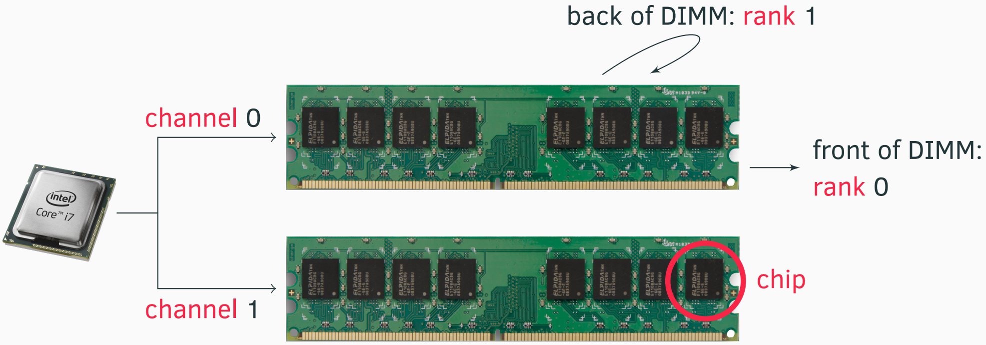
How is DRAM organized?



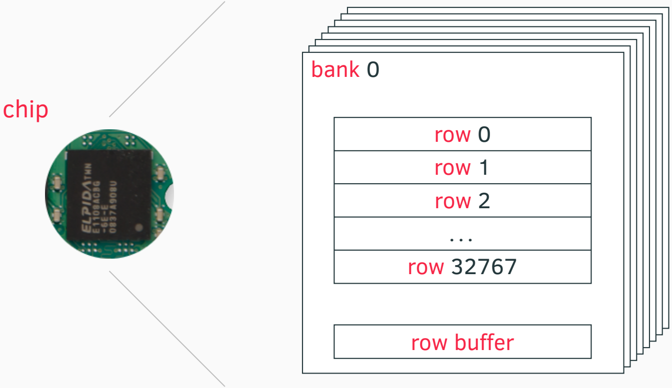
How is DRAM organized?



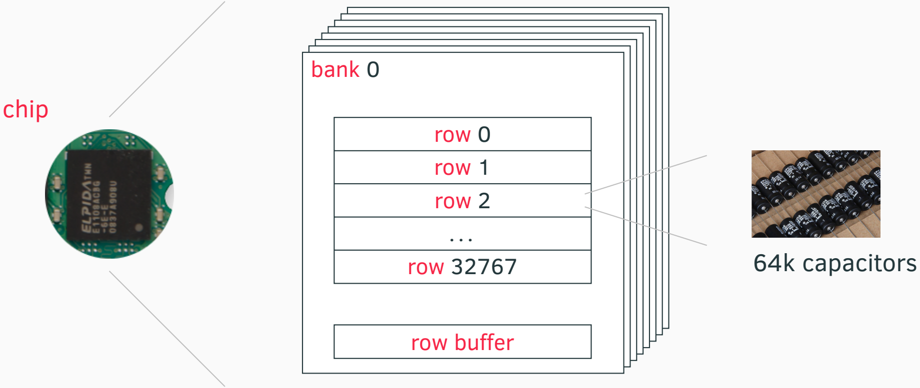
How is DRAM organized?



DRAM organization



DRAM organization



Reading from DRAM

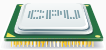
The Row buffer

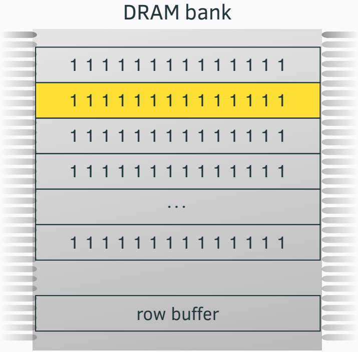
Capacitors discharge when reading bits

- Buffer the bits when reading them from the cells
- Write the bits back to the cells when done reading

= Row buffer

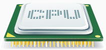
How reading from DRAM works

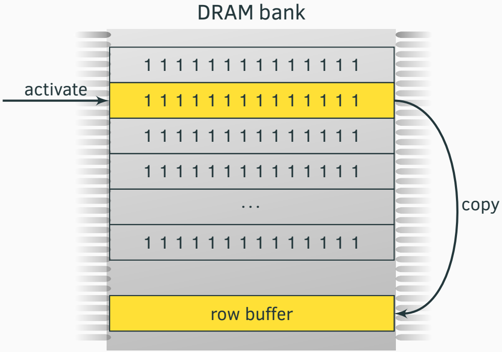




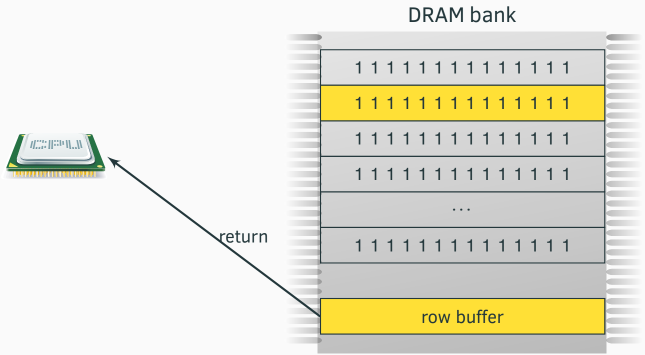
CPU reads row 1,
row buffer empty!

How reading from DRAM works

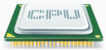




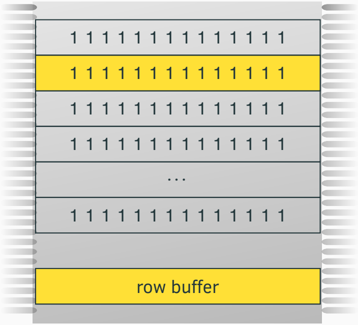
How reading from DRAM works



How reading from DRAM works

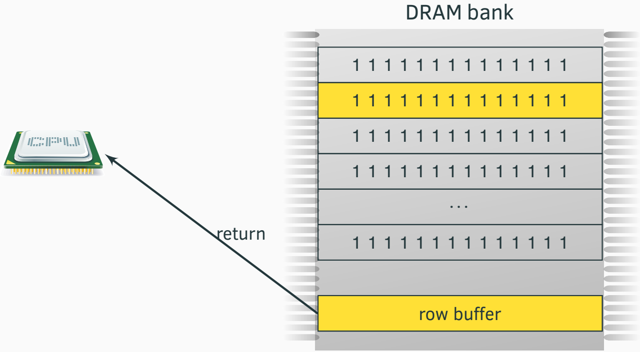


DRAM bank



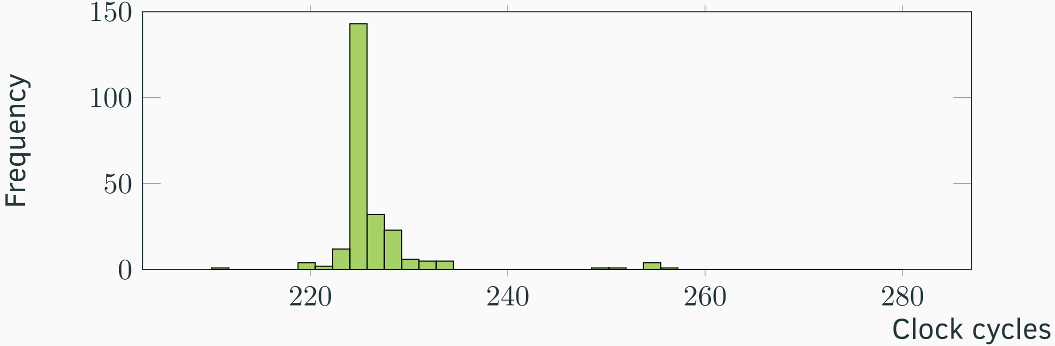
CPU reads row 1,
row buffer now full!

How reading from DRAM works



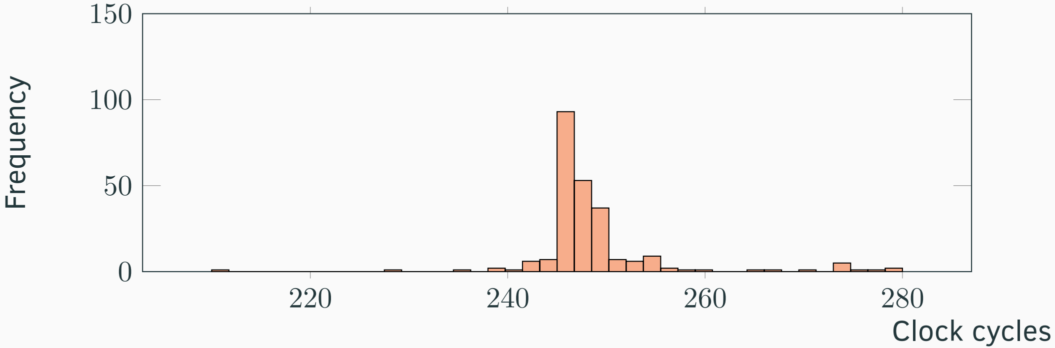
Less work!
Is it faster?

We can measure a difference



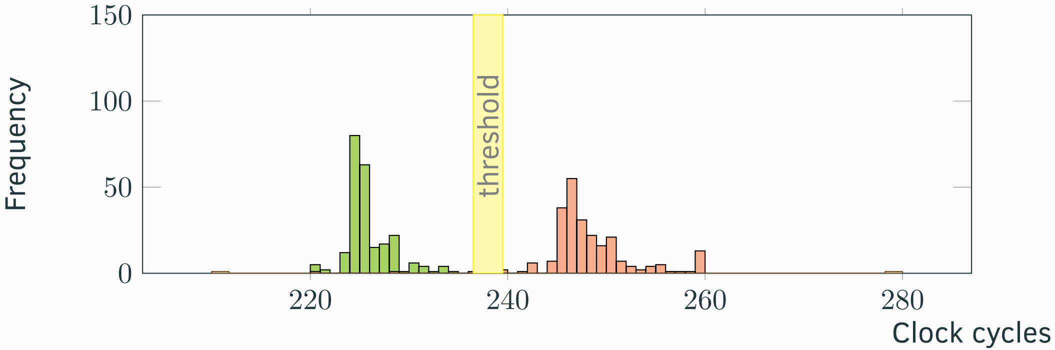
Row hit

We can measure a difference



Row conflicts

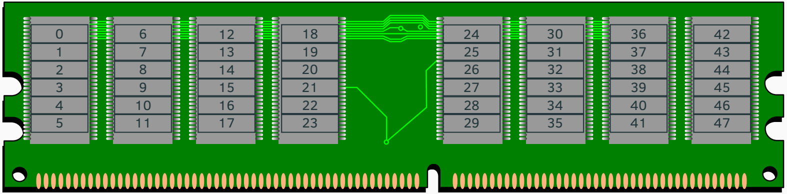
We can measure a difference

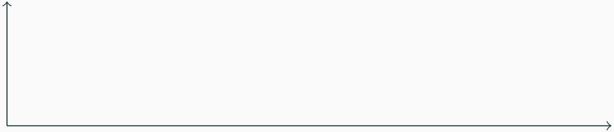


Difference between row hits (≈ 225 cycles) and row conflicts (≈ 247 cycles)

Reverse Engineering the Mapping

Reversing the mapping function - Approach



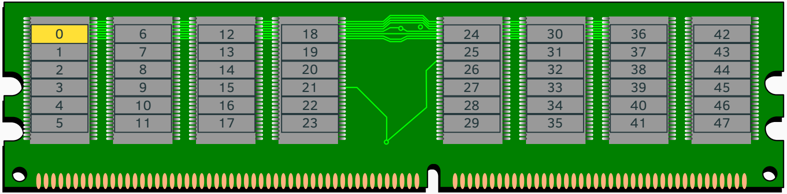


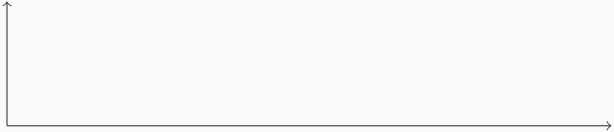
Different bank

Same bank

Reversing the mapping function - Approach

Select random base address in one bank

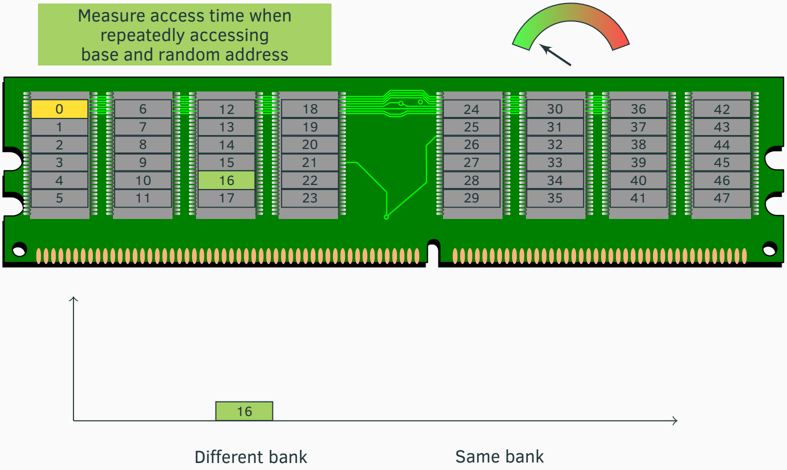




Different bank

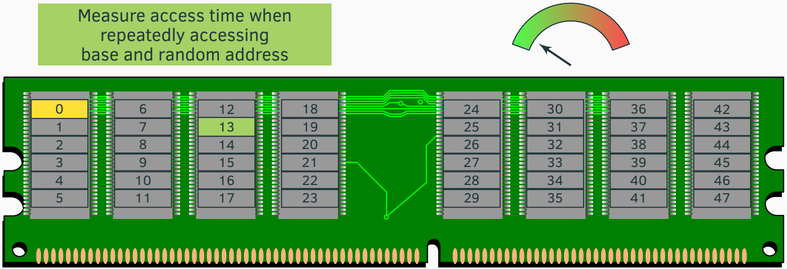
Same bank

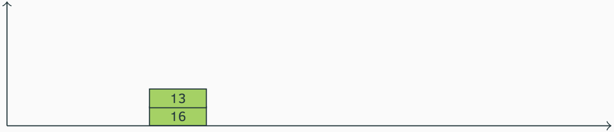
Reversing the mapping function - Approach



Reversing the mapping function - Approach

Measure access time when repeatedly accessing base and random address



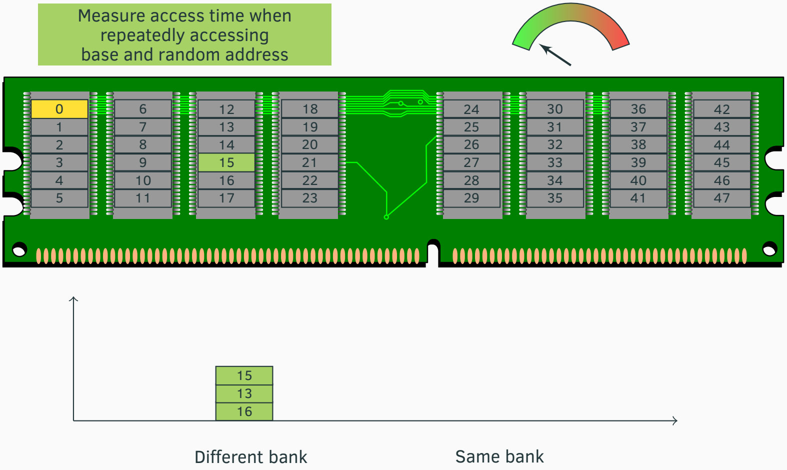


Different bank

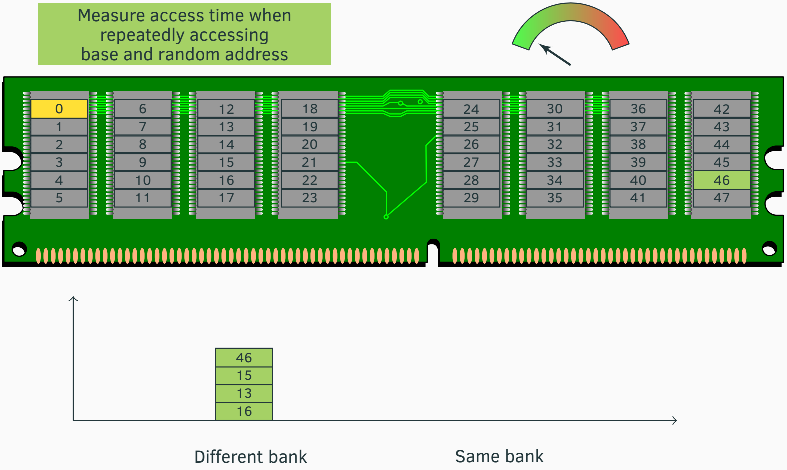
Same bank

Reversing the mapping function - Approach

Measure access time when
repeatedly accessing
base and random address

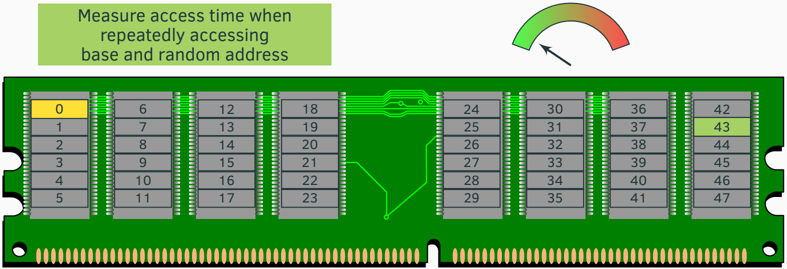


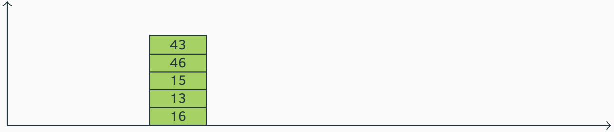
Reversing the mapping function - Approach



Reversing the mapping function - Approach

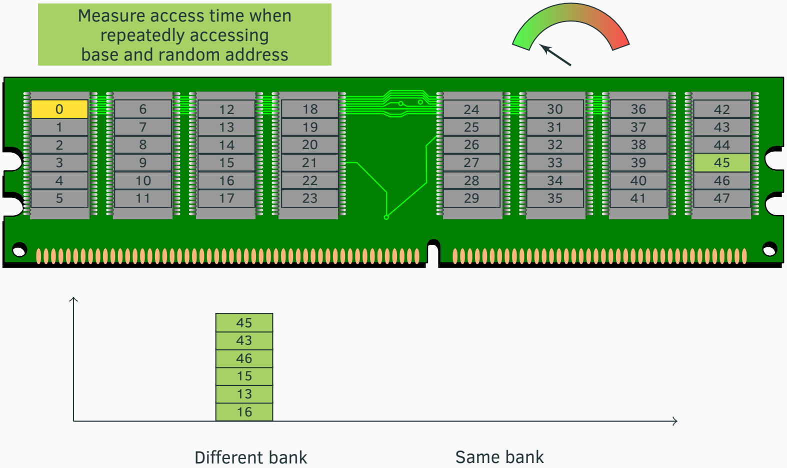
Measure access time when
repeatedly accessing
base and random address





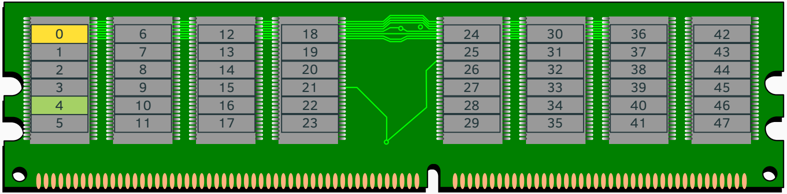
Reversing the mapping function - Approach

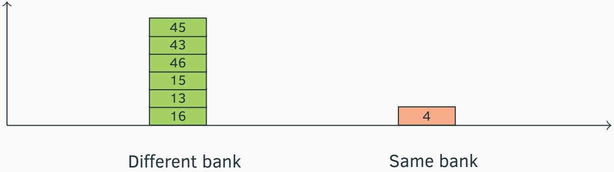
Measure access time when repeatedly accessing base and random address



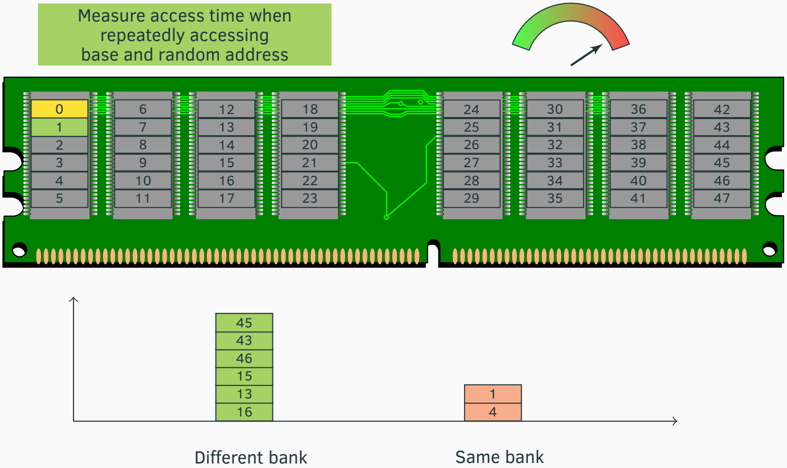
Reversing the mapping function - Approach

Measure access time when repeatedly accessing base and random address

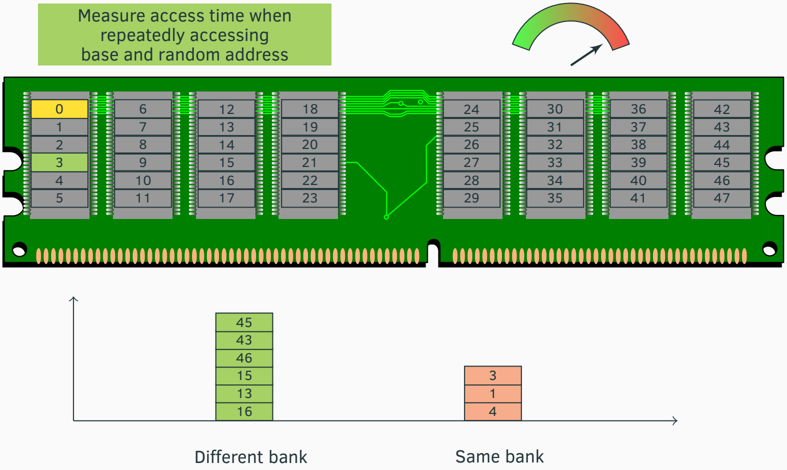




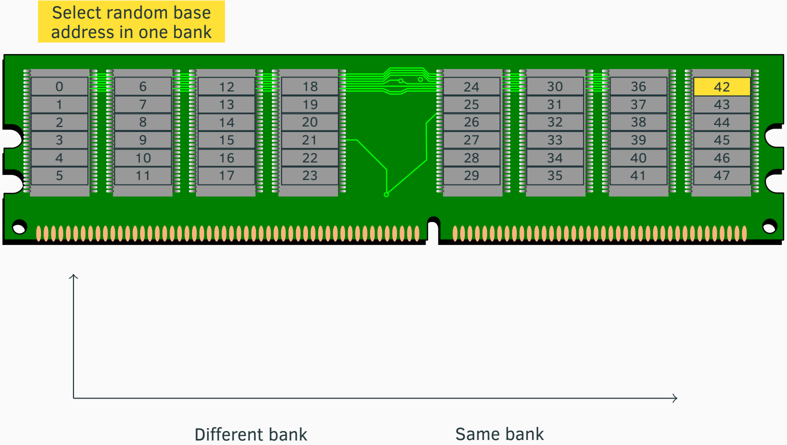
Reversing the mapping function - Approach



Reversing the mapping function - Approach

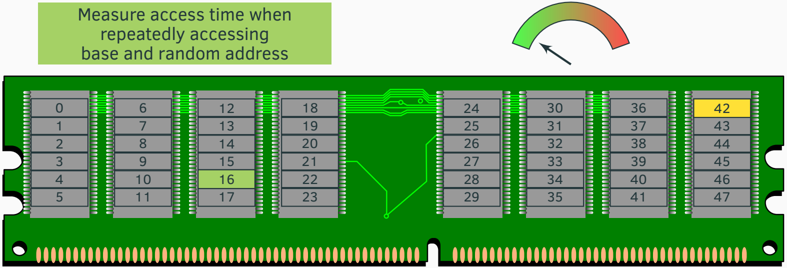


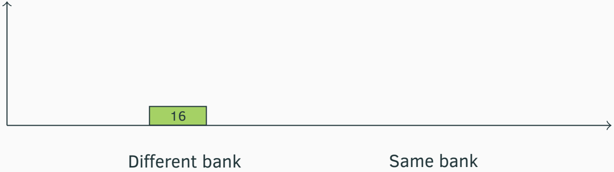
Reversing the mapping function - Approach



Reversing the mapping function - Approach

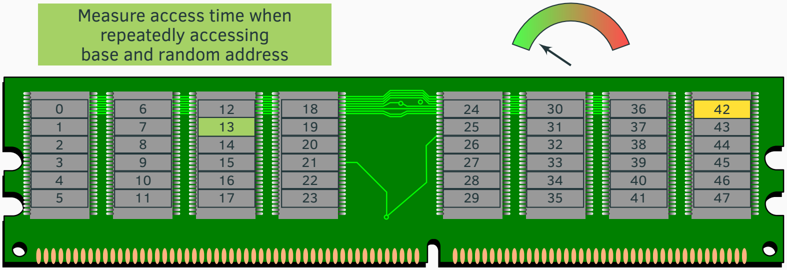
Measure access time when
repeatedly accessing
base and random address

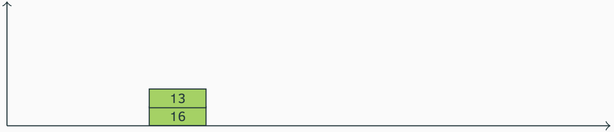




Reversing the mapping function - Approach

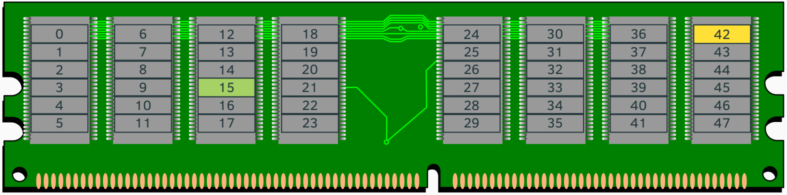
Measure access time when repeatedly accessing base and random address

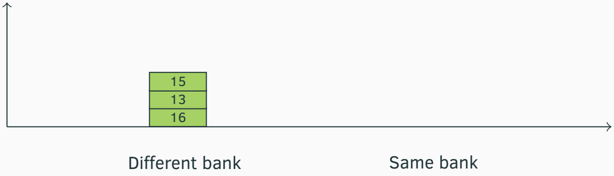




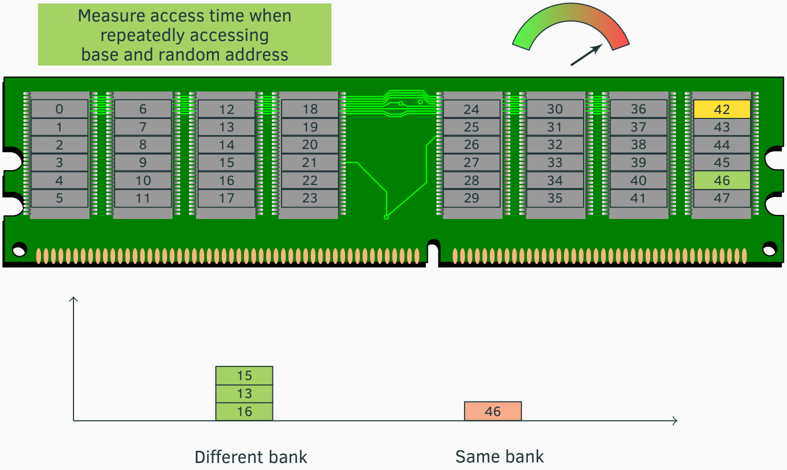
Reversing the mapping function - Approach

Measure access time when repeatedly accessing base and random address

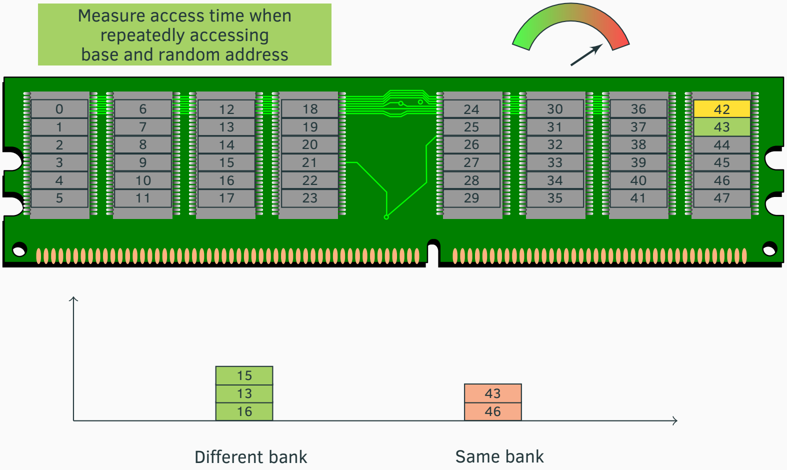




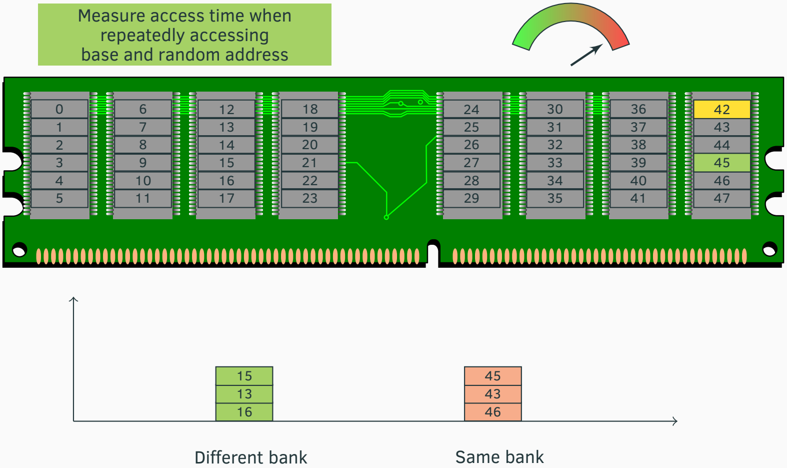
Reversing the mapping function - Approach



Reversing the mapping function - Approach

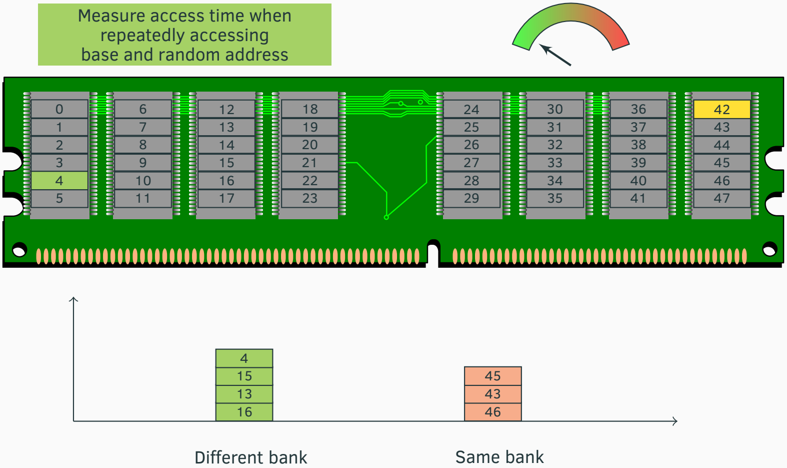


Reversing the mapping function - Approach



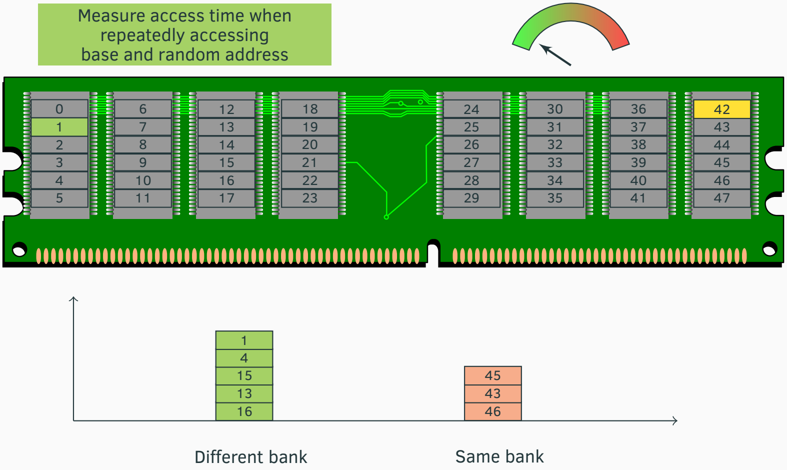
Reversing the mapping function - Approach

Measure access time when repeatedly accessing base and random address



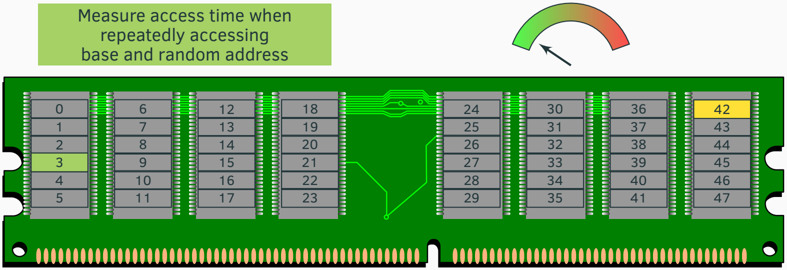
Reversing the mapping function - Approach

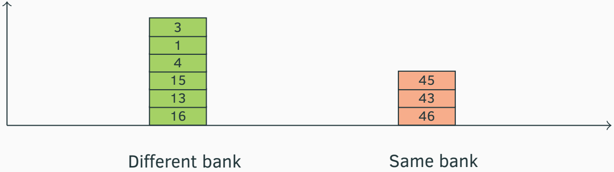
Measure access time when repeatedly accessing base and random address



Reversing the mapping function - Approach

Measure access time when
repeatedly accessing
base and random address





Reversing the mapping function - Approach

- Repeat the process for all banks

Reversing the mapping function - Approach

- Repeat the process for all banks
- For each bank, we have a set of addresses that map to this bank

Reversing the mapping function - Approach

- Repeat the process for all banks
- For each bank, we have a set of addresses that map to this bank
- We can see it as a linear equation system

Reversing the mapping function - Approach

- Repeat the process for all banks
- For each bank, we have a set of addresses that map to this bank
- We can see it as a linear equation system
- Solving it gives us the bits used for the mapping functions

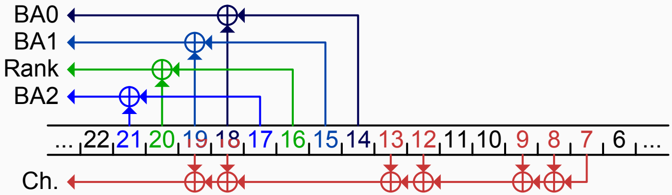
Reversing the mapping function - Approach

- Repeat the process for all banks
- For each bank, we have a set of addresses that map to this bank
- We can see it as a linear equation system
- Solving it gives us the bits used for the mapping functions
- The alternative: generate every possible XOR function and check if it yields the same result for all addresses in the set

Reversing the mapping function - Approach

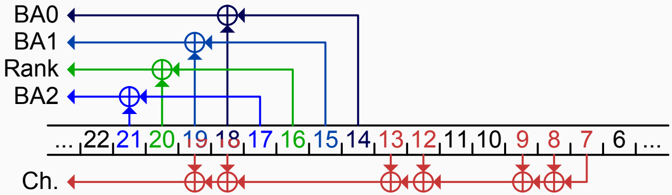
- Repeat the process for all banks
- For each bank, we have a set of addresses that map to this bank
- We can see it as a linear equation system
- Solving it gives us the bits used for the mapping functions
- The alternative: generate every possible XOR function and check if it yields the same result for all addresses in the set
- This is still very fast (in the order of seconds)

Results



- We developed a toolkit that reverse engineers the mapping fully automatically

Results



- We developed a toolkit that reverse engineers the mapping fully automatically
- We tested it on Ivy Bridge, Haswell, Skylake, ARMv7 and ARMv8

Attacks

Spying

- We want to spy on the behaviour of a victim

Spying

- We want to spy on the behaviour of a victim
- The victim will not know that we spy on it

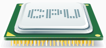
Spying

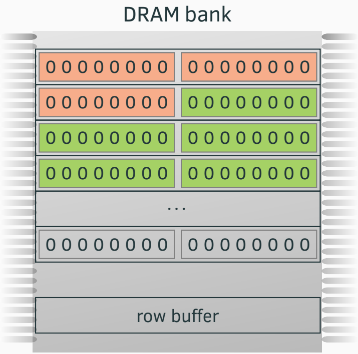
- We want to spy on the behaviour of a victim
- The victim will not know that we spy on it
- We can use row hits to get useful information

Spying

- We want to spy on the behaviour of a victim
- The victim will not know that we spy on it
- We can use row hits to get useful information
- Advantage over cache attacks: it works across CPUs

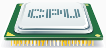
Attack Primitive: Row hit

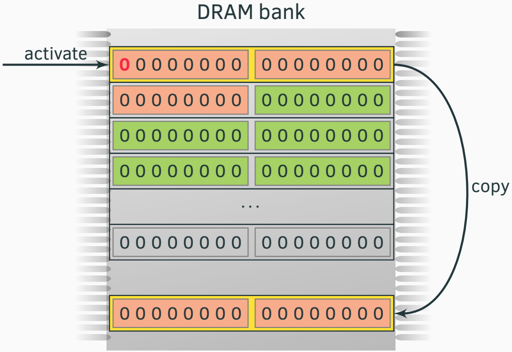




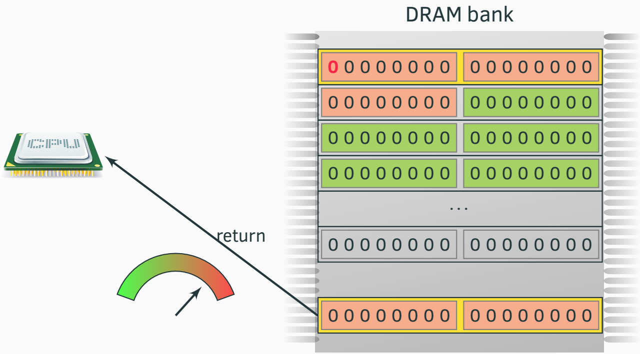
Spy activates row 0, get copied to row buffer

Attack Primitive: Row hit

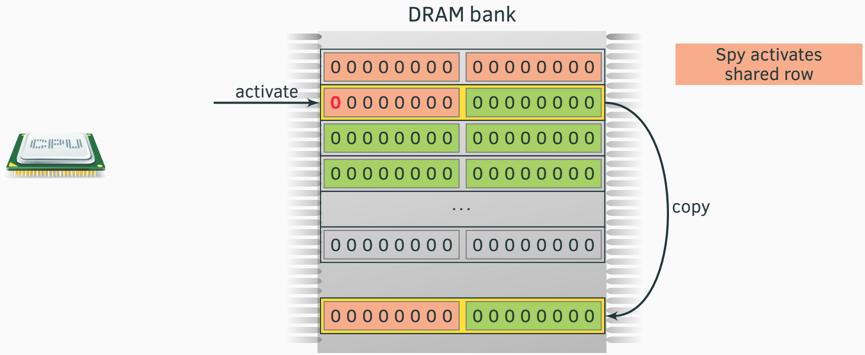




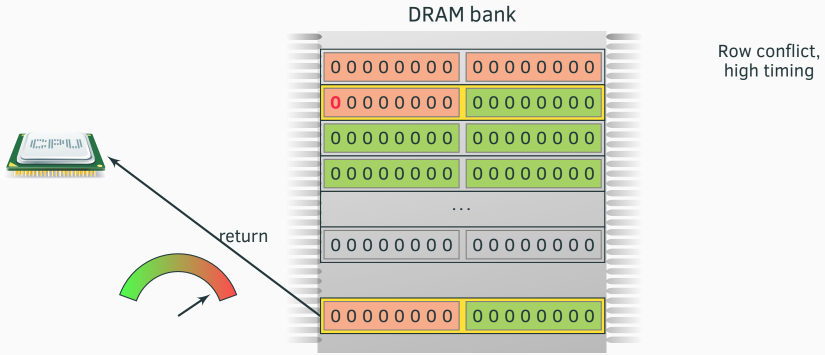
Attack Primitive: Row hit



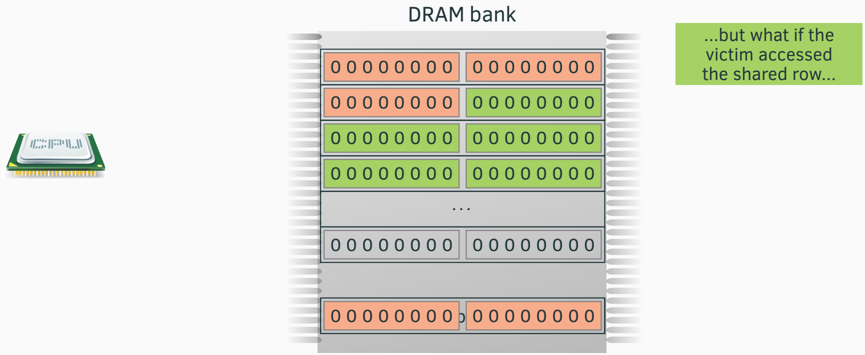
Attack Primitive: Row hit



Attack Primitive: Row hit

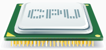


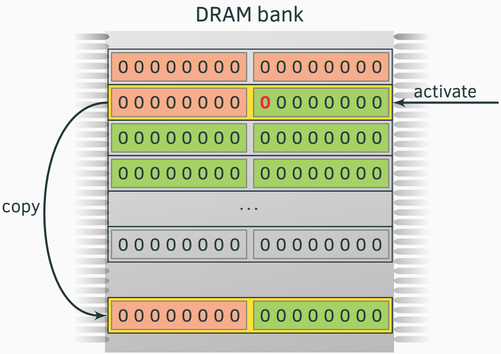
Attack Primitive: Row hit



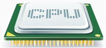
Attacks

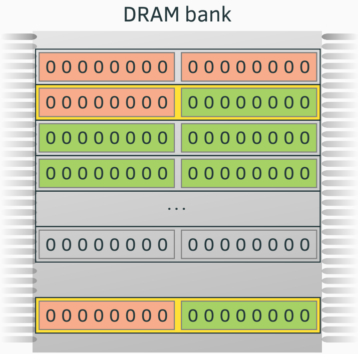
Attack Primitive: Row hit





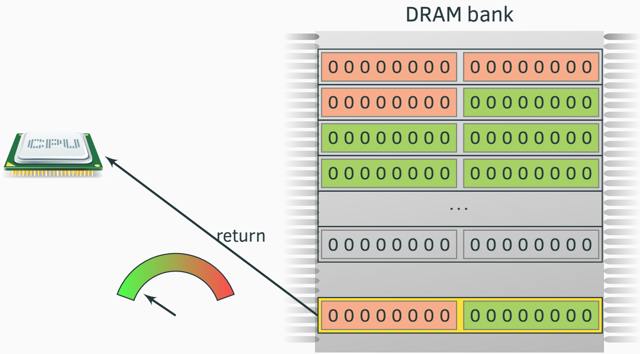
Attack Primitive: Row hit





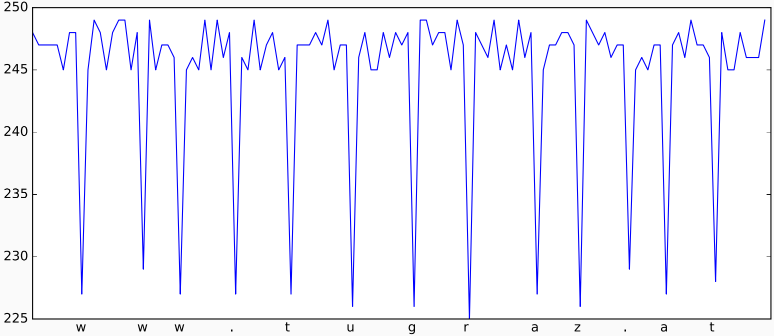
...before the spy activates it

Attack Primitive: Row hit



Row hit, faster

Result: Spying on Firefox



Covert channel

What is a covert communication?

Covert channel

What is a covert communication?

- Two programs would like to communicate

Covert channel

What is a covert communication?

- Two programs would like to communicate but are not allowed to do so

Covert channel

What is a covert communication?

- Two programs would like to communicate but are not allowed to do so
- All “normal” channels are blocked or monitored

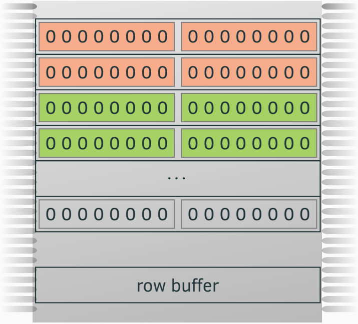
Covert channel

What is a covert communication?

- Two programs would like to communicate but are not allowed to do so
- All “normal” channels are blocked or monitored
- They have to find a side channel

Attack Primitive: Row miss

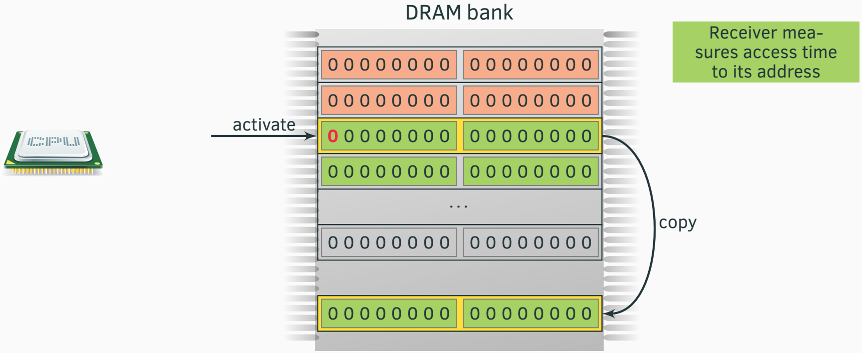
DRAM bank



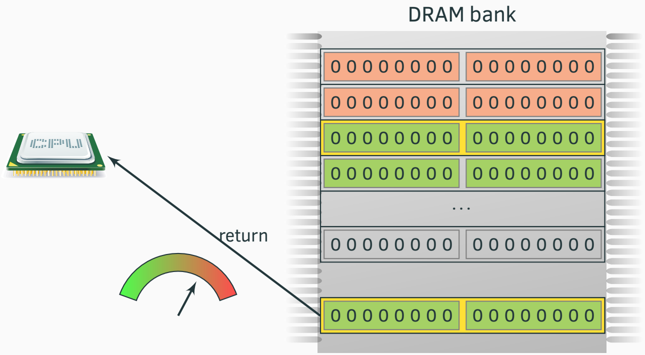
Sender and receiver decide on one bank

Attacks

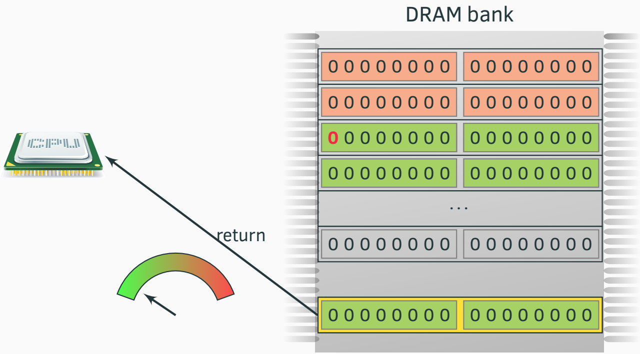
Attack Primitive: Row miss



Attack Primitive: Row miss

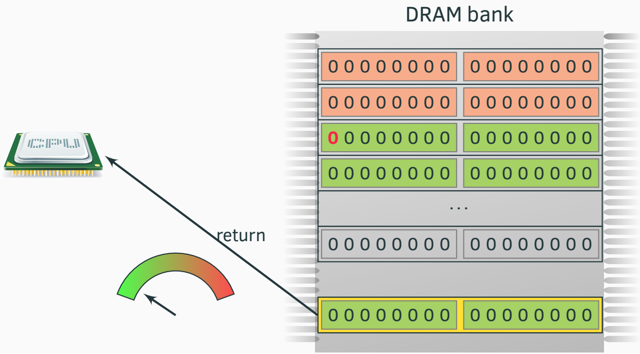


Attack Primitive: Row miss

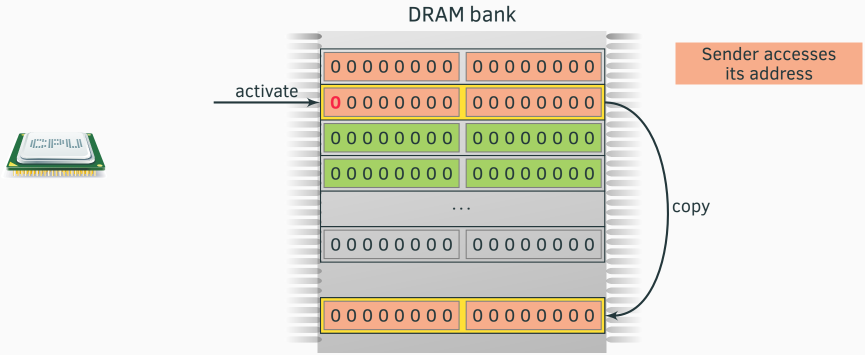


Repeated access always has low access times

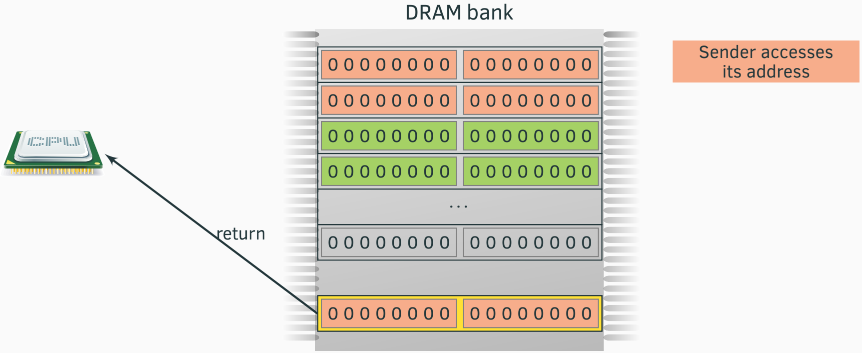
Attack Primitive: Row miss



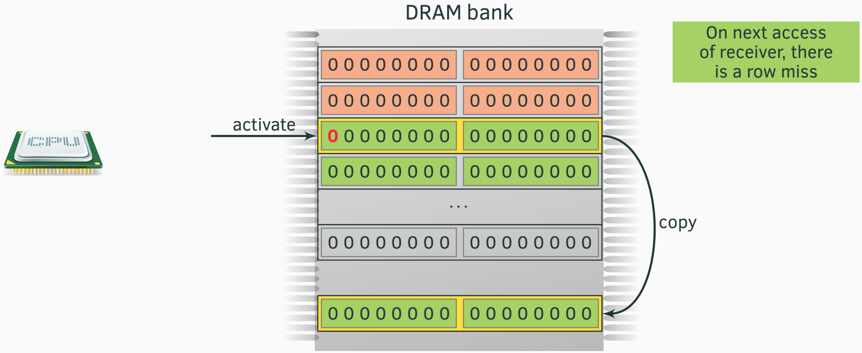
Attack Primitive: Row miss



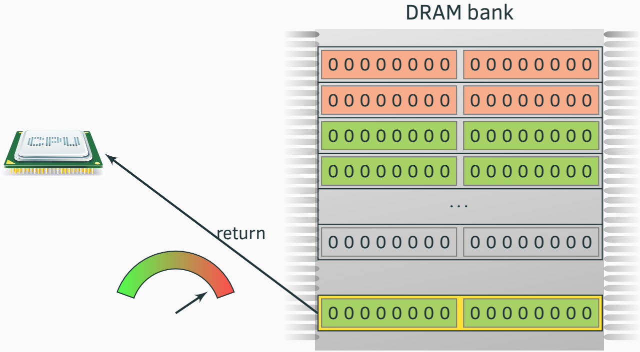
Attack Primitive: Row miss



Attack Primitive: Row miss



Attack Primitive: Row miss



Receiver has high access time

DRAM Covert Channel

- Sender and receiver agree on a bank (can be hardcoded)

DRAM Covert Channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both sender and receiver select a different row inside this bank

DRAM Covert Channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both sender and receiver select a different row inside this bank
- Receiver measures access time for this row

DRAM Covert Channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both sender and receiver select a different row inside this bank
- Receiver measures access time for this row
- Sender can transmit 0 by doing nothing and 1 by causing row conflict

DRAM Covert Channel

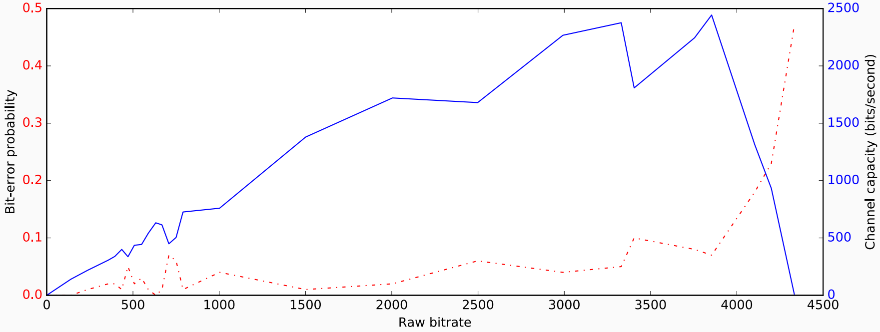
- Sender and receiver agree on a bank (can be hardcoded)
- Both sender and receiver select a different row inside this bank
- Receiver measures access time for this row
- Sender can transmit 0 by doing nothing and 1 by causing row conflict
- If measured timing was “fast” sender transmitted 0.

DRAM Covert Channel

- Sender and receiver both inside the VM

DRAM Covert Channel

- Sender and receiver both inside the VM



JavaScript Covert Channel

JavaScript Covert Channel

- JavaScript running in the browser on the host

JavaScript Covert Channel

- JavaScript running in the browser on the host
- Browser acts as receiver

JavaScript Covert Channel

- JavaScript running in the browser on the host
- Browser acts as receiver
- Sender in VM without internet access

JavaScript Covert Channel

- JavaScript running in the browser on the host
- Browser acts as receiver
- Sender in VM without internet access
- Problem: No addresses in JavaScript

JavaScript Covert Channel

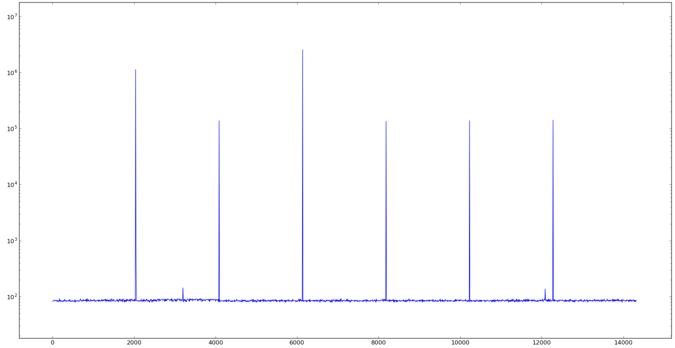
- JavaScript running in the browser on the host
- Browser acts as receiver
- Sender in VM without internet access
- Problem: No addresses in JavaScript
- $\rightarrow$ Cannot apply DRAM functions

The Problem - Physical Addresses

- Iterate over a large array and measure timing

The Problem - Physical Addresses

- Iterate over a large array and measure timing
- We can detect the page borders due to pagefaults



JavaScript Covert Channel

- We only have to trick the victim to visit our page

JavaScript Covert Channel

- We only have to trick the victim to visit our page
- Transmission of approximately $11\,\mathrm{bit/s}$

JavaScript Covert Channel

- We only have to trick the victim to visit our page
- Transmission of approximately $11\,\mathrm{bit/s}$
- Enough to steal keys or passwords

Conclusion

Summary

- We discovered a new attack vector

Summary

- We discovered a new attack vector
- Advantage over cache attacks: it works across CPUs

Summary

- We discovered a new attack vector
- Advantage over cache attacks: it works across CPUs
- Demonstrated two use cases:

Summary

- We discovered a new attack vector
- Advantage over cache attacks: it works across CPUs
- Demonstrated two use cases:
    - Spy on other processes

Summary

- We discovered a new attack vector
- Advantage over cache attacks: it works across CPUs
- Demonstrated two use cases:
    - Spy on other processes
    - Covert channel across CPUs

Summary

- We discovered a new attack vector
- Advantage over cache attacks: it works across CPUs
- Demonstrated two use cases:
  - Spy on other processes
  - Covert channel across CPUs
- Implemented the covert channel in JavaScript

Contribution

- DRAM as a novel attack vector

Contribution

- DRAM as a novel attack vector
  Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM addressing for cross-cpu attacks. (USENIX Security 16).

Contribution

- DRAM as a novel attack vector
  Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM addressing for cross-cpu attacks. (USENIX Security 16).
- DRAM covert channel in JavaScript

Contribution

- DRAM as a novel attack vector
  Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM addressing for cross-cpu attacks. (USENIX Security 16).
- DRAM covert channel in JavaScript
  Schwarz, M. and Fogh, A. (2016). DRAMA: How your DRAM becomes a security problem (Black Hat Europe 2016)

Contribution

- DRAM as a novel attack vector
  Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM addressing for cross-cpu attacks. (USENIX Security 16).
- DRAM covert channel in JavaScript
  Schwarz, M. and Fogh, A. (2016). DRAMA: How your DRAM becomes a security problem (Black Hat Europe 2016)
- Fully automatic DRAM reverse engineering tool

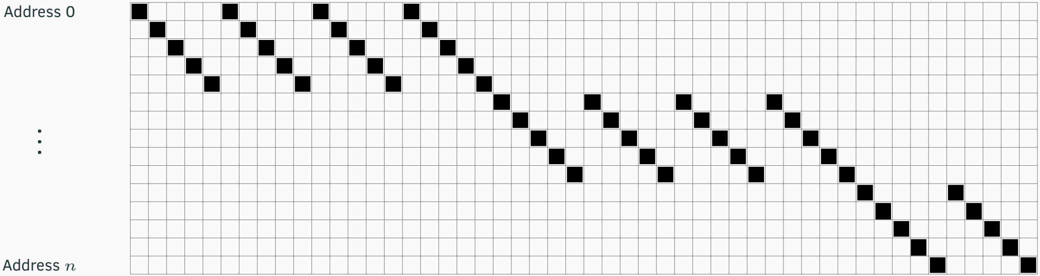
Contribution

- DRAM as a novel attack vector
  Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). DRAMA: Exploiting DRAM addressing for cross-cpu attacks. (USENIX Security 16).
- DRAM covert channel in JavaScript
  Schwarz, M. and Fogh, A. (2016). DRAMA: How your DRAM becomes a security problem (Black Hat Europe 2016)
- Fully automatic DRAM reverse engineering tool
  https://github.com/iaik/drama

Thank you for your attention!

Additional: Covert Channel Transmission

The gory details - Eviction



The gory details - bits

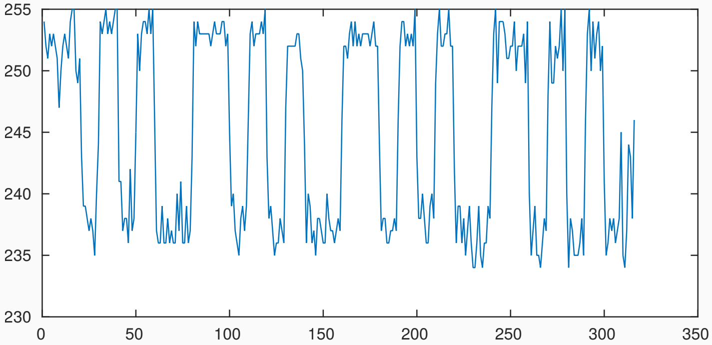


Figure 1: Multiple measurements per bit to have a reliable detection.

The gory details - bits

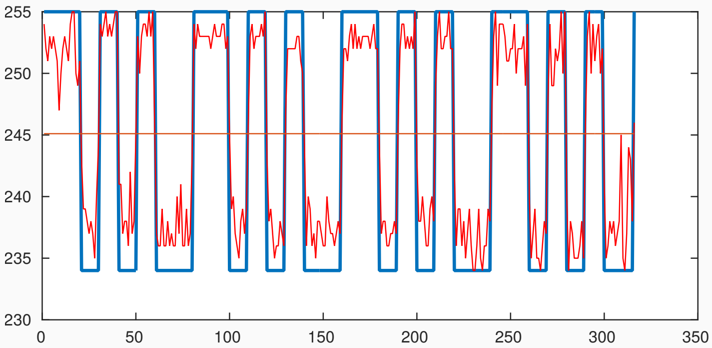
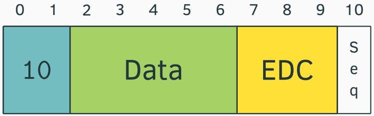


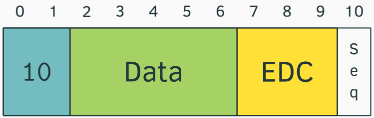
Figure 1: Multiple measurements per bit to have a reliable detection.

The gory details - Packets



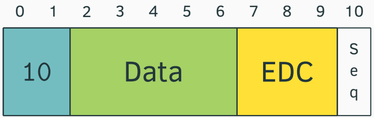
- Communication is based on packets

The gory details - Packets



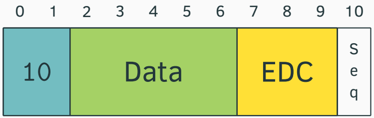
- Communication is based on packets
- Packet starts with a 2-bit preamble

The gory details - Packets



- Communication is based on packets
- Packet starts with a 2-bit preamble
- Data integrity is checked by an error-detection code (EDC)

The gory details - Packets



- Communication is based on packets
- Packet starts with a 2-bit preamble
- Data integrity is checked by an error-detection code (EDC)
- Sequence bit indicates whether it is a retransmission or a new packet

Additional: Accuracy

Accuracy

- Not the whole physical page must be in one row

Accuracy

- Not the whole physical page must be in one row
- Depending on the mapping function, a page can be distributed over multiple rows

Accuracy

- Not the whole physical page must be in one row
- Depending on the mapping function, a page can be distributed over multiple rows
- This is the case if address bits 0 to 11 are used for the mapping

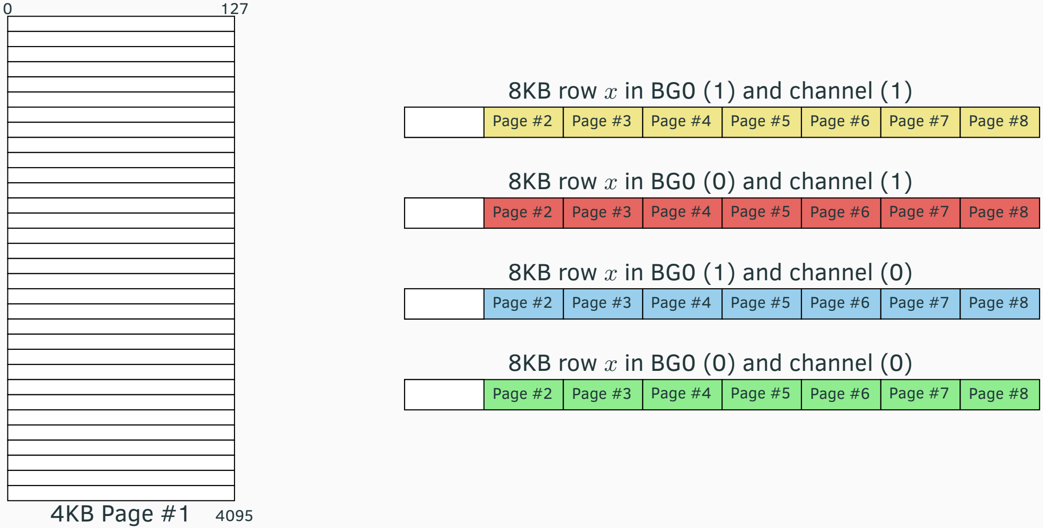
Accuracy

- Not the whole physical page must be in one row
- Depending on the mapping function, a page can be distributed over multiple rows
- This is the case if address bits 0 to 11 are used for the mapping
- For example: Skylake uses low bits for channel (bits 8 and 9) and bankgroup (bit 7)

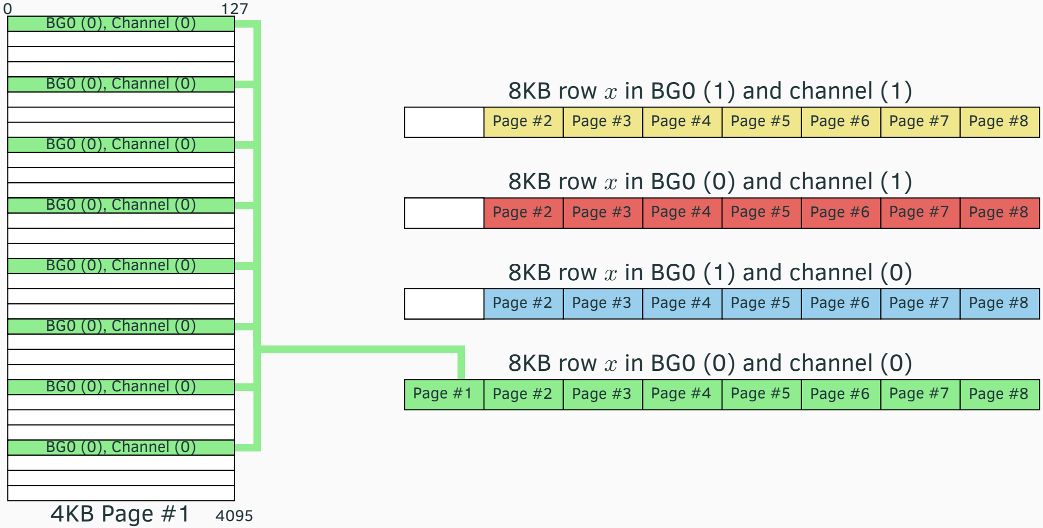
Accuracy

- Not the whole physical page must be in one row
- Depending on the mapping function, a page can be distributed over multiple rows
- This is the case if address bits 0 to 11 are used for the mapping
- For example: Skylake uses low bits for channel (bits 8 and 9) and bankgroup (bit 7)
- One physical page is distributed over 4 rows

Accuracy

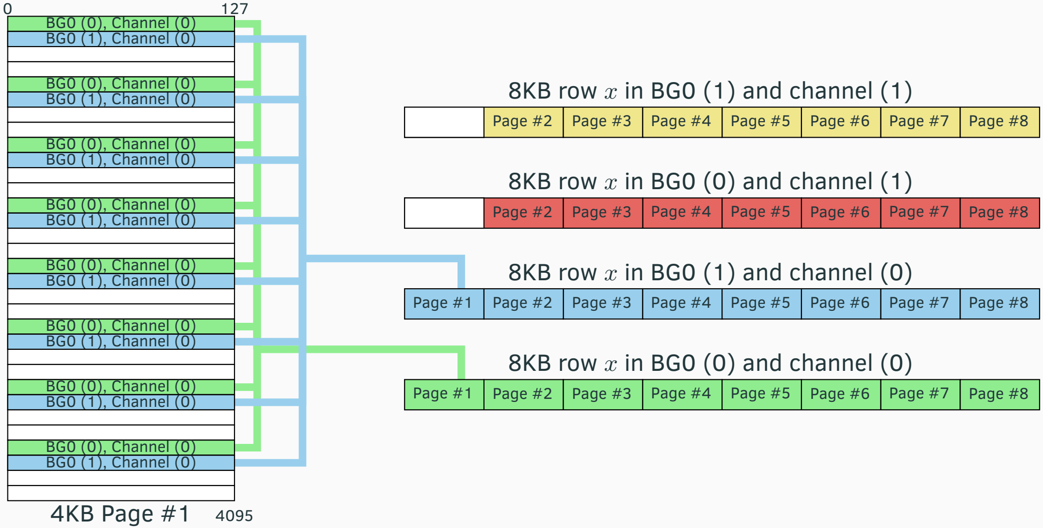

0
127
4KB Page #1
4095
8KB row x in BG0 (1) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (1) and channel (0)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (0)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8

Accuracy



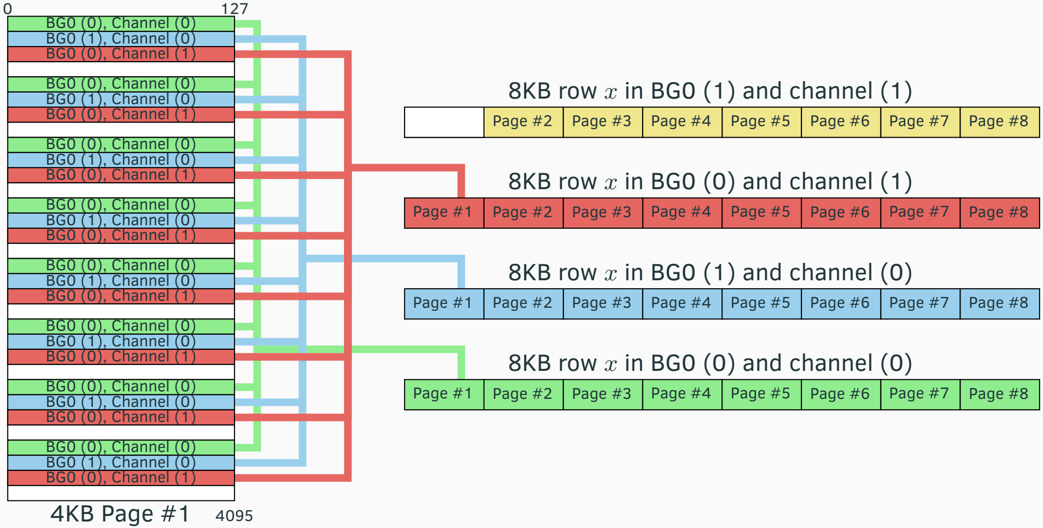
0
127
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
BG0 (0), Channel (0)
4KB Page #1
4095
8KB row x in BG0 (1) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (1) and channel (0)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (0)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8

Accuracy

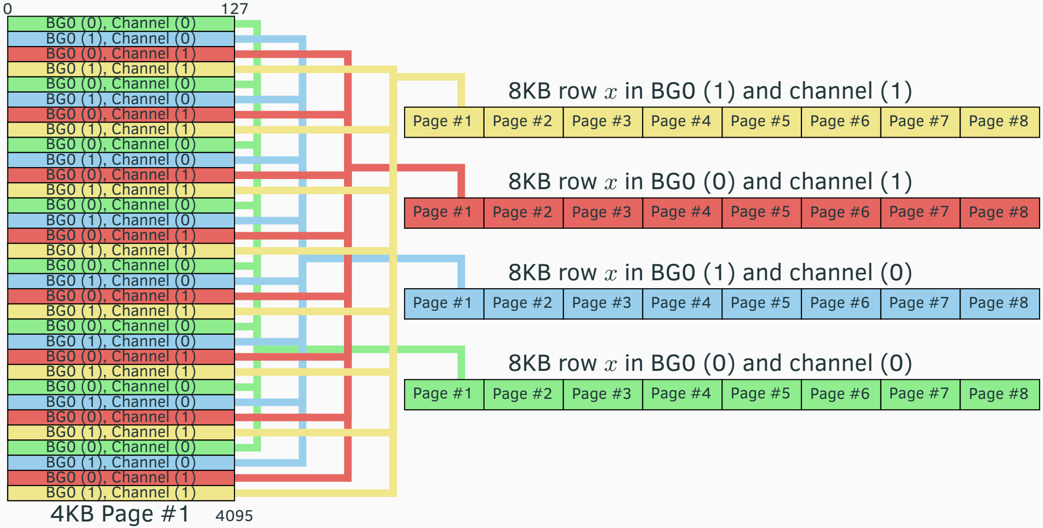


0
127
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (0)
BG0 (1), Channel (0)
4KB Page #1
4095
8KB row x in BG0 (1) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (1)
Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (1) and channel (0)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (0)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8

Accuracy


0
127
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (1)
4KB Page #1
4095
8KB row x in BG0 (1) and channel (1)
Page #2
Page #3
Page #4
Page #5
Page #6
Page #7
Page #8
8KB row x in BG0 (0) and channel (1)
Page #1
8KB row x in BG0 (1) and channel (0)
8KB row x in BG0 (0) and channel (0)

Accuracy


0
127
BG0 (0), Channel (0)
BG0 (1), Channel (0)
BG0 (0), Channel (1)
BG0 (1), Channel (1)
4KB Page #1
4095
8KB row x in BG0 (1) and channel (1)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (1)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (1) and channel (0)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8
8KB row x in BG0 (0) and channel (0)
Page #1 Page #2 Page #3 Page #4 Page #5 Page #6 Page #7 Page #8

References i