

Rowhammer in the Wild

Large-Scale Insights from FlippyR.AM

Martin Heckel^{1,2} (@lunkw1ll)

Daniel Gruss¹ (@lavados)

Florian Adamsky² (@c1t)

¹ Graz University of Technology

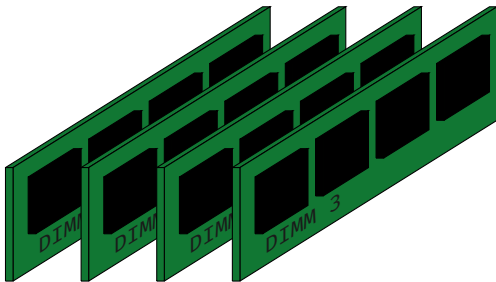
² Hof University of Applied Sciences



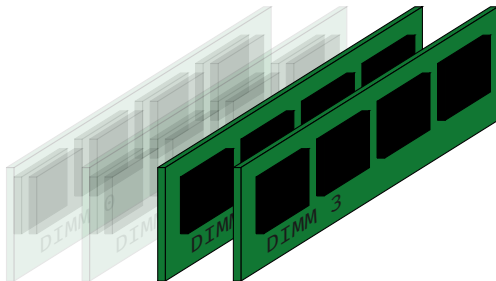
Kofinanziert von der Europäischen Union

What previously happened on
Rowhammer...

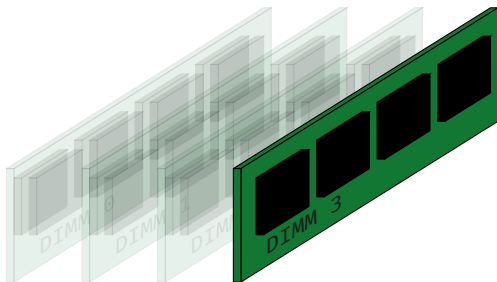
System DRAM

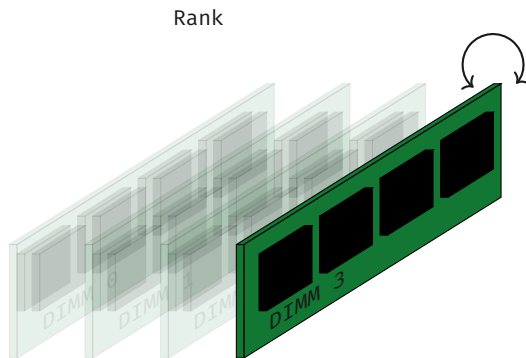


Channel

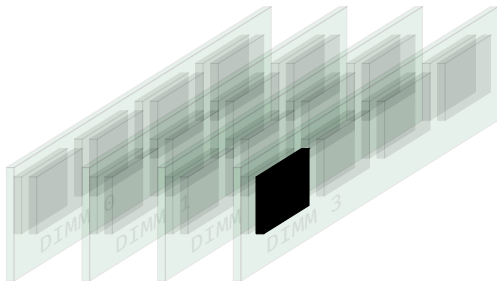


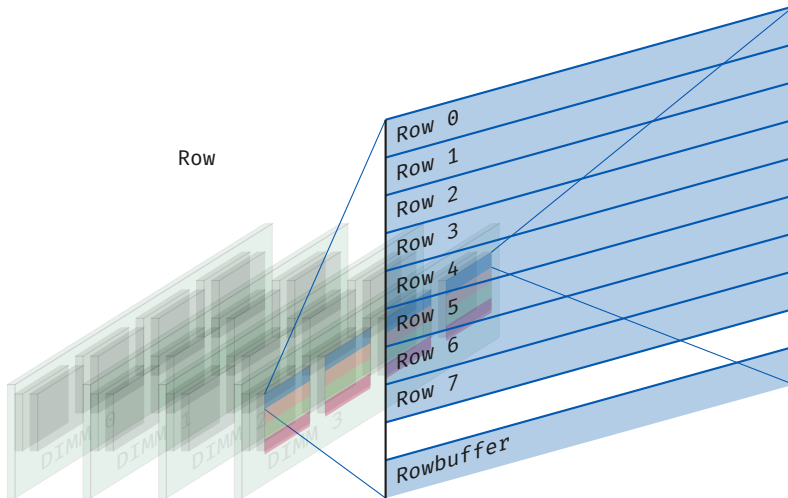
DIMM

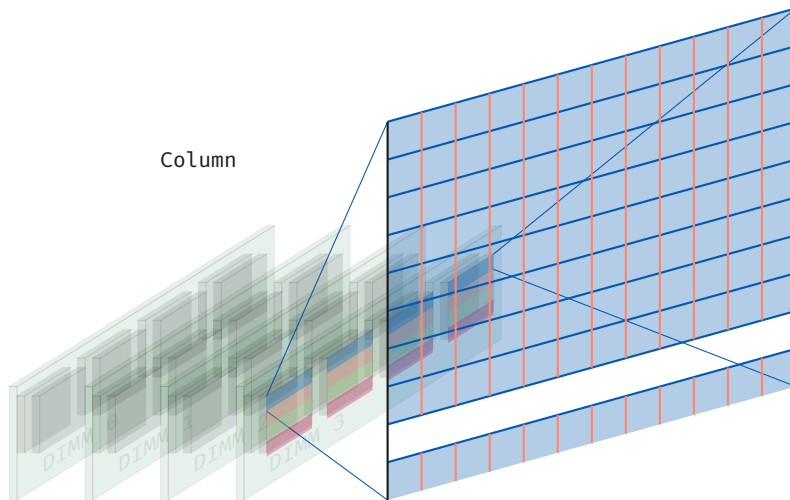


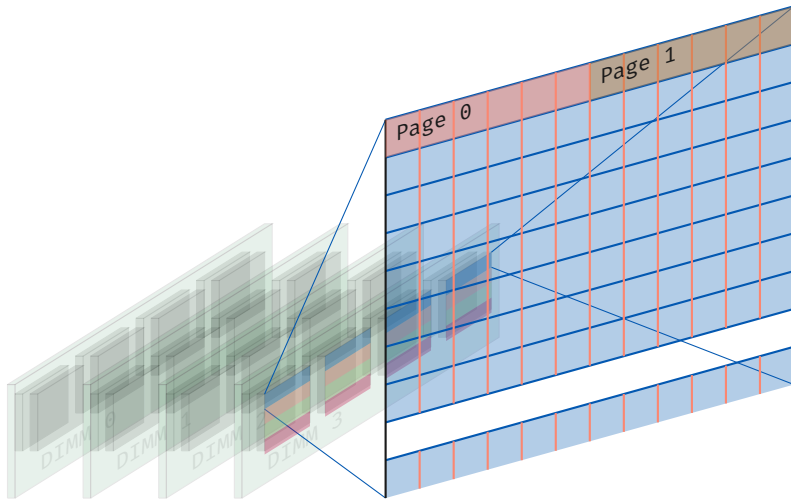


Chip

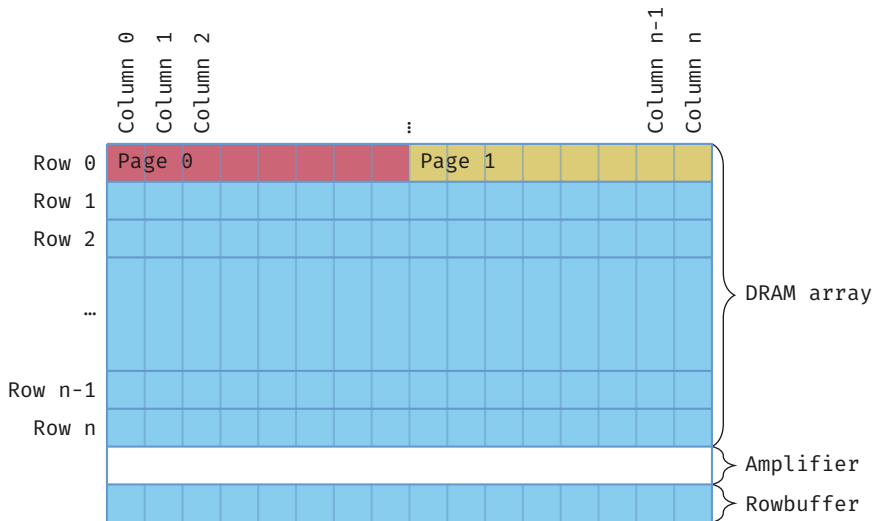


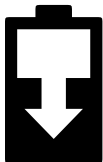


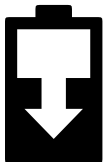




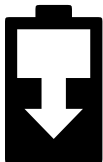
Structure within a DRAM bank



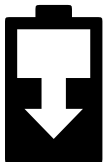




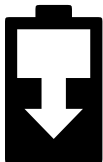
1. Capacitor loses its charge over time



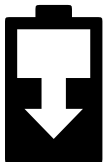
1. Capacitor loses its charge over time
 - Cells must be refreshed regularly (refresh rate)



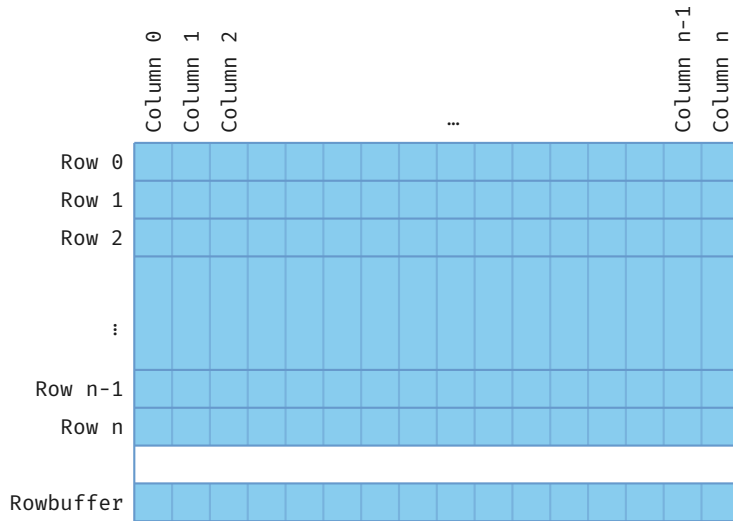
1. Capacitor loses its charge over time
 - Cells must be refreshed regularly (refresh rate)
 - Cells are normally refreshed every 64 ms



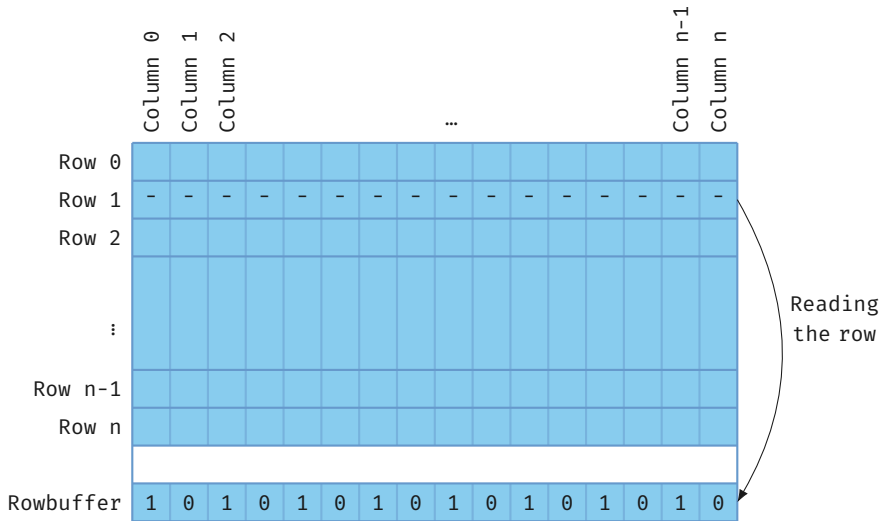
1. Capacitor loses its charge over time
 - Cells must be refreshed regularly (refresh rate)
 - Cells are normally refreshed every 64 ms
2. When reading a row, we destroy the data in this row



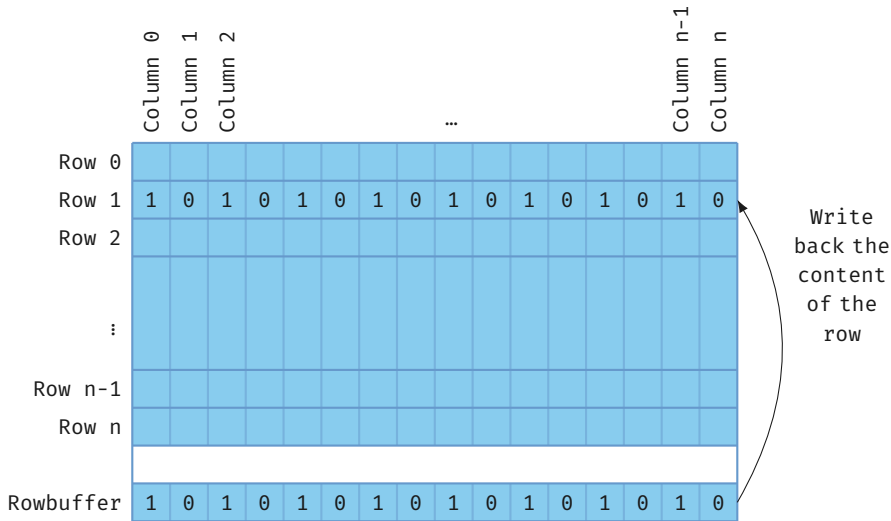
1. Capacitor loses its charge over time
 - Cells must be refreshed regularly (refresh rate)
 - Cells are normally refreshed every 64 ms
2. When reading a row, we destroy the data in this row
 - Intermediate memory in the row buffer



	Column 0	Column 1	Column 2	...										Column n-1	Column n
Row 0															
Row 1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Row 2															
⋮															
Row n-1															
Row n															
Rowbuffer															



Row Buffer



Simple Example of Rowhammer

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer

↗

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
➔	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer

↗

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

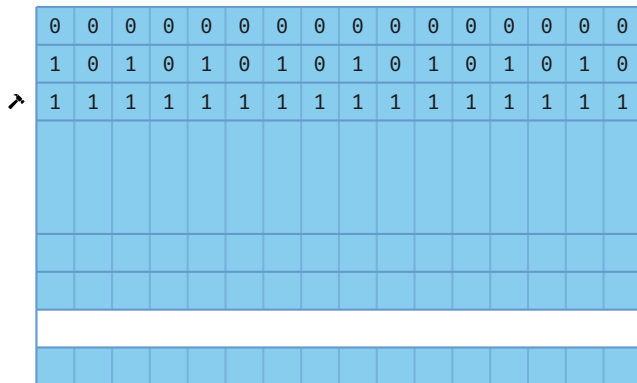
```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

```
clflush (Row 2)
```

```
jmp hammertime
```

Simple Example of Rowhammer

We can touch this!



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

hammertime:

```
mov (Row 0), %eax
```

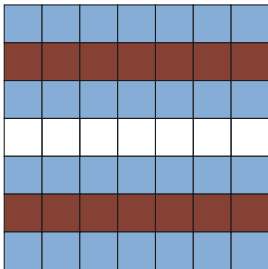
```
mov (Row 2), %ebx
```

```
clflush (Row 0)
```

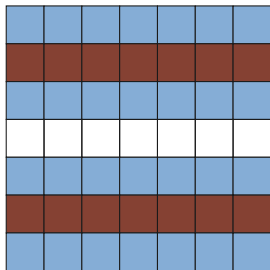
```
clflush (Row 2)
```

```
jmp hammertime
```

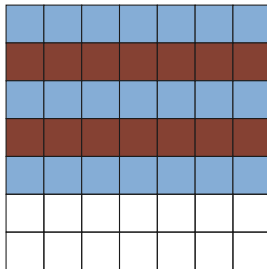

Single-Sided



Single-Sided



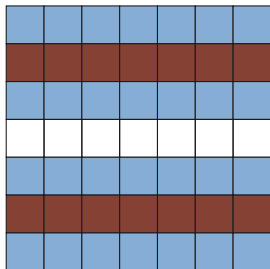
Double-Sided



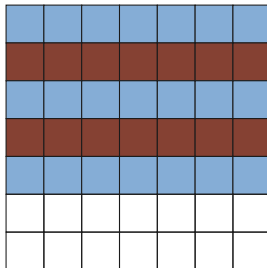
Various Hammering Patterns



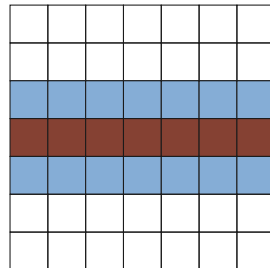
Single-Sided



Double-Sided

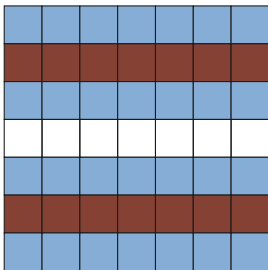


One-Location

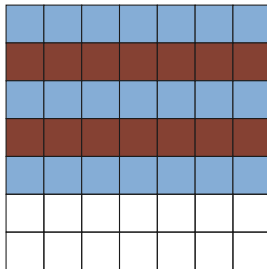


Various Hammering Patterns

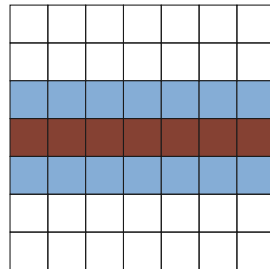
Single-Sided



Double-Sided



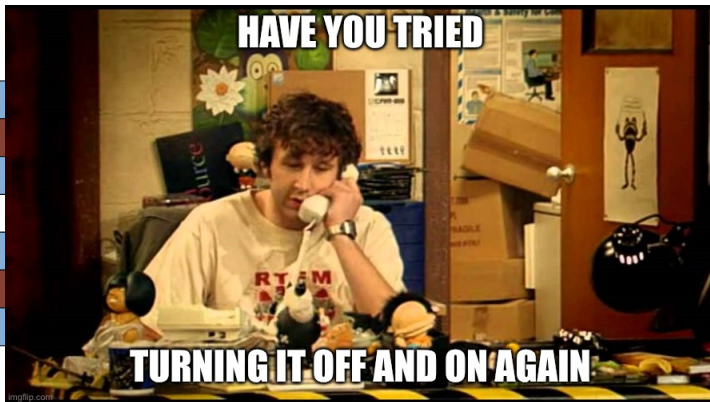
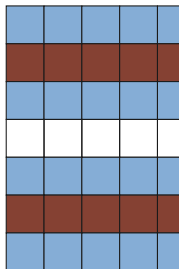
One-Location



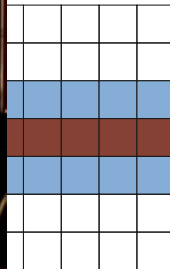
... and several more (e.g., many-sided hammering)

Various Hammering Patterns

Single-Sided



on



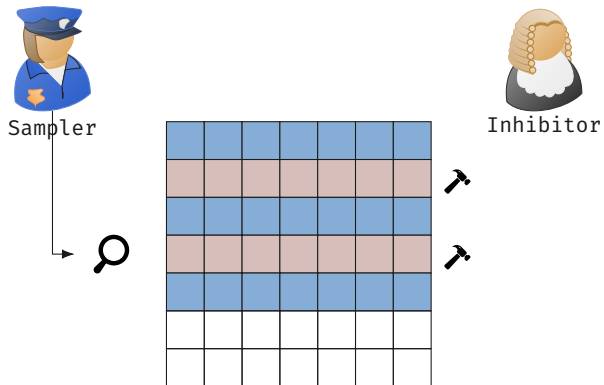


Sampler

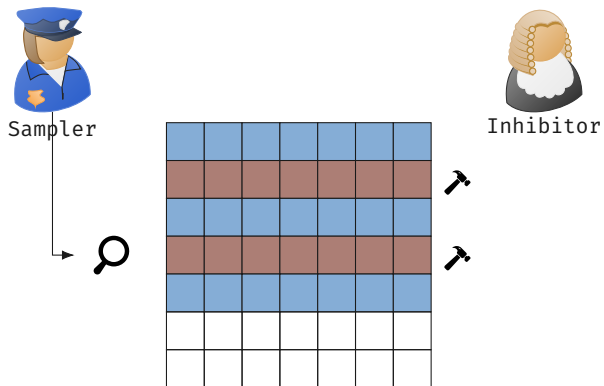
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Red	Red	Red	Red	Red	Red	Red	Red
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
Red	Red	Red	Red	Red	Red	Red	Red
Blue	Blue	Blue	Blue	Blue	Blue	Blue	Blue
White	White	White	White	White	White	White	White
White	White	White	White	White	White	White	White



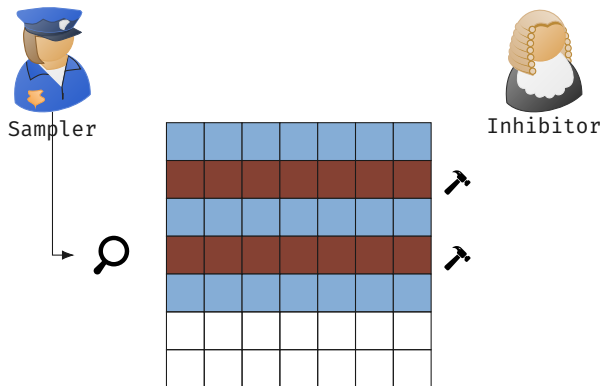
Inhibitor

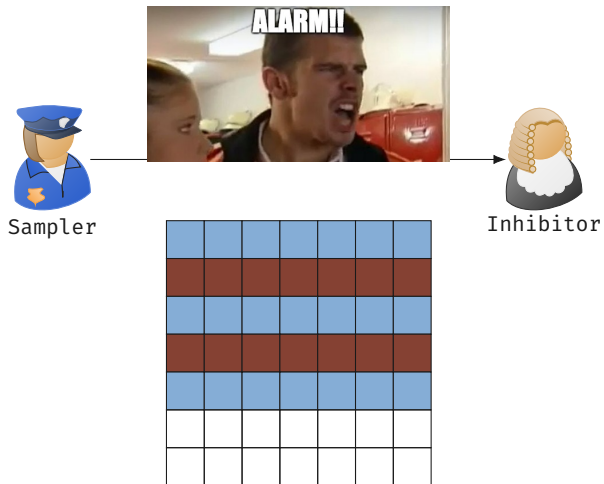


Target Row Refresh (TRR)



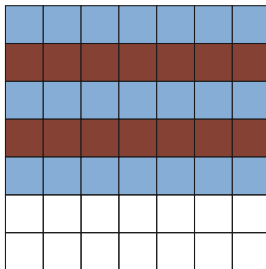
Target Row Refresh (TRR)







Sampler



Inhibitor

TRR







- Non-uniform Rowhammer Fuzzer



- Non-uniform Rowhammer Fuzzer
- Randomizes three characteristics:



- Non-uniform Rowhammer Fuzzer
- Randomizes three characteristics:

Frequency: How often the aggressor row is accessed



- Non-uniform Rowhammer Fuzzer
- Randomizes three characteristics:

Frequency: How often the aggressor row is accessed

Phase: First hammer after start of a pattern



- Non-uniform Rowhammer Fuzzer
- Randomizes three characteristics:

Frequency: How often the aggressor row is accessed

Phase: First hammer after start of a pattern

Amplitude: How many consecutive hammers



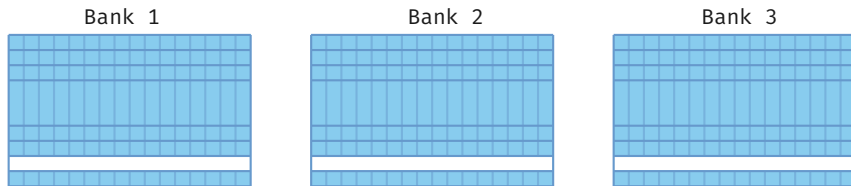
- Non-uniform Rowhammer Fuzzer
- Randomizes three characteristics:

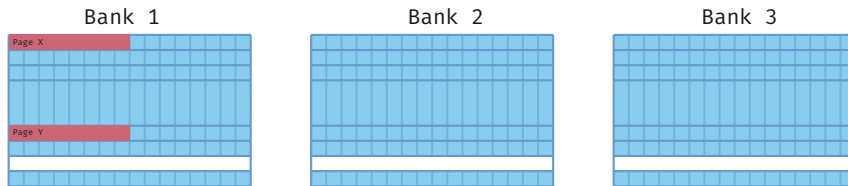
Frequency: How often the aggressor row is accessed

Phase: First hammer after start of a pattern

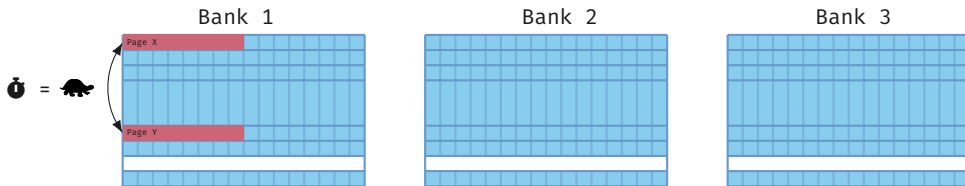
Amplitude: How many consecutive hammers

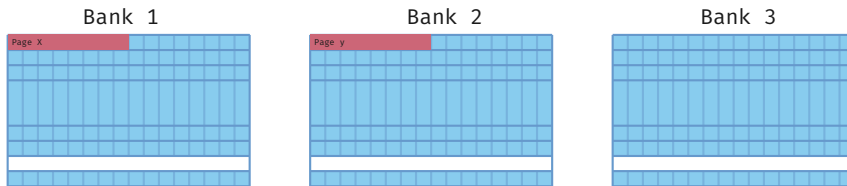
- Found bit flips in all 41 DIMMs tested



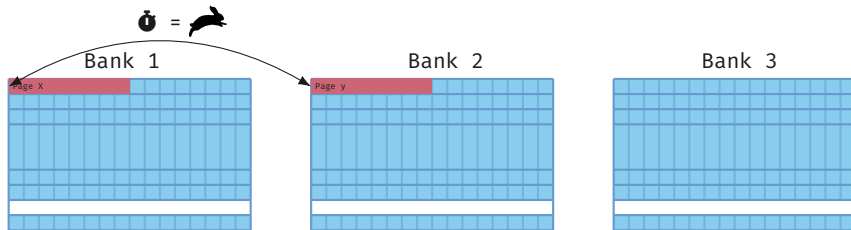


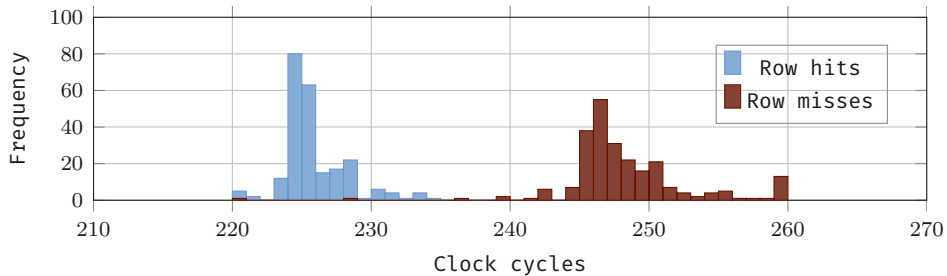
Reverse Engineering DRAM Addressing Functions



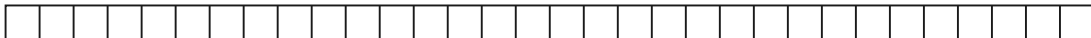


Reverse Engineering DRAM Addressing Functions



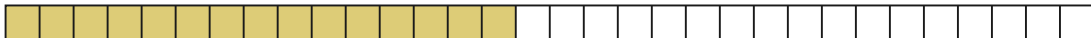


Physical Address (32 bit, $2^{32} = 4 \text{ GiB}$)



Physical Address (32 bit, $2^{32} = 4 \text{ GiB}$)

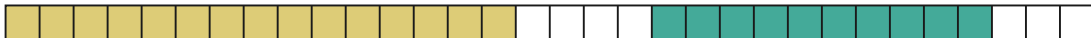
$2^{15} = 32\,768$ Rows



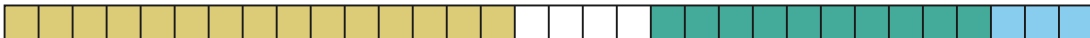
Physical Address (32 bit, $2^{32} = 4 \text{ GiB}$)

$2^{15} = 32\,768$ Rows

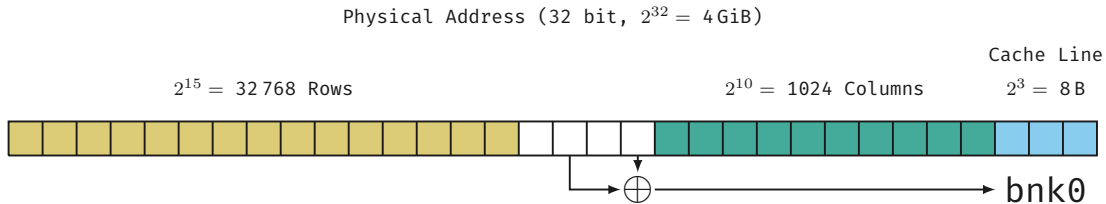
$2^{10} = 1\,024$ Columns



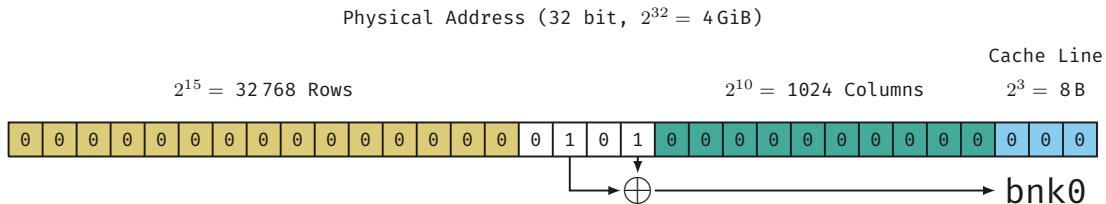
Cache Line

$$2^{10} = 1024 \text{ Columns}$$
$$2^3 = 8 \text{ B}$$


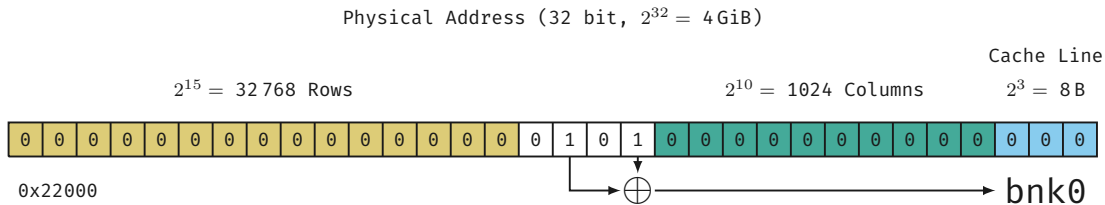
Bank Addressing Functions



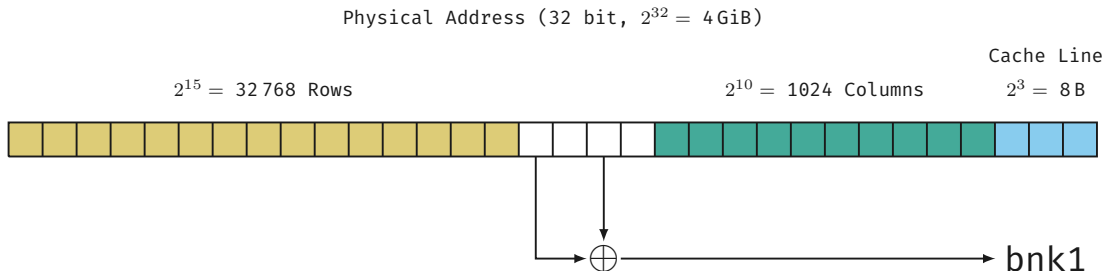
Bank Addressing Functions



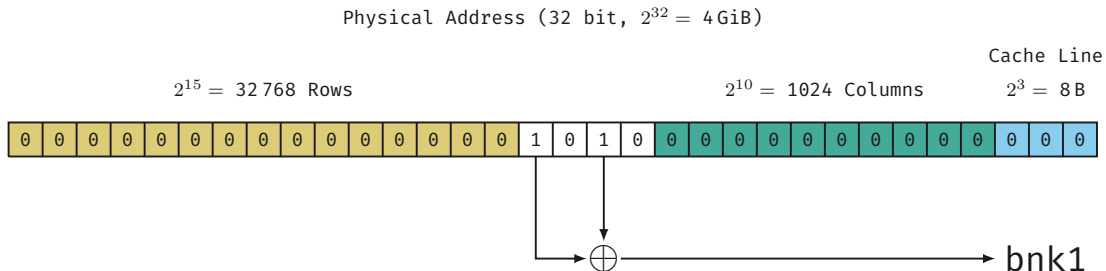
Bank Addressing Functions



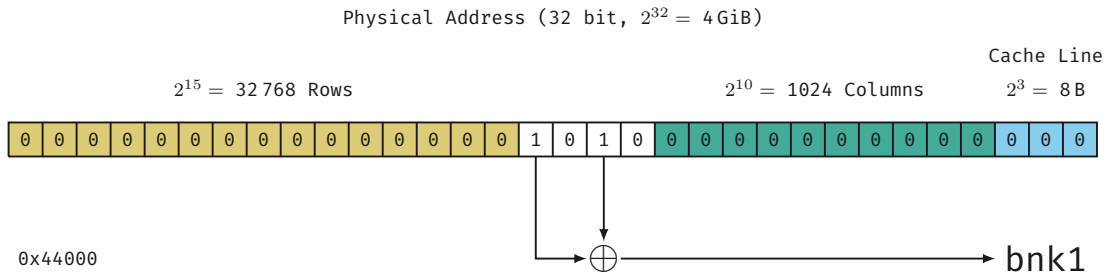
Bank Addressing Functions



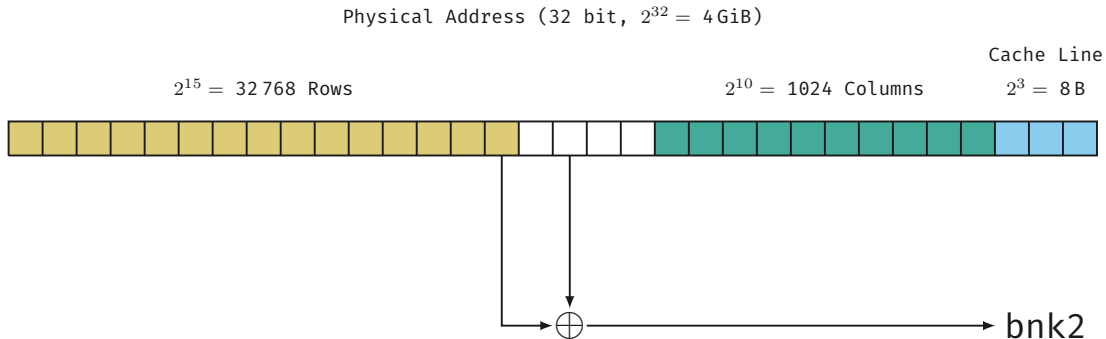
Bank Addressing Functions



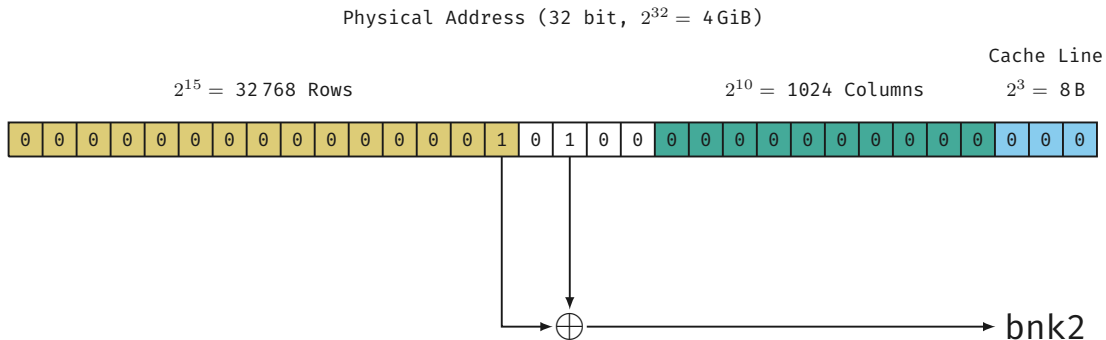
Bank Addressing Functions



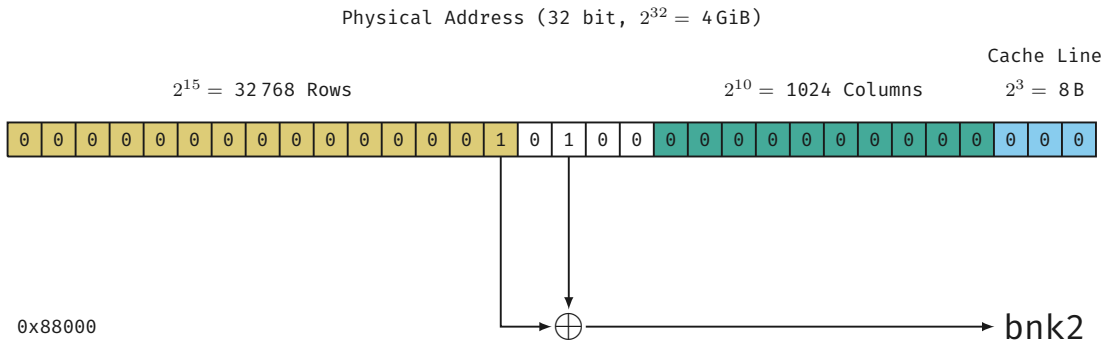
Bank Addressing Functions



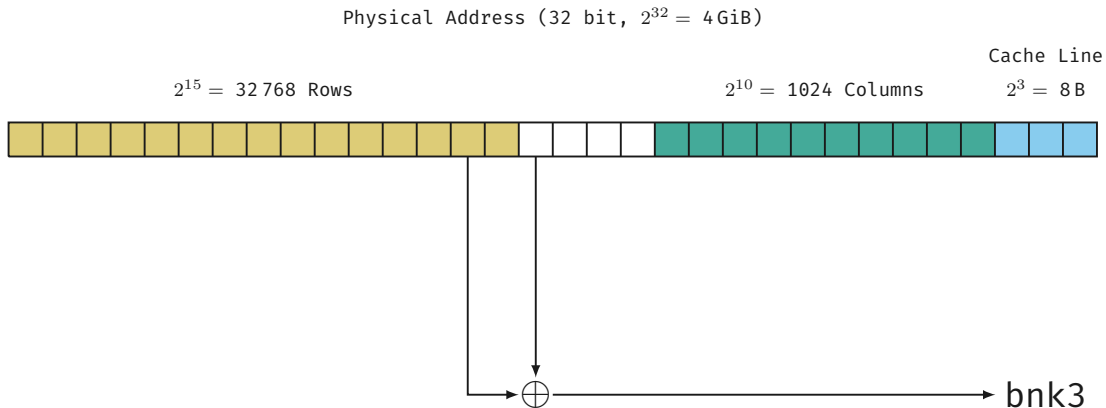
Bank Addressing Functions



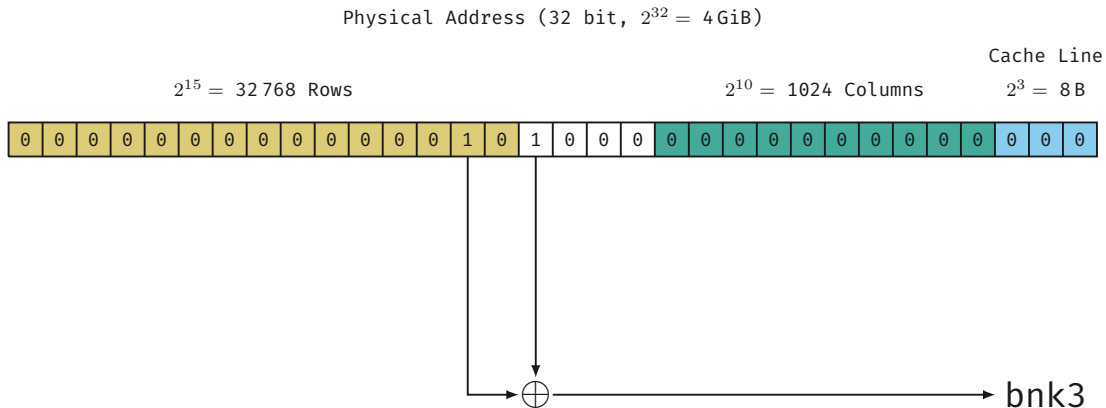
Bank Addressing Functions



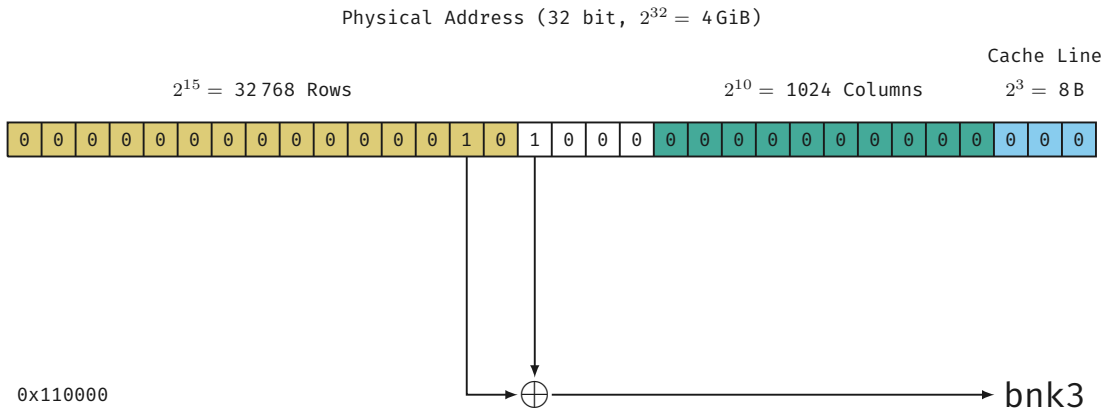
Bank Addressing Functions



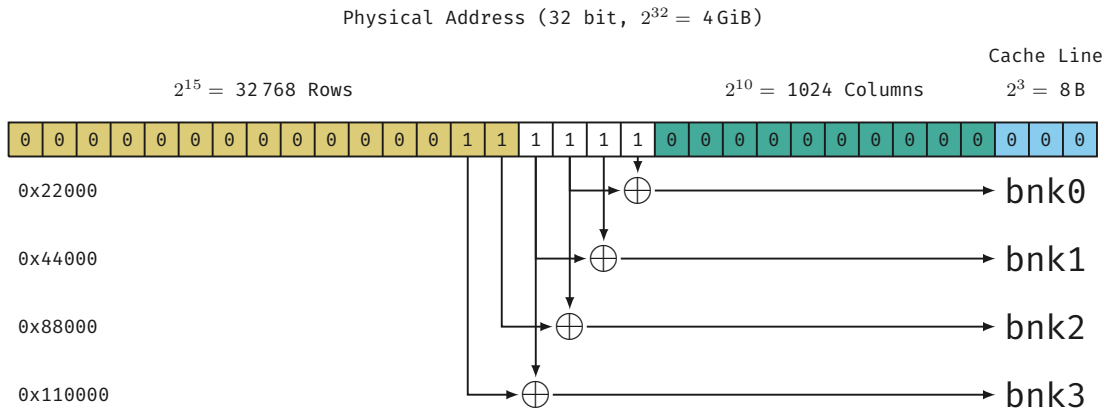
Bank Addressing Functions



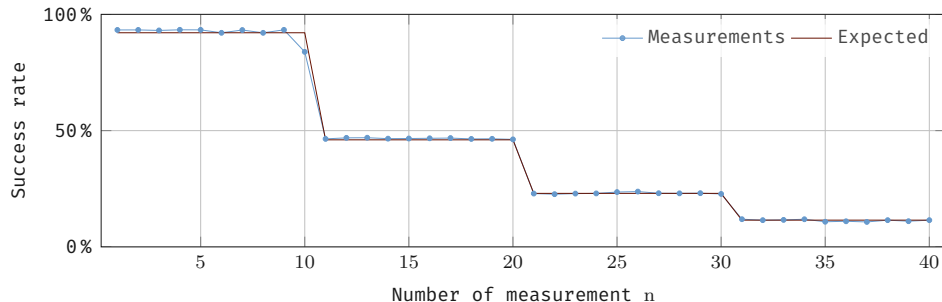
Bank Addressing Functions



Bank Addressing Functions

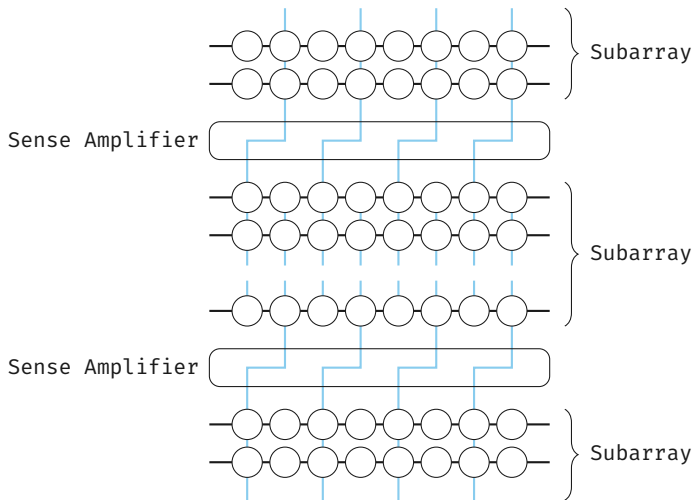


DRAM Addressing Function Verification

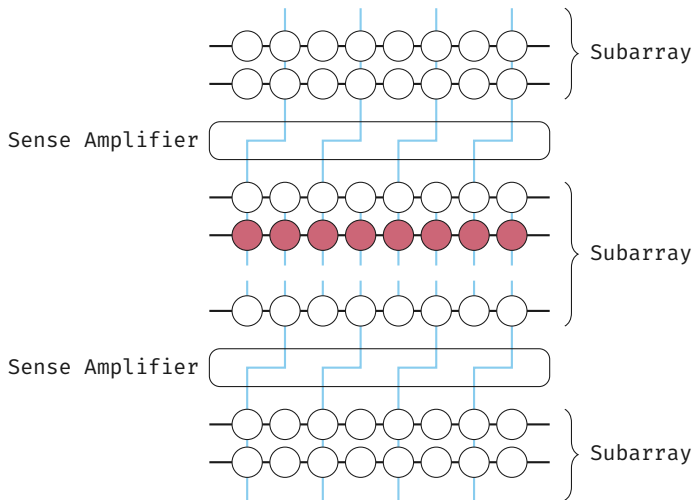


Novel Insights over the last Year

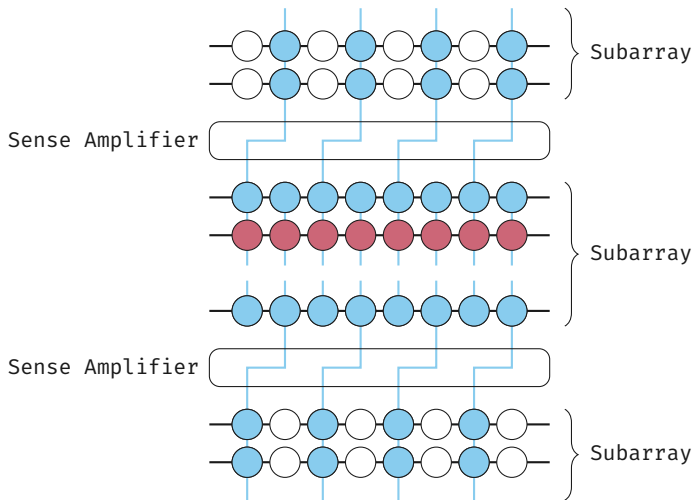
Column-based disturbance errors – ColumnDisturb



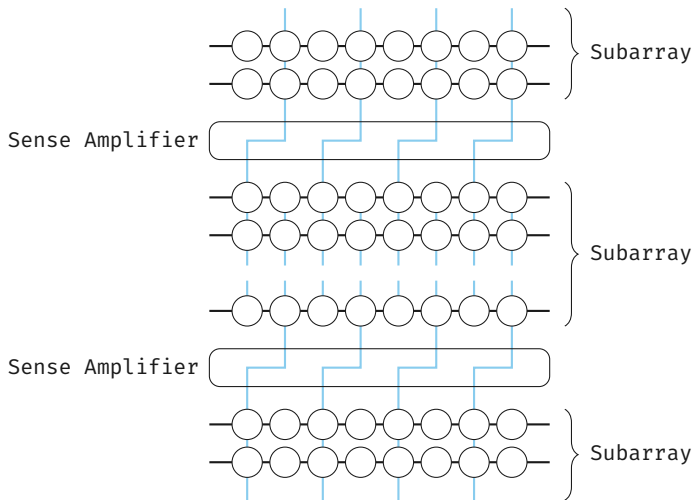
Column-based disturbance errors – ColumnDisturb



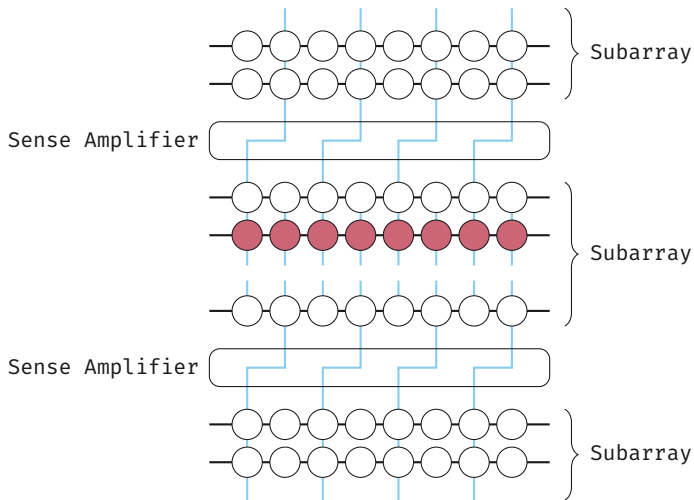
Column-based disturbance errors – ColumnDisturb



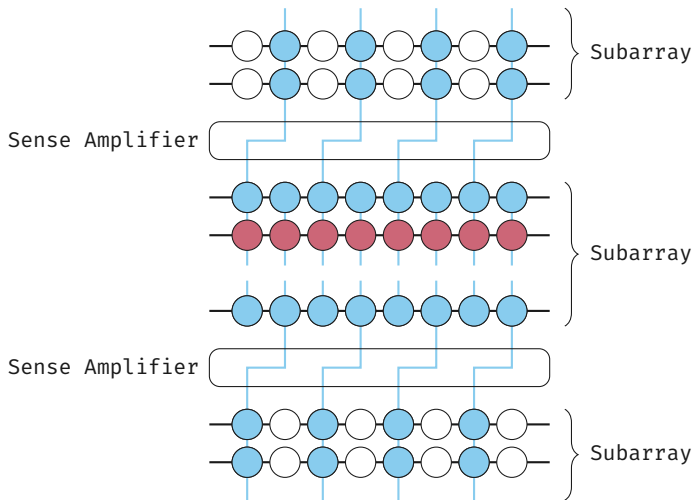
Column-based disturbance errors – ColumnDisturb



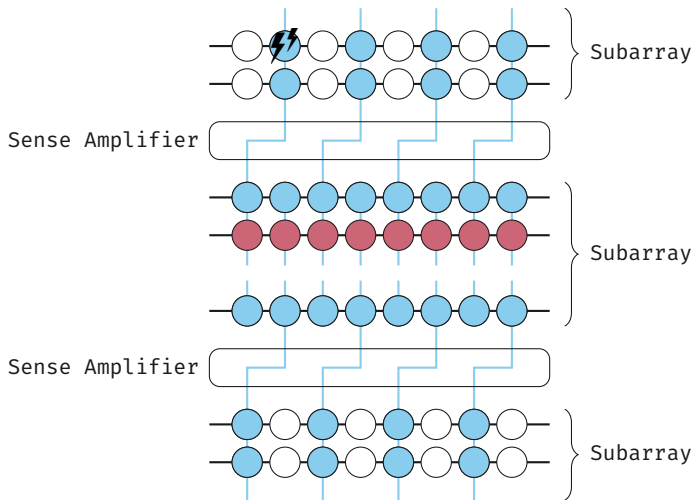
Column-based disturbance errors – ColumnDisturb

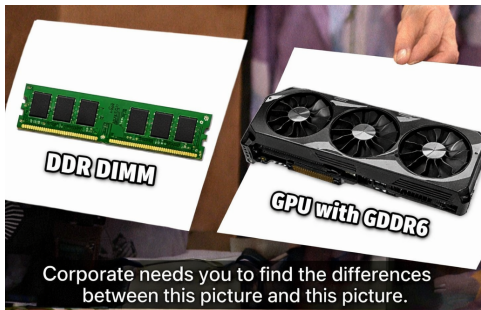


Column-based disturbance errors – ColumnDisturb

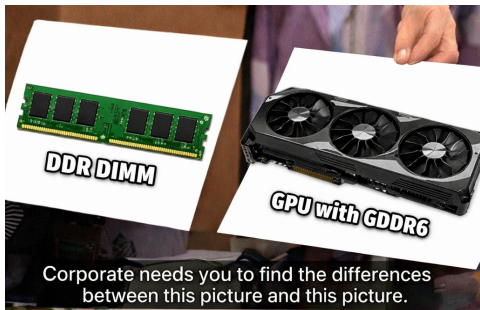


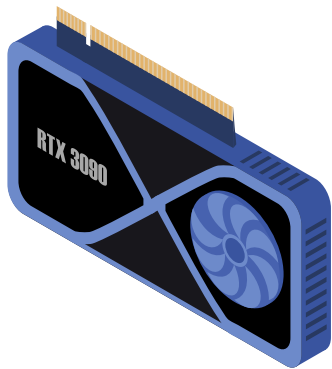
Column-based disturbance errors – ColumnDisturb



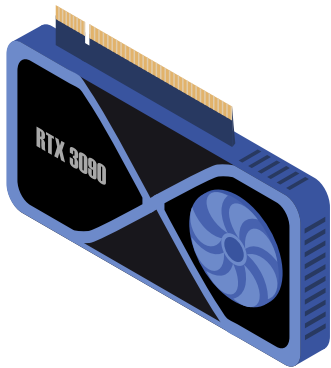


Rowhammer Attacks on GPUs – GPUHammer

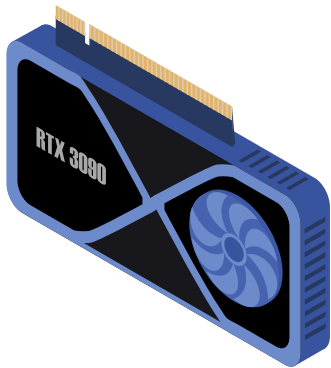




- Rowhammer attacks on GDDR6 (GPUs)



- Rowhammer attacks on GDDR6 (GPUs)
- Evicting with `discard` and `.volatile`



- Rowhammer attacks on GDDR6 (GPUs)
- Evicting with `discard` and `.volatile`
- Degrading accuracy of ML models from 80 % down to 0.02 %



- Multiple experiments on bit flip stability on a total of 98 DIMMs



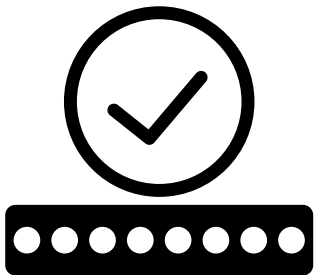
- Multiple experiments on bit flip stability on a total of 98 DIMMs
- 99.91 % fingerprinting accuracy

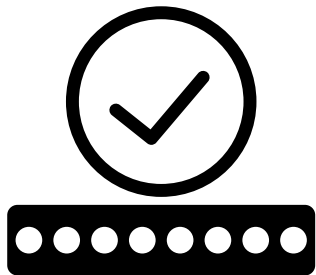


- Multiple experiments on bit flip stability on a total of 98 DIMMs
- 99.91 % fingerprinting accuracy
- Fingerprint extraction in less than 5 s

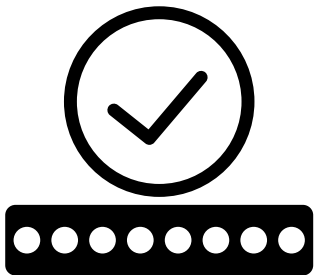


- Multiple experiments on bit flip stability on a total of 98 DIMMs
- 99.91 % fingerprinting accuracy
- Fingerprint extraction in less than 5 s
- Fingerprints were stable over a period of 10 days

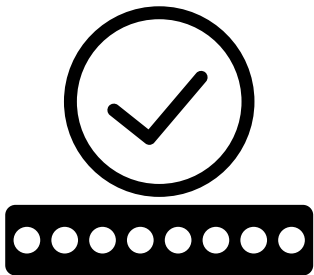




- Hynix TRR sampler can be easily confused



- Hynix TRR sampler can be easily confused
- Reverse-engineered Intel's ECC matrix on Skylake and Cascade Lake



- Hynix TRR sampler can be easily confused
- Reverse-engineered Intel's ECC matrix on Skylake and Cascade Lake
- Specific attack templates resulting in undetectable attacks

“ To allow for improved efficiency in scheduling and switching between tasks, some flexibility in the absolute refresh interval is provided for postponing and pulling-in refresh command.

”

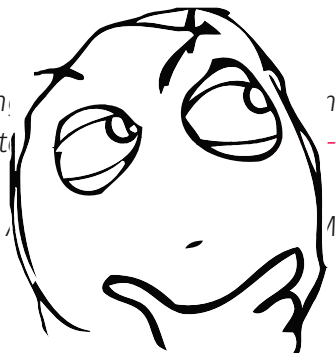
JEDEC Solid State Technology Association. DDR4 SDRAM

“ To allow for improved efficiency in scheduling and switching between tasks, some flexibility in the absolute refresh interval is provided for *post-poning and pulling-in refresh command*. ”

JEDEC Solid State Technology Association. DDR4 SDRAM

“ To allow for improved efficiency in scheduling tasks, some flexibility in the absolute refresh interval, *poning and pulling-in refresh command.*

JEDEC Solid State Technology Association

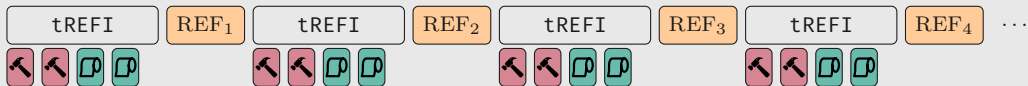


”

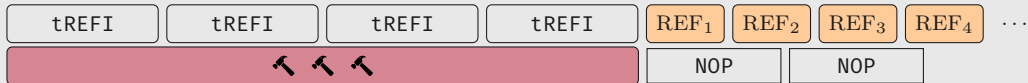
Not Postponed



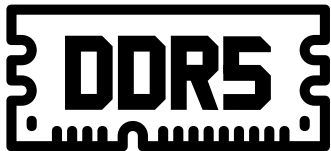
Not Postponed

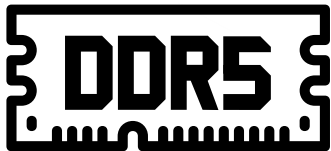


Postponed

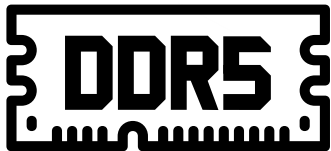




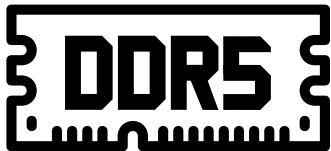




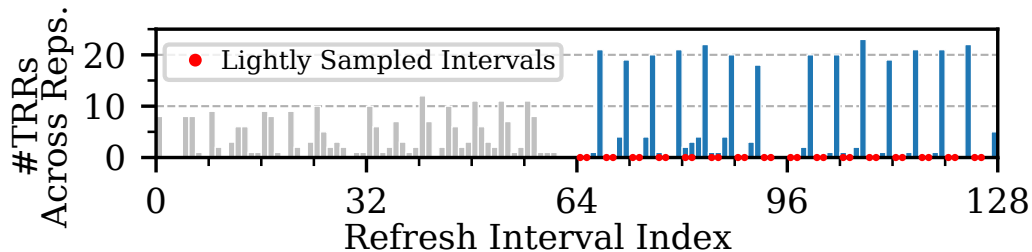
- Higher Refresh Rate (every $3.9\ \mu\text{s}$)

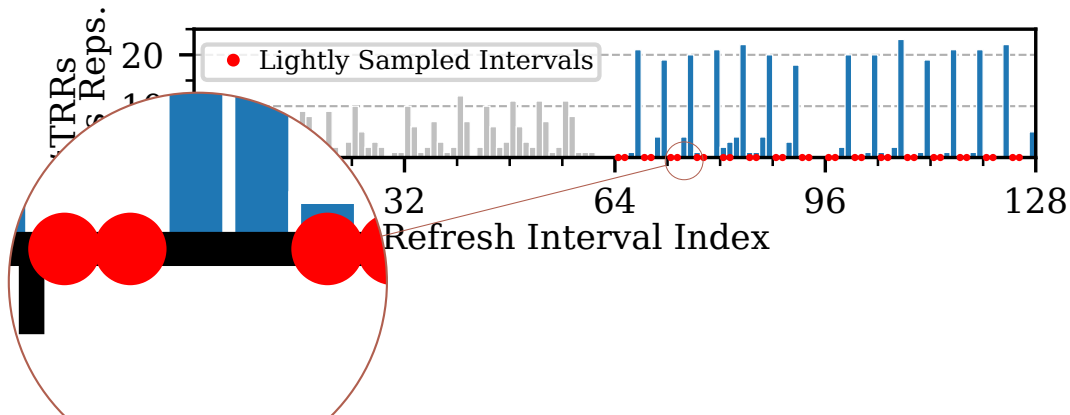


- Higher Refresh Rate (every $3.9\ \mu\text{s}$)
- Refresh Management (RFM)



- Higher Refresh Rate (every $3.9\ \mu\text{s}$)
- Refresh Management (RFM)
- On-Die ECC (ODECC)





ONE ACT, TWO ACTS, THREE ACTS

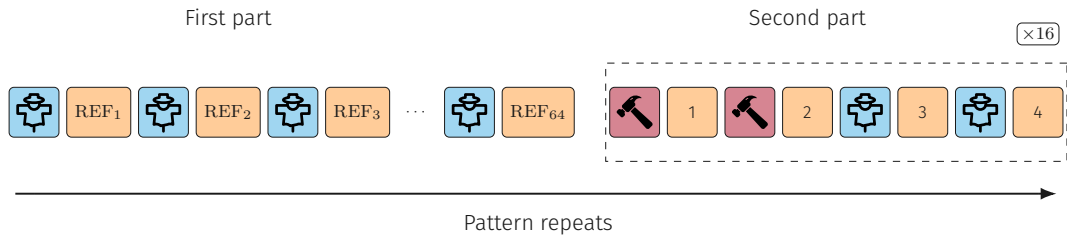
#TRRs
Across Reps.



HA HA HA

imgflip.com

New Patterns for DDR5 (Phoenix)





FlippyRAM – Large-Scale Rowhammer Study





User Agreements
Privacy Policy
Risk Agreement



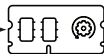
User Agreements System Information
Privacy Policy Retrieval
Risk Agreement



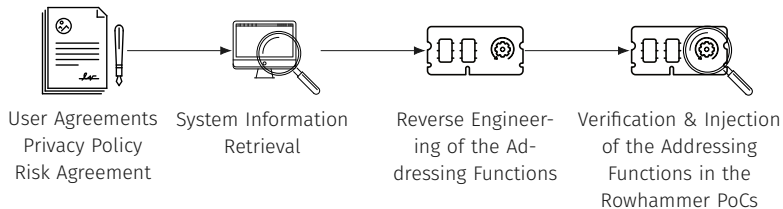
User Agreements
Privacy Policy
Risk Agreement

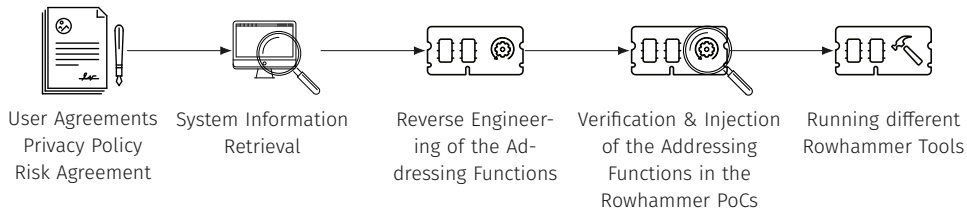


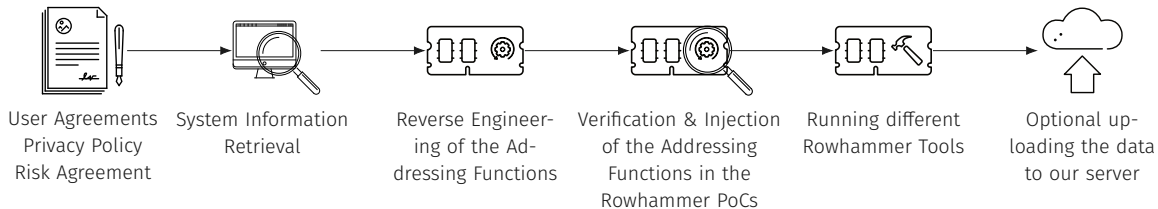
System Information
Retrieval



Reverse Engineer-
ing of the Ad-
dressing Functions







How could Users participate?





- Get a free bootable USB stick from us



How could Users participate?



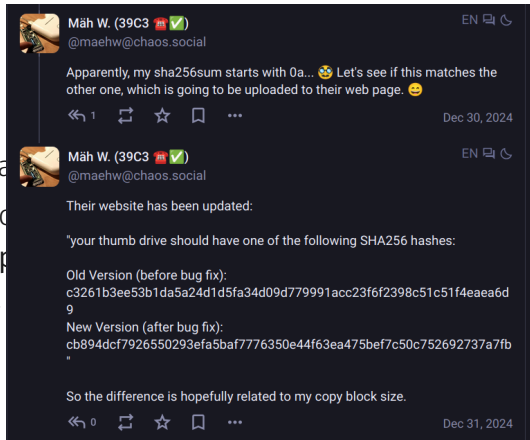


- Get a free bootable USB stick from us
- or download bootable ISO from <https://FlippyR.am> – and verify the hash either way!

How could Users participate?



- Get a
- or do
- http
- way!



either

- Flashing hundreds of USB sticks over Christmas

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas
- Cheap USB sticks

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas
- Cheap USB sticks
- Actually a small bug in the framework:

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas
- Cheap USB sticks
- Actually a small bug in the framework:
 - Plan: write a nice summary for users

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas
- Cheap USB sticks
- Actually a small bug in the framework:
 - Plan: write a nice summary for users
 - Testing first stick (took ≈ 8 hours) while flashing more $\rightarrow \approx 700$ drives done

Hash mismatches and other hickups

- Flashing hundreds thousands
- Cheap USB sticks
- Actually a small bug in the f
 - Plan: write a nice summary
 - Testing first stick (took ≈



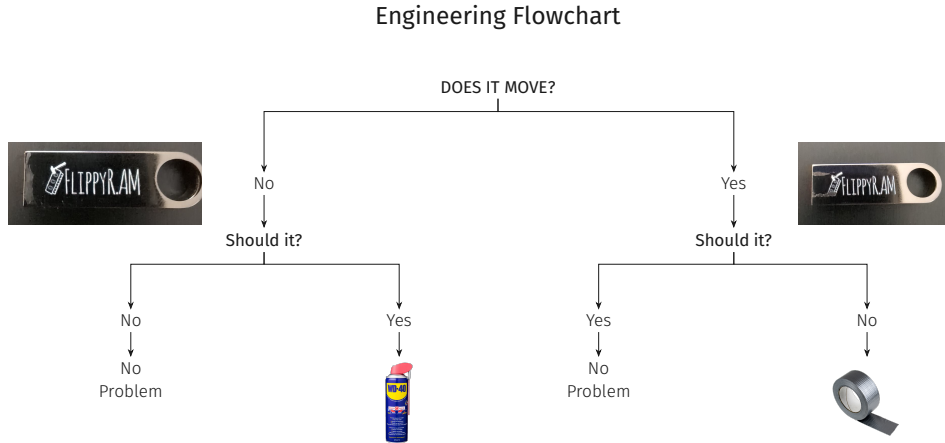
done

- Flashing ~~hundreds~~ thousands of USB sticks over Christmas
- Cheap USB sticks
- Actually a small bug in the framework:
 - Plan: write a nice summary for users
 - Testing first stick (took ≈ 8 hours) while flashing more $\rightarrow \approx 700$ drives done
 - Bug: summary is missing $\rightarrow$ Bugfix $\rightarrow$ different hash!





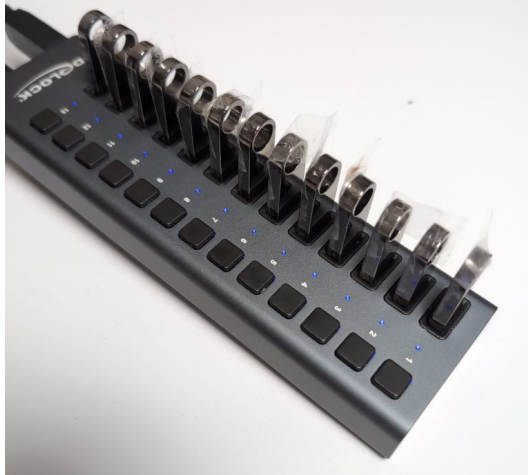




More complicated flashing Process required



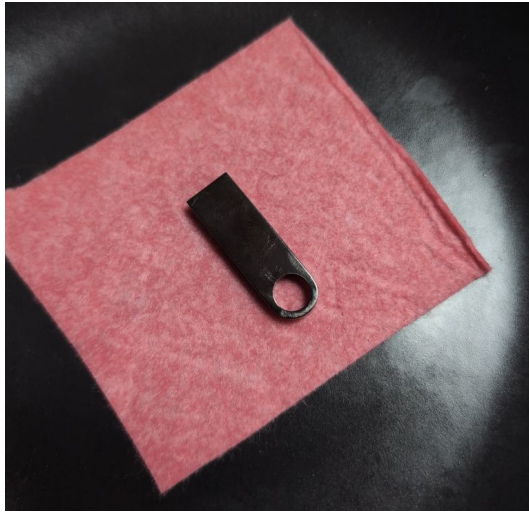
More complicated flashing Process required



More complicated flashing Process required



More complicated flashing Process required



USB Sticks: How it started



USB Sticks: How it started

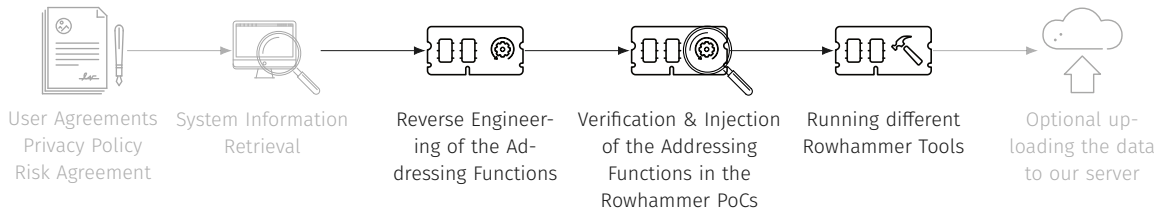


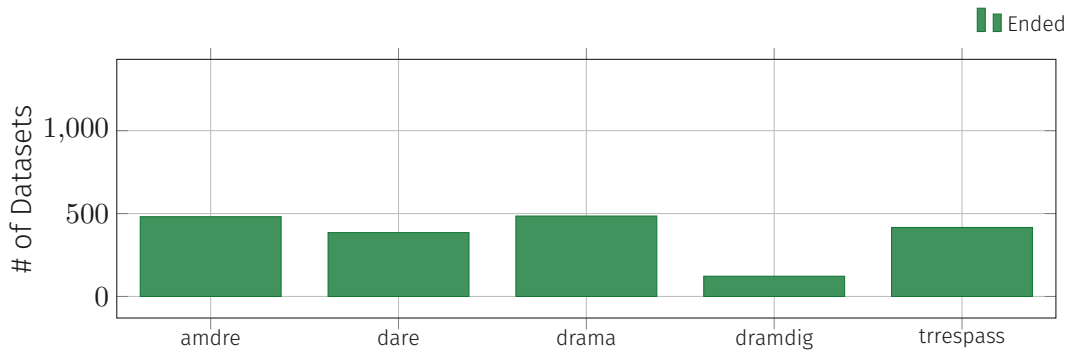
USB Sticks: How it started — How it's going

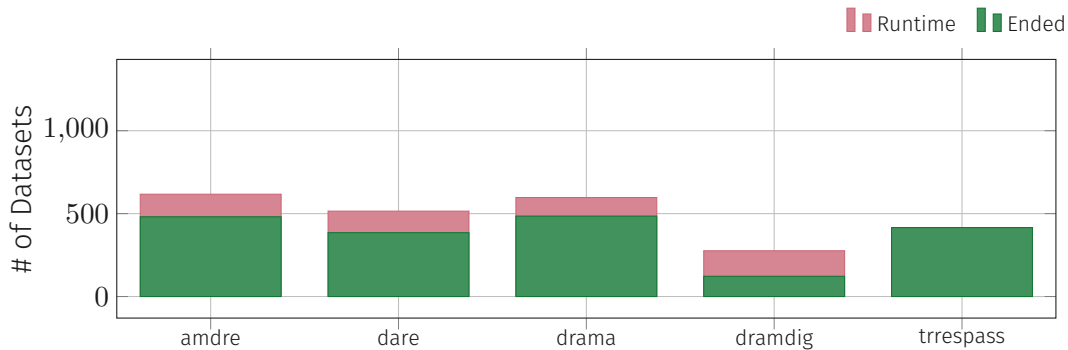


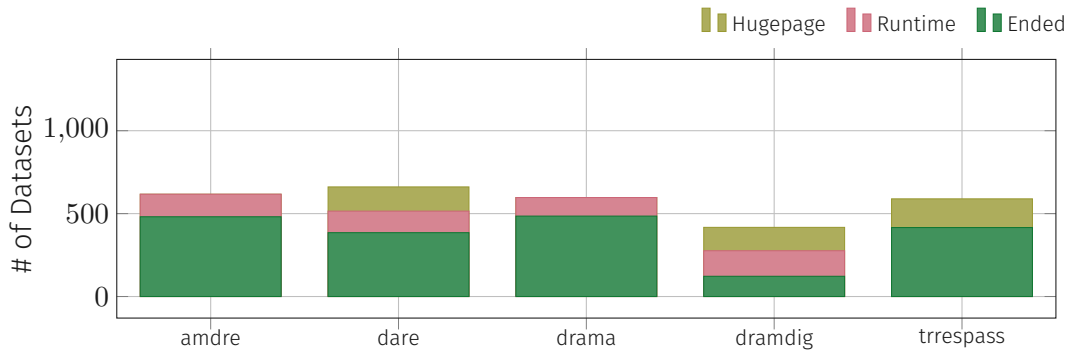


Results

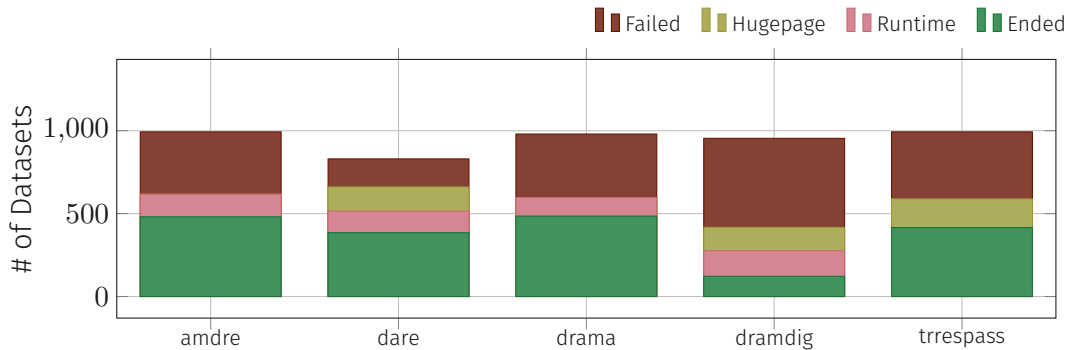




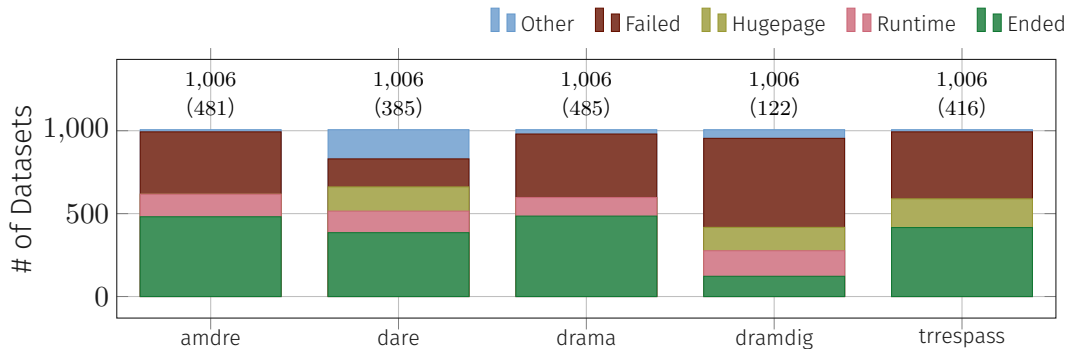




Reverse Engineering of the Addressing Functions

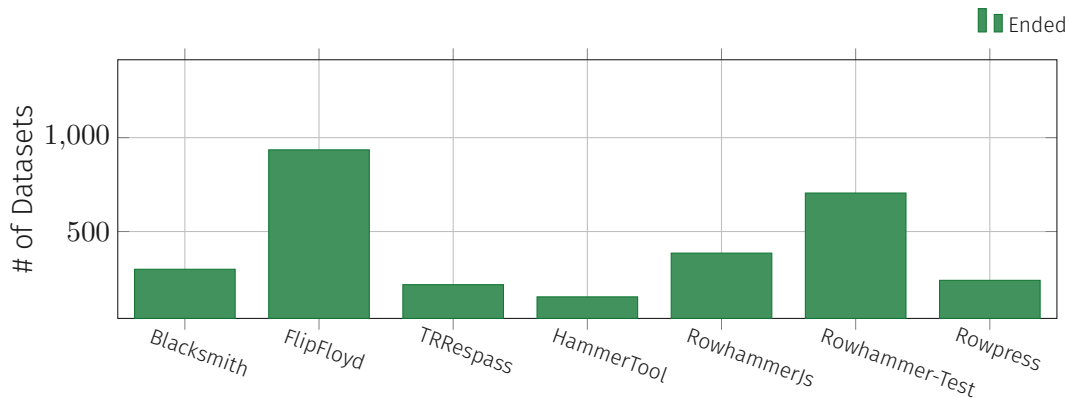


Reverse Engineering of the Addressing Functions

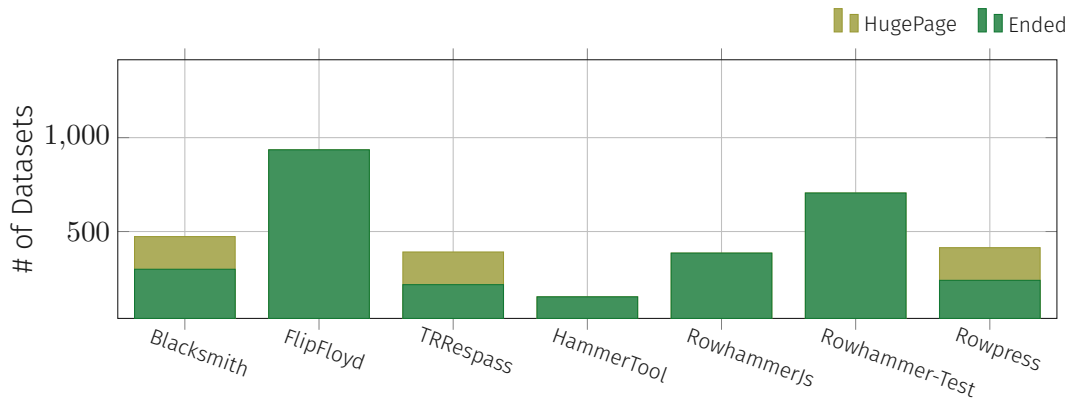


👁 (#1): Majority of cases: reverse-engineering tools fail, crash, or exceed time limits!

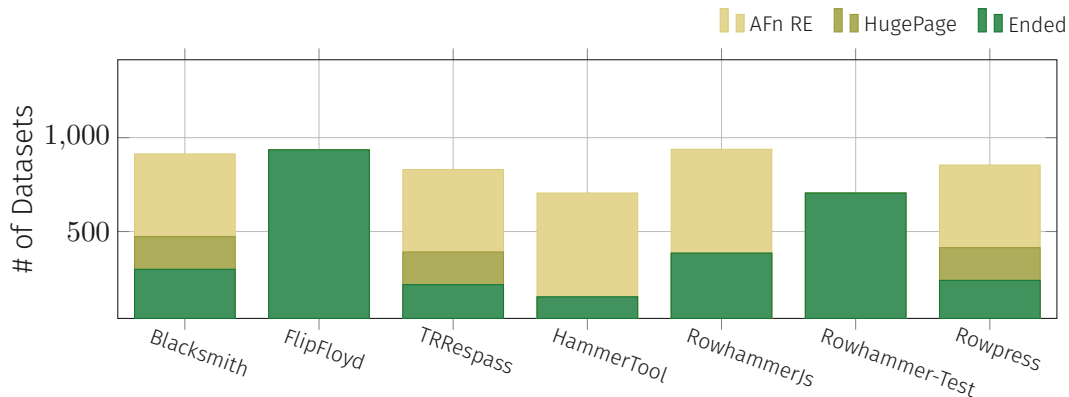
Running different Rowhammer Tools



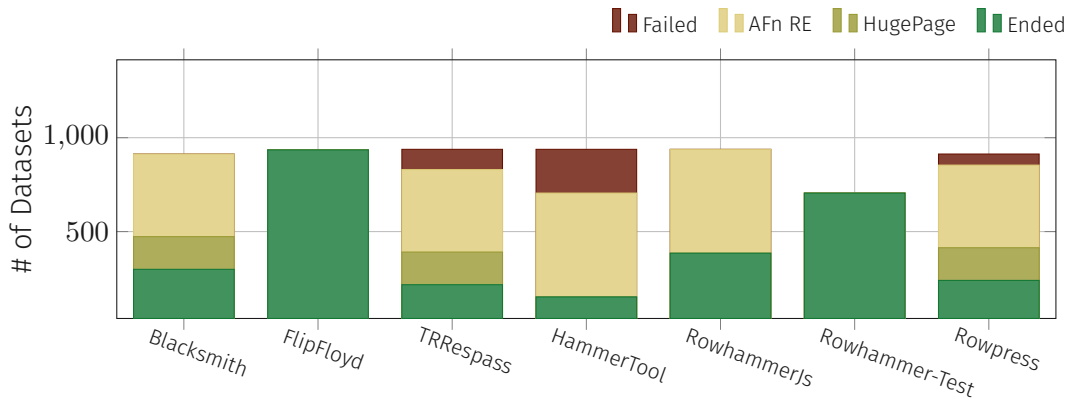
Running different Rowhammer Tools



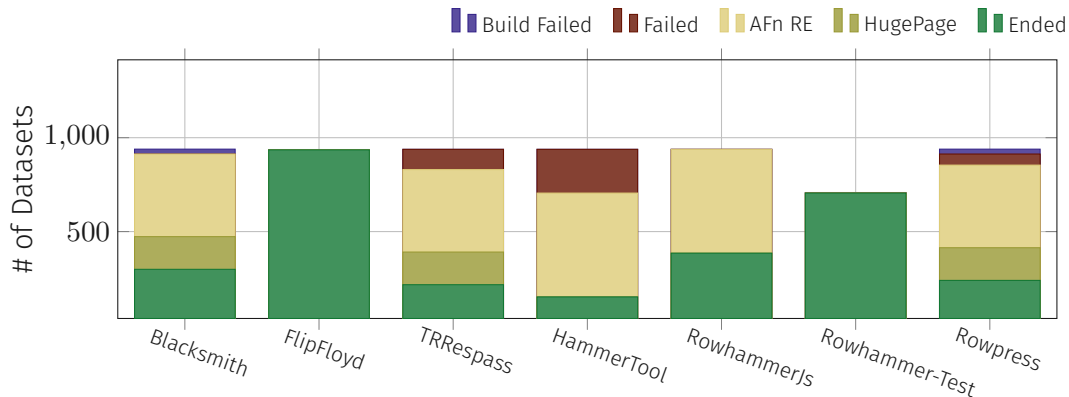
Running different Rowhammer Tools



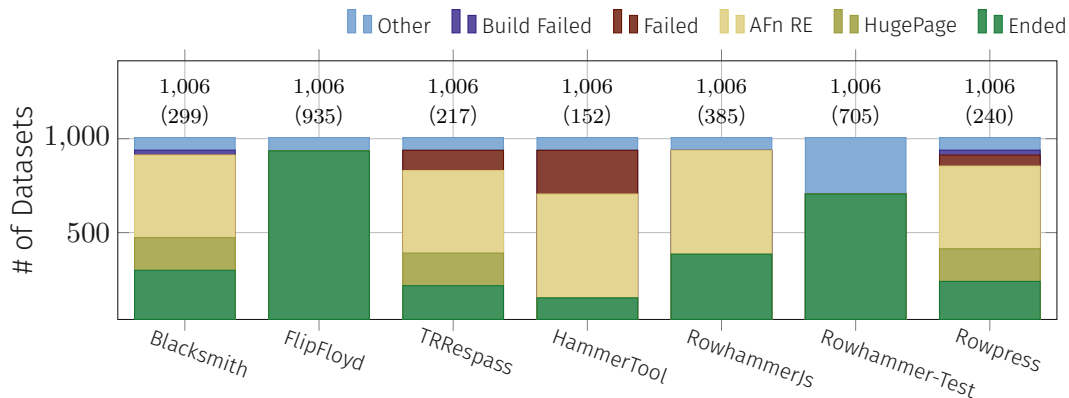
Running different Rowhammer Tools



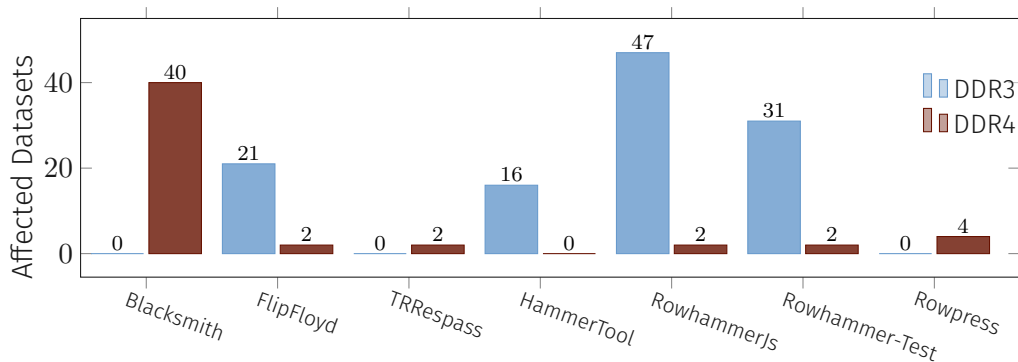
Running different Rowhammer Tools



Running different Rowhammer Tools

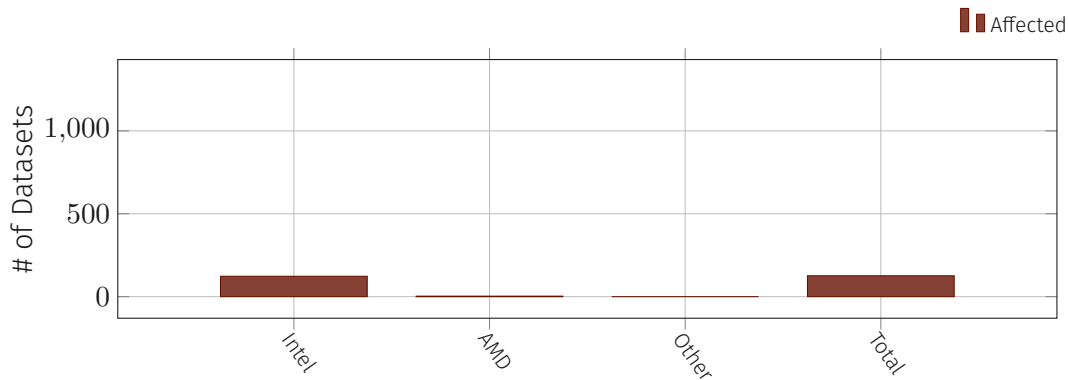


👁 (#2): Many Rowhammer tools failed because of missing DRAM functions or 1 GiB hugepages.

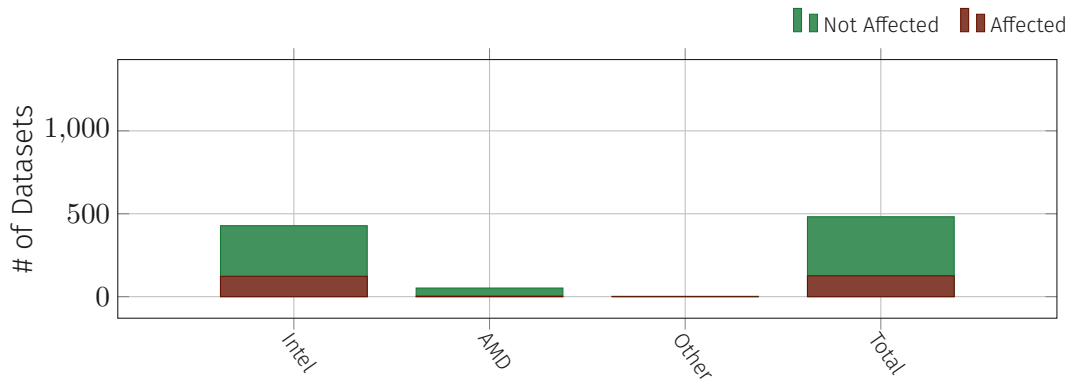


👁 (#3): 126 (12.5 %) out of 1006 datasets are vulnerable to fully-automated Rowhammer attacks!

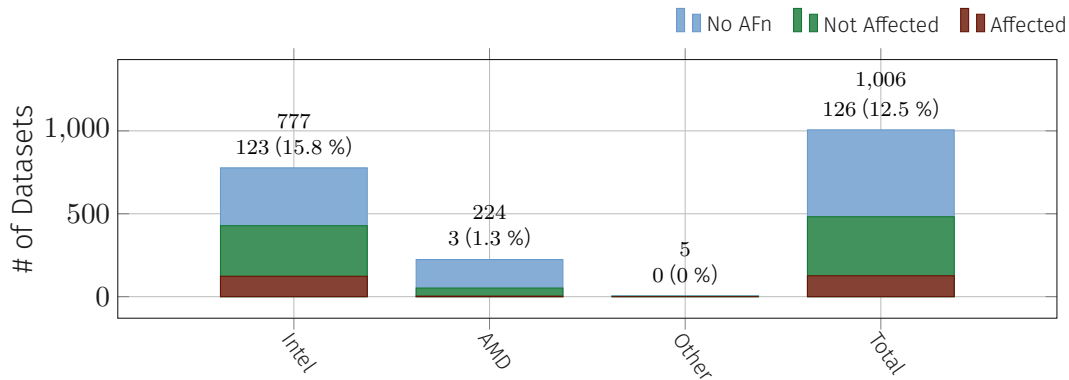
Affected systems by CPU Vendor



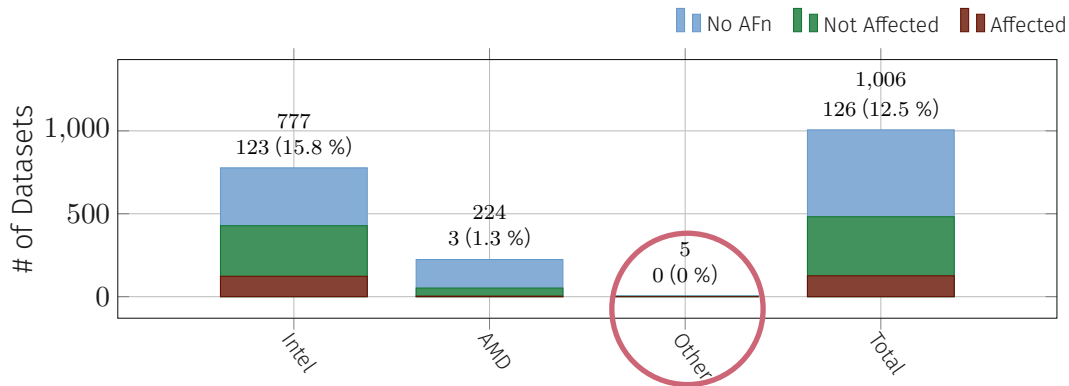
Affected systems by CPU Vendor



Affected systems by CPU Vendor

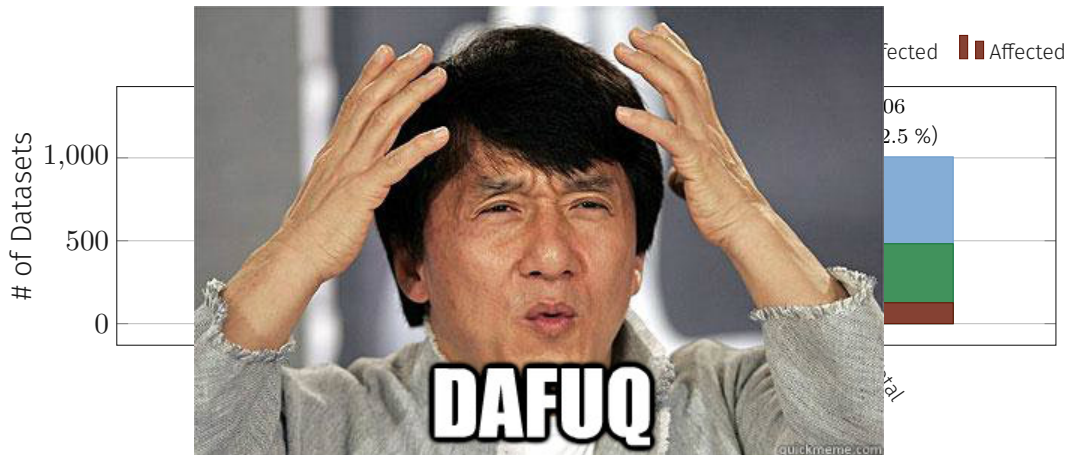


Affected systems by CPU Vendor



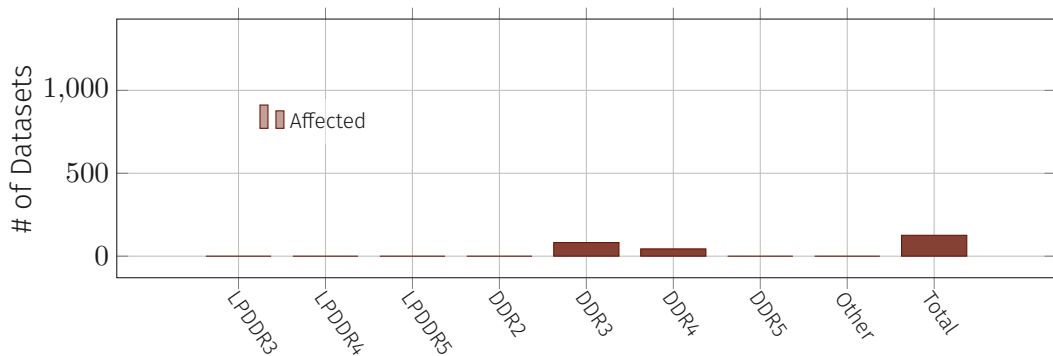
Affected systems by CPU Vendor



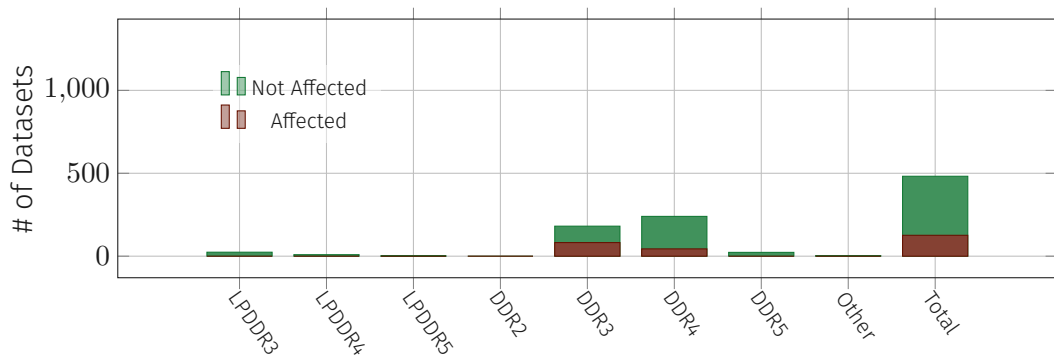


👁 (#4): Mainly tools for Intel, fewer AMD tools, especially when we started the study → we expect AMD to be equally affected.

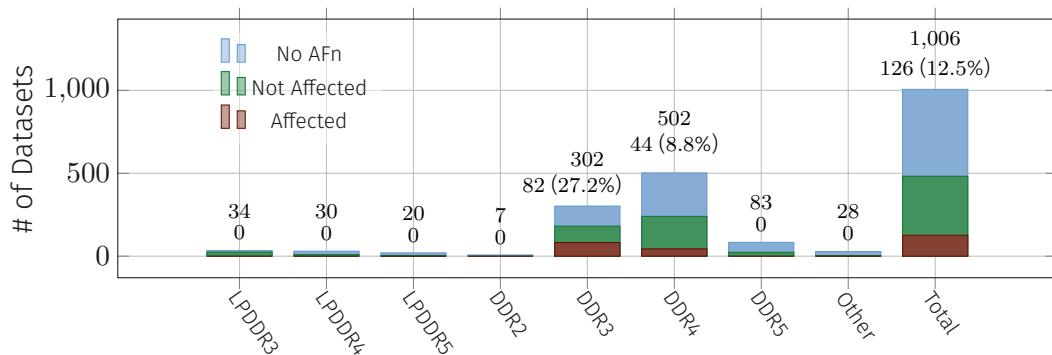
Affected DIMMs by DRAM Generation



Affected DIMMs by DRAM Generation

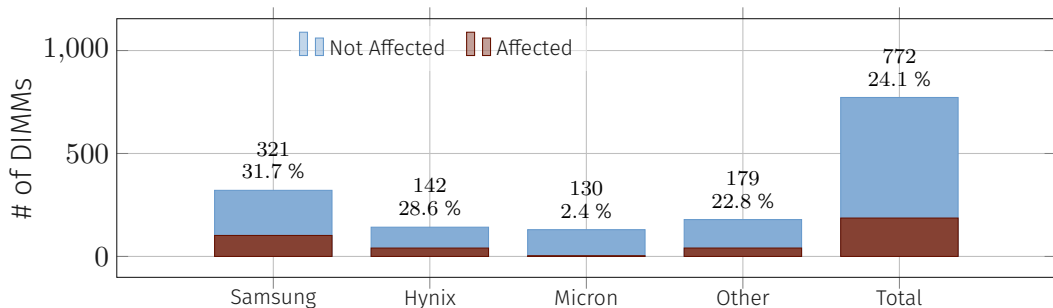


Affected DIMMs by DRAM Generation



👁 (#5): DDR3 → simple fast patterns (RowhammerJS);
DDR4 with TRR → pattern fuzzing for non-uniform patterns
(Blacksmith)

Affected DIMMs by DRAM Vendor



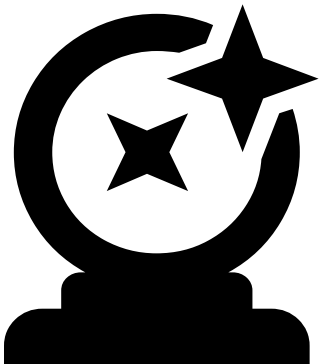
👁 (#6): DRAM from Samsung, Hynix, and third-party resellers similarly affected by Rowhammer but only 2.4 % of Micron DIMMs?

Rowhammer in the Future

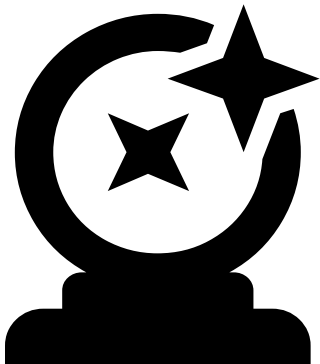




- More proprietary mitigations, similar to TRR



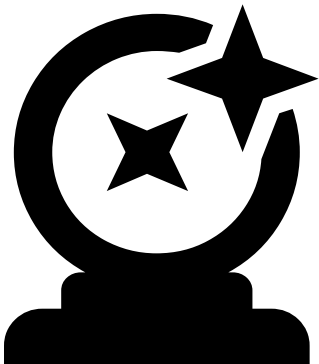
- More proprietary mitigations, similar to TRR
- **Problem:** Nobody can verify how good/effective they are



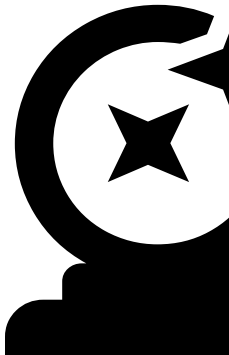
- More proprietary mitigations, similar to TRR
- **Problem:** Nobody can verify how good/effective they are
- The cat-and-mouse game will continue



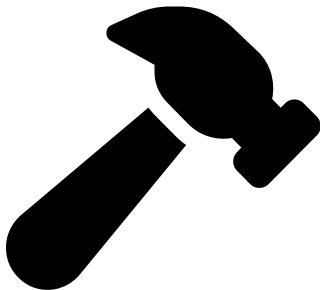
ons, similar to TRR
ify how good/effective
e will continue

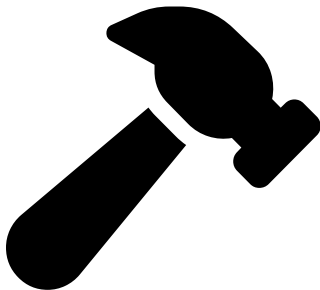


- More proprietary mitigations, similar to TRR
- **Problem:** Nobody can verify how good/effective they are
- The cat-and-mouse game will continue
- **Solution:** Publish mitigation algorithms so they can be analyzed by everybody

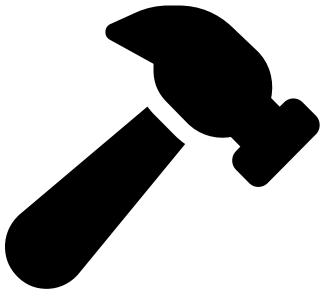


similar to TRR
w good/effective
continue
gorithms so they

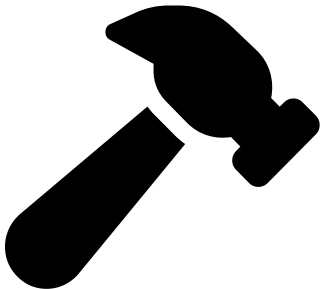




- Better addressing function reverse-engineering



- Better addressing function reverse-engineering
- Better tools on different platforms

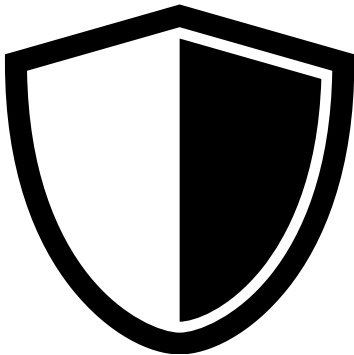


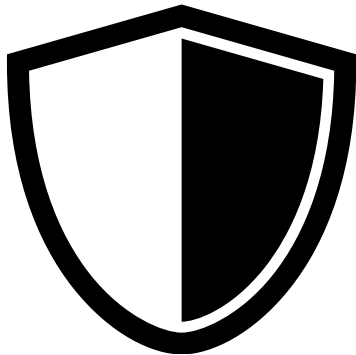
- Better addressing function reverse-engineering
- Better tools on different platforms
- More exploits



se-engineering
ms

- Process “improvements”



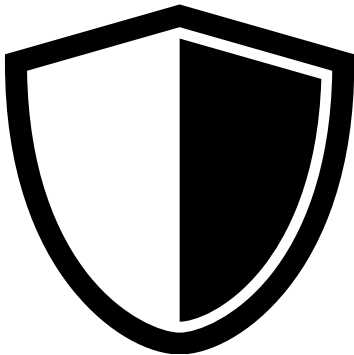


- Process “improvements”
 - back to 300nm? more distance between cells?

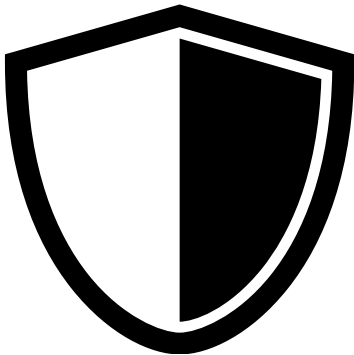
- Process “improvements”



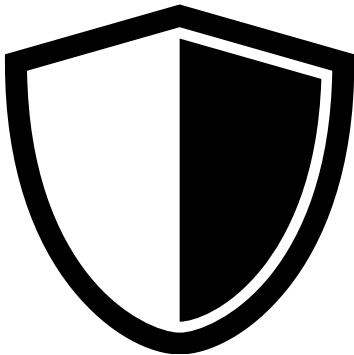
te between cells?



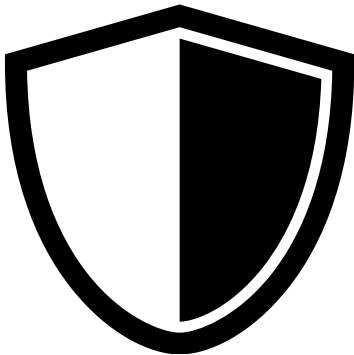
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)



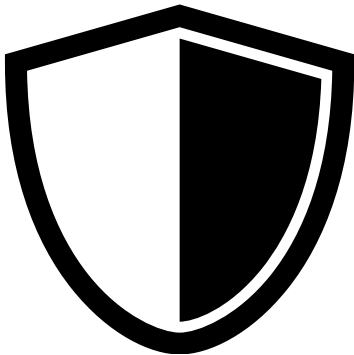
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant



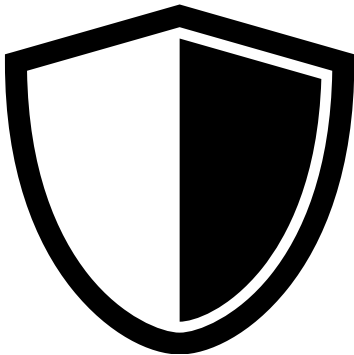
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb



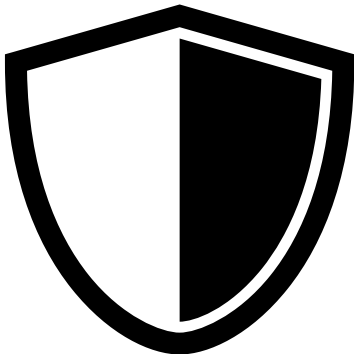
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)



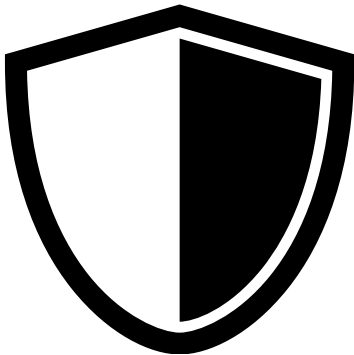
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)
 - Increase refresh rate, TRR, ...,



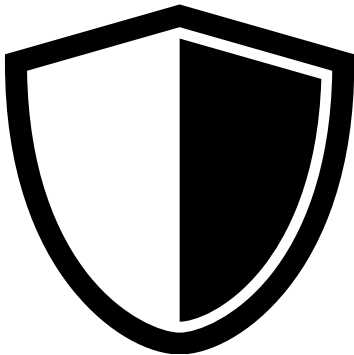
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)
 - Increase refresh rate, TRR, ..., **Memory Band-Aid**



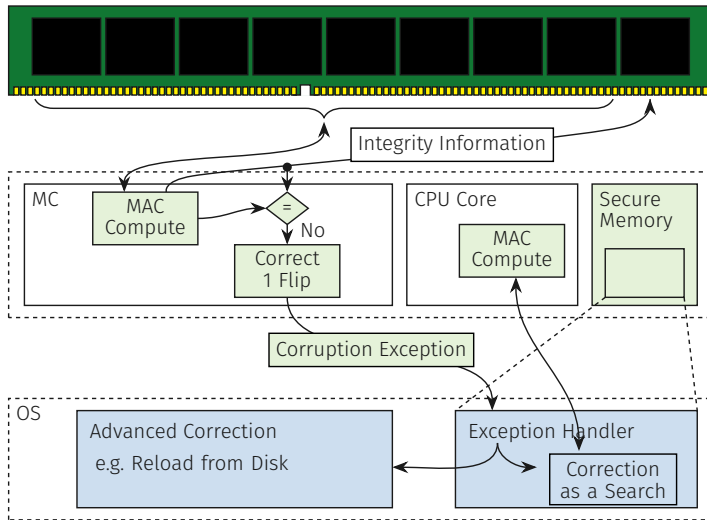
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)
 - Increase refresh rate, TRR, ..., **Memory Band-Aid**
- Integrity Protection



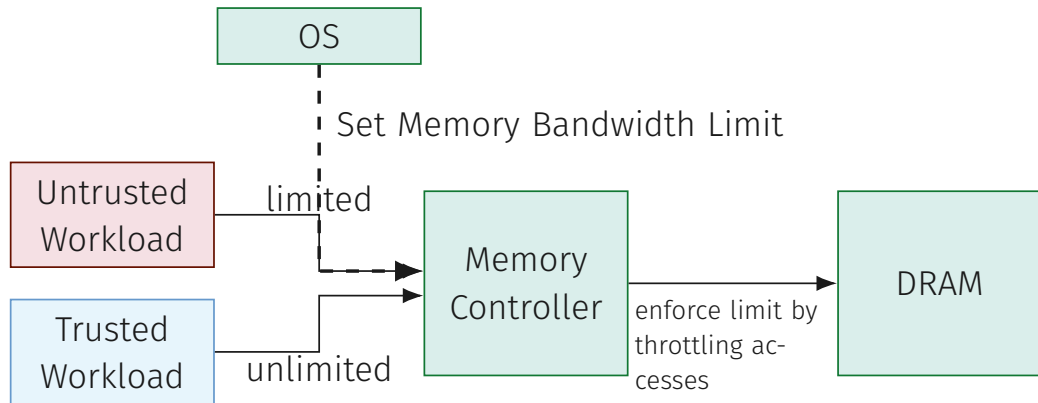
- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)
 - Increase refresh rate, TRR, ..., **Memory Band-Aid**
- Integrity Protection
 - ECC, more ECC, Chipkill, ...,

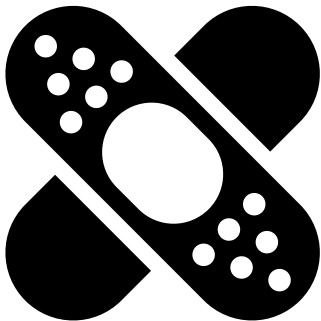


- Process “improvements”
 - back to 300nm? more distance between cells?
- Spatial isolation (e.g., CATT, ZebRAM, ...)
 - Problem 1: bit flips sometimes more distant
 - Problem 2: ColumnDisturb
- Changing ratio between memory accesses and refreshes (more refreshes!)
 - Increase refresh rate, TRR, ..., **Memory Band-Aid**
- Integrity Protection
 - ECC, more ECC, Chipkill, ..., **CSl:Rowhammer**

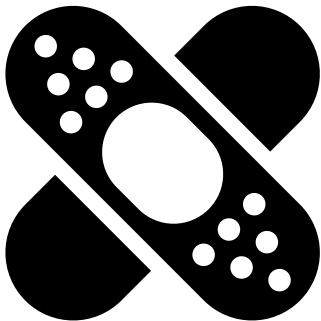


# Errors	# MAC Comp.	Avg Duration
1	17	11 ns
2	771	3.68 μ s
3	33 800	124 μ s
4	1.51×10^6	6.65 ms
5	6.91×10^7	261 ms
6	3.07×10^9	12.8 s
7	1.21×10^{11}	9.11 min
8	5.72×10^{12}	6.11 h

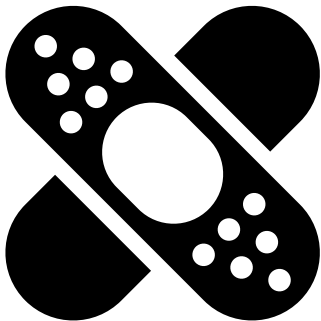




- Requires hardware changes



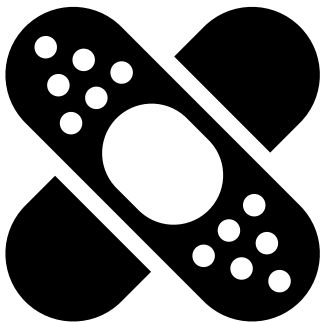
- Requires hardware changes
- Per-bank limit



- Requires hardware changes
- Per-bank limit
- Per-process limit



- Requires hardware changes
 - Per-bank limit
 - Per-process limit
- no overhead for trusted processes



- Requires hardware changes
 - Per-bank limit
 - Per-process limit
- no overhead for trusted processes
- Other processes: 0 % to 9.4 % in macro-benchmarks





- it is easy to add new tools



- it is easy to add new tools
- ... and test your additions on many systems



- it is easy to add new tools
- ... and test your additions on many systems
- ... and compare against existing tools



- it is easy to add new tools
- ... and test your additions on many systems
- ... and compare against existing tools
- ... and help others reproduce your results



- Rowhammer can be weaponized today ($\approx 12.5\%$ of systems)



- Rowhammer can be weaponized today ($\approx 12.5\%$ of systems)
- $\approx 2x$ with addressing functions and $2x$ with CPU diversity solved
→ $\approx 50\%$ of systems



- Rowhammer can be weaponized today ($\approx 12.5\%$ of systems)
- $\approx 2x$ with addressing functions and $2x$ with CPU diversity solved
→ $\approx 50\%$ of systems
- Defense game needs to change:



- Rowhammer can be weaponized today ($\approx 12.5\%$ of systems)
- $\approx 2x$ with addressing functions and $2x$ with CPU diversity solved
→ $\approx 50\%$ of systems
- Defense game needs to change:
 - New defense, secret parameters → find out parameters → bypass



- Rowhammer can be weaponized today ($\approx 12.5\%$ of systems)
- $\approx 2x$ with addressing functions and $2x$ with CPU diversity solved
→ $\approx 50\%$ of systems
- Defense game needs to change:
 - ~~Security by obscurity~~

Rowhammer in the Wild

Large-Scale Insights from FlippyR.AM

Martin Heckel^{1,2} (@lunkw1ll)

Daniel Gruss¹ (@lavados)

Florian Adamsky² (@c1t)

¹ Graz University of Technology

² Hof University of Applied Sciences



Kofinanziert von der Europäischen Union



- Toilet Paper Blank from Slide 20 is from Dazzle UI and is under the CC Attribution License