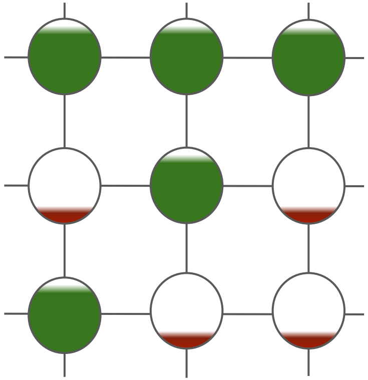
TRRespass: Exploiting the Many Sides of Target Row Refresh

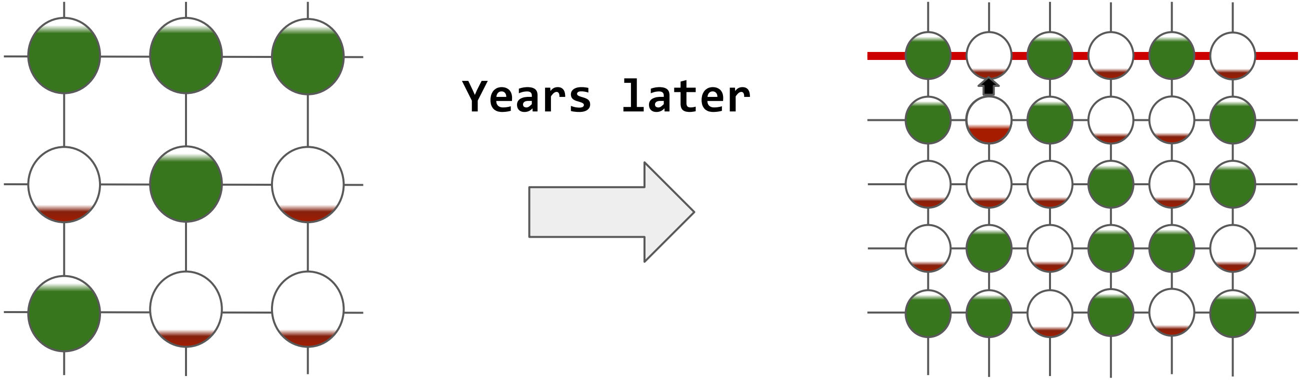
Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen
Onur Mutlu, Cristiano Giuffrida, Herbert Bos, **Kaveh Razavi**

DRAM



The Rowhammer problem

We have reduced transistor without caring for reliability/security



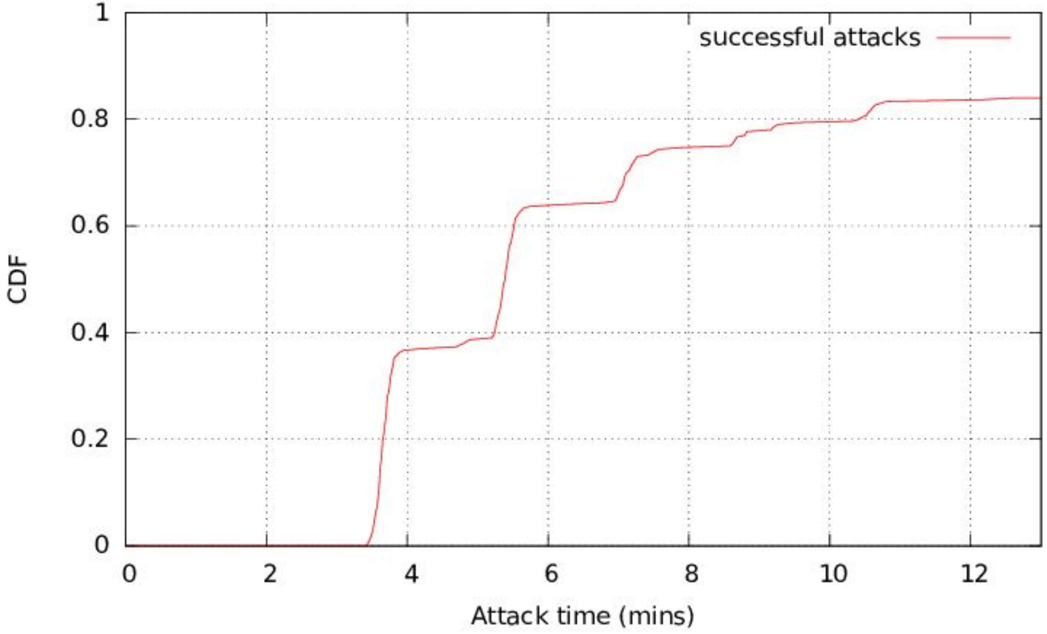
Rowhammer: affects 87% of deployed DDR3 memory.

Kim et al., “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,” ISCA’14

So what can you do with it?

root@home:#

In the cloud



Razavi et al., “Flip Feng Shui: Hammering a Needle in the Software Stack,” SEC’16

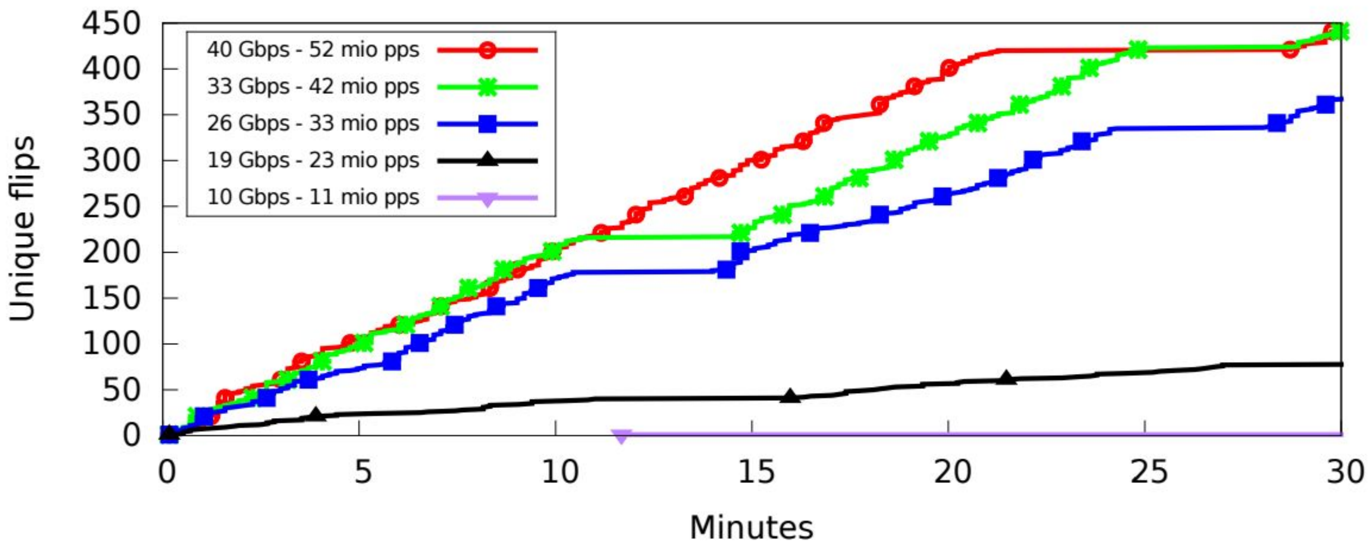
Rooting Android phones

Device	#flips	1st exploitable flip after
LG Nexus 5[1]	1058	116s
LG Nexus 5[4]	0	-
LG Nexus 5[5]	747,013	1s
LG Nexus 4	1,328	7s
OnePlus One	3,981	942s
Motorola Moto G (2013)	429	441s
LG G4 (ARMv8 – 64-bit)	117,496	5s

22 seconds to root on 18 out of 27 tested phones.

Van der Veen et al., "Drammer: Deterministic Rowhammer Attacks on Mobile Phones," CCS'16

And over the network...



Tatar et al., “Throwhammer: Rowhammer Attacks over the Network and Defenses,” ATC’18

What about DDR4?

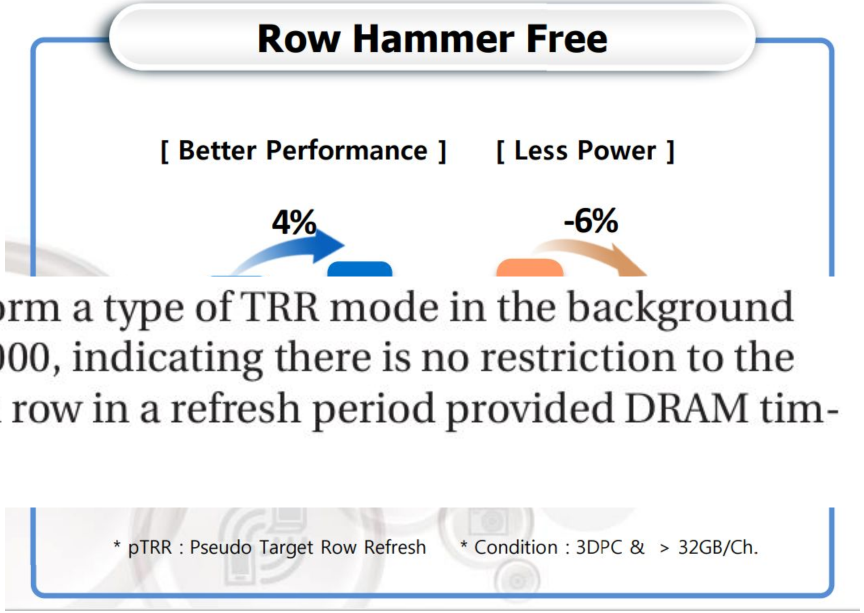
Rowhammer: affects 87% of deployed **DDR3** memory.

Drammer: bit flips on Pixel phones with LPDDR4 (2016)

ThirdIO’s Mark Lanteigne reports flips on DDR4 DIMMs (2016)

Gruss et al. report flips on DDR4 (SP’18, 2018)

Recent DDR4 systems

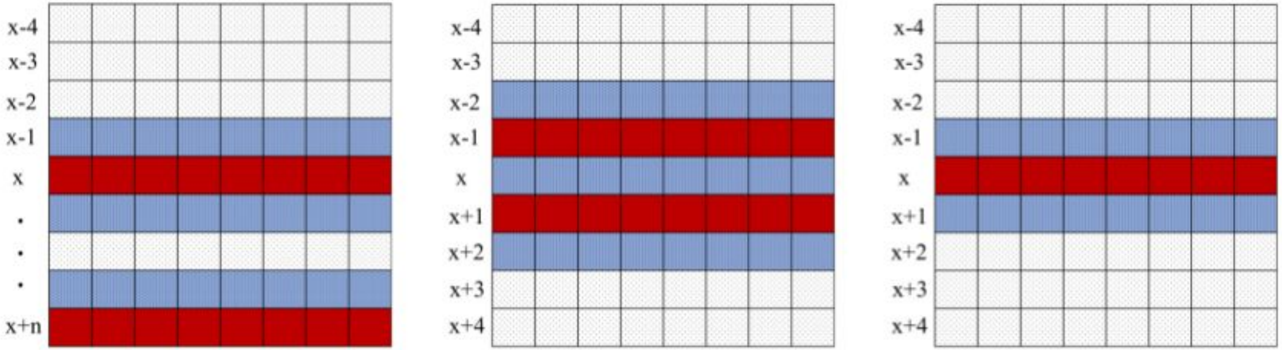


Micron's DDR4 devices automatically perform a type of TRR mode in the background and provide an MPR Page 3 MPR3[3:0] of 1000, indicating there is no restriction to the number of ACTIVATE commands to a given row in a refresh period provided DRAM timing specifications are not violated.

TRR: Keep track of intensely activated rows and refresh their neighbors.

So is Rowhammer a solved problem?

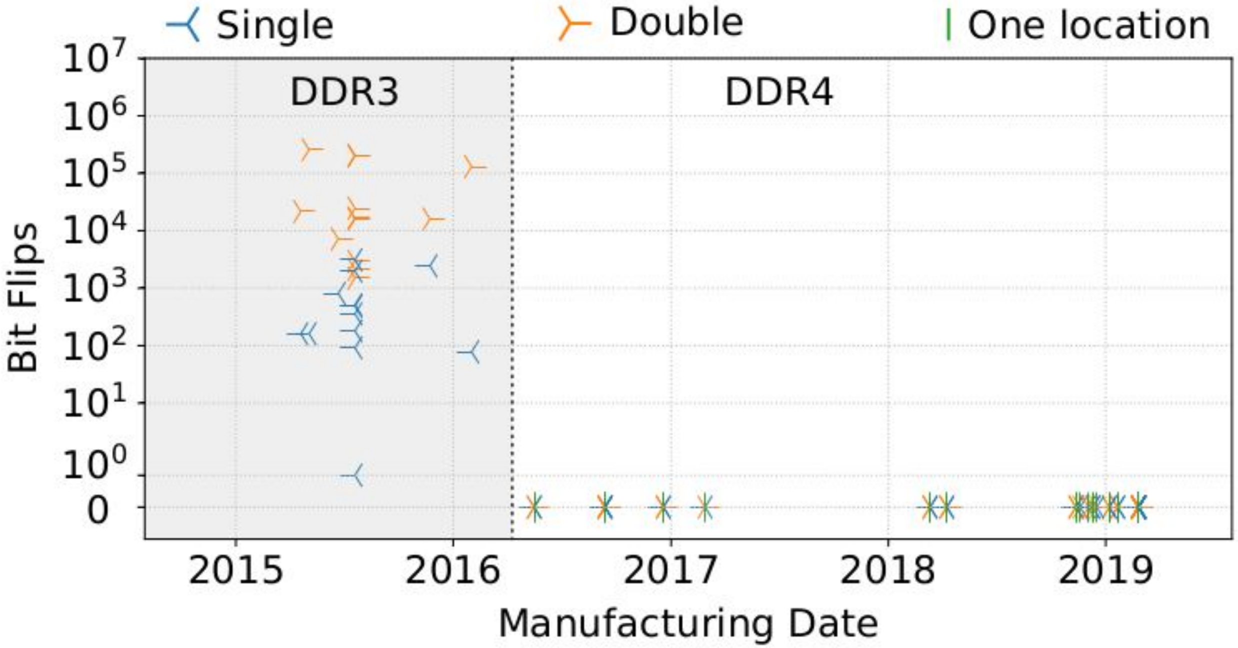
42 recent DIMMs from Samsung, Micron and Hynix (95%+ of market)



(a) Single-sided (b) Double-sided (c) One-location

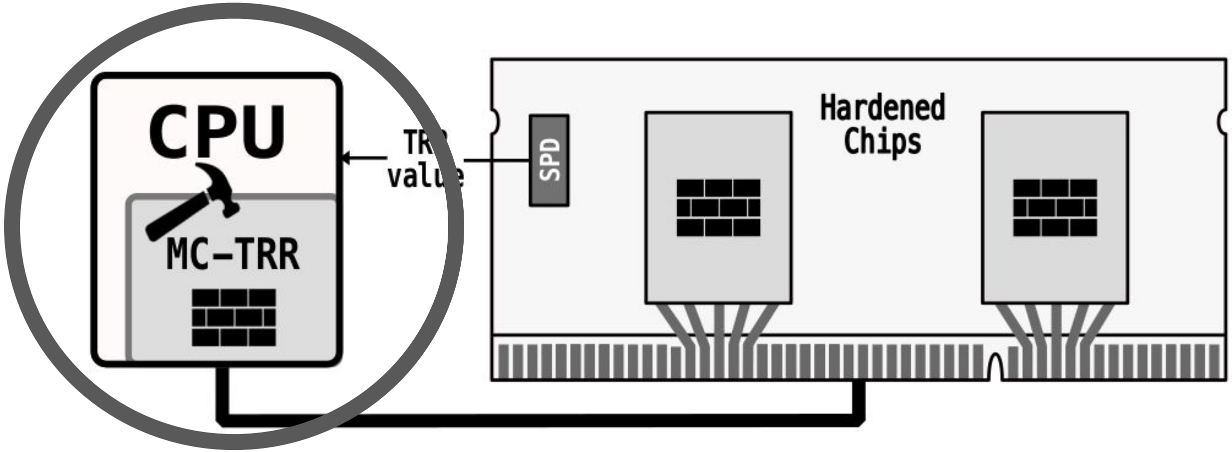
Known hammering patterns.

Results

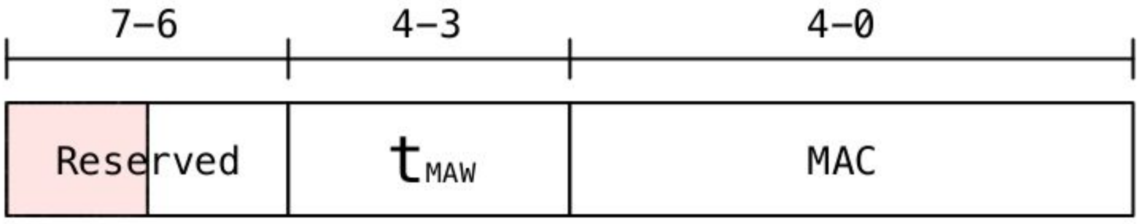

Single
Double
One location
DDR3
DDR4
Bit Flips
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
2015
2016
2017
2018
2019
Manufacturing Date

So what is TRR and is it really effective?

After a few weeks of reading patents...



Maximum Activation Count



MAW: Maximum Activation Window

MAC: Maximum Activation Count

TRR-compliant DRAM modules advertise these values

Memory controller tunes its mitigation based on them

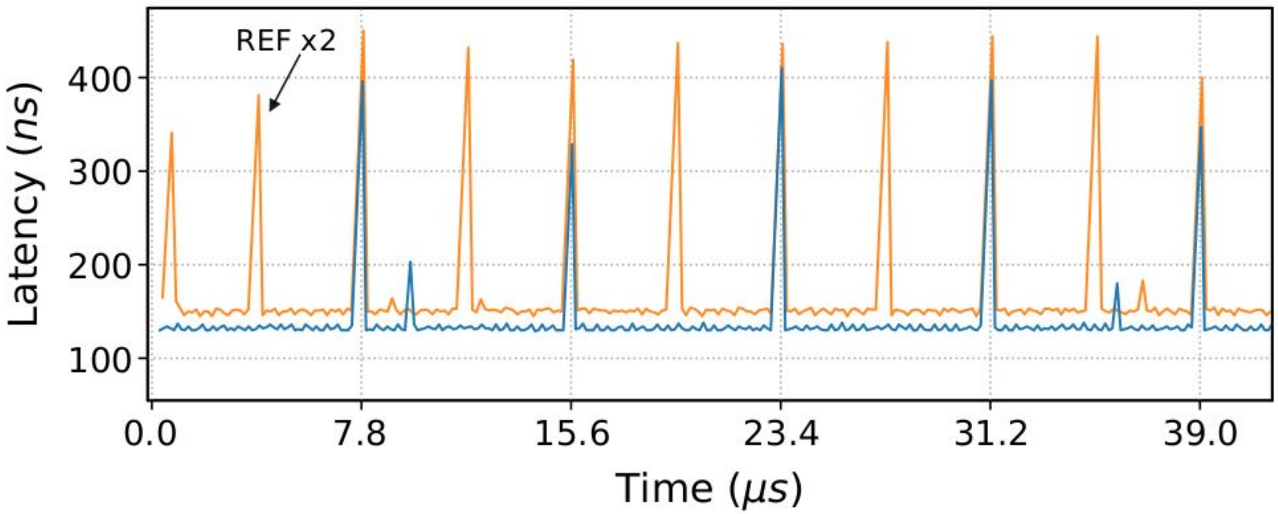
Detecting MC-based mitigations

Using memory access time → Detecting extra refresh commands

Using Rowhammer bit flips → Detecting existence of mitigation

With Timing

Xeon E5-2620 v2

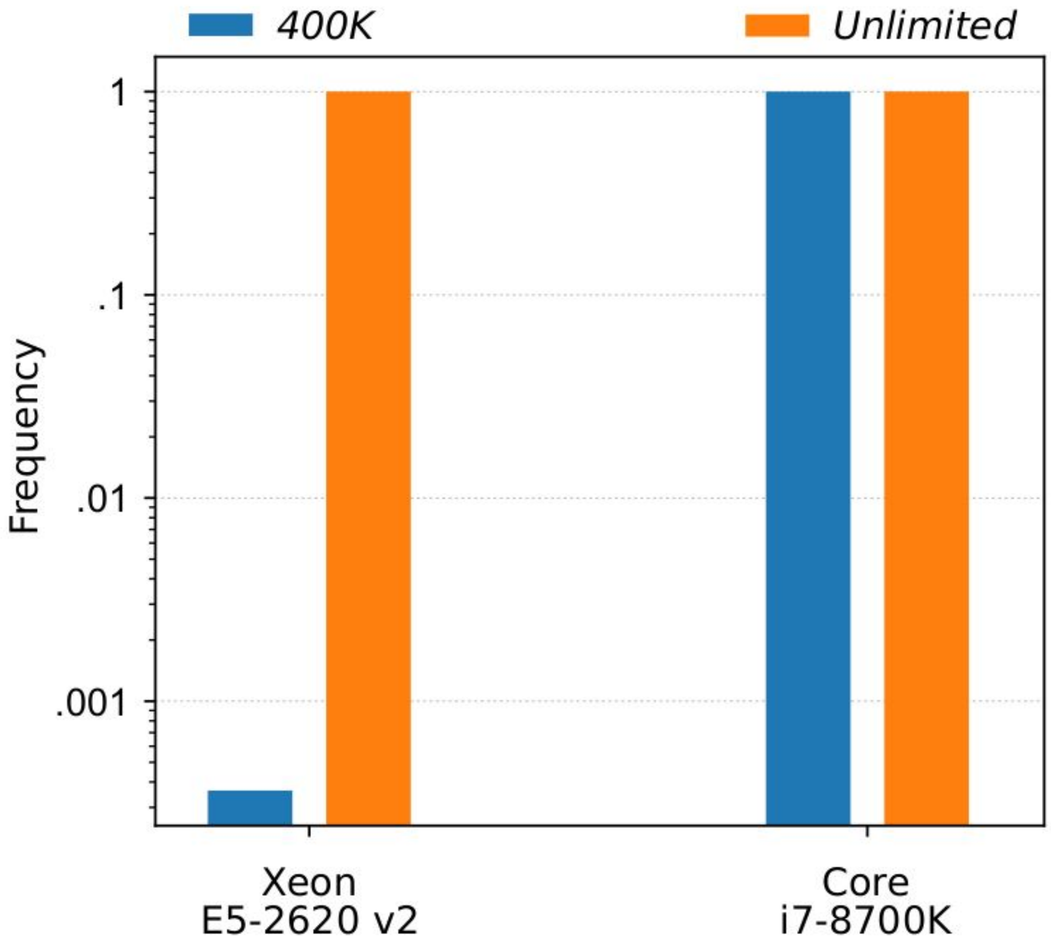


MAC: *Untested* *Unlimited*

With Flips

pTRR: Stops most flips

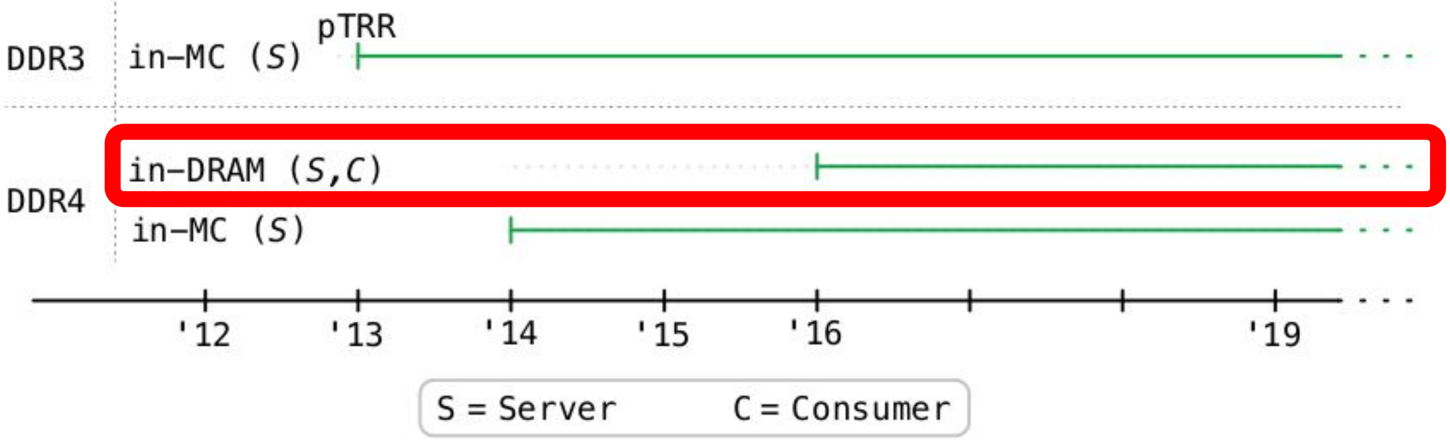
No mitigation on clients



MC-based mitigations on different platforms

CPU	*Family*	*Year*	*DRAM generation*	*Defense*
Server Line				
Xeon E5-2620 v4	Broadwell	2016	DDR4	`REF`×2
Xeon E5-2620 v2	Ivy Bridge	2013	DDR3	p-TRR
Xeon E3-1270 v3	Haswell	2013	DDR3	—
Consumer Line				
Core i9-9900K	Coffee Lake R	2018	DDR4	—
Core i7-8700K	Coffee Lake	2017	DDR4	—
Core i7-7700K	Kaby Lake	2017	DDR4	—
Core i7-5775C	Broadwell	2015	DDR3	—

TRR timeline



All our DDR4 DIMMs after ‘16 have MAC set to unlimited

No bit flip with all known Rowhammer patterns

Understanding in-DRAM TRR

Using memory access time ✖ All accesses take the same

Using Rowhammer bit flips ✖ No bit flips

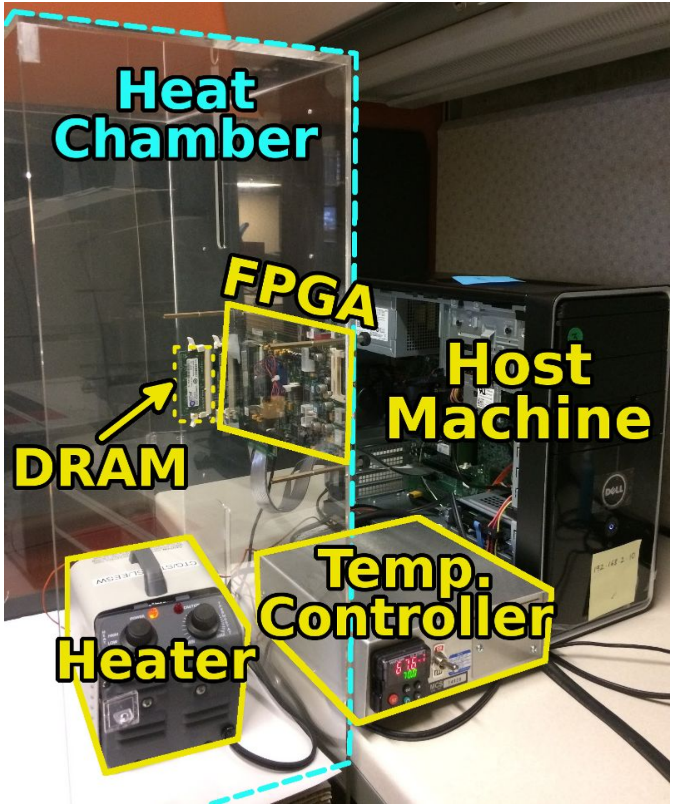
Any TRR solution:

1) Sampling mechanism → Happens at memory access
2) Inhibitor mechanism → Extra refreshes

When do extra internal refreshes happen?

SoftMC

- Open-source platform for DRAM studies
- Support for DDR4
- Precise control over DRAM commands
    - ACTIVATE, READ/WRITE, PRECHARGE, REFRESH
- Run DRAM out of spec



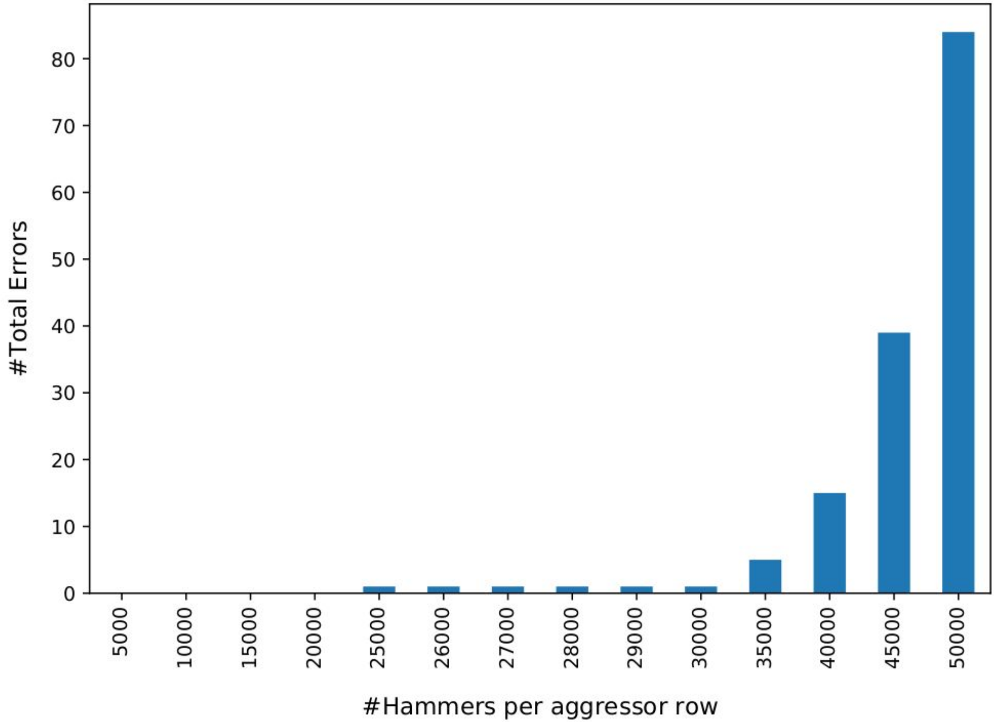
Extra refreshes

On a DIMM from manufacturer C:

No REFRESH command (generally 1 every 7.8us)

1) Write values in memory
2) Hammer for 64 ms (refresh cycle)
3) Check for flips

Results



JEDEC → No retention failure for 64ms

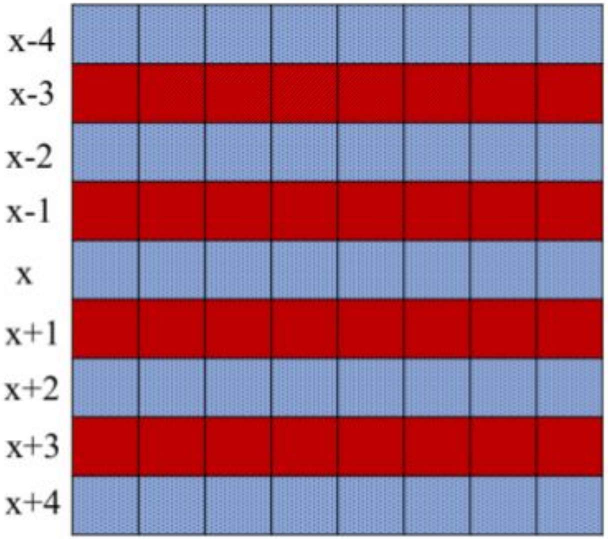
Rowhammer flips → Extra refreshes happen at **REFRESH**

DDR4 cells are leakier than DDR3 cells

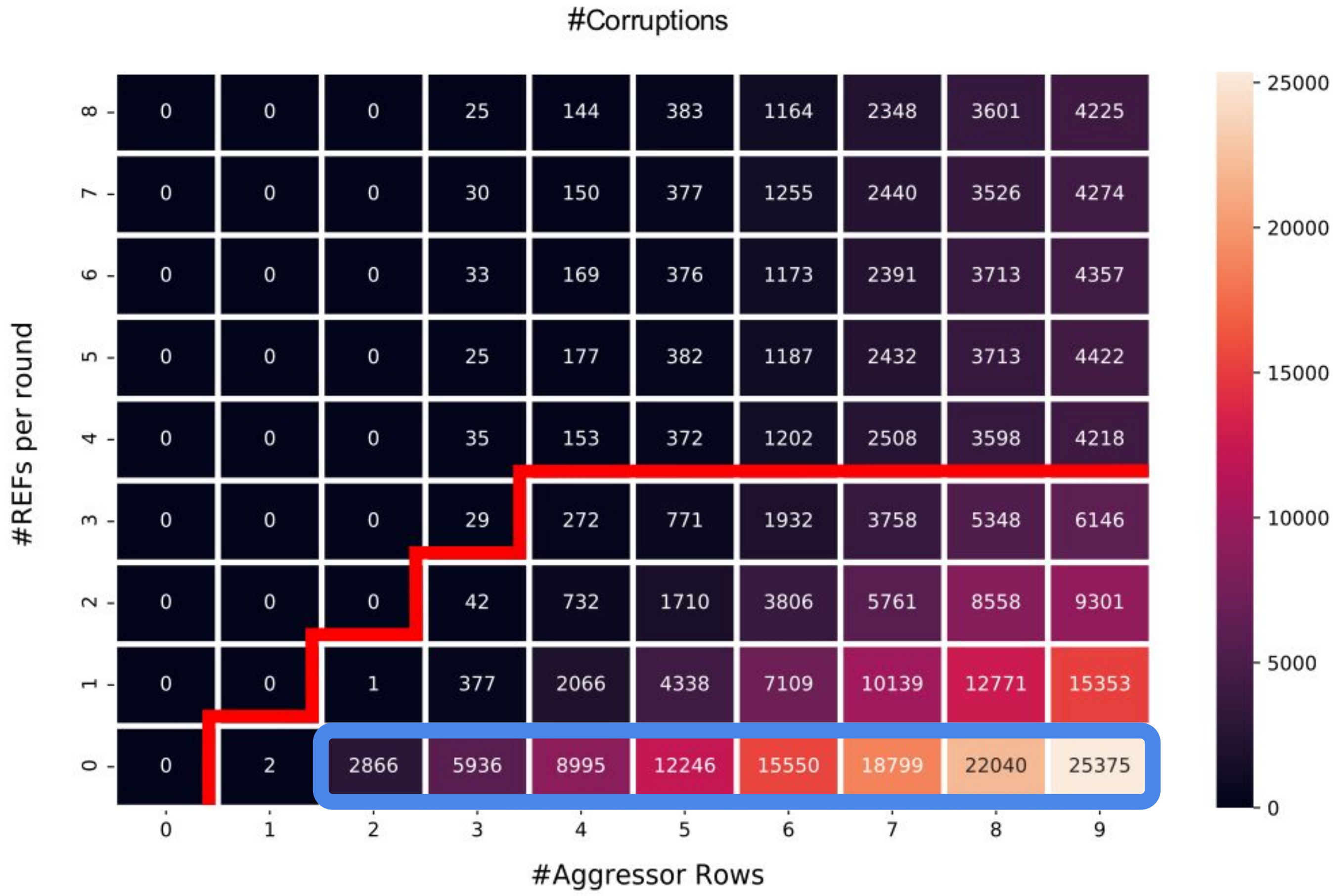
Figuring out the sampler size

Pick N aggressor rows

1) Hammer each for 10K (N x 10K)
2) Send M REFRESH commands
3) Repeat for 10 times

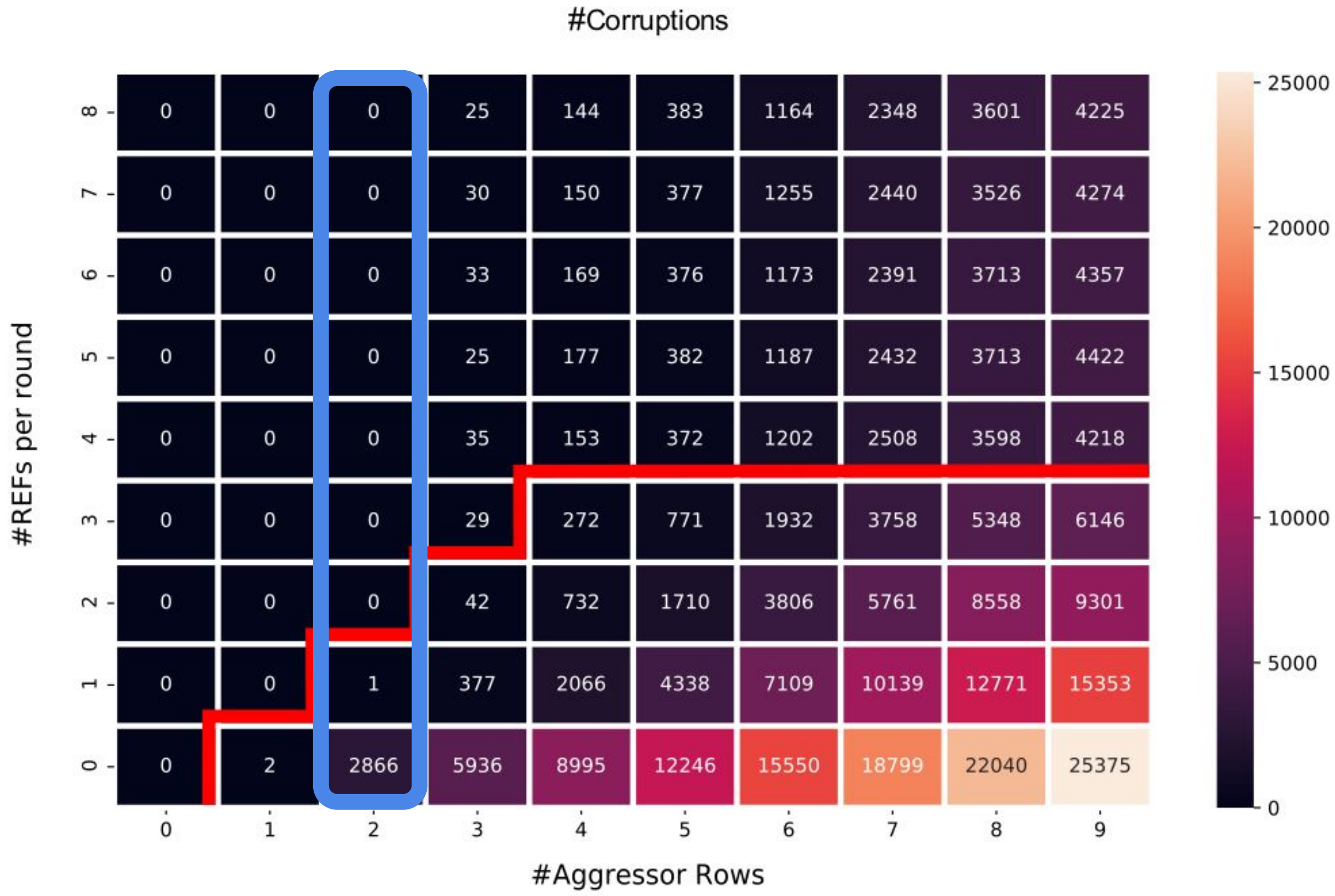


Results



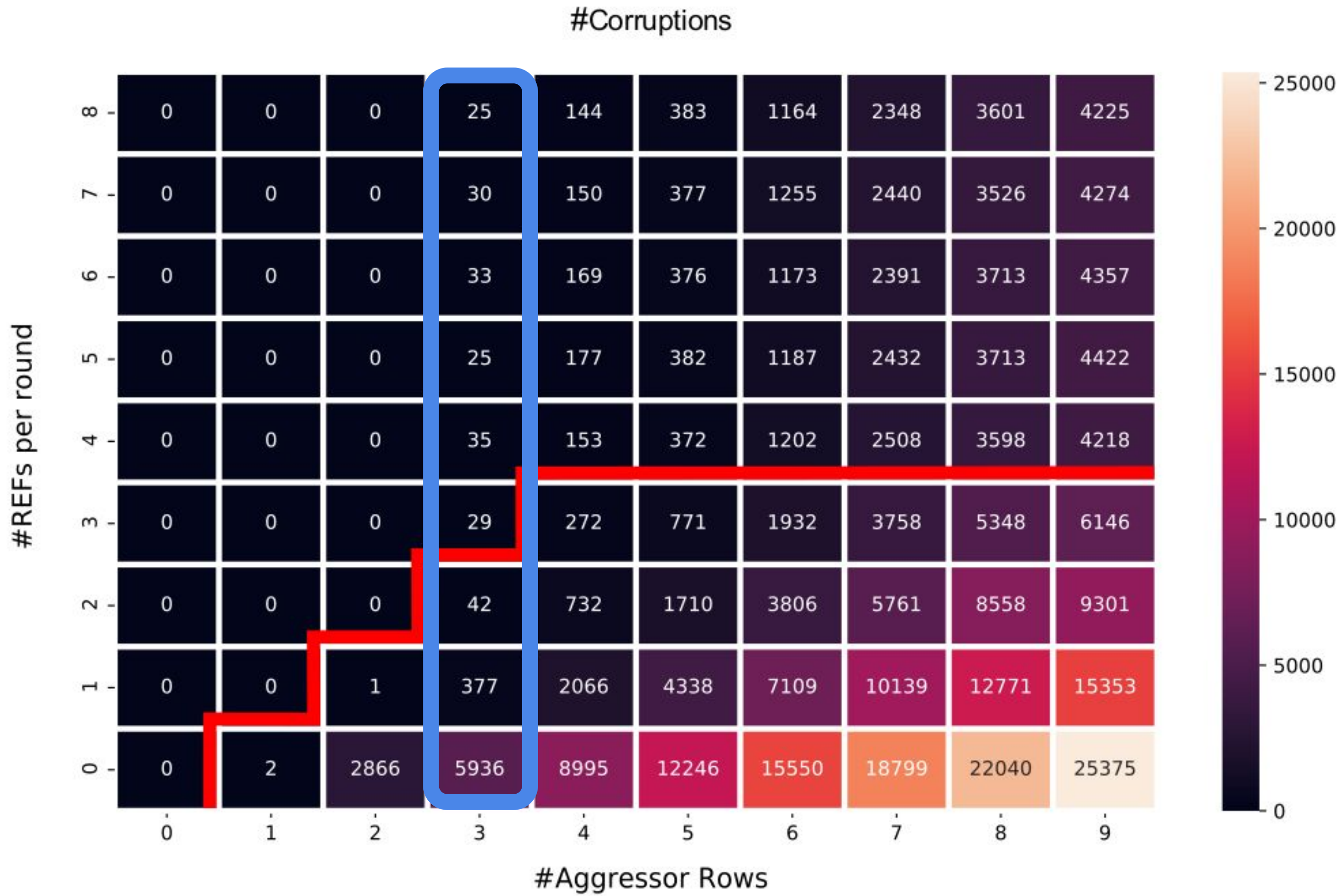
Flips are distributed uniformly over the victims

Results



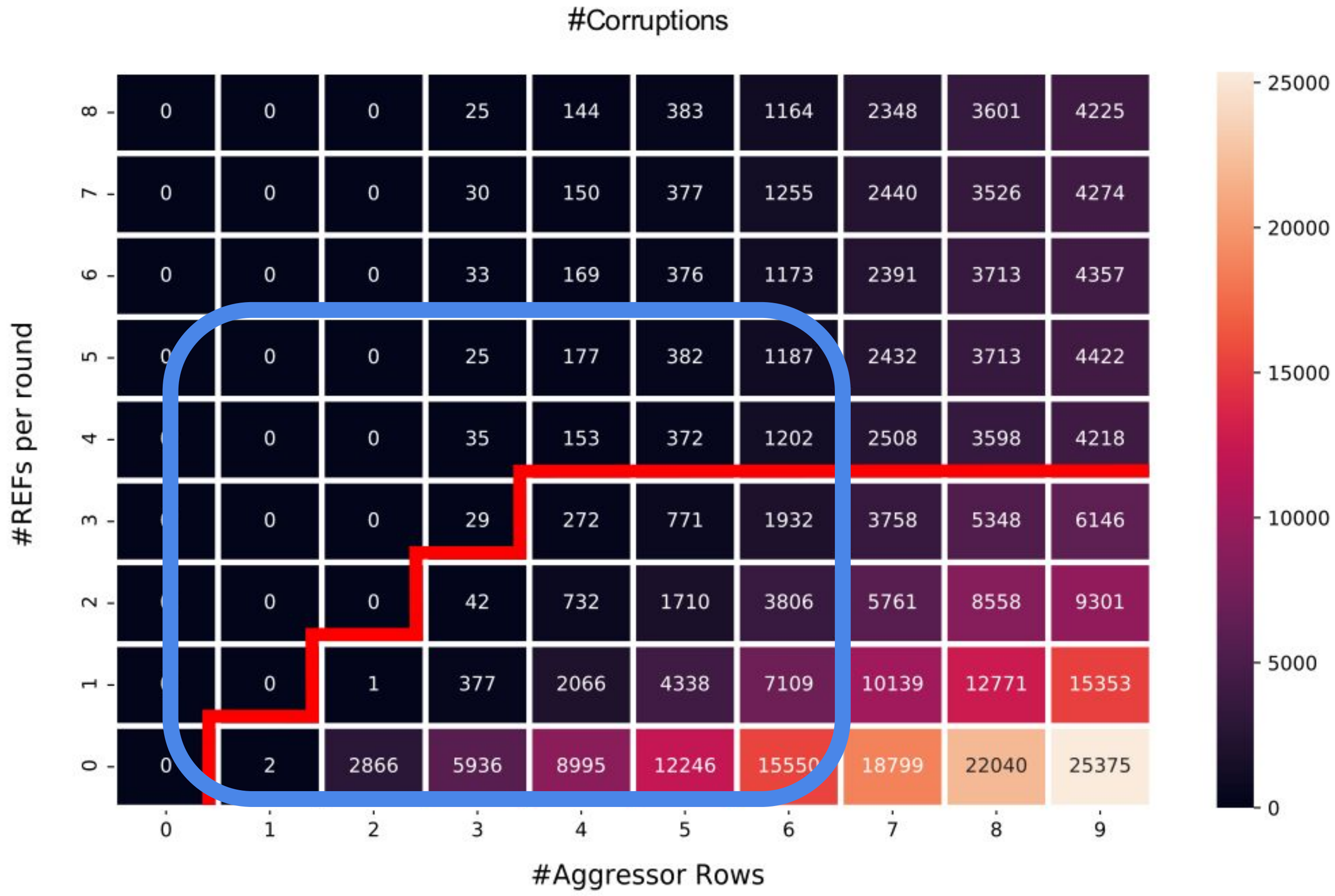
1 TRR per REFRESH command

Results



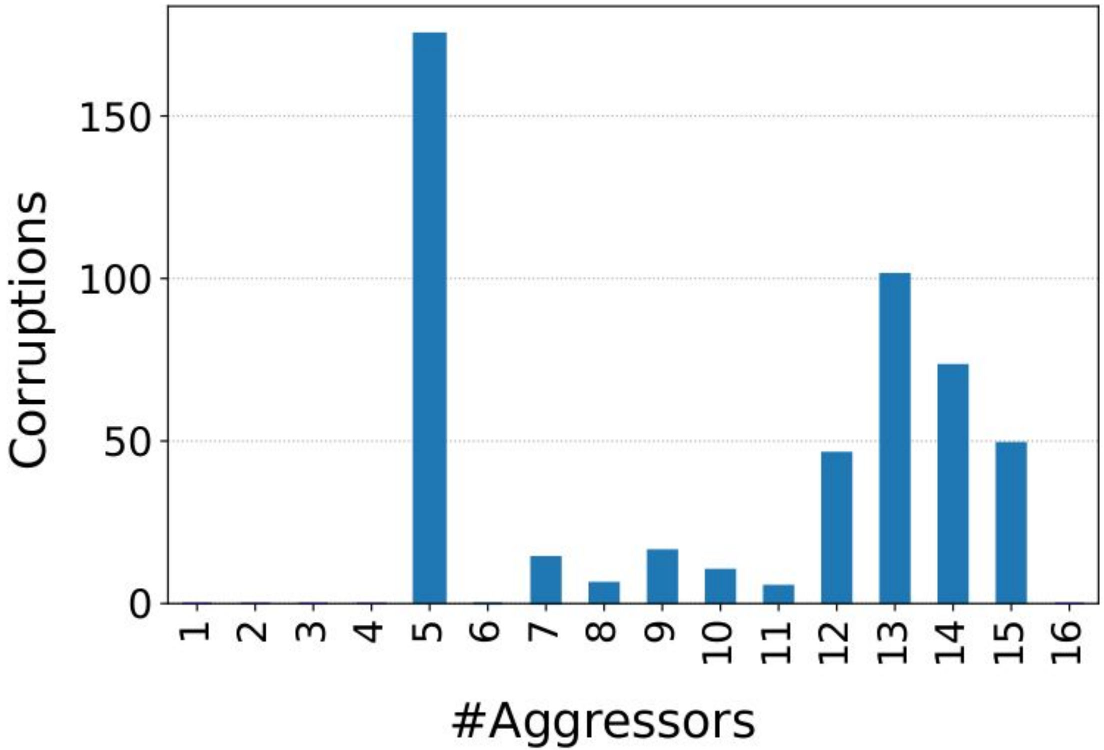
Remaining flips likely due to aggressor being discarded

Results



Sampler size is 4 given that number of flips plateau after 4 REFs

Flips with native REFRESH rate



Other DIMMs?

Reverse engineered DIMMs from Manufacturer A: very different

Newer DIMMs from Manufacturer C? Different mitigation

Can we automate the analysis to try on different DIMMs?

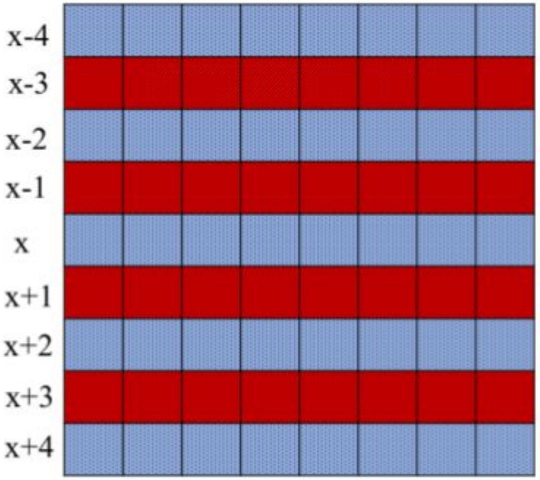
Meet TRRespass

A Rowhammer fuzzer:

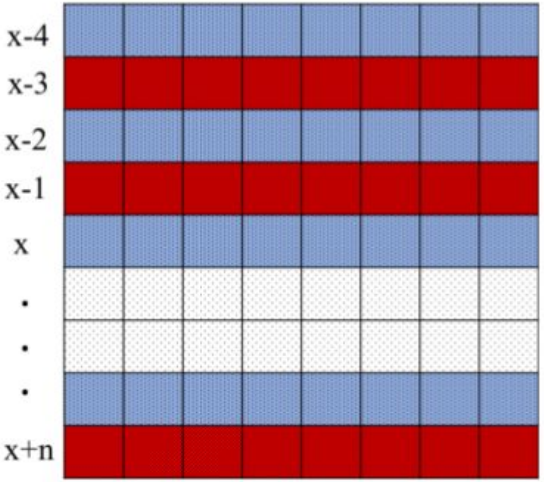
1) Cardinality (# aggressor rows)
2) Aggressor row location

- Executed from the CPU (DRAM mapping reverse engineering)
- Allocate a block of memory and try many random patterns

Successful patterns: many-sided Rowhammer



4-sided



Assisted double-sided

Results

Module	Date (yy-ww)	Freq. (MHz)	Size (GB)	Organization			MAC	Found Patterns	Best Pattern	Corruptions			Double Refresh
				Ranks	Banks	Pins				Total	$1 \to 0$	$0 \to 1$	
$\mathcal{A}_{0,1,2,3}$	16-37	2132	4	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{A}_{4}$	16-51	2132	4	1	16	×8	UL	4	9-sided	7956	4008	3948	—
$\mathcal{A}_{5}$	18-51	2400	4	1	8	×16	UL	—	—	—	—	—	—
$\mathcal{A}_{6,7}$	18-15	2666	4	1	8	×16	UL	—	—	—	—	—	—
$\mathcal{A}_{8}$	17-09	2400	8	1	16	×8	UL	33	19-sided	20808	10289	10519	—
$\mathcal{A}_{9}$	17-31	2400	8	1	16	×8	UL	33	19-sided	24854	12580	12274	—
$\mathcal{A}_{10}$	19-02	2400	16	2	16	×8	UL	488	10-sided	11342	1809	11533	✓
$\mathcal{A}_{11}$	19-02	2400	16	2	16	×8	UL	523	10-sided	12830	1682	11148	✓
$\mathcal{A}_{12,13}$	18-50	2666	8	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{A}_{14}$	19-08†	3200	16	2	16	×8	UL	120	14-sided	32723	16490	16233	—
$\mathcal{A}_{15}{}^{\ddagger}$	17-08	2132	4	1	16	×8	UL	2	9-sided	22397	12351	10046	—
$\mathcal{B}_{0}$	18-11	2666	16	2	16	×8	UL	2	3-sided	17	10	7	—
$\mathcal{B}_{1}$	18-11	2666	16	2	16	×8	UL	2	3-sided	22	16	6	—
$\mathcal{B}_{2}$	18-49	3000	16	2	16	×8	UL	2	3-sided	5	2	3	—
$\mathcal{B}_{3}$	19-08†	3000	8	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{B}_{4,5}$	19-08†	2666	8	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{B}_{6,7}$	19-08†	2400	4	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{B}_{8}{}^{\diamond}$	19-08†	2400	8	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{B}_{9}{}^{\diamond}$	19-08†	2400	8	1	16	×8	UL	2	3-sided	12	—	12	✓
$\mathcal{B}_{10,11}$	16-13†	2132	8	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{0,1}$	18-46	2666	16	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{2,3}$	19-08†	2800	4	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{4,5}$	19-08†	3000	8	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{6,7}$	19-08†	3000	16	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{8}$	19-08†	3200	16	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{9}$	18-47	2666	16	2	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{10,11}$	19-04	2933	8	1	16	×8	UL	—	—	—	—	—	—
$\mathcal{C}_{12}{}^{\ddagger}$	15-01†	2132	4	1	16	×8	UT	25	10-sided	190037	63904	126133	✓
$\mathcal{C}_{13}{}^{\ddagger}$	18-49	2132	4	1	16	×8	UT	3	9-sided	694	239	455	—

TRRespass' preliminary port to ARM

Limitations:

1) No DRAM mapping functions
2) No access to large pages
3) Detect N bank conflicts? Hammer.

Mobile Phone	*Year*	*SoC*	*Memory (GB)*	*Found Patterns*
Google Pixel	2016	MSM8996	4†	✓
Google Pixel 2	2017	MSM8998	4	—
Samsung G960F/DS	2018	Exynos 9810	4	—
Huawei P20 DS	2018	Kirin 970	4	—
Sony XZ3	2018	SDM845	4	—
HTC U12+	2018	SDM845	6	—
LG G7 ThinQ	2018	SDM845	4†	✓
Google Pixel 3	2018	SDM845	4	✓
Google Pixel 4	2019	SM8150	6	—
OnePlus 7	2019	SM8150	8	✓
Samsung G970F/DS	2019	Exynos 9820	6	✓
Huawei P30 DS	2019	Kirin 980	6	—
Xiaomi Redmi Note 8 Pro	2019	Helio G90T	6	—

† LPDDR4 (not LPDDR4X)

Conclusion

Open PhD Positions @ ETH

Rowhammer is alive and kicking on latest systems

Maintaining data consistency in DRAM has become hard

Security through obscurity is trickier with DRAM

kaveh@ethz.ch @kavehrazavi