Comp 590-184: Hardware Security and Side-Channels

Lecture 20: Rowhammer Continued

April 14, 2026

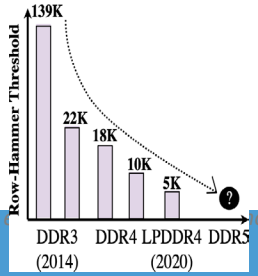
Andrew Kwong

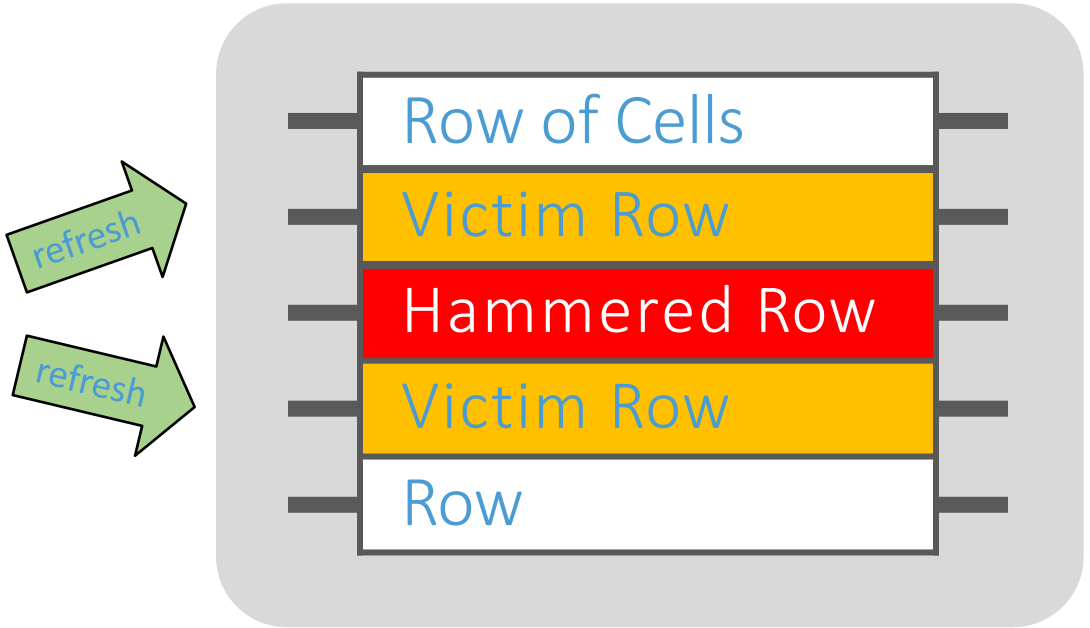
THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

RowHammer Mitigations: A game of cat and mouse

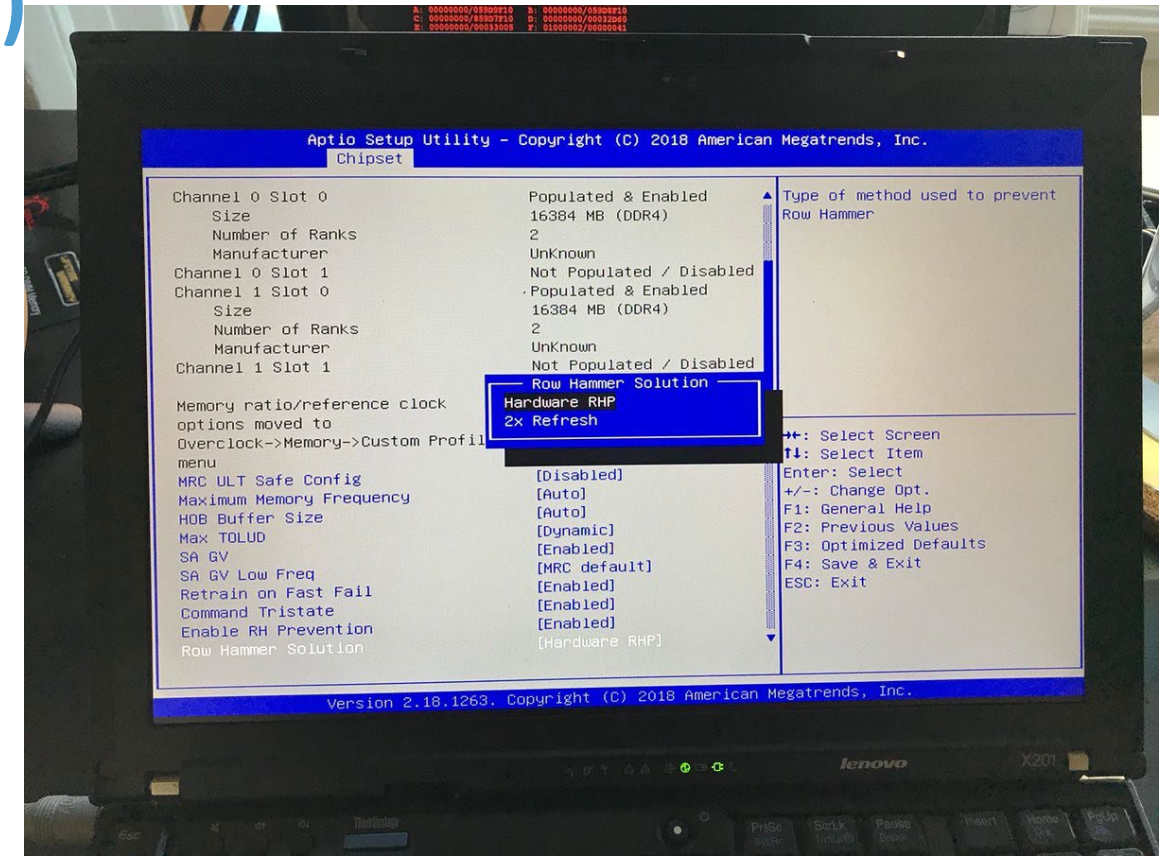
PARA: Probabilistic Row Activation

- Pick a probability "p"
- After closing a row, we activate (i.e., refresh) one of its neighbors with a low probability: $p = 0.005$
- Question: how to pick "p"? What is the consequence?



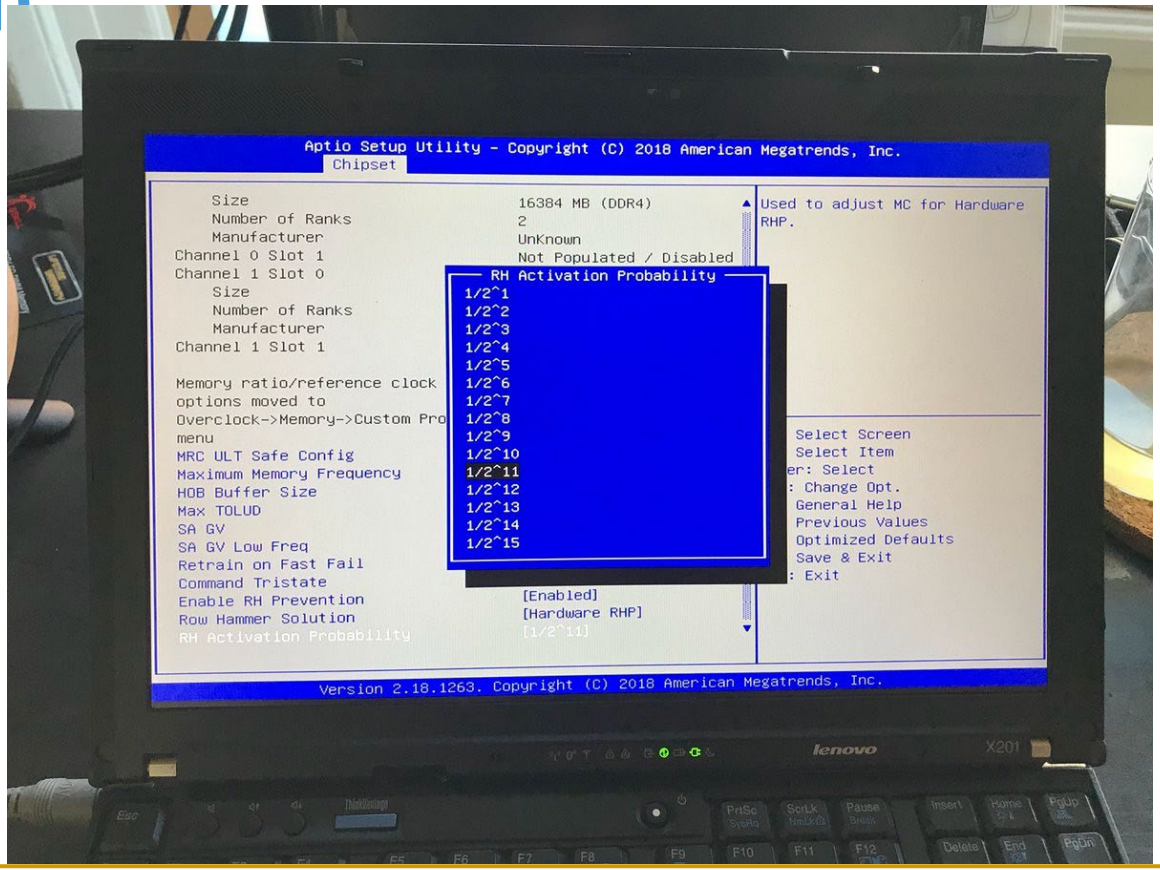


Probabilistic Activation in Real Life (I)



https://twitter.com/isislovecruft/status/1021939922754723841

Probabilistic Activation in Real Life (II)

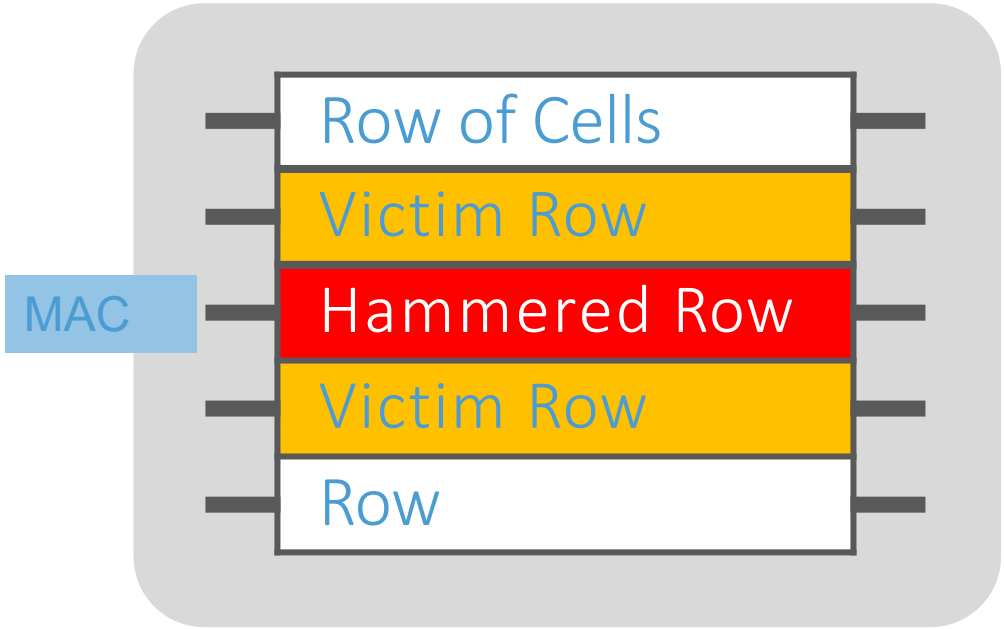


https://twitter.com/isislovecruft/status/1021939922754723841

Counter-based Row Activation

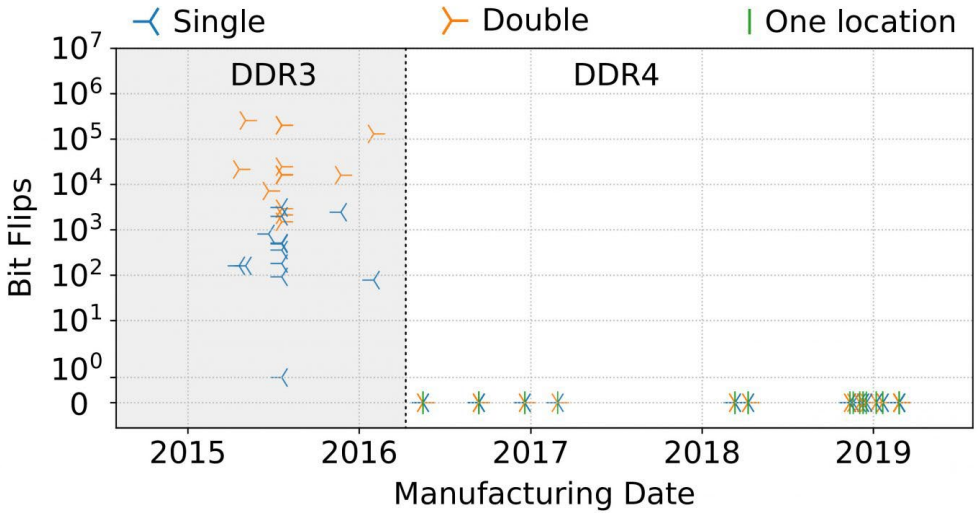
- Maintain a counter to track the number of accesses per row
  - Increment the counter when accessing a row
  - When reaching the Maximum Activation Count (MAC), activate the neighboring rows
  - After activating, reset the counter

- Deployed in actual hardware (DDR4+DDR5) in 2016 as Targeted Row Refresh (TRR)

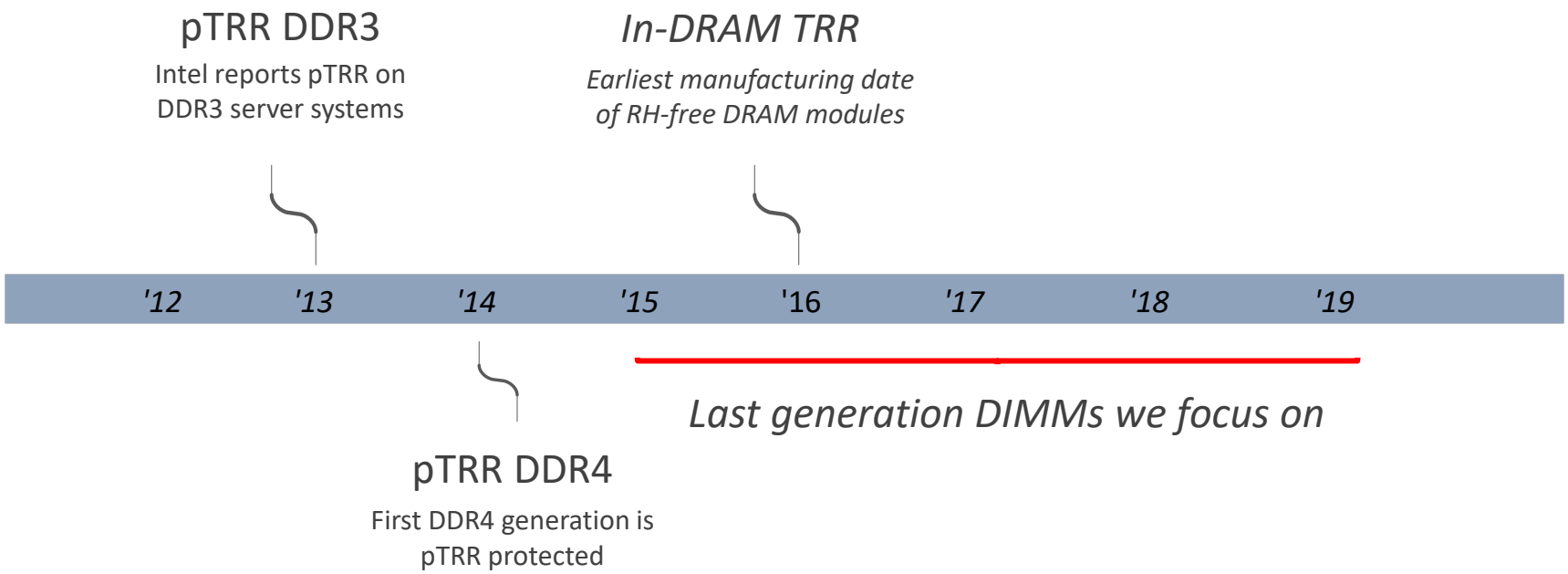


TRR

- Rowhammer is solved now, right?



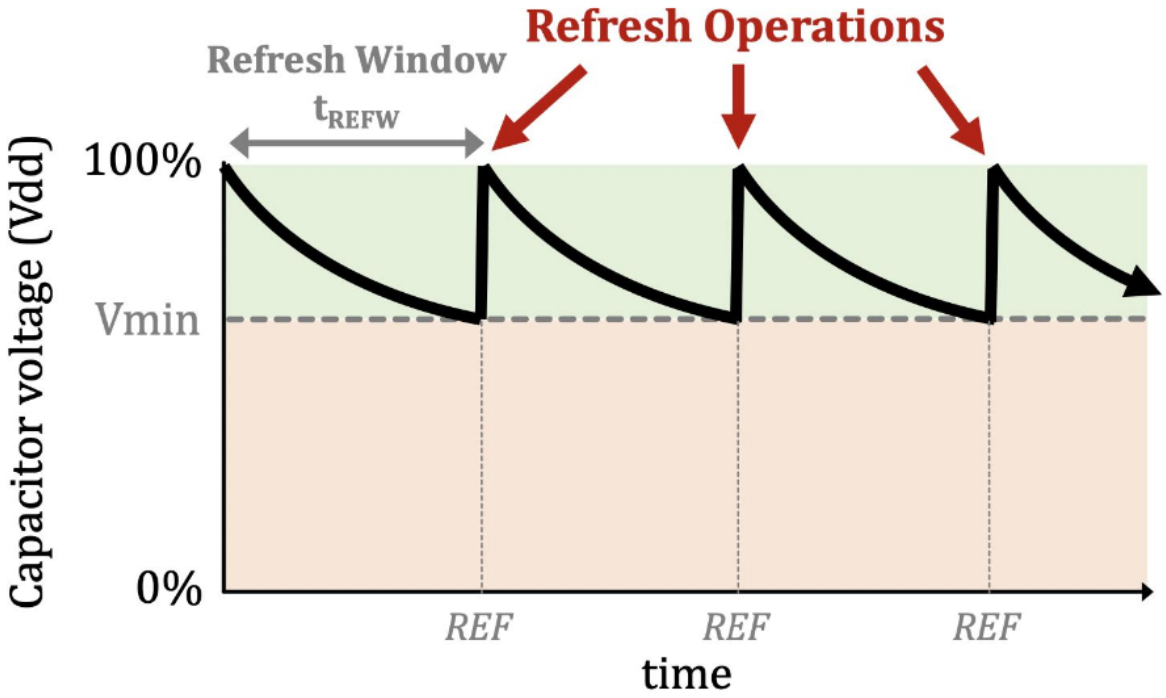
Timeline


pTRR DDR3
Intel reports pTRR on DDR3 server systems
In-DRAM TRR
Earliest manufacturing date of RH-free DRAM modules
'12
'13
'14
'15
'16
'17
'18
'19
Last generation DIMMs we focus on
pTRR DDR4
First DDR4 generation is pTRR protected

Trrespass

- Memory vendors advertise RowHammer-free devices
- What is Target Row Refresh (TRR)? Not a single mitigation!
- Reverse-engineering of in-DRAM mitigations
  - The Many-sided RowHammer
  - Hammering up to 20 aggressor rows
- 3 major vendors all vulnerable: Samsung, Micron, SK Hynix
  - Currently representing over 95% of the DRAM market

Periodic refreshes are required to preserve the stored data as the cells leak charge.



DRAM Refresh

- DRAM is dynamic because data must be refresh periodically
  - Retention time (i.e., 64ms)
- The MC issues a **REFRESH** command every 7.8µs
  - Only a small portion of memory is refreshed with a command
  - 8192 refreshes within a 64ms interval

Building blocks

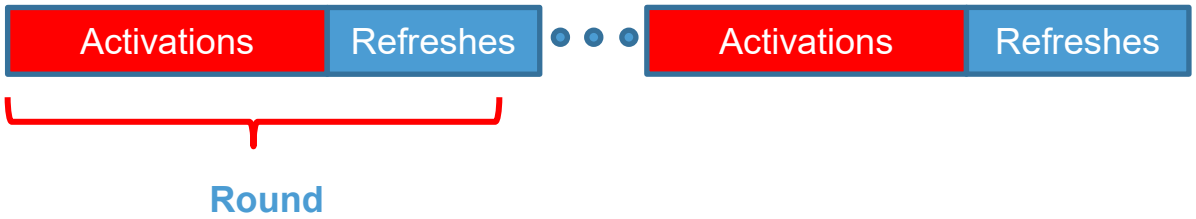
Abstractions:

- **Sampler**
  - Track aggressor rows activations
  - Keep a set of rows
- **Inhibitor**
  - Prevent bit flips
  - Refresh victims

Case study: Vendor C

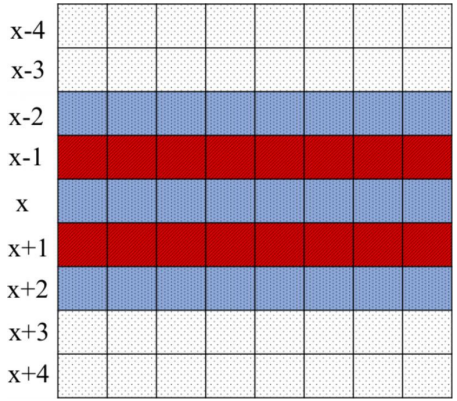
How big is the sampler?

- Pick **N** aggressor rows
- Perform a series of hammers (activations of aggressors)
  - **8K activations**
- After each series of hammers, issue **R refreshes**
- **10 Rounds**

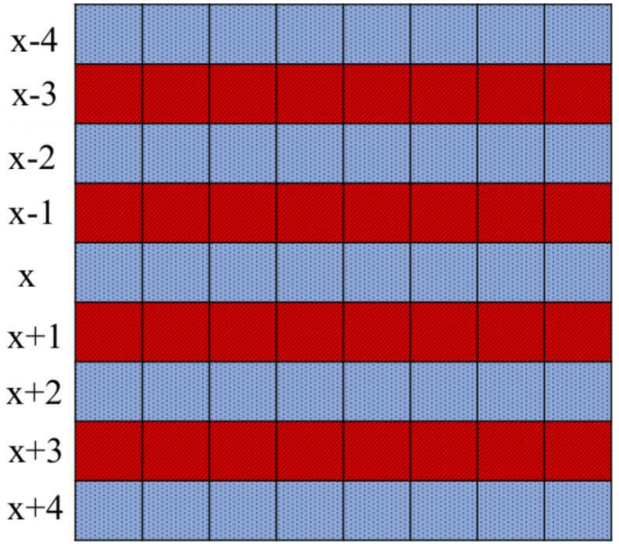


TRRespass (Oakland 2020)

- Made the observation that not every row can be tracked

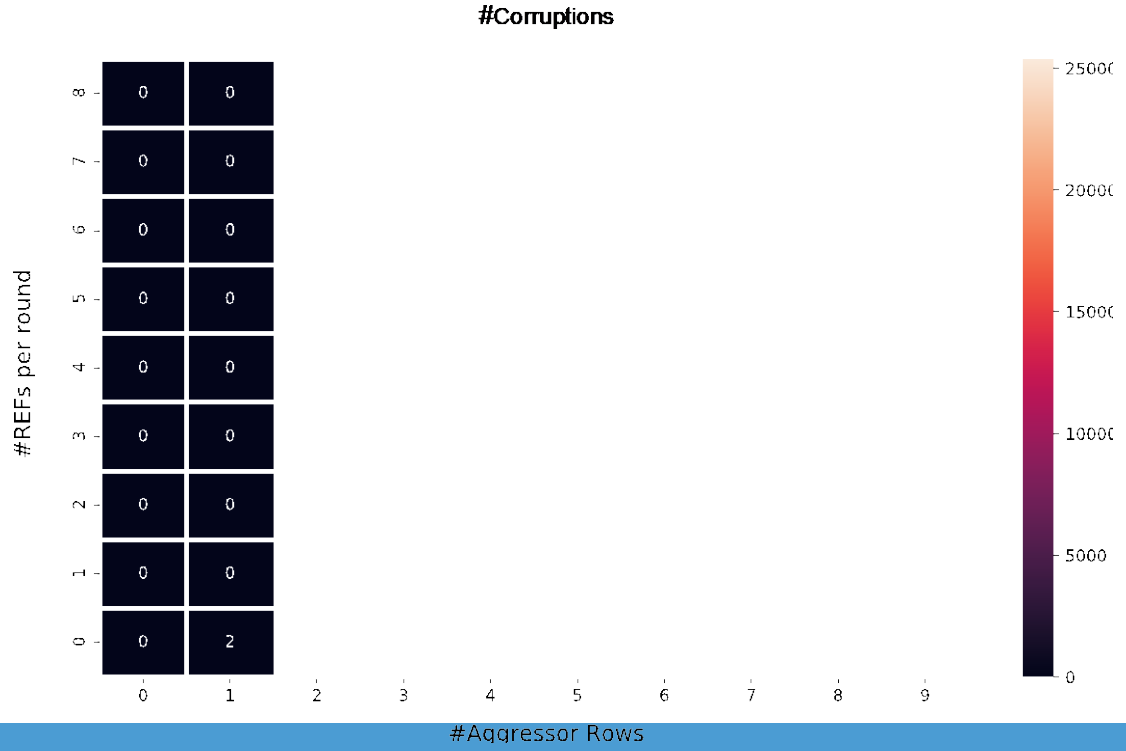


Double sided Rowhammer

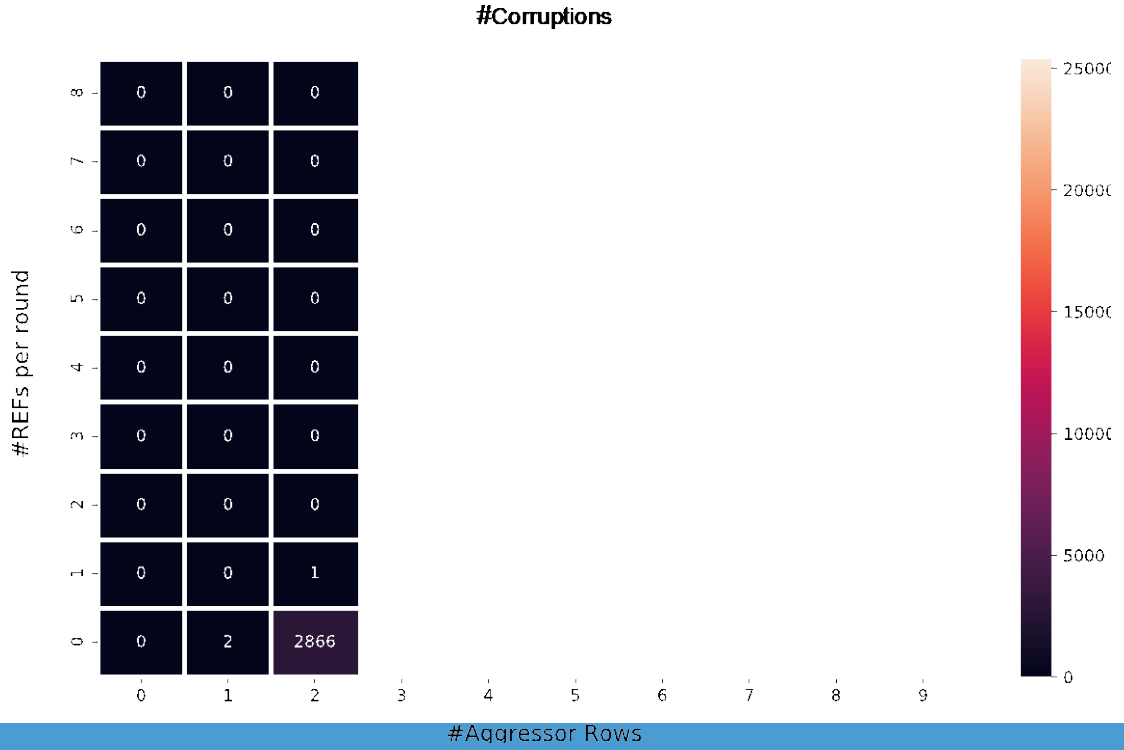


Many sided Rowhammer

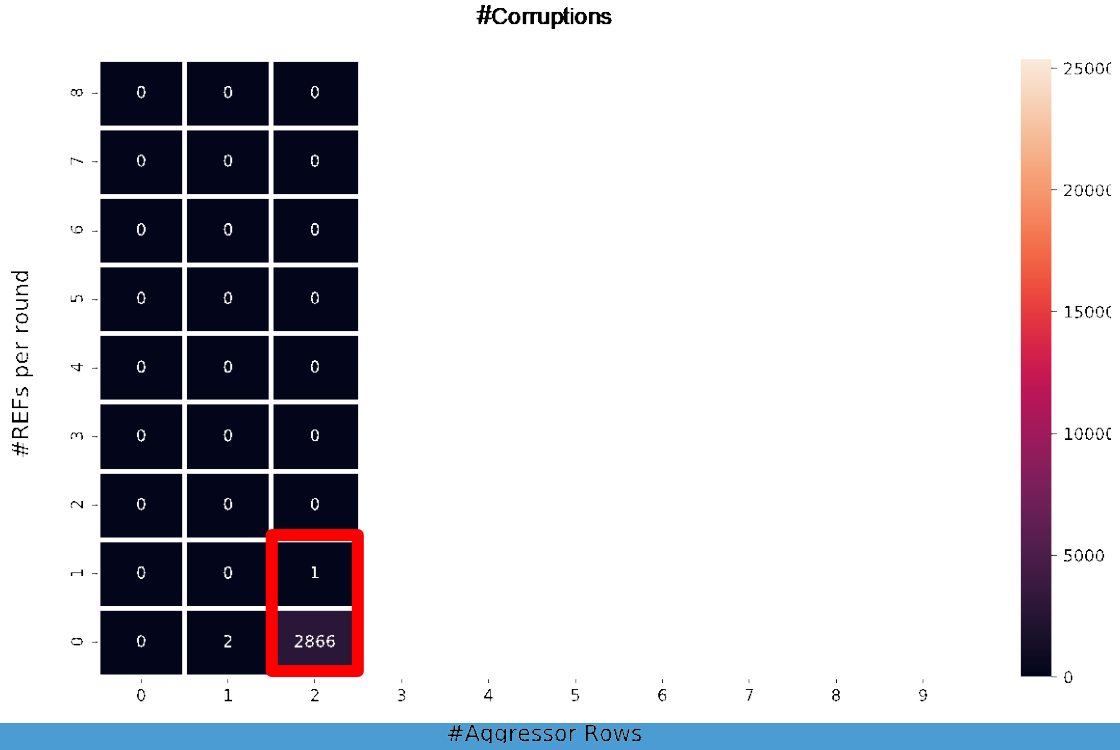
Case study: Vendor C



Case study: Vendor C



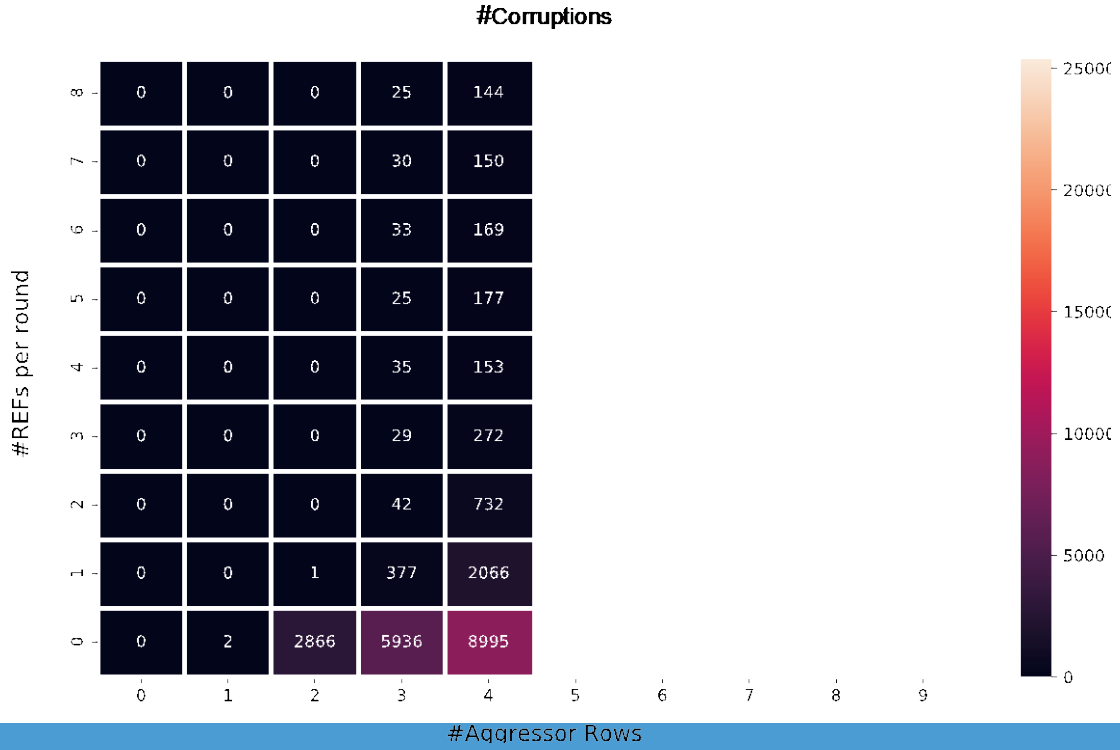
Case study: Vendor C



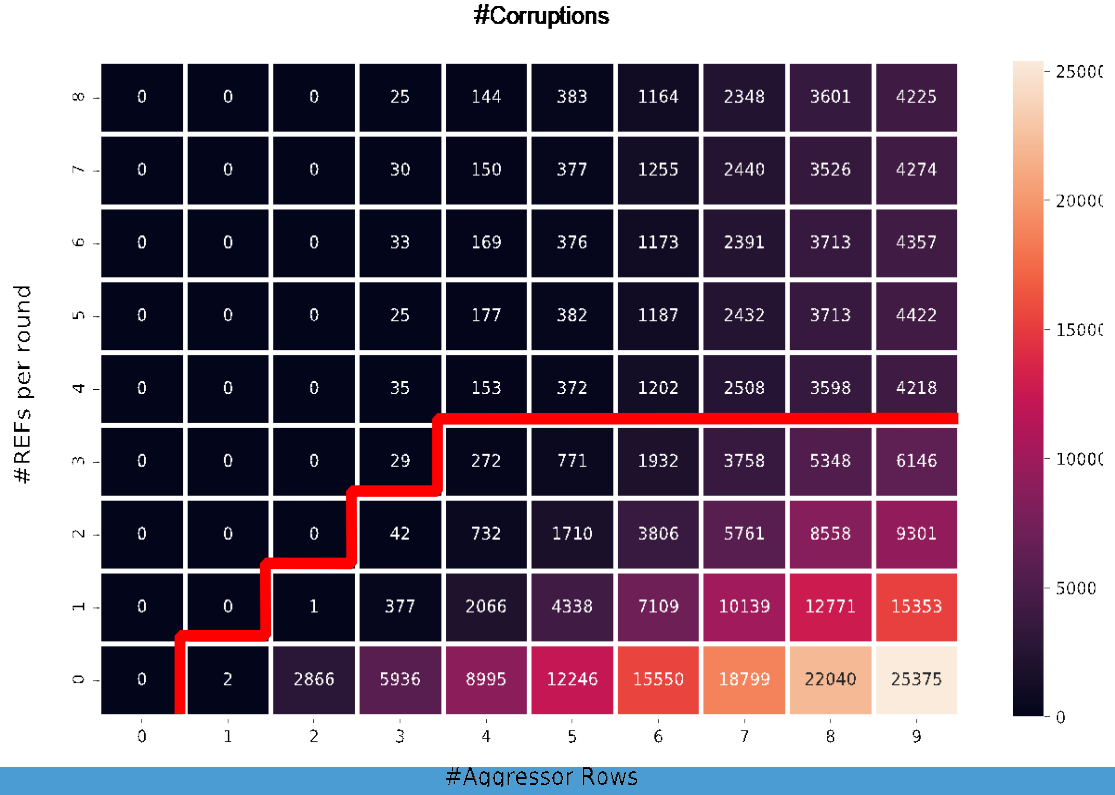
Case study: Observations

- The TRR mitigation acts on every refresh command
- The mitigation can refresh only a single victim within a refresh operation

Case study: Vendor C



Case study: Vendor C



Case study: Observations

- The TRR mitigation acts on every refresh command
- The mitigation can refresh only a single victim within a refresh operation
- **Sweeping the number of refresh operations and aggressor rows while hammering reveals the sampler size**

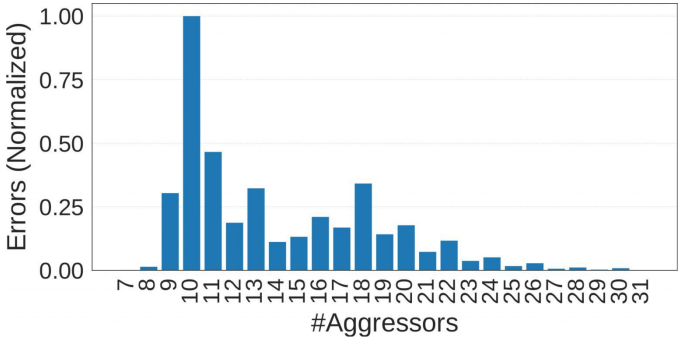
TRRespass: Results

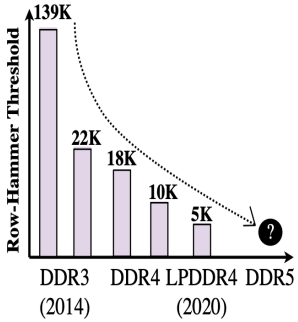
- **42** DIMMS from **3 of the major vendors**: Samsung, Micron, SK Hynix
  - 95% of the market
- Testing 256MB of contiguous memory against the best pattern
- **13** DIMMs with bit flips
  - Multiple effective patterns for each of them
- Bit flips with **double refresh**

TRRespass (Oakland 2020)

- Hammer too many rows for TRR to track them

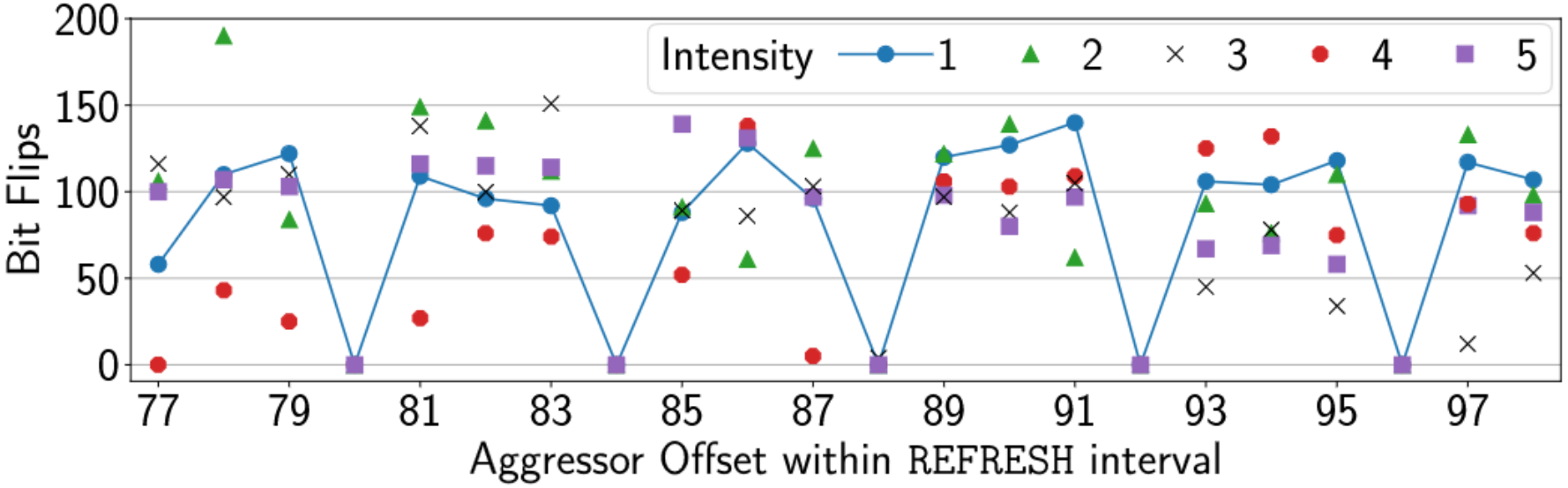
Module	Date	Best Pattern	1->0	0->1
A4	16-51	9-sided	4008	3948
A8	17-09	19-sided	10289	10519
A9	17-31	19-sided	12580	12274
A10	19-02	10-sided	1809	11533
A11	19-02	10-sided	1682	11148
A14	19-08	14-sided	16490	16233
A15	17-08	3-sided	12351	10046
B0	18-11	3-sided	10	7
B1	18-11	3-sided	16	6
B2	18-49	3-sided	2	3
B9	19-08	3-sided	–	12
C12	15-01	10-sided	63904	126133
C13	18-49	9-sided	239	455





Sychronized Hammering

- TRR Refreshes only occur at regular refreshed each interval
  - Piggybacked on regular refresh mechanism
- Only certain memory offsets within refresh interval are sampled
- Sychronizing a hammering pattern evades detection
  - Access to a row always happens at same offset within a refresh interval


Intensity
1
2
3
4
5
Bit Flips
200
150
100
50
0
77
79
81
83
85
87
89
91
93
95
97
Aggressor Offset within REFRESH interval

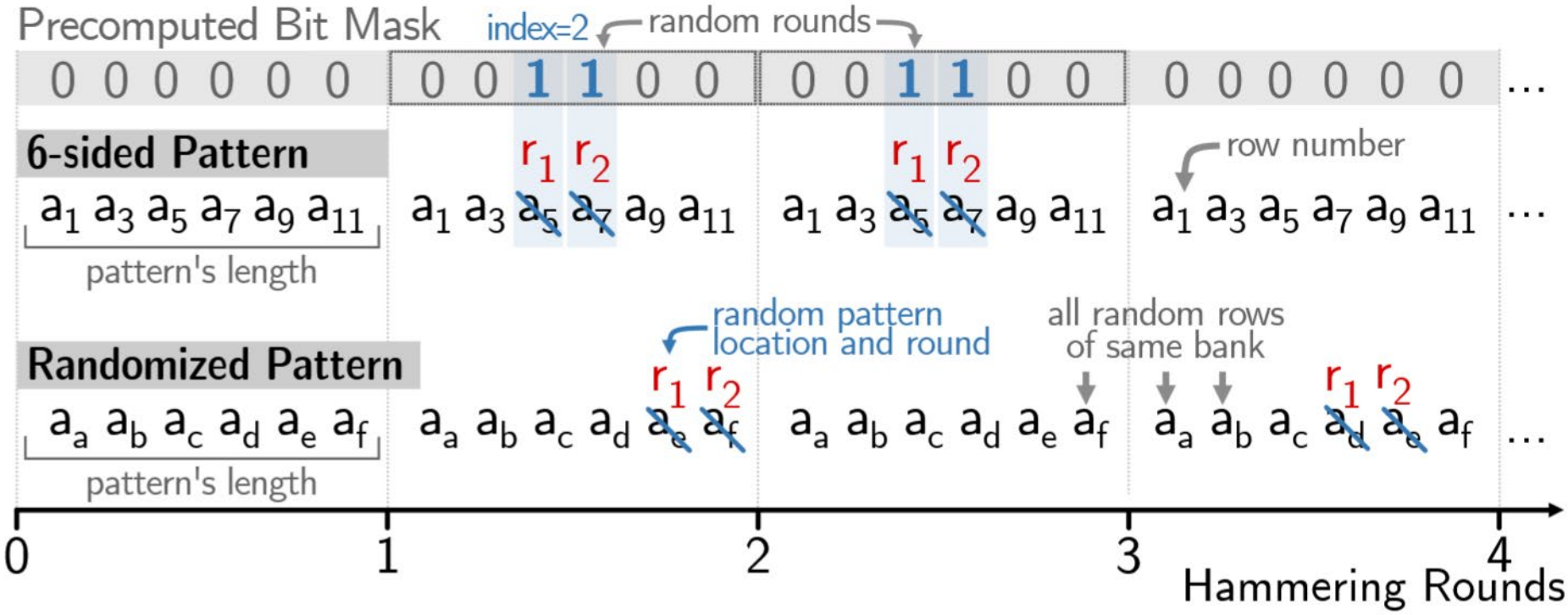
- Rowhammer is solved this time, right?

DIMM	TRRespass [12]			
	$\lvert\mathbb{P}^{+}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{fuzz}}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{swp}}\rvert$	$\lvert\mathbb{F}^{0\to1}_{\text{swp}}\rvert$
$\mathcal{A}_0$	0	–	–	–
$\mathcal{A}_1$	12	12	5	5
$\mathcal{A}_2$	715	16,054	7,404	4,563
$\mathcal{A}_3$	326	852	114	58
$\mathcal{A}_4$	78	105	22	9
$\mathcal{A}_5$	0	–	–	–
$\mathcal{A}_6$	4	11	4	4
$\mathcal{A}_7$	0	–	–	–
$\mathcal{A}_8$	0	–	–	–
$\mathcal{A}_9$	14	844	1	1
$\mathcal{A}_{10}$	367	961	505	280
$\mathcal{A}_{11}$	261	479	38	25
$\mathcal{A}_{12}$	0	–	–	–
$\mathcal{A}_{13}$	0	–	–	–
$\mathcal{A}_{14}$	1	1	4	0
$\mathcal{A}_{15}$	0	–	–	–
$\mathcal{A}_{16}$	688	5,499	1,450	983
$\mathcal{A}_{17}$	711	12,196	3,871	2,690
$\mathcal{A}_{18}$	14	14	1	1
$\mathcal{A}_{19}$	0	–	–	–
$\mathcal{B}_0$	0	–	–	–
$\mathcal{B}_1$	0	–	–	–
$\mathcal{B}_2$†	7	8	5	3
$\mathcal{B}_3$	0	–	–	–
$\mathcal{B}_4$	0	–	–	–
$\mathcal{B}_5$	0	–	–	–
$\mathcal{B}_6$	0	–	–	–
$\mathcal{B}_7$	0	–	–	–
$\mathcal{B}_8$†	0	–	–	–
$\mathcal{B}_9$†	0	–	–	–
$\mathcal{C}_0$	0	–	–	–
$\mathcal{C}_1$	0	–	–	–
$\mathcal{C}_2$	0	–	–	–
$\mathcal{C}_3$	0	–	–	–
$\mathcal{C}_4$	0	–	–	–
$\mathcal{C}_5$	0	–	–	–
$\mathcal{D}_0$	0	–	–	–
$\mathcal{D}_1$	3	3	0	–
$\mathcal{D}_2$	0	–	–	–
$\mathcal{D}_3$	8	8	1	1

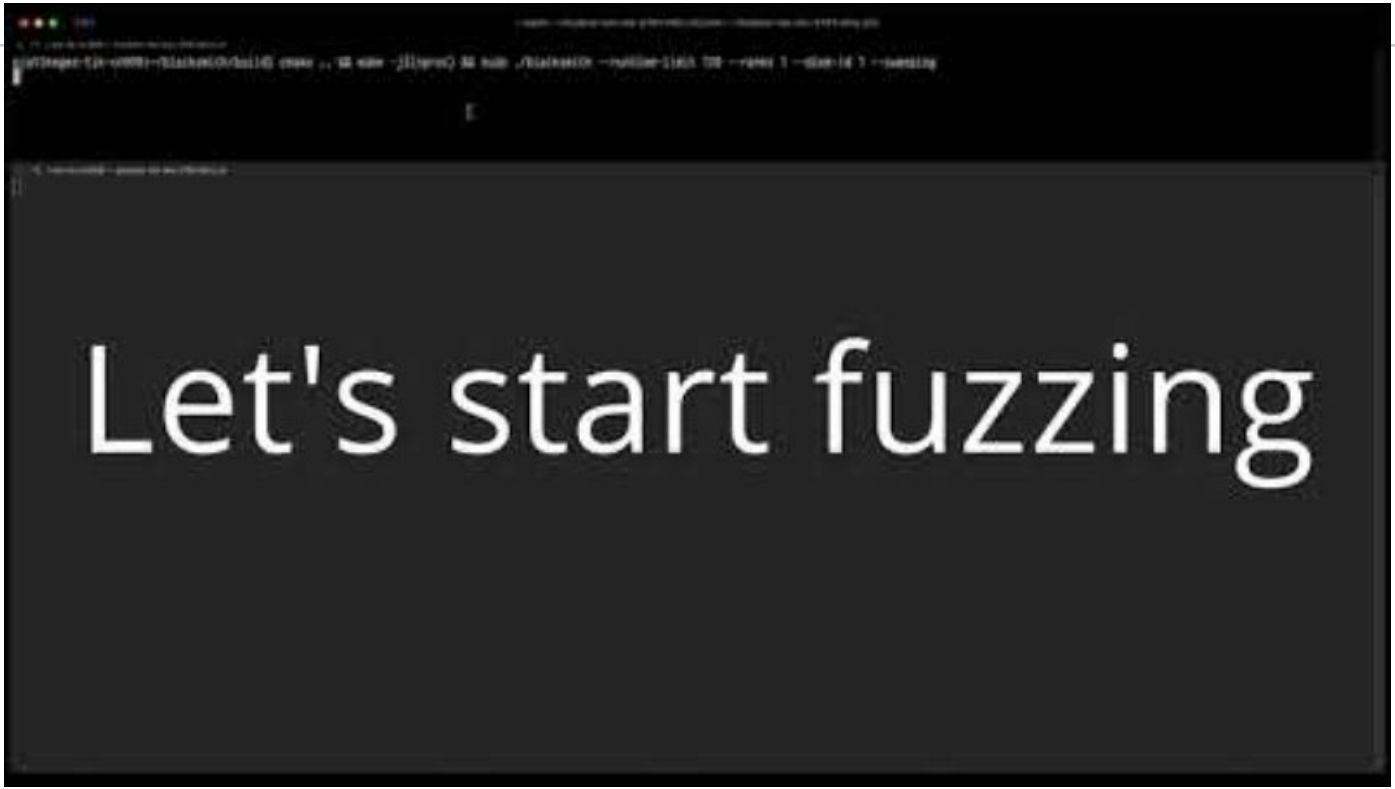
Blacksmith (Oakland 2022)

- Uniform patterns are easy to track
  - They maximize aggressor throughput
- Try non-uniform patterns to bypass TRR
  - Vary intensity, span multiple TREFI
- Bit flips found on all 40 dimms!

DIMM	Blacksmith				TRRespass [12]			
	$\lvert\mathbb{P}^{+}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{fuzz}}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{swp}}\rvert$	$\lvert\mathbb{F}^{0\to1}_{\text{swp}}\rvert$	$\lvert\mathbb{P}^{+}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{fuzz}}\rvert$	$\lvert\mathbb{F}^{\text{total}}_{\text{swp}}\rvert$	$\lvert\mathbb{F}^{0\to1}_{\text{swp}}\rvert$
$\mathcal{A}_0$	47	1,061	82,183	41,471	0	–	–	–
$\mathcal{A}_1$	116	2,125	12,134	6,095	12	12	5	5
$\mathcal{A}_2$	462	106,815	134,702	68,801	715	16,054	7,404	4,563
$\mathcal{A}_3$	82	239	1,746	890	326	852	114	58
$\mathcal{A}_4$	460	1,604	5,132	2,602	78	105	22	9
$\mathcal{A}_5$	42	7,771	113,190	57,655	0	–	–	–
$\mathcal{A}_6$	102	17,790	98,425	49,296	4	11	4	4
$\mathcal{A}_7$	66	3,415	32,090	15,988	0	–	–	–
$\mathcal{A}_8$	83	11,105	92,660	46,914	0	–	–	–
$\mathcal{A}_9$	349	1,176	4,889	2,461	14	844	1	1
$\mathcal{A}_{10}$	350	1,282	3,051	1,532	367	961	505	280
$\mathcal{A}_{11}$	202	632	3,171	1,630	261	479	38	25
$\mathcal{A}_{12}$	74	13,641	43,581	22,149	0	–	–	–
$\mathcal{A}_{13}$	72	9,889	59,721	30,320	0	–	–	–
$\mathcal{A}_{14}$	51	9,729	64,083	32,543	1	1	4	0
$\mathcal{A}_{15}$	67	8,333	52,580	26,483	0	–	–	–
$\mathcal{A}_{16}$	372	61,493	99,552	51,029	688	5,499	1,450	983
$\mathcal{A}_{17}$	425	57,245	138,601	70,902	711	12,196	3,871	2,690
$\mathcal{A}_{18}$	126	12,689	80,601	40,876	14	14	1	1
$\mathcal{A}_{19}$	107	2,543	11,599	5,736	0	–	–	–
$\mathcal{B}_0$	9	11	63	22	0	–	–	–
$\mathcal{B}_1$	7	14	506	256	0	–	–	–
$\mathcal{B}_2$†	9	41	15	7	7	8	5	3
$\mathcal{B}_3$	1	2	111	58	0	–	–	–
$\mathcal{B}_4$	101	177	1,107	577	0	–	–	–
$\mathcal{B}_5$	19	24	14	6	0	–	–	–
$\mathcal{B}_6$	18	41	78	46	0	–	–	–
$\mathcal{B}_7$	4	4	70	34	0	–	–	–
$\mathcal{B}_8$†	4	6	258	131	0	–	–	–
$\mathcal{B}_9$†	40	86	1,223	625	0	–	–	–
$\mathcal{C}_0$	1	3	26	16	0	–	–	–
$\mathcal{C}_1$	16	29	28	8	0	–	–	–
$\mathcal{C}_2$	82	282	2,551	1,242	0	–	–	–
$\mathcal{C}_3$	6	7	636	296	0	–	–	–
$\mathcal{C}_4$	31	57	769	385	0	–	–	–
$\mathcal{C}_5$	23	58	1,028	516	0	–	–	–
$\mathcal{D}_0$	26	250	10,646	5,329	0	–	–	–
$\mathcal{D}_1$	37	458	6,655	3,406	3	3	0	–
$\mathcal{D}_2$	3	16	2,030	1,008	0	–	–	–
$\mathcal{D}_3$	41	463	6,797	3,475	8	8	1	1
$\sum$	**4,133**		**1.168 M**		**3,209**		**13,425**	


Precomputed Bit Mask
index=2
random rounds
0 0 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 0 0 0 0
6-sided Pattern
r1 r2
r1 r2
row number
a1 a3 a5 a7 a9 a11
a1 a3 a5 a7 a9 a11
a1 a3 a5 a7 a9 a11
a1 a3 a5 a7 a9 a11
pattern's length
random pattern location and round
all random rows of same bank
Randomized Pattern
r1 r2
r1 r2
aa ab ac ad ae af
aa ab ac ad ae af
aa ab ac ad ae af
aa ab ac ad ae af
pattern's length
0
1
2
3
4
Hammering Rounds

Blacksmith (Oakland 2022)



Browser Attacks from DDR4

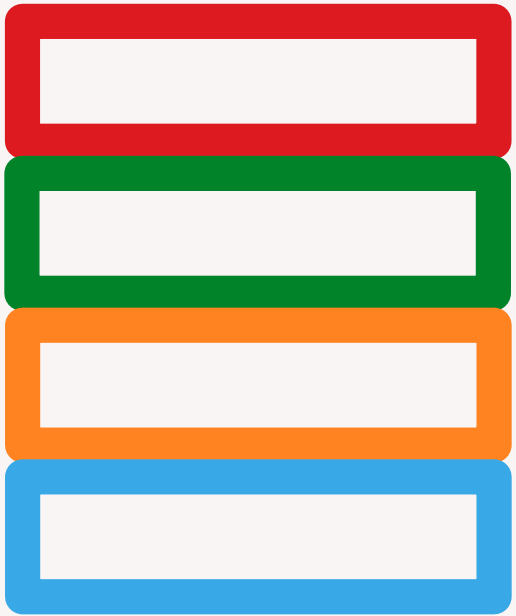
- No clflush in browser
  - Have to use eviction sets
- Eviction sets for a many-sided hammering pattern is really slow

Cache Eviction

Eviction sets

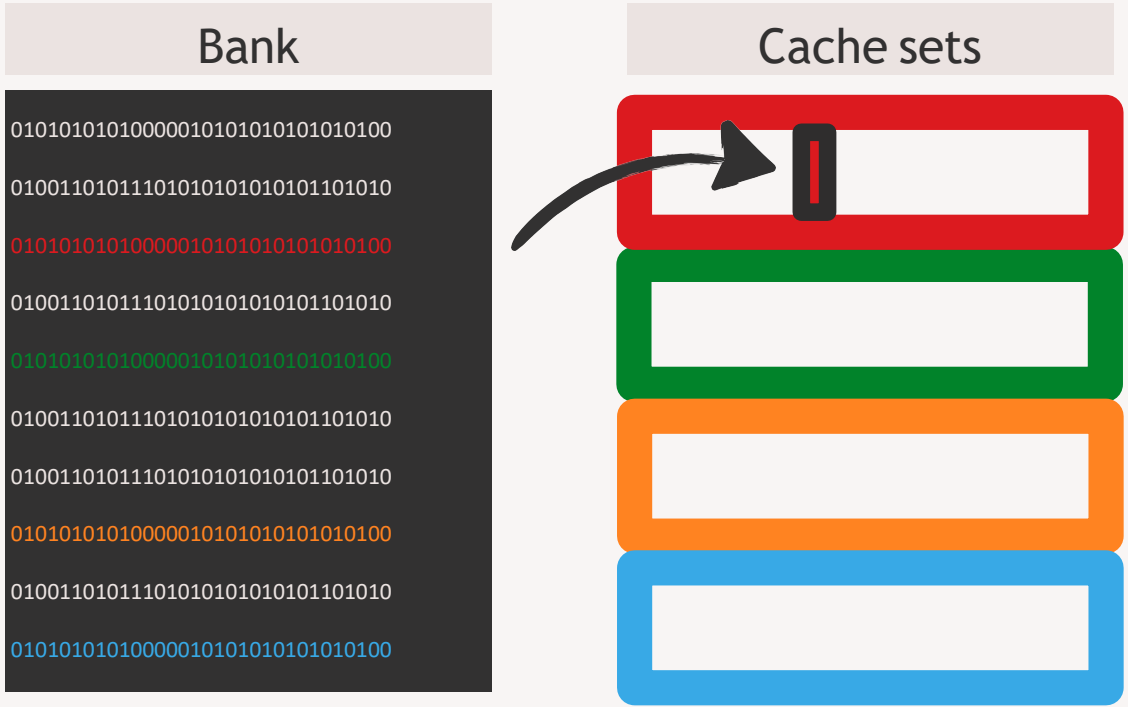
Bank

Cache sets



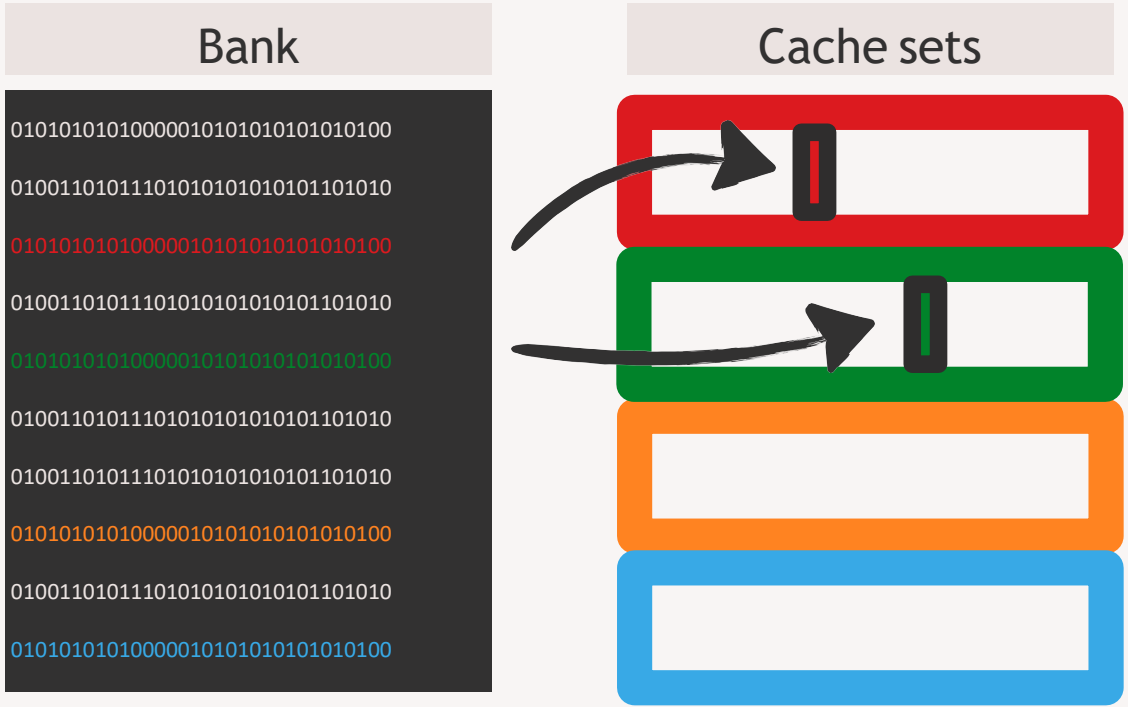
Cache Eviction

Eviction sets



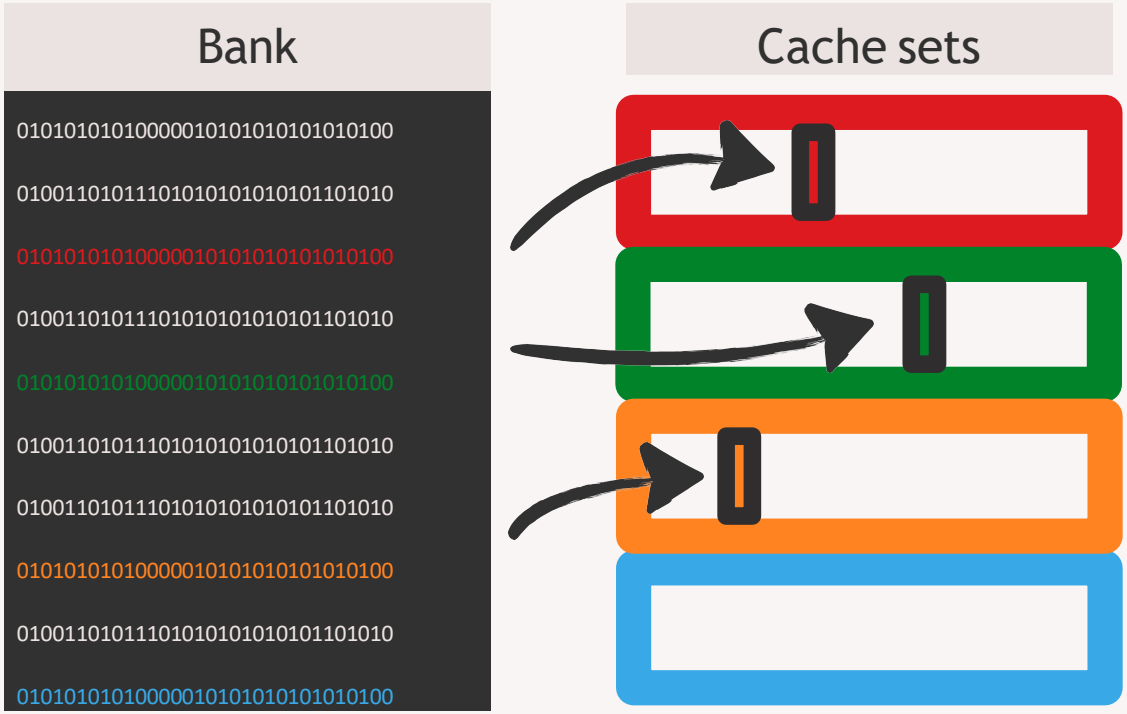
Cache Eviction

Eviction sets



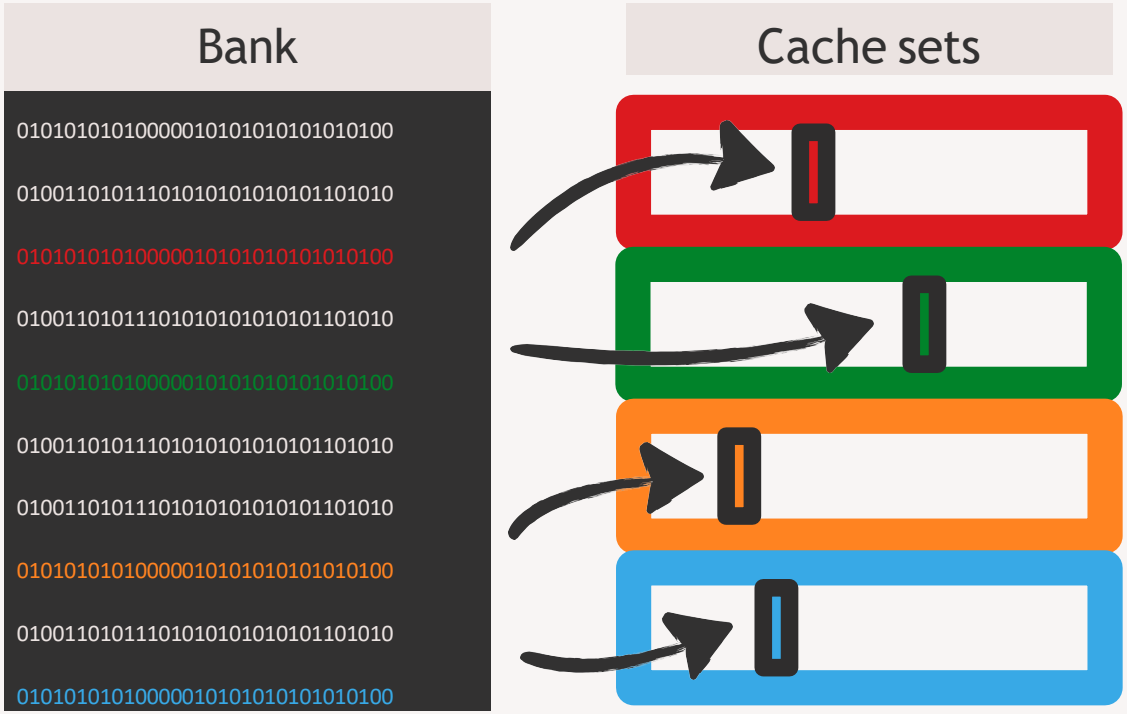
Cache Eviction

Eviction sets



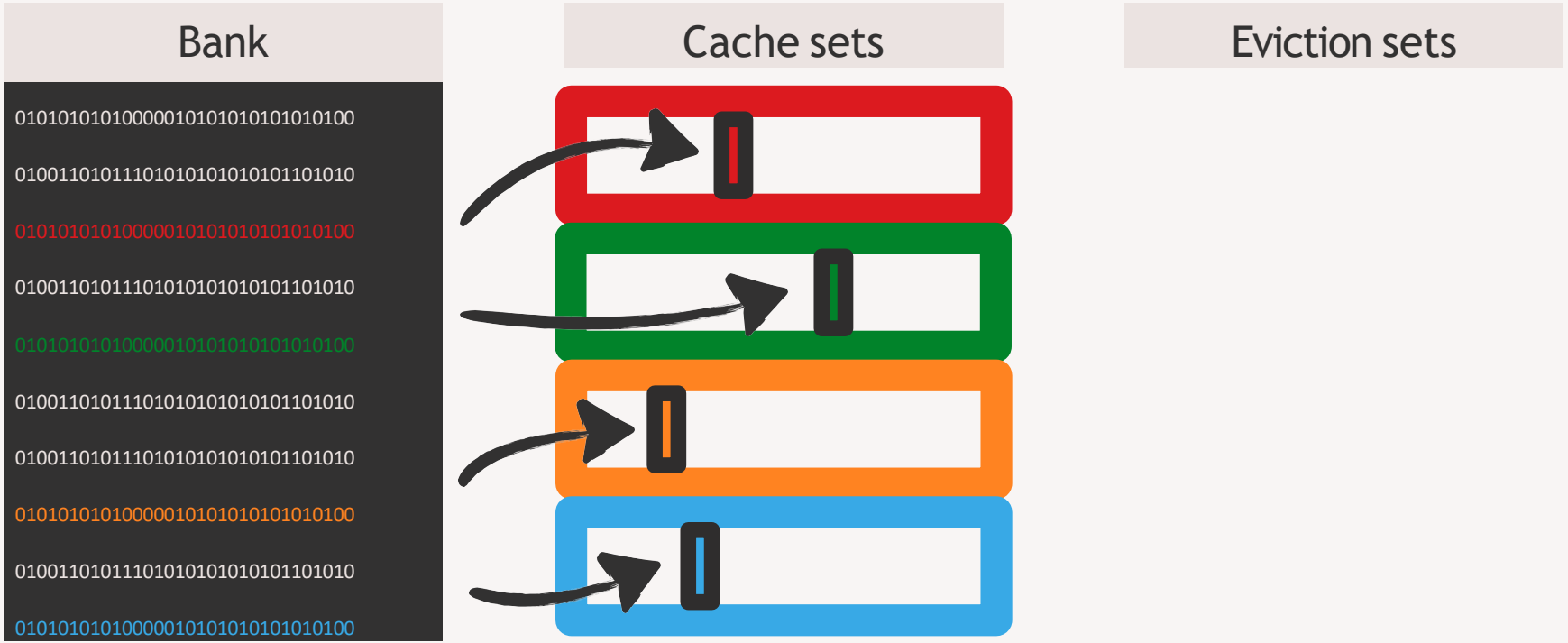
Cache Eviction

Eviction sets



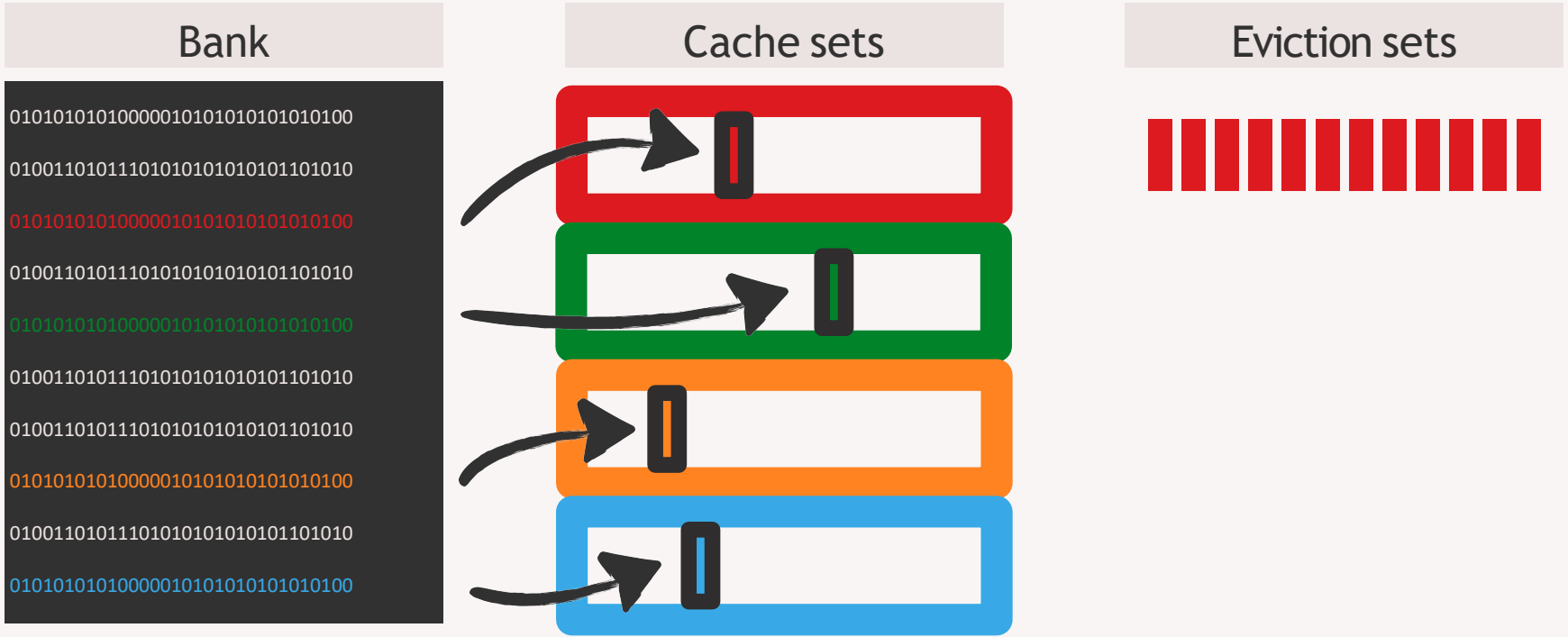
Cache Eviction

Eviction sets



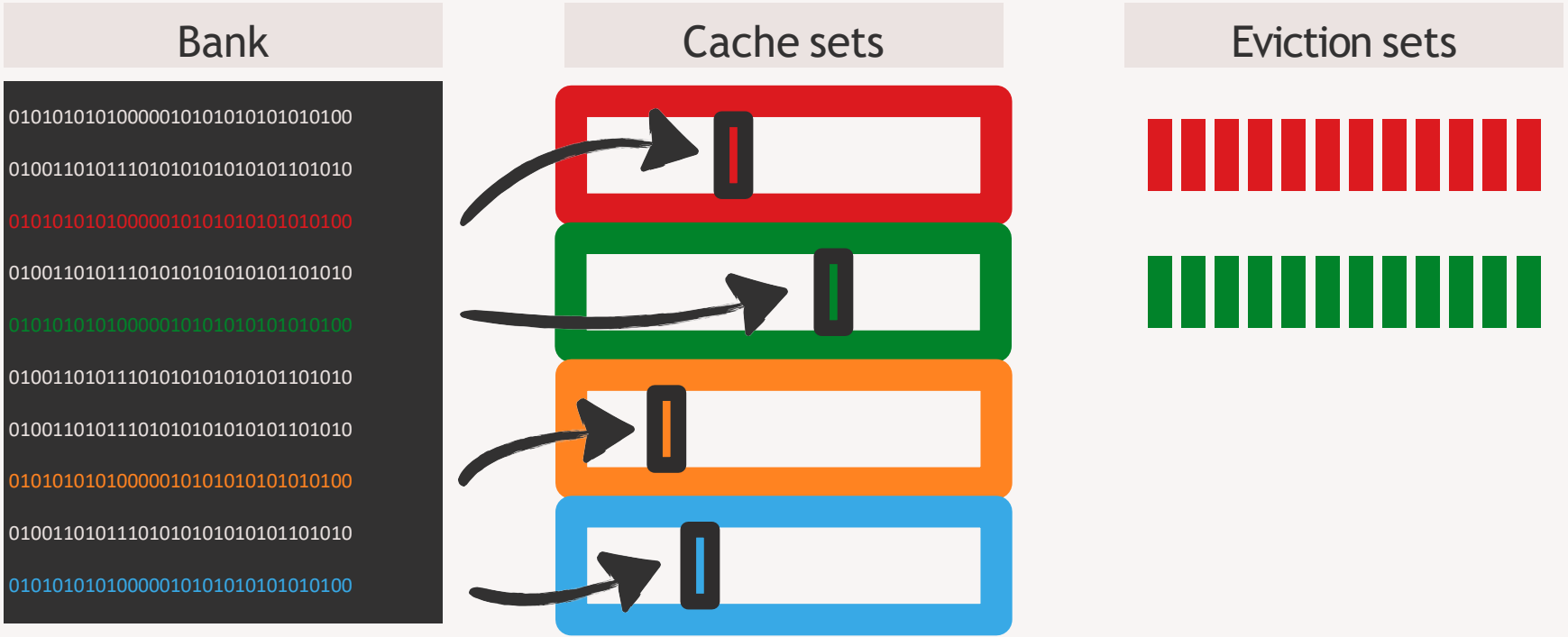
Cache Eviction

Eviction sets



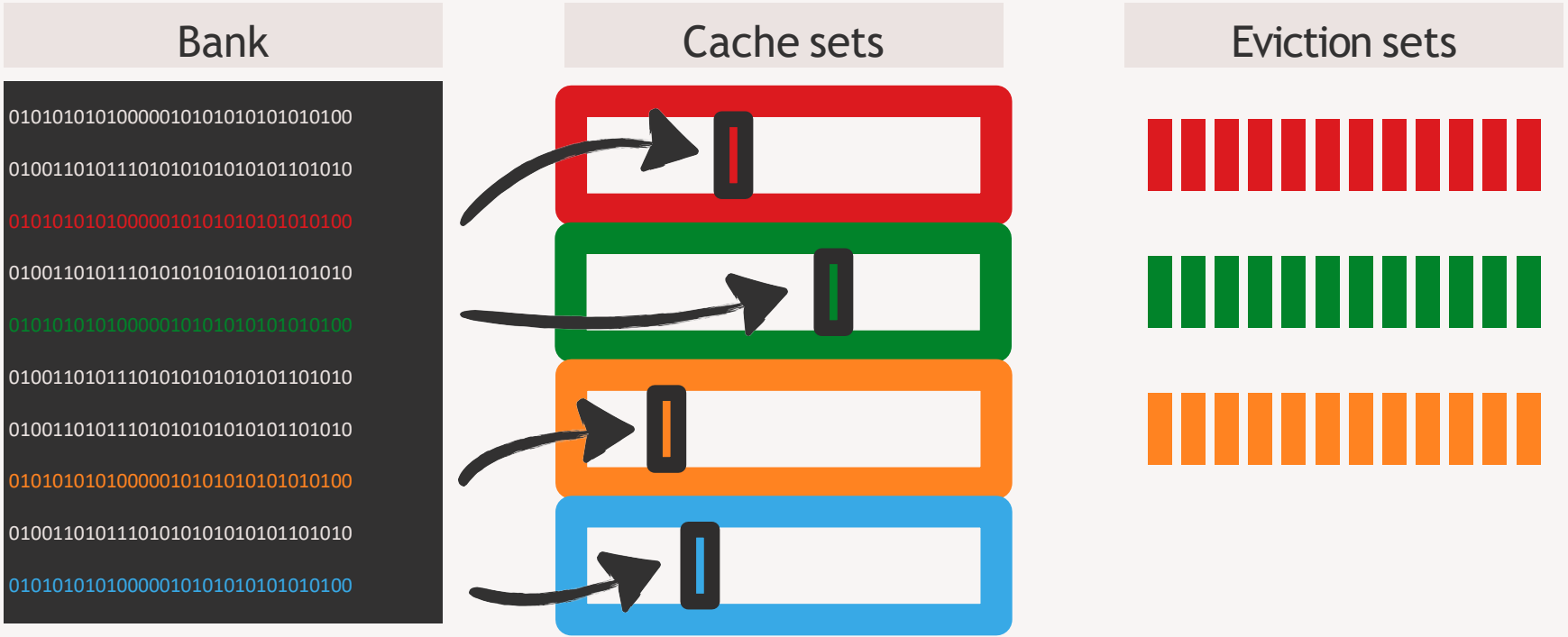
Cache Eviction

Eviction sets



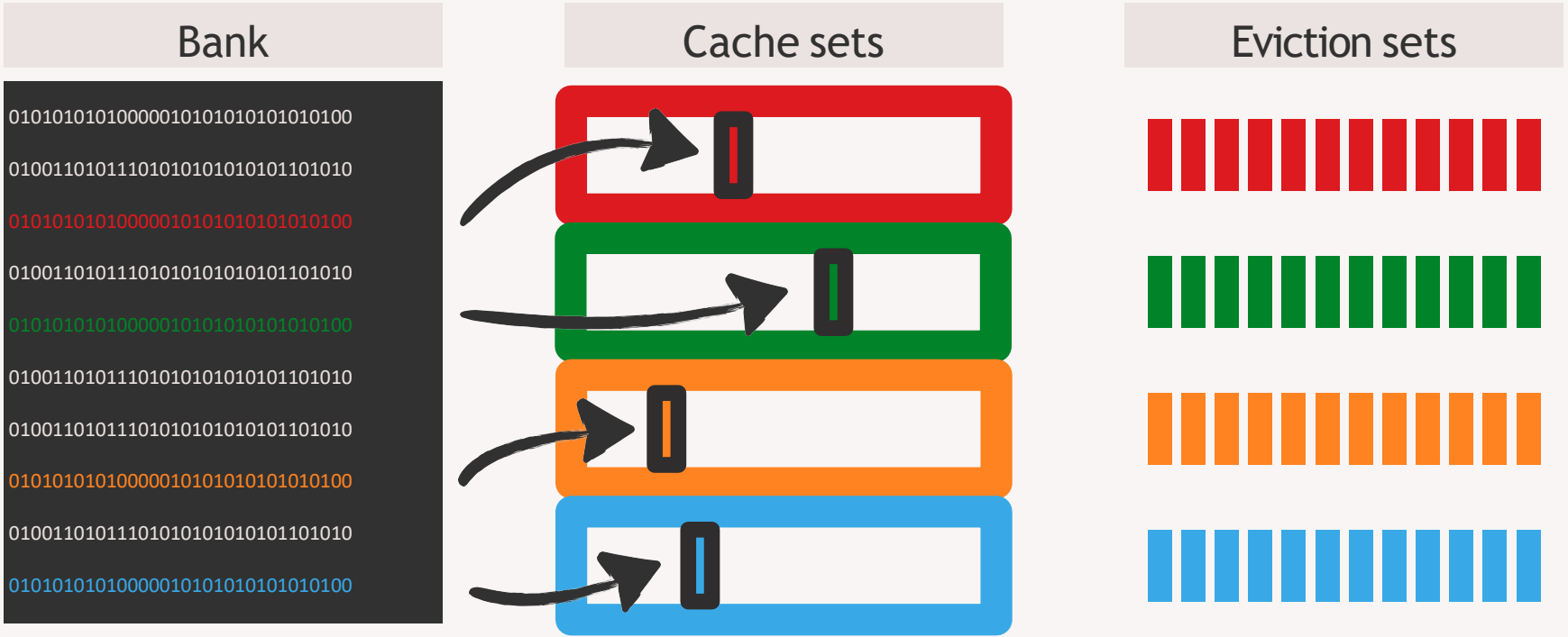
Cache Eviction

Eviction sets



Cache Eviction

Eviction sets

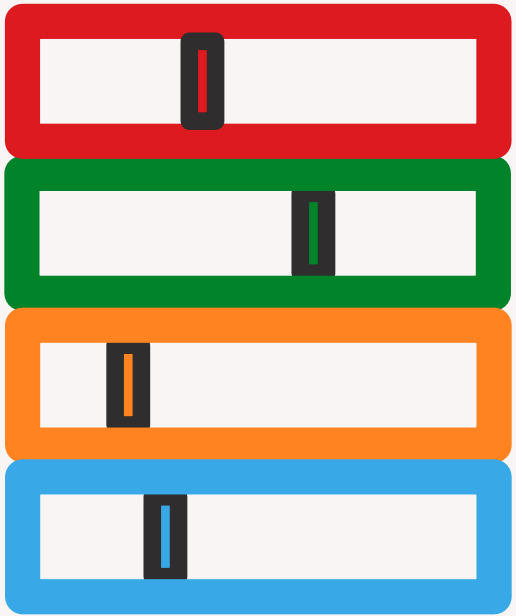


Cache Eviction

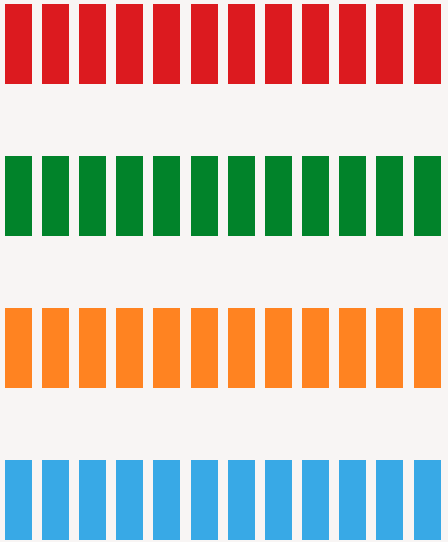
Eviction sets

Bank

Cache sets

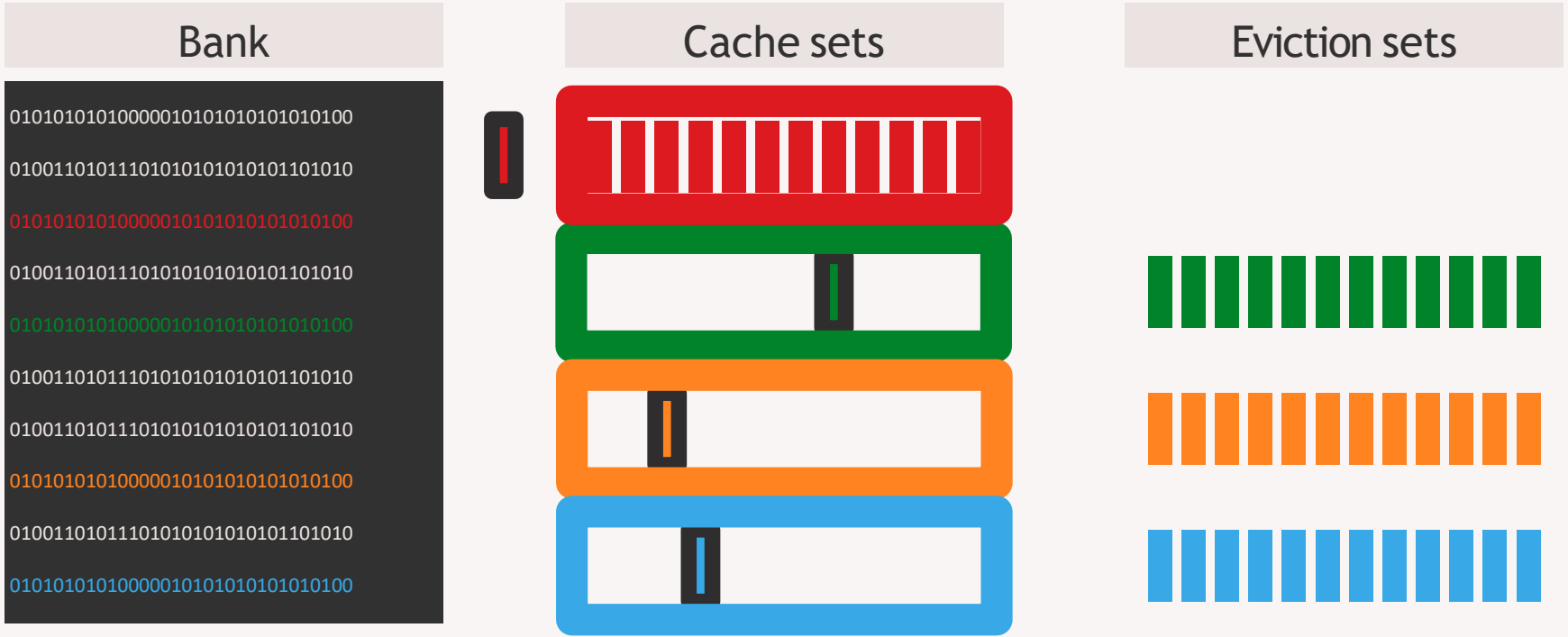


Eviction sets



Cache Eviction

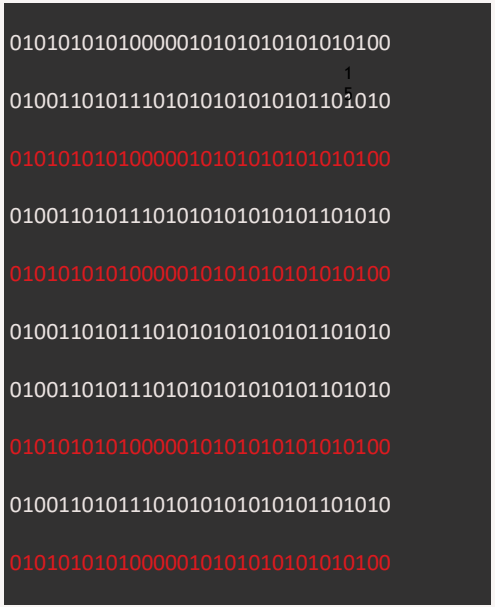
Eviction sets



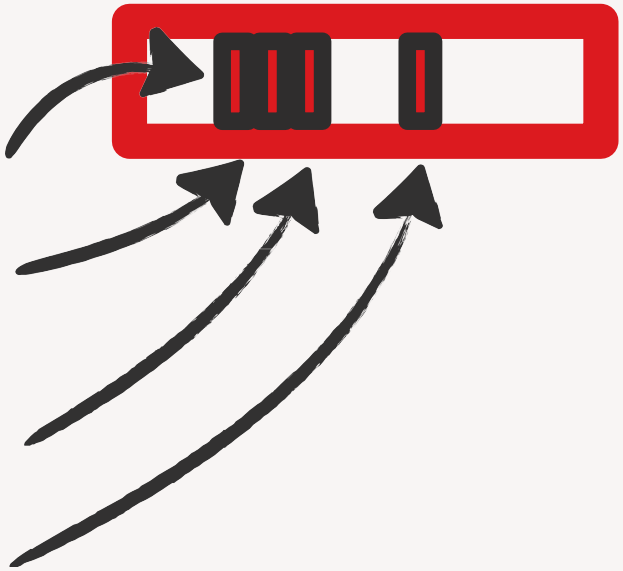
Cache Eviction

Same set?

Bank



Cache sets

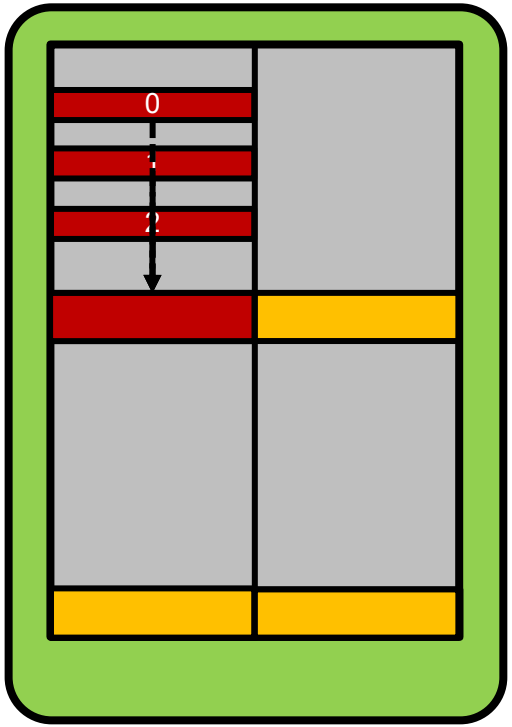


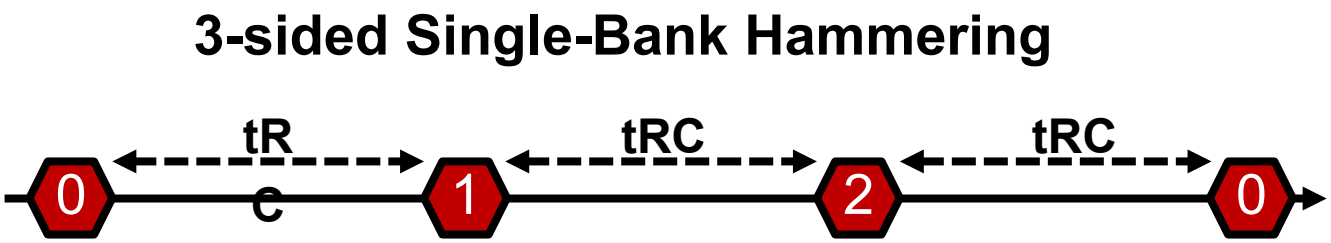
Eviction sets

Many-sided Rowhammer in JavaScript Is Possible!

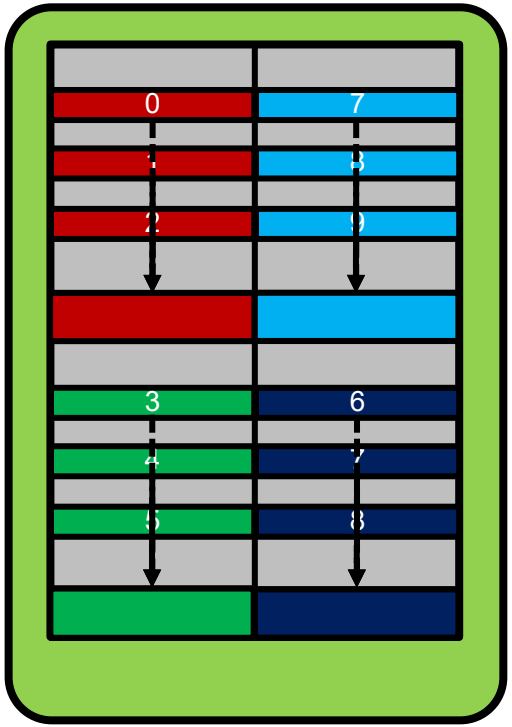
- Possible if you create a synchronized self-evicting pattern
- Despite all hardware and software mitigations
- Re-enables past JavaScript-based Rowhammer attacks in 2021
    - Built exploit for breaking out of the sandbox

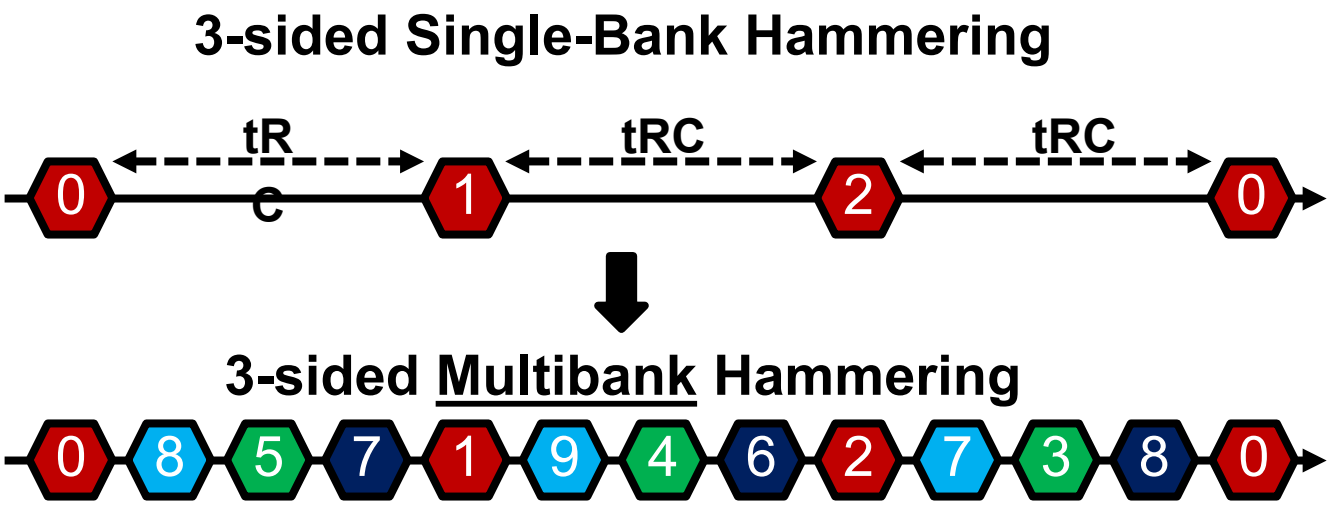
N-sided Hammering



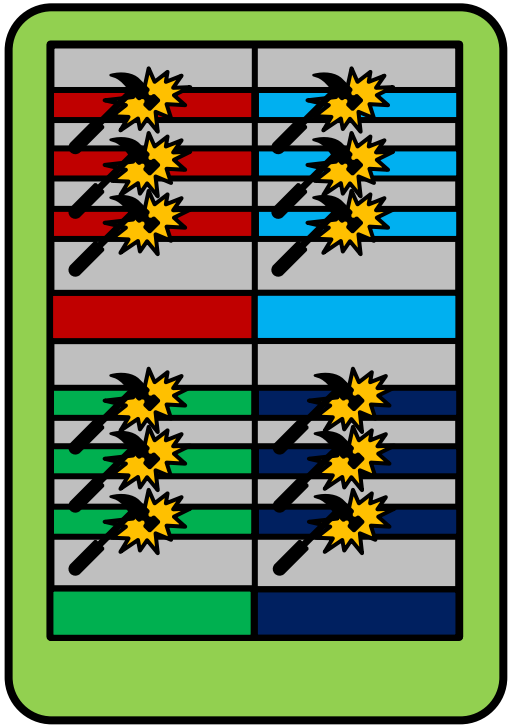


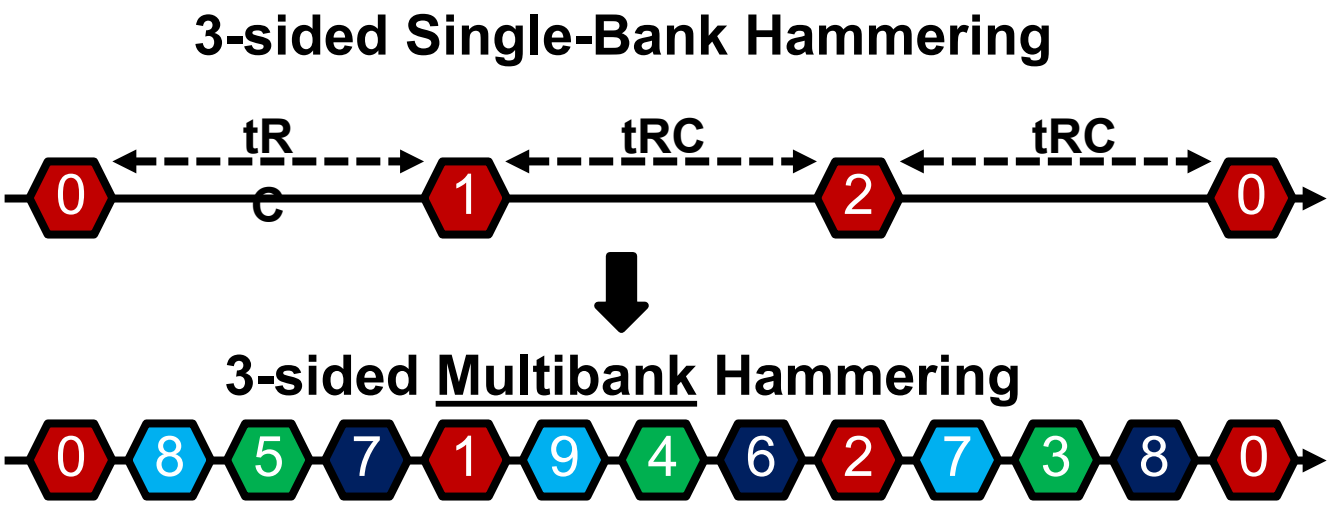
Multibank Hammering



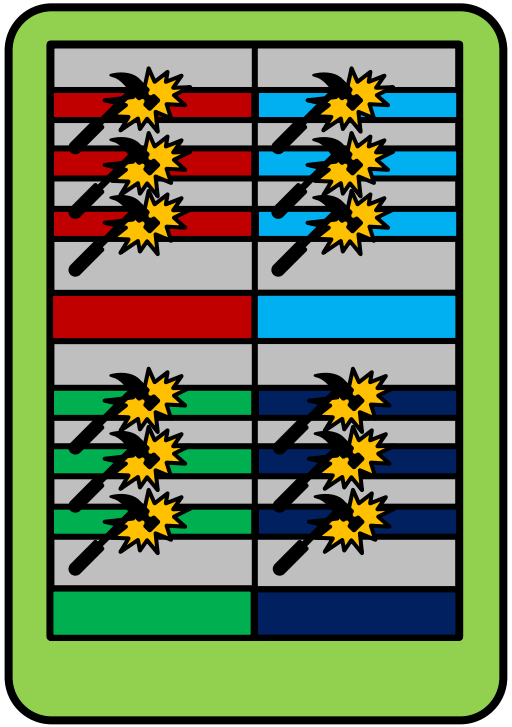


Multibank Hammering





Multibank Hammering



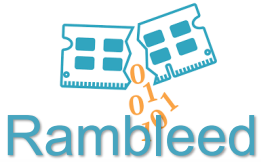
3-sided Multibank Hammering

8x more flips per iteration with 4-bank, i7-7700

19 flips/1000 iters, 2 bank, 12th gen

sudo 

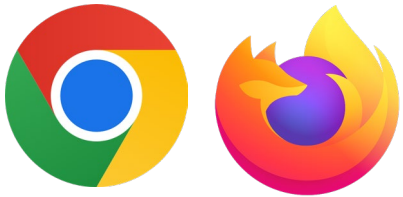
390 s average, 879× speedup



1.56 bits/s, first on DDR4 memory

Multibank Hammering

First successful Rowhammer attack under Default Browser Settings on DDR4

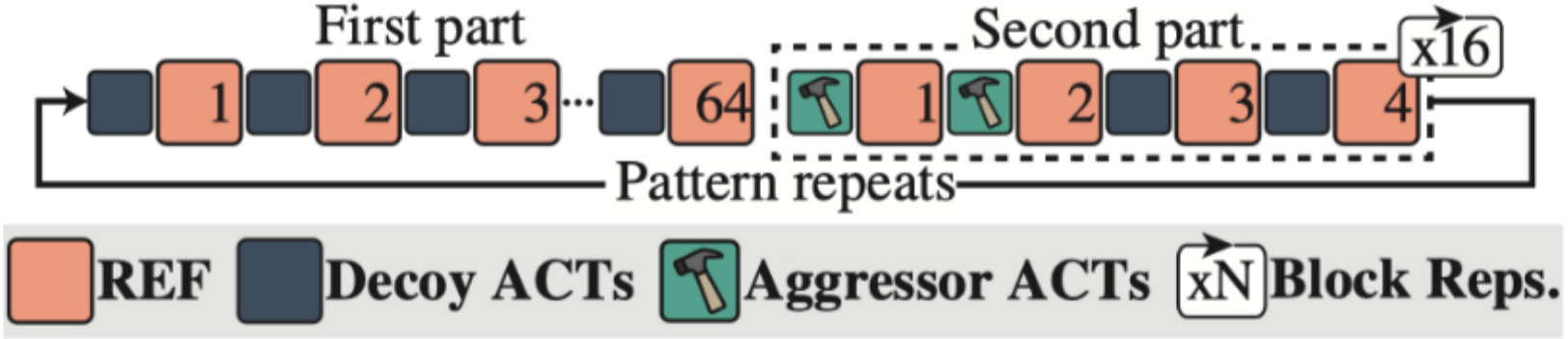


DDR5 Mitigations

- DDR4 is broken
  - Need a new generation of DRAM to start over
- DDR5 explicitly designed to be Rowhammer free
- Density results in bit flips at only 3200 activations

Phoenix

- State of the art techniques didn’t work on DDR5 DIMMS
- Even non-uniform patterns tracked
- What about non-form patterns spanning many refresh intervals?



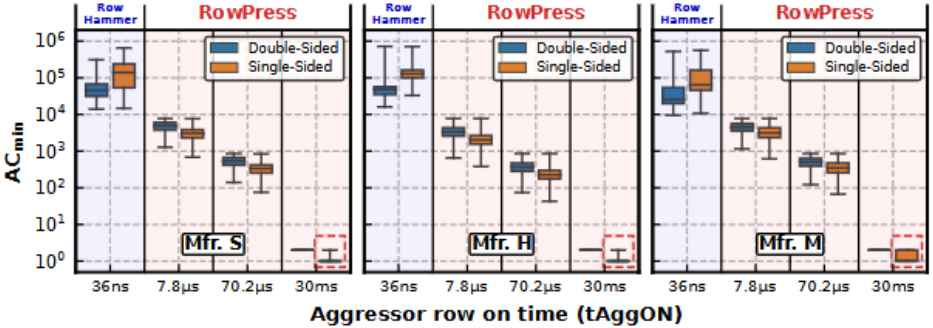
- Bit flips on all 15 Hynix DIMMS!

Table: Evaluation of DDR5 UDIMMs. For each DIMM, we report its manufacturing date (MfD.); size (Sz.); DRAM geometry (number of ranks (RK), bank groups (BG), banks per bank group (BA), and row bits to the power of two(R)); and the Rowhammer pattern it is vulnerable to (§ 5). For each DIMM, we sweep its pattern over 256MB of memory and report the number of one-to-zero (1→0) and zero-to-one (0→1) bit flips. We also report the average time to find the first exploitable bit flip for three Rowhammer attacks: the page table entry (PTE) attack, the RSA key attack, and the sudo binary attack.

ID	MfD.	Sz.	Geom.	Patterns		Sweep [#BFs]		Attacks [mm:hh]		
	[yy-mm]	[GiB]		(P128 / P2608)		1→0	0→1	PTE	RSA	sudo
H0	21-12	8	1,4,4,16	✓	✗	3 329	5 460	00:10	04:57	80:45
H1	22-07	16	1,8,4,16	✓	✗	2 917	4 582	00:14	07:06	49:37
H2	22-08	16	1,8,4,16	✓	✗	2 629	4 082	00:15	06:38	–
H3	22-12	8	1,4,4,16	✓	✗	2 837	4 526	00:11	06:36	–
H4	22-12	32	2,8,4,16	✗	✓	207	318	02:55	–	–
H5	22-12	32	2,8,4,16	✓	✗	3 922	5 821	00:11	04:17	–
H6	23-01	32	2,8,4,16	✓	✗	6 592	9 672	00:07	02:29	20:19
H7	23-01	32	2,8,4,16	✗	✓	566	860	00:39	–	–
H8	23-01	32	2,8,4,16	✗	✓	156	240	04:06	–	–
H9	23-01	32	2,8,4,16	✗	✓	314	486	02:17	09:25	–
H10	23-02	32	2,8,4,16	✗	✓	304	461	07:27	23:57	–
H11	24-01	16	1,8,4,16	✗	✓	12 523	18 446	00:06	01:19	12:41
H12	24-01	16	1,8,4,16	✓	✗	10 833	15 917	00:05	01:27	21:17
H13	24-04	16	1,8,4,16	✓	✗	8 520	12 761	0:005	01:38	–
H14	24-12	16	1,8,4,16	✗	✓	24	19	20:15	–	–

RowPress

- TRR mechanisms rely on identifying frequently activated rows
- Rowpress achieves equal disturbance effects, with far fewer activations



Proposed Rowhammer Mitigations

Zebram

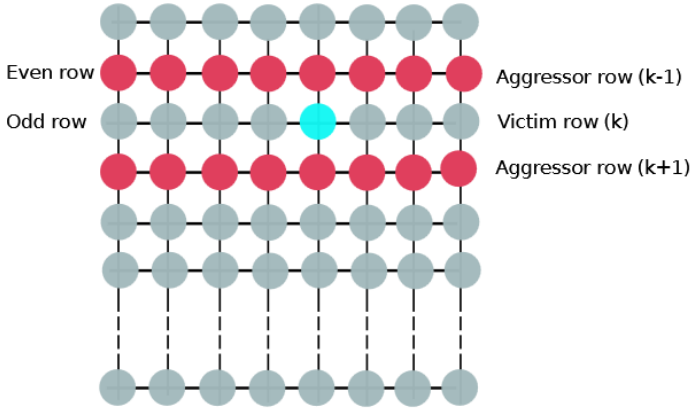


Figure 3: Hammering even-numbered rows can only induce bit flips in odd-numbered rows and vice versa.

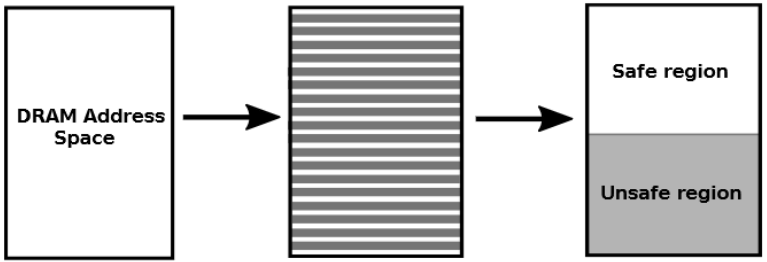
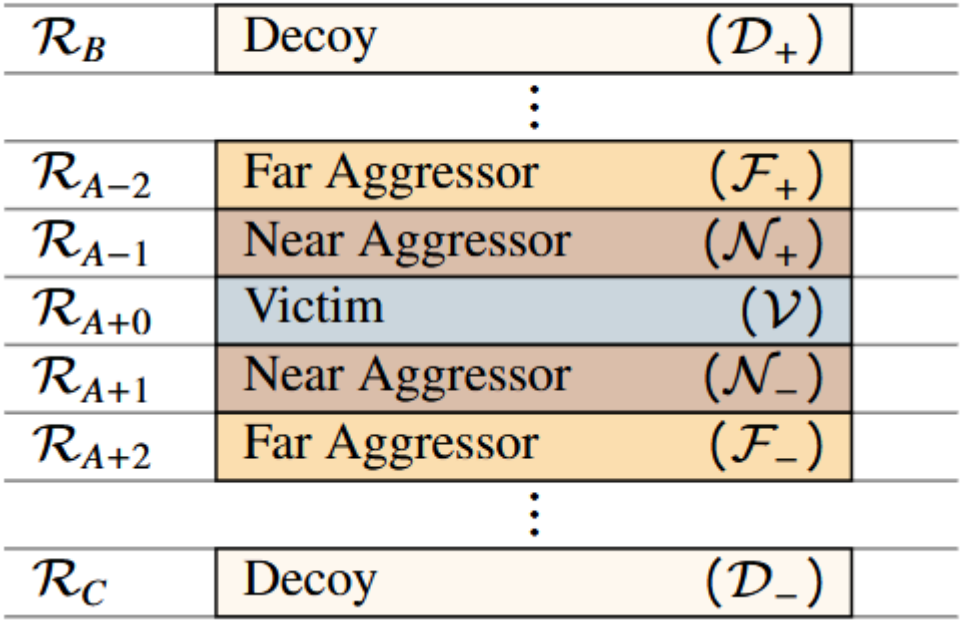


Figure 4: Splitting the memory into safe and unsafe regions using even and odd rows in a zebra pattern.

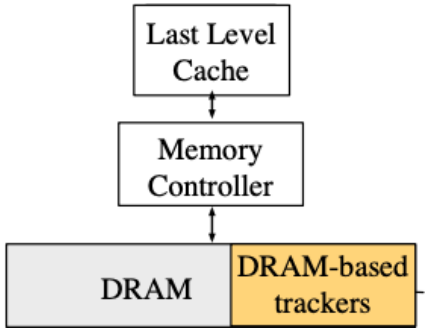
Half-Double: Hammering From the Next Row Over

- Flips bits from more than 1 row away



Counter-based Row Activation

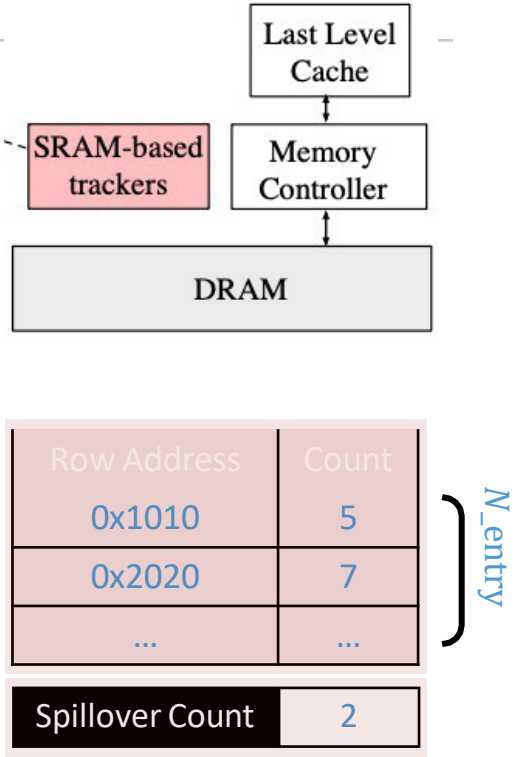
- Maintain a counter to track the number of accesses per row
  - Increment the counter when accessing a row
  - When reaching a threshold, activate the neighboring rows
  - After activating, reset the counter
- How much storage overhead for RAC?
  - Example: 8GB memory with 1M rows, each counter 2 bytes
  - Answer?
- What factors affect the performance overhead?



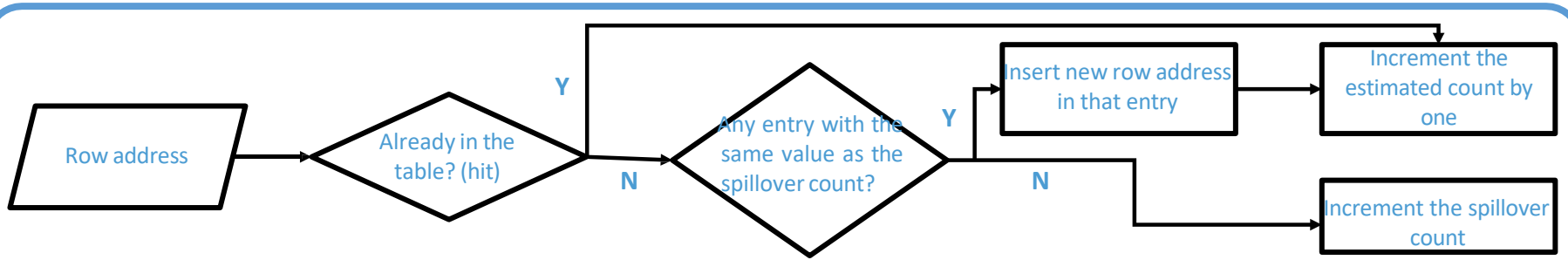
SRAM-based Trackers

- Naïve: one counter per row
  - What is the problem?
- Do it smartly: using the Misra-Gries Algorithm
  - The Rowhammer tracking problem is very similar to the frequent elements problem
  - Given a stream of $\boldsymbol{W}$ items, the algorithm identifies **all the items** that appear more than $\boldsymbol{T}$ times, as long as

$$N_{entry} > {}^{W}/_{T} - 1$$



Graphene Aggressor Tracking



Graphene Analysis

$$N_{entry} > {}^{W}/_{T} - 1$$

- In the original paper (2020)
  - W Max number of ACTs in a refresh window: 1,360K
  - T Threshold for aggressor tracking: 12.5K (actual threshold = 3.2K)
  - N_{entry} Number of table entries: 108
  - Each entry: 16 bits for row address; 15 bits for counting value up to T
  - Memory type: Content-addressable memory (CAM)

- Using more realistic values for W and T on modern machines results in over 50x higher N_entry
  - How about Rowpress?

Mitigation Design Considerations

- Cost
- Performance
- Usability
- security

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL