Understanding the Security Benefits and Overheads of Emerging Industry Solutions to DRAM Read Disturbance

Oğuzhan Canpolat
Giray Yağlıkçı  Geraldo Oliveira  Ataberk Olgun
Oğuz Ergin  Onur Mutlu
Executive Summary

Problem:
- DRAM continues to become more vulnerable to read disturbance
- Latest update (April 2024) to DDR5 standard introduces **Per Row Activation Counting (PRAC)** to mitigate read disturbance
- No prior work investigates PRAC’s **security** and **performance**

Goal: Rigorously analyze and characterize the **security** and **performance** implications of the DDR5 standard **PRAC** mechanism

Mathematical analysis & extensive simulations show that PRAC:
- provides security as long as no bitflip occurs below **10 activations**
- has **non-negligible** performance (10%) and energy (18%) **overheads**
- **poorly scales** for future DRAM chips, leading to significant **overheads** on performance (49%) and energy (136%)
- allows memory performance attacks to hog significant amount of **DRAM throughput** (up to 79% throughput loss)

[SAFARI](https://github.com/CMU-SAFARI/ramulator2)
DRAM Organization

DRAM Channel

DRAM Module

SAFARI
DRAM Organization

- DRAM Array
- Bank
- Bitline
- Wordline
- Sense amplifiers (row buffer)
- DRAM Cell
- Off-chip channel
1 **ACTIVATE**: Fetch the row into the **row buffer**

2 **READ/WRITE**: Retrieve or update data

3 **PRECHARGE**: Prepare the array for a new ACTIVATE

**ACTIVATION** and **PRECHARGE** are time-consuming operations
DRAM Access Latency

Command
- ACTIVATE
- PRECHARGE

Data

Duration
row-cycle time ($t_{RC}$)

Next ACT
Repeatly **opening** (activating) and **closing** (precharging) a DRAM row causes **read disturbance bitflips** in nearby cells.
The minimum number of activations that causes a bitflip is called the **RowHammer threshold** ($N_{RH}$).
Read Disturbance Vulnerabilities (II)

- DRAM chips are more vulnerable to read disturbance today.

- Read disturbance bitflips occur at much lower activation counts (more than two orders of magnitude decrease in less than a decade):

  - 139K [Kim+, ISCA’14]
  - 9.6K [Kim+, ISCA’20]
  - <1K (RowPress) [Luo+, ISCA’23]

It is critical to prevent read disturbance bitflips effectively and efficiently for highly vulnerable systems.
Existing RowHammer Mitigations (I): Preventive Refresh

**DRAM Subarray**

Row 0  \textit{Victim Row}

Row 1  \textit{Victim Row}

Row 2  \textit{Aggressor Row}

Row 3  \textit{Victim Row}

Row 4  \textit{Victim Row}

\textbf{Refreshing} potential victim rows \textit{mitigates} RowHammer bitflips

[Kim+ ISCA'20]
Existing RowHammer Mitigations (II): DRAM Aggressor Row Tracking or Estimation

Necessary to accurately **track** or **estimate** aggressor DRAM row activation counts to **preventively refresh** potential victim rows

[Kim+ ISCA'20]
Outline

Background

Industry Solutions to Read Disturbance

Security Analysis

Performance and Energy Evaluation

Memory Performance Attacks

Conclusion
Preventive refresh is a **blocking** operation

Memory controller could cause **faulty** operation by accessing the memory module during refresh
Earlier JEDEC DDR5 specifications introduce Refresh Management (RFM) commands. Memory controller sends an RFM command to allow time for preventive refreshes.
Periodic Refresh Management (PRFM)
Memory controller *periodically* issues RFM commands.

Per Row Activation Counting and Back-Off (PRAC)
DRAM chip *tracks* row activations and *requests* RFMs by sending *back-off* signals.
Industry Solutions to Read Disturbance: Periodic Refresh Management (PRFM)

PRFM tracks activations with low accuracy, causing high number of preventive refreshes, leading to large performance and energy overheads.
Periodic Refresh Management (PRFM)
Memory controller *periodically* issues RFM commands

Per Row Activation Counting and Back-Off (PRAC)
DRAM chip *tracks* row activations and *requests* RFMs by sending *back-off*
**Industry Solutions to Read Disturbance: Per Row Activation Counting**

<table>
<thead>
<tr>
<th>Counters</th>
<th>DRAM Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

PRAC allows **accurate tracking** of aggressor row activations.
Industry Solutions to Read Disturbance: Per Row Activation Counting DRAM Timings

<table>
<thead>
<tr>
<th>Counters</th>
<th>DRAM Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
<tr>
<td>0</td>
<td>10101010101010101010101010</td>
</tr>
</tbody>
</table>

Row counter updates are not completely parallelized with DRAM access.

**PRAC increases row-cycle time** \((t_{RC})\) by \(~10\%\)

\[\text{PRAC increases row-cycle time}\]

\[\text{(t}_{\text{RC}}\text{) by }\sim10\%\]

\[\text{row-cycle time (t}_{\text{RC}}\text{)}\]
## Industry Solutions to Read Disturbance: Per Row Activation Counting DRAM Timings

Timing parameter changes for DDR5-3200AN speed bin
[JEDEC JESD79-5C, April 2024]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Change</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{RP}$</td>
<td>+21ns</td>
<td>+140%</td>
</tr>
<tr>
<td>$t_{RAS}$</td>
<td>-16ns</td>
<td>-50%</td>
</tr>
<tr>
<td>$t_{RTP}$</td>
<td>-2.5ns</td>
<td>-33%</td>
</tr>
<tr>
<td>$t_{WR}$</td>
<td>-20ns</td>
<td>-66%</td>
</tr>
<tr>
<td>$t_{RC}$</td>
<td>+5ns</td>
<td>+10%</td>
</tr>
</tbody>
</table>
Industry Solutions to Read Disturbance: Per Row Activation Counting (PRAC)

Back-Off Threshold

Row Counters
0
0
0

DRAM Commands
ACT ACT ACT ACT RFM RFM

normal traffic (180 ns) recovery (N RFMs)

PRAC-N
Mathematical Security Analysis Methodology

- **Wave attack** [Yağlıkçı+ 2021]: worst-case access pattern
  - maximizes hammer count by using decoy rows
  - on a system with **PRFM**
  - on a system with **PRAC**

- **Parameters:**
  - **Starting row set size**: # of rows that the wave attack hammers
  - **RFM threshold** (PRFM)
  - **Back-Off threshold** (PRAC)

- **Result**: Worst possible (highest) activation count that an attacker can achieve to a row
Security Analysis: Secure PRFM Configurations

Wave Attack Parameter

Starting Row Set Size ($|R_1|$)
- 2K
- 8K
- 32K
- 4K
- 16K
- 64K

Defense Parameter

RFM Threshold ($RFM_{th}$)

Highest activation count an attacker can achieve

Higher is worse
Security Analysis: Secure PRFM Configurations

Wave Attack Parameter

Starting Row Set Size ($|R_1|$)

- 2K
- 8K
- 32K
- 4K
- 16K
- 64K

RowHammer Threshold = 1024

RFM Threshold ($RFM_{th}$)

Defense Parameter

Higher is worse

Highest activation count an attacker can achieve
Security Analysis: Secure PRFM Configurations

The chart illustrates the highest activation count an attacker can achieve for different starting row set sizes. The x-axis represents the RFM threshold ($RFM_{th}$), and the y-axis shows the highest activation count. The legend indicates the starting row set sizes, with different colors representing 2K, 8K, 32K, 4K, 16K, and 64K. The higher the activation count for a given RFM threshold, the worse the configuration is considered to be.
Security Analysis: Secure PRFM Configurations

PRFM must send RFM commands **very frequently** (every ~8 ACTs) to prevent bitflips at **low** RowHammer thresholds (below 128)
Security Analysis: Secure PRAC Configurations

Highest activation count an attacker can achieve

Higher is worse

Less frequent back-off signals

Back-Off Threshold ($N_{BO}$)
Security Analysis: Secure PRAC Configurations

PRAC is configurable for secure operation against RowHammer thresholds as low as 10.
Outline

Background

Industry Solutions to Read Disturbance

Security Analysis

Performance and Energy Evaluation

Memory Performance Attacks

Conclusion
Evaluation Methodology

• **Performance and energy consumption evaluation:** cycle-level simulations using Ramulator 2.0 [Luo+, CAL 2023] and DRAMPower [Chandrasekar+, DATE 2013]

• **System Configuration:**
  - **Processor**: 4 cores, 4.2GHz clock frequency, 4-wide issue, 128-entry instruction window
  - **DRAM**: DDR5, 1 channel, 2 rank/channel, 8 bank groups, 4 banks/bank group, 64K rows/bank
  - **Memory Ctrl.**: 64-entry read and write requests queues, Scheduling policy: FR-FCFS with a column cap of 4
  - **Last-Level Cache**: 8 MiB (4-core)

• **Comparison Points**: 3 state-of-the-art RowHammer mitigations
  - Best-performing: Graphene [Park+ 2020]
  - Lowest processor chip area: PARA [Kim+ 2014]
  - Area-optimized best-performing: Hydra [Qureshi+ 2022]

• **Workloads**: 60 4-core workload mixes
  - SPEC CPU2006, SPEC CPU2017, TPC, MediaBench, YCSB
Performance Comparison: Industry Solutions

1. **PRFM**
   Memory controller *periodically* issues RFM

2. **PRAC-N**
   Memory controller issues $N$ RFMs each with *back-off*

3. **PRAC+PRFM**
   Memory controller issues RFM *periodically* and with *back-offs*

4. **PRAC-Optimistic**
   PRAC-4 with *no* change in DRAM timing parameters
Experimental Results: Performance Overhead and Its Scaling

Higher is better

Normalized Weighted Speedup

RowHammer Threshold ($N_{RH}$)

Graphene PARA PRAC-4 PRAC+PRFM
Hydra PRFM PRAC-1 PRAC-Optimistic

Lower is worse
Experimental Results: Performance Overhead and Its Scaling

PRAC has non-negligible performance overhead (10%) due to increased access latency.
Experimental Results: Performance Overhead and Its Scaling

Graphene and Hydra outperform PRAC at relatively high $N_{RH}$ values.
Experimental Results: Performance Overhead and Its Scaling

Above $N_{RH}$ of 32, PRAC overhead only slightly increases due to timely preventive refreshes.

Below $N_{RH}$ of 32, PRAC overhead significantly increases due to conservative thresholds against a wave attack.
Experimental Results: Performance Overhead and Its Scaling

PRAC-Optimistic outperforms all evaluated mitigation mechanisms (above $N_{RH}$ of 32)
Experimental Results: Performance Overhead and Its Scaling

PRFM’s system performance overheads significantly increase (by 37x) as $N_{RH}$ decreases.
Experimental Results: DRAM Energy Overhead and Its Scaling

Higher is worse

Lower is worse
PRAC has **non-negligible** DRAM energy overhead (18%) due to **increased** timing parameters.
Experimental Results: DRAM Energy Overhead and Its Scaling

Above $N_{RH}$ of 32, PRAC overhead only slightly increases due to timely preventive refreshes.

Below $N_{RH}$ of 32, PRAC overhead significantly increases due to conservative thresholds against a wave attack.
Experimental Results: DRAM Energy Overhead and Its Scaling

PRFM’s DRAM energy overhead significantly increase (to 33x) as $N_{RH}$ decreases
Outline

Background

Industry Solutions to Read Disturbance

Security Analysis

Performance and Energy Evaluation

Memory Performance Attacks

Conclusion
Memory Performance Attacks

Access pattern to trigger **most** back-offs with **fewest** activations possible by targeting a single row

Mathematically hogs up to **79% of DRAM throughput** of future DRAM chips

Degrades system performance by up to **65%** (53% on average)
More in the Paper

- **Detailed Background**
  - More information on PRAC and RFM

- **Security Analysis**
  - Threat Model
  - Secure Configurations

- **Evaluation**
  - Storage Analysis: PRAC and PRFM incur low storage overheads and scale well with decreasing \( N_{RH} \) values

- **Memory Performance Attacks**
  - Simulation results between \( N_{RH} \) values of 128 and 16
We present the first rigorous security, performance, energy, and cost analyses of the state-of-the-art on-DRAM-die read disturbance mitigation method, widely known as Per Row Activation Counting (PRAC), with respect to its description in the updated (as of April 2024) JEDEC DDR5 specification. Unlike prior state-of-the-art that advises the memory controller to periodically issue a DRAM command called refresh management (RFM), which provides the DRAM chip with time to perform its countermeasures, PRAC introduces a new back-off signal. PRAC’s back-off signal propagates from the DRAM chip to the data integrity of other physically close but unaccessed DRAM rows. RowHammer [1] is a prime example of DRAM read disturbance, where a DRAM row (i.e., victim row) can experience bitflips when at least one nearby DRAM row (i.e., aggressor row) is repeatedly activated (i.e., hammered) [1, 3–69] more times than a threshold, called the \textit{minimum hammer count to induce the first bitflip} ($N_{RH}$). RowPress [70] is another prime example of DRAM read disturbance that amplifies the effect of RowHammer and consequently reduces $N_{RH}$. 

https://arxiv.org/abs/2406.19094
Open Sourced

https://github.com/CMU-SAFARI/ramulator2
Outline

Background

Industry Solutions to Read Disturbance

Security Analysis

Performance and Energy Evaluation

Memory Performance Attacks

Conclusion
Conclusion

We rigorously analyzed and characterized the security and performance implications of recently introduced industry solutions to DRAM read disturbance.

Mathematical analysis & extensive simulations show that PRAC:

• provides security as long as no bitflip occurs below 10 activations
• has non-negligible performance (10%) and energy (18%) overheads
• poorly scales for future DRAM chips, leading to significant overheads on performance (49%) and energy (136%)
• allows memory performance attacks to hog significant amount of DRAM throughput (up to 79% throughput loss)

Future work: More research is needed to improve PRAC by

• reducing the overheads due to increased DRAM timing parameters
• solving the exacerbated performance impact as \( N_{RH} \) decreases
• stopping preventive refreshes from being exploited by memory performance attacks
Understanding the Security Benefits and Overheads of Emerging Industry Solutions to DRAM Read Disturbance

Oğuzhan Canpolat
Giray Yağlıkçı  Geraldo Oliveira  Ataberker Olgun
Oğuz Ergin  Onur Mutlu
Understanding the Security Benefits and Overheads of Emerging Industry Solutions to DRAM Read Disturbance

BACKUP SLIDES

Oğuzhan Canpolat
Giray Yağlıkçı Geraldo Oliveira Ataberk Olgun
Oğuz Ergin Onur Mutlu
### Security Analysis: Wave Attack (I)

<table>
<thead>
<tr>
<th></th>
<th>101010101010101010101010</th>
<th>101010101010101010101010</th>
<th>101010101010101010101010</th>
<th>101010101010101010101010</th>
<th>101010101010101010101010</th>
<th>101010101010101010101010</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>101010101010101010101010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Secure configuration is not trivial
Wave Attack on Different Mechanisms

Wave attack buildup

Graphene - Hydra

![Diagram showing wave attack on different mechanisms]