# DRFM and the Art of Rowhammer Sampling

Salman Qazi sqazi@google.com Google

# ABSTRACT

This paper focuses on Memory-Controller (MC) side Rowhammer mitigation for DDR5. MC-side mitigation consists of two parts: First, a tracker to identify the aggressor rows. We focus on randomized aggressor-row tracking, as such tracking can be performed with negligible storage overhead. Second, a command to let the MC inform the DRAM chip to perform victim-refresh for the specified aggressor row. To facilitate this, prior works assumed a per-bank Nearby Row Refresh (NRR) command. However, the JEDEC DDR5 standard does not support NRR. It supports Directed Refresh Management (DRFM). DRFM can simultaneously perform mitigations for one row each in 8 (DRFMsb) or 32 (DRFMab) banks. As DRFM stalls 8-32 banks, it incurs high overheads. For example, at a threshold of 1K, PARA incurs slowdowns of 7.9% with NRR, 23.5% with DRFMsb, and 83% with DRFMab. Can the overheads of DRFM-based methods be reduced with an alternative method of doing probabilistic selection? In this paper, we compare three sampling methods for DRFM-based mitigations: (1) PARA, which selects aggressor rows with an independent-and-identical distribution (IID) probability, (2) MINT, which selects aggressor row using a URAND sampling of one element within a fixed-sized window, and (3) A new method, MIST, which performs harmonic sampling, whereby rows are continuously selected with decreasing probability while ensuring that all rows within an undefined interval still have a uniform chance of getting mitigated. We show that MIST is better suited for DRFM interface, as it can improve the mitigation-parallelism offered by DRFM - when a neighboring bank issues a DRFM, MIST maximizes the likelihood that all banks have an aggressor row to mitigate, thereby reducing the rate of DRFM. At a threshold of 1K, MIST incurs an average slowdown of only 4.18%.

# **1** INTRODUCTION

Computer systems are only as secure and reliable as the underlying hardware. Rowhammer [7] breaks the isolation between neighboring rows of memory in the same bank. This poses both a reliability and a security challenge. A program with access to one row may inadvertently or maliciously cause changes to data in another row, potentially belonging to a different program. The memory isolation provided by the CPU through virtualization becomes ineffective if an attacker can modify memory outside its address space.

In response to this, the industry initially focused on proprietary in-DRAM Rowhammer mitigations, known as *Targeted Row Refresh (TRR)*. However, soon many of these proprietary mitigations were circumvented by a combination of fuzzing and reverse engineering [2] [4] [3]. An MC-based Rowhammer defense can enable SoC vendors to securely mitigate Rowhammer without relying on proprietary and obscure defenses provided by DRAM vendors.

<sup>6</sup>DRAMSec 2025<sup>7</sup>, June 21, 2025, Tokyo, Japan 2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM https://doi.org/10.1145/nnnnnnnnnn Moinuddin Qureshi moin@gatech.edu Georgia Tech

Rowhammer mitigations operate by refreshing the victim rows of a given aggressor row. However, the manufacturer of the Memory Controller may not know the row layout within DRAM chips used with their Memory Controller. To counter this, the MC-based Rowhammer mitigations often assume the existence of a *Neighboring Row Refresh (NRR)* command supported by the DRAM chip. The memory controller can issue the NRR command to an individual bank, identifying a Rowhammer aggressor row. The DRAM chip will then refresh the potential victim rows of this aggressor.

However, JEDEC took a different approach to supporting MCbased mitigations in DDR5. The DDR5 *Directed Refresh Management* (*DRFM*) implementation splits the operation into two parts: designating the aggressor rows and carrying out the mitigation. The designation of the aggressor rows is done on a per-bank basis. However, mitigation occurs in batches. Two different granularities are provided: *DRFMsb* will refresh neighboring rows for 8 banks, while *DRFMab* will refresh neighboring rows for 32 banks. During DRFM the corresponding banks are busy and cannot service any requests. DRFM can cause significant slowdowns compared to NRR. Figure 1 shows the average slowdown of PARA with DRFMab, DRFMsb, and NRR, as the threshold is reduced from 4K to 1K. At the threshold of 1K, NRR has a slowdown of 7.9%, whereas DRFMsb incurs 23% and DRFMab incurs 83%, almost 3x-8x higher than NRR.



Figure 1: Slowdown of PARA when implemented with DRAMab, DRFMsb, and NRR for TRH of 1K/2K/4K

Can the slowdown of DRFM-based mitigations be reduced by using a probabilistic selection method different from PARA, while still ensuring that all rows have an equal chance of being mitigated? In this paper, we compare three different probabilistic selections: PARA, MINT, and a novel sampling method, MIST, which is designed with DRFM in mind. As a bank can receive a DRFM at any time (triggered by another bank), sampling must ensure that a bank has a row to mitigate in case a DRFM arrives from another bank. This ensures that the mitigation parallelism offered by DRFM is optimally used, thereby reducing the rate of DRFM.

Overall, our paper makes the following contributions:

- Compare and contrast different sampling methods for probabilistic Rowhammer tracking, in the context of DRFM
- Propose MIST: a probabilistic Rowhammer mitigation that makes effective use of the DDR5 DRFM interface

'DRAMSec 2025', June 21, 2025, Tokyo, Japan

#### 2 BACKGROUND ON DRFM

Rowhammer mitigation consists of two components. First, a tracker that can identify the aggressor row. Second, a mitigating action that refreshes the victim rows associated with the given aggressor row. Randomized methods of aggressor row detection (such as PARA [7]) incur negligible storage overheads. In this work, we focus on such randomized trackers. If the knowledge about row mapping were public (for example, consecutive row addresses are physically adjacent to each other), then the MC could perform mitigation by simply activating the adjacent victim rows. Unfortunately, DRAM chips use proprietary mappings, and prior research has shown that consecutive row numbers do not always correspond to neighboring rows on real DRAM chips [18].

**NRR:** Previous literature [11] [9] proposes a *Neighboring Row Refresh (NRR)* (sometimes called *Adjacent Row Refresh (ARR)*) interface to allow a memory controller supplier to implement Rowhammer defenses without having the knowledge of DRAM internals. This interface provides a single command that takes an aggressor row number as input. The DRAM chip then refreshes all the rows that may be potential victims of this aggressor row. With NRR, the key assumption is that mitigating an aggressor row stalls only a single bank (corresponding to the row being mitigated).

**DRFM:** DDR5 [5] provides support for MC-side mitigation. It contains two parts. First, sampling the aggressor row into an internal register (called *DRFM Address Register*, or DAR). MC samples the address of the aggressor row into DAR by simply using the new PRE+S command instead of PRE command when precharging the bank after an access to the aggressor row. Second, there is a mitigation command that comes in two flavors. First, *DRFMab* triggers a mitigation for all banks that have a sampled aggressor row. There is a maximum of 32 banks in DDR5. Second, *DRFMsb* triggers mitigation for a single bank in each of the 8 bank group.



# Figure 2: Overview of DRFM (a) Sampling address into DAR (b) Mitigation command that can stall 8-32 banks

Figure 2 shows an overview of the DRFM sampling operation and the DRFM mitigation operation. Each bank contains a single sampled register (DAR). When DRFM arrives at a bank, the address in DAR (if valid) is mitigated. If the DAR does not contain a valid address, then an internal address tracked by TRR can be mitigated (however, the MC does not get to know the mitigated address).

**Need for Exploiting RLP:** DRFM offers *Rowhammer-Mitigation Level Parallelism (RLP)* as 8-32 banks could be mitigated with a single command. However, existing schemes, such as PARA, are unable to exploit RLP. The security of PARA depends on sending a mitigation right after the row is sampled. We observe an RLP of approximately 1 with both DRFMab and DRFMsb. We observe that we can reduce the slowdown of DRFM by increasing the RLP.

# **3 IMPACT OF SAMPLING ALGORITHM**

To ensure security, a randomized sampling algorithm must ensure that all rows have an equal probability of getting mitigated. Otherwise, the attacker can exploit the non-uniformity by focusing the attack on positions that are less likely to be mitigated. However, there are multiple ways of doing "uniform" sampling. In this section, we will analyze three different methods of "uniform sampling" in the context of a probabilistic DRFM-based Rowhammer defense. The three implementations lead to differing results in terms of performance, security, and simplicity. As DRFMsb is more efficient than DRFMab, we assume that all algorithms mitigate using DRFMsb.

**PARA (IID Sampling):** On an activation, PARA [7] selects the row for mitigation with probability p (see Figure 3). Thus, PARA performs *Independent and Identically Distributed (IID)* selection. The selection parameter (p) is based on the target  $T_{RH}$  and acceptable failure rate (for all trackers, we target a bank failure rate of 1 per 10K years of continuous attack). For example, tolerating a threshold of 1000 requires p = 1/50. The sampling decisions in PARA are independent, and the past decisions should have no bearing on the future ones. Therefore, the security of PARA is easy to prove. However, the security of PARA relies on doing a mitigation right after the selection (delaying the mitigation can cause additional activations to the selected row and thus reduce security). The coupling of sampling and mitigation in PARA causes poor performance with DRFM. For example, the recommended PARA implementation (of sending a mitigation right after row selection) has an RLP of 1.



Figure 3: Overview of Randomized Trackers: PARA and MINT. PARA performs IID selection with probability *p*. MINT performs URAND selection of one entry from Window (*W*).

**MINT (URAND Sampling):** MINT [14] is a probabilistic tracker for uniform selection (see Figure 3). MINT operates on a window size of *W*, where *W* is the number of activations between consecutive mitigations. Before starting a new window, MINT performs a *Uniform Random (URAND)* selection of a number between 1 and *W* and mitigates whichever row is activated at that position in the window. The properties of MINT differ significantly from those of PARA. For example, if only a single row is repeatedly activated throughout the window, it is guaranteed to be selected. For a threshold of 1000, MINT uses W=50, so the probability of selection is 1/50.

For securely implementing MINT with DRFM, we sample the selected address in DAR at precharge (using the Pre+S command). However, we wait until the end of the window to issue a DRFM (if the DAR is not already mitigated by a DRFM issued by another bank). The decoupling of sampling and issuing of DRFM is important for security. Coupling sampling and mitigation for MINT can create a vulnerability where an attacker can sense the sampling and mitigation (using a timing side channel) and focus the activations on rows that get accessed between sampling and the end of the window (as such activations are guaranteed not to be selected).

**MIST (Harmonic Sampling):** MIST is our novel DRFM-based Rowhammer defense that is designed to maximize the RLP of DRFM. The problem with MINT is that it requires the notion of a fixed-size window for selection. Due to the parallelism of DRFM, a bank can be affected by a DRFM intended for another bank. Therefore, in reality, the window-size between two DRFMs is *dynamic* and the selection must adapt to such dynamic window sizes, while ensuring a maximum limit for the window size. Can we do uniform selection of all items in a window, without knowing the window size? Yes, with the insight of *harmonic sampling* employed by MIST.

Consider a window size of 1. The row is always sampled into DAR. Now, when a second item arrives (window size of 2), we want both, the first and the second item, to survive in DAR with probability 1/2. This can be accomplished by overwriting the DAR with the second item with probability 1/2. Now, a third item arrives (window size of 3). We want all three items (first, second, third) to be selected with probability 1/3. This can be accomplished by overwriting the DAR with the third item with probability 1/3 (by design, the third item is sampled with probability 1/3 and the remaining 2/3 is equally divided into the first and the second item). By induction, when the Nth item arrives (window size of N), we overwrite the DAR value with probability 1/N. When the window size reaches a predefined maximum value (W), and DAR contains a valid value, we issue a DRFM for mitigation. When a DRFM is received (issued by the given bank or a neighboring bank), the address in DAR gets mitigated, and the window size is reset to 0. As rows in a window get selected (and overwrite existing values) with probability 1, 1/2, 1/3, and so on, we call this form of sampling *Harmonic Sampling*. With Harmonic Sampling for a dynamic window size of w, each of the *w* items get mitigated with a uniform probability of 1/w.



MaxWindow=W

Figure 4: Overview of MIST. MIST employs Harmonic Sampling to ensure that (a) all positions have uniform probability of mitigation (b) at any point of a non-zero window, the DAR contains a valid item for mitigation for DRFM (even if the DRFM is issued by the neighboring bank).

Figure 4 shows the overview of MIST. While MIST handles a dynamic window size, there is a maximum limit to the size of the window (W). Note that while the sampling probability of each position is harmonically decreasing, earlier rows can be overwritten by the later rows; therefore, they can also get evicted. When both sampling and replacement are taken into account, all rows have a uniform probability of being present in the DAR when the sampling window is terminated (by DRFM, issued by the given bank or neighboring bank). As the window size is reset on the DRFM, all banks that share the DRFM restart their sampling, and the bank with the most activation will trigger the DRFM for all these banks.

**Security of MIST:** By construction, it is guaranteed that ACTs going to each bank are sampled by MIST with a probability of at least 1/W. This stems from the fact that either a window will be complete (i.e. have length W) or it will terminate early. If the window is complete, then the probability of mitigation of each element in the window will be 1/W. If the window ends early (because another bank reached the end of its window first), then the number of elements in the window will be fewer than W, resulting in a probability of each element being mitigated exceeding 1/W. We use techniques similar to those used in MINT [14] to derive a sampling probability p and hence the window size W = 1/p. For a threshold of 1000, MIST uses max W=50, so the probability of selection for all rows remains  $\geq 1/50$ .

The window closure for all batched banks takes place when any bank in the batch reaches the end of the window. This approach does not reveal any information about what was actually sampled. Note that no extra ACTs are required for side-channel mitigation: we simply use PRE+S for those ACTs that are selected to be sampled.

**Note on PRNG:** Weaknesses in PRNG can have a significant impact on the security of all probabilistic Rowhammer defenses [12]. Presence of side channels, for example, in the case of the PARA implementation, has a greater risk. Some PRNG implementations, such as those based on LFSRs, in theory, permit an attacker to predict future outputs based on past outputs. We recommend using a cryptographically secure PRNG implementation if feasible. If an LFSR must be used, it must at least be wide enough to make brute forcing of the internal state infeasible and any side channels revealing past sampling decisions must be avoided.

## 4 EVALUATION METHODOLOGY

We use DRAMSim3 [10], a detailed memory system simulator, to model the DDR5 configuration. Table 1 shows the configuration for our baseline system. We use the Minimalist Open Page (MOP) [6] policy as it performs the best for our configuration. We assume that the time taken by the NRR command is the same as DRFMsb.

Out-of-Order Cores	8 cores at 4GHz, 4-wide		
ROB size	256		
Last Level Cache (Shared)	8MB, 16-Way, 64B lines, LRU		
Memory size	32GB – DDR5		
Memory bus speed	3 GHz (6000 MT/s)		
Channels	1 (one 32GB DIMM)		
Banks x Ranks x Sub-Channels x Rows	32×1×2×128K		
tRCD - tPRE - tRC	14ns – 14ns – 46ns		
tDRFMsb, tDRFMab	240 ns and 280 ns		
Page Closure	Open Page Policy		
Address Mapping	MOP4 [6]		

**Table 1: Baseline System Configuration** 

# 4.1 Workload Characterization

We use 12 benchmarks from SPEC2017 [1] with an MPKI of at least 1, 6 from Graph-Analytics Platform (GAP) [15] and 4 from STREAM. We use representative sections of the traces. We run the applications in 8-core rate-mode and continue executing until each core completes 250 million instructions. We use weighted speedup as a performance metric.

'DRAMSec 2025', June 21, 2025, Tokyo, Japan



Figure 5: Performance Impact of PARA, MINT, and MIST with DRFMsb at threshold of 1K. PARA and MINT incurs an average slowdown of 23% and 31% respectively, whereas, MIST incurs an average slowdown of 4.18%.



Figure 6: Rowhammer-Mitigation Level Parallelism (RLP) of PARA, MINT, and MIST. Both PARA and MINT have an average RLP of 1, whereas, MIST has an average RLP of approximately 8 (which is similar to the ideal value for DRFMsb).

# 5 RESULTS AND ANALYSIS

# 5.1 Impact on Performance

Figure 5 shows the slowdown for PARA, MINT, and MIST when implemented with DRFMsb for a threshold of 1K. The bar labeled *Gmean* shows the Geometric Mean over all the 22 workloads. PARA and MINT incur an average slowdown of 23% and 31% respectively, whereas, MIST incurs an average slowdown of only 4.18%. This occurs because MIST is better able to exploit the parallelism offered by DRFM. When a DRFM arrives, all the banks in the batch have something to mitigate, which resets their windows and delays the further onset of DRFM.

#### 5.2 Impact on RLP

Figure 6 shows the RLP for PARA, MINT, and MIST when implemented with DRFMsb. We define RLP as the average number of banks that perform a mitigation when the batch receives a DRFM. For PARA and MINT, as they do not synchronize their mitigation with other banks, the episode of one bank triggering a DRFM when other banks do not have anything to mitigate is quite high. Their average RLP is approximately 1.

MIST is designed with the awareness that a DRFM may arrive at anytime due to the need for mitigation at a neighboring bank. So, each bank tries to retain a valid entry in the DAR, such that when a DRFM arrives, the bank is able to mitigate the entry (and reset the window). The average RLP of MIST is approximately 8, which is close to the idealized value for a DRFMsb. We note that for cam4, the RLP of MIST is lower than 8. This occurs because the memory activations are non uniformly spread across the banks. When a bank reaches the maximum activation for MIST (50 for a bank for the threshold of 1K), some of the banks do not receive even a single activation and have nothing to mitigate.

# 5.3 Comparison to PRAC

To mitigate Rowhammer in a principled manner within the DRAM chips, JEDEC introduced *Per-Row Activation Counter (PRAC)*. PRAC extends the DRAM array such that each row has an activation counter. It also changes the DRAM timings to support the read-modify-write of the activation counter. Unfortunately, the increased memory timings (tRC increases by about 10% and tRP increases by 150%) can cause significant slowdown, regardless of the tolerated threshold. Table 2 compares the slowdown of PRAC and MIST for thresholds of 500 to 4K. We implement PRAC using MOAT [13].

**Table 2: Comparison of MIST with PRAC** 

Threshold	4K	2K	1K	500
MIST	0.96%	2.00%	4.19%	8.77%
PRAC	9.70%	9.70%	9.70%	9.70%

The slowdown of PRAC remains high (on average, 9.70%) across all thresholds (500-4K). PRAC slowdown is not due to mitigations but due to the latency for counter updates. The slowdown with MIST using DRFMsb is only 0.96% at the current threshold of 4K and is still lower than PRAC (8.77% vs. 9.70%) at threshold of 500. Thus, MIST+DRFM can provide an attractive alternative to PRAC. DRFM and the Art of Rowhammer Sampling



Figure 7: Performance Impact of Delayed-PARA (PARA-D), Delayed-MINT (MINT-D), and MIST with DRFMsb. PARA-D incurs an average slowdown of 8.41%, MINT-D of 4.23%, and MIST incurs 4.18%.



Figure 8: Rowhammer-Mitigation Level Parallelism (RLP) of PARA-D, MINT-D, and MIST. PARA-D has an RLP of 3.24, MINT-D has an RLP of 7.45 and MIST has an RLP of 7.97 (close to the ideal value of 8 for DRFMsb).

# 5.4 Comparison with Delayed Mitigation

Each bank is provisioned with only a single DAR register per bank. A recent work [17] explores another way to increase the RLP for PARA and MINT is to delay mitigation until there is a conflict for the DAR (if so, issue a DRFM and then sample the conflicting row). In this section, we consider such delayed mitigation and call such designs PARA-D and MINT-D, respectively. For design details of PARA-D and MINT-D, we refer the readers to DREAM [17].



Figure 9: Overview of delayed PARA/MINT. It performs a *Tracker-Check (Chk)* before ACT, and if ACT will be sampled and the DAR is full, we issue a DRFM before the ACT.

**Design:** Figure 9 shows the overview of Delayed PARA/MINT. To enable delayed mitigation, we first do a tracker check (Chk) before doing the activation to determine if the upcoming ACT will get sampled into the DAR. Based on the Chk, there are three scenarios: **①** The DAR is empty, and the tracker decides to sample the ACT, we perform the activation, and when we need to do row closure, we use Pre+S to sample the row into DAR (DRFM is not issued). **②**  The tracker decides not to sample the activation. In this case, the subsequent row closure happens with regular precharge. ③ The tracker decides to sample the activation, however, the DAR already contains a valid entry. We first issue a DRFM command (which clears the DAR), then perform the activation, and at row closure, we use Pre+S to write to DAR.

**Impact on Slowdown:** The delayed mitigation for PARA-D and MINT-D allows other banks the time to write to their own DAR. Improved RLP reduces the need for frequent DRFM. Figure 7 shows the slowdown of PARA-D, MINT-D and MIST. PARA-D incurs an average slowdown of 8.41%, MINT-D of 4.23%, and MIST of 4.18%.

**Impact on RLP:** Figure 8 shows the RLP for PARA-D, MINT-D and MIST. On average, PARA-D has an RLP of 3.24, MINT-D of 7.45, and MIST of 7.97. Thus, delayed mitigation can help both PARA and MINT, however, they are still less effective than MIST.

**Disadvantage of Delaying Mitigation:** Delaying the DRFM until the next time another row needs to be sampled can affect the tolerated threshold, as the attacker could cause extra activations during this period. Delaying the mitigation for PARA until next sampling increases the tolerated threshold by about 15% (see Appendix-A of [17]). A design may need additional complexity (or more frequent mitigation) to account for the increased threshold. With MIST, we guarantee that the mitigation is issued within the maximum window size, thus there is no delay past the window and no impact on the tolerated threshold. MIST has lower performance and complexity compared to alternative sampling methods.

# 6 CONCLUSION

For DDR5, implementing MC-side Rowhammer mitigation requires DRFM. As DRFM stalls an increased number of banks (8-32) conventional methods of sampling, such as PARA, can cause significant slowdowns. In this paper, we compare the performance impact of three sampling techniques: PARA, MINT, and MIST. MIST is our novel algorithm that uses harmonic sampling to exploit the increased RLP with DRFM. We show that MIST can tolerate a 1K threshold while incurring only 4.18% slowdown and has lower overhead than PRAC even at thresholds as low as 500.

We note that a recent patent [19] from Qualcomm describes a scheme in which a DRFM-like command is issued when a PRNG output matches a command index, similar to MINT with shifting window sizes. However, the patent does not describe the security and performance analysis of such a design.

# ACKNOWLEDGEMENT

We thank Hritvik Taneja for re-evaluating MIST within the same setup as DREAM [17]. We thank the anonymous reviewers of DRAMSec for their suggestions and feedback.

# REFERENCES

- [1] [n. d.]. SPEC CPU2017 Benchmark Suite. http://www.spec.org/cpu2017/
- [2] Pietro Frigo, Emanuele Vannacc, Hasan Hassan, Victor Van Der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. [n. d.]. TRRespass: Exploiting the many sides of target row refresh. In *IEEE-SP*.
- [3] Hasan Hassan et al. 2021. Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications. In *MICRO*. 1198–1213.
- [4] Patrick Jattke, Victor Van Der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. 2022. Blacksmith: Scalable rowhammering in the frequency domain. In 2022 IEEE Symposium on Security and Privacy (SP). IEEE, 716–734.
- [5] JEDEC. Sept 2022. DDR5 SDRAM JEDEC Standard (JESD79-5B).
- [6] Dimitris Kaseridis, Jeffrey Stuecheli, and Lizy Kurian John. 2011. Minimalist open-page: A DRAM page-mode scheduling policy for the many-core era. In MICRO. 24–35.
- [7] Yoongu Kim et al. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. ISCA 42, 3 (2014), 361–372.
- [8] Andreas Kogler, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. 2022. Half-Double: Hammering from the next row over. In USENIX Security Symposium.
- [9] Eojin Lee, Ingab Kang, Sukhan Lee, G Edward Suh, and Jung Ho Ahn. 2019. TWiCe: preventing row-hammering by exploiting time window counters. In ISCA. 385–396.
- [10] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce L. Jacob. 2020. DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator. *IEEE Comput. Archit. Lett.* 19, 2 (2020), 110–113.
- [11] Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, and Jae W. Lee. 2020. Graphene: Strong yet Lightweight Row Hammer Protection. In MICRO.
- [12] Salman Qazi and Daniel Moghimi. 2024. SoothSayer: Bypassing DSAC Mitigation by Predicting Counter Replacement. DRAMSec4 (2024).
- [13] Moinuddin Qureshi and Salman Qazi. 2025. MOAT: Securely Mitigating Rowhammer with Per-Row Activation Counters. In ASPLOS. 698–714.
- [14] Moinuddin Qureshi, Salman Qazi, and Aamer Jaleel. 2024. MINT: Securely Mitigating Rowhammer with a Minimalist In-DRAM Tracker. In MICRO. IEEE.
- [15] K. Asanovic S. Beamer and D. Patterson. 2015. The GAP benchmark suite. In arXiv preprint arXiv:1508.03619.
- [16] Stefan Saroiu and Alec Wolman. 2022. How to configure row-sampling-based rowhammer defenses. In DRAMSec.
- [17] Hritvik Taneja and Moinuddin Qureshi. 2025. DREAM: Enabling Low-Overhead Rowhammer Mitigation via Directed Refresh Management. In ISCA. IEEE.
- [18] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Defeating software mitigations against rowhammer: a surgical precision hammer. In *RAID*-2018. 47–66.
- [19] Victor van der Veen, Mossadiq Saifuddin, Pankaj Deshmukh, Behnam Dashtipour, and David Hartley. [n. d.]. DRAM WITH QUICK RANDOM ROW REFRESH FOR ROWHAMMER MITIGATION.

# APPENDIX-A: SECURITY MODEL FOR MIST

For a given maximum window size of W, the security model of MIST is similar to MINT with a window of W. This is because, for the optimal attack pattern for a bank under MIST, a DRFM is issued no earlier than W activations. Fewer activations in the window only increase the probability of detection. So, an attacker would issue W activations to a bank, and only then a DRFM would occur. In such a case, all the rows activated within the window are selected with probability p=1/W and MIST degenerates into MINT. Note that MIST is simply a performance optimization, even if the security for worst-case pattern is the same as MINT.

The security model for MINT/MIST can be derived using the following assertions: (1) The selection decision for the current window is not influenced by which addresses are accessed in the past/future windows. (2) The likelihood of a row getting selected is invariant to the order of the items appearing in the window as all items are selected with equal probability (3) To maintain a low likelihood of getting selected, the attacker must avoid having multiple activations to the same attacked row within the window, otherwise the selection probability increases proportionately to the number of copies, significantly weakening the attack. Thus, for the best attack, the attacker is restricted to having a single activation on the given attack row within the window. (4) There are only a limited number of windows within tREFW, and any window in which the attacked row is not activated reduces the time available for the attacker, so for the best attack, the attacker must activate the attacked row exactly once per window. Thus, the optimal attack pattern for a given attack row is exactly one activation of the row in each window.

Let there be N ((tREFW-8192\*tRFC)/(W\*tRC+tDRFM)) windows in tREFW. Let T be the Rowhammer threshold. We use the Saroiu-Wolman method [16] to compute the probability of failure ( $P_f$ ) for a single row, which denotes T consecutive unmitigated activations within N activations. As the window contains W slots, the attacker can concurrently attack W rows, so the probability of failure  $P_{fW}$ on W rows is the product of W and  $P_f$ . We select the double-sided threshold (TRHD) such that the Mean-Time-To-Failure (MTTF) for a bank is 30K years. For our target MTTF, the TRHD of MINT/MIST is approximately 20\*W. Thus, for W=50, we get TRHD=1K.

We note that, since DRFM uses the *Bounded Refresh (BR)* mode, which is designed to mitigate transitive attacks, such as Half Double [8] (by refreshing distant rows with a small probability), we do not consider such attacks in the MIST security model.

#### APPENDIX-B: HW IMPLEMENTATION OF MIST

MINT requires a selection probability that reduces (harmonically) as the number of items in the window increases. Like PARA and MINT, MIST requires a pseudo-random number generator (PRNG). As we are interested in an MC-based implementation (where the logic area is less constrained than the in-DRAM setting), we use a cipher to generate the random numbers. Consider a 64-bit PRINCE cipher, initialized with a random key and the cycle count being provided as input. The output of the cipher is the random value. To select with probability 1/K, we compare if the generated N-bit PRNG value is smaller than  $(2^N)/K$ . If so, the item is selected.

DRFM and the Art of Rowhammer Sampling

To avoid integer division at runtime, we can initialize a table that contains the division results for various values of K (K=2,3,4 up-to W). Then, the decision becomes comparing the PRNG value to the corresponding value of the table. For W=100, we need a 100-entry table, where each entry has a given number of bits (N).

Ideally, we want to have as few bits in the table as possible while still ensuring security. Due to the imprecision of the entry size, not all positions may be mitigated with the same probability p=1/W.

Let  $M_i$  be the *Mitigation Probability of position* i within the window. This value is a product of the *Selection Probability* ( $P_i$ ) and the *Survival Probability* ( $S_i$ ). The survival probability denotes the probability that no other item is selected in the remaining window.

$$M_i = P_i \cdot S_i \tag{1}$$

$$S_i = \prod_{K=i+1}^{W} (1 - P_K)$$
 (2)

Now, due to hardware imprecision, each item, instead of being selected with probability  $P_i$ , is selected with probability  $\hat{P_i}$ . Then, we can compute  $\hat{S_i}$  using  $\hat{P_i}$ . Similarly, we can compute the resulting mitigation probability for each position  $\hat{M_i}$  as  $\hat{P_i} \cdot \hat{S_i}$ .

Ideally,  $M_i = 1/W$  for all positions, if hardware implementation was precise. However, due to imprecision, some positions may have lower  $\hat{M_i}$ . We evaluated  $\hat{M_i}$  for all positions (i=1 to W) and found the minimum value for  $\hat{M_i}$  (as that would be the optimal position for the attacker to place the attack line).

For our implementation, we select the N-bit value in our table such that the worst-case  $\hat{M}_i$  is within 1% (relative value) of the ideal value of  $M_i$ . Thus, the overall impact on TRHD remains within 1% (for example, the hardware approximations would result in a TRHD of 1.01K when the target is 1K). For obtaining such a bound, we would need 12/13/14 bits per entry for W=25/50/100 (TRHD of 500/1K/2K). The total SRAM cost of the table would be 37 bytes, 87 bytes, and 175 bytes, respectively for TRHD of 500/1K/2K. As the table is shared by all banks and ranks within the channel, the SRAM overhead of the proposed implementation is negligibly small.

#### **APPENDIX-C: SECURITY REQUIREMENTS**

Our paper compares three probabilistic schemes: PARA, MINT, and MIST. All of these schemes are designed to provide a uniform probability of mitigation. However, these schemes have different security requirements and associated performance overheads.

The security requirement for PARA depends on doing mitigation right after selection. Any delay can allow the attacker to cause additional activations in the time between selection and mitigation. For performance, we want to delay the mitigation (DRFM), whereas for security, we want to send the mitigation quickly.

MINT was developed in an in-DRAM setting, where mitigation occurs at the end of the window (during REF or using RFM). For an MC-side implementation, it is essential that the mitigation is done such that the selection decision remains unknown (for example, sending mitigation right after selection will allow the attacker to use the timing channel to know which row was selected and attack the row during the remaining activations within the window, which are guaranteed not to be selected). So, for MINT security, mitigation (DRFM) must be delayed until the end of the window, whereas sampling can still be performed when activation occurs.

As MIST performs continuous sampling (with decaying probability), sending a DRFM as soon as sampling occurs will cause unacceptable performance overhead. For example, the first row is accessed, it is sampled, and we issue a DRFM. Thus, the performance of MIST relies on NOT sending a DRFM until the end of the window (however, if another bank issues a DRFM, then having something already sampled that can be mitigated). Thus, MIST maintains the same security as MINT; however, for performance, sampling and mitigation are decoupled, and sampling is performed more frequently than mitigation.