Fundamentally Understanding and Solving RowHammer

Onur Mutlu
onur.mutlu@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

Ataberk Olgun
ataberk.olgun@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

A. Giray Yağlıkcı
giray.yaglikci@safari.ethz.ch
ETH Zürich
Zürich, Switzerland

ABSTRACT

We provide an overview of recent developments and future directions in the RowHammer vulnerability that plagues modern DRAM (Dynamic Random Memory Access) chips, which are used in almost all computing systems as main memory.

RowHammer is the phenomenon in which repeatedly accessing a row in a real DRAM chip causes bitflips (i.e., data corruption) in physically nearby rows. This phenomenon leads to a serious and widespread system security vulnerability, as many works since the original RowHammer paper in 2014 have shown. Recent analysis of the RowHammer phenomenon reveals that the problem is getting much worse as DRAM technology scaling continues: newer DRAM chips are fundamentally more vulnerable to RowHammer at the device and circuit levels. Deeper analysis of RowHammer shows that there are many dimensions to the problem as the vulnerability is sensitive to many variables, including environmental conditions (temperature & voltage), process variation, stored data patterns, as well as memory access patterns and memory control policies. As such, it has proven difficult to devise fully-secure and very efficient (i.e., low-overhead in performance, energy, area) protection mechanisms against RowHammer and attempts made by DRAM manufacturers have been shown to lack security guarantees.

After reviewing various recent developments in exploiting, understanding, and mitigating RowHammer, we discuss future directions that we believe are critical for solving the RowHammer problem. We argue for two major directions to amplify research and development efforts in: 1) building a much deeper understanding of the problem and its many dimensions, in both cutting-edge DRAM chips and computing systems deployed in the field, and 2) the design and development of extremely efficient and fully-secure solutions via system-memory cooperation.

CCS CONCEPTS

• **Hardware → Dynamic memory**; **Hardware reliability**; • **Security and privacy → Systems security**.

KEYWORDS

DRAM, Security, Vulnerability, Technology Scaling, Reliability, Safety, Errors, Memory Systems, Fault Attacks, RowHammer

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3568350

ACM Reference Format:
Onur Mutlu, Ataberk Olgun, and A. Giray Yağlıkcı. 2023. Fundamentally Understanding and Solving RowHammer. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3566097.3568350

1 INTRODUCTION

DRAM is the dominant technology used for main memory in almost all computing systems due to its low latency and low cost per bit. Modern DRAM chips suffer from a vulnerability commonly known as RowHammer [72, 95, 96, 98]. RowHammer is caused by repeatedly accessing (i.e., hammering) one or more (aggressor) memory rows. Hammering a row creates electromagnetic interference between the aggressor row and its physically-neighboring (victim) rows. Due to this interference, cells in victim rows lose the ability to correctly retain their data, which leads to data corruption (i.e., bitflips). These bitflips are repeatable: if hammering an aggressor row causes a particular cell to experience a bitflip, doing so again will lead to the same bitflip with high probability [72].

Unfortunately, DRAM becomes increasingly more susceptible to RowHammer bitflips as its storage density increases (i.e., DRAM cell size and cell-to-cell spacing reduce). Our recent work [70] across 1580 real DRAM chips of six different types shows that, in the last decade, the minimum number of aggressor row activations needed to cause a RowHammer bitflip (i.e., the *RowHammer threshold*) has reduced by more than 10x and the number of bitflips caused by the same number of row activations has increased by 500x. A recent work [88] shows that commodity workloads on state-of-the-art servers might already activate individual DRAM rows at rates exceeding the RowHammer threshold.

On the one hand, such RowHammer bitflips can lead to system reliability and safety problems, when caused by non-malicious applications. On the other hand, malicious applications can be written to induce RowHammer bitflips in a targeted manner, so as to specifically degrade system security, privacy, safety and availability. For example, by carefully selecting rows to hammer, an attacker can induce bitflips in sensitive data stored in DRAM. Many prior works, some of which are reviewed in [98], show that RowHammer can be exploited in many ways to compromise system security (integrity, confidentiality, availability) in real systems, since it breaks *memory isolation* on top of which modern system security principles are built. As such, RowHammer greatly threatens many aspects of computing system robustness (which include system reliability, safety, security, privacy, and availability) in a widespread and profound manner due to the prevalent usage of DRAM in modern computing systems.

RowHammer-like disturbance issues have also been shown to be present in emerging NVM (non-volatile memory) technologies [2, 39, 68, 83, 101]. For these technologies to be viable, dependable, and secure, anticipation and solution of such issues are critical.

In this paper, we provide an overview of the state-of-the-art research and development focusing on the RowHammer vulnerability. To this end, we first provide a very brief review of RowHammer research until circa 2020 (Section 2), building on a recent overview paper [98] that in more detail covers much of the RowHammer developments until its publication. Then, we describe two major developments in 2020 (Section 3), TRRespass [38] and Revisiting RowHammer [70], which experimentally show that RowHammer is an open and increasingly worsening problem, which motivated a large body of follow-on work in both industry and academia. Afterwards, we provide a broad overview of other recent developments in RowHammer (Section 4). We consider three major types of RowHammer developments: 1) works directed toward exploiting RowHammer, 2) works that aim to understand and model RowHammer, and 3) works that propose techniques to mitigate/solve the RowHammer problem. Finally, we discuss future directions in RowHammer research (Section 5). We argue for two major broad directions to focus more in future research and development efforts: 1) building a much deeper understanding of the problem and its many dimensions/sensitivities, in both cutting-edge DRAM chips and computing systems deployed in the field, and 2) the design and development of extremely efficient and fully-secure architectural solutions, which we believe can be achieved via much better system-memory cooperation (as discussed in [94, 99]).

2 A BRIEF OVERVIEW OF ROWHAMMER UNTIL 2020

Since its first public introduction and scientific analysis in 2014 [72], RowHammer has led to significant follow-on work in both academia and industry, spanning multiple different communities, including computer architecture, security, dependability, circuits, devices, and systems. We briefly review works that appeared within the timeframe 2014-2019. A more comprehensive overview of such works, as well as others in popular technical media, can be found in [98], and we refer the reader to that overview for more detail.

The ISCA 2014 work that introduced RowHammer [72] demonstrated that more than 80% of the real commodity DDR3 DRAM modules tested, from all three major DRAM manufacturers, are vulnerable to RowHammer. That is, real bitflips are possible to induce using real user-level programs [125] on commonly-used CPU-based systems. The work argued that RowHammer is a DRAM technology scaling problem and since device- and circuit-level solutions are difficult and costly, RowHammer should be solved via system-memory cooperation [72, 94]. This work also suggested that one can exploit RowHammer bitflips to construct various types of *disturbance attacks* that inject errors into other programs, crash the system, or hijack control of the system.

Many future works building on Kim et al. [72] did exactly that, i.e., they developed various types of attacks that exploit RowHammer on real computing systems. These works include techniques that compromise system integrity as well as confidentiality on various types of systems, including mobile and server systems [1, 10, 12, 13, 16, 18, 20, 21, 26, 35, 37, 42, 43, 50, 53, 59, 77, 84, 95, 96, 113, 114, 117, 121, 134, 135, 141, 142, 145, 146, 151, 160]. A more detailed overview of these works can be found in Section III-A of [98].

Multiple works at the device and circuit levels aimed to develop a low-level understanding of the causes and effects of RowHammer via low-level modeling and simulation of devices and circuits, and in limited cases via experiments on DDR3 DRAM chips. These works include [109, 110, 123, 156, 159]. A more recent work in 2021 [147] complements this understanding with some newer device level observations. Similarly, RowHammer-like behavior has been analyzed in NVM (non-volatile memory) devices via device and circuit level simulation studies [2, 39, 68, 83, 101]. We refer the reader to Sections III-C and III-H of [98] for a more detailed overview of such device/circuit level works that aim to develop a better low-level understanding of the causes and effects of RowHammer.

The original RowHammer paper in ISCA 2014 [72] proposed seven different solution directions to RowHammer, several of which were later implemented in variations in memory controllers and DRAM chips. Building on these, many other academic and industrial works in the 2014-2019 timeframe proposed various solutions to RowHammer in both hardware and software. These works include [4–9, 18, 19, 36, 40, 41, 49, 51, 52, 69, 76, 78, 79, 82, 102, 135, 139, 140, 146, 150]. A detailed overview of these solutions can be found in Section III-B of [98]. The BlockHammer paper in HPCA 2021 [155] provides a more up-to-date overview of major solutions proposed until that time, with a detailed analysis of 14 solutions across four different desirable properties and a rigorous evaluation of six state-of-the-art solutions, along with an open-source release of their source codes [127].

2.1 RowHammer Mitigations in Industry

After the public introduction of RowHammer in 2014, both system and DRAM manufacturers took action to mitigate the problem in the field and in future DRAM chips. As described in the original RowHammer work [72], the solutions that can be deployed in the field are limited due to the limited programmability support provided by modern memory controllers. As such, a major solution deployed in the field has been to increase the refresh rate of DRAM, as described in a security release by Apple [4]. Unfortunately, increasing the refresh rate is not an effective or desirable solution [72] due to its high performance and energy overheads [85].

On the system side, memory controller manufacturers like Intel introduced mechanisms (e.g., [89], inspired by *PARA, probabilistic adjacent row activation* [72]), broadly called *pTRR* (pseudo Target Row Refresh) [38, 63], to mitigate RowHammer in future systems. Unfortunately, such memory controller based victim row refresh mechanisms are not aware of physical adjacency of aggressor and victim rows in a DRAM chip and, as such, they might not provide complete protection against RowHammer.

On the DRAM side, DRAM manufacturers introduced TRR (target row refresh) [38] mechanisms and claimed that their new DDR4 chips are RowHammer-free [38, 47, 81, 92] with the protection provided by these mechanisms. TRR is an umbrella term used for mechanisms that refresh target rows which are somehow determined to be accessed frequently. Unfortunately, DRAM manufacturers did not (and still do not) describe how their implementations securely prevent RowHammer or reveal how their implementations work.

As such, circa 2019-2020, it was unclear whether or not RowHammer bitflips were possible in real DDR4 DRAM chips. As we will see next, two major works in 2020 showed that they were.

3 MAJOR DEVELOPMENTS IN 2020

We first describe two major works, *TRRespass* [38] and *Revisiting RowHammer* [70], which clearly demonstrate that RowHammer is an open and worsening problem, state-of-the-art DRAM chips are vulnerable, and proposed mitigations are not scalable/effective into the future.

TRRespass [38] is the first work to show that TRR-protected DDR4 DRAM chips that are advertised as Rowhammer-free are actually vulnerable to RowHammer in the field. This work partially reverse engineers the TRR and pTRR mechanisms employed in modern DRAM chips and memory controllers. To overcome such protection mechanisms, TRRespass introduces the *many-sided RowHammer attack*, whose key idea is to hammer *many* (i.e., more than two) rows to bypass TRR mitigations, e.g., by overflowing proprietary TRR tables that detect aggressor rows. Using this many-sided RowHammer attack, the work demonstrated bitflips in real DDR4 DRAM chips as well as LPDDR4(X) DRAM chips and showed that RowHammer attacks are possible on systems that employ such chips. As such, it was clear that the solutions implemented in industry were not secure. TRRespass also argued that *security by obscurity*, as employed by DRAM manufacturers, is not a good solution approach. Later follow-on work, called Uncovering TRR (U-TRR) [47], showed in 2021 that one can almost completely reverse engineer the entire TRR mechanism employed in any DRAM chip, by using an FPGA-based DRAM testing infrastructure (i.e., SoftMC [48, 126] and DRAM Bender [103]) and a methodology that uses retention errors as side channels to discover when the DRAM-internal TRR mechanism refreshes a victim row. U-TRR demonstrates that, by doing so, one can craft specialized hammering/access patterns that essentially induce large numbers of bitflips on any examined chip.

Revisiting RowHammer [70] takes a device/circuit-level approach to understand the scaling properties of RowHammer by measuring the fundamental vulnerability (i.e., with TRR mechanisms turned off) of three different types of DRAM chips across at least two different generations. This work tested 1580 DRAM chips and experimentally demonstrated that RowHammer is *indisputably getting worse* in newer generation DRAM chips: when hammered, newer DRAM chips experience the first bitflip much earlier (e.g., some after only 4800 double-sided hammers) and they experience much higher numbers of bitflips than older DRAM chips (as demonstrated in Figure 1[1]). In other words, the number of activations to induce a RowHammer bitflip (i.e., the *RowHammer threshold*) reduced from 139K single-sided in 2014 [72] to 4.8K double-sided in 2020 [70]. Revisiting RowHammer also showed that if the scaling trend continues as such, all known solutions at the time would either not work in future DRAM chips that are even more vulnerable (e.g., with a RowHammer threshold of 256 or 128) or have prohibitively large performance overheads. To our knowledge, this work is the largest scaling study of RowHammer to date, covering many different types and generations of DRAM chips and its results demonstrate the criticality and difficulty of the RowHammer problem as technology node size shrinks in DRAM manufacturing.

[1]Figure 1 (reproduced from Revisiting RowHammer [70]) illustrates a quantitative summary of both the reduction in double-sided hammer count (x-axis) and the increase in RowHammer bitflip rate (y-axis) with newer DRAM generations (e.g., DDR4-old in blue to DDR4-new in yellow) across three major DRAM manufacturers (A, B, C).

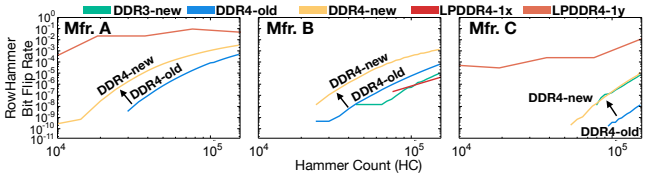

Figure 1: Hammer count (HC) vs. RowHammer bitflip rate across 1580 DRAM chips from five different type-node configurations. Reproduced from [70].

Alarmed by the key experimental results and takeaways demonstrated by these two works [38, 70], industry took action to gather forces to more seriously mitigate the RowHammer problem. To this end, a major RowHammer task group was (re)organized in JEDEC (Joint Electron Device Engineering Council [58]). This task group produced whitepapers describing how to proceed to solve RowHammer [56, 57], which argue for limited system-DRAM cooperation. The mission of this task group continues.

JEDEC and DRAM manufacturers also added a new RFM (Refresh Management) feature to the DDR5 standard [55] to aid the in-DRAM mitigation mechanisms. The key idea of RFM is to provide in-DRAM TRR mechanisms with extra time (as needed) to securely refresh all potential victim rows. RFM requires the memory controller to count the number of activations at DRAM bank granularity and issue an RFM refresh command when the activation count reaches a threshold value (based on the RowHammer threshold and the strength of the in-DRAM TRR mechanism of a DRAM chip), such that a RowHammer defense mechanism implemented inside the DRAM chip can refresh the victim rows. However, because the memory controller tracks row activation counts at DRAM bank granularity, the activation count can frequently reach the threshold even when there is no RowHammer attack in the system. As a result, the memory controller can issue many unnecessary RFM commands, each of which makes a DRAM bank unavailable for hundreds of nanoseconds, causing performance overhead.

4 RECENT ROWHAMMER DEVELOPMENTS

TRRespass [38] and Revisiting RowHammer [70], by demonstrating the importance and openness of the problem, catalyzed the ongoing RowHammer research and development efforts. We briefly review such efforts from 2019-2020 until now.

4.1 Exploiting RowHammer

Many works since 2019-2020 developed new and different ways to exploit the RowHammer vulnerability (e.g., [24, 25, 28, 31, 38, 47, 54, 75, 77, 86, 106, 120, 143, 144, 149, 157, 162, 164, 165]). Among these, we briefly describe RAMBleed [77] and RowHammer-driven fault attacks on neural networks [50, 86, 120, 144, 157] since they demonstrate different uses of RowHammer from taking over a system. We also briefly describe new attacks that build on many-sided hammering, including SMASH [28], Blacksmith [54], and Half-Double [75], as they provide new insights into how vulnerable existing DDR4 DRAM chips are to RowHammer.

RAMBleed [77] shows that RowHammer bitflips can be used to break confidentiality in an existing DRAM-based system: RowHammer bitflips can be used as a side channel to determine the data

values stored in a location a user-level program does not have read access to. This work demonstrates an attack against OpenSSH [105] where RAMBleed is used to leak an encryption key.

Several works [50, 86, 144, 157] demonstrate that targeted RowHammer bitflips can be used to greatly degrade the inference accuracy of a neural network. These works highlight potential safety and security issues that may be caused by RowHammer in systems that rely on neural networks for decision making (e.g., autonomous vehicles). A recent work in 2022 [120] demonstrates that RowHammer bitflips can be used as a side channel to recover the weights stored in a neural network, which breaks confidentiality and violates privacy.

Some works that build on TRRespass demonstrate new RowHammer attacks that are relatively easy to perform on DDR4 DRAM chips. SMASH (Synchronized MAny-Sided Hammering) [28] successfully triggers RowHammer bitflips from JavaScript code by exploiting many-sided hammering and synchronizing access patterns with DRAM refresh operations to bypass TRR mitigations. This work demonstrates an end-to-end JavaScript exploit to fully compromise the Firefox web browser in 15 minutes. Blacksmith [54] uses automated fuzzing in the frequency domain to discover non-uniform access patterns that can bypass TRR mechanisms more effectively than uniform many-sided hammering. By doing so, it generates access patterns that hammer aggressor rows with different phases, frequencies, and amplitudes, finding complex patterns that trigger Rowhammer bitflips on all 40 tested DDR4 modules.

Building on the concept of many-sided RowHammer attacks, Google introduced the Half Double hammering pattern in 2021 [75, 115, 116]. Half Double shows that hammering a "far" neighbor row that is physically one row away from the victim row many times and then hammering the immediately-adjacent "near" neighbor row of the victim row a much smaller number of times leads to bitflips in some DDR4 DRAM chips. This is concerning because, when TRR-based RowHammer mitigations are employed, hammers performed on the *far* neighbor row can lead to refreshes performed by TRR on the *near* neighbor row, which in turn can lead to bitflips in the victim row. This work highlights the intricacies in bitflip mechanisms in modern DRAM chips and alerts that defenses should be carefully designed to work in the presence of such intricacies.

Many other works from 2020-2022 (e.g., [24, 25, 31, 106, 143, 149, 162, 164, 165]) demonstrate various other RowHammer exploits, advancing our understanding of how RowHammer bitflips can be used to degrade system security. We refer the reader to the individual papers for more detail. We believe that it is critical to push the boundaries of system security research by understanding the different ways in which RowHammer bitflips can cause security issues and making RowHammer exploits more powerful, especially in the presence of stronger mitigation mechanisms.

4.2 Understanding RowHammer

Recent works since 2020 (e.g., [2, 24, 39, 68, 70, 101, 106, 107, 147, 153]) study RowHammer from different aspects to develop a better understanding. Among these, we briefly describe two major works: *A Deeper Look into RowHammer* [107] and *RowHammer under Reduced Wordline Voltage* [153] because these two works analyze new aspects of RowHammer by rigorously testing real DRAM chips.

A Deeper Look into RowHammer [107] presents an experimental characterization using 248 DDR4 and 24 DDR3 modern DRAM chips from four major DRAM manufacturers to reveal how the RowHammer vulnerability is affected by three fundamental properties: 1) DRAM chip temperature, 2) aggressor row active time, and 3) victim DRAM cell's physical location. The results clearly indicate that a RowHammer bitflip is more likely to occur 1) in a bounded range of temperature, 2) if the aggressor row is active for longer time, and 3) in certain physical regions of the DRAM module under attack. This work also shows how its findings can be used to improve both RowHammer attacks and RowHammer defenses, highlighting the importance of such deeper understanding from the perspective of both attackers and defenders.

RowHammer under Reduced Wordline Voltage [153] presents an experimental characterization using 272 real DDR4 DRAM chips from three major manufacturers to demonstrate how reducing the wordline voltage affects both RowHammer vulnerability and DRAM operation. The authors show that reducing wordline voltage provides a significant reduction in the number of RowHammer bitflips and increase in the minimum number of aggressor row activations needed to cause a RowHammer bitflip, without significantly affecting reliable DRAM operation. This work highlights how deeply understanding low-level effects on RowHammer can aid in developing more RowHammer-resilient DRAM systems.

Several other works from 2020-2022 (e.g., [24, 44, 106, 108, 147]) present various other simulation and real experiment-based analyses to gain insights into RowHammer and its effects, furthering our fundamental understanding of the RowHammer phenomenon. We refer the reader to the individual papers for more detail. We believe that it is critical to continue to develop a broader and deeper understanding of RowHammer to improve both RowHammer attacks and defenses, on the path to designing systems that can effectively and efficiently guard against RowHammer.

4.3 Mitigating RowHammer

Revisiting RowHammer from ISCA 2020 [70] clearly demonstrated that, as RowHammer continues to worsen in real DRAM chips, it is critical to develop fully-secure, low-overhead, and scalable mitigation techniques. As such, 2020 and later years saw a surge in new RowHammer solutions. We briefly cover major recent ideas and directions in RowHammer mitigation.

Graphene [111] uses the Misra-Gries algorithm [93] for online frequent item counting [17, 33, 34, 67] to track and identify frequently activated rows. Graphene then refreshes the rows neighboring those with activation counts that are close to the RowHammer threshold. This work, while effective, secure, and low-performance-overhead, requires large area overhead when scaled to address the worsening RowHammer vulnerability, as it uses area-expensive content addressable memory for storing metadata [155].

BlockHammer [127, 155] uses counting Bloom Filters to identify frequently-activated rows and throttles accesses to those rows whose activation counts are close to the RowHammer threshold. BlockHammer thus does not require any proprietary information about DRAM chips (e.g., physical row adjacency information) and therefore can be implemented completely and securely in the memory controller. This work experimentally demonstrates that BlockHammer's performance overheads are low (similar to other best

prior mechanisms) when there is no RowHammer attack. When there is a RowHammer attack, BlockHammer improves both system performance and energy consumption by throttling the attacker thread. This work also introduces a metric called RowHammer Likelihood Index (RHLI), and shows that this metric can be accurately computed to identify RowHammer attacks and report them to the system software. As such, BlockHammer provides an example of a system-level approach to mitigate RowHammer attacks' impact.[2] This work deeply analyzes 14 major RowHammer defense mechanisms, with many quantitative and qualitative comparisons, and forms a basis for methodically comparing different RowHammer defenses.[3]

More recently, Self-Managing DRAM (SMD) [46, 130] takes a very different approach to RowHammer mitigation: SMD modifies the DRAM interface such that the DRAM chip can reject an activation command issued by the memory controller and thus gain time to perform internal maintenance operations, such as RowHammer mitigation. SMD observes that RowHammer mitigation is essentially a DRAM maintenance operation that can be best implemented within the DRAM chip based on device-level information available to DRAM manufacturers. SMD essentially provides "breathing room" (i.e., extra time) to the DRAM chip to autonomously implement such maintenance operations completely and securely within the DRAM chip, at low overheads. The results are promising: by carefully scheduling RowHammer mitigation actions at a fine enough granularity across different regions of a DRAM chip, in-DRAM RowHammer mitigation mechanisms inspired by PARA [72] and BlockHammer [155] (i.e., SMD-PRP and SMD-PRP+) can lead to low performance and energy overheads [46].

Randomized Row Swap (RRS) [131] and AQUA [133] propose to relocate aggressor rows whose activation counts are close to the RowHammer threshold. These works adopt Graphene's approach to frequently-activated row detection. When such an aggressor row is detected, RRS swaps it with another randomly chosen row in the same bank, while AQUA moves the aggressor row into a dedicated quarantine region that stores the most frequently accessed rows. By doing so, both RRS and AQUA prevent a DRAM row from being activated enough times to induce a RowHammer bitflip. RRS and AQUA do *not* need to know the physical layout of DRAM rows or make modifications to DRAM chips and thus they are compatible with commodity DRAM chips. On the downside, RRS and AQUA both perform data relocation over the memory bus, which requires off-chip data movement and exacerbates the already pressing data movement overhead in modern systems [14, 15, 27, 64, 94, 97, 137]. Row-relocation based RowHammer defenses can be accelerated with in-DRAM data copy support [23, 45, 104, 122, 129, 136, 148].

Two recent works in 2022 aim to broadly reduce the overheads of RowHammer mitigations. HiRA [154] shows that the performance overheads of refresh-based in-DRAM RowHammer mitigation mechanisms can be reduced by performing refreshes in parallel with accesses and other refreshes [22, 74], and this can be done, to an extent, in real DRAM chips. Hydra [119] shows that the hardware cost of counters used to track row activations can be reduced by storing such counters in DRAM and caching them in small structures on chip. These two works highlight the various overheads of RowHammer mitigation and take a step in improving efficiency of broad classes of mitigation techniques.

Many other works from 2020-2023 (e.g., [3, 11, 29, 30, 32, 38, 44, 47, 60–62, 65, 71, 80, 87, 88, 90, 91, 100, 108, 118, 119, 132, 152, 154, 161, 163, 166]) propose various other RowHammer mitigation mechanisms. We refer the reader to the individual papers for more detail. As RowHammer is greatly worsening with DRAM technology scaling, we strongly believe that there is still a critical need and large potential to solve RowHammer at very low cost and very high efficiency, as we discuss in Section 5.2.

5 FUTURE DIRECTIONS

Aside from continuing the path of prior research, including discovering and developing new RowHammer attacks and access patterns, we believe there are two major directions that are critical for future research to investigate and amplify efforts in.

5.1 Building a Fundamental & Comprehensive Understanding of RowHammer

Even though there are various detailed characterization studies performed to understand various properties of RowHammer [24, 70, 72, 106, 107, 109, 153], there is still a lot we do not know about RowHammer, its properties/sensitivities, and the manifestations of such properties in cutting-edge and future DRAM chips. It is critical to fundamentally understand the various properties of RowHammer under different conditions and access patterns, in order to develop fully-secure and efficient solutions (as we argue for in Section 5.2).

We believe that at least several properties of RowHammer are critical to fundamentally understand going forward: sensitivity to 1) aging of DRAM chips, 2) environmental conditions (e.g., operating temperature, supply voltage), and 3) memory access patterns. There is no detailed characterization study that evaluates if and how the aging of a DRAM chip affects its RowHammer vulnerability. While some environmental conditions and memory access patterns are experimentally demonstrated to significantly affect the RowHammer vulnerability of a DRAM chip [70, 75, 107, 153], further research is needed to develop a more detailed understanding on the relationship between such properties and the RowHammer vulnerability of a DRAM chip. This understanding is necessary and critical to develop given that all existing RowHammer defense mechanisms rely on and future RowHammer defense mechanisms will likely rely on the measured RowHammer vulnerability of DRAM chips (e.g., the RowHammer threshold value) to securely prevent bitflips. Understanding the individual and combined effects of RowHammer's sensitivities to aging of DRAM chips, environmental conditions, and memory access patterns could yield accurate methodologies and infrastructures that can efficiently evaluate the RowHammer vulnerability of a given DRAM chip and facilitate the development of holistic solutions that completely prevent RowHammer across the entire computing system. We believe FPGA-based infrastructures for testing DRAM chips, such as SoftMC [48, 126] and DRAM Bender [103, 128], are critical to enabling such studies, as they have been in the past.

[2] BlockHammer is freely and openly available [127], along with six other RowHammer mitigation mechanisms [72, 78, 111, 138, 140, 158], implemented in Ramulator [73, 124].
[3] The BlockHammer paper [155] identifies four major desirable properties of a RowHammer defense, and demonstrates that BlockHammer is the only mechanism that satisfies all four properties. We refer the reader to Section 9 in [155] for more detail.

We also believe that it is important to more profoundly understand the effects of RowHammer on real systems and real applications, both malicious and non-malicious. To this end, it is critical to do research that pushes the boundaries of generating RowHammer bitflips on many different types of systems, including mobile and server CPUs, GPUs, accelerators, FPGAs, as well as different DRAM and memory types, including HBM and emerging NVM technologies. Such research can not only discover new problems and sensitivities but also pave the way to generalized solutions that are applicable to many systems.

5.2 Designing Extremely Efficient Solutions to RowHammer

As the RowHammer vulnerability worsens with DRAM technology scaling, developing extremely efficient and fully-secure RowHammer solutions becomes increasingly important. Even though many prior works develop various software- and hardware-level RowHammer solutions, these solutions incur non-negligible and increasingly more significant system performance, energy, and hardware area overheads as RowHammer vulnerability worsens.

We believe that developing new low-cost (in terms of performance, energy, area) and provably-secure RowHammer solutions is critical to efficiently preventing RowHammer bitflips going forward. As DRAM continues to scale, RowHammer bitflips can occur at smaller activation counts and thus a benign workload's DRAM row activation rates can approach or even exceed the RowHammer threshold [88, 119, 131, 133, 155]. Thus, a system may experience bitflips or frequently trigger RowHammer defense mechanisms even without a malicious party performing a RowHammer attack in the system, leading to data corruption or significant performance degradation. To avoid such problems, we advocate co-architecting of the system and the memory together. A holistic solution that takes a system-memory co-design approach (as advocated earlier by [66, 72, 94, 99, 112]) can both prevent RowHammer bitflips and detect RowHammer attacks while at the same time avoiding potential performance and denial-of-service problems due to both RowHammer attacks and RowHammer mitigation mechanisms. For example, such a system can efficiently relocate/isolate data or throttle/relocate/isolate threads such that the performance of non-malicious applications is unaffected by RowHammer attacks or mitigations.[4] With worsening RowHammer vulnerability, such holistic solutions could pave the way to extremely efficient and fully-secure defenses against RowHammer.

We also believe that more flexible and efficient RowHammer solutions can take advantage of the wide variation in RowHammer vulnerability (as shown by [47, 70, 107, 153]) across 1) cells in a DRAM chip, 2) DRAM chips, 3) manufacturers, 4) DRAM types and generations, 5) environmental conditions, and 6) data patterns, to statically and dynamically adapt to system and workload characteristics. In current practice, RowHammer solutions need to be configured for the DRAM chip with the smallest RowHammer threshold. Such solutions are overly aggressive, as they try to cover the worst possible case, i.e., the most RowHammer-vulnerable DRAM chip that is acceptable to be sold, and can therefore induce large system performance and energy overheads that are unnecessary in the common case. We believe future RowHammer solutions need to be easily (re)configurable or programmable, such that they can statically and dynamically adapt to system and workload characteristics, and there is significant research needed towards enabling such flexible and efficient solutions.[5]

[4] BlockHammer [127, 155] takes a step towards this direction by throttling the threads that are identified as RowHammer attacks. We believe that there is more to be done to further reduce the performance problems due to both Rowhammer attacks and RowHammer mitigations.

[5] We believe the PARA defense mechanism [72] could be a suitable starting point for exploring such (re)configurable yet low-cost solutions due to its 1) fixed hardware cost that does *not* increase with worsening RowHammer vulnerability and 2) easily reconfigurable nature where updating the probability threshold directly tunes its aggressiveness, as described in our recent HiRA work at MICRO 2022 [154].

6 CONCLUSION

We provided a brief overview of the history and current state of research and development on the RowHammer vulnerability. We described major future research directions which we believe are critical to fundamentally understanding and solving RowHammer. We conclude that even though much research and development is done on the topic, a lot more needs to be done going forward, since RowHammer is a fundamental DRAM technology scaling problem that is getting worse in newer DRAM chips that will continue to be employed across almost all computing systems. We hope the discussion and ideas provided in this paper provide a useful path for the community to find ways of fundamentally understanding and efficiently solving the RowHammer problem.

ACKNOWLEDGMENTS

We thank the organizers of the ASP-DAC 2023 conference for the invitation to contribute this invited paper and deliver an associated invited talk. We acknowledge many SAFARI Research Group Members who have contributed to some of the works described in this paper, especially Jeremie Kim and Hasan Hassan. We thank all members of the SAFARI Research Group for the stimulating and scholarly intellectual environment they provide. We acknowledge the generous gift funding provided by our industrial partners (especially by Google, Huawei, Intel, Microsoft, VMware), which has been instrumental in enabling the decade+ long research we have been conducting on RowHammer. This work was in part supported by the Microsoft Swiss Joint Research Center.

REFERENCES

[1] M. T. Aga *et al.*, "When Good Protections Go Bad: Exploiting Anti-DoS Measures to Accelerate Rowhammer Attacks," in *HOST*, 2017.
[2] S. Agarwal *et al.*, "Rowhammer for Spin Torque based Memory: Problem or not?" in *INTERMAG*, 2018.
[3] S. M. Ajorpaz *et al.*, "EVAX: Towards a Practical, Pro-active & Adaptive Architecture for High Performance & Security," in *MICRO*, 2022.
[4] Apple Inc., "About the Security Content of Mac EFI Security Update 2015-001," https://support.apple.com/en-us/HT204934, 2015.
[5] Z. B. Aweke *et al.*, "ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks," in *ASPLOS*, 2016.
[6] K. Bains *et al.*, "Row Hammer Refresh Command," US Patents: 9,117,544 9,236,110 10,210,925, 2015.
[7] K. Bains *et al.*, "Method, Apparatus and System for Providing a Memory Refresh," US Patent: 9,030,903, 2015.
[8] K. S. Bains and J. B. Halbert, "Distributed Row Hammer Tracking," US Patent: 9,299,400, 2016.
[9] K. S. Bains and J. B. Halbert, "Row Hammer Monitoring Based on Stored Row Hammer Threshold Value," US Patent: 10,083,737, 2016.
[10] A. Barenghi *et al.*, "Software-only Reverse Engineering of Physical DRAM Mappings for Rowhammer Attacks," in *IVSW*, 2018.
[11] T. Bennett *et al.*, "Panopticon: A Complete In-DRAM Rowhammer Mitigation," in *DRAMSec*, 2021.

[12] S. Bhattacharya and D. Mukhopadhyay, "Curious Case of RowHammer: Flipping Secret Exponent Bits using Timing Analysis," in *CHES*, 2016.
[13] S. Bhattacharya and D. Mukhopadhyay, "Advanced Fault Attacks in Software: Exploiting the Rowhammer Bug," in *Fault Tolerant Architectures for Cryptography and Hardware Security*, 2018.
[14] A. Boroumand *et al.*, "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks," in *PACT*, 2021.
[15] A. Boroumand *et al.*, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," in *ASPLOS*, 2018.
[16] E. Bosman *et al.*, "Dedup Est Machina: Memory Deduplication as An Advanced Exploitation Vector," in *S&P*, 2016.
[17] R. Boyer and J. S. Moore, "A Fast Majority Vote Algorithm," Technical Report 35, Institute for Computer Science, UT Austin, 1982.
[18] F. Brasser *et al.*, "Can't Touch This: Software-Only Mitigation Against Rowhammer Attacks Targeting Kernel Memory," in *USENIX Security*, 2017.
[19] L. Bu *et al.*, "SRASA: a Generalized Theoretical Framework for Security and Reliability Analysis in Computing Systems," *Journal of Hardware and Systems Security*, 2018.
[20] W. Burleson *et al.*, "Invited: Who is the Major Threat to Tomorrow's Security? You, the Hardware Designer," in *DAC*, 2016.
[21] S. Carre *et al.*, "OpenSSL Bellcore's Protection Helps Fault Attack," in *DSD*, 2018.
[22] K. K. Chang *et al.*, "Improving DRAM Performance by Parallelizing Refreshes with Accesses," in *HPCA*, 2014.
[23] K. K. Chang *et al.*, "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM," in *HPCA*, 2016.
[24] Y. Cohen *et al.*, "HammerScope: Observing DRAM Power Consumption Using Rowhammer," in *CCS*, 2022.
[25] L. Cojocar *et al.*, "Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers," in *S&P*, 2020.
[26] L. Cojocar *et al.*, "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks," in *S&P*, 2019.
[27] G. F. de Oliveira *et al.*, "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks ," *IEEE Access*, 2021.
[28] F. de Ridder *et al.*, "SMASH: Synchronized Many-Sided Rowhammer Attacks from JavaScript," in *USENIX Security*, 2021.
[29] F. Devaux and R. Ayrignac, "Method and Circuit for Protecting a DRAM Memory Device from the Row Hammer Effect," US Patent: 10,885,966, 2021.
[30] S. Enomoto *et al.*, "Efficient Protection Mechanism for CPU Cache Flush Instruction Based Attacks," *IEICE Transactions on Information and Systems*, 2022.
[31] M. Fahr Jr *et al.*, "When Frodo Flips: End-to-End Key Recovery on FrodoKEM via Rowhammer," *CCS*, 2022.
[32] A. Fakhrzadehgan *et al.*, "SafeGuard: Reducing the Security Risk from RowHammer via Low-Cost Integrity Protection," in *HPCA*, 2022.
[33] M. Fang *et al.*, "Computing Iceberg Queries Efficiently," *VLDB*, 1998.
[34] M. Fischer and S. Salzberg, "Finding a Majority Among N Votes: Solution to Problem 81-5," *Journal of Algorithms*, 1982.
[35] A. P. Fournaris *et al.*, "Exploiting Hardware Vulnerabilities to Attack Embedded System Devices: A Survey of Potent Microarchitectural Attacks," *Electronics*, 2017.
[36] T. Fridley and O. Santos, "Mitigations Available for the DRAM Row Hammer Vulnerability," http://blogs.cisco.com/security/mitigations-available-for-the-dram-row-hammer-vulnerability, March 2015.
[37] P. Frigo *et al.*, "Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU," in *S&P*, 2018.
[38] P. Frigo *et al.*, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
[39] P. R. Genssler *et al.*, "On the Reliability of FeFET On-Chip Memory," *TC*, 2022.
[40] H. Gomez *et al.*, "DRAM Row-hammer Attack Reduction using Dummy Cells," in *NORCAS*, 2016.
[41] Z. Greenfield *et al.*, "Row Hammer Condition Monitoring," US Patent: 8,938,573, 2015.
[42] D. Gruss *et al.*, "Another Flip in the Wall of Rowhammer Defenses," in *S&P*, 2018.
[43] D. Gruss *et al.*, "Rowhammer.js: A Remote Software-Induced Fault Attack in Javascript," in *DIMVA*, 2016.
[44] J.-W. Han *et al.*, "Surround Gate Transistor With Epitaxially Grown Si Pillar and Simulation Study on Soft Error and Rowhammer Tolerance for DRAM," *IEEE TED*, 2021.
[45] H. Hassan *et al.*, "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *ISCA*, 2019.
[46] H. Hassan *et al.*, "A Case for Self-Managing DRAM Chips: Improving Performance, Efficiency, Reliability, and Security via Autonomous in-DRAM Maintenance Operations," arXiv:2207.13358, 2022.
[47] H. Hassan *et al.*, "Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications," in *MICRO*, 2021.
[48] H. Hassan *et al.*, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," in *HPCA*, 2017.
[49] Hewlett-Packard Enterprise, "HP Moonshot Component Pack Version 2015.05.0," http://h17007.www1.hp.com/us/en/enterprise/servers/products/moonshot/component-pack/index.aspx, 2015.
[50] S. Hong *et al.*, "Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks," in *USENIX Security*, 2019.
[51] G. Irazoqui *et al.*, "MASCAT: Stopping Microarchitectural Attacks Before Execution," *IACR Cryptology*, 2016.
[52] N. Izzo, "Reliably Achieving and Efficiently Preventing Rowhammer Attacks," Ph.D. dissertation, Politecnico Milano, 2017.
[53] Y. Jang *et al.*, "SGX-Bomb: Locking Down the Processor via Rowhammer Attack," in *SysTEX*, 2017.
[54] P. Jattke *et al.*, "Blacksmith: Scalable Rowhammering in the Frequency Domain," in *S&P*, 2022.
[55] JEDEC, *JESD79-5: DDR5 SDRAM Standard*, 2020.
[56] JEDEC, *JEP300-1: Near-Term DRAM Level RowHammer Mitigation*, 2021.
[57] JEDEC, *JEP301-1: System Level RowHammer Mitigation*, 2021.
[58] JEDEC, "JEDEC Global Standards for the Microelectronics Industry," https://www.jedec.org/, 2022.
[59] S. Ji *et al.*, "Pinpoint Rowhammer: Suppressing Unwanted Bit Flips on Rowhammer Attacks," in *ASIACCS*, 2019.
[60] B. K. Joardar *et al.*, "Learning to Mitigate RowHammer Attacks," in *DATE*, 2022.
[61] B. K. Joardar *et al.*, "Machine Learning-based Rowhammer Mitigation," *TCAD*, 2022.
[62] J. Juffinger *et al.*, "CSI: Rowhammer–Cryptographic Security and Integrity against Rowhammer (to appear)," in *S&P*, 2023.
[63] M. Kaczmarski, "Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – Component Selection Guidelines," http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf, page 13, 2014.
[64] S. Kanev *et al.*, "Profiling a Warehouse-Scale Computer," in *ISCA*, 2015.
[65] I. Kang *et al.*, "CAT-TWO: Counter-Based Adaptive Tree, Time Window Optimized for DRAM Row-Hammer Prevention," *IEEE Access*, 2020.
[66] U. Kang *et al.*, "Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling," in *The Memory Forum*, 2014.
[67] R. M. Karp *et al.*, "A Simple Algorithm for Finding Frequent Elements in Streams and Bags," *Transactions on Database Systems*, 2003.
[68] M. N. I. Khan and S. Ghosh, "Analysis of Row Hammer Attack on STTRAM," in *ICCD*, 2018.
[69] D.-H. Kim *et al.*, "Architectural Support for Mitigating Row Hammering in DRAM Memories," *IEEE CAL*, 2014.
[70] J. S. Kim *et al.*, "Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques," in *ISCA*, 2020.
[71] M. J. Kim *et al.*, "Mithril: Cooperative Row Hammer Protection on Commodity DRAM Leveraging Managed Refresh," in *HPCA*, 2022.
[72] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.
[73] Y. Kim *et al.*, "Ramulator: A Fast and Extensible DRAM Simulator," *CAL*, 2015.
[74] Y. Kim *et al.*, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.
[75] A. Kogler *et al.*, "Half-Double: Hammering From the Next Row Over," in *USENIX Security*, 2022.
[76] R. K. Konoth *et al.*, "ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks," in *OSDI*, 2018.
[77] A. Kwong *et al.*, "RAMBleed: Reading Bits in Memory Without Accessing Them," in *S&P*, 2020.
[78] E. Lee *et al.*, "TWiCe: Preventing Row-Hammering by Exploiting Time Window Counters," in *ISCA*, 2019.
[79] E. Lee *et al.*, "TWiCe: Time Window Counter Based Row Refresh to Prevent Row-Hammering," *IEEE CAL*, 2018.
[80] G.-H. Lee *et al.*, "CryoGuard: A Near Refresh-Free Robust DRAM Design for Cryogenic Computing," in *ISCA*, 2021.
[81] J. Lee, "Green Memory Solution," Samsung Electronics, Investor's Forum, 2014.
[82] Lenovo, "Row Hammer Privilege Escalation," https://support.lenovo.com/us/en/product_security/row_hammer, 2015.
[83] H. Li *et al.*, "Write Disturb Analyses on Half-Selected Cells of Cross-Point RRAM Arrays," in *IRPS*, 2014.
[84] M. Lipp *et al.*, "Nethammer: Inducing Rowhammer Faults Through Network Requests," arXiv:1805.04956, 2018.
[85] J. Liu *et al.*, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.
[86] L. Liu *et al.*, "Generating Robust DNN with Resistance to Bit-Flip based Adversarial Weight Attack," *IEEE TC*, 2022.
[87] K. Loughlin *et al.*, "Stop! Hammer Time: Rethinking Our Approach to Rowhammer Mitigations," in *HotOS*, 2021.
[88] K. Loughlin *et al.*, "MOESI-Prime: Preventing Coherence-Induced Hammering in Commodity Workloads," in *ISCA*, 2022.
[89] I. A. Lovecruft, "Tweet about RowHammer Mitigation on x210," https://twitter.com/isislovecruft/status/1021939922754723841, 2018.

[90] E. Manzhosov *et al.*, "Revisiting Residue Codes for Modern Memories," in *MICRO*, 2022.
[91] M. Marazzi *et al.*, "ProTRR: Principled yet Optimal In-DRAM Target Row Refresh," in *S&P*, 2022.
[92] Micron Inc., *SDRAM, 4Gb: x4, x8, x16 DDR4 SDRAM Features*, 2014.
[93] J. Misra and D. Gries, "Finding Repeated Elements," *Science of Computer Programming*, 1982.
[94] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *IMW*, 2013.
[95] O. Mutlu, "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser," in *DATE*, 2017.
[96] O. Mutlu, "RowHammer and Beyond," in *COSADE*, 2019.
[97] O. Mutlu *et al.*, "A Modern Primer on Processing in Memory," in *Emerging Computing: From Devices to Systems — Looking Beyond Moore and Von Neumann*. Springer, 2021. [Online]. Available: https://arxiv.org/abs/2012.03112
[98] O. Mutlu and J. Kim, "RowHammer: A Retrospective," *IEEE TCAD Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
[99] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems," *SUPERFRI*, 2014.
[100] A. Naseredini *et al.*, "ALARM: Active LeArning of Rowhammer Mitigations," https://users.sussex.ac.uk/~mfb21/rh-draft.pdf, 2022.
[101] K. Ni *et al.*, "Write Disturb in Ferroelectric FETs and Its Implication for 1T-FeFET AND Memory Arrays," *IEEE EDL*, 2018.
[102] S. Oh and J. Kim, "Reliable Rowhammer Attack and Mitigation Based on Reverse Engineering Memory Address Mapping Algorithms," in *WISA*, 2018.
[103] A. Olgun *et al.*, "DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips," arXiv:2211.05838 [cs.AR], 2022.
[104] A. Olgun *et al.*, "PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM," *TACO*, 2022.
[105] OpenSSH, "OpenSSH: Keeping Your Communiqués Secret," https://www.openssh.com/, 2017.
[106] L. Orosa *et al.*, "SpyHammer: Using RowHammer to Remotely Spy on Temperature," arXiv:2210.04084, 2022.
[107] L. Orosa *et al.*, "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses," in *MICRO*, 2021.
[108] J. H. Park *et al.*, "Row Hammer Reduction Using a Buried Insulator in a Buried Channel Array Transistor," *IEEE TED*, 2022.
[109] K. Park *et al.*, "Experiments and Root Cause Analysis for Active-precharge Hammering Fault in DDR3 SDRAM under 3× nm Technology," *Microelectronics Reliability*, 2016.
[110] K. Park *et al.*, "Statistical Distributions of Row-hammering Induced Failures in DDR3 Components," *Microelectronics Reliability*, 2016.
[111] Y. Park *et al.*, "Graphene: Strong yet Lightweight Row Hammer Protection," in *MICRO*, 2020.
[112] M. Patel *et al.*, "A Case for Transparent Reliability in DRAM Systems," arXiv:2204.10378, 2022. [Online]. Available: https://arxiv.org/abs/2204.10378
[113] P. Pessl *et al.*, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," in *USENIX Security*, 2016.
[114] D. Poddebniak *et al.*, "Attacking Deterministic Signature Schemes using Fault Attacks," in *EuroS&P*, 2018.
[115] S. Qazi *et al.*, "Half-Double: Next-Row-Over Assisted RowHammer," https://github.com/google/hammer-kit/blob/main/20210525_half_double.pdf, 2021.
[116] S. Qazi *et al.*, "Introducing Half-Double New Hammering Technique for DRAM RowHammer Bug," https://security.googleblog.com/2021/05/introducing-half-double-new-hammering.html, 2021.
[117] R. Qiao *et al.*, "A New Approach for RowHammer Attacks," in *HOST*, 2016.
[118] M. Qureshi, "Rethinking ECC in the Era of Row-Hammer," *DRAMSec*, 2021.
[119] M. Qureshi *et al.*, "Hydra: Enabling Low-Overhead Mitigation of Row-Hammer at Ultra-Low Thresholds via Hybrid Tracking," in *ISCA*, 2022.
[120] A. S. Rakin *et al.*, "DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories," in *S&P*, 2022.
[121] K. Razavi *et al.*, "Flip Feng Shui: Hammering a Needle in the Software Stack," in *USENIX Security*, 2016.
[122] S. H. S. Rezaei *et al.*, "NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories," *IEEE CAL*, 2020.
[123] S.-W. Ryu *et al.*, "Overcoming the Reliability Limitation in the Ultimately Scaled DRAM using Silicon Migration Technique by Hydrogen Annealing," in *IEDM*, 2017.
[124] SAFARI Research Group, "Ramulator — GitHub Repository," https://github.com/CMU-SAFARI/ramulator.
[125] SAFARI Research Group, "RowHammer — GitHub Repository," https://github.com/CMU-SAFARI/rowhammer, 2014.
[126] SAFARI Research Group, "SoftMC — GitHub Repository," https://github.com/CMU-SAFARI/softmc, 2017.
[127] SAFARI Research Group, "BlockHammer — GitHub Repository," https://github.com/CMU-SAFARI/blockhammer, 2021.
[128] SAFARI Research Group, "DRAM Bender — GitHub Repository," https://github.com/CMU-SAFARI/DRAM-Bender, 2022.
[129] SAFARI Research Group, "PiDRAM Source Code," https://github.com/CMU-SAFARI/PiDRAM, 2022.
[130] SAFARI Research Group, "Self-Managing DRAM (SMD) Source Code," https://github.com/CMU-SAFARI/SelfManagingDRAM, 2022.
[131] G. Saileshwar *et al.*, "Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation Between Aggressor and Victim Rows," in *ASPLOS*, 2022.
[132] S. Saroiu *et al.*, "The Price of Secrecy: How Hiding Internal DRAM Topologies Hurts Rowhammer Defenses," in *IRPS*, 2022.
[133] A. Saxena *et al.*, "AQUA: Scalable Rowhammer Mitigation by Quarantining Aggressor Rows at Runtime," in *MICRO*, 2022.
[134] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," *Black Hat*, 2015.
[135] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," http://googleprojectzero.blogspot.com.tr/2015/03/exploiting-dram-rowhammer-bug-to-gain.html, 2015.
[136] V. Seshadri *et al.*, "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization," in *MICRO*, 2013.
[137] V. Seshadri *et al.*, "Ambit: In-memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," in *MICRO*, 2017.
[138] S. M. Seyedzadeh *et al.*, "Mitigating Wordline Crosstalk Using Adaptive Trees of Counters," in *ISCA*, 2018.
[139] S. M. Seyedzadeh *et al.*, "Counter-based Tree Structure for Row Hammering Mitigation in DRAM," *IEEE CAL*, 2017.
[140] M. Son *et al.*, "Making DRAM Stronger Against Row Hammering," in *DAC*, 2017.
[141] A. Tatar *et al.*, "Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer," in *RAID*, 2018.
[142] A. Tatar *et al.*, "Throwhammer: Rowhammer Attacks Over the Network and Defenses," in *USENIX ATC*, 2018.
[143] Y. Tobah *et al.*, "SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks," in *S&P*, 2022.
[144] M. C. Tol *et al.*, "Toward Realistic Backdoor Injection Attacks on DNNs using RowHammer," arXiv:2110.07683, 2022.
[145] V. van der Veen *et al.*, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *CCS*, 2016.
[146] V. van der Veen *et al.*, "GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM," in *DIMVA*, 2018.
[147] A. J. Walker *et al.*, "On DRAM RowHammer and the Physics on Insecurity," *IEEE TED*, 2021.
[148] Y. Wang *et al.*, "FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching," in *MICRO*, 2020.
[149] Z. Weissman *et al.*, "JackHammer: Efficient Rowhammer on Heterogeneous FPGA–CPU Platforms," arXiv:1912.11523, 2020.
[150] X.-C. Wu *et al.*, "Protecting Page Tables from RowHammer Attacks using Monotonic Pointers in DRAM True-Cells," *ASPLOS*, 2019.
[151] Y. Xiao *et al.*, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *USENIX Security*, 2016.
[152] A. G. Yağlıkcı *et al.*, "Security Analysis of the Silver Bullet Technique for RowHammer Prevention," arXiv:2106.07084, 2021.
[153] A. G. Yağlıkcı *et al.*, "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices," in *DSN*, 2022.
[154] A. G. Yağlikci *et al.*, "HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips," in *MICRO*, 2022.
[155] A. G. Yağlıkcı *et al.*, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
[156] T. Yang and X.-W. Lin, "Trap-Assisted DRAM Row Hammer Effect," *EDL*, 2019.
[157] F. Yao *et al.*, "Deephammer: Depleting the Intelligence of Deep Neural Networks Through Targeted Chain of Bit Flips," in *USENIX Security*, 2020.
[158] J. M. You and J.-S. Yang, "MRLoc: Mitigating Row-Hammering Based on Memory Locality," in *DAC*, 2019.
[159] D. Yun *et al.*, "Study of TID Effects on One Row Hammering using Gamma in DDR4 SDRAMs," in *IRPS*, 2018.
[160] Z. Zhang *et al.*, "Triggering Rowhammer Hardware Faults on ARM: A Revisit," in *ASHES*, 2018.
[161] Z. Zhang *et al.*, "Leveraging EM Side-Channel Information to Detect Rowhammer Attacks," in *S&P*, 2020.
[162] Z. Zhang *et al.*, "PTHammer: Cross-User-Kernel-Boundary Rowhammer Through Implicit Accesses," in *MICRO*, 2020.
[163] Z. Zhang *et al.*, "SoftTRR: Protect Page Tables against Rowhammer Attacks using Software-only Target Row Refresh," in *USENIX ATC*, 2022.
[164] Z. Zhang *et al.*, "Implicit Hammer: Cross-Privilege-Boundary Rowhammer through Implicit Accesses," *IEEE TDSC*, 2022.
[165] M. Zheng *et al.*, "TrojViT: Trojan Insertion in Vision Transformers," arXiv:2208.13049, 2022.
[166] R. Zhou *et al.*, "LT-PIM: An LUT-based Processing-in-DRAM Architecture with RowHammer Self-Tracking," *IEEE CAL*, 2022.