IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 February 2024, accepted 2 May 2024, date of publication 4 June 2024, date of current version 14 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409389

RESEARCH ARTICLE

SpyHammer: Understanding and Exploiting RowHammer Under Fine-Grained Temperature Variations

LOIS OROSA[1,2], (Member, IEEE), ULRICH RÜHRMAIR[3,4], A. GIRAY YAĞLIKÇI[1], (Graduate Student Member, IEEE), HAOCONG LUO[1], ATABERK OLGUN[1], (Graduate Student Member, IEEE), PATRICK JATTKE[1], MINESH PATEL[1], (Member, IEEE), JEREMIE S. KIM[1], KAVEH RAZAVI[1], (Member, IEEE), AND ONUR MUTLU[1], (Fellow, IEEE)

[1]Information Technology and Electrical Engineering Department, ETH Zürich, 8092 Zürich, Switzerland
[2]Galicia Supercomputing Center (CESGA), 15705 Santiago de Compostela, Spain
[3]Experimental Quantum Physics Department, LMU München, 80539 Munich, Germany
[4]Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT 06269, USA

Corresponding author: Lois Orosa (lorosa@cesga.gal)

The work of Ulrich Rührmair was supported by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-21-1-0039.

ABSTRACT RowHammer is a DRAM vulnerability that can cause bit errors in a victim DRAM row solely by accessing its neighboring DRAM rows at a high-enough rate. Recent studies demonstrate that new DRAM devices are becoming increasingly vulnerable to RowHammer, and many works demonstrate system-level attacks for privilege escalation or information leakage. In this work, we perform the first rigorous fine-grained characterization and analysis of the correlation between RowHammer and temperature. We show that RowHammer is very sensitive to temperature variations, even if the variations are very small (e.g., $\pm 1\,^{\circ}C$). We leverage two key observations from our analysis to spy on DRAM temperature: 1) RowHammer-induced bit error rate consistently increases (or decreases) as the temperature increases, and 2) some DRAM cells that are vulnerable to RowHammer exhibit bit errors only at a particular temperature. Based on these observations, we propose a new RowHammer attack, called SpyHammer, that spies on the temperature of DRAM on critical systems such as industrial production lines, vehicles, and medical systems. SpyHammer is the first practical attack that can spy on DRAM temperature. Our evaluation in a controlled environment shows that SpyHammer can infer the temperature of the victim DRAM modules with an error of less than $\pm 2.5\,^{\circ}C$ at the 90^{th} percentile of all tested temperatures, for 12 real DRAM modules (120 DRAM chips) from four main manufacturers.

INDEX TERMS Rowhammer, DRAM, security, temperature.

I. INTRODUCTION

RowHammer is a DRAM vulnerability where a DRAM cell experiences a bitflip when its nearby cells are rapidly and frequently accessed [1]. Recent works [2], [3] demonstrate that modern DDR4 DRAM devices are more vulnerable to RowHammer than their predecessor DDR3 devices, suggesting that RowHammer is an important DRAM system design concern that is becoming increasingly severe as DRAM manufacturing technology nodes scale down. Using RowHammer, many works demonstrate attacks that escalate privilege at system-level, leak secret information, and manipulate critical application outputs [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33].

We perform the first rigorous fine-grained characterization and analysis of the correlation between RowHammer and

The associate editor coordinating the review of this manuscript and approving it for publication was Vivek Kumar Sehgal.

temperature, which lead to eight insightful observations and three key takeaways. Based on our observations and takeaways, we demonstrate SpyHammer, a new attack that uses RowHammer to spy on DRAM temperature with high accuracy. Our attack can be performed with minimal knowledge of the target computing system and can be used to compromise the security, confidentiality and privacy of critical systems that use DRAM. SpyHammer can compromise a victim computing system to achieve two goals. First, it can identify the utilization of a computer system, as the compute and memory intensity of a workload can change the temperature of the system. For example, an attacker can use SpyHammer to infer when a server is at its peak utilization by spying on its temperature. Second, SpyHammer can measure the ambient temperature, which may convey information about the state of a larger system that contains the target computing system (e.g., a car, a drone, or an industrial manufacturing machinery). For example, the temperature of a car's engine may rise if the engine is operating at high revolutions per minute.

SpyHammer not only compromises security and confidentiality, but also privacy. For example, by spying the temperature of a house (or different rooms of a house), an attacker can infer the habits of the person(s) living in that house. Tracking the temperature could give information about at which times the person(s) leave or enter a room in the house.

SpyHammer leverages two key observations about RowHammer to spy on DRAM temperature: 1) RowHammer-induced bit error rate consistently increases (or decreases) when temperature increases, and 2) some DRAM cells that are vulnerable to RowHammer experience bit errors only at a specific temperature. Using these observations, SpyHammer infers the temperature of DRAM chips by only characterizing DRAM cells that exhibit RowHammer-induced bit errors in the address space of the attacker without requiring any hardware or system software modifications. We propose two variants of the SpyHammer attack, each with a different threat model.

The first variant of SpyHammer can identify *relative temperature changes*, and it does not require prior physical access to or knowledge about the victim DRAM module. We observe that the correlation between Bit Errors per Row (*BER*) and temperature follows a similar trend in different DRAM modules of the same model and manufacturing date. We use this observation to spy on relative temperature changes. The key idea is to infer the model and manufacturing date of the victim DRAM module, and use a module with the same characteristics (to which the attacker has physical access and can control the operating temperature of) to infer the correlation between *BER* and temperature of the victim DRAM module. Since the attacker has *no* prior information about the victim DRAM module, the attacker must reverse engineer the victim DRAM module using remote RowHammer-based techniques [18], [34]. To estimate the temperature of the victim DRAM module, we propose to build a polynomial regression model using a DRAM module that has similar characteristics as the victim DRAM module.

The second variant of SpyHammer spies on *absolute temperatures*, which requires characterizing the victim DRAM module before the attack. The key idea is to build an accurate polynomial regression model using the characterization data of the victim DRAM module. This model is then used in the attack to accurately infer the victim DRAM module's temperature.

Reliably monitoring the *BER* of a DRAM module requires the attacker to hammer a large region of memory, which might increase the complexity of the attack. To reduce the number of DRAM accesses to the victim DRAM module, we propose a SpyHammer optimization that leverages the observation that some DRAM cells experience bitflips only at one particular temperature. We call these cells *canary cells*.[1]

The enrollment phase identifies the canary cells of the DRAM module. In this process, the cells that flip at only one temperature are added to the canary cell set. After the enrollment phase, an attacker can estimate the temperature of the victim DRAM by monitoring only a few selected canary cells, which reduces the number of memory accesses required to perform the attack.

To evaluate SpyHammer, we perform an extensive and thorough DRAM RowHammer characterization on 12 real DRAM modules (120 DRAM chips) using a temperature resolution of 1 °C in a controlled environment. Our results show that our methodology can infer 1) absolute temperatures (with prior characterization of the victim DRAM module) with an error of ±2.5 °C, and 2) relative temperature changes (without prior characterization of the victim DRAM module) with an error of ±3.5 °C, for all 12 DRAM modules we test, at the 90th percentile of tested temperature points (i.e., from 50 °C to 95 °C, with 1 °C step size).

We make the following main contributions:

- We perform the first rigorous *fine-grained* characterization and analysis of the correlation between RowHammer and temperature using 12 real DDR4 DRAM modules (120 DRAM chips).
- We show that RowHammer is very sensitive to temperature variations, even if the variations are very small (e.g., ±1 °C).
- We propose SpyHammer, the first RowHammer attack that can spy on DRAM temperature *without* any modification to the victim system. SpyHammer uses only the attacker's memory space (i.e., it does *not* corrupt the victim's memory space).
- We propose two variants of SpyHammer: 1) a variant that can spy on relative temperature changes *without* any prior information about or changes to the victim DRAM module, and 2) a variant that can spy on absolute temperature changes when the attacker has physical

[1] In all possible temperature points within a temperature range (given a particular temperature resolution), a canary cell experiences a bitflip at one and only one temperature point.

access to the victim DRAM module before deploying the attack.
- We perform a detailed study of the accuracy of the two SpyHammer variants, which shows that an attacker can spy with a maximum error of 1) ±2.5 °C on absolute temperature values, and 2) ±3.5 °C on relative temperature changes, in *all* 12 DRAM modules (120 DRAM chips) from the four major manufacturers we test.[2]

II. BACKGROUND

We provide a brief introduction to DRAM organization and RowHammer vulnerability. For more detailed background, we refer the reader to prior works [1], [2], [3], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76].

A. DRAM ORGANIZATION

The memory controller communicates with DRAM modules over one or more DRAM channels. Each module contains a set of DRAM chips that operate in lockstep. The DRAM cells within a DRAM chip are organized hierarchically. A DRAM chip comprises multiple DRAM banks that can operate independently. DRAM cells in a DRAM bank are laid out in a two-dimensional structure of rows and columns. Each DRAM cell on a DRAM row is connected to a common wordline via access transistors. A bitline connects a column of DRAM cells to a DRAM sense amplifier to access data.

Accesses to DRAM devices are typically performed in cache block granularity (64-bytes) in contemporary systems. An access to a DRAM cache block works in three steps. First, the memory controller sends an ACT command to activate a specific row within a DRAM bank, which prepares the row for a columns access (i.e., copies the row to the sense amplifiers). Second, the memory controller sends a READ (WRITE) command to read (write) a column in the row. Third, once all operations to the active row are completed, the memory controller sends a PRE command that closes the row and prepares the DRAM bank to open a new DRAM row (i.e., it precharges the bank).

B. ROWHAMMER

Modern DRAM devices are subject to disturbance failures caused by high frequency accesses (i.e., hammer) to DRAM rows (i.e., aggressor rows) that result in bitflips in physically nearby rows that are not being accessed (i.e., victim rows). This phenomenon is referred to as RowHammer [1], [2], [15], [23], [77], [78], [79], [80]. RowHammer-induced bitflips are exacerbated as DRAM technology nodes shrink and DRAM cells come closer to each other. This results in newer, higher-density DRAM chips to become more vulnerable to RowHammer [2] and other read disturbance effects [75].

[2]At the 90th percentile of tested temperature points

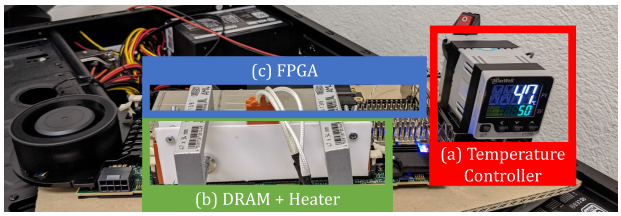


FIGURE 1. DRAM bender infrastructure: (a) temperature controller, (b) DRAM module clamped with heater pads, and (c) FPGA board programmed with DRAM bender [87].

These bitflips manifest after a row's activation count reaches a certain threshold value within a refresh window (usually denoted as MAC [81] or HC_{first} [2]).

Prior works devise many different RowHammer-based attacks, such as denial of service [17], [18], privilege escalation [4], [5], [6], [7], [9], [17], [18], [22], [31], [82], [83], secret data leakage [25], [32], [33], manipulation of the application correctness [24], [30] or private key recovery [84], [85]. A subset of these attacks require no physical access to a victim computing system; for example, attacks leveraging RDMA [34] or attacks in JavaScript programs [6].

III. METHODOLOGY

In order to thoroughly characterize the correlation between RowHammer and temperature and analyze the potential of the SpyHammer attack, we use an FPGA-based infrastructure that allows us to avoid uncontrolled interference in the system that might skew the results and lead to wrong insights and conclusions. To perform a SpyHammer attack on a real commodity computer system, we can use the methodology proposed in previous works [3], [31], [86] (not demonstrated in this paper).

A. TESTING INFRASTRUCTURE

We experimentally study DDR4 DRAM chips across a wide range of temperatures. We use the DRAM Bender framework [87], [88], which supports DDR4 modules, and a highly accurate temperature controller infrastructure.

1) DRAM BENDER

Figure 1 shows the DRAM Bender setup for testing DDR4 DRAM modules. We use the Xilinx Alveo U200 [89] FPGA board in all of our tests.

We use an FPGA board with DRAM Bender (Figure 1c) to perform all our RowHammer tests. We monitor and adjust the temperature of DRAM chips under test with a temperature controller (Figure 1a). This infrastructure provides us with fine-grained control over the timing between DRAM commands. We enforce all timing parameters defined by JEDEC [81] to ensure reliable operation.

2) TEMPERATURE CONTROLLER

To regulate the temperature in DRAM modules, we use silicone rubber heaters pressed to both sides of the DDR4

module (Figure 1c). To reduce the heat leakage, we apply two layers of insulation around the DRAM module under test and the heater pads: 1) a layer of reflective aluminum sheets covering the DRAM and the heater pads and 2) a layer of insulation sheets made of PTFE, a heat-resistant material. To measure the actual temperature of DRAM chips, we use a thermocouple, which we place between the rubber heaters and the DDR4 chips. We connect the heater pads and the thermocouple to a Maxwell FT200 temperature controller (Figure 1a), which keeps the temperature stable by implementing a closed-loop PID controller. Our host machine communicates with the temperature controller via an RS485 channel. Using this feature, we build custom software that enables us to automate the management of the temperature and integrate it into our testing infrastructure. In our tests using this infrastructure, we measure temperature with an accuracy of $\pm 0.1\,^{\circ}C$.

B. TESTING METHODOLOGY

1) DISABLING SOURCES OF INTERFERENCE

We disable all DRAM self-regulation events except the calibration signals, such as ZQ, for signal integrity so that we ensure that the observed errors are solely caused by RowHammer. We also make sure that our tests finish before retention errors manifest.

To the best of our knowledge, we also disable all DRAM-level (e.g., TRR [81]) and system-level RowHammer mitigation mechanisms (e.g., pTRR [90]) along with all forms of rank-level error-correction codes (ECC), which could obscure RowHammer bitflips. Based on the prior work's observations [2], [3], on-DRAM-die RowHammer mitigation mechanisms (i.e., TRR) take action when the DRAM services a refresh (REF) command. The DRAM modules we test do not implement error correction internally.

2) ROWHAMMER ACCESS SEQUENCE

We use a common access sequence used in previous works [1], [2], [77] as the worst-case access pattern, in which we 1) hammer the two rows that are adjacent to the victim row (i.e., aggressor rows), and 2) access the aggressor rows as frequently as possible. In our tests, we perform a double-sided RowHammer attack [1], [2].

3) DATA PATTERN

We conduct our experiments on a DRAM module by using the module's worst-case data pattern (*WCDP*). We identify the *WCDP* as the pattern that experiences the largest number of bitflips among seven different data patterns that prior research on DRAM characterization uses [2], [36], [46], [47], [48], [49], [58], presented in Table 1: colstripe, checkered, rowstripe, and random (we also test the complements of the first three). For each RowHammer test, we write the corresponding data pattern to the victim row (V in Table 1), and to the 8 previous ($V - [1 \ldots 8]$) and next ($V + [1 \ldots 8]$) physically-adjacent rows.

TABLE 1. Data patterns used in our RowHammer analyses.

Row Address	Colstripe†	Checkered†	Rowstripe†	Random
$V^* \pm [0, 2, 4, 6, 8]$	0x55	0x55	0x00	random
$V^* \pm [1, 3, 5, 7]$	0x55	0xaa	0xff	random

*V is the physical address of the victim row
†We also test the complements of these patterns

TABLE 2. Summary of DDR4 DRAM chips tested.

Manufacturer (Mfr.)	Model†	Module Id.	#Chips	Density	Die	Org.	Date (year/week)
A (Micron)	9TBJ	1	16	16GB	B	×4	19/11
	9TBJ	2	16	16GB	B	×4	19/11
	9TBJ	3	16	16GB	B	×4	19/11
B (Samsung)	8GNT	4	8	8GB	F	×8	21/02
	8GNT	5	8	8GB	F	×8	21/02
	8GNT	6	8	8GB	F	×8	21/02
C (Hynix)	S8/4	7	8	4GB	D	×8	19/46
	S8/4	8	8	4GB	D	×8	19/46
	S8/4	9	8	4GB	D	×8	19/46
D (Nanya)	PGRK	10	8	8GB	C	×8	21/12
	PGRK	11	8	8GB	C	×8	21/12
	PGRK	12	8	8GB	C	×8	21/12

† Last 4 digits of the model reference.

4) METRICS

We compare the *BER* across all our tests at a constant hammer count of 150K per aggressor row. We also identify the DRAM cells that flip only at a particular temperature point (i.e., canary cells).

5) ITERATIONS

To collect reliable results and estimate temperatures with SpyHammer, we repeat every single experiment 20 times for a particular DRAM module and temperature. We use the 20 repetitions of the experiments in different ways, depending on the particular evaluation (e.g., for estimating the accuracy on absolute temperature values in Section VII-A, we use the first 10 repetitions to build the estimation model and the other 10 repetitions to estimate the model's accuracy).

C. TESTED DRAM MODULES

Table 2 summarizes the 12 DDR4 modules (120 DRAM chips) we test from four major manufacturers. With the goal of testing our hypotheses (i.e., we can spy on the temperature if we know the model of the victim DRAM, or we can reverse engineer it), we test 3 modules from each manufacturer that are exactly the same model, and have exactly the same manufacturing date.

IV. FINE-GRAINED TEMPERATURE CHARACTERIZATION

We characterize RowHammer under fine-grained temperature variations. We focus on the study of both *BER* (Section VII-A) and canary cells (Section IV-B), as they are the most useful to spy on temperature (Section V-A).

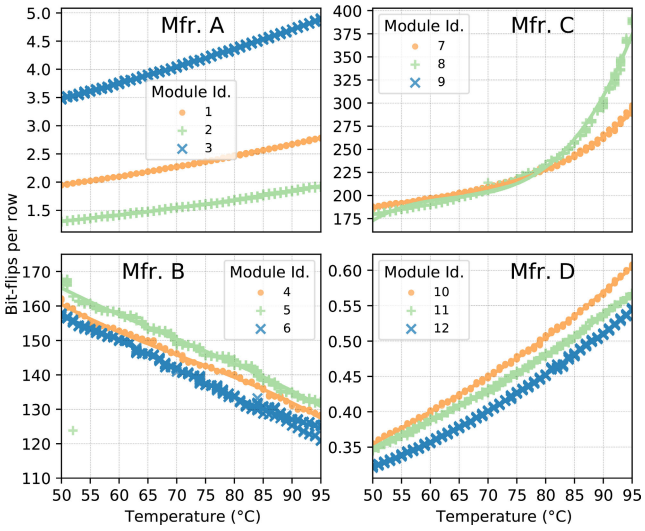


FIGURE 2. Correlation between RowHammer-induced bit flips per row (*BER*) and temperature.

A. EFFECT OF TEMPERATURE ON ROWHAMMER BER

We perform a fine-grained characterization of the correlation between *BER* and temperature, for four different manufacturers.

Figure 2 shows the RowHammer-induced *BER* for temperatures from 50°C to 95°C, with a resolution of 1°C. We analyze the 12 DRAM modules described in Table 2. We test a 192MB memory region (i.e., 24K DRAM rows) for each DRAM module, and we repeat each experiment 20 times for each tested temperature. For each module, we also plot the polynomial regression model (a continuous line with the same color as the Module's color) that better fits the BER changes across the entire temperature range.

Observation 1. *The absolute BER values of 2 identical modules might differ significantly.*

Although identical modules (i.e., same manufacturer, model, and fabrication date) might have similar *BER* (e.g., modules from manufacturer B), other modules *BER*'s might differ significantly. For example, module Id. 3 from Mfr. A shows on the order of 2.5× more bit flips than module Id. 2.

Thus, we cannot assume that two identical modules have similar *BER*. From all the DRAM modules we tested, we can only assume that identical modules *BER*'s are in the same order of magnitude.

Observation 2. *The BER might increase or decrease depending of the DRAM manufacturer.*

For the same manufacturer, we always observe a consistent trend among all DRAM modules we test, which is consistent with the observations of a previous work [77]. Particularly, the *BER* increases with temperature in Mfrs. A, C and D, and it decreases with temperature for Mfr. B.

Observation 3. *The BER of two identical modules when the temperature changes follows a similar trend.*

When we plot the polynomial regression, we observe that identical modules follow very similar curves in the plot. For example, although the absolute *BER* values differ slightly, all curves from Mfr. D modules have approximately the same shape in the figure. The exception is module 8 (Mfr. C), which follows a different curve than modules 7 and 9. We speculate that Mfr. C might have used different chips in module 8, even when the external labeling is exactly the same as in the other 2 modules.

Observation 4. *Most modules show a nearly linear relation between BER and temperature*

This is the case for Mfrs. A, B and D. However, module 8 from Mfr. C has a non-linear relation between *BER* and temperature when the temperature is higher than 70 °C.

Observation 5. *The BER is very stable across different repetitions of the same experiment, using the same module.*

The *BER* across 20 repetitions of the experiment in one module is very stable, as we can observe in the figure by the little variation within the y-axis for each temperature.

Takeaway 1. *The evolution of the BER values when increasing temperature follows a similar curve in DRAM modules with the same characteristics, even if the absolute BER values differ significantly.*

1) BER VARIATIONS IN DIFFERENT DRAM REGIONS

We study the *BER* of 48 DRAM regions of 4MB per DRAM module, for all modules we test, for temperatures from 50°C to 95°C, with a resolution of 1°C.

Figure 3 shows a box plot[3] that illustrates the correlation between the *BER* and the temperature. For each temperature, the represented data is the *BER* from the 48 different DRAM regions of the DRAM module.

Observation 6. *The variation of BER values across different DRAM regions in a module is reasonably small.*

We observe in the figure that, for all DRAM modules, the box at each temperature is pretty narrow, which indicates that the *BER* is very similar for all 48 regions of each DRAM module.

Observation 7. *The correlation of BER with temperature follows the same trend in different regions.*

We observe that both the boxes and the whiskers follow the same trend, from which we can infer that the *BER* curves from all regions are similar. We confirm this observation by comparing the curves from all 48 regions for each DRAM module (not shown in the figure).

Takeaway 2. *The evolution of BER values when increasing temperature follows a similar curve in different DRAM regions within the same DRAM module.*

[3] In a box plot [91], the box shows the lower and upper quartile of the data (i.e., the box spans the 25th to the 75th percentile of the data). The line in the box represents the median. The bottom and top whiskers each represent an additional 1.5× the *inter-quartile range* (IQR, the range between the bottom and the top of the box) beyond the lower and upper quartile, respectively.

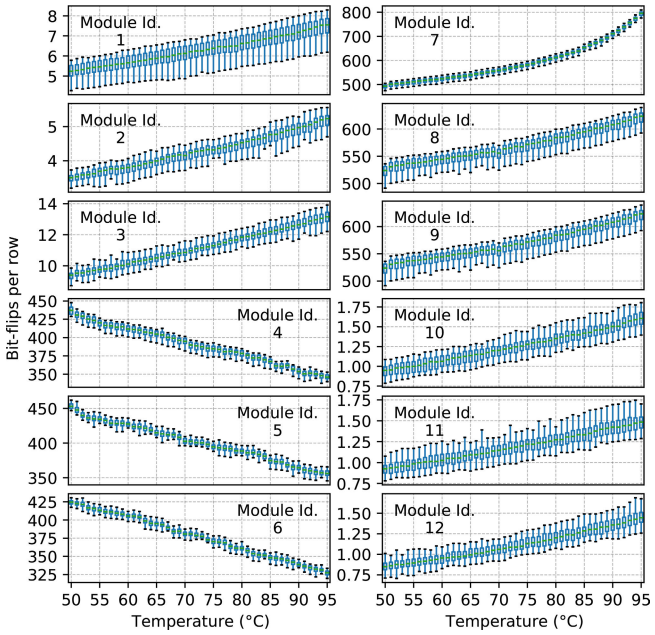


FIGURE 3. Box plot of RowHammer-induced *BER* in 48 different 4MB memory regions from the same DRAM module.

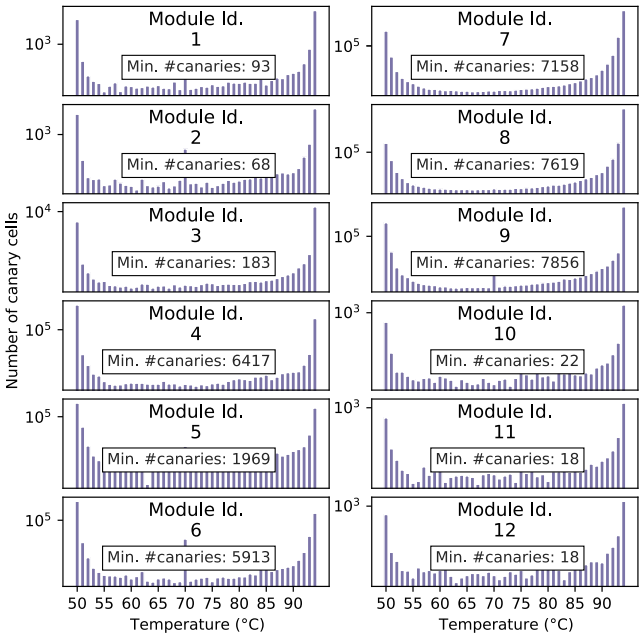


FIGURE 4. Number of canary DRAM cells at each temperature point.

B. CHARACTERIZATION OF CANARY CELLS

We characterize all DRAM modules to identify the canary cells (i.e., cells vulnerable to RowHammer at one specific temperature, but not vulnerable at any other temperature) at each temperature we test. To identify canary cells, we select those cells that 1) experience bit flips at least once in 10 repetitions of the experiment at a particular temperature point, and 2) only experience bit flips at that temperature point.

Figure 4 shows the number of canary cells (in logarithmic scale) per temperature point, and the minimum number of canary cells across all temperature points (Min. #canaries).

Observation 8. *There are plenty of canary cells throughout the entire temperature range when using a high temperature resolution (*1 °C*).*

All temperature points we test have at least 18 canary cells (module Id. 11 and 12) in the worst case, in all modules we test. The minimum number of canary cells is large enough, as only one canary cell is needed to estimate the temperature at a particular temperature point. [4]

Takeaway 3. *For a given temperature range, with a high temperature resolution (*1 °C*), there are plenty of canary cells in all modules we test.*

[4]Canary cells at the limits of the temperature range are more abundant than in the middle of the range. This phenomenon is caused by the limited temperature range, so some canary cells at the lower or upper limits might not be canary cells on an extended temperature range. For example, a canary cell at 50 °C in a 50 °C-90 °C temperature range, might not be a canary cell in the tamperature range 49 °C-90 , as the cell might also flip at both 50 °C and 49 °C.

V. OVERVIEW: THE SPYHAMMER ATTACK

We describe how to perform a SpyHammer attack that spies on the *temperature* of the victim DRAM chip, leveraging the observations and takeaways we make in Section IV. Section V-A describes how to perform a SpyHammer attack that spies on *relative temperature changes*, and Section V-B describes how to perform a SpyHammer variant that spies on *absolute temperatures*.

A. SPYING ON RELATIVE TEMPERATURE VARIATIONS USING SPYHAMMER

The basic SpyHammer attack is based on two key observations. First, the RowHammer-induced *BER* consistently increases (decreases) when the temperature increases [77] (Observation 2 in Section IV). We use this observation to infer if the temperature increases or decreases compared to a reference temperature point by just monitoring the *BER* in a DRAM region. Second, the form of the curve that relates *BER* and temperature is usually very similar across modules from the same manufacturer and manufacturing date (Observation 3 in Section IV).

Based on these observations, SpyHammer can detect relative changes on DRAM chip temperature by 1) continuously monitoring the *BER* of the victim DRAM module at different points in time, and 2) correlating the *BER* changes with temperature changes on the victim DRAM module using a temperature-*BER* polynomial regression model obtained from a DRAM module that has the same characteristics as the victim DRAM module.

1) CANARY CELL OPTIMIZATION

Obtaining consistent and reliable *BER* numbers might require hammering a large region of the DRAM module, which can

make the SpyHammer attack intrusive. To solve this problem, we propose an optimization to reduce the number of hammers needed to spy on temperature. This optimization is based on the observation that some DRAM cells are vulnerable to RowHammer only at a very narrow temperature range (Observation 8 in Section IV). We use this observation to make SpyHammer faster and less intrusive by identifying the cells that flip only at one temperature point (i.e., canary cells). By using canary cells instead of *BER* to spy on temperature, we can reduce the number of DRAM accesses to perform the SpyHammer attack.

The canary cell optimization is performed in two steps. First, the attacker enrolls canary cells for different temperatures by associating vulnerable DRAM cells to a particular *BER* value (i.e., to a particular temperature). For an accurate enrollment phase, the victim DRAM experiments should operate at different temperatures, so the *BER* and the vulnerable DRAM cells also change. For enrolling as many canary cells as possible, it is important that the victim DRAM module experiences all possible temperature variations in regular operational conditions. Depending on the particular system to monitor and the required accuracy of the attack, a complete and full enrollment process might take from a few hours to one year (e.g., the system might experience different temperatures at different seasons of the year). Second, after the canary cells are identified, the attacker can monitor these canary cells to spy on the temperature of the victim DRAM module.

2) THREAT MODEL

We assume an attacker with *no* prior knowledge or physical access to the victim DRAM module. Thus, the victim's software and hardware cannot be characterized or modified by the attacker prior to the attack. With the goal of building models that correlate *BER* with temperature, the attacker can purchase many DRAM modules from different manufacturers and has the infrastructure to control the DRAM temperature of those modules in a fine-grained manner. To build an accurate model, the attacker needs to characterize a DRAM module very similar to the victim module (e.g., the same manufacturer and model), which requires knowing or reverse-engineering the manufacturer and model of the victim DRAM module remotely.

B. SPYING ON ABSOLUTE TEMPERATURE VALUES USING SPYHAMMER

This variant of the SpyHammer attack aims to spy on absolute temperature values. To this end, the attacker must have physical access to the DRAM module and characterize it under different temperature conditions before the attack, using a similar methodology as in Section V-A, but without the need to reverse-engineer the DRAM model. Because the attacker uses the victim DRAM module to build the polynomial regression model, and they have local and accurate control over the DRAM temperature, the attack is more precise, as we demonstrate in Section VII-A. The rest of the attack is identical to the attack in Section V-A. We discuss the limitations of this attack model in Section VIII-B.

Threat Model: We assume an attacker that has access to the victim DRAM module prior to performing the attack and can characterize the bit errors caused by RowHammer in a controlled environment at different temperatures.

VI. THE SIX STEPS OF THE SPYHAMMER ATTACK

This section describes in detail the six steps of the SpyHammer attack on relative temperature changes (Section V-A):

1) Identify the victim DRAM module
2) Build a polynomial regression model from a DRAM module that is the same model as the victim
3) Allocate a contiguous DRAM memory region in the victim system
4) Monitor the *BER* of the victim DRAM module continuously
5) Enroll canary cells in the victim DRAM module
6) Monitor canary cells in the victim DRAM module to infer its temperature

A. STEP 1: IDENTIFY THE VICTIM DRAM MODULE

SpyHammer requires reverse engineering the manufacturer and model of the victim DRAM module. Our methodology identifies the DRAM manufacturer first and then identifies the technology node of the DRAM module. There are several techniques to reverse-engineer the victim DRAM manufacturer and model remotely, without physical access or modification to the victim system. Next, we explain some of these techniques an attacker can use based on new observations and observations from previous works.

We classify our methodologies into two categories: 1) methodologies that are very simple and quick to implement but that can lead to certain inaccuracies (Section VI-A1), and 2) methodologies that are more complex but that generally provide more accurate results (Section VI-A2). The best methodology for each case can be chosen based on the required accuracy or attack time.

1) QUICKLY IDENTIFYING THE DRAM MANUFACTURER AND MODEL

We can quickly infer some characteristics of the DRAM manufacturer of the victim DRAM module remotely by using two simple techniques based on RowHammer. These techniques can not accurately distinguish between Mfr. A and Mfr. D modules, so they might have to be complemented with more sophisticated techniques (Section VI-A2) when required.

a: MFR. B: UNIQUE LOGICAL-TO-PHYSICAL ROW MAPPING

DRAM manufacturers use DRAM internal mapping schemes to translate memory-controller-visible row addresses to physical row addresses in DRAM [1], [16], [26], [36], [37], [47], [56], [59], [64], [92], [93], [94], [95], [96], [97]. We use the logical-to-physical row mapping to uniquely identify

TABLE 3. Logical-to-physical mapping of DRAM row addresses.

Logical-to-Physical Row Mapping	Manufacturer (Mfr.)
$phy[x] = log[x]^{\dagger}$	A (Micron), C (Hynix), and D (Nanya)
$phy[0] = log[0]$ $phy[1] = log[3] \oplus log[1]$ $phy[2] = log[2] \oplus log[3]$ $phy[y] = log[y]^{\ddagger}$	B (Samsung)

† where x ranges from 0 to N.
‡ where y ranges from 3 to N.

Mfr. B modules. We observe that Mfr. A, Mfr. C, and Mfr. D modules use a sequential logical-to-physical row mapping, whereas Mfr. B uses a unique logical-to-physical row mapping in which two adjacent row addresses sent from the memory controller might map to non-adjacent physical locations in the DRAM chip.

To discover the logical-to-physical row mapping of a DRAM module, we use the observation that, when performing a single-sided RowHammer attack, the rows with more RowHammer-induced bitflips are the rows that are physically adjacent to the attacker row. Using this observation, we perform a simple iterative algorithm that infers the logical-to-physical mapping. Table 3 shows the logical-to-physical row mapping from all four major DRAM manufacturers we test. In the table, each element in the address, physical (*phy*) or logical (*log*), represents a bit, where bit 0 is the least significant bit of the address.

We make two observations. First, Mfr. B is the only manufacturer that uses a non-sequential logical-to-physical row mapping. Second, all Mfr. B modules have the same logical-to-physical row mapping in all modules we test. We verify that Mfr. B uses this mapping also for modules other than the ones we use for performing our thorough characterization (Table 2). We conclude that an attacker can identify Mfr. B modules solely by reverse-engineering their unique logical-to-physical row mapping.

b: MFR. A AND MFR. D: SINGLE-SIDED ROWHAMMER TESTS

We make the new observation that performing single-sided RowHammer usually only affects one neighboring row in Mfr. A and Mfr. D modules. For example, hammering row X causes bitflips only in $X+1$, and hammering row $X+1$ causes bitflips only in row X. In Mfr. B and Mfr. C modules, when performing a single-sided RowHammer attack, two victim rows are susceptible to bitflips. We speculate that this observation is caused by microarchitectural design decisions.[5] implementation. We conclude that we can use a single-sided RowHammer attack to identify DRAM modules from Mfr. A and D.

[5]DRAM manufacturers do not reveal any detail about the internal DRAM microarchitecture

c: IDENTIFYING THE DRAM MODEL

Modules from the same manufacturer might have different characteristics, depending on the manufacturing process, manufacturing date, etc. To identify the particular DRAM model of a DRAM module, we use the observation that the *BER* of modules from the same manufacturer but different technology nodes differ significantly. This observation could also be used to infer the DRAM manufacturer, as the absolute *BER* values from different manufacturers also differ very significantly in the modules we test (see Section VII-A).

2) ACCURATELY IDENTIFYING THE DRAM MANUFACTURER AND MODULE

U-TRR [86] assesses the security guarantees of recent DRAM chips by reverse engineering proprietary on-die RowHammer mitigation mechanisms, commonly known as Target Row Refresh (TRR). TRR detects and refreshes potential RowHammer-victim rows, but its exact implementations are not openly disclosed. U-TRR is based on the new observation that data retention failures in DRAM enable a side channel that leaks information on how TRR refreshes potential victim rows. The authors show in their evaluation that it is possible to reverse engineer the TRR mechanism for many different DRAM models from 3 major manufacturers. Our observation is that each manufacturer implements TRR in a different way, and, for modules from the same manufacturer, there are also a wide variety of TRR implementations. We conclude that we can use the techniques proposed by [86] to uniquely identify a particular DRAM model.

B. STEP 2: BUILD THE POLYNOMIAL REGRESSION MODEL

Although the absolute *BER* of two different modules from the same manufacturer and technology node might differ significantly, *BER* and temperature are very correlated (i.e., the shape of the polynomial regression curve is very similar in most cases). Thus, using the data obtained from the *BER* characterization, an attacker can 1) build a polynomial regression model from a DRAM module with the same characteristics as the victim DRAM module and 2) use the model to estimate the relative temperature changes of the victim DRAM module.

Using a polynomial regression model, we can estimate the temperature change. In our evaluation, we demonstrate that a polynomial regression model of order 3 is enough to estimate temperature with $\pm 5\,^{\circ}$C accuracy when estimating relative temperature changes, and $\pm 1\,^{\circ}$C accuracy when estimating absolute temperatures, for most DRAM modules we test.

C. STEP 3: ALLOCATE A CONTIGUOUS MEMORY REGION

Unlike other RowHammer attacks [7], SpyHammer does not require sophisticated techniques to place the victim memory row (not controlled by the attacker) into a particular physical row that is adjacent to the aggressor row controlled by the attacker. Instead, in SpyHammer, both the victim and the aggressor row are in the attacker's own memory space.

The attacker only has two requirements when allocating memory from the victim system. First, the memory region should be as contiguous as possible (i.e., the memory space is not very fragmented), so when performing hammers, the aggressor and victim row are neighbors with high probability. This is important not only for maximising the *BER*, but also for not corrupting the memory of other processes that use memory in adjacent memory regions. Second, for using the canary cell optimization, the memory region should not migrate to different physical locations during the attack, as the canary cells change from region to region. When using *BER* to spy on temperature, this is not required, as different regions within the same DRAM module have similar *BER* (see Section VII-A).

To identify if the physical memory region changes at different points in time, the attacker can use memory deduplication to reverse-map any physical page into a virtual page [7].

D. STEP 4: MONITOR THE BER OF THE VICTIM DRAM MODULE

SpyHammer is carried out in a memory region entirely in the attacker's address space. To characterize the *BER* of the memory region, the attacker simply needs to perform a RowHammer attack to every row within their address space with a double-sided RowHammer attack (as described in Section III), and count the total number of RowHammer-induced bitflips in the memory region. More sophisticated attacks can be used to trigger the attack (e.g., Blacksmith [31]), but we evaluate a double sided attack to simplify the comparison and to extract more clear conclusions.

The attacker initially establishes an estimated reference temperature by correlating the measured *BER* to the estimated temperature using the polynomial regression model (Step 3). In subsequent *BER* measurements and temperature estimations, the attacker can estimate the relative temperature changes using the model.

E. STEP 5: ENROLL CANARY CELLS

To improve performance and reduce the probability of the SpyHammer attack being detected, we identify and monitor canary cells, i.e., cells that only flip at a particular temperature point. By using canary cells, an attacker only needs to characterize (i.e., hammer) a few cells in the memory region instead of the entire memory region. Before using canary cells to estimate temperature, the attacker needs to identify those canary cells in a variety of different temperatures (i.e., enrollment). Enrollment requires characterizing the victim system during a long-enough time period to the system work in a wide variety of temperatures.

The canary enrollment process registers the cells that will be used as canaries for detecting temperature changes. This process requires monitoring DRAM cells while characterizing the *BER* (Step 4). To this end, the attacker 1) associates every different *BER* value (with a specified error tolerance) with the DRAM cells that experience bitflips at only at the associated *BER* (i.e., canary cells), and 2) eliminates duplicated canary cells from the previously generated *BER*-canary cells pairs. By using this methodology iteratively, the attacker can identify which cells are vulnerable to RowHammer at a specific *BER* value.

F. STEP 6: MONITOR CANARY CELLS TO INFER THE TEMPERATURE OF THE VICTIM'S DRAM

After enrolling the canary cells, the attacker monitors canary cells (i.e., check if the enrolled canary cells are vulnerable to RowHammer) to detect temperature changes, instead of accessing large regions of DRAM to calculate the *BER*. By comparing the *BER* associated with different canary cells, the attacker can infer relative temperature changes using the same methodology as used in Step 4.

To minimize the number of DRAM accesses, the attacker can use a limited number of canary cells per temperature or access DRAM rows with many canary cells.

VII. SPYHAMMER EVALUATION

We evaluate SpyHammer's accuracy and sensitivities when using both *BER* (Section VII-A) and canary cells (Section IV-B) to spy on relative temperature changes and absolute temperature.

A. BER ANALYSIS

From Observations 1-5 and Takeaway 1 (Section IV-A), we infer that the similar correlation between *BER* and temperature between modules from the same manufacturer can be used to estimate the temperature of the victim DRAM module.[6] To do so, we find that a polynomial regression model of order 3 is sufficient to accurately model the correlation between *BER* and temperature for all modules we test. Table 4 shows the polynomial regression equations inferred for each DRAM module.

We observe that the regression curve is similar between modules from the same manufacturer. We conclude that we can spy on temperature by using a polynomial regression model from a DRAM module that is the same model as the victim DRAM module. However, the accuracy of this methodology can significantly change depending on the DRAM module and threat model.

1) METHODOLOGY FOR MEASURING ACCURACY

To evaluate the accuracy of our methodology to estimate temperature using a polynomial regression model, we select a sequence of 720 random temperature points in the tested range (i.e., from 50 °C to 95 °C), and we measure the accuracy of the estimation for all temperatures in the sequence. For spying on relative temperature changes, we use the change between consecutive temperature points in the sequence, and for spying on absolute temperature, we use

[6]For a particular model of a Mfr. D module (not shown in our evaluation results), we cannot observe enough RowHammer-induced bitflips to make conclusive observations about the relation between RowHammer and Temperature.

TABLE 4. Polynomial regression that models the correlation between *BER* and temperature.

Id.	Polynomial Regression
1	$y = -(1.9 * 10^{-6}) * x^3 + (4.8 * 10^{-4}) * x^2 - (2.2 * 10^{-2}) * x + 2.1$
2	$y = +(6.8 * 10^{-7}) * x^3 - (6.4 * 10^{-5}) * x^2 + (1.2 * 10^{-2}) * x + 0.8$
3	$y = +(4.0 * 10^{-7}) * x^3 + (3.0 * 10^{-5}) * x^2 + (2.0 * 10^{-2}) * x + 2.3$
4	$y = -(1.9 * 10^{-4}) * x^3 + (4.0 * 10^{-2}) * x^2 - 3.5 * x + 260$
5	$y = +(9.5 * 10^{-5}) * x^3 - (2.1 * 10^{-2}) * x^2 + 0.8 * x + 157.7$
6	$y = +(2.4 * 10^{-4}) * x^3 - (4.9 * 10^{-2}) * x^2 + (2.5) * x + 121.8$
7	$y = +(1.4 * 10^{-3}) * x^3 - 0.2 * x^2 + 15.6 * x - 152.2$
8	$y = +(5.1 * 10^{-3}) * x^3 - 1.0 * x^2 + 64.1 * x - 1201.8$
9	$y = +(8.1 * 10^{-5}) * x^3 - (9.9 * 10^{-3}) * x^2 + (0.9) * x + 167.6$
10	$y = +(4.5 * 10^{-7}) * x^3 - (6.3 * 10^{-5}) * x^2 + (7.4 * 10^{-3}) * x + 0.1$
11	$y = +(3.4 * 10^{-7}) * x^3 - (4.3 * 10^{-5}) * x^2 + (5.7 * 10^{-3}) * x + 0.1$
12	$y = -(1.2 * 10^{-7}) * x^3 + (6.4 * 10^{-5}) * x^2 + (2.5 * 10^{-3}) * x + 0.3)$

TABLE 5. Maximum temperature error (in °C) for 90th percentile of the error distribution, when estimating relative temperature changes.

Module Id.	1	2	3	4	5	6	7	8	9	10	11	12
Error (L)† (°C)	7.8	12.6	5.6	3.9	4.6	3.9	14.5	6.6	23.1	2.0	1.9	1.0
Error (S)‡ (°C)	3.1	3.1	3.3	3.4	2.4	3.2	2.6	3.3	2.6	2.0	1.9	1.0

† for relative temperature changes up to 45 °C.
‡ for relative temperature changes up to 5 °C

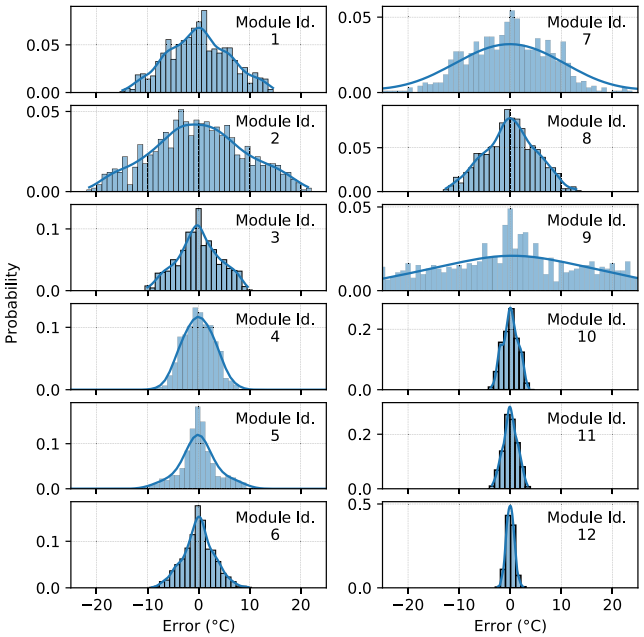


FIGURE 5. Error of the relative temperature change estimation obtained using a polynomial regression model from a DRAM module that is the same model as the victim DRAM module.

each temperature point in the sequence. We make sure that the sequence of random temperatures includes extreme temperature changes (e.g., from 95 °C to 50 °C), and small temperature changes (e.g., from 50 °C to 51 °C).

2) ACCURACY WHEN SPYING ON RELATIVE TEMPERATURE CHANGES

We study the accuracy of spying the relative temperature changes of the victim DRAM module using a polynomial regression model obtained from a DRAM module that is the same model as the victim DRAM module. Figure 5 shows the probability distribution of the temperature error when we use the polynomial regression model. For each victim DRAM module of each manufacturer, we use a polynomial regression model obtained from one of the other two modules we test from the same manufacturer.

Table 5 shows the maximum error in the estimation of the relative temperature changes (in °C), for 90th percentile of the error distribution. We show values for temperature changes up to 45 °C (L), and for temperature changes up to 5 °C (S), which should be the common case if the *BER* monitoring frequency is large enough.

We make two observations. First, the error of the temperature estimated by the regression model is reasonably low for modules from Mfr. B (Module Id. 4, 5, 6) and Mfr. D (Module Id. 10, 11, 12), because the correlation between *BER* and temperature is very similar across all modules from the same manufacturer. We find that the error is larger for Mfr. A (Module Id. 1, 2, 3) and Mfr. C (Module Id. 7, 8, 9) because in each of these manufacturers, there is one module that has a slightly different curve than the other two modules. Second, in many cases, the estimation error is zero (i.e., error = 0 °C). We observe that when the temperature change is small (e.g., <5 °C), the estimation error is small for all modules (<3.5 °C), whereas when the temperature change is large (e.g., up to 45 °C) the estimation error is much larger (only 3 modules have less than 3.5 °C error).

We make two conclusions. First, for all possible temperature changes (i.e., from very small to very large temperature changes), a polynomial regression model works well if the modules show very similar correlations between *BER* and temperature. Second, for small temperature changes, we can get accurate temperature estimations from all modules we test. By sampling at a high enough frequency, this should be always the case.

3) ACCURACY WHEN SPYING ON ABSOLUTE TEMPERATURE VALUES

We study the accuracy of spying on the absolute temperature of the victim DRAM module using a polynomial regression model obtained from the victim DRAM module itself. Figure 6 shows the probability distribution of the error when estimating the absolute temperature.

Table 6 shows the maximum error in the estimation of the absolute temperatures (in °C), for 90th percentile of the error distribution.

We make the main observation that the absolute temperature estimations are very accurate for most of the DRAM modules. All modules we tested show an error lower than 2.5 °C for 90th percentile of the error distribution.

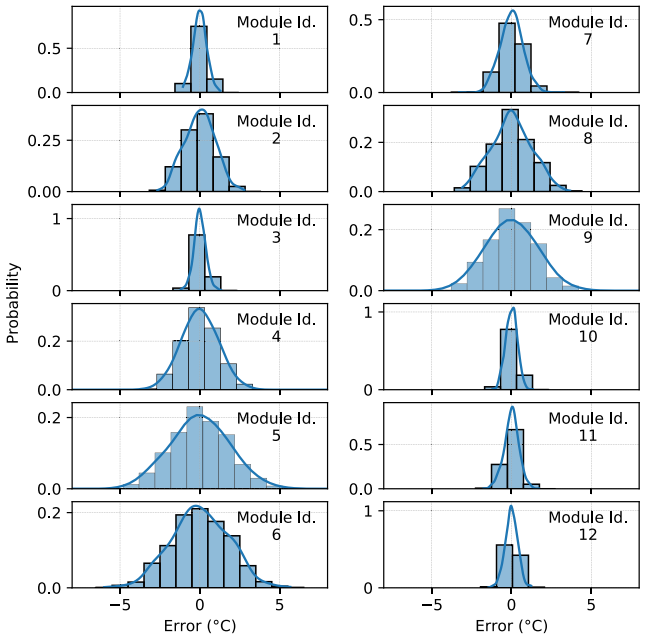


FIGURE 6. Error of the absolute temperature estimation obtained with a polynomial regression model from the victim DRAM module.

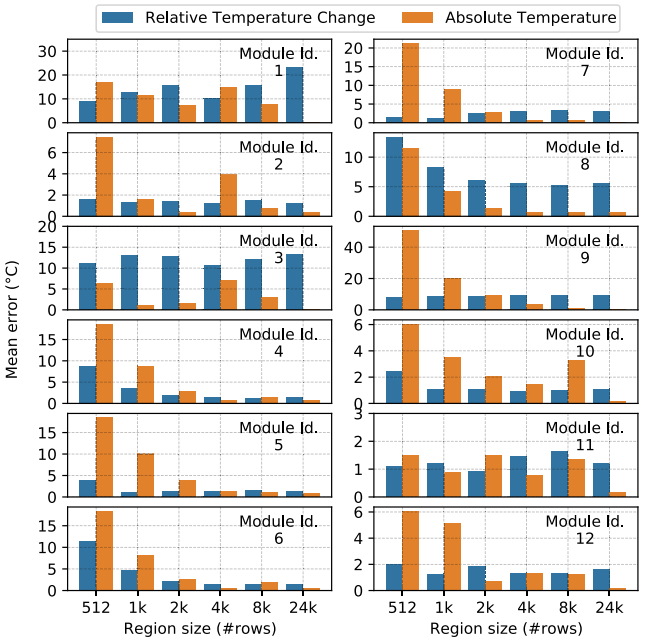


FIGURE 7. Mean error of the temperature estimated by the polynomial regression model for different region sizes.

TABLE 6. Maximum temperature error (in °C) for 90th percentile of the error distribution, when estimating absolute temperatures.

Module Id.	1	2	3	4	5	6	7	8	9	10	11	12
Error (°C)	0.5	1.2	0.5	1.4	2.3	2.3	0.9	1.7	1.9	0.5	0.6	0.5

We conclude that 1) using a regression model from the victim DRAM module provides accurate results, and 2) a polynomial regression model is enough to accurately estimate absolute temperatures in most modules.

4) BER VARIATIONS IN DIFFERENT DRAM REGIONS

We conclude that the attacker can use any region in memory to perform the attack, which simplifies the requirements of the attack (i.e., no need for reverse engineering the physical location of the memory region to find an appropriate region).

5) SENSITIVITY TO DIFFERENT REGION SIZES

To improve performance and reduce the probability of the attack being detected, the attacker can reduce the size of the DRAM region used for performing the attack. Figure 7 shows the mean error of the temperature change estimated by the polynomial regression model for different region sizes from 512 rows (4MB) to 24k rows (192MB). The color of the bars represents the two evaluated threat models: 1) spying on relative temperature changes, and 2) spying on absolute temperatures.

We make three observations. First, we can reduce the region size from 24k to 2k with minor to no increase in the mean error for all cases we test. Second, for region sizes lower than 2k, the mean error increases significantly in many cases. For example, modules from Mfr. B (Module Id. 4, 5 6) show a very significant error increase in the absolute temperature estimation. Third, the mean error values show similar trends for both relative temperature changes and absolute temperature, for most modules we test. However, the error increases more quickly when reducing the region size when estimating absolute temperatures.

We conclude that the attacker can use regions of size as small as 2k rows (i.e., 16MB) for spying on temperature, while maintaining reasonable levels of accuracy.

B. CANARY CELL ANALYSIS

From Obserbation 8 and Takeaway 3 (Section IV-B), we infer that all modules we test have a large enough number of canary cells (i.e., more than one) for each temperature point, so they can be used to reliably infer the DRAM temperature.

1) CANARY CELL ACCURACY

We measure the accuracy of SpyHammer when using canary cells. Figure 8 shows the probability distribution of the temperature errors when spying on absolute temperature using canary cells.[7] We use 10 iterations of our experiments to perform the enrollment process (i.e., identify the canary cells), and another 10 different iterations of our experiments to monitor the canary cells and spy on temperature.

We make two main observations. First, canary cells can estimate the temperature accurately (i.e., no errors) with more than 25% probability, with some cases close to 50% (e.g., Module Id. 4, 8). Second, canary cells can be used to estimate temperature with less than ±5 °C error with very high probability.

[7]Similarly, canary cells can be also used to estimate relative temperature changes.

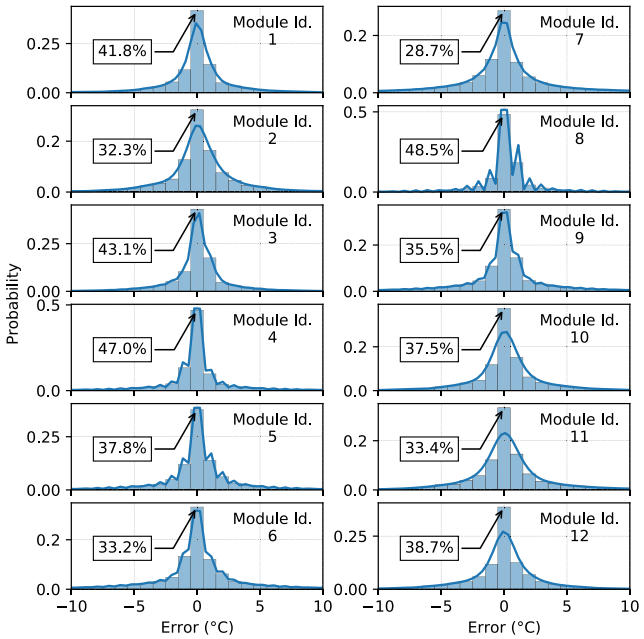


FIGURE 8. Probability distribution of the temperature estimation errors using canary cells.

We conclude that canary cells can be used to estimate temperature with significant accuracy while reducing the number of hammers significantly. For example, if an attacker wants to know if the temperature is 60 °C, it only has to monitor a canary cell that flips only at that temperature point, instead of hammering a large region of memory (see Section VII-A).

VIII. DISCUSSION AND LIMITATIONS

A. DISCUSSION

SpyHammer is the first practical attack that spies on DRAM temperature without any software or hardware modification to the victim system. As many prior works demonstrate, DRAM devices are becoming increasingly vulnerable to RowHammer [2], [77], and they can be remotely induced at system level [6], [34], without physical access, which makes SpyHammer challenging to mitigate.

Unlike other RowHammer attacks [5], [6], [7], [34], SpyHammer is much simpler to perform because of two main reasons. First, SpyHammer happens entirely in the attacker address space, not requiring to trigger bitflips in other processes, which makes the attack difficult to detect. Second, the attacker does not require performing complex memory templates, and it does *not* depend on the memory allocator of the operating system [7]. As we demonstrate in Section VII-A, the correlation between *BER* and temperature is very similar across different regions of the same DRAM module, thus SpyHammer also does not need to understand the exact physical location of the allocated memory.

For these reasons, we believe that SpyHammer is a simple practical attack that can compromise the security and privacy of any system that uses modern DRAM modules.

B. LIMITATIONS

SpyHammer has one main limitation. To spy on absolute temperature changes, the attacker needs to characterize the victim DRAM module before performing the attack, in an environment controlled by the attacker with precise temperature control. Without previous access to the victim DRAM device, we can only infer relative temperature changes over time. This is also the case for any mechanism proposed by a previous work [98]. The reason of this limitation is that each DRAM chip presents unpredictable variation caused by process variation, even if we compare it with other modules from the same manufacturer and technology node. For example, we find that the *BER* of two modules from the same manufacturer and technology node can differ up to a factor of 2.5× (see Figure 2).

IX. COUNTERMEASURES

SpyHammer can spy on DRAM temperature without any modification to the victim system. The correlation between RowHammer-induded errors and temperature is inherent to the DRAM device. There are two types of countermeasures against SpyHammer.

First, general RowHammer defense mechanisms that prevent against RowHammer bit flips, independently of the temperature. A good RowHammer defense mechanism that can mitigate most RowHammer bit flips would be also effective to prevent SpyHammer. There is an emerging body of work that provides efficient RowHammer defense mechanisms [1], [19], [51], [81], [90], [97], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135] that can also be used to mitigate SpyHammer. For example, BlockHammer [97] selectively throttles memory accesses that could otherwise potentially cause RowHammer bitflips. The key idea of BlockHammer is to 1) track row activation rates using area-efficient Bloom filters, and 2) use the tracking data to ensure that no row is ever activated rapidly enough to induce RowHammer bitflips.

Second, specific RowHammer defense mechanisms that obfuscate the relation between *BER* and temperature by introducing a temperature-dependent parameters in the mechanism.

Thus, we conclude that SpyHammer should be mitigated with general and effective RowHammer mitigation mechanisms. To date, existing DRAM modules still do not employ such effective techniques that are proven to be secure [2], [3], [31], [77], [86].

X. RELATED WORK

SpyHammer is the first attack that can remotely spy on temperature without compromising the victim system (i.e., no software or hardware modification to the victim system).

A. ATTACKS THAT SPY ON TEMPERATURE

Xiong et al. [98] proposes an attack that spies on DRAM temperature by using the observation that when the temperature increases [36], the leakage of the DRAM cells also increases, thus retention errors are manifested earlier. The main limitation of this attack is that, to observe bit errors caused by retention failures, the cell needs to be discharged for many seconds. This work has two main limitations. First, the technique requires disabling DRAM refreshes in the victim system, which can corrupt the data in critical data structures of the operating system, which would break the system. Second, disabling refreshes requires modifications to the victim system, which in turn requires the attacker to have physical access to the device before conducting the attack. Compared to Xiong et al. [98], SpyHammer can spy on temperature without any hardware or software modifications to the victim system. Also, SpyHammer does not require to disable refreshes, which enables remotely spying on temperature without corrupting the data of other processes running on the same system.

B. OTHER ROWHAMMER ATTACKS

Many prior works exploit RowHammer to perform system-level attacks [4], [5], [6], [7], [9], [17], [18], [22], [24], [25], [30], [31], [32], [33], [82], [83], [84], [85], [136], [137]. These attacks can perform denial of service [17], [18], privilege escalation [4], [5], [6], [7], [9], [17], [18], [22], [31], [82], secret data leakage [25], [32], [33], [137], manipulation of the application correctness [24], [30] or recovering private keys [84], [85]. Compared to these existing works, SpyHammer is a much less intrusive attack, as it does not involve manipulating data from any other process other than the attacker process.

C. CHARACTERIZATION OF REAL DRAM CHIPS

There are several works that extensively characterize RowHammer using real DRAM chips [1], [2], [77], [78], [80], [138], [139], [140], [141]. The first RowHammer work [1] that that investigates the vulnerability in detail for the first time 1) analyzes 129 commodity DDR3 DRAM modules, 2) characterizes the sensitivity of RowHammer to refresh rate, activation rate, and the physical distance between aggressor and victim rows, and 3) analyzes several potential solutions. The second extensive RowHammer characterization [2], conducted in 2020, analyzes RowHammer scalability by performing experiments on 1580 DDR3, DDR4, and LPDDR4 commodity DRAM chips from different DRAM generations and technology nodes, demonstrating that RowHammer has become more problematic over time. The third work [77], conducted in 2021, studies the sensitivity of RowHammer to DRAM chip temperature, aggressor row active time, and victim DRAM cell's physical location, by performing experiments on 248 DDR4 and 24 DDR3 modern DRAM chips from four major manufacturers.

Even though these works rigorously characterize various RowHammer aspects, they do not analyze the effects of temperature in great detail: 1) they do not perform fine-grained analyses (e.g., 1 °C steps), 2) they do not empirically analyze their observations from the attacker perspective in great detail, and 3) they do not build and experimentally demonstrate new RowHammer attacks based on the correlation between RowHammer vulnerability and temperature.

D. PUFS AND PHYSICAL CRYPTOGRAPHY

Our work also relates to the recent areas of physical unclonable functions (PUFs) and physical cryptography. In these areas, the intrinsic, physical characteristics of hardware are employed to either enable new cryptographic and security schemes, or to launch new classes of attacks. Optical PUFs and digital silicon PUFs have been pioneered early on by Pappu et al. and Gassend et al. [142], [143], and have been advanced in a large number of follow-up works [144], [145], [145], [146], [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167]. Also, DRAM PUFs and Rowhammer PUFs have been proposed recently [50], [168], [169], [170], [171].

XI. SUMMARY AND CONCLUSION

Recent studies demonstrate that new DRAM devices are becoming increasingly more vulnerable to RowHammer, and many works demonstrate system-level attacks for privilege escalation or information leakage. In this work, we provide the first analysis of RowHammer under fine-grained temperature variations, yielding nine key observations. We leverage our empirical observations to spy on DRAM temperature. We build a new RowHammer attack, called SpyHammer, that spies on the temperature of critical systems such as industrial production lines, self-driving or semi-automated vehicles, and medical systems, without any modification to the victim system. We propose two variants of SpyHammer for two different threat models. First, if the attacker cannot characterize the victim DRAM module before the attack, they can use SpyHammer to spy on *relative temperature changes* on the victim system. Second, if the attacker can characterize the victim DRAM module before the attack, they can use SpyHammer to spy on *absolute temperature*. Our evaluation shows that SpyHammer can 1) spy on relative temperature changes with an error of ± 3.5 °C, and 2) spy on absolute temperature changes with an error of ± 2.5 °C, for all 12 DRAM modules from four manufacturers, at the 90^{th} percentile of tested temperature points.

We conclude that SpyHammer is a simple and effective attack that can spy on temperature of critical systems with no modifications or prior knowledge about the victim system. We believe that SpyHammer can be a potential threat to the security and privacy of systems until a definitive and completely-secure RowHammer defense mechanism is adopted, which is a large challenge given that RowHammer vulnerability continues to worsen with technology scaling.

ACKNOWLEDGMENT

The authors thank the SAFARI Research Group members for useful feedback and the stimulating intellectual environment

they provide. They also acknowledge the generous gifts provided by their industrial partners, including Google, Huawei, Intel, Microsoft, and VMware, and support from the Microsoft Swiss Joint Research Center.

REFERENCES

[1] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proc. ISCA*, 2014.

[2] J. S. Kim, M. Patel, A. G. Yağlıkçı, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "Revisiting RowHammer: An experimental analysis of modern devices and mitigation techniques," in *Proc. ISCA*, 2020, pp. 638–651.

[3] P. Frigo, E. Vannacc, H. Hassan, V. v. der Veen, O. Mutlu, C. Giuffrida, H. Bos, and K. Razavi, "TRRespass: Exploiting the many sides of target row refresh," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 747–762.

[4] M. Seaborn and T. Dullien, "Exploiting the DRAM RowHammer bug to gain kernel privileges," *Black Hat*, vol. 15, p. 71, Mar. 2015.

[5] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic RowHammer attacks on mobile platforms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016.

[6] D. Gruss, C. Maurice, and S. Mangard, "RowHammer.Js: A remote software-induced fault attack in Javascript," 2015, *arXiv:1507.06955*.

[7] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a needle in the software stack," in *Proc. USENIX Secur.*, 2016, pp. 1–18.

[8] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM addressing for cross-CPU attacks," in *Proc. USENIX Secur.*, 2016, pp. 565–581.

[9] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation," in *Proc. USENIX Secur.*, 2016, pp. 19–35.

[10] E. Bosman, K. Razavi, H. Bos, and C. Giuffrida, "Dedup est machina: Memory deduplication as an advanced exploitation vector," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 987–1004.

[11] S. Bhattacharya and D. Mukhopadhyay, "Curious case of RowHammer: Flipping secret exponent bits using timing analysis," in *Proc. CHES*, 2016, pp. 602–624.

[12] R. Qiao and M. Seaborn, "A new approach for RowHammer attacks," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 161–166.

[13] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-bomb: Locking down the processor via RowHammer attack," in *Proc. 2nd Workshop Syst. Softw. Trusted Execution*, Oct. 2017, pp. 1–6.

[14] M. T. Aga, Z. B. Aweke, and T. Austin, "When good protections go bad: Exploiting anti-DoS measures to accelerate RowHammer attacks," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 8–13.

[15] O. Mutlu, "The RowHammer problem and other issues we may face as memory becomes denser," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1116–1121.

[16] A. Tatar, C. Giuffrida, H. Bos, and K. Razavi, "Defeating software mitigations against RowHammer: A surgical precision hammer," in *Proc. RAID*, 2018, pp. 47–66.

[17] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of RowHammer defenses," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 245–261.

[18] M. Lipp, M. T. Aga, M. Schwarz, D. Gruss, C. Maurice, L. Raab, and L. Lamster, "Nethammer: Inducing RowHammer faults through network requests," 2018, *arXiv:1805.04956*.

[19] V. van der Veen, M. Lindorfer, Y. Fratantonio, H. P. Pillai, G. Vigna, C. Kruegel, H. Bos, and K. Razavi, "GuardION: Practical mitigation of DMA-based RowHammer attacks on ARM," in *Proc. DIMVA*, 2018, pp. 92–113.

[20] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi, "Grand pwning unit: Accelerating microarchitectural attacks with the GPU," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 195–210.

[21] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ECC memory against RowHammer attacks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 55–71.

[22] S. Ji, Y. Ko, S. Oh, and J. Kim, "Pinpoint RowHammer: Suppressing unwanted bit flips on RowHammer attacks," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Jul. 2019, pp. 549–560.

[23] O. Mutlu and J. S. Kim, "RowHammer: A retrospective," *Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 8, pp. 1555–1571, May 2019.

[24] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *Proc. USENIX Secur.*, 2019, pp. 497–514.

[25] A. Kwong, D. Genkin, D. Gruss, and Y. Yarom, "RAMBleed: Reading bits in memory without accessing them," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 695–711.

[26] L. Cojocar, J. Kim, M. Patel, L. Tsai, S. Saroiu, A. Wolman, and O. Mutlu, "Are we susceptible to RowHammer? An end-to-end methodology for cloud providers," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 712–728.

[27] Z. Weissman, T. Tiemann, D. Moghimi, E. Custodio, T. Eisenbarth, and B. Sunar, "JackHammer: Efficient RowHammer on heterogeneous FPGA-CPU platforms," 2020, *arXiv:1912.11523*.

[28] Z. Zhang, Y. Cheng, D. Liu, S. Nepal, Z. Wang, and Y. Yarom, "PThammer: Cross-user-kernel-boundary RowHammer through implicit accesses," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 28–41.

[29] SAFARI Research Group. *RowHammer—GitHub Repository*. Accessed: Jan. 1, 2024. [Online]. Available: https://github.com/CMU-SAFARI/RowHammer

[30] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *Proc. USENIX Security*, 2020, pp. 1463–1480.

[31] P. Jattke, V. Van Der Veen, P. Frigo, S. Gunter, and K. Razavi, "BLACKSMITH: Scalable Rowhammering in the frequency domain," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 716–734.

[32] Y. Cohen, K. S. Tharayil, A. Haenel, D. Genkin, A. D. Keromytis, Y. Oren, and Y. Yarom, "HammerScope: Observing DRAM power consumption using RowHammer," in *Proc. CCS*, 2022, pp. 547–561.

[33] Y. Tobah, A. Kwong, I. Kang, D. Genkin, and K. G. Shin, "SpecHammer: Combining spectre and RowHammer for new speculative attacks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 681–698.

[34] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "ThRowHammer: RowHammer attacks over the network and defenses," in *Proc. USENIX ATC*, 2018, pp. 213–226.

[35] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 1–12, 2012.

[36] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 60–71, Jun. 2013.

[37] B. Keeth and R. Baker, *DRAM Circuit Design: A Tutorial, 2001*. New York, NY, USA: IEEE Press, 2001.

[38] O. Mutlu and T. Moscibroda, "Stall-time fair memory access scheduling for chip multiprocessors," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2007, pp. 146–160.

[39] T. Moscibroda and O. Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," in *Proc. USENIX Secur.*, 2007.

[40] O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2008, pp. 63–74.

[41] Y. Kim, D. Han, O. Mutlu, and M. Harchol-Balter, "ATLAS: A scalable and high-performance scheduling algorithm for multiple memory controllers," in *Proc. 16th Int. Symp. High-Perform. Comput. Archit.*, Jan. 2010, pp. 1–12.

[42] L. Subramanian, D. Lee, V. Seshadri, H. Rastogi, and O. Mutlu, "The blacklisting memory scheduler: Achieving high performance and fairness at low cost," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 8–15.

[43] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (SALP) in DRAM," *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 368–379, Sep. 2012.

[44] M. K. Qureshi, D.-H. Kim, S. Khan, P. J. Nair, and O. Mutlu, "AVATAR: A variable-retention-time (VRT) aware refresh for DRAM systems," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2015, pp. 427–437.

[45] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "ChargeCache: Reducing DRAM latency by exploiting row access locality," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 581–593.

[46] K. K. Chang, A. Kashyap, H. Hassan, S. Ghose, K. Hsieh, D. Lee, T. Li, G. Pekhimenko, S. Khan, and O. Mutlu, "Understanding latency variation in modern DRAM chips: Experimental characterization, analysis, and optimization," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modelling Comput. Sci.*, Jun. 2016, pp. 323–336.
[47] D. Lee, S. Khan, L. Subramanian, S. Ghose, R. Ausavarungnirun, G. Pekhimenko, V. Seshadri, and O. Mutlu, "Design-induced latency variation in modern DRAM chips: Characterization, analysis, and latency reduction mechanisms," in *Proc. SIGMETRICS*, 2017, pp. 1–36.
[48] K. K. Chang, A. G. Yağlıkçı, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms," *SIGMETRICS*, 2017, pp. 1–41.
[49] M. Patel, J. S. Kim, and O. Mutlu, "The reach profiler (REAPER): Enabling the mitigation of DRAM retention failures via profiling at aggressive conditions," in *Proc. ISCA*, 2017, pp. 255–268.
[50] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices," in *Proc. HPCA*, 2018, pp. 194–207.
[51] H. Hassan, M. Patel, J. S. Kim, A. G. Yağlıkçı, N. Vijaykumar, N. M. Ghiasi, S. Ghose, and O. Mutlu, "CROW: A low-cost substrate for improving DRAM performance, energy efficiency, and reliability," in *Proc. ISCA*, 2019, pp. 129–142.
[52] K. K. Chang, D. Lee, Z. Chishti, A. R. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu, "Improving DRAM performance by parallelizing refreshes with accesses," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2014, pp. 356–367.
[53] K. K. Chang, P. J. Nair, D. Lee, S. Ghose, M. K. Qureshi, and O. Mutlu, "Low-cost inter-linked subarrays (LISA): Enabling fast inter-subarray data movement in DRAM," in *Proc. HPCA*, 2016, pp. 568–580.
[54] S. Ghose, A. G. Yaglikci, R. Gupta, D. Lee, K. Kudrolli, W. Liu, H. Hassan, K. Chang, N. Chatterjee, A. Agrawal, M. O'Connor, and O. Mutlu, "What your DRAM power models are not telling you: Lessons from a detailed experimental study," in *Proc. SIGMETRICS*, 2018, pp. 288–301.
[55] H. Hassan, N. Vijaykumar, S. Khan, S. Ghose, K. Chang, G. Pekhimenko, D. Lee, O. Ergin, and O. Mutlu, "SoftMC: A flexible and practical open-source infrastructure for enabling experimental DRAM studies," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 241–252.
[56] S. Khan, D. Lee, and O. Mutlu, "PARBOR: An efficient system-level technique to detect data-dependent failures in DRAM," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2016, pp. 239–250.
[57] S. Khan, C. Wilkerson, D. Lee, A. R. Alameldeen, and O. Mutlu, "A case for memory content-based detection and mitigation of data-dependent failures in DRAM," *IEEE Comput. Archit. Lett.*, vol. 16, no. 2, pp. 88–93, Jul. 2017.
[58] S. Khan, D. Lee, Y. Kim, A. R. Alameldeen, C. Wilkerson, and O. Mutlu, "The efficacy of error mitigation techniques for DRAM retention failures: A comparative experimental study," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2014, pp. 519–524.
[59] V. Seshadri, T. Mullins, A. Boroumand, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Gather-scatter DRAM: In-DRAM address translation to improve the spatial locality of non-unit strided accesses," in *Proc. 48th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2015, pp. 267–280.
[60] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. MICRO*, 2017, pp. 273–287.
[61] J. Kim, M. Patel, H. Hassan, and O. Mutlu, "Solar-DRAM: Reducing DRAM access latency by exploiting the variation in local bitlines," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 282–291.
[62] J. S. Kim, M. Patel, H. Hassan, L. Orosa, and O. Mutlu, "D-RaNGe: Using commodity DRAM devices to generate true random numbers with low latency and high throughput," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 582–595.
[63] M. Patel, J. S. Kim, H. Hassan, and O. Mutlu, "Understanding and modeling on-die error correction in modern DRAM: An experimental study using real devices," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 13–25.
[64] M. Patel, J. S. Kim, T. Shahroodi, H. Hassan, and O. Mutlu, "Bit-exact ECC recovery (BEER): Determining DRAM on-die ECC functions by exploiting DRAM data retention characteristics," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 282–297.
[65] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-latency DRAM: A low latency and low cost DRAM architecture," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 615–626.
[66] D. Lee, L. Subramanian, R. Ausavarungnirun, J. Choi, and O. Mutlu, "Decoupled direct memory access: Isolating CPU and IO traffic by leveraging a dual-data-port DRAM," in *Proc. Int. Conf. Parallel Archit. Compilation (PACT)*, Oct. 2015, pp. 174–187.
[67] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "RowClone: Fast and energy-efficient in-DRAM bulk data copy and initialization," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2013, pp. 185–197.
[68] H. Luo, T. Shahroodi, H. Hassan, M. Patel, A. G. Yaglikci, L. Orosa, J. Park, and O. Mutlu, "CLR-DRAM: A low-cost DRAM architecture enabling dynamic capacity-latency trade-off," in *Proc. ISCA*, 2020, pp. 666–679.
[69] V. Seshadri and O. Mutlu, "In-DRAM bulk bitwise execution engine," 2019, *arXiv:1905.09822*.
[70] Y. Wang, L. Orosa, X. Peng, Y. Guo, S. Ghose, M. Patel, J. S. Kim, J. G. Luna, M. Sadrosadati, N. M. Ghiasi, and O. Mutlu, "FIGARO: Improving system performance via fine-grained in-DRAM data relocation and caching," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 313–328.
[71] L. Orosa, Y. Wang, M. Sadrosadati, J. S. Kim, M. Patel, I. Puddu, H. Luo, K. Razavi, J. Gómez-Luna, H. Hassan, N. Mansouri-Ghiasi, S. Ghose, and O. Mutlu, "CODIC: A low-cost substrate for enabling custom in-DRAM functionalities and optimizations," in *Proc. ISCA*, 2021, pp. 484–497.
[72] Y. Wang, A. Tavakkol, L. Orosa, S. Ghose, N. Mansouri Ghiasi, M. Patel, J. S. Kim, H. Hassan, M. Sadrosadati, and O. Mutlu, "Reducing DRAM latency via charge-level-aware look-ahead partial restoration," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2018, pp. 298–311.
[73] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana, "Self-optimizing memory controllers: A reinforcement learning approach," in *Proc. ISCA*, 2008, pp. 39–50.
[74] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: A high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation," in *Proc. ISCA*, 2014, pp. 349–360.
[75] H. Luo, A. Olgun, A. G. Yağlıkçı, Y. C. Tuğrul, S. Rhyner, M. B. Cavlak, J. Lindegger, M. Sadrosadati, and O. Mutlu, "RowPress: Amplifying read disturbance in modern DRAM chips," in *Proc. ISCA*, 2023, pp. 1–18.
[76] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "DRISA: A DRAM-based reconfigurable in-situ accelerator," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 288–301.
[77] L. Orosa, A. G. Yaglikci, H. Luo, A. Olgun, J. Park, H. Hassan, M. Patel, J. S. Kim, and O. Mutlu, "A deeper look into RowHammer's sensitivities: Experimental analysis of real DRAM chips and implications on future attacks and defenses," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 1182–1197.
[78] A. G. Yaglikçi, H. Luo, G. F. De Oliviera, A. Olgun, M. Patel, J. Park, H. Hassan, J. S. Kim, L. Orosa, and O. Mutlu, "Understanding RowHammer under reduced wordline voltage: An experimental study using real DRAM devices," in *Proc. 52nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2022, pp. 475–487.
[79] O. Mutlu, A. Olgun, and A. G. Yaglikci, "Fundamentally understanding and solving RowHammer," in *Proc. 28th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2023, pp. 1–8.
[80] A. Olgun, M. Osseiran, A. G. Yağlıkcı, Y. Can Tuğrul, H. Luo, S. Rhyner, B. Salami, J. G. Luna, and O. Mutlu, "An experimental analysis of RowHammer in HBM2 DRAM chips," 2023, *arXiv:2305.17918*.
[81] *DDR4 SDRAM Standard*, JEDEC Standard JESD79-4C, 2020.
[82] Z. Zhang, W. He, Y. Cheng, W. Wang, Y. Gao, D. Liu, K. Li, S. Nepal, A. Fu, and Y. Zou, "Implicit hammer: Cross-privilege-boundary RowHammer through implicit accesses," *IEEE Trans. Dependable Secure Comput.*, pp. 1–18, 2022.

[83] A. Kogler, J. Juffinger, S. Qazi, Y. Kim, M. Lipp, N. Boichat, E. Shiu, M. Nissler, and D. Gruss, "Half-Double: Hammering from the next row over," in *Proc. USENIX Secur.*, 2022, pp. 3807–3824.

[84] M. Fahr, H. Kippen, A. Kwong, T. Dang, J. Lichtinger, D. Dachman-Soled, D. Genkin, A. Nelson, R. Perlner, A. Yerukhimovich, and D. Apon, "When frodo flips: End-to-end key recovery on FrodoKEM via RowHammer," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 979–993.

[85] K. Mus, Y. Doröz, M. C. Tol, K. Rahman, and B. Sunar, "Jolt: Recovering TLS signing keys via RowHammer faults," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1719–1736.

[86] H. Hassan, Y. C. Tugrul, J. S. Kim, V. van der Veen, K. Razavi, and O. Mutlu, "Uncovering in-DRAM RowHammer protection mechanisms: A new methodology, custom RowHammer patterns, and implications," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021.

[87] A. Olgun, H. Hassan, A. G. Yaglikçi, Y. C. Tugrul, L. Orosa, H. Luo, M. Patel, O. Ergin, and O. Mutlu, "DRAM bender: An extensible and versatile FPGA-based infrastructure to easily test state-of-the-art DRAM chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 12, pp. 5098–5112, Dec. 2023.

[88] SAFARI Research Group. *DRAM-Bender—GitHub Repository*. Accessed: Jan. 1, 2024. [Online]. Available: https://github.com/CMU-SAFARI/DRAM-Bender

[89] *Xilinx Alveo U200 FPGA Board*. Accessed: Jan. 1, 2024. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/alveo/u200.html

[90] B. Aichinger, "DDR memory errors caused by row hammer," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2015, pp. 1–5.

[91] J. Tukey, *Exploratory Data Analysis*. Reading, MA, USA: Addison-Wesley, 1977.

[92] R. T. Smith, J. D. Chlipala, J. F. M. Bindels, R. G. Nelson, F. H. Fischer, and T. F. Mantz, "Laser programmable redundancy and yield improvement in a 64 K DRAM," *IEEE J. Solid-State Circuits*, vol. SSC-16, no. 5, pp. 506–514, Oct. 1981.

[93] M. Horiguchi, "Redundancy techniques for high-density DRAMs," in *Proc. 2nd Annu. IEEE Int. Conf. Innov. Syst. Silicon*, Oct. 1997, pp. 22–29.

[94] K. Itoh, *VLSI Memory Chip Design*. Springer, 2001.

[95] S. Khan, C. Wilkerson, Z. Wang, A. R. Alameldeen, D. Lee, and O. Mutlu, "Detecting and mitigating data-dependent DRAM failures by exploiting current memory content," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 27–40.

[96] A. Barenghi, L. Breveglieri, N. Izzo, and G. Pelosi, "Software-only reverse engineering of physical DRAM mappings for RowHammer attacks," in *Proc. IEEE 3rd Int. Verification Secur. Workshop (IVSW)*, Jul. 2018, pp. 19–24.

[97] A. G. Yaglikçi, M. Patel, J. S. Kim, R. Azizi, A. Olgun, L. Orosa, H. Hassan, J. Park, K. Kanellopoulos, T. Shahroodi, S. Ghose, and O. Mutlu, "BlockHammer: Preventing RowHammer at low cost by blacklisting rapidly-accessed DRAM rows," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 345–358.

[98] W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser, and J. Szefer, "Spying on temperature using DRAM," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 13–18.

[99] Apple Inc. (Jun. 2015). *About the Security Content of Mac EFI Security Update 2015-001*. [Online]. Available: https://support.apple.com/en-us/HT204934

[100] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in DRAM memories," *IEEE Comput. Archit. Lett.*, vol. 14, no. 1, pp. 9–12, Jan. 2015.

[101] K. Bains, J. Halbert, C. Mozak, T. Schoenborn, and Z. Greenfield, "Row hammer refresh command," U.S. Patent 9 117 544, Aug. 25, 2015.

[102] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "ANVIL: Software-based protection against next-generation RowHammer attacks," in *Proc. ASPLOS*, 2016, pp. 743–755.

[103] K. S. Bains and J. B. Halbert, "Distributed row hammer tracking," U.S. Patent 9 299 400, Aug. 4, 2016.

[104] K. S. Bains and J. B. Halbert, "Row hammer monitoring based on stored row hammer threshold value," U.S. Patent 9 384 821, Jul. 19, 2016.

[105] H. Gomez, A. Amaya, and E. Roa, "DRAM row-hammer attack reduction using dummy cells," in *Proc. IEEE Nordic Circuits Syst. Conf. (NORCAS)*, Nov. 2016, pp. 1–4.

[106] F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, "Can't touch this: Software-only mitigation against RowHammer attacks targeting kernel memory," in *Proc. USENIX Secur.*, 2017, pp. 117–130.

[107] M. Son, H. Park, J. Ahn, and S. Yoo, "Making DRAM stronger against row hammering," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2017, pp. 1–6.

[108] R. K. Konoth, M. Oliverio, A. Tatar, D. Andriesse, H. Bos, C. Giuffrida, and K. Razavi, "ZebRAM: Comprehensive and compatible software protection against RowHammer attacks," in *Proc. OSDI*, 2018, pp. 697–710.

[109] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Mitigating wordline crosstalk using adaptive trees of counters," in *Proc. ISCA*, 2018, pp. 612–623.

[110] E. Lee, I. Kang, S. Lee, G. Edward Suh, and J. Ho Ahn, "TWiCe: Preventing row-hammering by exploiting time window counters," in *Proc. ISCA*, 2019, pp. 385–396.

[111] I. Kang, E. Lee, and J. H. Ahn, "CAT-TWO: Counter-based adaptive tree, time window optimized for DRAM row-hammer prevention," *IEEE Access*, vol. 8, pp. 17366–17377, 2020.

[112] Y. Park, W. Kwon, E. Lee, T. J. Ham, J. Ho Ahn, and J. W. Lee, "Graphene: Strong yet lightweight row hammer protection," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 1–13.

[113] A. G. Yağlıkçı, J. S. Kim, F. Devaux, and O. Mutlu, "Security analysis of the silver bullet technique for RowHammer prevention," 2021, *arXiv:2106.07084*.

[114] F. Devaux and R. Ayrignac, "Method and circuit for protecting a DRAM memory device from the row hammer effect," U.S. Patent 10 885 966 B1, Jan. 5, 2021.

[115] J. M. You and J.-S. Yang, "MRLoc: Mitigating row-hammering based on memory locality," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[116] *High Bandwidth Memory (HBM) DRAM*, JEDEC Standard JESD235C, 2020.

[117] *DDR5 SDRAM Standard*, JEDEC Standard JESD79-5, 2020.

[118] *LPDDR5 SDRAM Standard*, JEDEC Standard JESD209-5A, 2020.

[119] Z. Greenfield and T. Levy, "Throttling support for row-hammer counters," U.S. Patent 9 251 885 B2, Feb. 2, 2016.

[120] J. Woo, G. Saileshwar, and P. J. Nair, "Scalable and secure row-swap: Efficient and safe row hammer mitigation in memory systems," in *Proc. HPCA*, 2023, pp. 374–389.

[121] J. Juffinger, L. Lamster, A. Kogler, M. Eichlseder, M. Lipp, and D. Gruss, "CSI: RowHammer—Cryptographic security and integrity against RowHammer," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1702–1718.

[122] M. Wi, J. Park, S. Ko, M. J. Kim, N. Sung Kim, E. Lee, and J. H. Ahn, "SHADOW: Preventing row hammer in DRAM with intra-subarray row shuffling," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2023, pp. 333–346.

[123] M. Marazzi, F. Solt, P. Jattke, K. Takashi, and K. Razavi, "REGA: Scalable RowHammer mitigation with refresh-generating activations," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1684–1701.

[124] R. Zhou, S. Tabrizchi, M. Morsali, A. Roohi, and S. Angizi, "P-PIM: A parallel processing-in-DRAM framework enabling row hammer protection," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.

[125] A. D. Dio, K. Koning, H. Bos, and C. Giuffrida, "Copy-on-flip: Hardening ECC memory against RowHammer attacks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023.

[126] G. Saileshwar, B. Wang, M. Qureshi, and P. J. Nair, "Randomized row-swap: Mitigating row hammer by breaking spatial correlation between aggressor and victim rows," in *Proc. 27th ACM Int. Conf. Architectural Support for Program. Lang. Operating Syst.*, Feb. 2022, pp. 1056–1069.

[127] M. Qureshi, A. Rohan, G. Saileshwar, and P. J. Nair, "Hydra: Enabling low-overhead mitigation of RowHammer at ultra-low thresholds via hybrid tracking," in *Proc. ISCA*, 2022, pp. 669–710.

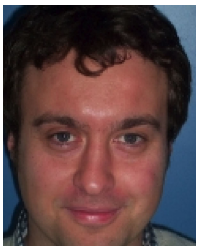
[128] M. J. Kim, J. Park, Y. Park, W. Doh, N. Kim, T. J. Ham, J. W. Lee, and J. H. Ahn, "Mithril: Cooperative row hammer protection on commodity DRAM leveraging managed refresh," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Apr. 2022, pp. 1156–1169.

[129] A. Fakhrzadehgan, Y. N. Patt, P. J. Nair, and M. K. Qureshi, "SafeGuard: Reducing the security risk from row-hammer via low-cost integrity protection," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Apr. 2022, pp. 373–386.

[130] A. Saxena, G. Saileshwar, P. J. Nair, and M. Qureshi, "Aqua: Scalable RowHammer mitigation by quarantining aggressor rows at runtime," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 108–123.

[131] B. K. Joardar, T. K. Bletsch, and K. Chakrabarty, "Machine learning-based RowHammer mitigation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 5, pp. 1393–1405, May 2023.
[132] M. Marazzi, P. Jattke, F. Solt, and K. Razavi, "ProTRR: Principled yet optimal in-DRAM target row refresh," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 735–753.
[133] K. Loughlin, S. Saroiu, A. Wolman, Y. A. Manerkar, and B. Kasikci, "MOESI-prime: Preventing coherence-induced hammering in commodity workloads," in *Proc. ISCA*, 2022, pp. 670–684.
[134] F. N. Bostanci, I. E. Yüksel, A. Olgun, K. Kanellopoulos, Y. C. Tuğrul, A. G. Yağlıçi, M. Sadrosadati, and O. Mutlu, "CoMeT: Count-min-sketch-based row tracking to mitigate RowHammer at low cost," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Mar. 2024.
[135] A. Olgun, Y. C. Tugrul, F. N. Bostanci, I. E. Yüksel, H. Luo, S. Rhyner, A. G. Yaglikçi, G. F. Oliveira, and O. Mutlu, "ABACuS: All-bank activation counters for scalable and low overhead RowHammer mitigation," in *Proc. USENIX Secur.*, 2024.
[136] I. Kang, W. Wang, J. Kim, S. van Schaik, Y. Tobah, D. Genkin, A. Kwong, and Y. Yarom, "SledgeHammer: Amplifying RowHammer via bank-level parallelism," in *Proc. USENIX*, 2024.
[137] Y. Tobah, A. Kwong, I. Kang, D. Genkin, and K. G. Shin, "Go go gadget hammer: Flipping nested pointers for arbitrary data leakage," in *Proc. USENIX*, 2024.
[138] Z. Lang, P. Jattke, M. Marazzi, and K. Razavi, "BLASTER: Characterizing the blast radius of RowHammer," in *Proc. DRAMSec*, 2023.
[139] W. He, Z. Zhang, Y. Cheng, W. Wang, W. Song, Y. Gao, Q. Zhang, K. Li, D. Liu, and S. Nepal, "WhistleBlower: A system-level empirical study on RowHammer," *IEEE Trans. Comput.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10014649
[140] A. G. Yağlıkçı, Y. C. Tuğrul, G. F. Oliveira, I. E. Yüksel, A. Olgun, H. Luo, and O. Mutlu, "Spatial variation-aware read disturbance defenses: Experimental analysis of real DRAM chips and implications on future solutions," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Mar. 2024.
[141] I. E. Yuksel, Y. C. Tugrul, A. Olgun, F. N. Bostanci, A. G. Yaglikci, G. F. de Oliveira, H. Luo, J. G. Luna, M. Sadrosadati, and O. Mutlu, "Functionally-complete Boolean logic in real DRAM chips: Experimental characterization and analysis," in *Proc. HPCA*, 2024.
[142] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002.
[143] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, Nov. 2002, pp. 148–160.
[144] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, Jun. 2007, pp. 9–14.
[145] P. Lugli, A. Mahmoud, G. Csaba, M. Algasinger, M. Stutzmann, and U. Rührmair, "Physical unclonable functions based on crossbar arrays for cryptographic applications," *Int. J. Circuit Theory Appl.*, vol. 41, no. 6, pp. 619–633, Jun. 2013.
[146] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," in *Proc. TrustCom*, 2011, pp. 33–47.
[147] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, and W. Burleson, "Power and timing side channels for PUFs and their efficient exploitation," *Cryptol. ePrint Arch.*, Jan. 2013. [Online]. Available: https://eprint.iacr.org/2013/851.pdf
[148] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Rührmair, "Application of mismatched cellular nonlinear networks for physical cryptography," in *Proc. 12th Int. Workshop Cellular Nanosc. Netw. Appl. (CNNA)*, Feb. 2010, pp. 1–6.
[149] Q. Chen, G. Csaba, X. Ju, S. B. Natarajan, P. Lugli, M. Stutzmann, U. Schlichtmann, and U. Rührmair, "Analog circuits for physical cryptography," in *Proc. ISICAS*, 2009, pp. 121–124.
[150] U. Rührmair, "SIMPL systems as a keyless cryptographic and security primitive," in *Cryptography and Security: From Theory To Applications*. Springer, 2012, pp. 329–354.
[151] M. van Dijk and U. Rührmair, "Protocol attacks on advanced PUF protocols and countermeasures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.
[152] U. Rührmair, "Physical Turing machines and the formalization of physical cryptography," *Cryptol. ePrint Arch.*, Jan. 2011.
[153] M. Sauer, P. Raiola, L. Feiten, B. Becker, U. Rührmair, and I. Polian, "Sensitized path PUF: A lightweight embedded physical unclonable function," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 680–685.
[154] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, M. Stutzmann, and U. Rührmair, "Circuit-based approaches to SIMPL systems," *J. Circuits, Syst. Comput.*, vol. 20, no. 1, pp. 107–123, Feb. 2011.
[155] Y. Gao, C. Jin, J. Kim, H. Nili, X. Xu, W. Burleson, O. Kavehei, M. van Dijk, D. C. Ranasinghe, and U. Rührmair, "Efficient erasable PUFs from programmable logic and memristors," *Cryptol. ePrint Arch.*, Jan. 2018.
[156] R. Horstmeyer, S. Assawaworrarit, U. Ruhrmair, and C. Yang, "Physically secure and fully reconfigurable data storage using optical scattering," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 157–162.
[157] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. CHES*, 2007, pp. 63–80.
[158] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–33, 2009.
[159] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *Proc. IEEE WIFS*, Dec. 2010, pp. 1–12.
[160] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "The butterfly PUF protecting IP on every FPGA," in *Proc. HOST*, 2008, pp. 67–70.
[161] U. Rührmair, "Secret-free security: A survey and tutorial," *J. Cryptograph. Eng.*, vol. 12, no. 4, pp. 387–412, Nov. 2022.
[162] G. Csaba, X. Ju, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Rührmair, "On-chip electric waves: An analog circuit approach to physical uncloneable functions," *Cryptol. ePrint Arch.*, May 2009.
[163] C. Jaeger, M. Algasinger, U. Rührmair, G. Csaba, and M. Stutzmann, "Random PN-junctions for physical cryptography," *Appl. Phys. Lett.*, vol. 96, no. 17, 2010, Art. no. 172103.
[164] U. Rührmair, Q. Chen, M. Stutzmann, P. Lugli, U. Schlichtmann, and G. Csaba, "Towards electrical, integrated implementations of SIMPL systems," in *Proc. WISTP*, 2010, pp. 277–292.
[165] U. Ruehrmair, M. Stutzmann, J. Finley, C. Jirauschek, G. Csaba, P. Lugli, E. Biebl, R. Dietmueller, K. Mueller, and H. Langhuth, "Method for security purposes," 2012. [Online]. Available: https://patentimages.storage.googleapis.com/e5/02/8d/2dfc86ef407546/US20120168506A1.pdf
[166] U. Rührmair and M. van Dijk, "On the practical use of physical unclonable functions in oblivious transfer and bit commitment protocols," *J. Cryptograph. Eng.*, vol. 3, no. 1, pp. 17–28, Apr. 2013.
[167] U. Rührmair and M. van Dijk, "Practical security analysis of PUF-based two-player protocols," in *Proc. CHES*, 2012, pp. 251–267.
[168] F. Tehranipoor, N. Karimian, K. Xiao, and J. Chandy, "DRAM based intrinsic physical unclonable functions for system level security," in *Proc. 25th Great Lakes Symp. VLSI*, May 2015, pp. 15–20.
[169] S. Sutar, A. Raha, and V. Raghunathan, "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems," in *Proc. Int. Conf. Compliers, Architectures, Sythesis Embedded Syst. (CASES)*, Oct. 2016, pp. 1–10.
[170] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Intrinsic RowHammer PUFs: Leveraging the RowHammer effect for improved security," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 1–7.
[171] N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Intrinsic run-time RowHammer PUFs: Leveraging the RowHammer effect for run-time cryptography and improved security," *Cryptography*, vol. 2, no. 3, p. 13, 2018.

LOIS OROSA (Member, IEEE) received the Ph.D. degree from the University of Santiago de Compostela, Spain, in 2013. He was a Senior Researcher with the SAFARI Research Group, ETH Zürich, Switzerland. He is currently the Director of the Galicia Supercomputing Center (CESGA). His current research interests include computer architecture, hardware security, reliability, memory systems, machine learning (ML) accelerators, and quantum computing. For more information, please see his webpage at https://loisorosa.github.io/.

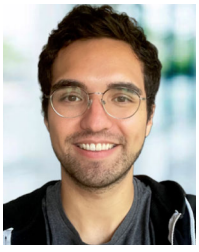


ULRICH RÜHRMAIR received the M.Sc. degree in mathematics from Oxford University, the Ph.D. degree in computer science from TU Berlin, and the Ph.D. degree in electrical engineering from TU Munich. Since 2022, he has been a Co-Speaker of the research focus on "physics and security" with the Center for Advanced Studies, LMU München. He is currently a Research Professor with the University of Connecticut. He is also a Guest Professor with LMU München. His research interests include applied cryptography and computer security in general, as well as physical unclonable functions (PUFs) and related physical security primitives in particular. He has been the Founder and the current Steering Committee Chair of the ASHES Workshop, an annual workshop at ACM CCS, since 2017. He is an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Journal of Cryptographic Engineering*, *Journal on Hardware and Systems Security*, and *EURASIP Journal on Information Security*.

A. GIRAY YAĞLIKÇI (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and the M.Sc. degree in computer engineering from the TOBB University of Economics and Technology, Ankara, Turkey, and the M.Sc. degree in computer science from the University of Notre Dame, Notre Dame, IN, USA. He is currently pursuing the Ph.D. degree with the Safari Research Group, ETH Zürich, working with Prof. Onur Mutlu. In particular, his Ph.D. research focuses on understanding and solving the RowHammer vulnerability, on which he has several publications. His research is partly supported by Google and the Microsoft Swiss Joint Research Center. His research interests include computer architecture, systems, and hardware security, with a focus on DRAM robustness and performance. He holds an Honorable Mention from the Intel Hardware Security Academic Award, in 2021. He is the First-Place Winner in the Graduate Category of the Student Research Competition in PACT, in 2023.

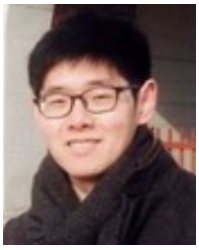
HAOCONG LUO received the B.Eng. degree in computer science from ShanghaiTech University and the M.Sc. degree in computer science from ETH Zürich, where he is currently pursuing the Ph.D. degree with the Safari Research Group, advised by Prof. Onur Mutlu. His current research interests include computer architecture, with a focus on memory systems and DRAM.

ATABERK OLGUN (Graduate Student Member, IEEE) received the master's degree in computer engineering from the TOBB University of Economics and Technology. He is currently pursuing the Ph.D. degree with ETH Zürich. His research interests include memory system reliability, safety, and performance. For more information, please see his webpage at https://aolgun.com.



PATRICK JATTKE received the B.Sc. degree in computer science from TU Darmstadt, Germany, and the M.Sc. degree in IT systems engineering from HPI, Potsdam, Germany. He is currently pursuing the Ph.D. degree with ETH Zürich, Switzerland. His main research interests include memory disturbance errors, specifically Rowhammer.

MINESH PATEL (Member, IEEE) received the dual B.S. degrees in physics and electrical engineering from UT Austin and the Ph.D. degree from ETH Zürich. His Ph.D. thesis focuses on overcoming performance, reliability, and security challenges in memory systems. In particular, his dissertation identifies and addresses new challenges for system-level error detection and mitigation targeting memory chips with integrated error correcting codes (ECC). He also worked collaboratively on understanding and solving the RowHammer vulnerability, near-data processing, efficient virtual memory management, and new hardware security primitives. His graduate work has been recognized with several honors, including the DSN 2019 and MICRO 2020 Best Paper Awards, the William Carter Dissertation Award in Dependability, the ETH Doctoral Medal, and induction into the ISCA Hall of Fame.



JEREMIE S. KIM received the B.S. and M.S. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2015, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, in August 2020. His current research interests include computer architecture, memory latency/power/reliability, hardware security, and bioinformatics, and he has several publications on these topics. His thesis won the 2020 EDAA Outstanding Dissertation Award in the area of "new directions in safety, reliability and security-aware hardware design, validation and test for systems and circuits."

KAVEH RAZAVI (Member, IEEE) is currently an Assistant Professor with ETH Zürich, where he leads the Computer Security Group. His research interests include systems and security. More recently, he has been involved in the discovery and exploitation of many high-profile hardware vulnerabilities in commodity hardware components, such as DRAM and CPU. These efforts have won him and his collaborators many awards, including the Best Paper Award from IEEE S&P and MICRO, and five Pwnies at BlackHat.

ONUR MUTLU (Fellow, IEEE) is currently a Professor of computer science with ETH Zürich. He is also a Faculty Member with Carnegie Mellon University, where he previously held the Strecker Early Career Professorship. A variety of techniques he, along with his group and collaborators, has invented over the years have influenced industry and are employed in commercial microprocessors and memory/storage systems. He started the Computer Architecture Group, Microsoft Research, from 2006 to 2009, and held various product and research positions at Intel Corporation, AMD, VMware, and Google. His research interests include computer architecture, systems, hardware security, and bioinformatics. He is an ACM Fellow and an elected member of the Academy of Europe. His computer architecture and digital design course lectures are freely available (https://www.youtube.com/OnurMutluLectures). His research group makes a wide variety of software and hardware artifacts freely available online (https://safari.ethz.ch/).