A Rowhammer Reproduction Study Using the Blacksmith Fuzzer

Lukas Gerlach, Fabian Thomas, Robert Pietsch, and Michael Schwarz

CISPA Helmholtz Center for Information Security
Saarbrücken, Saarland, Germany
<firstname>.<lastname>@cispa.de

Abstract. Rowhammer is a hardware vulnerability that can be exploited to induce bit flips in dynamic random access memory (DRAM), compromising the security of a computer system. Multiple ways of exploiting Rowhammer have been shown and even in the presence of mitigations such as target row refresh (TRR), DRAM modules remain partially vulnerable. In this paper, we present a large-scale reproduction study on the Rowhammer vulnerability using the Blacksmith Rowhammer fuzzer. The main focus of our study is the impact of the fuzzing environment. Our study uses a diverse set of 10 DRAM chips from various manufacturers, with different capacities and memory frequencies. We show that the runtime, used seeds, and DRAM coverage of the fuzzer have been underestimated in previous work. Additionally, we study the entire hardware setup's impact on the transferability of Rowhammer by fuzzing the same DRAM on 4 identical machines. The transferability study heavily relates to Rowhammer-based physically unclonable functions (PUFs) which rely on the stability of Rowhammer-induced bit flips. Our results confirm the findings of the Blacksmith fuzzer, showing that even modern DRAM chips are vulnerable to Rowhammer. In addition, we show that PUFs are challenging to achieve on commodity systems due to the high variability of Rowhammer bit flips.

1 Introduction

The Rowhammer effect was first documented in 2014 [15]. With this effect, bit flips can be induced in the DRAM by rapidly accessing adjacent memory cells in the DRAM. While first deemed unexploitable, multiple exploitation techniques [2,8,22,23,27,28,29] have been presented, making Rowhammer a threat to the security of systems. Consequently, in addition to multiple academic mitigations [15,30,25,26,31,18,20,12], industry also tried to prevent the exploitation of this effect [13]. However, some implemented mitigations have been bypassed using new hammering techniques [6,9,17,11]. Mounting Rowhammer with countermeasures such as target row refresh (TRR) or error correction code (ECC) memory in place requires sophisticated techniques often discovered using specialized fuzzers to search for hammering patterns automatically [6,3,11].

These fuzzers were evaluated on a large set of different DRAM modules (up to 42 DIMMS) where they report impressive results [6]. However, in contrast to

fuzzing papers on software [16], these results are often not reproduced. One reason is that microarchitectural attacks, particularly Rowhammer, require specific expertise [5]. There is a non-trivial configuration phase of the system. In the case of Rowhammer, it is necessary to reverse engineer the DRAM memory addressing function before an attack can be mounted [21,10,7]. As this function depends on the CPU's memory controller, this step adds additional complexity to a rowhammer attack. In addition, attack parameters that work for the specific DRAM module under test must be found. Moreover, multiple DIMMs are required for statistically significant results, making these experiments more expensive than fuzzing software. Hence, even given open-source Rowhammer fuzzers [11,6], it is not trivial to reproduce the results.

In this paper, we evaluate the *reproducibility* of Rowhammer fuzzing runs, showing that the current metric of flips in 12 h is not ideal for comparing fuzzers. For the results, we focus on the *transferability* of Rowhammer patterns across *identical* machines and *identical* DIMMs. More precisely, our analysis focuses on the following research question: *Does the same DIMM show different bit flips on different but identical machines?*

For our setup, we rely on the state-of-the-art open-source Rowhammer fuzzer Blacksmith [11]. Our analysis with 10 DIMMs on 4 *identical* machines shows that the reported results are reproducible on a wide range of commonly-available DIMMs and desktop computers without requiring any special hardware setup or software configuration. Surprisingly, our non-optimized setup discovers 5397 bit flips on average, which aligns with the original paper [11]. However, as the default configuration of the fuzzer only uses 1 GB of randomly mapped memory, the number of discovered flips in 12 h varies significantly, from 0 to 18 142 on the same DIMM. Hence, this shows that the number of bit flips in such a short time frame and over such a comparatively small memory region is not the ideal metric to compare the performance of fuzzers.

To further analyze the *transferability* of the discovered bit flips, we rely on 3 identical DIMMs and 4 identical desktop setups. We analyze both the difference in bit flip between the identical DIMMs on the same machine, and the same DIMM on identical machines. To enable such an analysis, we modify Blacksmith to deterministically map and sweep hammering patterns over a defined memory area. In line with previous work, we show that bit flips are unique to DIMMs, even if they are of the same model. However, we show that the used machine also has a significant impact. When we used a fixed 256 MB memory region and hammering pattern, in the most extreme case, we achieve on average 13 282 bit flips on 3 machines, but no bit flip on the 4th machine. This insight that even identical setups significantly impact bit flips directly affects proposals that rely on Rowhammer as a physical-unclonable function [24,1].

Contributions. The contributions of this paper are:

1. Reproducing the results of Jattke et al. [11] on 4 identical machines and 10 DIMMs, showing that current metrics for comparing Rowhammer fuzzers are not ideal.

2. Study of the transferability of bit flips between identical machines, showing that the impact of the used hardware was underestimated in previous work.
3. Evaluation of DRAM as a physically unclonable function, showing that such constructions suffer from practical issues.

Outline. Section 2 provides the necessary background. Section 3 introduces our evaluation methodology. Section 4 presents our results. Section 5 evaluates the usability of Rowhammer as a PUF. Section 6 discusses the implications of our study and potential future work. Section 7 concludes.

2 Background

In the following, we introduce the necessary background to understand the remainder of this paper.

2.1 Rowhammer

Rowhammer is a vulnerability that affects modern DRAM modules, allowing attackers to induce bit flips. Modern DRAM modules are physically organized into rows containing memory cells. Each memory cell can hold a single bit, represented by a capacitor's charge level. Because the capacitor constantly discharges, it must be regularly refreshed to prevent data loss. By design of the DRAM module, data can only be accessed in rows, discharging all capacitors in the addressed row. Therefore, after both read and write accesses, the entire row must be refilled with both the changed and unchanged data. Rowhammer attacks abuse this effect by repeatedly rapidly accessing one or more DRAM rows ("aggressor" rows) to influence neighboring DRAM rows ("victim" rows). Capacitors in neighboring rows then lose their charge more quickly, up to a point where the charge decreases below a threshold necessary to reliable detect the bit's value. This behavior allows an attacker to corrupt memory, thereby creating a potential security threat. As demonstrated by previous research [15,23,22], Rowhammer attacks can reliably flip bits at attacker-chosen addresses by accessing DRAM rows in fixed patterns. Such targeted bit flips can be used for sandbox escapes and privilege-escalation attacks on various systems, including desktop computers [15,8], mobile devices [28], and cloud servers [29], even by a remote attacker [27,19]. In an attempt to prevent Rowhammer attacks, a class of hardware mitigations called Target Row Refresh (TRR) [6] was proposed. These mitigations detect frequently-accessed rows and refresh them and other rows in proximity. While TRR prevents Rowhammer attacks relying on simple hammering patterns, it has been shown that the majority of TRR implementations is still vulnerable to more sophisticated Rowhammer attacks [6,11,17].

2.2 Blacksmith Rowhammer Fuzzer

Blacksmith [11] is a state-of-the-art Rowhammer fuzzer aiming to find memory access patterns that yield particularly high rates of Rowhammer-induced

bit flips. It improves on previous Rowhammer attacks by not only considering patterns with a uniform distribution of accesses onto the aggressor rows, but also allowing non-uniform patterns. Moreover, the fuzzer considers patterns that span over multiple refresh intervals, as these significantly improve the number of induced bit flips. Because the search space for non-uniform fuzzing patterns of varying lengths is large, Blacksmith uses a heuristic (called fuzzing in the frequency domain by the authors) to efficiently explore promising hammering patterns. This method has been shown to be highly effective in finding patterns with a high rate of Rowhammer-induced bit flips. In addition, the generated patterns have been shown to be able to bypass mitigations like TRR. TRR is bypassed by particular hammering patterns that are outside the detection scope of current hardware countermeasures [6,11,17].

2.3 Rowhammer as Physically-unclonable Function (PUF)

Rowhammer can be leveraged as a Physically-unclonable Function (PUF) [24,1], a security primitive that uses the physical variations specific to individual hardware components to generate unique and unpredictable cryptographic keys. For Rowhammer, the physical variation in DRAM cells can be used to generate a unique bit flip sequence that can be used as a basis for further cryptographic operations. As the variations in susceptibility to bit flips vary between DRAM modules, even ones from an identical manufacturing run, the bit flips of each module are unique. However, the reliability of this approach is still an active area of research, as the stability and predictability of the bit sequence can be affected by various factors, such as temperature and voltage fluctuations and the quality of the DRAM module itself. Therefore, PUFs based on Rowhammer require a fuzzy extractor construction [4] to deal with the underlying unstableness of Rowhammer bit flips. In addition, attacks against Rowhammer-based PUF constructions have been proposed [32]. It was shown that given enough observations of a PUF scheme using Rowhammer, an attacker can predict the subsequent PUF response, alwowing an attacker to imitate the PUF generated, therefore breaking its security guarantees. In addition, a denial of service attack was shown. While not directly undermining the security guarantees of Rowhammer-based PUFs, this attack renders a PUF based on Rowhammer unusable.

3 Evaluation Methodology

In this section, we describe our methodology for reproducing the Blacksmith Rowhammer fuzzer as well as the impact of the remaining computing system on the presence and frequency of Rowhammer bit flips. By reproducing the results found with the Blacksmith fuzzer, we gain valuable insight into the intricacies in the evaluation of Rowhammer fuzzers. In addition, we further characterize the distribution of Rowhammer bit flips in the tested DRAM modules. To combine both our evaluation goals, we use the Blacksmith fuzzer to find flips on 10

different DDR4 modules and repeat the process on 4 identical machines (Section 3.2). As the Blacksmith fuzzer heavily relies on randomization, it depends heavily on the desired property one wants to validate or falsify whether the data it produces is useful. We modify Blacksmith to be deterministic for reproducible runs by employing a fixed seed and ensuring that the same physical memory is mapped between runs. Based on the performed fuzzing runs, we create a data set that we also use for our analysis in Section 4.

3.1 Testing Setup

This section describes our testing setup regarding both hardware and methodology. We provide details about the machines and DRAM modules used during our evaluation and introduce the testing setup for the Blacksmith fuzzer.

Tested Machines We conduct our tests using 4 identical machines, each with an Intel Core i7 9700K CPU, HP 8591 mainboard, and HP HQ-TRE 71025 power supply. All machines run a freshly-installed stock Ubuntu 22.04 using the 5.19.0 Linux kernel. We verify that each machine uses microcode version 0xf0 from May 10, 2022, which is the default microcode with the installed Linux distribution. We reset the BIOS settings to eliminate the influence of BIOS settings on the fuzzing runs to factory default settings. In addition, all machines are placed in one room to minimize the impact of different temperature environments. In the remainder of the paper, we refer to these machines as machine $\mathcal{A}$ to machine $\mathcal{D}$.

Tested DRAM Modules We use 10 different DDR4 DRAM modules of varying memory sizes from 3 different manufacturers. The tested modules range in memory sizes from 8 GB to 32 GB. In addition, they cover a wide range of memory speeds from 2133 MHz up to 3200 MHz. The manufacturing date of the DRAM modules ranges from 2016 to 2022. Further information on the tested DRAM modules is provided in Table 1.

3.2 Fuzzing and Memory Sweep

We set up Blacksmith to run for 12 h, consistent with the evaluation by Jattke et al. [11]. For each module, we perform one run on machines $\mathcal{A}$ to $\mathcal{D}$. We label each run with a combination of DRAM module and machine, e.g., $\mathcal{A}_1$ for machine $\mathcal{A}$ and DRAM module 1. During each fuzzing run, Blacksmith searches for patterns that produce bit flips. We label a fuzzing run successful if it produces at least one bit flip. When a successful fuzzing run is found, it is followed by a memory sweep, where a randomly-mapped 256 MB memory range is tested again using the pattern that produced the most bit flips in the initial run. This configuration is consistent with the Blacksmith paper and allows for a more thorough inspection of bit flips in a smaller chunk of memory.

To study the transferability of the bit flips between the different machines, we rely on the best fuzzing patterns, i.e., the ones producing the most bit flips in our initial runs. To ensure that we always test the same physical memory cells

Table 1: Module information of our tested DIMMs.

Manufacturer	Module Id	Manufactured	Size	Ranks	Frequency	Flips Observed
A	Module 1	2018 Q2	32 GB	2	2666 MHz	✓
	Module 2	2018 Q2	32 GB	2	2666 MHz	✓
	Module 3	2018 Q2	32 GB	2	2666 MHz	✓
	Module 4	2018 Q2	32 GB	2	2666 MHz	✓
	Module 5	2021	8 GB	1	3200 MHz	✓
	Module 6	2021	8 GB	1	3200 MHz	✓
B	Module 7	2017	8 GB	2	3200 MHz	✓
	Module 8	2022	16 GB	2	3200 MHz	✓
C	Module 9	2016	8 GB	2	2133 MHz	✗
	Module 10	2016	8 GB	2	2133 MHz	✗

on the DRAM module, we modify Blacksmith to always map the same memory region when sweeping a fuzzing pattern. We test the best fuzzing patterns found on the identical modules 1, 2 and 4[1] on each of the 4 identical machines, allowing us to study the transferability of bit flips under an identical setup.

4 Results

We present the results of the fuzzing evaluation with regard to the individual runs performed with the unmodified Blacksmith fuzzer. By analyzing the distribution of the found bit flips, we gain additional insights into the bit-flip characteristic of the analyzed DRAM modules. We make several interesting observations. First, 8 out of the 10 tested modules are affected by Rowhammer bit flips on at least one of the test machines, confirming claims from prior work that most modules are affected. Second, Section 4.1 shows potential areas for improvement when using simple evaluation criteria such as the number of bit flips or bit flips over time. Third, Section 4.2 shows that the bit flips are not uniformly distributed across rows but instead localized within a smaller subset of the rows, and that bit flips with lower hamming weight are more likely, i.e., bit flips where only a few memory cells change their value. Finally, a factor that was previously not analyzed and has considerable influence is the machine used during Rowhammer testing, as we show in Section 4.3. As a result, we gain valuable insights into the specifics of Rowhammer fuzzer evaluation that show that current evaluations are not ideal for comparing fuzzers. Building on the result that Rowhammer depends on the entire underlying system, we perform a case study in Section 5 showing that its use as a PUF, as proposed by previous work [24,1], is very challenging on a commodity system.

[1] DRAM module 3 broke during testing and is therefore excluded in the transferability study.

4.1 Temporal Distribution of Bit Flips

Two standard time-based metrics when evaluating software fuzzers are crashes and coverage over time [16]. While these metrics originate from software fuzzing, they can also find application in evaluating more specialized fuzzers such as Blacksmith. However, as Blacksmith does not produce crashes, we use the number of bit flips over time reported during each fuzzing run.

A common methodology in software fuzzing is dedicating crashes to isolate a unique trigger for each crash [16]. Deduplication, when used in the practical context of vulnerability discovery, simplifies the analysis of crash root causes. Further, deduplication eases the evaluation of a software fuzzer as one crash may have many different root causes, which, when not deduplicated, can also complicate the comparison between different fuzzers. Similarly to the deduplication of crashes in software fuzzing, we also deduplicate the bit flips and count the number of unique bit flips over time. While multiple deduplication granularities are possible as the DRAM is hierarchically organized, we decided to duplicate on a per-row basis. We experimentally evaluated multiple deduplication strategies and found that a per-row deduplication is an effective middle ground allowing for interesting observations. This means that we only count the new rows where a bit flip occurs not the position inside a row. As observed in Section 4.2, up to 75 % of all memory cells are vulnerable when hammering long enough.

As seen in Figure 1, Blacksmith finds bit flips quickly. On average, the time to the first bit flip is 47.78 min, excluding runs that do not produce flips. The higher time to first bit flip is easily explained by two outlier samples where the time to the first bit flip takes multiple hours. This also reflects in the much lower median time to the first bit flip of 3.78 min. Similar differences between the time to first bit flip have been observed by Jattke et al. [11]. In addition, once the first bit flip occurred, further bit flips are consistently found. These observations highlight that Blacksmith's fuzzing strategy can consistently produce patterns that induce bit flips. Overall, a linear relationship exists between time and the number of bit flips, both when considering unique and non-unique bit flips, as seen in Figure 1. We also calculate the factor with which unique bit flips occur less than general bit flips. We observe that unique bit flips occur around 3 orders of magnitude less often than just any observed bit flip (to be more precise, general bit flips occur 1694 times more often than unique ones). While this factor is comparatively high, the high number of overall bit flips also implies a high number of unique bit flips. Even more interesting, the factor with which unique vs. non-unique bit flips occur, stabilizes quickly after starting a fuzzing run. It also remains stable during the 12 h duration of the run. The fact that this factor stays constant over the time of the fuzzing run indicates that there are still new bit flips found at the end of the run. Therefore, the 12 h duration for each fuzzing run is likely too short to exhaust all unique bit flips in the 1 GB memory area under test by fuzzing. Future work could evaluate similar fuzzers based on the time until no unique flips are found. This strategy can lead to two potentially interesting metrics.

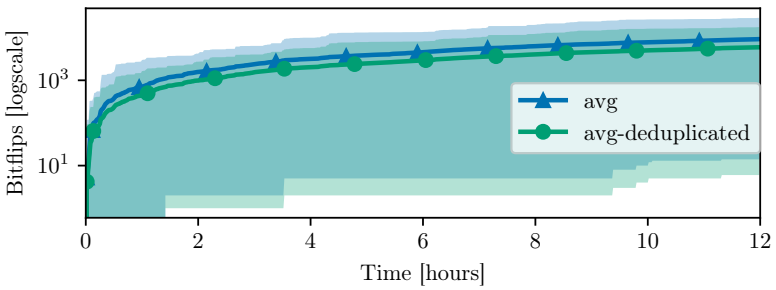

Fig. 1: Flips over time observed by the fuzzing runs. Both duplicated and non-deduplicated bit flips are plotted with the minimum and maximum flips at the current time point as well as the average.

First, the time until no more unique bit flips are found indicates how long a fuzzer needs to reach its optimal coverage, after which point fuzzing for longer produces diminishing returns. The number of unique bit flips yields insight into the maximum reachable performance of the fuzzer. Combining these metrics by dividing the total number of unique bit flips by the time needed to reach a point where little to no new bit flips are found gives a combined metric for the fuzzing performance. Intuitively, one wants many unique bit flips as fast as possible, favoring fuzzers that efficiently explore the DRAM to produce new ones. In addition, this metric enables comparing long-running fuzzers that try to scan the memory thoroughly for a maximum amount of bit flips with ones that are optimized to produce bit flips as quickly as possible.

Takeaway Simply evaluating Rowhammer fuzzers by the number of bit flips they produce in a fixed time interval oversimplifies the comparison.

Second, the coverage, again similar to classic software fuzzers, is another interesting metric. In contrast to coverage metrics from software testing, a much simpler coverage metric suffices for Rowhammer. We monitor how many of the total DRAM rows in the 1 GB memory interval are accessed by Blacksmith during its 12 h fuzzing campaign. As illustrated in Figure 2, during each of the 12 h fuzzing runs, a coverage of over 90.5 % is reached. Due to the randomized fuzzing strategy that Blacksmith employs, the coverage differs between fuzzing runs, with different runs producing different minimal and maximal coverage. We also see that the increase in global coverage slows down over time as Blacksmith randomly picks addresses in the DRAM. This leads to a situation where the more prolonged the fuzzing is, the more the DRAM has been covered, and the less likely it is to uncover new DRAM locations randomly. This observation can be valuable if a fuzzer should reach maximal global coverage quickly, in which case the design employed by Blacksmith is not ideal. Instead, already

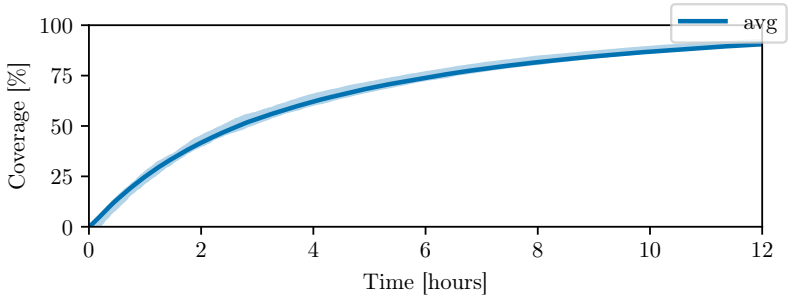

Fig. 2: Coverage of the 1 GB memory region over the time of the 12 h fuzzing run. The margins show the minimum and maximum coverage achieved over all runs at each point in time.

tested DRAM locations could be blocklisted in favor of new untested locations, therefore enabling rapid exploration of the provided memory.

Takeaway Coverage of current Rowhammer fuzzers can be improved by avoiding already fuzzed DRAM cells.

4.2 Spatial Distribution of Bit Flips

In addition to the temporal resolution, i.e., how long it takes to find bit flips, the spatial distribution of the bit flips is a relevant metric. An important starting point is the distribution of Rowhammer-susceptible DRAM rows over the analyzed DRAM module.

We compute the distribution of bit flips among the tested DRAM rows. In the interest of keeping the results readable, we compute a histogram over the rows where we group 256 rows into one bucket. The results of this evaluation are shown in Figure 3, where it can be observed that the bit flips are not uniformly distributed across the DRAM rows but are concentrated in a small subset of rows. In addition, we observe that this effect is consistent over multiple machines and DRAM modules as well as over the random memory ranges picked by Blacksmith for each fuzzing run. As we observe this effect on independent fuzzing runs, we hypothesize it is more a property of the DRAM itself and not one of the fuzzing patterns generated by Blacksmith. In addition, we observe the same effect even if the memory ranges used for fuzzing reside at a fixed physical address, implying that this effect is not due to the specifics of address allocation from the OS. Our current hypothesis is that the physical memory layout of the DRAM chip is reflected in the observed bit flips. As the physical layout does not need to match the physical addresses, we assume that a more vulnerable DRAM integrated circuit on the DRAM module is indexed at regular intervals yielding the observed patterns. While this effect has to be better understood to determine its root

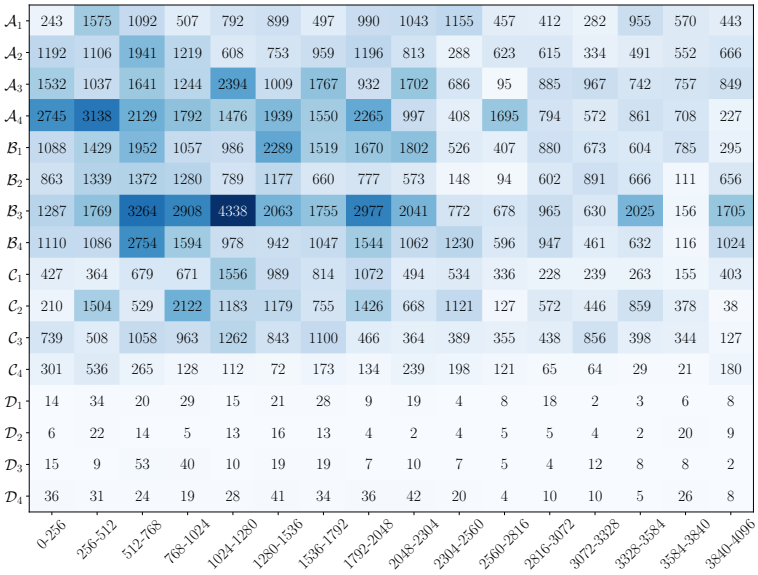

Fig. 3: Accumulated bit flips per row range on $\mathcal{A}_1$ to $\mathcal{D}_4$ averaged over all banks. It can be seen that flips are not evenly distributed across rows.

cause, we can see potential consequences. A fuzzer searching for Rowhammer bit flips could employ a search strategy favoring memory regions where bit flips already occurred to target the localized bit flips. In addition, an attacker can employ a similar strategy to scan the memory for the required bit flip patterns.

Takeaway Bit flips are not uniformly distributed across DRAM rows. Instead, they are local, mostly occurring in a smaller subset of the rows.

In addition to analyzing the distribution of bit flips across different DRAM rows, we analyze the Hamming weight of bit-flip patterns induced during each fuzzing run. A pattern is a 64-bit bitmask representing the difference between the original and flipped data, i.e., the location of bit flips. Figure 4 shows the results of summing up the occurrences of Hamming weights per machine-DRAM pair. Note that we accumulate the data of all patterns by summing the Hamming weights of the patterns. While most patterns have a Hamming weight of 1, we find up to 5 bit flips per quadword, i.e., a Hamming weight of 5, on $\mathcal{C}_1$and $\mathcal{C}_4$. This is in line with work by Kim et al. [15,14]. They report up to 4 bit flips. Further, we analyze the maximum flip rate per row. We find up to 364 bit flips per row (8 KiB = 65536 bits) on $\mathcal{C}_4$ with pattern $\mathcal{A}$, giving a maximum flip rate of 0.56 %. This is again in line with Kim et al. [14] who report a maximum flip rate between 0.1 % and 1 % for new DDR4 chips. When analyzing the distribution of bits in the bit-flip patterns, we find that the flipped bits are uniformly distributed in a

Fig. 4: Distribution of Hamming weights per quadword on machine-DRAM combinations accumulated over all patterns that produce bit flips. The distribution is close to a normal distribution with a mean at around 32.

quadword. This aligns with results by Gruss et al. [8], who report that bit-flip locations are uniformly distributed when averaging over multiple pages.

Takeaway In rare cases, Blacksmith finds patterns that induce up to 5 bit flips per quadword. Further, Blacksmith reaches a maximum flip rate of 0.56 % per hammering run. As reported by related work, the bit-flip locations are uniformly distributed in a quadword.

4.3 Transferability

A key goal of our study is to analyze the transferability of Rowhammer bit flips on the same DRAM module but on different, identical machines. For this, we analyze the observed bit flips induced by a hammering pattern between different machines. The vital difference to the transferability study of Jattke et al. [11] is that we examine the transferability of hammering patterns between machines and not DRAM modules.

We perform the memory sweep on all machines with the best pattern found for the DRAM module on one machine. Each pattern is replayed over the *exact same* physical memory range in the *exact same* order. We take the above steps to eliminate side effects caused by the choice of memory region. As seen in Figure 3, bit flips are not uniformly distributed across rows. Therefore, picking a different memory and row range could bias the results.

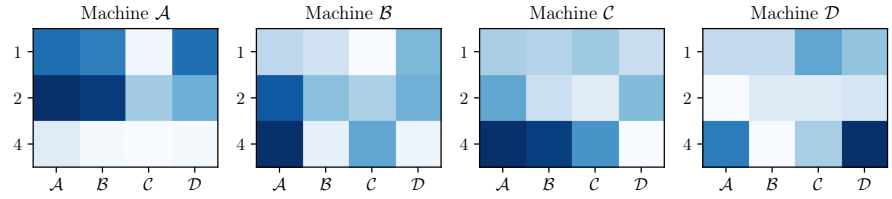

Fig. 5: Transferability of patterns between different machines. Each heatmap illustrates fuzzing runs on one machine over the 3 tested DIMMs with patterns found on all other machines. Darker colors indicate more observed bit flips.

Figure 5 illustrates our results. We observe that the occurrence and count of bit flips depend on the machine. In the most extreme cases, this can lead to cases where we do observe bit flips on one machine but not the other using the same hammering pattern and DRAM module. We suspect that manufacturing differences in non-DRAM parts of the system could induce these differences. One example could be the different frequency scaling behavior of CPUs depending on minor manufacturing differences. Similar differences could also influence the mainboard components and power supply, leading to a different number of observed bit flips.

Takeaway The combination of the machine, hammering pattern and DRAM module impacts the presence and count of bit flips.

5 Case Study: Rowhammer as Physically Unclonable Function

In this section, we further quantify the extent to which the machine in that a DRAM module is installed influences the observed bit flips. This effect directly impacts not only the evaluation of Rowhammer fuzzers but also the use of DRAM as a physically-unclonable function (PUF) [24,1]. Assuming that the bit flips are randomly distributed and dependent on manufacturing differences unique to each DRAM module, a random and unique bit stream can be obtained. This bit stream can then be used as an entropy source for further operations, such as serving as a seed in cryptographic operations. We experimentally quantify the additional error induced by manufacturing differences in the same model of different CPUs.

Schaller et al. [24] conclude that an error rate of 5 % achieved in their experiments is acceptable when using a fuzzy extractor construction [4]. To stay comparable to the work of Schaller et al. [24], we also use the Jaccard index as a measurement of error. The Jaccard index J can be computed between two sets of measurements H_1 and H_2, obtained during two hammering runs, using the formula $J = \frac{|H_1 \cap H_1|}{|H_1 \cup H_2|}$. We store the rows index of the rows that produced

d_J	Combination	Run A	Run B
0.18		1	2
0.11	$\mathcal{A}_1$	1	3
0.16		2	3
0.78		1	2
1.00	$\mathcal{A}_2$	1	3
1.00		2	3
0.39		1	2
0.35	$\mathcal{A}_4$	1	3
0.18		2	3
1.00		1	2
0.30	$\mathcal{B}_1$	1	3
1.00		2	3
0.47		1	2
0.33	$\mathcal{B}_2$	1	3
0.43		2	3
1.00		1	2
1.00	$\mathcal{B}_4$	1	3
1.00		2	3

d_J	Combination	Run A	Run B
1.00		1	2
0.18	$\mathcal{C}_1$	1	3
1.00		2	3
0.39		1	2
0.35	$\mathcal{C}_2$	1	3
0.41		2	3
1.00		1	2
1.00	$\mathcal{C}_4$	1	3
0.35		2	3
1.00		1	2
0.82	$\mathcal{D}_1$	1	3
1.00		2	3
0.78		1	2
0.53	$\mathcal{D}_2$	1	3
0.76		2	3
1.00		1	2
0.50	$\mathcal{D}_4$	1	3
1.00		2	3

Fig. 6: Comparison of three deterministic runs (same machine, same DRAM, same pattern) on machines $\mathcal{A}$-$\mathcal{D}$. The jackard index d_j shows how consistent the runs are, with $d_j = 0$ we observe the same bitflips between two different runs.

bit flips in these sets H_1 and H_2. We focus on the error rate between different machine-DRAM pairs, which is the Jaccard distance computed as $d_J = 1 - J$, where J is the Jaccard index between two fuzzing runs using the same pattern and memory range.

5.1 Reliability

To ensure that machine-DRAM pairs can be used as PUF, we analyze 3 repeated fuzzing runs on the same machine-DRAM pair. Figure 6 illustrates the results. For $\mathcal{A}$, both DRAM modules 1 and 4 give low error rates between the 3 runs, while DRAM 2 has a high error rate. Consequently, modules 1 and 4 could be used as PUF with $\mathcal{A}$, while module 2 cannot be used. On $\mathcal{B}$ to $\mathcal{D}$, we see contrary results. While DRAM module 2 performs best, both modules 1 and 4 cannot be used on any of the 3 machines. On $\mathcal{D}$ we even see that no module works reliably, with error rates above 50 %.

We conclude that DRAM modules may work perfectly with one machine while inducing high error rates on another machine. Further, we conclude that some machines induce bit flips less reliably than others and, therefore, are not suitable for usage as PUFs. Additionally, we conclude that multiple hammering runs are needed for reliable results as can be seen with the runs on $\mathcal{C}_1$, where two runs compared well, while the others had no common flipped rows. Multiple reasons can potentially explain the high error rates on some machines:

– **More diverse hardware**: Our experiments are performed on DRAM modules from multiple vendors. In addition, additional variations could be introduced as we use commodity computers instead of a specialized testing setup.

– **System noise**: We perform our experiments under a normal Linux setup introducing additional unrelated noise on the DRAM modules.
– **Hammering patterns**: We use complex hammering patterns generated by Blacksmith, which could introduce additional noise.

Quantifying if and to which extent the above points change the occurrence of bit flips requires further studies. Especially the relationship between the complexity of a hammering pattern and the determinism of the induced bit flips is important for both the usage of Rowhammer as a PUF and also the reliability of Rowhammer exploits.

Takeaway System noise, hardware differences, and different hammering patterns make it unlikely to trigger identical bit flips reliably on the same DIMM.

5.2 Uniqueness

In this section, we investigate on which parameters the uniqueness of Rowhammer as PUF depends. For that, we evaluate both pairs of the same machine and differing DRAM and differing machine and the same DRAM.

Table 2 gives an overview of the best (lowest error rate) 15 pairs of combinations of machine, DRAM, and pattern. While most pairs result in error rates above 40 %, we find 3 cases with lower error rates. In all of these cases, both memory sweeps that are compared are performed with the same DRAM module, only the machine changes. Further, we see no cases of low error rates when using a different machine as well as a different DRAM module from our test pool. From these observations, we draw the following conclusions:

1. **DRAM has more impact on the error rate than the machine**: The pairs with the lowest error rate all have in common that they share the same DRAM module. Thus, we conclude that DRAM has a higher impact on a low error rate than the machine. This further leads us to the next conclusion.
2. **Uniqueness is guaranteed when DRAM is secret, i.e., not shared**: We see no pair of machine-DRAM combinations with error rates below 50 %, where only the machine matches. Thus, we argue that a secret DRAM module, i.e., a non-shared one, already fulfills the uniqueness guarantees of a PUF. Section 5.1 in contrast shows that the machine has a high impact on the reliability and applicability of Rowhammer as PUF.

Takeaway PUFs with Rowhammer are difficult to achieve on commodity systems. While there are no false positives (no combination of DRAM and machines is the same), the false positive rate is high as it heavily depends on the machine (minor change in the environment leads to non-reproducible output of the PUF).

d_J	Combination A	Combination B	Same machine	Same DRAM	Pattern
0.21	$\mathcal{B}_1$	$\mathcal{C}_1$	✗	✓	$\mathcal{B}$
0.29	$\mathcal{A}_1$	$\mathcal{B}_1$	✗	✓	$\mathcal{A}$
0.35	$\mathcal{A}_2$	$\mathcal{B}_2$	✗	✓	$\mathcal{B}$
0.40	$\mathcal{B}_2$	$\mathcal{C}_2$	✗	✓	$\mathcal{C}$
0.42	$\mathcal{A}_2$	$\mathcal{C}_2$	✗	✓	$\mathcal{A}$
0.43	$\mathcal{A}_4$	$\mathcal{C}_4$	✗	✓	$\mathcal{A}$
0.52	$\mathcal{A}_2$	$\mathcal{C}_2$	✗	✓	$\mathcal{C}$
0.55	$\mathcal{A}_2$	$\mathcal{C}_1$	✗	✗	$\mathcal{C}$
0.59	$\mathcal{B}_2$	$\mathcal{C}_1$	✗	✗	$\mathcal{C}$
0.60	$\mathcal{B}_4$	$\mathcal{C}_4$	✗	✓	$\mathcal{C}$
0.62	$\mathcal{C}_1$	$\mathcal{C}_2$	✓	✗	$\mathcal{C}$
0.62	$\mathcal{C}_1$	$\mathcal{C}_2$	✓	✗	$\mathcal{C}$
0.62	$\mathcal{C}_2$	$\mathcal{C}_1$	✓	✗	$\mathcal{C}$
0.62	$\mathcal{C}_2$	$\mathcal{C}_1$	✓	✗	$\mathcal{C}$
0.67	$\mathcal{C}_4$	$\mathcal{D}_4$	✗	✓	$\mathcal{D}$

Table 2: The 15 combinations of machine, DRAM, and pattern, whith the lowest error rate. At least one of the three parameters is different in between the runs. We compute the jackard distance d_j for each of the tested combinations.

6 Discussion

In this section we discuss the implications of our work and the potential for future work building on our research.

6.1 Implications

In this paper we reproduced the results presented in the Blacksmith paper and quantified the impact of hardware other than the DRAM module on the presence of Rowhammer bit flips. Our results show that Blacksmith effectively finds bit flips in 8 of the 10 tested DRAM modules. However, our results also show that the hardware, beyond the DRAM module, significantly impacts the occurrence and distribution of bit flips. These results do not only have implications on the evaluation of Rowhammer fuzzers but also on the usage of DRAM as a PUF. We show that depending on the machine, even if they have identical hardware, the change in bit flips and therefore the error rate of a Rowhammer-based PUF can reach over 80 %.

Our results confirm the ongoing threat of Rowhammer as an attack vector. The fact that Blacksmith can find bit flips in 8 of the 10 tested DRAM modules again underlines the need for effective Rowhammer countermeasures. Our results also show that the spatial distribution of bit flips is an important metric to consider when evaluating a Rowhammer attack's effectiveness.

6.2 Future Work

Future work could include evaluating the performance of Blacksmith against other Rowhammer fuzzers. Our study shows that Blacksmith is effective in finding bit flips in various DRAM modules, but it is unclear how it compares to other search strategies employed in other fuzzers. Our analysis also shows that a simple metric such as bit flips in a 12 h run can be misleading when comparing Rowhammer fuzzers. Therefore, future work should explore methods to fairly and reliably compare fuzzers, which is challenging given that the amount of Rowhammer bit flips strongly depends on the tested DRAM module.

Another avenue for future research is to analyze the impact of further factors, such as specific CPU and mainboard features, on the occurrence and distribution of bit flips. In addition, a study with more diverse hardware and uniform DRAM modules could be performed, further quantifying the impact of hardware other than the DRAM module on Rowhammer. This information could aid the design of better Rowhammer countermeasures and more effective Rowhammer attacks.

Lastly, fundamental observations regarding the Rowhammer effect could be further quantified. We observed that the bit flips found by Blacksmith were not uniformly distributed across DRAM. In addition, further characterizing the bit flip patterns found by Blacksmith beyond the hamming weight of the bit flips could reveal insights about the distribution of bit flips inside a single row.

7 Conclusion

In this paper, we presented a reproduction study on the Rowhammer security vulnerability using the Blacksmith Rowhammer fuzzer. We also investigated the impact of the entire hardware setup on the transferability of Rowhammer flips across different but identical machines. With 8 out of 10 DDR4 DIMMs showing bit flips, the findings revealed that the Rowhammer vulnerability is still prevalent in modern DRAM chips and could be easily reproduced using the correct tools. Additionally, identical machines were utilized to determine the transferability and variability of the Rowhammer effect between different system setups. This transferability study was closely related to Rowhammer-based physically unclonable functions (PUF), which depend on the stability of Rowhammer-induced bit flips. In conclusion, the results confirmed the findings of the Blacksmith fuzzer, demonstrating that even modern DRAM chips were vulnerable to Rowhammer. However, we also showed that the fair comparison of Rowhammer fuzzers is difficult due to the hardware-dependent fuzzing environment. Finally, our results indicate that achieving PUF on commodity systems is challenging due to the high variability of Rowhammer bit flips.

Acknowledgment

We thank our anonymous reviewers for their valuable feedback. We thank Michele Marazzi for fruitful discussions.

References

1. N. A. Anagnostopoulos, T. Arul, Y. Fan, C. Hatzfeld, A. Schaller, W. Xiong, M. Jain, M. U. Saleem, J. Lotichius, S. Gabmeyer *et al.*, "Intrinsic run-time row hammer pufs: Leveraging the row hammer effect for run-time cryptography and improved security," *Cryptography*, 2018.
2. E. Bosman, K. Razavi, H. Bos, and C. Giuffrida, "Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector," in *S&P*, 2016.
3. L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks," in *S&P*, 2019.
4. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *EUROCRYPT*, 2004.
5. C. Easdon, M. Schwarz, M. Schwarzl, and D. Gruss, "Rapid Prototyping for Microarchitectural Attacks," in *USENIX Security*, 2022.
6. P. Frigo, E. Vannacci, H. Hassan, V. van der Veen, O. Mutlu, C. Giuffrida, H. Bos, and K. Razavi, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P*, 2020.
7. L. Gerlach, S. Schwarz, N. Faroß, and M. Schwarz, "Efficient and Generic Microarchitectural Hash-Function Recovery," in *S&P*, 2024.
8. 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 *S&P*, 2018.
9. H. Hassan, Y. Can Tuğrul, 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 *IEE MICRO*, 2021, extended classification tree and PoCs at https://transient.fail/.
10. C. Helm, S. Akiyama, and K. Taura, "Reliable reverse engineering of intel dram addressing using performance counters," in *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2020.
11. P. Jattke, V. van der Veen, P. Frigo, S. Gunter, and K. Razavi, "Blacksmith: Scalable rowhammering in the frequency domain," in *S&P*, 2022.
12. J. Juffinger, L. Lamster, A. Kogler, M. Eichlseder, M. Lipp, and D. Gruss, "Csi: Rowhammer-cryptographic security and integrity against rowhammer," in *IEEE S&P*, 2022.
13. M. Kaczmarski, "Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – component selection guidelines," August 2014. [Online]. Available: http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf
14. J. S. Kim, M. Patel, A. G. Yaglikçi, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," *ISCA*, 2020.
15. 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 *ISCA*, 2014.
16. G. Klees, A. Ruef, B. Cooper, S. Wei, and M. Hicks, "Evaluating fuzz testing," in *SIGSAC*, 2018.
17. 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 *USENIX Security Symposium*, 2022.

18. E. Lee, I. Kang, S. Lee, G. E. Suh, and J. H. Ahn, "Twice: Preventing row-hammering by exploiting time window counters," in *ISACA*, 2019.
19. M. Lipp, M. T. Aga, M. Schwarz, D. Gruss, C. Maurice, L. Raab, and L. Lamster, "Nethammer: Inducing Rowhammer Faults through Network Requests," in *SILM Workshop*, 2020.
20. Y. Park, W. Kwon, E. Lee, T. J. Ham, J. H. Ahn, and J. W. Lee, "Graphene: Strong yet lightweight row hammer protection," in *MICRO*, 2020.
21. P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," in *USENIX Security Symposium*, 2016.
22. R. Qiao and M. Seaborn, "A New Approach for Rowhammer Attacks," in *International Symposium on Hardware Oriented Security and Trust*, 2016.
23. K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip feng shui: Hammering a needle in the software stack," in *USENIX Security Symposium*, 2016.
24. 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 *Hardware Oriented Security and Trust (HOST*, 2017.
25. S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Mitigating wordline crosstalk using adaptive trees of counters," in *ISCA*. IEEE, 2018.
26. M. Son, H. Park, J. Ahn, and S. Yoo, "Making dram stronger against row hammering," in *DAC*, 2017.
27. A. Tatar, R. Krishnan, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer Attacks over the Network and Defenses," in *USENIX ATC*, 2018.
28. 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 *CCS*, 2016.
29. Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation," in *USENIX Security Symposium*, 2016.
30. A. G. Yağlıkçı, M. Patel, J. S. Kim, R. Azizibarzoki, A. Olgun, L. Orosa, H. Hassan, J. Park, K. Kanellopoullos, T. Shahroodi, S. Ghose, and O. Mutlu, "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," in *HPCA*, 2021.
31. J. M. You and J.-S. Yang, "Mrloc: Mitigating row-hammering based on memory locality," in *DAC*, 2019.
32. S. Zeitouni, D. Gens, and A.-R. Sadeghi, "It's hammer time: how to attack (rowhammer-based) dram-pufs," in *DAC*, 2018.