Pinpoint Rowhammer: Suppressing Unwanted Bit Flips on Rowhammer Attacks

Sangwoo Ji
Pohang University of Science and Technology
Pohang, South Korea
sangwooji@postech.ac.kr

Youngjoo Ko
Pohang University of Science and Technology
Pohang, South Korea
y0108009@postech.ac.kr

Saeyoung Oh
Pohang University of Science and Technology
Pohang, South Korea
osy4997@postech.ac.kr

Jong Kim
Pohang University of Science and Technology
Pohang, South Korea
jkim@postech.ac.kr

ABSTRACT

In recent studies, sophisticated attack vectors that use a Rowhammer bug have been developed. These attacks are dangerous, given that they can corrupt data stored in arbitrary memory rows without accessing them. Successful Rowhammer attacks require to flip data of the target cell. However, non-target cells are also corrupted by the attacks. Such unwanted bit flips can lead to unexpected consequences such as an attack failure and a system crash.

We propose a novel Rowhammer method, namely, Pinpoint Rowhammer, which flips the target bit while suppressing unwanted bit flips. The basic idea is the use of an effective data pattern for the target bit and ineffective data patterns for non-target bits. We evaluate the proposed method by conducting 107,965 attack instances on four different dynamic random-access memory (DRAM) modules. The proposed method increases the attack success rate from 28.9% to 72.4%, when compared with the state-of-the-art method (double-sided Rowhammer). In addition, the proposed method suppresses 99.7% of the unwanted vulnerable cells.

CCS CONCEPTS

• **Security and privacy → Hardware attacks and countermeasures**; *Security in hardware*; Hardware reverse engineering.

KEYWORDS

Rowhammer, hardware security, memory vulnerability

ACM Reference Format:
Sangwoo Ji, Youngjoo Ko, Saeyoung Oh, and Jong Kim. 2019. Pinpoint Rowhammer: Suppressing Unwanted Bit Flips on Rowhammer Attacks. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3321705.3329811

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329811

1 INTRODUCTION

Computer hardware is assumed to be trustworthy in many software-based security mechanisms. However, numerous hardware security bugs have been reported. Nowadays, the Rowhammer bug has attracted significant attention, as an attacker can change the data stored in dynamic random-access memory (DRAM). Rowhammer breaks the trustworthiness assumption of hardware by directly corrupting the memory without any software vulnerability. The attacker using Rowhammer only requires the reading or writing of neighboring rows of a victim row. After Kim et al. [13] demonstrated the prevalence of the Rowhammer bug on DDR3 memory modules, several sophisticated attacks that exploit Rowhammer have been introduced. For example, Rowhammer is used to escalate privileges by corrupting a page table entry (PTE) [8, 20, 25, 27], to conduct fault attacks on cryptographic systems [2, 19], and to bypass Linux sudo authentication by corrupting instruction sequences [7].

Most Rowhammer attacks use a two-step strategy: scan and reproduce. In the scan phase, an attacker scans a memory region for a vulnerable cell. The vulnerable cell can be exploited if the bit position of the cell matches the attack requirements. For example, in a PTE corruption attack [20], the bit that represents the physical page number (from the 12^{th}–31^{st} bit) should be flipped. In the reproduction phase, the attacker repetitively conducts Rowhammer attacks to the same row that contains the target cell, to induce that bit flip again. This strategy is feasible, given that bit flips caused by Rowhammer are reproducible [13, 27].

However, the Rowhammer attacks fail when unwanted bit flips occur. As Rowhammer affects the entire row that contains the target cell, the data stored in other cells are also flipped by the attack. Bit flips of the non-target cells are referred to as unwanted bit flips. Unwanted bit flips corrupt a portion of the page belonging to a victim or another process, which should not be modified. The corruption may incur unexpected results such as an attack failure and a system crash.

In this paper, we identify two causes of unwanted bit flips: incomplete scanning and a victim row that contains multiple vulnerable cells. First, detecting all of the vulnerable cells in memory rows is almost impossible. One vulnerable cell is revealed by a single Rowhammer attempt, but another vulnerable cell is revealed after several attack attempts. Thus, it is highly probable that the scan phase does not identify several vulnerable cells, which are problematic in the reproduction phase. When an attacker repeats the

Rowhammer attack, vulnerable cells that are not detected during the scan phase may be flipped. These unwanted bit flips are not expected by the attacker. Second, the scan phase may detect a row that contains multiple vulnerable cells. If a row has multiple vulnerable cells, the repetition of the Rowhammer attack flips the target bit and additional bits. Thus, the attacker cannot use that particular row as a victim row because unwanted bit flips from other vulnerable cells are inevitable.

We propose a novel Rowhammer method, namely, Pinpoint Rowhammer, which flips the target bit while suppressing unwanted bit flips. The method is based on the observation that the induction of a bit flip is related to the data of the neighboring rows (data pattern). Moreover, each vulnerable cell has its own set of effective data patterns which induce the bit flip, whereas other patterns are ineffective at flipping the bit. Pinpoint Rowhammer applies the effective data pattern to the target bit and the ineffective data patterns to non-target bits. However, as mentioned before, the determination of the effectiveness of a pattern is not simple, given that Rowhammer does not consistently flip vulnerable cells. If the scan phase fails to detect a vulnerable cell, an effective pattern is misclassified as ineffective to the cell. Thus, Rowhammer is conducted with the false ineffective pattern, and it will not suppress unwanted bit flips. To solve this problem, the neighboring rows are alternately overwritten with other ineffective patterns during the attack, denoted as an alternating pattern. As a result, the proposed method suppresses unwanted bit flips by reducing the effect of the false ineffective patterns.

We evaluate the performance of Pinpoint Rowhammer experimentally using the unwanted flip suppression rate and attack success rate. All the experiments are carried out using DDR3 chips from three DRAM manufacturers. We conduct a total of 107,965 Rowhammer attack instances. To measure the unwanted flip suppression rate, we count the number of vulnerable cells suppressed by Pinpoint Rowhammer. Moreover, we measure the attack success rate in two different attack scenarios. In the first scenario, the scan phase detects a row containing a single vulnerable cell which is used as the target cell. The proposed method is expected to suppress the undetected vulnerable cells during the reproduction phase. In the second scenario, the scan phase detects a row that contains multiple vulnerable cells including the target cell. Thus, the proposed method is expected to suppress non-target vulnerable cells and undetected vulnerable cells. In all the experiments, Pinpoint Rowhammer demonstrates 99.7% unwanted flip suppression rate. Moreover, the average attack success rate of the method is 71.6%, which is greater than the attack success rate of the state-of-the-art Rowhammer method by factors of 2 and 6, depending on the attack scenario.

This paper makes the following contributions:

- We analyze the effects of the unwanted bit flips on the existing attacks and identify two causes of the unwanted bit flips.
- We propose a novel data pattern (i.e., the alternating pattern) to suppress the unwanted bit flips using the relationship between the bit flips and data patterns.

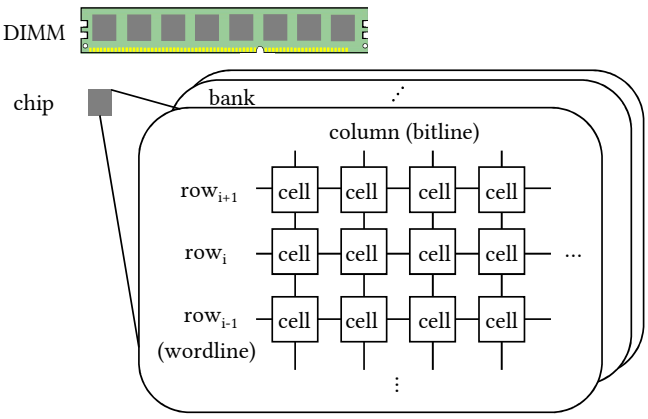


Figure 1: DRAM internal structure

- We propose Pinpoint Rowhammer which induces the target bit flip without inducing unwanted bit flips. Moreover, we evaluate the proposed method using commodity DRAMs.

2 BACKGROUND

2.1 DRAM Architecture

DRAM has a hierarchical structure (Figure 1). A channel physically connects a memory controller to dual inline memory modules (DIMMs). A DIMM contains DRAM chips on one side or both sides. Each side of a DIMM is termed a rank. A rank consists of eight (DDR3) or sixteen (DDR4) banks, and each bank has two-dimensional (2D) arrays of cells. A cell is the smallest unit of the DRAM architecture, and a cell consists of a capacitor and a transistor.

A cell represents a logical bit using its capacitor. The capacitor is either charged or uncharged with electrons. Based on the configuration of the manufacturer, the charged state of the capacitor represents the logical value 1 (true-cell) or logical value 0 (anti-cell) [16]. Given that the charged electrons are subject to leakage over time, data represented by the cell is changed when the voltage of the cell decreases below the threshold. Hence, DRAM periodically recharges cells to maintain the charged state of the cells. To recharge the cells, DRAM internally reads data from a row and re-writes the data to the same row. The procedure is termed the *refresh* of DRAM. The default refresh interval is 64 ms in DDR3 and DDR4 DRAMs.

2.2 Rowhammer Bug

Kim et al. revealed that most commodity DDR3 DRAM modules are vulnerable to disturbance errors [13]. They identified the root cause of disturbance errors as the repeated toggling of the same row. They repeatedly accessed (hammered) the same row to induce disturbance errors. A rapid voltage leak was then observed in cells in the neighboring rows because of electrical interference such as electromagnetic coupling, conductive bridges, and hot-carrier injection [13]. When the voltage of a cell decreases below the threshold prior to the DRAM refresh, the charged cell is changed to the uncharged cell. Therefore, a logical bit of the cell is also changed (i.e, 1

$\rightarrow 0$. In some systems, $0 \rightarrow 1$). The change is referred to as a bit flip, and the bug is referred to as a Rowhammer bug. In Figure 1, the hammering of row_i electrically interferes with row_{i+1} and row_{i-1}; thus, the bits of row_{i+1} and row_{i-1} are flipped. In the example, the hammered row (row_i) is referred to as an aggressor row.

The Rowhammer bug has the following two characteristics. First, bit flips are reproducible [13, 27]. When the same row is hammered, the same bits tend to be flipped. Second, Rowhammer with certain data patterns tends to induce more bit flips than that with other data patterns. Previous work revealed that the RowStripe [13] and killer [14] patterns cause more bit flips.

2.3 Rowhammer Attacks

Rowhammer Methods. Many researchers have developed effective Rowhammer methods. Kim et al. [13] repeatedly accessed a row to induce bit flips on neighboring rows. As the method hammers one row, it is referred to as single-sided Rowhammer. Since there are two neighboring rows that can affect the same row, Seaborn and Dullien [20] proposed double-sided Rowhammer which hammers the two rows. In Figure 1, row_{i+1} and row_{i-1} are hammered. Thus, the sandwiched row (row_i) is affected by both of the upper and lower rows, and the bits of row_i are flipped. For convenience sake, row_{i+1}, row_i, and row_{i-1} of double-sided Rowhammer are referred to as the upper aggressor row, victim row, and lower aggressor row, respectively. As double-sided Rowhammer induces more bit flips than others, it is the state-of-the-art Rowhammer method. Hence, most recent work used the method to perform Rowhammer [19, 20, 25, 27].

Since the locations of rows are determined by their physical addresses, attackers require additional information about the physical address space to conduct double-sided Rowhammer. In previous work, hugepage [19, 25] or the reverse engineered mapping algorithm between physical addresses and DRAM location [27] were used to conduct double-sided Rowhammer.

Attack Vectors. Seaborn and Dullien [20] proposed a PTE corruption attack to escalate privileges. To obtain the read and write permissions to an arbitrary page, the attack requires the flipping of a bit related to permissions or the physical page number of the page. These are exploitable bits for the PTE corruption attack. In the attack, an attacker attempts to corrupt the mapping between a virtual page and a physical page where the attacker has the read and write permissions of the virtual page. If the mapped physical page number is corrupted by the bit flip, the attacker gains the read and write permissions to the changed physical page. Gruss et al. [8] ported the PTE corruption attack to the JavaScript environment using a cache eviction method instead of a `clflush` instruction. Van der Veen et al. [25] conducted a PTE attack on the ARM architecture by exploiting the characteristic of Android ION memory allocator. If these PTE corruption attacks flip a non-exploitable bit, the attacks fail, and several side effects are induced.

A line of work attempted to corrupt instruction sequences. Seaborn and Dullien [20] proposed a sandbox escape attack that corrupts an instruction to escape from Chrome Native Client (NaCl). Bits that compose the operand register number or base address of the sandbox can be exploited by this attack. Gruss et al. [7] proposed an opcode flipping attack that induces a bit flip in the opcode sequence.

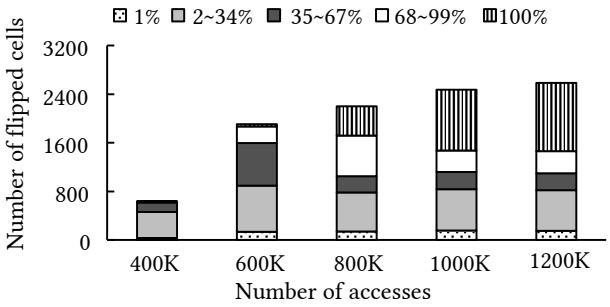


Figure 2: Reproducibility of bit flips over the number of accesses

The attack flips a bit in the sudoers.so file, and therefore the attack bypasses the authentication system. This attack also has a limited set of exploitable bits. If a non-exploitable bit is flipped, the changed instruction induces a system crash or an unexpected outcome.

Different attacks were also developed. Bosman et al. [3] used Rowhammer to counterfeit a JavaScript object, to access arbitrary memory location. Razavi et al. [19] proposed an RSA key corruption attack. They exploited the characteristic of which one bit change in the public key simplifies the factorization of the public key [19]. Jang et al. [10] proposed a processor denial-of-service attack on the Intel SGX environment. Moreover, Tatar et al. [23] conducted Rowhammer attack by only sending the network packets to the victim machine. Recently, Frigo et al. [6] demonstrated a new attack vector with graphics processing unit (GPU).

Unwanted Bit Flip. In previous work, unwanted bit flips of Rowhammer and a solution to this problem were discussed [27]. It was reported that double-sided Rowhammer affects the upper row (row_{i+2}) of the upper aggressor row and the lower row (row_{i-2}) of the lower aggressor row, in addition to the victim row (row_i). Hence, they attempted to allocate all the memory units of row_{i+2}, row_{i-2}, and row_i, to prevent other processes from being corrupted by the attack. However, their work was not sufficient to solve the problem of unwanted bit flips. Their method cannot guarantee the functionality of the attack when an unwanted bit flip occurs on the victim page. Since the victim page must be allocated on the victim row, unwanted bit flips on the victim page are inevitable. Therefore, we analyze the effects of the unwanted bit flips on the victim page, as presented in Section 3.2. Moreover, we propose a method to suppress unwanted bit flips while inducing the target flip, as presented in Section 4.

3 ROWHAMMER'S UNWANTED BIT FLIP

When a Rowhammer attack is conducted, non-target cells are flipped by the attack, and the bit flips of non-target cells are referred to as unwanted bit flips. In this section, we present the details of the two causes of the unwanted bit flips and their effects to the existing attacks.

3.1 Causes of Unwanted Bit Flips

Undetected Vulnerable Cells. Existing Rowhammer attacks leverage the reproducibility of bit flips [3, 7, 19, 25, 27]; however, the

Table 1: Prevalence of rows containing multiple vulnerable cells in commodity DRAM modules

Manufacturer	Serial number	ID	Total vulnerable rows	Vulnerable rows containing multiple flips	Total vulnerable pages	Vulnerable pages containing multiple flips
Saumsung	M378B5273DH0-CH9	A_1	3.4×10^5	3.3×10^5 (97.4%)	6.7×10^5	6.7×10^5 (98.6%)
	M378B5173QH0-CK0	A_2	3.2×10^5	2.9×10^5 (90.0%)	5.6×10^5	3.8×10^5 (68.1%)
	M378B5173QH0-CK0	A_3	3.1×10^5	2.6×10^5 (83.8%)	5.1×10^5	3.0×10^5 (58.7%)
	M378B5273DH0-CK0	A_4	3.5×10^5	3.3×10^5 (96.3%)	6.8×10^5	6.7×10^5 (97.8%)
	M378B5273DH0-CH9	A_5	3.4×10^5	3.3×10^5 (97.4%)	6.7×10^5	6.7×10^5 (98.6%)
	M378B5173EB0-CK0	A_6	5.2×10^2	2.2×10^2 (42.0%)	6.6×10^2	1.8×10^2 (27.9%)
Hynix	HTM351U6CFR8C-H9	B_1	3.1×10^5	2.2×10^5 (72.4%)	4.9×10^5	2.7×10^5 (56.9%)
	HTM351U6CFR8C-PB	B_2	3.8×10^5	3.7×10^5 (95.9%)	7.1×10^5	5.6×10^5 (79.2%)
	HTM351U6CFR9C-PB	B_3	3.9×10^5	3.8×10^5 (97.8%)	7.3×10^5	6.3×10^5 (85.3%)
	HTM351U6CFR8C-H9	B_4	3.3×10^5	2.5×10^5 (73.6%)	6.4×10^5	3.9×10^5 (60.5%)
Micron	MT16JTF51264AZ-1G6M1	C_1	3.5×10	0 (0%)	3.5×10	0 (0%)
	MT16JTF51264AZ-1G6M1	C_2	3.6×10	0 (0%)	3.6×10	0 (0%)

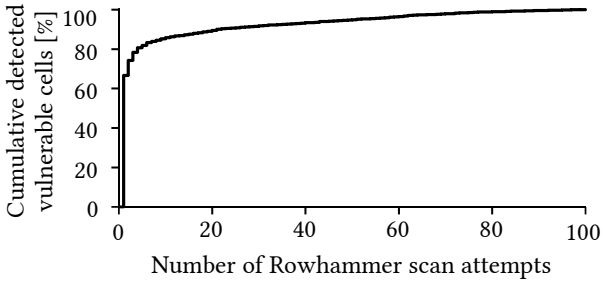


Figure 3: Cumulative detected vulnerable cells over the number of scan attempts

reproducibility of bit flips is not guaranteed [13, 27]. To investigate the reproducibility of bit flips, we conduct a proof-of-concept experiment with the number of accesses ranging from 400k–1,200k. We randomly select the 4,096 rows and fill the rows with the RowStripe pattern [24]. We conduct 100 attempts of double-sided Rowhammer on them. As shown in Figure 2, only 43.5% of the bit flips are consistent for all 100 attempts with 1,200k accesses. Meanwhile, 5.9% of the bit flips are induced only once in the 100 attempts with 1,200k accesses. The results imply that the majority of cells are not flipped consistently; thus, the reproducibility of bit flips is not guaranteed.

We further evaluate the number of attempts required for the detection of all the vulnerable cells in the region. Figure 3 shows that 1,921 (66.6%) vulnerable cells are detected within a single attempt of Rowhammer. It should be noted that the 99^{th} scan attempt detects a new vulnerable cell, unseen before, and no single attempt discovers the entire vulnerable cells. The experimental results indicate that there is a high probability that the result of each scan attempt has undetected vulnerable cells.

From the perspective of the attacker, undetected vulnerable cells are problematic. Attackers do not expect cells that are not flipped during the scan phase to be flipped in the reproduction phase. However, the undetected vulnerable cells can be flipped during the reproduction phase. According to Figure 3, the detection of all the vulnerable cells requires a large number of scan attempts. Furthermore, attackers may not detect all the vulnerable cells after several days of scanning [13]. Therefore, the existence of undetected vulnerable cells is inevitable. Hence, we need a way to suppress the unwanted bit flips, even if they are not detected during the scan phase.

Multiple Vulnerable Cells in a Row. Existing attacks cannot guarantee their functionality if the victim row has multiple vulnerable cells. Most attacks have specific sets of exploitable bits (Section 2.3) and function correctly only if one of the exploitable bits is flipped. If another bit is flipped, the functionality of the attacks cannot be guaranteed. Moreover, attacks undergo side effects such as memory corruption of other processes and a system crash. However, current attacks have no method to selectively induce a bit flip only on the desired bit. For successful Rowhammer attacks, an appropriate method is required for the selective induction of the bit flips.

To measure the prevalence of a row containing multiple vulnerable cells, we conduct a quantitative evaluation of 16 different DDR3 modules from three DRAM manufacturing companies. The experiment is carried out in a bare metal environment, to scan as many rows as possible (Section 5.1). The experimental results show that 12 modules are vulnerable to Rowhammer, and Table 1 presents the results of these 12 vulnerable modules. The vulnerable DRAM modules tend to have multiple vulnerable cells in a single row. For example, among the vulnerable rows of the B_3 module, 97.1% of them have multiple vulnerable cells. On the other hand, the results also show that C_1 and C_2 modules do not have any rows with multiple vulnerable cells. We conjecture that this is due to the small number of vulnerable cells in these modules. As the rows with multiple vulnerable cells are prevalent, our method should flip the target cell without flipping others.

3.2 Effects of Unwanted Bit Flips

In general, at least two 4KB pages are co-resident in a row because the size of a row is 8KB. Hence, the victim row is comprised of the victim page and co-resident pages. Attackers aim to induce the

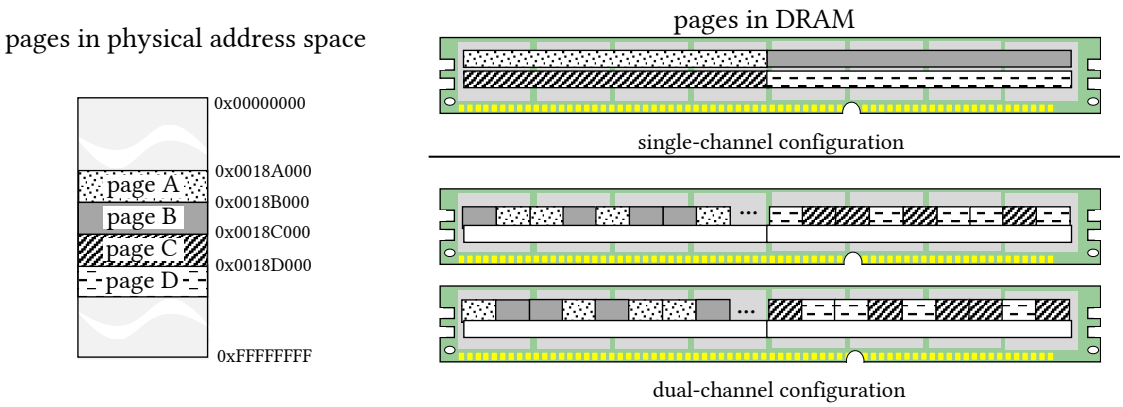


Figure 4: Pages on a single row in the single-channel and dual-channel configurations

target bit flip on the victim page. However, unwanted bit flips may occur on the victim page and co-resident pages. In this subsection, we explain the effects of unwanted bit flips on each page.

Unwanted Bit Flips on the Victim Page. When an unwanted bit flip occurs on the victim page, the bit flip degrades the correctness of the attack. This is because the victim page is related to the attack procedure. In Table 2, we conduct an analysis of the existing attacks, to reveal whether an attack failure or a system crash occurs due to the unwanted bit flip. For example, in the privilege escalation (PTE corruption) attack [20], the victim page is a 4KB page full of PTEs, and the attacks attempt to corrupt the target PTE. An unwanted bit flip may occur in the target PTE or another PTE. If another bit of the target PTE undergoes an unwanted bit flip, any access through the corrupted PTE is affected (e.g., change of the present bit). If another PTE undergoes an unwanted bit flip, another physical address translation is affected, which results in unexpected side effects (e.g., wrong physical base address). In the instruction corruption attack [7], the victim page is full of instructions. The attack attempts to corrupt the target instruction. However, an unwanted bit flip changes a non-target instruction into an invalid or inappropriate instruction. The changed instruction leads to an unsuccessful attack or a system crash.

Unwanted Bit Flips on a Co-Resident Page. An unwanted bit flip can occur on a co-resident page, which is on the victim row. The co-resident physical page could be mapped to the virtual page used by the same attack process or the virtual page used by another process, which includes the system process. The effects of unwanted bit flips are related to the process they corrupt. If they corrupt a system process memory, the bit flips may cause a system crash or segmentation fault. Moreover, even if a non-critical process memory is corrupted, the corrupted process may suffer from various errors.

Co-resident pages in a row are determined by the mapping algorithm between the physical addresses and memory bank/row, which depends on the hardware configuration. A single-channel configuration and a dual-channel configuration have different mapping algorithms [18, 27]. In a single-channel configuration, two pages are co-resident in a row (Figure 4). In contrast, a dual-channel configuration has more than two pages in a row because consecutive cache line accesses are interleaved in two channels. With an increase in the number of co-resident pages, there is an increase in the probability that the Rowhammer attack will induce unwanted bit flips on co-resident pages.

Table 2: Effects of unwanted bit flips. (●) The attack suffers the problem. (○) The attack is resistant to the problem. (◐) We added footnotes to special cases.

	Attack failure	Co-resident page corruption (e.g., system crash)
Privilege escalation[20]	●	●
Sandbox escape[20]	●	●
Rowhammer.js[8]	●	●
Dedup est machina[3]	●	●
Flip Feng Shui[19]	◐†	●
Xiao et al.[27]	●	●
Drammer[25]	●	◐‡
Gruss et al.[7]	◐†	●

† Attackers can check the presence of unwanted bit flips on the victim page because attackers have the read permission to the victim page.
‡ All pages in the victim row are controlled by the attacker unless mapping algorithm between physical addresses and memory addresses is complex (i.e., dual-channel configuration).

4 METHOD: PINPOINT ROWHAMMER

We develop a novel Rowhammer method to suppress unwanted bit flips during the Rowhammer attacks, namely, Pinpoint Rowhammer. The proposed method is designed to achieve two goals: inducing the target bit flip and suppressing unwanted bit flips. The method leverages the dependency between bit flips and data patterns. In the remainder of this section, we show the dependency between bit flips and data patterns. Thereafter, we present a detailed discussion on Pinpoint Rowhammer.

Threat Model. It is assumed that an attacker accesses a victim machine with user privileges. The attacker can change the data of aggressor rows, which belong to the attacker. However, the attacker

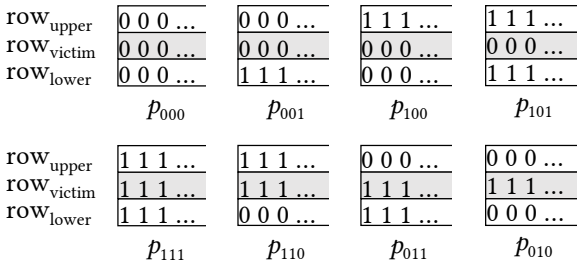


Figure 5: Eight data patterns for dependency check

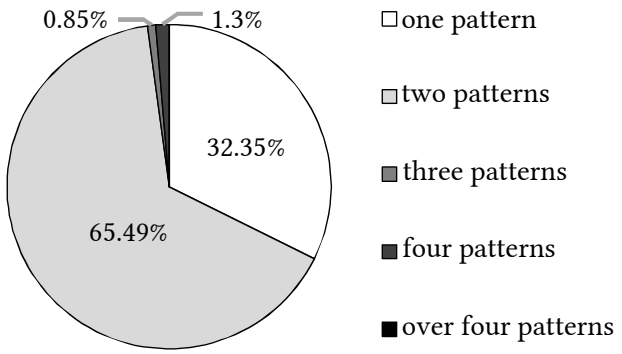


Figure 6: The number of effective patterns for a cell

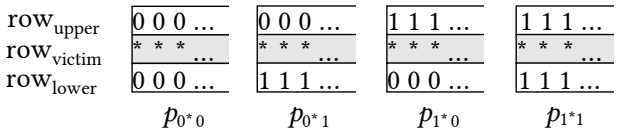


Figure 7: Victim data–agnostic data patterns

cannot change the data of the victim row. Moreover, the data stored in the victim row at the time of the attack is unknown. Hence, the victim row is filled with random data in the evaluation. It is assumed that the attacker conducts double-sided Rowhammer using transparent hugepages, as achieved in [20].

4.1 Dependency between Bit Flips and Data Patterns

Pinpoint Rowhammer is based on the key observation that induction of a bit flip depends on the data of the aggressor and victim rows. Prior work found that there is a relationship between a bit flip and the data of other cells [13]. Moreover, a couple of data patterns were found to induce more bit flips than others [13, 14].

We hypothesize that a bit flip is induced only if a particular data pattern is stored in the aggressor rows. We conduct experiments with eight data patterns, which are generated by modifying the data of the upper aggressor row, victim row, and lower aggressor row (Figure 5). A pattern is denoted as p_{ijk}; where i, j, and k represent the data of the upper aggressor row, victim row, and lower aggressor row, respectively. The effectiveness of a pattern with respect to a cell $f(c,\ p_{ijk})$ and a set of effective patterns for a cell $E(c)$ are defined as follows:

$$f(c,\ p_{ijk}) = \begin{cases} 1 & \text{if cell } c \text{ is flipped by } p_{ijk} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$E(c) = \{p_{ijk} \mid f(c,\ p_{ijk}) = 1\} \quad (2)$$

We investigate the number of data patterns that flip a cell $|E(c)|$. Figure 6 shows that 32.4% of the vulnerable cells are flipped by one data pattern. At the same time, 65.5% of the vulnerable cells are flipped by two patterns. Only 2.2% of the vulnerable cells are flipped by three or four patterns, and no cell is flipped by more than four patterns. The results indicate that most cells are flipped by a couple of specific data patterns. This confirms the hypothesis that a bit flip is induced by Rowhammer with a particular data pattern, and this observation is used to develop Pinpoint Rowhammer.

4.2 Pinpoint Rowhammer

The main idea of Pinpoint Rowhammer is the use of different data patterns on the target bit and non-target bits. For the target bit, an effective pattern p_e is used to induce the bit flip (where $p_e \in E(c_{\text{target}})$). Meanwhile, an ineffective pattern p_i is used for each non-target bit, to suppress unwanted bit flips (where $p_i \in \overline{E(c_{\text{non-target}})}$).

The simplest approach to leverage ineffective patterns is to select one of them. However, the selection of a pattern is unreliable due to the probability of using misclassified patterns. As explained in Section 3.1, vulnerable cells may not be flipped during a Rowhammer scan with a pattern, although the pattern is effective to the cell. Consequently, the pattern is misclassified as ineffective to the cells in that case. If this misclassified pattern is selected for the suppression of the bit flip, Rowhammer may induce unwanted bit flips during the reproduction phase, and the simple approach fails. In this study, we develop an alternating pattern to reduce the effect of misclassified effective patterns.

Alternating Pattern. We observe two characteristics of bit flips from the experiments in Section 3 and Section 4. First, a large number of accesses is needed to flip a vulnerable cell (Figure 2). Second, a vulnerable cell is only flipped by Rowhammer with its effective patterns (Figure 6). Hence, it is concluded that a large number of accesses with an effective pattern is needed to flip a vulnerable cell.

We propose a scheme to break the two characteristics for suppressing bit flips, namely, an alternating pattern. The alternating pattern alters the ineffective patterns during a Rowhammer attack, instead of relying on one ineffective pattern. It uses two or more ineffective patterns alternately with an interval. Thus, the number of accesses with a single pattern is reduced. Even if a misclassified effective pattern is used, the number of accesses with the misclassified pattern is insufficient to induce bit flips. Therefore, the two characteristics are not preserved, and bit flips are suppressed.

To alter the ineffective patterns, the number of usable patterns and their effectiveness should be determined. Among the upper aggressor row, victim row, and lower aggressor row, an attacker has a

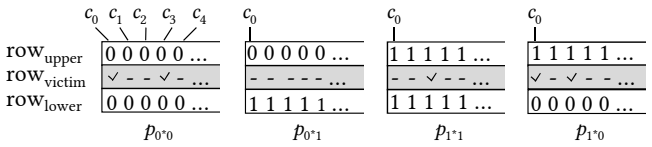
write permission to the upper and lower aggressor rows. Therefore, four victim-agnostic patterns are obtained by changing the data of the upper and lower aggressor rows (Figure 7). A victim-agnostic pattern is denoted as p_{i*j}; where i and j represent the data of the upper and lower aggressor rows, respectively. The effectiveness of a victim-agnostic pattern is calculated as follows. First, we identify two related data patterns that have the same upper and lower data with the given victim-agnostic pattern. For example, p_{011} and p_{001} are two related patterns for p_{0*1}. A victim-agnostic pattern is considered as effective to a bit when at least one of the two related patterns is effective.

$$f(c,\ p_{i*j}) = f(c,\ p_{i0j})\ ||\ f(c,\ p_{i1j}) \tag{3}$$

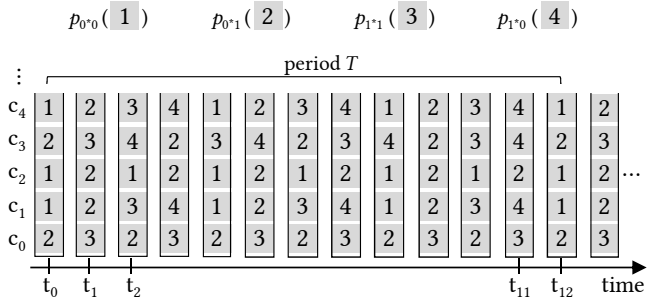
This is because an attacker has no knowledge of the data stored in the victim row at the time of the attack. The data can be either 0 or 1; thus, the attacker should avoid using a victim-agnostic pattern that may cause bit flips depending on the data of the victim row. In the above example, if either p_{011} or p_{001} is effective, p_{0*1} is considered as effective. Otherwise, p_{0*1} is considered as ineffective.

We explain the alternating pattern using an example, as presented in Figure 8. The example shows five cells, and the alternating pattern suppresses their bit flips. Other cells are omitted for brevity. Figure 8a shows the results of the scan phase. According to the results, c_0, c_2, and c_3 are vulnerable cells, whereas c_1 and c_4 are not vulnerable. We alternate the ineffective patterns for each cell. Figure 8a shows that $E(c_0) = \{p_{0*0}, p_{1*0}\}$ and $\overline{E(c_0)} = \{p_{0*1}, p_{1*1}\}$. Therefore, we alternate p_{0*1} and p_{1*1} for c_0 (Figure 8b). At the same time, the results show that $E(c_1) = \phi$ and $\overline{E(c_1)} = \{p_{0*0}, p_{0*1}, p_{1*1}, p_{1*0}\}$; thus, we alternate p_{0*0}, p_{0*1}, p_{1*1} and p_{1*0} for c_1. Each cell follows its own sequence of the ineffective patterns. A cell is rarely flipped by all the four patterns. In that case, a sequence of the four patterns is used for the cell. At time t_0, the data stored in c_0, c_1, c_2, c_3, and c_4 are p_{0*1}, p_{0*0}, p_{0*0}, p_{0*1}, and p_{0*0}, respectively. Subsequently, the attacker writes the next ineffective pattern to each cell, i.e., p_{1*1}, p_{0*1}, p_{0*1}, p_{1*1}, and p_{0*1}. As shown in Figure 8, a batch of 12 data patterns is repeated, and the interval of the same pattern is $t_{12} - t_0$. The size of the interval affects the bit flip suppression rate and the attack success rate. This phenomenon is discussed further in this section.

We conduct a proof-of-concept experiment to measure the effectiveness of the alternating pattern. We randomly select 512 rows, which have 355 vulnerable cells. We conduct Rowhammer to them with the alternating pattern. Among the 355 cells, the alternating pattern suppresses 351 (98.9%) vulnerable cells. The alternating pattern is dependent on the effectiveness of patterns retrieved from the scan phase. However, the effectiveness of patterns can be misclassified because of the undetected vulnerable cells. Therefore, the worst case of the alternating pattern is that all the vulnerable cells are not detected during the scan phase. In that case, the alternating pattern uses effective patterns without recognizing that they are effective. We measure the suppression rate of the alternating pattern in the worst case scenario, wherein all the detected vulnerable cells are neglected. Thus, we alternate p_{0*0}, p_{0*1}, p_{1*1} and p_{1*0} for every vulnerable cell regardless of the patterns' actual effectiveness. The alternating pattern suppresses 341 (96.1%) of the cells, even in the worst case scenario. As mentioned before, although effective patterns are involved in the alternating pattern, the number of accesses of the effective patterns is reduced when compared with Rowhammer with a single effective pattern. Hence, the worst case results show the suppression rate that is comparable to the typical alternating pattern.



(a) Result of scan phase. The check mark represents a bit flip, and the hyphen represents no bit flip.



(b) An example of the alternating pattern. Each pattern is abbreviated as a number.

Figure 8: Building blocks of Pinpoint Rowhammer

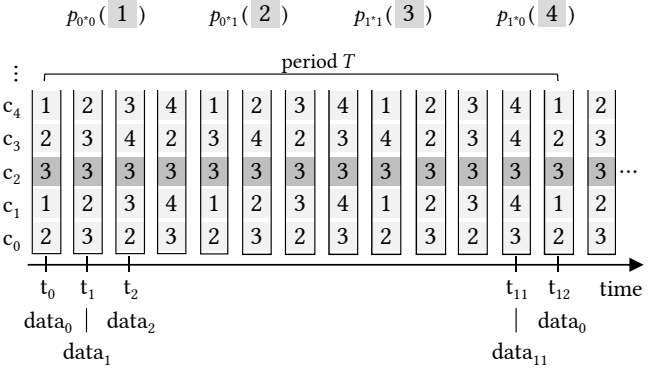


Figure 9: An example of Pinpoint Rowhammer that flips c_2.

Putting It Together. Pinpoint Rowhammer uses an effective pattern for the target bit and the alternating pattern for other bits. Figure 9 shows an example of a Pinpoint Rowhammer process that attempts to flip the cell c_2. For simplicity, the example uses the same results of the scan phase presented in Figure 8a. The target cell (c_2) has two effective patterns (p_{1*1} and p_{1*0}). If a target cell has multiple effective patterns, the patterns are empirically prioritized in the order of p_{1*1}, p_{0*0}, p_{1*0}, and p_{0*1}. Therefore, we use p_{1*1} to flip c_2. It should be noted that Pinpoint Rowhammer also repeats a batch of 12 data patterns (from the data_0–data_{12}) as the alternating pattern.

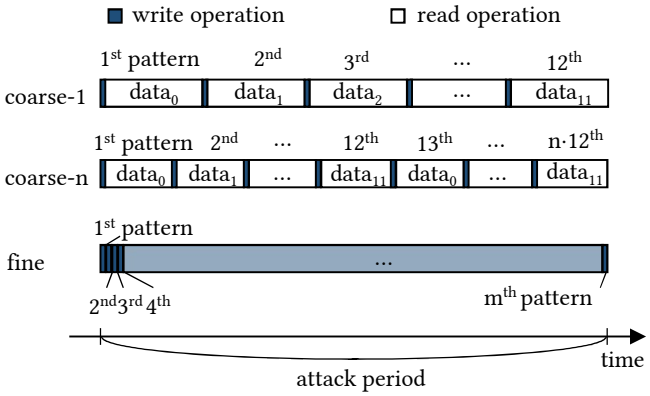


Figure 10: Operation sequence of Pinpoint Rowhammer

Pinpoint Rowhammer is required to use write operations because the alternating pattern overwrites the data of aggressor rows. The results obtained from previous work [13] and this study (Figure 11) reveal that Rowhammer using write operations is less effective than Rowhammer using read operations. Thus, we reduce the rate of write operations by inserting a sequence of read operations between two data patterns. In the example presented in Figure 9, the $data_0$ (p_{0*1}, p_{0*0}, p_{1*1}, p_{0*1}, and p_{0*0} for the five cells) are written to the aggressor rows at t_0. Before the cells are overwritten with the $data_1$, Rowhammer is conducted using read operations. Consequently, Rowhammer affects the victim row with $data_0$ during the read operations. After the given interval, Pinpoint Rowhammer overwrites the rows with the next data ($data_1$) and conducts Rowhammer using read operations. The method repeats this procedure until the attack is finished.

We evaluate the effectiveness of Pinpoint Rowhammer over a range of alternating granularities (alternating intervals). Three cases of operation sequences for an attack attempt are described in Figure 10. First, the coarse-1 sequence consists of a single batch of the 12 data patterns that are equally interleaved. In this case, the number of read operations between the two data pattern writes is large. Second, the coarse-n sequence consists of n batches of the 12 data patterns. Therefore, the number of read operations between two data pattern writes is smaller than that of the coarse-1 case. Third, the fine sequence consists of m data patterns, and the sequence only contains write operations. The following two metrics are measured: the target bit flip rate and unwanted bit flip rate. In Figure 11, the coarse-1 sequence yields the highest target flip ratio. The target flip ratio decreases when the alternating granularity is finer (i.e., large n for coarse-n). The fine sequence yields the lowest target flip rate. On the other hand, the unwanted flip rate of the coarse-1 sequence is 4.1%, which is higher than that of the other sequences, with the exception of coarse-2. The coarse-1 sequence is the best configuration for the high target flip rate, whereas the fine sequence is the best for the low unwanted flip rate. Attackers can choose the alternating granularity based on whether their priority is the high target flip rate or the low unwanted flip rate. We use the coarse-1 sequence for subsequent experiments in this study, to induce the target flip.

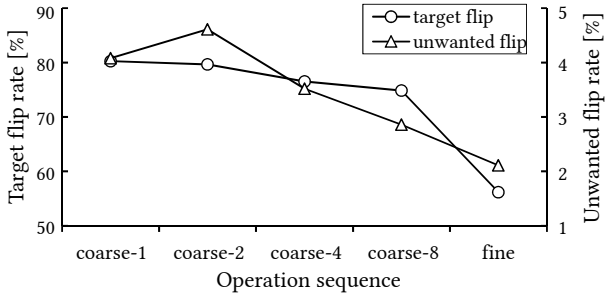


Figure 11: Pinpoint Rowhammer over different alternating granularity (B_3 module)

```
1  function pinpointrowhammer(total_access,
       alter_granularity, row, bank, column)
2  {
3   for(i=0; i<alter_granularity*12; i++)
4   {
5    // retrieve a pinpoint data
6    upper_data, lower_data =
         get_ith_data(i%12)
7
8    // write the pinpoint data
9    for(column=0; column<1024; column++)
10   {
11    // write 8 bytes data to (row, bank,
          column)
12    rowhammer_with_write(row, bank, column
          , upper_data, lower_data)
13   }
14   // Rowhammer with read operation
15   for(int j=0; j<total_access/
         alter_granularity-1024*2; j++)
16    rowhammer_with_read(bank, row)
17  }
18 }
```

Code 1: Pseudo code of Pinpoint Rowhammer

Pseudo Code. Code 1 is a pseudo code of Pinpoint Rowhammer. Line 3 handles the number of batches of 12 data patterns. Line 6 retrieves the i-th data for the upper and lower aggressor rows. The data consists of an effective pattern for the target bit and the alternating pattern for the other bits. Lines 9–13 write the data to the upper and lower aggressor rows. The writing of the entire row consumes 1,024 iterations because the row size is 8KB, and 8 bytes are written in each iteration. Thereafter, Pinpoint Rowhammer performs read operations in lines 15–16. The number of accesses with the read operation is controlled by the variable alter_granularity. Attackers can adjust the alternating granularity to increase or decrease the number of read operations. Lines 6–16 are repeated to conduct Pinpoint Rowhammer with the following data.

5 EXPERIMENTAL SETUP

5.1 Experimental Environments

Two environments are used in this study. First, we build a bare metal environment on the Intel i7-4770 CPU and Dell 0KWVT8 motherboard. The machine is used to measure the prevalence of multiple bit flips in a row, as shown in Table 1. We implement a

tiny operating system (OS) that functions only as a Rowhammer attack. We run the tiny OS in the bare metal environment from a USB bootloader. Hence, we can scan the entire memory region except for a small portion of the reserved area. Second, we build another environment on the Intel i5-4460 CPU and Asrock b85m pro4 motherboard. This machine is used to conduct all the proof-of-concept experiments and Rowhammer attacks in this study.

5.2 Experimental Procedure

Rowhammer Configuration. We use the double-sided version of Pinpoint Rowhammer, which is described in Section 4. We use double-sided Rowhammer as a baseline method [20]. The RowStripe pattern is used for the baseline method [13]. The Rowhammer attacks (Pinpoint Rowhammer and the baseline) access rows 1,200k times over a period of 128 ms, which is double the 64-ms refresh interval [13]. Thus, each aggressor row is accessed 600k times.

Detecting Vulnerable Rows. For the evaluation of Pinpoint Rowhammer, we scan a large amount of memory to select victim rows. We conduct Rowhammer with the RowStripe pattern to detect vulnerable cells [13]. If a row undergoes at least one bit flip, we use the row in further experiments. These rows are categorized into two groups, which represent two different attack scenarios. Each row in the first group contains a single bit flip, whereas each row in the second group contains multiple bit flips. In the first scenario, the proposed method is expected to induce the target bit flip while suppressing the bit flips of undetected vulnerable cells. In the second scenario, the proposed method is expected to induce the target bit flip while suppressing detected and undetected vulnerable cells. As a result, we demonstrate that Pinpoint Rowhammer can suppress unwanted bit flips even if they are not detected during the scan phase, and the method can selectively induce a bit flip among multiple vulnerable cells.

Conducting Rowhammer. We conduct Pinpoint Rowhammer and the baseline Rowhammer attacks on the two groups. Pinpoint Rowhammer requires information on the effective patterns of each cell in the victim row. Hence, we conduct the additional scan phase with the eight patterns presented in Figure 5, and the effectiveness of each pattern is classified based on the scan results. Thereafter, the alternating pattern is generated using the ineffective patterns, and Pinpoint Rowhammer combines it with the effective pattern of the target bit (Figure 9).

We evaluate the performance of the proposed method using two attack strategies. First, we conduct a Rowhammer attack once and analyze the results of the attack. Second, we conduct the Rowhammer attack multiple times until the target bit flip is induced. If the target flip is induced, we terminate the repetition and analyze the results of the attack. The limit of the repetition is set as 10, and the attack is regarded as a failure if the target bit flip is not induced by the end of the 10^{th} attack attempt. In that case, we analyze the result after the 10^{th} attack attempt.

Data of the Victim Row. In this experiment, the victim row is filled with random data, with the exception of the target bit. The random data represents the fact that attackers cannot write data to the victim row, and the data stored in the victim row is determined by the victim process. Nevertheless, the target bit is filled with the RowStripe data to guarantee that the target bit could be flipped.

Table 3: The number of attack instances on each module

Module	Attack instances (a single flip)	Attack instances (multiple flips)
A_2	5,037	11,687
A_3	5,113	10,906
B_1	10,000	25,529
B_3	10,000	29,693

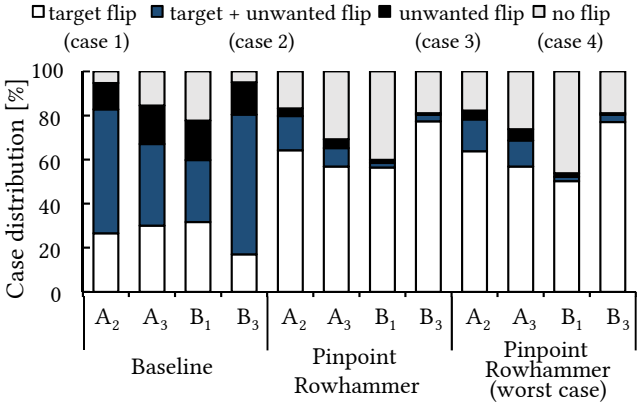
The fixed data of the target bit is required for the evaluation of the target bit flip rate.

6 EVALUATION

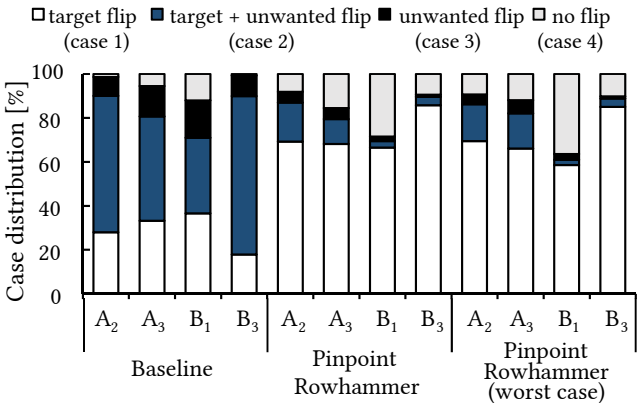
For a thorough comparison, we conduct an analysis of the Rowhammer attack results. We categorize the results of Rowhammer attack into four cases. The first case is that the target bit is flipped without any unwanted bit flips. This case represents the complete success of the Rowhammer attack, whereas underlying three cases represent the attack failure. The second case is that an unwanted bit is flipped, in addition to the target bit flip. In this case, attackers cannot guarantee the success of the attack, although the target bit is flipped. The unwanted bit flip may cause unexpected behavior such as an attack failure, a system crash, and memory corruption (Table 2). Moreover, an attack attempt that induces unwanted flips can be detected by the system administrator because of its side effects. Therefore, we regard this case as a failure, whereas previous work regarded this case as a success without considering the side effects of unwanted bit flips. The third case is that an unwanted bit is flipped without the target bit flip. This case also induces the side effects, as presented in Table 2. The last case is that no flips occur; thus, no side effects are encountered.

Attacking a Row Containing a Single Bit Flip. We conduct the baseline Rowhammer and Pinpoint Rowhammer attacks on rows of the first group. Each row in the first group contains a single bit flip. We conduct 30,150 attack instances on the rows (Table 3), and the attack instances are conducted on four different modules (A_2, A_3, B_1, and B_3). One attack instance consists of three attack attempts: the baseline attack, the Pinpoint Rowhammer attack, and the worst case simulation of the Pinpoint Rowhammer attack (neglecting the scan results for the non-target cells). Each attack instance is applied to a different row.

Figure 12 shows the results of the 30,150 attack instances. Each bar is divided into four parts. The darker parts of each bar (case 2 and 3) represent unwanted bit flips. In Figure 12a, an average of 61.7% of the baseline attacks yield unwanted bit flips. On the other hand, 9.7% of the Pinpoint Rowhammer attacks yield unwanted bit flips, and 10.8% of the worst case Pinpoint Rowhammer attacks yield unwanted bit flips. By suppressing the unwanted bit flips, the target and unwanted flip case (case 2) becomes the target flip case (case 1), and the unwanted flip case (case 3) becomes the no flip case (case 4). Hence, Pinpoint Rowhammer successfully increases the attack success rate (case 1) from 26.3% to 63.7%, on average, when compared with the baseline. It should be noted that the attack success rate (case 1) of the worst case Pinpoint Rowhammer (62.0%) is comparable to that of Pinpoint Rowhammer.

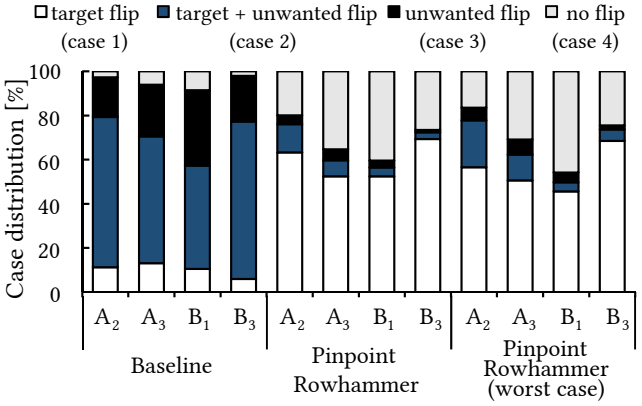


(a) One Rowhammer attempt

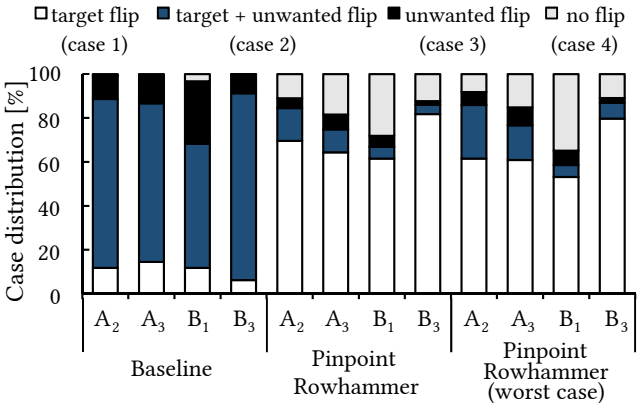


(b) Repeated Rowhammer attempts

Figure 12: Rowhammer results on rows of the first group (containing a flip)



(a) One Rowhammer attempt



(b) Repeated Rowhammer attempts

Figure 13: Rowhammer results on rows of the second group (containing multiple flips)

Figure 12b shows the attack results of the repeated Rowhammer attempts. As this attack strategy involves the repetition of Rowhammer until the target bit is flipped, the no flip rate is decreased. Compared with the single-attempt-strategy, as presented in Figure 12a, the no flip rate (case 4) of Pinpoint Rowhammer decreases from 26.6% to 15.4%, on average. The target flip rate (case 1) of Pinpoint Rowhammer increases from 63.7% to 72.4%. At the same time, the target and unwanted flip rate (case 2) increases from 7.3% to 9.0%, and the unwanted flip rate (case 3) increases from 2.4% to 3.2%. If Rowhammer is repeated until the target bit is flipped, the target flip rate and unwanted flip rate increase. The trade-off between the high target flip rate and low unwanted flip rate can be leveraged.

Attacking a Row Containing Multiple Bit Flips. We conduct the same experiments on the rows of the second group. Each row in the group contains multiple bit flips. We conduct 77,815 attack instances on the rows (Table 3). As each row has multiple bit flips, it is necessary to suppress the other known bit flips while flipping the target bit.

Figure 13 shows the results of the 77,815 attack instances. In Figure 13a, 85.0% of the baseline attacks yield unwanted bit flips (case 2 and 3). The result implies that the baseline method is not suitable for the attack on a row that contains multiple bit flips. In contrast to the baseline method, only 10.1% of the Pinpoint Rowhammer attacks yield unwanted bit flips. This rate is comparable to that obtained from the attacks conducted on the rows of the first group (Figure 12a). Therefore, Pinpoint Rowhammer is a suitable method for the attack on a row that contains multiple flips. Moreover, 15.3% of the worst case Pinpoint Rowhammer attacks yield unwanted bit flips. On average, Pinpoint Rowhammer increases the attack success rate (case 1) from 10.2% to 59.4%, when compared with the baseline. The worst case Pinpoint Rowhammer also increases the rate (case 1) to 55.3%.

We also evaluate the repeated Rowhammer attempts on these rows (Figure 13b). The no flip rate (case 4) of Pinpoint Rowhammer is reduced from 30.5% to 17.5%, on average, when compared with the single-attempt-strategy, as presented in Figure 13a. The target flip rate (case 1) of Pinpoint Rowhammer increases from 59.4% to 69.3%. At the same time, the target and unwanted flip rate (case

2) increases from 6.8% to 8.7%, and the unwanted flip rate (case 3) increases from 3.3% to 4.5%. The discrepancy between attack success rates (case 1) of the baseline method and PINPOINT ROWHAMMER is larger in the second group results (Figure 13) than in the first group results (Figure 12). Hence, PINPOINT ROWHAMMER is especially needed to attack a row that contains multiple bit flips.

Case Study on Unwanted Bit Flips. PINPOINT ROWHAMMER cannot completely eliminate unwanted bit flips; thus, we analyze the unwanted bit flips caused by PINPOINT ROWHAMMER. During the experiments of the 107,965 attack instances, PINPOINT ROWHAMMER induces a total of 12,681 unwanted bit flips. Among them, 10,841 (85.5%) unwanted bit flips are not detected in the scan phase. Hence, their alternating patterns contain effective patterns and cause bit flips. On the other hand, PINPOINT ROWHAMMER fails to suppress 1,840 (14.5%) unwanted bit flips that are detected during the scan phase.

We detect a total of 558,242 vulnerable cells during the scan phase, and PINPOINT ROWHAMMER attempts to suppress them. Consequently, PINPOINT ROWHAMMER suppresses 556,402 (99.7%) of them. According to the results, a small number of unwanted bit flips is inevitable. However, by conducting the scan phase for a long period of time, the number of unwanted bit flips can be reduced by eliminating the use of misclassified effective patterns.

7 DISCUSSION

Deployability of PINPOINT ROWHAMMER. PINPOINT ROWHAMMER is easily deployable to the existing Rowhammer attacks because the proposed method requires two simple conditions: the scanning of additional patterns and the write permission to the aggressor rows. First, the additional scanning with the eight patterns (Figure 5) is required after the detection of an exploitable bit. The additional scan requires a small modification of the code and incurs negligible time overhead. Second, the write permission to the aggressor rows is required, and most attack methods already have the write permission to the rows. Overall, PINPOINT ROWHAMMER can be deployed to the existing attacks that satisfy both conditions [3, 7, 8, 19, 20, 25, 27]. Moreover, the method is deployable to a future Rowhammer attack if the two conditions are preserved.

Although the double-sided PINPOINT ROWHAMMER is demonstrated in this study, PINPOINT ROWHAMMER is also deployable to single-sided Rowhammer and one-location Rowhammer [7, 13] by changing the data of the one aggressor row. We conduct a proof-of-concept experiment of single-sided PINPOINT ROWHAMMER, and the method yield a 100% suppression rate for the attacks on 215 vulnerable rows.

Limitations of PINPOINT ROWHAMMER. PINPOINT ROWHAMMER has two limitations. First, the proposed method cannot completely eliminate unwanted bit flips. Several execution of the scan phase may not be sufficient to detect all the effective patterns (Figure 3); thus, the presence of undetected vulnerable cells is inevitable. Hence, effective patterns are included in the alternating pattern, and they degrade the ability of PINPOINT ROWHAMMER. However, during the 107,965 worst case attacks, PINPOINT ROWHAMMER suppresses the 554,277 (99.3%) vulnerable cells without knowing the ineffective patterns for the cells. Moreover, it outperforms the traditional double-sided Rowhammer. Second, PINPOINT ROWHAMMER should allocate all the memory units of aggressor rows. When any unit of aggressor row is not allocated by PINPOINT ROWHAMMER, the method cannot overwrite the data of the unallocated regions. Therefore, the suppression of bit flips in this regions is not guaranteed. This limitation can be partially solved using a transparent huge page [19, 20, 25] or by revealing an address mapping of DRAM [18, 22, 27].

Countermeasures. In previous work, methods to defeat Rowhammer are discussed, which can be categorized into two groups. The first group of defenses requires a hardware modification to prevent bit flips. Target row refresh (TRR) [11] and pseudo target row refresh (pTRR) schemes refresh adjacent rows when a row is accessed more than a threshold. Probabilistic adjacent row activation (PARA) [12, 13] activates an adjacent row with a low probability when a row is accessed. Counter-based methods detect aggressor rows based on the value of the hardware counters and refresh adjacent rows [12, 15, 21]. Equipping error-correcting code (ECC) can be considered as a hardware-based countermeasure. However, it is reported that normal Rowhammer attack is reproducible in the ECC memory [5, 26]. These defenses require DRAM modification or the supports of a memory controller; thus, existing vulnerable DRAM modules are typically not well protected.

The second group of defenses develop software to defeat the Rowhammer attacks. Several methods use hardware performance counters to detect Rowhammer attacks in real time [1, 9, 17]. However, the methods cause huge overhead, and their thresholds are typically inaccurate because each module has its own characteristics. A method, namely, B-CATT, eliminates vulnerable pages from an allocation pool [4], but the method cannot be used in several modules [7]. Another method, namely, G-CATT, isolates kernel space [4] from user space; however, the isolation consumes large amount of memory depending on the hardware configuration [22].

8 CONCLUSIONS

Existing Rowhammer attacks are subject to unwanted bit flips. In this paper, we propose PINPOINT ROWHAMMER that flips the target bit without inducing unwanted bit flips. The proposed method is based on the alternating pattern that dynamically changes the data of aggressor rows during the attack. The alternating pattern consists of ineffective patterns that prevent vulnerable cells from being flipped. PINPOINT ROWHAMMER uses an effective pattern for the target bit and the alternating pattern for the other bits. Through experiments, we demonstrate that PINPOINT ROWHAMMER is very effective in flipping only the target cell.

PINPOINT ROWHAMMER is applicable to rows that contain multiple vulnerable cells, given that it is the first work to selectively induce bit flips. The method allows us to choose a desired bit flip and flips the bit exclusively. We believe this work can help researchers to understand the unexplored phenomenon of Rowhammer. Moreover, we believe that this research will contribute to the development of trustworthy DRAM chips that are protected from Rowhammer attacks.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIP) (No. 2017R1A2B4010914).

REFERENCES

[1] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd Austin. 2016. ANVIL: Software-based protection against next-generation rowhammer attacks. *ACM SIGPLAN Notices* 51, 4 (2016), 743–755.

[2] Sarani Bhattacharya and Debdeep Mukhopadhyay. 2016. Curious case of rowhammer: flipping secret exponent bits using timing analysis. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 602–624.

[3] Erik Bosman, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2016. Dedup est machina: Memory deduplication as an advanced exploitation vector. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 987–1004.

[4] Ferdinand Brasser, Lucas Davi, David Gens, Christopher Liebchen, and Ahmad-Reza Sadeghi. 2017. Cant touch this: Software-only mitigation against rowhammer attacks targeting kernel memory. In *USENIX Security Symposium*. 117–130.

[5] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In *S&P*. https://www.vusec.net/download/?t=papers/eccploit_sp19.pdf

[6] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi. 2018. Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In *2018 IEEE Symposium on Security and Privacy (SP)*. 357–372. https://doi.org/10.1109/SP.2018.00022

[7] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. 2018. Another flip in the wall of rowhammer defenses. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 245–261.

[8] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer. js: A remote software-induced fault attack in javascript. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 300–321.

[9] N Herath and A Fogh. 2015. These are Not Your Grand Daddys CPU Performance Counters– CPU Hardware Performance Counters for Security. *Black Hat* (2015).

[10] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. 2017. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In *Proceedings of the 2Nd Workshop on System Software for Trusted Execution (SysTEX'17)*. ACM, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/3152701.3152709

[11] JEDEC. 2018. Low Power Double Date Rate 4. (2018).

[12] Dae-Hyun Kim, Prashant J Nair, and Moinuddin K Qureshi. 2015. Architectural support for mitigating row hammering in DRAM memories. *IEEE Computer Architecture Letters* 14, 1 (2015), 9–12.

[13] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM SIGARCH Computer Architecture News*, Vol. 42. IEEE Press, 361–372.

[14] M Lanteigne. 2016. How rowhammer could be used to exploit weaknesses in computer hardware. (2016).

[15] Eojin Lee, Sukhan Lee, G Edward Suh, and Jung Ho Ahn. 2018. TWiCe: Time Window Counter Based Row Refresh to Prevent Row-Hammering. *IEEE Computer Architecture Letters* 17, 1 (2018), 96–99.

[16] Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu. 2013. An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms. In *ACM SIGARCH Computer Architecture News*, Vol. 41. ACM, 60–71.

[17] Mathias Payer. 2016. HexPADS: a platform to detect âĂIJstealthâĂİ attacks. In *International Symposium on Engineering Secure Software and Systems*. Springer, 138–154.

[18] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. 2016. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks.. In *USENIX Security Symposium*. 565–581.

[19] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. 2016. Flip Feng Shui: Hammering a Needle in the Software Stack.. In *USENIX Security symposium*. 1–18.

[20] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM rowhammer bug to gain kernel privileges. *Black Hat* 15 (2015).

[21] Seyed Mohammad Seyedzadeh, Alex K Jones, and Rami Melhem. 2017. Counter-based tree structure for row hammering mitigation in DRAM. *IEEE Computer Architecture Letters* 16, 1 (2017), 18–21.

[22] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Defeating software mitigations against rowhammer: a surgical precision hammer. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 47–66.

[23] Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Throwhammer: Rowhammer Attacks over the Network and Defenses. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 213–226. https://www.usenix.org/conference/atc18/presentation/tatar

[24] Ad J Van De Goor and Ivo Schanstra. 2002. Address and data scrambling: Causes and impact on memory tests. In *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on*. IEEE, 128–136.

[25] Victor Van Der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. 2016. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 1675–1689.

[26] Wired. 2018. AN INGENIOUS DATA HACK IS MORE DANGEROUS THAN ANYONE FEARED. (2018). https://www.wired.com/story/rowhammer-ecc-memory-data-hack/

[27] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. 2016. One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation.. In *USENIX Security Symposium*. 19–35.