# Leveraging EM Side-Channel Information to Detect Rowhammer Attacks

Zhenkai Zhang*§, Zihao Zhan†§, Daniel Balasubramanian†, Bo Li‡, Peter Volgyesi†, Xenofon Koutsoukos†
*Texas Tech University, †Vanderbilt University, ‡University of Illinois at Urbana-Champaign
§(The first two authors contributed equally to the paper)

***Abstract*—The rowhammer bug belongs to software-induced hardware faults, and has been exploited to form a wide range of powerful rowhammer attacks. Yet, how to effectively detect such attacks remains a challenging problem. In this paper, we propose a novel approach named RADAR (Rowhammer Attack Detection via A Radio) that leverages certain electromagnetic (EM) signals to detect rowhammer attacks. In particular, we have found that there are recognizable hammering-correlated sideband patterns in the spectrum of the DRAM clock signal. As such patterns are inevitable physical side effects of hammering the DRAM, they can "expose" any potential rowhammer attacks including the extremely elusive ones hidden inside encrypted and isolated environments like Intel SGX enclaves. However, the patterns of interest may become unapparent due to the common use of spread-spectrum clocking (SSC) in computer systems. We propose a de-spreading method that can reassemble the hammering-correlated sideband patterns scattered by SSC. Using a common classification technique, we can achieve both effective and robust detection-based defense against rowhammer attacks, as evaluated on a RADAR prototype under various scenarios. In addition, our RADAR does not impose any performance overhead on the protected system. There has been little prior work that uses physical side-channel information to perform rowhammer defenses, and to the best of our knowledge, this is the first investigation on leveraging EM side-channel information for this purpose.**

## I. Introduction

As a fundamental requirement for implementing security measures, memory protection prevents a process from modifying memory it does not own. However, this essential protection becomes at stake due to the discovery of a vulnerability, known as the rowhammer bug [34], in the underlying dynamic random-access memory (DRAM). The rowhammer bug belongs to the class of software-induced hardware faults, which makes unauthorized data modifications possible.

The existence of the rowhammer bug has been reported in numerous DRAM chips of DDR3 and DDR4 [34], [39]. Since its discovery, this hardware vulnerability has been continuously exploited to form a wide range of powerful rowhammer attacks. Examples of such attacks include sandbox escaping [26], [49], [54], privilege escalation [8], [25], [26], [54], [61], [64], cryptography subversion [7], [51], denial-of-service [33], [42], [66], and even confidentiality breaching [38]. Furthermore, rowhammer attacks have been effectively demonstrated in the presence of ECC mechanism [16] as well as in the context of only sending network packets [42], [60].

In response, many defense techniques against rowhammer attacks have been proposed in recent years, including several detection-based approaches [4], [27], [30], [32], [46]. Unfortunately, as more sophisticated rowhammer attacks are developed, the effectiveness of detection-based rowhammer defenses becomes questionable. As demonstrated in [25], all of the practical rowhammer attack detection approaches can be circumvented. In particular, by abusing the Intel SGX technology and closed-page memory controller policy, rowhammer attack detection based on either static analysis [32] or dynamic monitoring [4], [27], [30], [46] will become ineffective.

In this paper, we introduce a new direction to addressing the problem of rowhammer attack detection. Specifically, we propose to leverage certain electromagnetic (EM) emanations to effectively and robustly detect rowhammer attacks. EM side-channel information is capable of revealing much knowledge about the ongoing activity in a computing device, and it has been extensively exploited to breach confidentiality [2], [3], [18], [20]–[23], [31], [37], [50]. However, it has been realized that, as a double-edged sword, such side-channel information can also be used to help build security defenses [28], [45]. Following this line, for the first time, we utilize EM side-channel information to our advantage for rowhammer attack detection. Because EM emanations are inevitably issued during any computation and can be hardly suppressed by outside adversaries, our proposed approach can detect any potential rowhammer attacks including the extremely elusive ones that are hidden inside attacker-controlled SGX enclaves. Moreover, our detection approach does not degrade the performance or resource utilization of the system under protection.

The main contributions of this paper are as follows:

- We study the correlation between certain EM emanations and rowhammer attacks, based on which we propose a systematic rowhammer attack detection approach named RADAR (Rowhammer Attack Detection via A Radio).
- We propose the first approach to reversing the scattering effect of spread-spectrum clocking on EM side-channel information issued from high-frequency clocks in a computing device.
- We have implemented a RADAR prototype using a *$299* software-defined radio device, and we evaluate the effectiveness and robustness of our EM-based rowhammer attack detection under different scenarios.

There has been little prior work that uses physical side-channel information to perform rowhammer defenses, and to the best of our knowledge, this is the first investigation on leveraging EM side-channel information for this purpose.

The rest of this paper is organized as follows: Section II briefly sets the background; Section III formulates the threat model; Section IV presents a new direction to rowhammer attack detection; Section V studies the correlation between EM side-channel information and rowhammer attacks; Section VI proposes our RADAR system, which can achieve rowhammer attack detection in a non-intrusive manner; Section VII evaluates the proposed RADAR system; Section VIII gives the related work; and Section IX concludes this paper.

## II. Background

In this section, we provide some background information on DRAM organization, the rowhammer bug, and rowhammer attacks. Moreover, we briefly present the physical side effects leveraged in this paper, namely the EM emanations.

### A. DRAM Organization

Modern computing devices use DRAM as the main memory. For better memory bandwidth, DRAM is often partitioned into multiple channels. Each channel may be associated with multiple dual in-line memory modules (DIMMs). Each DIMM has one or more ranks (e.g., modern DIMMs can be single-/dual-/quad-/octal-rank), and each rank has multiple banks (e.g., normally there are 8 banks for DDR3 and 16 banks for DDR4). As depicted in Fig. 1, each bank can be viewed as a two-dimensional array of memory words, organized in rows and columns. The size of the memory word depends on the data bus width, and decides how many cells are needed to store its content (e.g., 64 cells are needed to store a 64-bit memory word). Each cell consists of a capacitor and a transistor, where the capacitor is either charged or uncharged to represent a binary value[1], and the transistor is used to access the capacitor. In each bank, there is also a row buffer, which can hold the contents of a single row. To access a cell, the corresponding row has to be activated first to put the contents of the row into the row buffer, and then the access is served from the row buffer. An activated row remains in the row buffer until being closed by the memory controller, and before then, consecutive accesses to that row will be served directly from the row buffer. Depending on what memory controller policy is being used, an active row can be closed due to different reasons: If the memory controller uses an open-page policy, the active row will not be closed until a different row in the same bank is accessed; and such a causal event is often called a row conflict. On the other hand, if a closed-page policy is employed, the memory controller will proactively close the row [25], [42].

Note that a DRAM cell can only keep its charged state for a short period of time, as its capacitor leaks its charge over time. In order to prevent any data loss, the cells must be refreshed regularly. DDR3 and DDR4 specifications require that the refresh interval must not be longer than 64ms. Normally, the refresh interval is between 32ms to 64ms.

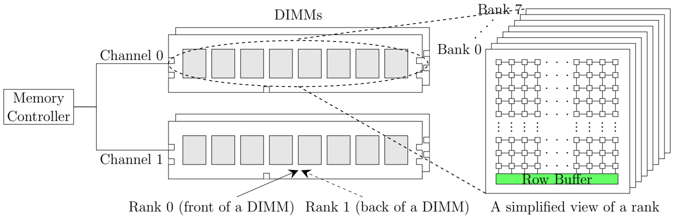


Fig. 1. A representative DRAM architecture (two channels and four dual-rank DIMMs). A rank consists of all the chips on the same side (front or back) of a DIMM.

### B. The Rowhammer Bug and Hammering

As DRAM becomes denser, the capacitor in a cell becomes smaller and the voltage margin separating '0' and '1' becomes lower, which unfortunately have reduced the overall DRAM reliability [44]. First thoroughly studied in [34], the rowhammer bug has become a well-known DRAM reliability issue: When a DRAM row is repeatedly activated and closed (namely, the row is hammered) enough times within a refresh interval, one or more bits in its physically adjacent rows can be flipped to the opposite value[2]. Usually, a row that is hammered is referred to as an aggressor row, and a row that has flipped bits is called a victim row.

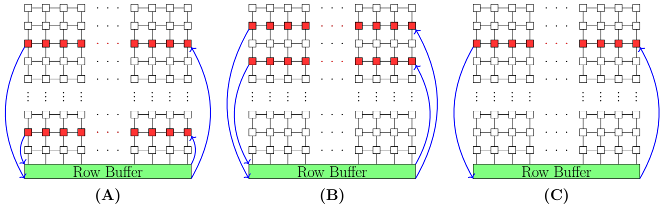


Fig. 2. Three possible hammering techniques in the literature: (A) single-sided hammering [34]; (B) double-sided hammering [54]; and (C) one-location hammering [25].

Since many of the memory controllers use an open-page policy, to trigger the rowhammer bug on such systems, two aggressor rows in the same bank need to be alternately activated. Consequently, the row buffer of that bank will alternately hold the contents of these two aggressor rows. If the two aggressor rows are not intentionally chosen to "sandwich" a row, it is termed as single-sided hammering, as shown in Fig. 2 (A). On the other hand, if the two aggressor rows are selected to specifically lie on both sides of another row, it is called double-sided hammering, as shown in Fig. 2 (B). As demonstrated in practice, double-sided hammering is much more effective and efficient than single-sided hammering [54].

[1] Depending on the implementation, some cells use the charged state to represent '1', while other cells use the charged state to represent '0'.

[2] The large current coupled with toggling the activation of a row repeatedly and rapidly accelerates the discharge rate of cells in the physically adjacent rows. Before the next refresh, if too much charge in a cell has been leaked, the stored bit information will be lost, namely the bit is flipped from 1 to 0 or from 0 to 1, depending on whether 1 or 0 is represented by the charged state.

Some new memory controllers may use a closed-page (or hybrid) policy, and in such cases even one aggressor row is sufficient to induce bit flips around the row, which is called one-location hammering [25], as shown in Fig. 2 (C).

### C. Rowhammer Attacks

Because the rowhammer bug allows one to modify the contents of a DRAM row without explicit permission, severe security risks are posed. Since the discovery of this devastating hardware vulnerability, many powerful attack vectors have been developed by exploiting the rowhammer bug to compromise the security defenses of a system. Usually, a rowhammer attack consists of three basic phases:

1) **Exploration phase**. In the first phase, the attacker intensively hammers the DRAM and searches for exploitable bit flips. The prerequisite for performing this phase is to design approaches used to trigger the rowhammer bug on the targeted system. More details will be described below.
2) **Manipulation phase**. In the second phase, the attacker steers the targeted security-critical data to the vulnerable memory location that has the exploitable bit flips found in the first phase. There are several approaches developed for this specific task, including memory spraying [54], memory grooming [61], memory waylaying [25], and memory ambush [13].
3) **Exploitation phase**. Once the security-critical data has been placed at the vulnerable location, in the third phase, the attacker triggers the rowhammer bug again to flip the bit(s), which achieves the final compromise.

When designing an approach to triggering the rowhammer bug on the targeted system, several technical challenges need to be overcome. One challenge is the lack of address mapping information, including both virtual-to-physical and physical-to-DRAM, which leads to some approaches using inefficient random testing [49], [54]. While memory deduplication can be exploited to ease this challenge [8], attackers have tried to recover such mapping information, especially by reverse engineering the physical-to-DRAM address mapping [47], [59], [64], for more efficient and effective double-sided hammering.

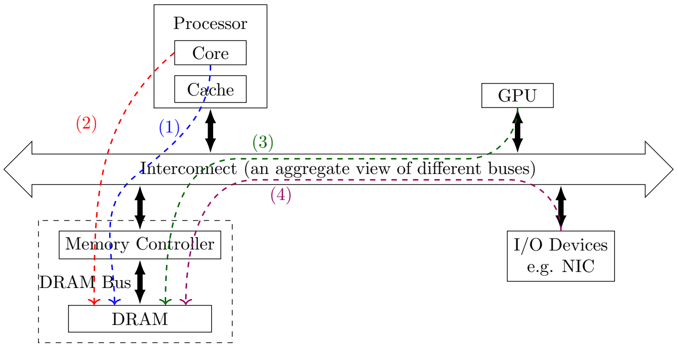


Fig. 3. Different types of techniques for rapidly and repeatedly access the same locations in DRAM.

The other challenge is how to access the underlying DRAM quickly enough. To trigger the rowhammer bug, the same location in DRAM has to be accessed rapidly; otherwise, even if the DRAM were extremely vulnerable to hammering, one would still not be able to exploit the bug for a successful rowhammer attack. However, due to the presence of the caches, most of the memory accesses to the same location can hardly reach the DRAM. (This is why the rowhammer bug is seldom triggered during the ordinary use of a computing device, even though the underlying DRAM might be extremely vulnerable.) Over the past few years, several techniques have been developed to overcome this challenge (e.g., to circumvent the effect of the caches). Fig. 3 shows a typical computing platform, and each of the dashed lines in the figure represents a possible path taken to enable fast access to the same location in DRAM: (1) flushing or evicting CPU caches [1], [4], [26], [34]; (2) bypassing CPU caches [49], [61]; (3) evicting GPU caches [19]; and (4) maneuvering DMA buffers from I/O devices [60].

### D. EM Emanations

Because the electric current in the circuitry of a computing device varies with time, EM emanations arise. As inevitable physical side effects, EM emanations carry information about the underlying electronic activities, which can be linked with certain high-level activities such as which instructions or loops are being executed. Thus, this information leakage has been exploited to facilitate certain attacks, e.g., stealing cryptographic keys [3], [21]–[23], or inferring privacy [18], [37]. Yet, other than being exploited for side-channel attacks, EM emanations have also been used to track code execution for ensuring control-flow integrity [28], [45] or profiling [10], [56].

The generated EM emanations are distributed widely on the spectrum. Although the sources of many of these emanations are unknown, a few of them are in fact easy to determine, e.g., the ones created by well-known periodic activities like clocking and DRAM refreshing. The EM-emanated signals created by these periodic activities are also strong and can propagate far. Interestingly, some of these signals may be unintentionally modulated by other activities in the form of amplitude modulation (AM) or frequency modulation (FM) [12], [48]. For example, signals emanated from switching voltage regulators may be AM-modulated by activities in the circuits they power, and signals generated by periodic DRAM refreshes may be AM-modulated by memory access activities [12]. Therefore, these signals act as carrier signals that convey information about the modulating activities.

## III. Threat Model

Assume an attacker has access to a system equipped with DDR3 or DDR4 memory modules. The attacker attempts to find out whether the DRAM of the system has the rowhammer bug, and if so, the attacker also scans for exploitable bit flips for a subsequent attack. Given the very low probability that exploitable bit flips can be found in the first few trials, the attacker needs to intensively hammer the DRAM for such bit flips. In this paper, we assume that the attacker will either utilize special instructions such as `clflush` (namely flushing the cache) or `movnti` (namely bypassing the cache), or constantly evict the corresponding cache lines, to achieve either double-sided, single-sided, or one-location hammering. To circumvent simple detections, the attacker may manipulate the system to run an SGX enclave, inside which the malicious activities are performed.

In this paper, we mainly focus on computing platforms that use DDR (instead of low-power DDR) and are seldom moved on a daily basis, such as personal computers and workstations. Although mobile/embedded systems are excluded, this actually includes most of the currently vulnerable systems that have a much wider rowhammer attack surface than mobile/embedded systems and are harder to protect [19], [61], [62].

Another assumption is that the attacker is not able to physically interfere with the EM emanations generated by the system, e.g., she cannot place a high-power radio transmitter nearby the target system and use it to jam the frequency band of interest. Note, however, that this assumption does not limit the applicability of our proposed method at all, due to the fact that rowhammer attackers rarely need or have physical access to the target systems.

## IV. New Direction to Rowhammer Detection

Under the stated threat model, developing effective detection-based defense techniques against the possible rowhammer attacks remains an open research problem [13], [25], [59]. In this section, we discuss why leveraging physical side-channel information, EM emanations in particular, can provide a feasible solution to this problem.

As we know, to effectively and robustly detect any type of attacks, we need to discover and rely on information that has a strong correlation with these attacks but can hardly be tampered or concealed by any attacker-controllable running program. Since physical side-channel information leaks much fine-grained knowledge about system activities and can hardly be corrupted by remote adversaries in reality, we can leverage such information to help detect anomalies, including rowhammer attacks.

A variety of physical side effects are inevitably generated during any activity of a computer system. For instance, power is consumed, heat is issued, EM signals are radiated, and even sound or light may be produced. Some of these side effects may have strong correlations with the operations of certain hardware components. As we can observe from Fig. 3, the memory controller, memory bus, and DRAM modules are the three hardware components that are always involved in a rowhammer attack, no matter which technique is employed to hammer the DRAM. Thus, we should primarily consider the physical side-channel information that is strongly correlated with the operations of these three hardware components. In this paper, we argue that we can leverage EM side-channel information for this purpose.

The rationale for leveraging EM side-channel information to detect rowhammer attacks lies in the following facts:

- As mentioned in Section II, EM emanations are inevitable physical side effects during any computation, issued both intentionally and unintentionally [2], [12], [48], [65].
- EM emanations can be measured in a contactless manner (e.g., via a radio device). This removes the need for unrealistic hardware modifications to guarantee practicality.
- Compared to other physical side-channel information like power consumption, EM emanations can provide more fine-grained and niche-targeting insight into an activity.
- Most importantly, as illustrated in [11], [12], [48], [65], rich information about memory activities can be found in some EM emanations.

In the following, we will present our investigation on finding information correlated with a potential rowhammer attack in EM emanations. Additionally, we will describe our system design that uses simple and affordable measurements to achieve an effective detection-based rowhammer attack defense. In the course of our discussion, we will use EM emanations and EM-emanated signals interchangeably.

## V. Finding Hammering Information in EM Side-Channel Emanations

As mentioned in Section II, rowhammer attacks need a hammering process to tentatively trigger the rowhammer bug, and then search for exploitable bit flips. The whole hammering process consists of many hammering attempts, each of which requires a large amount of toggling the activation of aggressor row(s) within a short period of time. In the following discussion, we will call such an activation toggling as a hammering iteration. Therefore, there is a fast and nearly-regular switching behavior in rowhammer attacks in nature. As a consequence, when the three aforementioned hardware components (namely, the memory controller, memory bus, and DRAM modules) are stressed by hammering, the information about the hammering activity is very likely carried by some EM-emanated signals at certain frequencies.

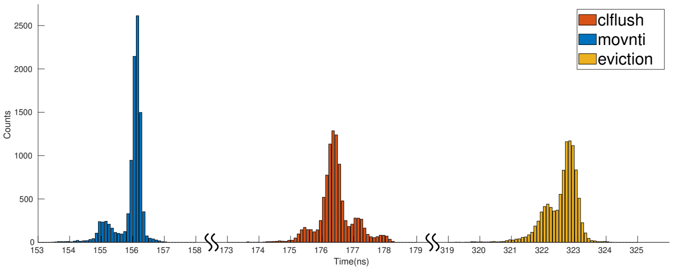


Fig. 4. The timing distributions of 10,000 hammering iterations in terms of approaches using `clflush`, `movnti`, and eviction.

Theoretically, such signals can be in any place of the EM spectrum, but most likely, they should be correlated with the frequency of the switching behavior. However, we do not specifically know the switching frequency, because there can be multiple approaches to triggering the rowhammer bug on the same machine, each of which may have different computational overhead in its hammering iteration. Moreover, the time consumed in each hammering iteration can hardly be identical, which will result in a small range of switching frequencies in the context of a single hammering attempt. For example, for each of the three most commonly used approaches, which are flushing the cache, bypassing the cache, and evicting the cache, Fig. 4 shows the corresponding timing distribution of 10,000 hammering iterations. The timing measurements are performed on a platform equipped with an Intel Haswell G3258 processor and 8 GiB DDR3-1333 DRAM. In the rest of this paper, unless stated otherwise, this is the platform used in the examples.

Nevertheless, the possible frequencies of this switching behavior are bounded to some extent. Because the rowhammer bug cannot be triggered if there are not enough times of hammering iterations in between two refreshes, the frequency has a lower bound. Obviously, the frequency must also have an upper bound, because memory accesses cannot be arbitrarily fast. (In effect, if the memory controller uses an open-page policy, there exists an even tighter upper bound due to row conflicts.)

### A. Direct EM Emanations

Given the fast switching behavior in a hammering attempt (e.g., the row buffer in a bank is repeatedly opened and closed along with discharging and charging the aggressor rows), we conjecture that there should be clear EM-emanated signals at the possible switching frequencies. Therefore, we are tempted to identify these signals directly.

However, there are some challenges and concerns in measuring such direct EM emanations, even though their existence is plausible: First, the switching periods are normally in the range of one hundred to several hundreds of nanoseconds, and therefore the corresponding frequencies are in a rather low spectral range, where much noise exists due to radio stations, appliances, and other sources. Second, these signals may be very weak, and measuring such long wavelength weak signals may require a physically large antenna or a special antenna whose return loss is minimal around the frequencies of interest.

In our experiments, we did not observe any EM-emanated signal that is strongly correlated with the hammering switching behavior in the frequency range of interest. Granted, we used only a software-defined radio with a telescopic whip antenna to try to capture such signals. Therefore, it may be possible to find some signals of interest if using some lab-grade instruments and carefully placing some customized EM probes close to the chips. However, if such equipment is required, the practicality of our detection approach will be decreased. For our purposes, we need to leverage other possible EM emanations containing hammering attempt information that can also be easily measured.

### B. AM-Modulated EM Emanations

As we know, many system modules like clocks and voltage regulators intrinsically create EM-emanated periodic signals. According to the study in [12], some of these periodic signals will be AM-modulated by certain types of activities, and thus information about the corresponding activities can be found in those modulated signals. Moreover, such signals are relatively strong and can propagate far, which lowers the requirements for measuring them. Inspired by this study, we investigate whether it is possible to find information about hammering attempts in some of such AM-modulated signals. As an educated guess, the hammering activity most likely modulates some periodic carrier signals generated in the aforementioned three hardware components.

As illustrated in [12], the strength of the EM emanations generated by the DRAM clock varies when the amount of activities driven by the clock changes, namely the emanations at the DRAM clock frequency will be AM-modulated by the DRAM activities. Therefore, our investigation will focus on finding hammering attempt information in the AM-modulated DRAM clock signals.

AM-modulation has a long history and is well understood. We know that when a carrier signal is AM-modulated, there are sidebands appearing on both sides of the carrier frequency in the spectrum, and each sideband is a mirror-image of the other relative to the carrier. These upper and lower sidebands correspond to the spectrum of the modulating activity, namely each modulating frequency will be present in each sideband.

Since nearly-regular and lasting switching behavior is associated with a hammering attempt, if the DRAM clock signal carries such hammering attempt information through AM-modulation, we expect to identify that information via some distinctive frequency patterns in the upper and lower sidebands of the modulated DRAM clock signal. We have conducted a large number of experiments that have verified the feasibility of this idea. For instance, Fig. 5 shows the power spectra of the DRAM clock signal measured using a software-defined radio under six scenarios: The first scenario (A) is the simplest one, in which only the system background tasks are running. The following two scenarios represent some common uses of a computer system, which are (B) playing a video and (C) browsing web pages. The last three scenarios are to (single-sided) hammer the underlying DRAM by means of the three most commonly used approaches: (D) using `clflush` instruction to flush the cache, (E) using non-temporal store `movnti` instruction to bypass the cache, and (F) loading from congruent addresses to evict the cache.

Given that DDR3-1333 memory modules are used in this example, the DRAM clock frequency is around 666∼668 MHz. On our platform, it is at 667.85 MHz, which corresponds to the tallest central spike in each spectrum of Fig. 5. Note that, to avoid a cluttered discussion, we turned off the spread spectrum clocking feature in the BIOS for now (the motherboard used in this example is ASUS Z87-A), and the problem caused by this feature as well as our solution will be discussed later.

From Fig. 5, we can observe distinguishable sideband patterns in the spectra when the underlying DRAM is being hammered, namely there are noticeable “bumps” located on both left and right sides of the central spike, which are circled

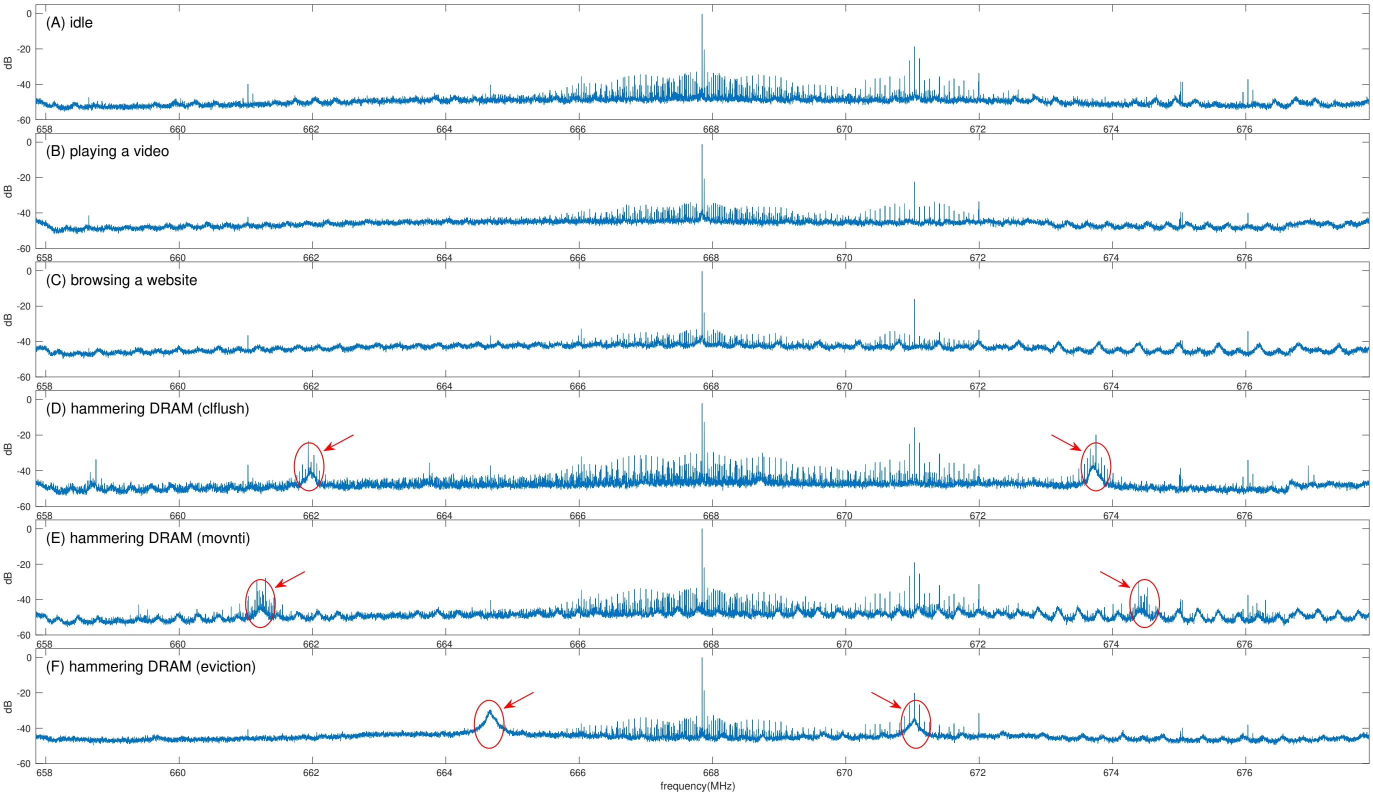


Fig. 5. The power spectra under six scenarios. Note that the vertical axis is on a logarithmic dB scale. Each spectrum is derived by averaging 78 FFTs of 16,384 values with 50% overlap sampled in 25 MHz over 32 ms.

and pointed to by arrows in Fig. 5 (D), (E), and (F). By referring to Fig. 4, we can actually find the relation between the times spent in hammering iterations and the frequencies where the sideband patterns of interest are located. Take the approach using `movnti` for an example. From Fig. 4, we can see the dominant period of hammering iterations is around 156 ns. As shown in Fig. 5 (E), the circled lower sideband patterns are at about 661.4 MHz (i.e., 667.85 MHz - 1000/156 MHz), and the circled upper ones are at about 674.3 MHz (i.e., 667.85 MHz + 1000/156 MHz). These hammering-correlated sideband patterns conform to the effect of AM-modulation, which illustrates that we can find hammering attempt information in the modulated DRAM clock signal. Furthermore, we can notice that the "bumps" in Fig. 5 (D) and (F) are slightly wider than that in Fig. 5 (E). This is because the timing variances when using `clflush` and eviction are larger than that when using `movnti`, as shown in Fig. 4.

Note that, since multimedia like videos is non-temporal data (namely data needed in the near future is not in the cache), there is *a large number of DRAM accesses* in the scenario (B). However, as shown in Fig. 5 (B), no obvious patterns of interest arise. Thus, it indicates that the presence of massive cache misses or DRAM accesses is **only a necessary but not a sufficient condition** for generating hammering-correlated sideband patterns. Normally, it is rare that a benign program generates high rate and periodic cache misses for more than 30 ms.

Furthermore, we can still observe these sideband patterns even after introducing some disturbance into the periodic behavior of a hammering attempt. (In such a case, the variance of hammering period is increased, so the "bumps" become wider and lower.) In other words, it is hard to conceal such patterns while maintaining sufficiently fast toggling rate of aggressor rows to trigger the rowhammer bug. In Section VII-D, we will illustrate some of the experimental results related to this random delay addition.

### C. *Spread-Spectrum Clocking*

One major difficulty in robustly detecting hammering-correlated sideband patterns is created by spread spectrum clocking (SSC), which has been commonly used in electronic products like computer systems for meeting electromagnetic compatibility (EMC) regulations. EMC standards impose allowable limits on the EM-emanated signal energy at any frequency above 30 MHz, and many high-frequency clock signals in a computer system (e.g., the DRAM clock) are strong enough to violate such legal limits. To achieve EMC, SSC uses FM-modulation to vary the clock frequency over a range so that the time spent by the clock signal at a particular frequency is reduced and the energy is spread over that range of frequencies [29].

Under the situation in which the underlying DRAM is being hammered, Fig. 6 (A) demonstrates the problem when SSC is

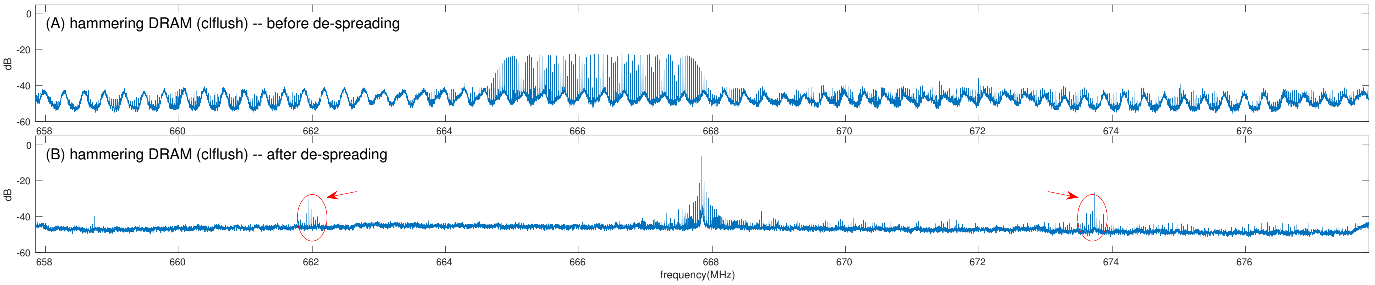


Fig. 6. The power spectra under hammering by means of `clflush` before and after de-spreading. Each spectrum is derived by averaging 78 FFTs of 16,384 values with 50% overlap sampled in 25 MHz over 32 ms.

turned on (which is the default option in most BIOSes). As we can observe in the spectrum, instead of a single spike at 667.85 MHz, the clock frequency now ranges from 664.85 MHz to 667.85 MHz as a consequence of SSC. Compared with the SSC-off clock signal power, when SSC is turned on, the signal power is indeed significantly reduced (more than 15 dB in the given example). However, we find that the frequency patterns of interest to our rowhammer attack detection are also attenuated due to SSC, such that the hammering-correlated sideband patterns become unrecognizable.

To overcome this problem, we need to de-spread the energy in the signal. The details of our de-spreading process will be described in the next section. Here, our aim is to show that hammering attempt information can be found in the EM-emanated DRAM clock signal. Fig. 6 (B) shows the power spectrum of the measured signal after de-spreading. Compared with Fig. 6 (A) which shows the spectrum of the original signal without de-spreading, we can clearly notice that the sideband patterns used for rowhammer attack detection reappear. Therefore, we conclude that information correlated with a potential rowhammer attack can be effectively found in certain EM emanations.

## VI. Rowhammer Attack Detection via A Radio

In this section, we propose a rowhammer attack detection system named RADAR (Rowhammer Attack Detection via A Radio), which detects potential rowhammer attacks by identifying hammering-correlated sideband patterns in the AM-modulated DRAM clock signal. The diagram of the proposed RADAR system is depicted in Fig. 7. In the following, we describe each component of our RADAR system.

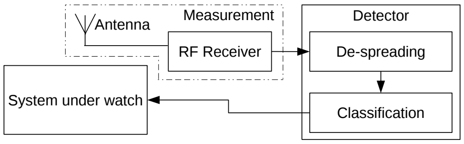


Fig. 7. RADAR system illustration.

### A. Measurement Component

In the first step, we use a measurement device to capture the EM-emanated DRAM clock signal. As the time spent in each hammering iteration could be as low as one hundred nanoseconds (e.g., when using one-location hammering on a high-performance platform), to capture both upper and lower hammering-correlated sideband patterns, the measurement device utilizing quadrature sampling should be able to support at least 20 MHz instantaneous bandwidth. Moreover, the clock frequency of interest may be as low as 400 MHz (e.g., DDR3-800) or as high as 1600 MHz (e.g., DDR4-3200), and thus it is more flexible to have a measurement device that can be tuned to all of the possible frequencies. Fortunately, inexpensive and reliable instruments exist. For simplicity and convenience, in our prototype, we use a software-defined radio for this task.

Because a clock signal is a square wave, there is an infinite number of harmonics in the frequency domain. Here we only consider the first harmonic. If there is too much noise around the fundamental frequency, we may try to rely on some higher-order harmonics.

The antenna used in our system should match the frequency range of interest. Given the possible DRAM clock frequencies, there are many antenna choices. Through experiments, we find that a cheap whip antenna (e.g., a telescopic one or just a piece of wire) suffices. The antenna can be placed inside or outside the case of the computer being monitored, but its position and orientation may need to be fine-tuned for the best performance.

### B. De-Spreading Component

As aforementioned, to robustly detect hammering-correlated sideband patterns, we need to counter the effect of SSC by de-spreading the energy in the measured clock signal. Given a clock signal whose frequency is $f_c$, SSC uses FM-modulation to vary the clock frequency in accordance with a signal $f_m(t)$ that is generated in the SSC hardware chip but undocumented. At time $t$, the instantaneous frequency $f_i(t)$ of the clock signal becomes:

$$f_i(t) = f_c + K f_m(t) \ , \tag{1}$$

where $K$ is some proportionality constant. In an analytic form, the effect of SSC is equivalent to multiplying the clock signal by a complex exponential function $\theta(t)$, which is defined as:

$$\theta(t) = e^{j2\pi \int_0^t K f_m(t) dt} \ , \tag{2}$$

where $j$ denotes $\sqrt{-1}$. If the DRAM is hammered when SSC is on, by reason of AM-modulation, the frequency patterns of interest in the sidebands are also shifted by $Kf_m(t)$ at time $t$. Hence, for the purpose of de-spreading, we just need to estimate $\theta(t)$ and multiply the measured signal by $\theta^{-1}(t)$.

Although the exact mathematical expression of $f_m(t)$ is not available, since we deal with sampled values in the system, as far as we are concerned, only the discrete values of $\theta(t)$ at the points of sampling are needed for de-spreading. We leverage quadrature sampling to measure the SSC-affected clock signal which also centers its spectrum at zero Hz. Let $v_k$ denote the $k$th sample corresponding to the clock signal at a specific time $\tau$, namely

$$v_k = |v_k| e^{j\phi_k} \ , \tag{3}$$

where $|v_k|$ is the magnitude of $v_k$ and $\phi_k$ is the phase angle of $v_k$. Using FM-demodulation, we can acquire:

$$\frac{d\phi_k}{dt} = 2\pi K f_m(\tau) \tag{4}$$

Therefore, at time $\tau$, the instantaneous value of $\theta(t)$ is derived:

$$\theta(\tau) = e^{j2\pi \int_0^\tau K f_m(t) dt} = e^{j(\phi_k + \Theta)} = e^{j\phi_k} e^{j\Theta} \ , \tag{5}$$

where $\Theta$ is a constant phase angle. Although we do not know the exact value of $\Theta$, we may simply assume it is 0, because a non-zero constant phase angle only shifts the signal in the time domain by a constant but does not affect our analysis in the frequency domain at all. Thus, we can simplify Eq. 5 to have it rely on only the values acquired by quadrature sampling:

$$\theta(\tau) = \frac{v_k}{|v_k|} = e^{j\phi_k} \tag{6}$$

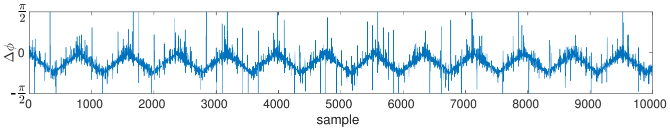


Fig. 8. The phase difference $\phi_k - \phi_{k\text{-}1}$ between successive samples, where $1 \leq k \leq 10,000$. Many of the apparent spikes above 0 are actually caused by *phase wrapping*, i.e., any negative difference in the range of $-\pi > \Delta\phi \geq -2\pi$ will be converted to an angle in the range of $0 < \Delta\phi \leq \pi$.

To de-spread each sampled value of the SSC-affected signal, the need for deriving the value of $\theta(t)$ at that point in time is not desirable, since it is better to derive such values when the amount of DRAM activities is little for less noise. Fortunately, it is known that $f_m(t)$ is a periodic function [29], namely we have $f_m(t) = f_m(t + T_m)$ where $T_m$ is the period of $f_m(t)$. Therefore, even though the sequence of the discrete $\theta(t)$ values may not be periodic[3], its phase difference sequence, which is equivalent to FM-demodulation, must be periodic over $T_m$. In other words, given a sampling frequency $f_s$, we have:

$$\phi_k - \phi_{k\text{-}1} \approx \phi_l - \phi_{l\text{-}1} \ , \text{ where } l = k + \lfloor T_m f_s \rfloor \tag{7}$$

For example, in terms of the platform used in Section V, Fig. 8 shows the phase difference sequence of 10,000 values of $\theta(t)$ over 0.4 ms (i.e., the sampling frequency is 25 MHz). Due to random noises, we can observe singular jumps, although the periodicity is obvious. By averaging the corresponding values, we can effectively remove the noise.

Let $\Delta[0 \ldots N{-}1]$ denote the phase difference sequence over a $T_m$, where $N = \lfloor T_m f_s \rfloor$. Note that we only need to derive $\Delta$ once for each hardware platform, as it is software-independent. When sampling the clock signal for $\Delta$ derivation, we do not have user processes running on the target system, and we also use a bandpass filter to attenuate frequencies outside the range of possible clock frequencies. The sampling frequency should be the same as the one used during detection.

To use $\Delta$ for de-spreading during detection, we first need to achieve $\Delta$ alignment, which is to find a point $p$ in the stream $\mathcal{S}$ of the sampled values such that the phase of $\theta(t)$ varies by $\Delta[0]$ between $\mathcal{S}[p+1]$ and $\mathcal{S}[p]$, by $\Delta[1]$ between $\mathcal{S}[p+2]$ and $\mathcal{S}[p+1]$, and so forth. It is straightforward to see that $\Delta$ alignment is periodic, namely if $\mathcal{S}[p]$ aligns with $\Delta$, $\mathcal{S}[p{+}kN]$ will also align with $\Delta$. (Because it is very likely that $T_m f_s$ is not an integer, strictly speaking, $\Delta$ alignment is quasiperiodic.) Our solution to this problem is based on the fact that a correct alignment leads to the maximum cross-correlation between the entries of $\Delta$ and the phase changes of $N$ successive sampled values, which implies the maximum cross-correlation between the sums of the first $1 \leq m \leq N$ entries of $\Delta$ and the phase changes of the $m$th point relative to the first point. Therefore, given a point $q$, we use the following equation to calculate the cross-correlation:

$$\begin{aligned} \rho(q) &= \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\frac{\mathcal{S}[q+m]}{|\mathcal{S}[q+m]|}}{\frac{\mathcal{S}[q]}{|\mathcal{S}[q]|}} e^{-j \sum\limits_{i=0}^{m-1} \Delta[i]} \right| \\ &= \left| \frac{1}{\frac{\mathcal{S}[q]}{|\mathcal{S}[q]|}} \right| \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\mathcal{S}[q+m]}{|\mathcal{S}[q+m]|} e^{-j \sum\limits_{i=0}^{m-1} \Delta[i]} \right| \\ &= \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\mathcal{S}[q+m]}{|\mathcal{S}[q+m]|} e^{-j \sum\limits_{i=0}^{m-1} \Delta[i]} \right| \end{aligned} \tag{8}$$

Using Eq. 8, we can start at an arbitrary point $q$ and compute $\rho(q + k(N+1))$ in the $(k+1)$st round, where $k \geq 0$, until the cross-correlation reaches a spike, which signifies we have found $\Delta$ alignment in that round. (Again, if $T_m f_s$ is an integer, we will need at most $N$ rounds to find $\Delta$ alignment. However, it is very likely that $T_m f_s$ is not an integer, and we may need more than $N$ rounds.) As an example, Fig. 9 shows the cross-correlation results in the first 800 rounds with respect to the example given in Fig. 6, and we can clearly see that the initial $\Delta$ alignment is found in the 266th round.

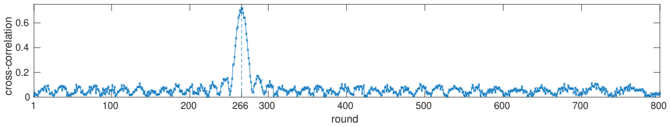


Fig. 9. Using cross-correlation to achieve the initial $\Delta$ alignment.

[3] If the integration of $Kf_m(t)$ over $T_m$ is an integer, $\theta(t)$ is periodic over $T_m$. If it is not an integer but a rational value, $\theta(t)$ is still periodic. If the integration is an irrational value, $\theta(t)$ is aperiodic.

After we have found the initial $\Delta$ alignment, say, $\mathcal{S}[p]$ aligns with $\Delta$, for each of the next $i \geq 1$ values after $\mathcal{S}[p]$, we use the following process to obtain the de-spread sequence $\mathcal{D}$:

$$\begin{aligned} \mathcal{D}[p+i] &= \mathcal{S}[p+i]e^{-j\varphi_{p+i}} \ , \text{ where} \\ \varphi_{p+i} &= \varphi_{p+i\text{-}1} + \Delta[(i-1) \bmod N] \ , \text{ and } \varphi_p = 0 \end{aligned} \tag{9}$$

As the rounding error introduced by $\lfloor T_m f_s \rfloor$ when $T_m f_s$ is not an integer will slowly make the alignment drift away, we need to periodically calibrate the alignment. Since the floor is taken, the accumulated error will reach a point where $\mathcal{S}[p+kN+1]$ aligns with $\Delta$, instead of $\mathcal{S}[p+kN]$. We solve this problem by computing two cross-correlations $\rho(p+kN)$ and $\rho(p+kN+1)$ together, where $k \geq 0$, on the fly in the de-spreading process, and introduce a delay to $\Delta$ if $\rho(p+kN+1)$ is larger, which means performing a right circular shift on $\Delta$ by one position, namely, we derive and use a new $\Delta$ as follows:

```
if $\rho(p+kN+1) > \rho(p+kN)$ then
    for $j = 0;\ j < N;\ j = j+1$ do
        $\Delta_{\text{new}}[(j+1) \bmod N] = \Delta[j]$;
```

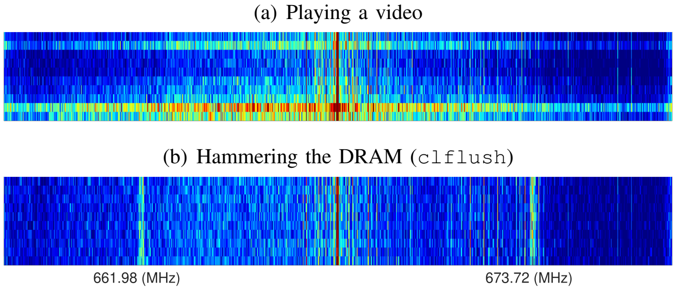
Interestingly, de-spreading will inadvertently help reduce background noise unrelated to the EM emanations of interest. This effect is due to the fact that de-spreading will act like SSC on such noise, whose energy will be scattered over a range of frequencies. Because of this, the robustness of the proposed system is increased, as later shown in Section VII-C.

### *C. Classification Component*

Having the stream of samples that are processed according to Eq. 9, we continuously perform FFTs to obtain a sequence of spectra. Each spectrum is treated as a feature vector that is fed into a classifier. Since the hammering-correlated sideband patterns are relatively easy to recognize, it is not hard to train an appropriate model to achieve accurate binary classification. However, if we predict there is a potential rowhammer attack as soon as certain hammering-correlated sideband patterns are identified in a single spectrum, the false positive rate may be high because similar patterns may transiently arise due to some factors like noise.

Recall that a hammering attempt lasts for a period of time, usually tens of milliseconds, which means that the hammering-correlated sideband patterns are very likely present in each spectrum derived within that period of time. On the other hand, if some similar sideband patterns appear in a spectrum, but not due to hammering, they may disappear in the next few spectra. Therefore, we can rely on this temporal dependency to achieve more accurate classification.

The sideband patterns of interest and temporal dependency imply that vertical lines are probably in the spectrogram if some hammering attempts are ongoing. For instance, Fig. 10 shows two spectrograms over 40 ms under two scenarios, and we can clearly observe two vertical stripes in the spectrogram, symmetric about the DRAM clock frequency (represented by the central red stripe), when using `clflush` to hammer the DRAM. In contrast, no such vertical stripes appear in the spectrogram corresponding to the video playing scenario.


(a) Playing a video

(b) Hammering the DRAM (`clflush`)

Fig. 10. Spectrogram patterns of different activities.

Since such patterns are local and share the property of space invariance, we decide to use convolutional neural network (CNN) that can automatically extract these local features and perform classification on the basis of them. The input to CNN is a *magnitude* spectrogram that is a sequence of $w$ magnitude spectra. The output from CNN is the probability of the input being in the hammering class after applying a softmax function. We use a sliding window of size $w$, whose stride is $s$ to successively feed the inputs. Note that $w$ and $s$ depend on several factors including sampling frequency, computational capacity, and classification accuracy. The values used in our RADAR prototype are described in Section VII-A.

We find that it is important to normalize the magnitude of each point in the spectrogram prior to training and classification. For each point, we normalize its magnitude by subtracting the mean and dividing by the standard deviation of the magnitudes of all the points in that spectrogram (i.e., using instance normalization). Note that we simply set the magnitudes of the points within $\pm 0.05$ MHz of the clock frequency to zero, namely, we zero out the central red stripes in Fig. 10. The rationale behind this is twofold. First, the power levels around the DRAM clock frequency totally dominate (e.g., more than 20 or even 40 dB as shown in Fig. 5), which can significantly affect the results of normalization. Second, we do not lose any useful information for our detection purpose, because the sideband patterns of interest induced by actual hammering attempts will not fall in this range; otherwise it will be too slow to trigger the rowhammer bug.

### *D. Discussion on the Use of Detection Information*

When suspicious sideband patterns are recognized, the detector will notify the system under watch that a rowhammer attack may be ongoing. To this end, the detector should be connected to the system through some standard communication interface like USB, and will send notification messages when potential hammering attempts are detected.

Upon receiving such a message, we may try to prevent the system from being compromised in a very simple fashion, which is to terminate all of the untrusted processes or the processes belonging to untrusted users. Although this approach can promptly thwart potential rowhammer attacks, it is overly conservative, since many non-malicious processes are also terminated. Alternatively, we may leverage the scheduling information to narrow down the list of suspicious processes (e.g., we can select the untrusted processes that were scheduled to run in the last 100 ms as suspicious ones).

As a matter of fact, it is very likely that tens of (or even hundreds of) hammering attempts are needed before finding some exploitable bit flips, especially if the underlying DRAM modules are not overly vulnerable (e.g., the number of bit flips is below a threshold during some test). In such scenarios, we can try to pinpoint the malicious process by individually scheduling each suspicious process to see which one can raise the alarm again. Of course, if the system under watch is very security-sensitive and/or the underlying DRAM is very vulnerable, we may wish to terminate all of the untrusted running processes as soon as a notification message from the detector is received.

## VII. Evaluation

We have implemented a RADAR prototype to demonstrate its practicality, and have evaluated it on four platforms that are summarized in Tab. I. As stated in Section III, an attacker has various choices of hammering techniques for rowhammer attacks. We show that our approach can protect a system from all these possible techniques. Before presenting the evaluation results, we will first describe our prototype in more detail.

TABLE I
Platforms on which our prototype is evaluated.

| Platform | Motherboard | CPU | Memory |
|---|---|---|---|
| A | Asus Z87-A | Intel G3258 | 8 GiB Hynix DDR3-1333 |
| B | Dell OptiPlex 990 | Intel i7-2600K | 8 GiB Samsung DDR3-1333 |
| C | Alienware Aurora R7 | Intel i7-8700K | 16 GiB Micron DDR4-2666 |
| D | Asus ROG Strix B350-F | AMD Ryzen 7 1800X | 32 GiB Samsung DDR4-2133 |

### A. Prototype of RADAR

We use a software-defined radio, LimeSDR, to acquire the EM-emanated DRAM clock signal data. The bandwidth we need is 25 MHz, and LimeSDR can provide 61.44 MHz RF bandwidth in the frequency range of 100 kHz – 3.8 GHz [41], which is more than sufficient for our needs. A LimeSDR costs *$299*. In fact, we need only an RF receiver instead of a full-duplex SDR, and thus a customized device can even be much cheaper. We simply use a 20 cm telescopic antenna or a self-built one from two pieces of 7.5 cm wire that can be easily placed inside a computer case. (Appendix B gives more details on antennas as well as their placement.)

For rapid prototyping, we use a dedicated computer to serve as the detector, on which the de-spreading and classification components run. The de-spreading component is implemented as a module of the GNU radio framework. The classification component is implemented under the PyTorch framework and integrated into the GNU radio using the C++ interface. (Note that using a dedicated computer is only for proof-of-concept. The whole detector can be implemented on an FPGA, which will be our future work.) The detector is connected to the system under watch via the USB interface[4].

We train a 3-layer CNN model using the positive and negative examples collected from the four platforms listed in Tab. I. Each platform contributes 5,000 positive examples as well as 5,000 negative ones. A standalone program is used to generate positive examples, which randomly selects aggressor rows and hammers 1/3 of them using `clflush`, another 1/3 of them using `movnti`, and the rest of them using eviction; and the negative examples are collected at random during the daily use of these platforms (e.g., browsing some web pages). Although not thoroughly investigated in this paper, we conjecture that there can be a generic model, which is trained using data from some representative platforms having different factors like case sizes and DRAM clock speeds. To preliminarily prove this, we evaluate the trained model on several additional platforms, whose data has never participated in the original training. The results are reported in Appendix C, which show that reliable detection can still be achieved on these unseen platforms. We leave the comprehensive study to our future work.

Given the 25 MHz sampling frequency[5], we perform 8192-point FFTs that can provide about 3 KHz frequency resolution and spans only 327.68 $\mu$s. The FFT overlap we use is 50%, and it means an FFT is performed with 4096 new points and 4096 previous points. To overcome noise, we average 20 spectra to derive a single spectrum, i.e., each averaged spectrum spans about 3.3 ms. For classification, we set the sliding window size to 12 and the stride to 1. In other words, the classification runs every 3.3 ms on the spectrogram of the last 40 ms.

Due to the tight timing constraints, we need to minimize the performance overhead incurred by de-spreading and classification. To achieve this, we optimize them by taking advantage of data-level parallelism. When implementing the de-spreading component, we use the AVX-256 SIMD instructions, whenever possible, to process multiple sampled values at a time. In terms of classification, we fall our back on GPU to provide sufficient acceleration. As mentioned before, these two components can be implemented on an FPGA, since FPGAs are truly parallel in nature. Our future work includes implementing the whole detector on the FPGA of LimeSDR.

### B. Effectiveness of RADAR

```
| mov (X), %0   | mov (X), %0   | movnti %0, (X) | movnti %0, (X) | evict (X)     | mov (X), %0   |
| mov (Y), %0   | mov (Y), %0   | movnti %0, (Y) | movnti %0, (Y) | evict (Y)     | clflush (X)   |
| clflush (X)   | clflush (X)   | mov (X), %0    | mov (X), %0    | mov (X), %0   |               |
| clflush (Y)   | clflush (Y)   | mov (Y), %0    | mov (Y), %0    | mov (Y), %0   |               |
| mfence        |               | mfence         |                |               |               |
|      (I)      |     (II)      |     (III)      |      (IV)      |      (V)      |     (VI)      |
```

Fig. 11. Different hammering loop bodies.

We first evaluate whether our RADAR system can effectively detect potential rowhammer attacks under simple situations, in which no memory-intensive tasks are running. The evaluation is performed in a normal working environment,

[4] A crossover USB cable having an embedded bridge controller is needed to connect two USB hosts. We use such a cable with a PL-2301 bridge controller.

[5] Since quadrature sampling is used, it provides 25 MHz bandwidth.

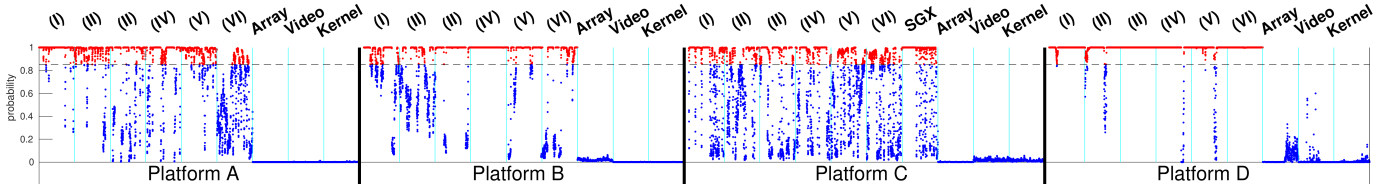


Fig. 12. The detection results in the form of the probability of hammering.

where computers with the same DRAM clock frequency are present but no closer than 1.8 m (note that later we will show the distance can be as close as 0 m), and the antenna stands outside on the metal case using a magnetic mount.

The effectiveness of our RADAR system is evaluated against the hammering methods illustrated in Fig. 11. The first five ones (I)–(V) use two addresses to perform single-/double-sided hammering via clflush, movnti, or eviction, and the last one (VI) tests one-location hammering following the tool *flipfloyd* [25]. We also evaluate the effects of a memory barrier mfence using (I) and (III). We note that it does not matter if single-sided or double-sided hammering is used with respect to the generation of hammering-correlated sideband patterns, and thus we use double-sided hammering on platforms A (Haswell) and B (Sandy Bridge) as their DRAM address mappings are available [47], [59], [64], and use single-sided hammering on platforms C and D.

We run each hammering executable for about 3 seconds in the order given by Fig. 11, and then we run three legitimate applications for about 3 seconds. The three applications are: (1) randomly accessing a large array of size 256 MiB, which will miss the caches and access the memory very often; (2) playing a video, which will continuously use non-temporal instructions to access the video; (3) using gcc to compile a Linux kernel, which will generate a large amount of processor-memory-storage traffic. Fig. 12 shows the detection results.

From the results, we can observe that malicious hammering attempts can be effectively detected for each platform under each scenario (that are represented by the red dots in the figure), and there are no false positives if the classification probability threshold is chosen sufficiently high (e.g., we simply use 0.85). We also notice some interesting phenomena when conducting these experiments. First, we find that not every hammering attempt can induce the sideband patterns of interest, although most of the attempts will. This is why the detector sometimes gives a probability output less than the threshold even during hammering. Second, compared to flushing or bypassing the cache, the patterns induced by eviction are less obvious, as indicated by the first row of Tab. II. Yet, they are still recognizable. Third, the use of memory barriers seems irrelevant to the appearance of such sideband patterns, although it ensures that all memory accesses reach the DRAM.

In addition, as studied in [25], rowhammer attacks may be hidden inside malicious SGX enclaves. Our conjecture is that, regardless of whether or not hammering is performed inside an SGX enclave, there should be no difference with respect to its characteristics in the DRAM clock spectrum. We have verified this speculation by evaluating our RADAR system against malicious SGX enclaves on platform C, as illustrated in Fig. 12. Thus, the proposed RADAR can effectively detect elusive rowhammer attacks.

The effectiveness of our RADAR system has also been evaluated against three well-known tools that are publicly available for demonstrating rowhammer attacks: (1) Google's *rowhammer-test*[6] [54], which uses a probabilistic approach to perform single-sided hammering, or takes advantage of the /proc/self/pagemap interface to acquire physical addresses for double-sided hammering; (2) Tatar's *hammertime*[7] [59], which can achieve more effective double-sided hammering by considering the detailed information about end-to-end address translation; and, (3) Gruss's *flipfloyd*[8] [25], which has a tool for testing one-location hammering.

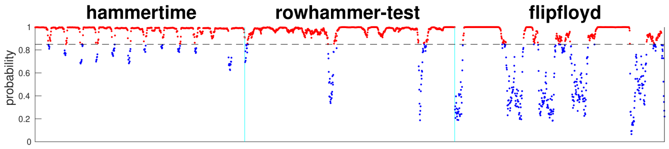


Fig. 13. The detection results on platform A *w.r.t.* three well-known tools.

We run each tool as is, and Fig. 13 shows the detection results when these tools are executed on platform A. We just use platform A as the example, because (1) platform A is very vulnerable to hammering; (2) *hammertime* only has the detailed address translation model for the platforms A and B; and (3) the detection results for other platforms are very similar to that for A. From the results, we can observe that our RADAR can effectively detect hammering attempts.

When running *hammertime* on platform A, on average there are 6.6 bit flips per second reported. (Both *rowhammer-test* and *flipfloyd* do not report any bit flip.) We implement a kernel module that "kills" all of the processes belonging to untrusted users upon receiving a message from the detector, and execute *hammertime* for 100 times. In each of the 100 trials, the hammering behavior was always detected as soon as it merely started. **We have not observed any bit flip before *hammertime* is detected and terminated, namely if only considering prevention of bit flips, the false negative**

[6] https://github.com/google/rowhammer-test

[7] https://github.com/vusec/hammertime

[8] https://github.com/IAIK/flipfloyd

**rate in this case is 0%**. The DRAMs on other platforms are originally less vulnerable, and when our RADAR is on, we have not observed any bit flip before any hammering tool is detected and terminated.

On the other hand, the false positive rate of our RADAR detection is also extremely low. As studied in [4], `gcc` induces many false positives under ANVIL; yet, from Fig. 12, we can observe that `gcc` introduces **no false positives** under RADAR. We also evaluate other SPEC 2006 benchmarks and Apache HTTP server on platform `A`. For SPEC benchmarks, we use their reference inputs, and for Apache server, we use the tool `ab` to generate heavy workloads. The representative results are shown in Fig. 14, and the results for other SPEC benchmarks are similar to `bzip2` (integer) and `lbm` (floating-point). From Fig. 14, we can clearly see that **no false positives** arise. Note that no floating-point SPEC benchmarks are used in [4], but we argue that these benchmarks should be included for evaluation due to their pervasive manipulation on very large matrices.

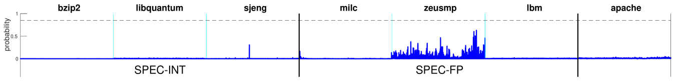


Fig. 14. The detection results on platform `A` *w.r.t.* SPEC integer and floating-point benchmarks as well as Apache HTTP server.

Since these benchmarks represent normal applications well, the results signify the aforementioned argument that a benign program barely has a behavior generating a high rate as well as periodic DRAM accesses for a long period of time (e.g., more than 30 ms) to trigger the alarm. (In fact, as indicated by the results of continuous accesses to a large array in a random way in Fig. 12, we argue that it is the same bank(s) that should be periodically accessed rather than just DRAM, which means it is even more difficult to find the alarm-triggering behavior in a benign program.) Note that although the result for `zeusmp` appears singular in contrast with others, we do not observe any false positive even when running its four instances in parallel. It shows that the possibility of synthesizing symmetric vertical stripes in the spectrogram by multiple simultaneously running benign programs is very low.

### C. Robustness of RADAR

There are two types of noise that may affect the operation of RADAR. The first type of noise is generated **internally**, due to a legitimate use of the computer system which changes the power of the DRAM clock signal constantly. To create such noise, we run different applications to impose loads on the memory system. To quantify the obviousness of the sideband patterns, we measure how relatively "tall" the patterns of interest are, namely the power difference between the patterns and their neighboring frequency components. The measurements are in dB and shown in Tab. II for sideband patterns caused by `clflush`, `movnti`, and eviction. Note that Tab. II only shows the first one that is recognizable for each case.

TABLE II
The relative power (measured in dB) of the hammering-correlated sideband patterns caused by `clflush`, `movnti`, and eviction respectively.

| Platform / Scenario | A | B | C | D |
|---|---|---|---|---|
| Baseline | 21.97/23.57/9.63 | 31.06/32.99/27.42 | 25.22/20.17/13.55 | 30.83/30.01/19.78 |
| `stress -m 10` | 13.49/16.57/8.37 | 30.89/26.94/17.97 | –/–/– | –/–/– |
| Playing a video | 21.39/22.85/9.97 | 33.61/29.83/27.20 | 21.10/18.65/14.11 | 29.02/25.24/12.89 |
| Compiling kernel | 21.30/22.78/8.92 | 32.86/27.32/26.99 | 23.20/16.60/13.35 | 28.27/29.21/15.01 |

The first row of Tab. II lists the baseline values for each platform having the minimum workload. As we can observe from the other rows, except for platforms `C` and `D` under `stress`, the patterns used for rowhammer attack detection are still discernible in the spectrum when much noise is created. Note that there are $n$ high memory traffic threads spawned by `stress -m` $n$. We find that, when running `stress -m 10` on platform `C` or `D`, all of the 10 threads can run in parallel with the hammering process leading to memory bandwidth exhaustion, so that up to five-fold time is spent in a hammering iteration. By contrast, platform `A` has a dual-core processor without SMT, which supports only 1 `stress` thread simultaneously running with the hammering process; hence, there is actually no difference between `stress -m 1` and `stress -m 10` on `A`, and the memory bandwidth is sufficient for its traffic. Surprisingly, on platform `B`, 7 `stress` threads can run in parallel with hammering, but our test shows that they together impose only 1.7 GB/s traffic, which is much less than the supported 20 GB/s bandwidth. (On other platforms, one single `stress` thread can generate 4∼5 GB/s traffic.) Note that, even with enough memory bandwidth, we have observed that the rowhammer bug is much harder to trigger while `stress` is running, let alone when bandwidth is exhausted. For example, platform `A` is very vulnerable to hammering, and on average 6.6 bit flips per second can be observed without running `stress`, but only 0.85 bit flips per second when running one `stress` thread. Therefore, the disappearance of the patterns of interest under extreme conditions is only a minor issue.

The second type of noise exists **externally**. In reality, there may be some neighboring computer systems having the same DRAM clock frequency as the one under RADAR's watch. To test whether our RADAR will become "confused", we settle platform `A` as the system under protection and observe how other platforms using DDR3-1333 affect the operation of RADAR.

TABLE III
The impacts of external noise on RADAR for platform A.

| Distance / Scenario | 1.5 m | 1.0 m | 0.5 m | 0 m |
|---|---|---|---|---|
| `B` & antenna out | none | none | none | none |
| `A`* & antenna out | none | slight | moderate† | moderate/severe† |
| `A`* & antenna in | none | none | none | none |

†These unfavorable impacts can be mitigated.

First, we gradually move platform `B` towards `A` starting from 1.5 meters away. The antenna stands outside on the metal case of platform `A`. The operation of RADAR is not affected at all, as shown in the first row of Tab. III. This is because, as mentioned in Section VI-B, de-spreading will inadvertently help reduce such noise. Recall that, for de-spreading, the hardware-dependent $\Delta$ is aligned and used to modulate the measured signal, and if the measured signal has components unrelated to the used $\Delta$, the energy of these components will be spread. Since the $\Delta$ of A is different from that of platform B, when using the $\Delta$ of A for de-spreading, the EM-emanated DRAM clock signals of B are unrelated to the used $\Delta$, and their energy is scattered to become negligible noise.

A more interesting scenario arises from using identical motherboards, as they have the same $\Delta$. Thus, we move another platform A*, which is the same as A, gradually towards A starting from 1.5 meters away. The antenna still stands outside on the metal case of A. When the distance is reduced to about 1 m, we start observing "bumps" regularly and symmetrically sweeping back and forth within ±3 MHz around the DRAM clock frequency in the spectrum. This phenomenon is due to the fact that the correct $\Delta$ alignment with the SSC-affected signal of A is most likely incorrect with respect to that of A* (unless they coincidently have the same SSC phase). As long as the antenna picks up the signal from A more than the signal from A*, the $\Delta$ remains aligned with the SSC-affected signal of A. As A* gets closer to A, the magnitudes of the sweeping "bumps" get increased, reaching the same level as the hammering-correlated sideband patterns. However, their impacts are moderate, because they have very distinguishing features such as the spikes forming the bumps are actually separated from each other by exactly 32 KHz (a behavior of SSC) so that we can take them into account in the classifier. The severe impacts come from the situation where the antenna is too close to A* such that the correct $\Delta$ alignment is disrupted. We find that the severe impacts can be avoided by carefully placing the antenna, e.g., on the other side of the case of A when A* and A are side-by-side.

The same experiment using A* is performed again but with the antenna placed inside the metal case of A. We use a very simple self-built antenna, which consists of two pieces of 7.5 cm metal wire. This antenna can easily placed inside the case of any computer, as shown in Appendix B. This time, no matter how close A* gets to A, there are no impacts on the operation of RADAR at all, as shown in the third row of Tab. III. The reason is straightforward: On one hand, the case of A acts as a EM shield, and on the other hand, the signal of A is much stronger inside the case. Note that there may be apparent reflection effects if the antenna is placed inside the case, but we notice that many spots can be found where reflections are not obvious and thus can be ignored.

### D. Resilience to Adaptive Attacks

To demonstrate the effectiveness of our RADAR on certain adaptive attacks, we create such a scenario where the adversary tries to circumvent detection by deliberately introducing some random delays into each hammering iteration of a hammering attempt, as illustrated in Fig. 15. The outer loop in Fig. 15 denotes a hammering attempt that hammers the DRAM for $N$ iterations. Inside each iteration, we use an inner loop to introduce some random delay, as its bound $b$ is randomly chosen in the range of 1 to $M$.

```
for $i := 0$ to $N-1$ do
    $b :=$ rand($M$)
    for $j := 0$ to $b-1$ do
      nop
    mov (X), %0
    mov (Y), %0
    clflush (X)
    clflush (Y)
    mfence
```

Fig. 15. Add random delay to each iteration to disturb the hammering period.

Fig. 16 presents the detection results on platform D under different $M$ values. As we can observe from the figure, even when $M$ reaches 500, it still cannot circumvent the detection. (Although theoretically we cannot prove that bit flips can be prevented when $M$ is 500, we do empirically notice that it becomes much harder/impossible to trigger the rowhammer bug on the evaluated platform when $M$ reaches 300, and no bit flips are induced when $M$ is 500.) We also show the DRAM clock spectra under different $M$ values in Appendix A, from which we can find the hammering-correlated sideband patterns indeed well recognizable.

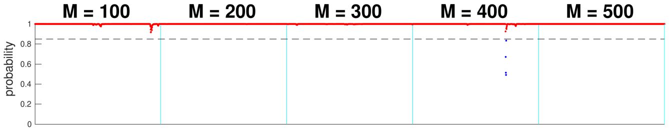


Fig. 16. The detection results on platform D *w.r.t.* different $M$ values.

Compared to the (I) results on platform D in Fig. 12, some of the results in Fig. 16 are even slightly better. As mentioned before, not every hammering attempt can induce the sideband patterns of interest, although most of the attempts will. Since the aggressor rows are randomly selected, different pairs were used in these two experiments, which caused a slight detection difference.

## VIII. Related Work

In this section, we mainly concentrate on existing rowhammer defenses that do not require unrealistic hardware modifications. In addition, we describe some related work on using physical side-channel information to bolster security defenses.

Since the activation of an aggressor row needs to be toggled enough times within a refresh interval to successfully trigger the rowhammer bug, a straightforward countermeasure is to double the refresh rate [40]. However, as shown in several tests [4], [39], this approach still cannot prevent the bug from being triggered, especially if the double-sided hammering technique is used [54]. Another straightforward defense is to use ECC memory to correct or detect bit flips [34], but it has been demonstrated that reliable rowhammer attacks in presence of ECC memory are still highly possible [16], [38].

Due to the explicit use of some special instructions like `clflush` in early rowhammer attacks, some mitigation techniques simply prohibit the use of such instructions [49], [54], but they cannot hinder eviction-based hammering [4], [26]. Given regularities found in various approaches to circumventing the effects of CPU caches, static code analysis has been used to identify suspicious binaries and estimate their intention levels to perform rowhammer attacks [32]. However, encryption and secure enclaves can be used to hide any malicious intention from static analysis [25], [53].

Based on certain characteristics observed in many rowhammer attacks, several dynamic detection approaches are proposed. Since a large number of last level cache misses are usually incurred in the hammering process, some detection techniques rely on hardware performance counters to capture suspicious activities for further analysis [4], [30]. Nevertheless, it is noticed that such cache misses will be concealed from CPU performance counters, e.g., when an attack is running inside an Intel SGX enclave [25], [53], which subverts the assumption made for the detection. Due to the traditionally used open-page policy in memory controllers, to trigger the rowhammer bug, two aggressor rows in the same bank need to be alternately activated. Consequently, some detection methods use such memory access patterns as an indication of rowhammer attacks [4], [17]. However, on some platforms, the memory controllers may be configured to use a closed-page policy to proactively close a row. In such scenarios, even one aggressor row is sufficient to induce bit flips around the row (named as one-location hammering) [25], [42], which makes access pattern based detection limited.

Usually, to successfully perform a rowhammer attack, an adversary not only needs the ability to trigger the rowhammer bug on the targeted system, but also needs to be capable of steering targeted security-critical data to some vulnerable rows for exploitation. Therefore, instead of detecting or impeding triggering the rowhammer bug, some mitigation techniques focus on hardening the system against rowhammer bug exploitation. Since the two early approaches to exploiting the rowhammer bug, memory spraying [54] and memory grooming [61], need to allocate a large portion of memory, prevention of memory exhaustion has been considered as a feasible countermeasure [26], [61]. Moreover, in [9], CATT is proposed to physically partition the main memory into different security domains, and each domain is segregated with one another by at least one unused DRAM row (i.e., a guard row), in which case, cross-domain bit flips become impossible. Unfortunately, two new approaches to exploiting the rowhammer bug, memory waylaying [25] and memory ambush [13], have been developed lately, which defeat the above-mentioned mitigation techniques.

Although CATT is no longer effective, the concept of guard rows is still valid and effective for absorbing exploitable bit flips. By using guard rows for fine-grained memory isolation, GuardION and ALIS can make the DMA-related hammerable area non-exploitable [60], [62]. To enable defenses against more general rowhammer attacks, ZebRAM is proposed in [36] to isolate all data rows with guard rows in a zebra pattern. To avoid wasting half of the DRAM, the guard rows in ZebRAM are used as an efficient swap space in memory. However, much performance overhead may still be caused for memory-intensive applications. On the contrary, our proposed technique does not incur any performance overhead due to its completely non-intrusive and passive nature.

There has been much research work on exploiting physical side-channel information for attacks [2], [3], [5], [18], [20]–[24], [31], [35], [37], [50], [52], [58]. Lately, many researchers have also started examining how to leverage such side-channel information to help defenses. For instance, power- or EM-based code execution tracking has been proposed to check whether the control flow integrity is violated [28], [43], [45]. Moreover, power or EM side-channel information has been used in discovering malware and anomalies on embedded devices [15], [55], [63], identifying the attacker ECU on in-vehicle networks [14], detecting intellectual property theft [6], [57], and so forth. Yet, there has been little prior work that uses physical side-channel information to perform rowhammer defenses, and to the best of our knowledge, only one very recent proposal leverages features in power traces to detect rowhammer attacks on embedded systems [63]. Our work is the first one on leveraging EM side-channel information to detect rowhammer attacks.

## IX. Conclusion

In this paper, we have investigated how to leverage EM side-channel information to detect rowhammer attacks. We have found that there are distinguishable sideband patterns correlated with hammering activities in the spectrum of the DRAM clock signal. Based on this observation, we have proposed and implemented a system named RADAR, which unveils and recognizes hammering-correlated sideband patterns to help set up defenses against even elusive next-generation rowhammer attacks (e.g., the ones concealing themselves inside some SGX enclaves). The effectiveness and robustness of RADAR have been demonstrated under different scenarios. Besides, RADAR does not degrade the performance or resource utilization of the computer system under protection.

In the future, we plan to implement the entire detector part of RADAR on an FPGA (e.g., the one on the used LimeSDR), and perform large-scale experiments in, e.g., a data center. In addition, we will conduct a thorough study on the possibility of the existence of a generic model for classification as well as investigate other properties of RADAR such as its power consumption[9].

## Acknowledgment

This work is supported in part by the National Science Foundation (CNS-1739328). The authors would like to thank the anonymous reviewers and the shepherd for their comments and suggestions that help us improve the quality of the paper.

[9] Our conjecture is that RADAR can actually help save energy compared to the traditional software-based rowhammer defenses. RF transceivers/receivers and FPGAs used by RADAR normally consume much less power than CPU. For example, the LMS7002M transceiver used by LimeSDR consumes only 550mW in its SISO mode (the mode used in this paper) and the Altera Cyclone EP4CE40F23 FPGA on the board is also low-power. By contrast, a high-end CPU consumes more than 100W when it is active, so reducing the CPU workload imposed by the traditional software-based defenses may help reduce the overall power consumption.

# REFERENCES

[1] M. T. Aga, Z. B. Aweke, and T. Austin, "When Good Protections Go Bad: Exploiting Anti-DoS Measures to Accelerate Rowhammer Attacks," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust*, HOST '17, 2017, pp. 8–13.

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM Side-Channel(s)," in *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, 2002, pp. 29–45.

[3] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. Zajic, and M. Prvulovic, "One&Done: A Single-Decryption EM-Based Attack on OpenSSL's Constant-Time Blinded RSA," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 585–602.

[4] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, 2016, pp. 743–755.

[5] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic Side-channel Attacks on Printers," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security '10, 2010.

[6] G. T. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-Channel Based Watermarks for Integrated Circuits," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust*, HOST '10, 2010, pp. 30–35.

[7] S. Bhattacharya and D. Mukhopadhyay, "Curious case of rowhammer: flipping secret exponent bits using timing analysis," in *International Conference on Cryptographic Hardware and Embedded Systems*, CHES '16, 2016, pp. 602–624.

[8] E. Bosman, K. Razavi, H. Bos, and C. Giuffrida, "Dedup est machina: Memory deduplication as an advanced exploitation vector," in *2016 IEEE symposium on security and privacy*, S&P '16, 2016, pp. 987–1004.

[9] F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, "CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 117–130.

[10] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead Profiling via EM Emanations," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, 2016, pp. 401–412.

[11] R. Callan, A. Zajić, and M. Prvulovic, "A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, 2014, pp. 242–254.

[12] ——, "FASE: Finding Amplitude-modulated Side-channel Emanations," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, 2015, pp. 592–603.

[13] Y. Cheng, Z. Zhang, S. Nepal, and Z. Wang, "Still Hammerable and Exploitable: on the Effectiveness of Software-only Physical Kernel Isolation," *CoRR*, vol. abs/1802.07060, 2018. [Online]. Available: http://arxiv.org/abs/1802.07060

[14] K.-T. Cho and K. G. Shin, "Viden: Attacker Identification on In-Vehicle Networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, 2017, pp. 1109–1123.

[15] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, K. Fu, and W. Xu, "WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices," in *Proceedings of the 2013 USENIX Conference on Safety, Security, Privacy and Interoperability of Health Information Technologies*, HealthTech '13, 2013.

[16] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks," in *2019 IEEE Symposium on Security and Privacy*, S&P '19, 2019.

[17] J. Corbet, "Defending against Rowhammer in the kernel," 2016. [Online]. Available: https://lwn.net/Articles/704920/

[18] M. Enev, S. Gupta, T. Kohno, and S. N. Patel, "Televisions, Video Privacy, and Powerline Electromagnetic Interference," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, 2011, pp. 537–550.

[19] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi, "Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU," in *2018 IEEE Symposium on Security and Privacy*, S&P '18, 2018, pp. 357–372.

[20] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results," in *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, 2001, pp. 251–261.

[21] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation," in *International Conference on Cryptographic Hardware and Embedded Systems*, CHES 2015, 2015, pp. 207–228.

[22] ——, "ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs," in *Topics in Cryptology - CT-RSA 2016*, 2016, pp. 219–235.

[23] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 2016, pp. 1626–1638.

[24] D. Genkin, A. Shamir, and E. Tromer, "Acoustic Cryptanalysis," *Journal of Cryptology*, vol. 30, no. 2, pp. 392–443, Apr. 2017.

[25] 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 *2018 IEEE Symposium on Security and Privacy*, S&P '18, 2018, pp. 489–505.

[26] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '16, 2016, pp. 300–321.

[27] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+Flush: A Fast and Stealthy Cache Attack," in *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721*, DIMVA '16, 2016, pp. 279–299.

[28] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch Me, but Don'T Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, 2017, pp. 1095–1108.

[29] K. B. Hardin, J. T. Fessler, and D. R. Bush, "Spread Spectrum Clock Generation for the Reduction of Radiated Emissions," in *Proceedings of IEEE Symposium on Electromagnetic Compatibility*, EMC '94, 1994, pp. 227–231.

[30] N. Herath and A. Fogh, "These are Not Your Grand Daddy's CPU Performance Counters - CPU Hardware Performance Counters for Security," in *Black Hat Briefings*, 2015.

[31] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, and G. Sigl, "Localized Electromagnetic Analysis of Cryptographic Implementations," in *Proceedings of the 12th Conference on Topics in Cryptology*, CT-RSA '12, 2012, pp. 231–244.

[32] G. Irazoqui, T. Eisenbarth, and B. Sunar, "MASCAT: Preventing Microarchitectural Attacks Before Distribution," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, 2018, pp. 377–388.

[33] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-Bomb: Locking Down the Processor via Rowhammer Attack," in *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, SysTEX '17, 2017, pp. 5:1–5:6.

[34] 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 *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, 2014, pp. 361–372.

[35] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, 1999, pp. 388–397.

[36] R. K. Konoth, M. Oliverio, A. Tatar, D. Andriesse, H. Bos, C. Giuffrida, and K. Razavi, "ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks," in *13th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '18, 2018, pp. 697–710.

[37] M. G. Kuhn, "Electromagnetic Eavesdropping Risks of Flat-Panel Displays," in *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*, PET '04, 2004, pp. 88–107.

[38] A. Kwong, D. Genkin, D. Gruss, and Y. Yarom, "RAMBleed: Reading Bits in Memory Without Accessing Them," in *41st IEEE Symposium on Security and Privacy*, S&P '20, 2020.

[39] M. Lanteigne, "How Rowhammer Could Be Used to Exploit Weaknesses in Computer Hardware," 2016. [Online]. Available: http://www.thirdio.com/rowhammer.pdf

[40] Lenovo Inc., "Row Hammer Privilege Escalation Lenovo Security Advisory (LEN-2015-009)," 2015. [Online]. Available: https://support.lenovo.com/us/en/product_security/row_hammer

[41] Lime Microsystems, "LimeSDR." [Online]. Available: https://limemicro.com/products/boards/limesdr

[42] M. Lipp, M. T. Aga, M. Schwarz, D. Gruss, C. Maurice, L. Raab, and L. Lamster, "Nethammer: Inducing Rowhammer Faults through Network Requests," *CoRR*, vol. abs/1805.04956, 2018. [Online]. Available: http://arxiv.org/abs/1805.04956

[43] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On Code Execution Tracking via Power Side-Channel," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 2016, pp. 1019–1031.

[44] O. Mutlu, "The RowHammer Problem and Other Issues We May Face As Memory Becomes Denser," in *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, 2017, pp. 1116–1121.

[45] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-Based Detection of Deviations in Program Execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, 2017, pp. 333–346.

[46] M. Payer, "HexPADS: A Platform to Detect "Stealth" Attacks," in *Proceedings of the 8th International Symposium on Engineering Secure Software and Systems*, ESSoS 2016, 2016, pp. 138–154.

[47] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 565–581.

[48] M. Prvulovic, A. Zajić, R. L. Callan, and C. J. Wang, "A Method for Finding Frequency-Modulated and Amplitude-Modulated Electromagnetic Emanations in Computer Systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 1, pp. 34–42, 2017.

[49] R. Qiao and M. Seaborn, "A new approach for rowhammer attacks," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust*, HOST '16, May 2016, pp. 161–166.

[50] J.-J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," in *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, E-SMART '01, 2001, pp. 200–210.

[51] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a Needle in the Software Stack," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1–18.

[52] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple Photonic Emission Analysis of AES: Photonic Side Channel Analysis for the Rest of Us," in *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems*, CHES '12, 2012, pp. 41–57.

[53] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware Guard Extension: Using SGX to Conceal Cache Attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '17. Springer, 2017, pp. 3–24.

[54] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," in *Black Hat Briefings*, 2015.

[55] N. Sehatbakhsh, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral Analysis for Anomaly Detection on Medical IoT and Embedded Devices," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust*, HOST '18, 2018, pp. 1–8.

[56] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic, "Spectral Profiling: Observer-effect-free Profiling by Monitoring EM Emanations," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49, 2016, pp. 59:1–59:11.

[57] D. Strobel, F. Bache, D. Oswald, F. Schellenberg, and C. Paar, "Scandalee: A Side-channel-based Disassembler Using Local Electromagnetic Emanations," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, 2015, pp. 139–144.

[58] T. Sugawara, D. Suzuki, M. Saeki, M. Shiozaki, and T. Fujino, "On Measurable Side-channel Leaks Inside ASIC Design Primitives," in *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems*, CHES '13, 2013, pp. 159–178.

[59] A. Tatar, C. Giuffrida, H. Bos, and K. Razavi, "Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer," in *Research in Attacks, Intrusions, and Defenses*, RAID '18, 2018, pp. 47–66.

[60] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer Attacks over the Network and Defenses," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 213–226.

[61] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 2016, pp. 1675–1689.

[62] V. van der Veen, M. Lindorfer, Y. Fratantonio, H. P. Pillai, G. Vigna, C. Kruegel, H. Bos, and K. Razavi, "GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '18, 2018, pp. 92–113.

[63] S. Wei, A. Aysu, M. Orshansky, A. Gerstlauer, and M. Tiwari, "Using Power-Anomalies to Counter Evasive Micro-architectural Attacks in Embedded Systems," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust*, HOST '19, 2019.

[64] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 19–35.

[65] A. Zajic and M. Prvulovic, "Experimental Demonstration of Electromagnetic Information Leakage From Modern Processor-Memory Systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 885–893, 2014.

[66] S. Zeitouni, D. Gens, and A.-R. Sadeghi, "It's Hammer Time: How to Attack (Rowhammer-based) DRAM-PUFs," in *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, 2018, pp. 65:1–65:6.

## Appendix A

Section VII-D has introduced an adaptive attack which tries to circumvent detection by adding some random delays into each hammering iteration. The random delay varying range is controlled by a parameter $M$. Fig. 17 shows the DRAM clock spectra of platform D under different $M$ values. (Note that the experiments are performed under normal circumstances where the SSC feature is always on.) As anticipated, when random delays are introduced, the periodic behavior of hammering is disrupted to some extent, and thus the hammering-correlated sideband patterns become less prominent than those without adding such delays. However, as illustrated in the figure, even when $M$ reaches 500, the patterns are still recognizable for its use in detection.

From Fig. 17 (A) that corresponds to the normal situation without adding random delays, we can observe three pairs of "bumps" very clearly on both sides of the central spike, which are circled and pointed to by arrows. They are located at about 1066 MHz $\pm k \times$ 3.9 MHz in the spectrum, where $k = 1, 2, 3$. The reason for this phenomenon is that the modulating signal generated by the switching behavior of hammering on platform D has strong second and third harmonics. Therefore, when this signal AM-modulates the DRAM clock carrier signal, the sideband patterns corresponding to the second and third harmonics will arise noticeably. Nevertheless, this does not cause any problem or difference for our detection method, since there are still two vertical stripes symmetric about the DRAM clock frequency in the spectrogram.

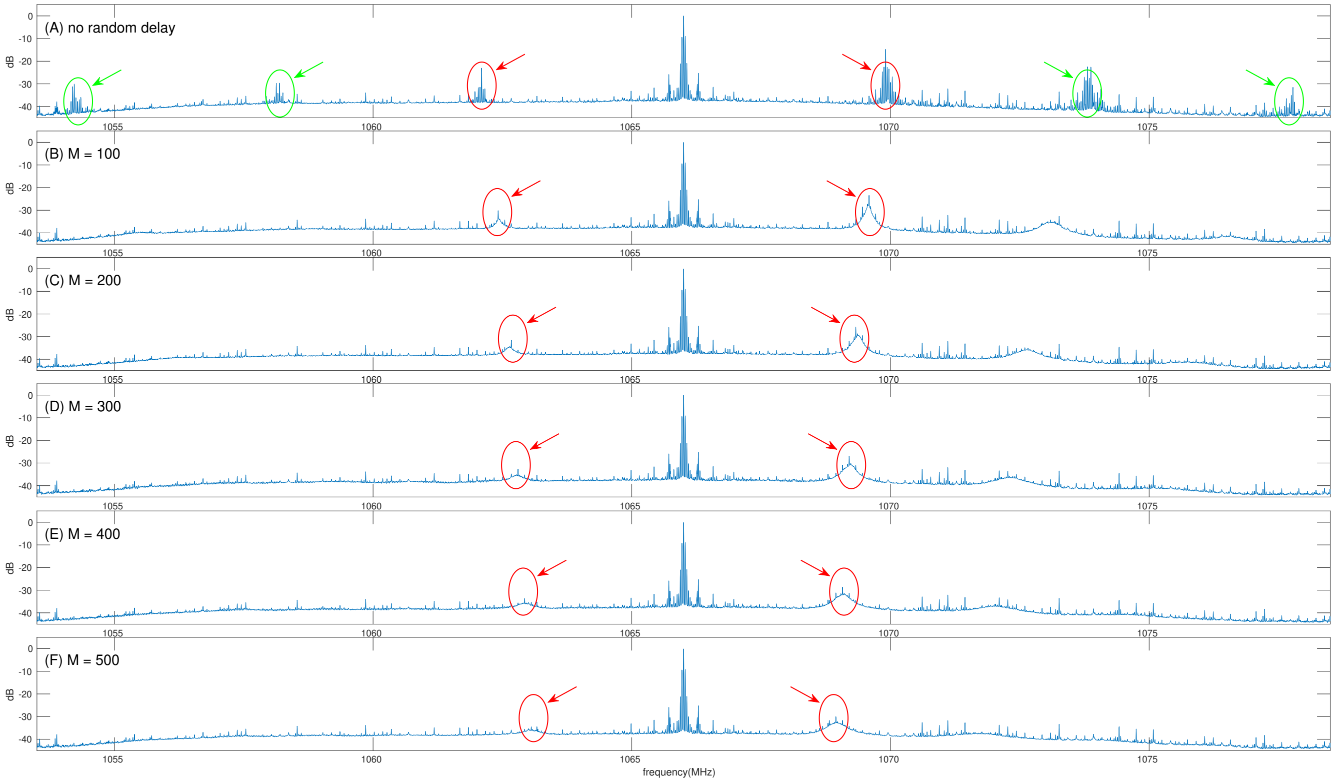


Fig. 17. The power spectra under different $M$. In a hammering attempt, *each* hammering iteration will be delayed by a loop whose bound is randomly chosen in the range of 1 to $M$. The larger $M$ is, the more disturbance is added into the hammering period.

## Appendix B

Each computer system under the watch of the proposed RADAR will be equipped with an antenna and a detector. The antenna used in our RADAR can be a very simple whip antenna, such as a telescopic antenna or just a piece of wire. Fig. 18 shows two antennas used in RADAR. The left one is a telescopic antenna, which has a magnetic mount to make itself easy to stand on the metal case of a computer. The right one is a self-built antenna, which consists of two pieces of metal wire connected to an antenna balun. The wire is coated with plastic for isolation.

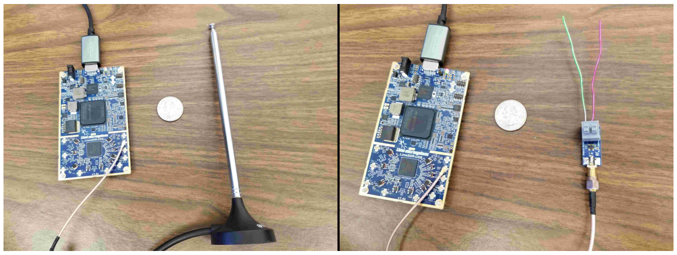


Fig. 18. Two antennas that have been used in RADAR. In both figures, the used LimeSDR is also shown.

As evaluated in Section VII-C, when two identical platforms are very close (e.g., right next to each other), we need to place the antenna inside the metal case. We can generally manage to place the telescopic antenna inside the mini tower (or bigger) cases. By contrast, our self-built antenna can be easily placed inside the case of **any size** (e.g., small form factor or server chassis).

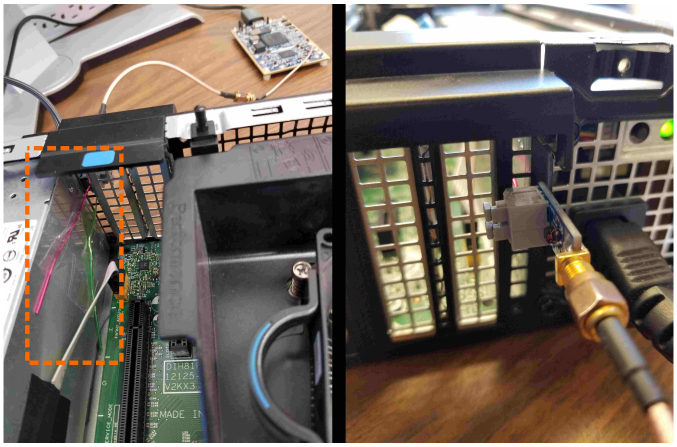


Fig. 19. Placing the self-built antenna inside the metal case of a SFF computer.

For example, Fig. 19 illustrates how to place our self-built antenna inside a small form factor (SFF) computer of size 31.2 × 29.0 × 9.3 cm. The antenna is inserted into the computer case through the holes on the backplate and taped on the power supply, which can be seen from the left part of Fig. 19 (denoted by the dashed line). We simply leave the antenna balun outside the case, as shown in the right part of Fig. 19. This placement just took us several minutes. Even though it was possible to spend longer time on placement in some situations, we argue that it might just need to happen once and can remain fixed if no significant changes need to be made on the hardware side of the platform later on.

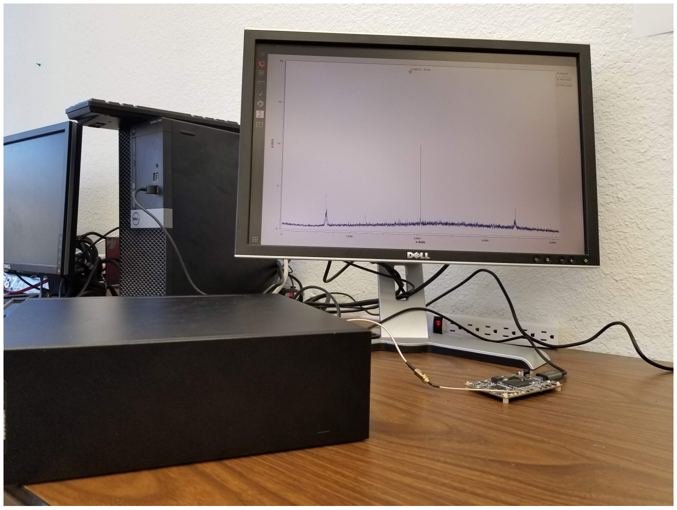


Fig. 20. Apparent hammering-correlated sideband patterns.

Given the aforementioned antenna placement, we execute a program for hammering. As we can observe from Fig. 20, the hammering-correlated sideband patterns are extremely clear. Again, the SSC is always on and the spectrum is shown after de-spreading.

## Appendix C

To preliminarily demonstrate a specifically tailored model is not necessary for classification, we evaluate our current CNN model on two additional platforms, whose data has never been used in the original training. These two platforms E and F are listed in Tab. IV. We can find that platforms A, B, and E are all equipped with DDR3-1333 modules, but their DRAM chip vendors are different (c.f., Tab. I). Likewise, both platforms D and F have DDR4-2133 modules, but their memory chips are also from different vendors.

Furthermore, we change the memory modules of E and F to form another two platforms E′ and F′. As listed in Tab. IV, E′ uses DDR3-1600 and F′ uses DDR4-2400. Note that both of these two DRAM speed types have never been involved in the original model training. We conduct the experiments listed in Fig. 11 on these four platforms.

TABLE IV
Additional platforms on which our prototype is further evaluated.

| Platform | Motherboard | CPU | Memory |
|---|---|---|---|
| E | Dell OptiPlex 3020 | Intel i5-4590 | 16 GiB Kingston DDR3-1333 |
| E′ | Dell OptiPlex 3020 | Intel i5-4590 | 8 GiB Micron DDR3-1600 |
| F | Dell XPS 8920 | Intel i7-7700K | 16 GiB Hynix DDR4-2133 |
| F′ | Dell XPS 8920 | Intel i7-7700K | 16 GiB Hynix DDR4-2400 |

The detection results are presented in Fig. 21. From the results we can observe that the model, trained using data from platforms A, B, C, and D, works well for recognizing potential attacks on platforms E, E′, F, and F′. Although data in terms of DDR3-1600 and DDR4-2400 modules has never been seen during the CNN model training, very good generalization has been achieved, which is able to classify new examples having symmetric vertical stripes in the spectrogram as possible rowhammer attacks.

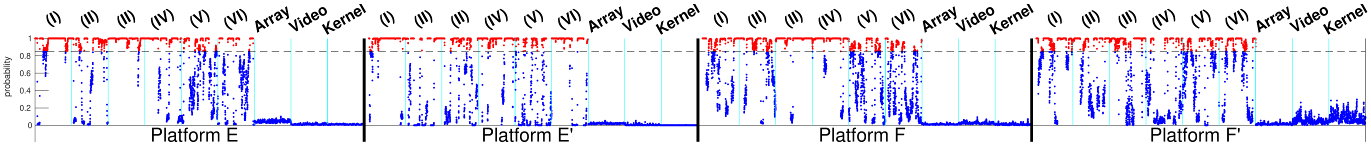


Fig. 21. The detection results on additional platforms in the form of the probability of hammering. The CNN model for classification is the one trained in Section. VII without any change.