NEW ATTACKS AND DEFENSES LEVERAGING ELECTROMAGNETIC SIDE-CHANNEL INFORMATION

By

Zihao Zhan

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in

Electrical Engineering

October 31, 2021

Nashville, Tennessee

ACKNOWLEDGMENTS

I would like to sincerely thank my advisors Dr. Zhang and Dr. Koutsoukos. It is impossible to finish this work without their continuous support, guidance and encouragement.

I would also like to express my gratitude to the rest of the committee members, Dr. Bhuva, Dr. White, and Dr. Robinson. I really appreciate their generous time, comments and suggestions.

My thanks also go to my labmates Mert Side and Sisheng Liang for offering me significant help in work and being such good friends with me.

Besides, I am extremely thankful to my friends, Dr. Roman Wang, Dixie Wang, Aubrey Tang, David Qiao, and so forth. Thank them for bringing so much fun and happiness to my life.

Last but not the least, I would also thank my family for their everlasting support and unconditional love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Hardware Security

As all kinds of computing devices get increasingly integrated into our daily lives in the age of IoT, security and privacy have become one of the top concerns for people. All computer systems, including personal computers, cloud servers, mobile phones, smart cards, and other embedded systems, consist of two parts: software and hardware. A lot of earlier research focused on the security of software, which is defined as the program implementing a specific algorithm to perform a task. However, in recent years, industry and academia start to realize the fact that software-level protection alone is far from enough to completely safeguard a computer system. Therefore, we have seen growing hardware security research that focuses on the hardware platforms where the software is implemented. The expanding hardware security research has unveiled many previously overlooked hardware vulnerabilities that may impose severe threats on computer systems. A malicious actor may exploit these vulnerabilities as a network client, a co-residing user, or anyone who can achieve a certain level of physical proximity to the targeted device. This section will briefly describe the hardware security problems.

Hardware vulnerabilities can be either intentionally or unintentionally introduced during the design and manufacturing phases of electronic devices. One of the most well-known intentionally inserted vulnerabilities is the hardware Trojan – a malicious modification on the circuits that compromises the hardware integrity. A hardware Trojan is usually triggered under certain conditions, like a particular internal logic state, a certain input trace, or some specific physical environment. Depending on the goals of the malicious actor, hardware Trojans can be inserted for different purposes. It can assist the attackers in bypassing the security mechanisms, affecting the computation outputs, undermining the system performance, or stealing proprietary information. The insertion of hardware Trojan can happen at multiple stages in a compromised IC supply chain. Sometimes it happens during the design phase with the presence of malicious insiders in semiconductor design companies. However, it is most common that the hardware Trojan is inserted during the manufacturing phase. Following the globalization process, the production of integrated circuits (ICs) is outsourced worldwide in order to reduce the cost and shorten the time-to-market. Consequently, the heavy involvement of third-party manufacturers makes it infeasible to secure the supply chain from malicious actors, and the insertion of hardware Trojan in ICs becomes much more likely to happen. The other kind of hardware vulnerability is introduced unintentionally. A lot of techniques are implemented in modern computer systems to improve performance, but many of them have been lately found to bring exploitable vulnerabilities at the same time. For example, the computer memory hierarchy separates the storage into multiple levels with varied sizes and access speeds. While this design significantly improves the processor's performance by allowing it to perform high-speed memory access from CPU caches, the cache effects, on the other hand, become one of the most commonly exploited vulnerabilities in side-channel attacks. Similar problems are also found in the translation lookaside buffer (TLB). Speculative execution is another technique implemented in modern CPUs taking full advantage of parallelism to speed up the tasks. Nevertheless, recent works Spectre and Meltdown Lipp et al. (2018b); Kocher et al. (2019) have discovered microarchitectural state changes caused by speculative executions can be exploited to leak sensitive information. Some unintentionally introduced vulnerabilities originate from the underlying implementations of electronic devices. The rowhammer bug reported in Kim et al. (2014) can cause bit-flip in a dynamic random-access memory (DRAM) cell by rapidly and repeatedly accessing memory in the physically adjacent rows. The root cause of this bug is the implementation of DRAM because bit information is represented by the charging states of capacitors, where electric leakage is accelerated by hammering activities. All kinds of physical side-channel effects are unintentional vulnerabilities coming from the intrinsic properties of electronic components or semiconductors. The computation-dependent current on a circuit unavoidably causes power side-channel effects. This constantly changing current further generates electromagnetic (EM) fields around it. Due to the heating effect of the electric current, the temperature can be changed in the surrounding environment. Additionally, The switching activity of transistors will cause photonic emissions. Sometimes the varying current flowing through components like inductors will cause mechanical vibrations to produce acoustic signals.

Various attacks can be launched by exploiting these hardware vulnerabilities. Data remanence attack is a technique attempting to recover and read data from storage devices where data is supposed to be erased. For instance, shortly after a computer shuts down, the data stored in random-access memory (RAM) should be lost. However, a cold boot attack can be implemented by freezing the RAM to lower the electric leakage rate, allowing data remanence in the RAM to be read after it is connected to power again. Fault attack is another common type of attack that compromises the system's integrity. The hardware fault can often be triggered when the device is working under unusual conditions. These conditions include extreme temperature, high-energy laser, strong electromagnetic radiations, clock glitches, and over/under power. As mentioned above, these unusual working conditions also include abnormal software behavior like rapid and repeated memory accesses to the same rows in a DRAM bank that is used to trigger the rowhammer bug. Implementation of fault attacks involves finding and locating the vulnerable location, triggering the fault in a controlled way to induce exploitable faults, and exploiting the faults. Another similar attack also tries to trigger faults on hardware, but it aims to inhibit the targeted device instead of sabotaging the integrity. This attack tries to trigger a massive amount of faults and does not require precise control of the fault injecting position. The targeted device can be put out of service or even permanently damaged. Unlike the active attacks mentioned above, the side-channel attack is a passive attack, which does not alter the implementation of the targeted hardware. As mentioned above, physical implementations of software on hardware platforms inevitably have computation-dependent physical effects, i.e., produce side-channel leakages. Timing side-channel attacks infer sensitive information by measuring the execution time of secret-dependent computation. Cache side-channel attacks eavesdrop on the secret by monitoring the cache states brought by the targeted computation. Apart from these logical side channels, physical side channels are featured by measurable physical variants brought by computations. Besides these logical side channels, a lot of research studies the leakage from the physical side effects of computation. These physical side channels include power consumption, electromagnetic emanation, acoustic emission, photonic emission, and thermal transmission. While side-channel information is often exploited to implement attacks, it can sometimes be leveraged for hardware anomaly detection. Like trusted programs, malware's computations also inevitably have side-channel effects, especially when the malware tries to alter some hardware behaviors. Compared to other defenses, side-channel based malware detection reduces the observer effect and brings low overhead to the protected system. So the side-channel leakage from hardware devices is a double-sided blade that may be used either for attacks or defenses.

An attacker attempts to breach the security to compromise confidentiality, integrity, or availability through exploiting hardware vulnerabilities. One of the most researched topics for confidentiality violation is cryptography breaking. Many cryptographic algorithms are considered invulnerable mathematically so that they are secure on the software side. However, many cryptography systems have been found vulnerable to hardware vulnerability exploitation. Numerous side-channel attacks have effectively broken all manner of cryptographic algorithms (including DES, AES, RSA, ECC, DH, MD5, SHA256, etc.) implemented on various platforms (smart cards, embedded systems, personal computers, cloud servers). Fault attacks can also be implemented to break cryptography systems. For some cryptographic algorithms, a single bit of fault injected into the parameters during computations can tremendously lower the difficulty of inferring the secret key. Besides cryptographic keys, hardware vulnerabilities can also cause the leakage of other proprietary information like protected intellectual properties, user activities, user identities, and commercial secrets. Data exfiltration is another exploitation that compromises confidentiality through implementing covert channels. Unlike common side-channel attacks that do not affect the computations on hardware, the construction of covert channels requires a sender to be inserted into the target device that induces side-channel effects on purpose to perform unauthorized data transfer. The receiver is deployed to measure this side-channel information and recover the exfiltrated data from it. Although covert channels have a stronger threat model than side-channel attacks and have more constraints for implementation, they can help reveal the upper bound of possible leakages through a specific side-channel. Some attacks aim to compromise the integrity by performing illegitimate data modifications on devices. By changing parameters in a program (e.g., a neural network model), an attacker can bias the computation and manipulate the output Yao et al. (2020). It can also be used to modify a page table entry (PTE) to give an unprivileged attacker full access to physical memory or tweak the instructions to escape a sandbox environment Seaborn and Dullien (2015). Sometimes the availability of systems is compromised by attacks. It can either be achieved by corrupting the software on hardware platforms or directly damage the hardware itself.

To better protect computer systems from the threat posed by hardware vulnerabilities, many countermeasures are proposed. However, some vulnerabilities are found in designs that are essential for the system performance or originated from the intrinsic property of electronic devices, so fixing them by simply disabling or re-implementing these functions is impractical. Most countermeasures apply patches to prevent known attacks, but it is rarely the case that we can eliminate the vulnerabilities completely. Besides, most applied countermeasures inevitably have some side effects like resource costing, performance impairment, financial inefficiency, etc. Balancing the trade-off between security and the negative effects has always been the main concern for countermeasure implementations. Accordingly, how to mitigate hardware attacks has always been a challenging problem, even when the vulnerabilities are identified.

On the opposite of exploiting hardware vulnerabilities to launch attacks, some techniques are implemented to improve the system security with the assistance of hardware designs. A typical example is the trusted execution environment (TEE), where hardware isolation is used to provide a more secure environment for sensitive applications like cryptography software. ARM TrustZone and Intel SGX are the two most well-known commercial implementations of TEEs. The physical unclonable function (PUF) is another hardware-assisted design to improve hardware security. By leveraging some physical characteristics introduced during manufacturing, a PUF can produce an output for a given input, which serves as an identifier of the circuit. Because the introduced physical properties are unpredictable and uncontrollable, it is almost impossible to clone a PUF with the same properties, making the output a unique identifier.

1.2 Electromagnetic Side-channel

The EM side-channel is one of the most researched side-channel effects because it can reveal much knowledge of the ongoing activity and has been extensively exploited to breach confidentiality. The history of public research on espionage using EM side-channel leakages can trace back to 1985 when EMR from the cathode-ray tube (CRT) display is found exploitable in reconstructing screen content Van Eck (1985). The earliest EM side-channel attacks targeting ICs were implemented to break cryptography on smart cards Gandolfi et al. (2001), shortly after power side-channel attacks were proposed Kocher et al. (1999). EM traces were collected as an equivalent of power traces in these attacks. Likewise, techniques like simple power analysis (SPA) and differential power analysis (DPA) were directly applied on EM traces as simple electromagnetic analysis (SEMA) and differential electromagnetic analysis (DEMA), respectively. Although the EM side-channel attack tends to obtain signals with a lower signal-to-noise ratio (SNR) than power traces, it has many advantages over the power side-channel attack. For example, EM signals can be collected non-invasively with a probe or antenna placing near the target hardware while measuring the power consumption usually requires connecting a power meter. Moreover, unlike power analysis that can only measure the global power consumption, EM analysis allows an attacker to measure the local power consumption with the appropriate placement of probes around the device. However, most early EM side-channel research focuses on relatively simple devices like smart cards and field-programmable gate arrays (FPGAs). Similar exploitation on more sophisticated systems was not well studied until when Longo et al. (2015) translated the EM analysis techniques for simpler deices to SoC-based devices and Genkin et al. (2015a) exploited EM leakages to perform key extraction from PCs. On these devices, exploiting EM side-channel information is much more complicated due to the complexity bought by factors like faster computation speed, higher parallelism, complicate scheduling policy, etc.

The root cause of EM side-channel leakage is that constantly changing electric current on any circuits must be present during computations, and electromagnetic fields are subsequently generated in the surrounding space. Afterward, the causal fields propagate through space in the form of waves, and these waves are named electromagnetic radiation (EMR). Based on the fact that magnetic fields and electric fields can induce current on conductors, magnetic probes and antennas can be used to receive the EMR. EM emanations from electric devices can be categorized into direct and indirect emanations. When a time-varying current is present on circuits, this current itself will generate electromagnetic fields around it, causing so-called direct EM emanations. Modern electric devices try to fit many components into small areas, and this current may function as a modulation of currents on nearby components due to the electromagnetic coupling effect. EM radiations caused by the modulated current hence carry information of the modulating current that can be recovered after appropriate demodulation, i.e., indirect emanations are caused by the current of interest. In general, exploitable indirect emanations tend to have higher frequencies and intensities compared to direct emanations. When strong sources like clocks generate the carrier signals of indirect emanations, a high SNR signal carrying sensitive information can travel over long distances. To collect, process, and analyze the EM signal, oscilloscopes and software-defined radios (SDR) are the most frequently used devices.

Based on the distance from an EMR source, the areas around the source can be categorized into near-field and far-field regions. Generally, the region within one wavelength radius from the source is the near-field region. In this region, electric and magnetic fields can exist independently, with one field dominates the other, and their intensities decrease fast with distance. Many near-field EM side-channel attacks collect the EM traces using probes closely located to the target devices, which are IC chips in most cases. To launch this kind of attack, the attacker normally needs to physically possess the device and sometimes may need to decapsulate a chip to obtain a stronger signal, making the attack semi-invasive. Generally, most near-field attacks measure direct emanations.

In contrast, far-field is usually defined for the region more than two wavelengths away from the radiation source. Electric and magnetic fields in this region tend to have equal intensities that degrade much slower with distance compared to in the near-field. Far-field EM side-channel attacks usually collect EM traces from a distance away, targeting high-power devices using antennas. Large electronic devices with high computation power, like PCs, usually consume power at more than hundreds of watts, and large current fluctuations can appear at some parts. Subsequently, some long wires where the varying current flows act as an unintentional antenna to emanate strong EM waves that can propagate over long distances. Compared to near-field attacks, far-field EM side-channel attacks require less physical proximity, which sometimes imposes more practical threats. However, devices capable of generating such strong EM emanations usually have more complicated hardware implementations than low-power devices, which significantly increases the difficulty of signal processing needed for information extraction. While a few strong direct emanations allow far-field attacks to be launched, far-field attacks often exploit strong indirect emanations with long propagation distances.

While far-field attacks can be more threatening than near-field attacks because of the longer attacking distance, few actual side-channel attacks are performed. Most previous attacks only perform data exfiltration through EM covert channels. Even for covert channels, the data exfiltration bandwidth is too low to make them serious threats. Several reasons are accounting for this. First, identifying exploitable signals in the far-field region can be challenging due to the widely existing background noise emitted from communication systems and nearby electronic devices. Besides, for devices capable of generating strong EM emanations, some techniques may be implemented to reduce the intensity of emitted signals. For example, spread-spectrum clocking (SSC) is implemented in modern computers to disperse the clock signals' energy in a narrow frequency range to a wide frequency range. With such techniques, the emanated EM signals are more easily buried by the noise and become less identifiable to the attackers. However, the complexity in exploitation does not mean that far-field EM leakage can not pose real hazards. On the contrary, many far-field EM emanations with robust data-carrying ability are embedded with much sensitive information, making them possible to cause severe information leakage. The limited amount of research in far-field EM side-channel is responsible for the lack of a clear understanding of the causal threat, making it difficult to prepare for potential attacks exploiting these leakages.

In this thesis, we focus on using EM side-channel information to detect attacks exploiting hardware vulnerabilities and disclosing new attacks exploiting EM side-channel information. Specifically, we propose an EM side-channel-based rowhammer detection techniques, find a powerful EM covert channel, and discover EM side-channel attacks than can steal user privacy.

1.3 Challenges

To summarize, the challenges for current research on hardware security and EM side-channel analysis include:

1. All different approaches to triggering the rowhammer bug make finding effective and practical mitigations to rowhammer attacks hard.

2. Most side-channel information is exploited to launch attacks, while how to leverage side-channel information for defense is not clear.

3. The spread-spectrum clocking technique is implemented on almost all modern computer systems, making some EM emanations have too low SNR to be useful.

4. Many substantial EM leakages have not been adequately addressed due to the lack of proper understandings of them.

5. While physical covert channels can pose severe threats for computer systems, they were often overlooked due to the weak performance of state-of-the-art ones.

6. EM side-channel research is mainly conducted on simple devices, leaving more complex devices (e.g., personal computers and cloud servers) exposed to unknown threats.

7. Leakages in far-field EM emanations can cause more practical threats, but they are difficult to identify, resulting in inadequate preparation for unknown attacks.

8. Widely-existing EM noise increases the difficulty of examining EM side-channel information in noisy frequency ranges. Thus specially-designed signal processing techniques are usually needed to extract the information of interest.

1.4 Contributions

The main contributions in this thesis are as follows:

- **Chapter 3**
  - 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.
- **Chapter 4**
  - We present a new physical covert channel named *BitJabber* that can allow expedited data exfiltration between air-gapped sender and receiver.
  - We designed a novel technique to produce a strong covert channel carrier that allows multiple modulation techniques to be applied.
  - We verify that our *BitJabber* covert channel is much more resilient to background noise compared with the state-of-the-art ones.
  - We demonstrate that this new covert channel can achieve reliable communication within a few meters, even under the scenario where the sender and the receiver are in separate rooms with walls in-between.
- **Chapter 5**
  - We present a new EM side-channel vulnerability that we have discovered in modern GPUs and can be exploited to carry out attacks at a distance and/or through a wall. We identify the ubiquitously used DVFS as the root cause of this side-channel and find that such a vulnerability exists in many GPUs of both NVIDIA and AMD.
  - We formulate a signal processing framework to address the challenges introduced by potential EM shielding and strong noise contamination. With the proposed techniques, we can exploit the EM emanations of interest even when they are greatly attenuated and/or overwhelmed by strong legitimate communication signals.

- We conduct two case studies on the exploitation of this newly found EM side-channel vulnerability. The first one is a website fingerprinting attack, and up to 93.2% accuracy can be achieved in a scenario where the attacker and victim are 6 meters apart. The second case study is a keystroke timing inference attack, where we show that keystroke events can be reliably detected to deduce inter-keystroke times.
- We show that even though disabling GPU DVFS can be an effective approach to mitigating the discovered EM side-channel vulnerability, it will unfortunately introduce another new one into many GPUs which can be exploited to mount comparable EM side-channel attacks. We also discuss some potential countermeasures.

1.5 Organization

The rest of this thesis is organized as follows: Chapter 2 describes the related work; Chapter 3 introduces the EM-based rowhammer attack detection technique; Chapter 4 presents a high-speed, through-wall, and long-distance EM covert channel exploiting the EM emanations from memory clocks; Chapter 5 implements techniques using DVFS-related EM emanations from GPU to reveal information about user activities; Chapter 6 presents some observations of correlations between GPU's EM emanations and deep neural network (DNN) model being trained or evaluated on it and proposes future work of EM-based DNN model reverse engineering; Chapter 7 concludes this thesis.

CHAPTER 2

Related Work

The main concentration on this thesis is about how to extract information of interest from EM emanations from computers, and how to use the information for detecting system anomalies (ongoing rowhammer attacks, to be specific), exfiltrating data rapidly and stealthily, and detecting user activities like website browsing and typing. The rowhammer bug has been recognized an excessively critical hardware vulnerabilities. Even though many countermeasures are proposed to solve this problem, all prior methods have certain limitations. Although the EM side-channel leakages from memory clocks have already been identified in prior research, no work has thought of using this information for rowhammer attack detection. Previous research on EM side-channel, especially far-field EM side-channel, mostly just reveal the possible leakages instead of implementing actual attacks, or only implementing covert channel attacks. Furthermore, physical covert channels implemented in prior work have obvious flaws with respect to data exfiltration bandwidth, obstacle penetrating ability, or transmission distance, making them barely capable of posing real threats. Besides memory clocks, other high-power electronic devices like discrete GPUs can also generate strong EM emanations. As a GPU is responsible for displaying content on screen, the EM leakages may contain information about content displayed on screen, which may be used for figerprinting websites browsed by users and timing keystroke activities causing screen content changes. In this section, we are presenting related work about rowhammer attack, side-channel-based defenses, EM side-channel analysis, physical covert channel, GPU-related security problems, and user activity monitoring attacks.

2.1 Rowhammer Attacks and Defenses

DRAM is widely used in modern electronic devices as the main memory. In DRAM, the bit ’0’ and ’1’ are represented using the charged states of capacitors. Because a capacitor leaks charge naturally, a regular refresh operation is necessary to keep the states unchanged and prevent possible data loss. However, DRAM nowadays is getting denser for performance improvement. Subsequently, the capacitor becomes smaller and the voltage margin separating ‘0’ and ‘1’ becomes lower, which unfortunately have reduced the overall DRAM reliability Mutlu (2017). Kim *et al.* first thoroughly studied the rowhammer problem in Kim et al. (2014), pointing out that repeated and rapid memory access to the same locations (i.e., hammering) may cause bits in the physically adjacent cells to be flipped to the opposite value. This is caused by the electromagnetic coupling effects that sometimes accelerate the leakage in a capacitor. The existence of the rowhammer bug seriously violates memory protection because a process can modify the data in memory without explicit permission. After the publication of Kim's work, a tremendous amount of follow-up research has been conducted, and new attacks and defenses related to the rowhammer bug have been proposed. Seaborn and Dullien (2015); Pessl et al. (2016); Gruss et al. (2016a); van der Veen et al. (2016); Razavi et al. (2016); Qiao and Seaborn (2016); Jang et al. (2017); Bosman et al. (2016); Gruss et al. (2018); Tatar et al. (2018b); Lipp et al. (2018a); Tatar et al. (2018a); Zhang et al. (2018); Cojocar et al. (2019); Kwong et al. (2020); Yao et al. (2020); Zhang et al. (2020a); Lenovo Inc. (2015); Lanteigne (2016); Aweke et al. (2016); Irazoqui et al. (2018); Schwarz et al. (2017); Corbet (2016); Konoth et al. (2018)

2.1.1 Rowhammer Attacks

Shortly after Kim's work, Google Project Zero did deeper investigations into the rowhammer bugs Seaborn and Dullien (2015). This work studied how the address selection strategy affects bit-flip difficulties, proposed multiple routes to perform hammering, and performed a series of tests on different machines to see how vulnerable they are to rowhammer attacks. Furthermore, they provided two examples of exploiting the rowhammer bug to perform privilege escalations on real systems. They showed that a Native Client (NaCl) program could escape from the sandbox environment by altering a validated, safe instruction to an unsafe one. The other exploitation showed that an unprivileged process on Linux could take over the entire physical memory space through modifying the page table entry(PTE). As Kim et al. (2014) first gave a comprehensive introduction of the rowhammer bug, Seaborn and Dullien (2015), on the other hand, demonstrated the non-negligible threats of rowhammer attacks. Later research showed more rowhammer attacks implemented in various environments, introduced more techniques that enhance rowhammer abilities, and demonstrated more exploits of rowhammer attacks. In Gruss et al. (2016a), rowhammer attacks were implemented from a remote using JavaScript. Attacks using JavaScript are more difficult than those using native code due to limited available instructions and less information about the system. The authors in this paper presented a proof-of-concept attack without any external information and user interactions. A cross-VM attack was presented in Razavi et al. (2016). By exploiting the memory deduplication feature, an attacker VM can make a victim VM's memory page be mapped to a physical location where bits are known to be vulnerable to rowhammer attacks exist. Hence, a bit flip can be triggered on the victim VM's memory page to modify the modulo used in RSA, which can significantly lower the difficulty of breaking the RSA cryptosystem. This cryptoanalysis could be further exploited to attack applications like OpenSSH and GPG on the victim VM. Xiao et al. (2016) is another work performing cross-VM attacks. The authors also provided a method to reverse-engineering the address mapping between physical addresses and DRAM locations in this work. Similarly, this mapping reverse-engineering was also performed in Pessl et al. (2016) to assist the rowhammer attacks. Rowhammer attack implemented on ARM devices was presented in van der Veen et al. (2016).

This work presented the first generic technique that can deterministically put target data in an attack-chosen vulnerable physical location. In Gruss et al. (2018), a new rowhammer attack was presented that can bypass all previous rowhammer defenses. Intel SGX feature was abused to hide the attacks from users and operating systems so that the performance counter-based rowhammer detection techniques are invalidated. They proposed a new method for steering the targeted data to a vulnerable location called memory waylaying, which is more stealthy compared to other memory steering methods. The authors also provided a new memory access pattern named one-location hammering that can trigger bit flips with memory controllers using closed-page policies. While most rowhammer attacks focused on breaching the data integrity, Kwong *et al.* first proposed a rowhammer attack that compromises the data confidentiality Kwong et al. (2020). This was achieved based on the finding that a bit flip only occurs when the hammered aggressor rows next to it have opposite bits stored in them. When two aggressor rows around a vulnerable bit of interest are controlled by an attacker, the bit value can be inferred by examining whether a bit flip happens using a timing side-channel.

Because the CPU cache is widely deployed in modern computer systems to achieve fast memory access, in order to perform rowhammer attacks, a lot of prior work has studied how to find methods avoiding the cache effects and access memory directly. The most common and straightforward way is using instructions like `clflush` after each memory access to flush the stored data out of CPU caches Kim et al. (2014); Seaborn and Dullien (2015). However, cache flush instructions may not be available in some execution environments. For example, as a countermeasure for rowhammer attacks performed in NaCl `clflush` instructions are disabled after Seaborn's work Seaborn and Dullien (2015). Consequently, much research found many so-called `clflush`-free rowhammer triggering methods. In Aweke et al. (2016), the authors proposed to perform fast repeated memory using specially-designed cache eviction strategies after studying the pseudo least recent used (PLRU) cache replacement policy. Similarly, a cache eviction-based rowhammer attack was also implemented in Gruss et al. (2016a). They launch the attack from a website with JavaScript, where cache flush instructions are not available. Another cache-eviction-based rowhammer triggering method was provided in Aga et al. (2017), where they also abused Intel's Cache Allocation Technology (CAT) feature to lower the difficulty of cache eviction. Another method for circumventing the cache effects is using non-temporal memory access instructions. In Qiao and Seaborn (2016), instructions like `movnti` and `movntdq` are found useful on x86 platforms to trigger the rowhammer bug by performing non-temporal stores. The possibility of using non-temporal memory access to trigger the rowhammer bug was discussed in van der Veen et al. (2016). However, this method was found not useful because the non-temporal memory access instructions only serve as a hint instead of guaranteeing the accessed data is uncached. Nevertheless, Zhang et al. (2018) disclosed a previously overlooked non-temporal memory instruction `dczva` that can be used to successfully trigger the rowhammer bug on ARMv8 platforms. Besides trying to issue the memory access command from the CPU, the rowhammer attack was also shown possible using memory access performed by integrated GPU Frigo et al. (2018). An integrated GPU can perform rapid and repeated memory access to the main memory to induce bit-flips. Direct memory access (DMA) is another technology frequently exploited to perform rowhammer attacks van der Veen et al. (2016); Tatar et al. (2018b) because it allows some services and devices to directly access the memory without CPU intervention.

2.1.2 Rowhammer Defenses

In Kim's work Kim et al. (2014), some mitigation techniques are proposed to protect devices against the rowhammer attack. However, most of them do not work due to the impractical hardware modification requirements, significant cost increases, and considerable performance overhead. In this section, we mainly concentrate on existing rowhammer defenses that do not require unrealistic hardware modifications.

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 Lenovo Inc. (2015). However, as shown in several tests Lanteigne (2016); Aweke et al. (2016), this approach still cannot prevent the bug from being triggered, especially if the double-sided hammering technique is used Seaborn and Dullien (2015). Another straightforward defense is to use ECC memory to correct or detect bit flips Kim et al. (2014), but it has been demonstrated that reliable rowhammer attacks in the presence of ECC memory are still highly possible Cojocar et al. (2019); Kwong et al. (2020).

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 Seaborn and Dullien (2015); Qiao and Seaborn (2016), but they cannot hinder eviction-based hammering Aweke et al. (2016); Gruss et al. (2016a). 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 Irazoqui et al. (2018). However, encryption and secure enclaves can be used to hide any malicious intention from static analysis Gruss et al. (2018); Schwarz et al. (2017).

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 Herath and Fogh (2015); Aweke et al. (2016). 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 Schwarz et al. (2017); Gruss et al. (2018), 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 Aweke et al. (2016); Corbet (2016). 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) Gruss et al. (2018); Lipp et al. (2018a), 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 Seaborn and Dullien (2015) and memory grooming van der Veen et al. (2016), need to allocate a large portion of memory, prevention of memory exhaustion has been considered as a feasible countermeasure van der Veen et al. (2016); Gruss et al. (2016a). Moreover, in Brasser et al. (2017), 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 Gruss et al. (2018) and memory ambush Cheng et al. (2018), 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 van der Veen et al. (2018); Tatar et al. (2018b). To enable defenses against more general rowhammer attacks, ZebRAM is proposed in Konoth et al. (2018) 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.

2.2 Physical Side-Channel-Based Defenses

There has been much research work on exploiting physical side-channel information for attacks Kocher et al. (1999); Gandolfi et al. (2001); Quisquater and Samyde (2001); Agrawal et al. (2002); Kuhn (2004); Backes et al. (2010); Enev et al. (2011); Schlösser et al. (2012); Heyszl et al. (2012); Sugawara et al. (2013); Genkin et al. (2015a, 2016a,b, 2017); Alam et al. (2018). However, many researchers have also examined how to leverage such side-channel information to help defenses. Traditionally hardware and software solutions were used widely to mitigate malware, which usually requires modification of the software or hardware design. Side-channel defense techniques are novel techniques for detecting malware by looking for the physical side-channel effects caused by computations performed by malware.

Agrawal *et al.* first proposed the idea using side-channel fingerprints of ICs to detect the hardware Trojan Agrawal et al. (2007). They described a theoretical framework and demonstrated preliminary experiments using power simulations. In their work, power traces were collected to build side-channel fingerprints. The Trojan circuits may be identified by comparing the fingerprints between the selected circuit and the original profiled circuits. This technique could effectively detect hardware Trojan circuits down to 0.01% of the size of the main circuit in simulations and showed the feasibility of detecting hardware Trojans in real ICs. Agrawal's technique in Agrawal et al. (2007) was improved by Rad by using localized power consumption traces Rad et al. (2008). In Rad's work, power traces measured from neighboring power ports are measured and compared to obtain power supply transient traces and help detect hardware Trojan. Aarestad proposed a similar region-based technique in Aarestad et al. (2010). A chip's steady-state current at multiple regions of the chip's surface was measured, and linear regression analysis was used to improve the sensitivity of hardware Trojans of previous approaches. The physical side-channel-based hardware Trojan detection approaches proposed above all require a profiling stage using circuits guaranteed Trojan-free (golden chips) to obtain a reference fingerprint. Narasimhan proposed a novel approach that does not need other circuits for profiling Narasimhan et al. (2011). Instead, the traces collected from the same circuit at different time windows were referenced. In a Trojan-infected circuit, when the same set of state transitions were performed, the uncorrelated state transitions in Trojans will cause variance in current signatures, and this feature could be used to identify the hardware Trojan. This work was later extended in Hoque et al. (2017) through improving Trojan detection coverage, detecting sequential Trojans, and improving detection capability. After Narasimhan's work, more golden chip-free hardware Trojan detection techniques were proposed. He proposed to detect hardware Trojans using EM side-channel He et al. (2017). The reference traces were obtained using a model simulating EM radiations when different signal transitions happen instead of measuring emissions from a golden chip. By comparing the frequency spectra of measured traces and reference traces, the insertion of a Hardware Trojan can be detected. A very recent work proposed to use backscattering side-channel effects to detect hardware Trojan Nguyen et al. (2020). Because any malicious modifications on circuits inevitably affect the impedance changes, which further cause differences in backscattered signals. These differences can be used as features to classify compromised and Trojan-free chips.

Besides detecting hardware Trojan in ICs, there are many other ways physical side-channel can be used to improve the security. Clark presented a power-consumption-based technique to detect malware on medical devices Clark et al. (2013). Power consumption traces were collected when normal and abnormal activities are performed on the secured device, and these traces were used to train a classifier through supervised learning. The evaluations showed that this technique has much higher malware detection effectiveness than traditional anti-virus software. Liu *et al.* proposed a power side-channel-based code execution tracking technique Liu et al. (2016). A hidden Markov model was used to examine the power traces for performing control-flow integrity checking and detecting anomalies' possible existence. Sehatbakhsh proposed a novel method using EM signals for program profiling in Sehatbakhsh et al. (2016). When a repetitive program activity (e.g., a loop) is performed, the active components will generate direct EM emanations with energy concentrated in a narrow frequency range or AM modulate other carrier signals. As a result, a feature can be observed in the frequency spectrum by looking for the "spikes" corresponding to direct emanations of components or modulating signals at the sidebands of AM modulated signal. Usually, different activities performed have different distributions of execution time and cause "spikes" at different frequency ranges in the spectra; thus, these spikes can be seen as fingerprints of certain activities. The ongoing activity can be identified, and the program execution can be successfully profiled by monitoring the spectra. Later this work was extended to detect anomalies in program execution by using EM emanations Nazari et al. (2017). The spike-monitoring techniques described in Sehatbakhsh et al. (2016) were used to profile programs. When a program is running, spikes in spectra corresponding to the program were identified. If malicious code is injected into the program, the execution will be slightly influenced, which can be observed from the positions and shapes found in spectra. The program profiling and program execution anomaly detection techniques described above are advantageous in terms of not using any resources from the monitored system. Han presented an EM-based contactless control-flow monitor ZEUS to ensure the execution control flow integrity of embedded programmable logic controllers (PLCs) Han et al. (2017). The idea is similar to program profiling approaches proposed in Sehatbakhsh et al. (2016). At the profiling stage, the EM signal traces were collected when different PLC instructions were executed. A long short-term memory deep neural network model was trained to distinguish the spectra corresponding to different instructions. Then the trained model was used to identify PLC instructions being executed and exam the controller's integrity. Because this monitoring technique is based on the physical side-channel analysis performed on an independent device, it does not introduce overheat to real-time systems with tight constraints. And the air gap between this monitor guaranteed that the controller would not be affected even if the monitor is hacked. Cheng provided the first fingerprinting work based on the CPU module's fingerprint in Cheng et al. (2019). In this work, the low-frequency magnetic inductions generated by inductors from CPU's voltage regulators were used to identify devices. A stimuli program is needed to run and keep a stable workload on the CPU. Then the EM induction was collected and preprocessed to confirm the spatial and temporal consistency. Features were selected from both the time and frequency domain of the processed signal to construct the fingerprint. The evaluations showed that this fingerprinting technique could even be used to identify identical laptops and mobile phones with 98.1% precision. With a longer fingerprinting time, the precision could be further improved. Wei *et al.* proposed a technique using power side-channel information to detect miro-architecture attacks Wei et al. (2019). In this work, power traces are collected, and machine learning classifiers are trained to identify attacks. There are also side-channel defenses include identifying the attacker ECU on in-vehicle networks Cho and Shin (2017), detecting intellectual property theft Becker et al. (2010); Strobel et al. (2015), 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 Wei et al. (2019). Our work is the first one on leveraging EM side-channel information to detect rowhammer attacks.

2.3 Electromagnetic Side-Channel Information Leakages

Since the first power side-channel attack was presented by Kocher *et al.* in Kocher et al. (1999), a tremendous amount of research has been conducted on physical side-channel analysis. Studied physicial side-channel effects includepower consumption Kocher et al. (1999); Messerges et al. (1999); Biham and Shamir (1999); Clavier et al. (2000); Messerges (2000); Coron et al. (2000); Mayer-Sommer (2000); Akkar et al. (2000); Brier et al. (2004); Chari et al. (2002); Agrawal et al. (2007); Genkin et al. (2015b); Zhao and Suh (2018); Pant (2008); Le Masle and Luk (2012); Su et al. (2017), EM emanations Rao and Rohatgi (2001); Quisquater and Samyde (2001); De Mulder et al. (2005); Aboulkassimi et al. (2013); Carlier et al. (2004); Heyszl et al. (2012); Genkin et al. (2016a); Alam et al. (2018); Sauvage et al. (2009); Merli et al. (2011); Longo et al. (2015); Guri et al. (2015a); Gilbert Goodwill et al. (2011); Batina et al. (2019), acoustic emissions Asonov and Agrawal (2004); Zhuang et al. (2009); Berger et al. (2006); Backes et al. (2010); Marquardt et al. (2011); Zhu et al. (2014); Genkin et al. (2014); Guri et al. (2020, 2017a); Kwong et al. (2019), photonic emissions Tsang et al. (2000); Ferrigno and Hlaváč (2008); Di-Battista et al. (2010); Skorobogatov (2009); Schlösser et al. (2012); Bertoni et al. (2015); Krämer et al. (2013); Carmon et al. (2016, 2017), and thermal radiations Hutter and Schmidt (2013); Brouchier et al. (2009); Masti et al. (2015); Guri et al. (2015b); Zalewski (2005); Mowery et al. (2011); Wodo and Hanzlik (2016); Kaczmarek et al. (2018); Andriotis et al. (2013); Abdelrahman et al. (2017). Among all these physical side-channels, the EM side-channel draws the most attention compared to the rest of them for the following reasons:

- EM side-channel analysis can be implemented without physical contact with the target device
- Many EM sources found in modern electronic devices can generate signals covering an extraordinary wide frequency range.

- EM signals have the potential of carrying a vast amount of information.
- EM analysis sometimes requires less physical proximity than the other signals and can be implemented with non-metal obstacles between the receiver and target device.

Given the fact that electric current in the circuitry of a device varies with time, EM emanations inevitably arise. The EM emanations generated by an electronic device are distributed widely in the spectrum. Not long after Kocher's work Kocher et al. (1999), EM side-channel effects were found exploitable for attacking cryptography systems. For example, SPA, DPA, and CPA can be implemented as SEMA, DEMA, and CEMA respectively Rao and Rohatgi (2001); Quisquater and Samyde (2001); De Mulder et al. (2005); Aboulkassimi et al. (2013).

The idea of electromagnetic analysis was first proposed by Jean-Jacques et al. Quisquater and Samyde (2001) by providing an approach that uses electromagnetic emanations of smart card processors to acquire cryptographic information in smart cards. In this framework, an EM sensor was placed very close to the smart card in a card reader. The idea is similar to power analysis proposed in Kocher et al. (1999), with the power consumption traces replaced by EM traces. Comparison between measured power consumption and EM emanations indicated that both traces have a similar pattern, but EM signals have lower noise. They also enumerated several countermeasures of EMA, including noise addition, EM radiation shielding, power consumption reduction, using asynchronous processors, dual-line logic, and modification architectures and design of the chip. When EMA was first proposed, the exploited EM signals are mainly emitted by the power lines, and the attacking methods follow the implementation of power analyses.

After the theory of EMA was proposed and developed, few practical experiments were conducted to launch attacks on real cryptography systems until Gandolfi's work Gandolfi et al. (2001). Karine used EM side-channel effects to attack actual cryptographic algorithms implemented on CMOS chips. A hand-made coiled copper wire probe was used in the experiments. Different circuit positions were probed, and the EM emanation around the CPU was found to be most data-dependent. They also mentioned that the measured signal strength could be stronger by placing the probe closer to a decapsulated chip. This idea was widely adopted in many later works to implement localized EMA Heyszl et al. (2012). The evaluations were performed by attacking DES, COMP128, and RSA. The performance of DEMA and DPA were compared, and in all situations, the EM signals had lower noise, and better results were obtained using DEMA.

Dakshin presented a systematic investigation of EM side-channel leakage from CMOS devices in Agrawal et al. (2002). The EM emanations were categorized into direct and indirect/unintentional emanations. The direct emanations are EM signals directly induced by the current flows in circuits. These direct emanations usually have very short propagation distances, and the probes must be put very close to circuits to obtain exploitable signals. Sometimes the current flows can modulate other strong carrier signals (e.g., clock signals) through coupling, and the modulated EM signal can be emitted from the components generating carrier signals. This so-called unintentional emanation was believed to be more useful than direct emanation because it had better propagation ability. Less invasive attacks can be carried out with the probes positioned further from target systems. This work highlights the strength of EM side-channel analysis by showing that the unintentional emanations can be observed and exploited even when the probe is far away from the target devices. It was also mentioned that multi-channel analysis could obtain more leakage than single-channel analysis.

Early EM analysis mainly targeted cryptography systems implemented in simple circuits like smart cards. Fewer attacks were implemented on widely used desktop and laptop personal computers because these devices are more complicated, have much higher clock speed, and generate more unpredictable noise. However, some recent research Zajic and Prvulovic (2014); Callan et al. (2015) revealed that information leakages on personal computers are greater than those simple devices because of the enormous power consumption.

The information leakage from processor-memory systems of modern desktop and laptop computers was first presented in Zajic and Prvulovic (2014) through experiments. Many experiments are performed on different systems to evaluate all kinds of properties of program-activity-dependent EM emanations. Direct emanations from components in different levels of the memory systems are measured. For all systems, transmission distances at different frequencies and directions are evaluated. The results indicated that EM information leakage could be reliably received at varying distances in all tested systems, and memory tends to produce the strongest EM emanation. Experiments evaluating shielding effects showed that metal shields could only decrease but not eliminate leakage through EM side-channel. The evaluation of signal dependence on data values showed that data being accessed have little influence on the EM signals most of the time.

While Zajic and Prvulovic (2014) highlighted the leakage of direct emanations from components in modern computers, Callan et al. (2015) raised people's awareness of unintentional emanations by developing a technique that can effectively find EM emanations that are amplitude-modulated by processor or memory activities. A series of micro-benchmarks varied in frequencies were executed on tested computers, and AM-modulated carriers could be identified by looking for expected sideband patterns of the modulated signal in the spectrum. They used this technique to successfully identify multiple AM-modulated signals emitted by various components, including voltage regulators, DRAM, and CPU cache.

Genkin presented the first physical side-channel attack on elliptic curves implementation on PC using EM side-channel effectGenkin et al. (2016a). The exploited software vulnerability is the operand-dependent execution of Libgcrypt's point addition operation. When operands of different lengths are used in the targeted software, different executions result in different EM emanations. A chosen ciphertext was input for decryption multiple times, and the EM signals collected at all decryption rounds are collected, aligned, and averaged. Then the averaged traces are displayed as a spectrogram in the frequency domain, and energy at a certain frequency range is used to identify the presence of point addition operation. The energy distribution over the frequency range during the operations is used to identify the operand of point addition. This attack exposed the vulnerabilities of personal computers in a very practical situation.

In 2018, Alam proposed a new EM attack targeting fixed-window RSA implementation on ARM processors Alam et al. (2018). This was the first attack that exploits the leakage during the brief computation of the lookup table index used in each window. Most prior attacks on RSA targeted the long-duration large-integer multiplications; thus, the brief lookup table index computation was left unprotected. Alam found that slight variance can be observed when different bit values are added to different bit positions of the computed lookup index. In the training stage, for each possible bit value added to each position, multiple reference traces are obtained using K-Means clustering for accounting for other unidentified noise. Then the key bits can be successfully identified by comparing with the reference traces at high accuracy. A great effort was spent in this attack to precisely locate and align the targeted execution because of the extremely short duration. Because the index computation always follows the squaring of large integers, a noticeable feature at the end of large-integer multiplication was used to locate the exploited trace. Using the features in large-integer multiplication, the execution sequences of a whole RSA decryption can be recovered. This attack was implemented non-intrusively on high-clock-rate devices and only required single decryption to recover the key.

Most power analyses exploit secret-dependent global power consumption, and many EM global EM analyses also take advantage of the EM emanation correlated with the global power consumption. However, sometimes secret-dependent operations increase the power consumption of certain components while decreasing the power consumption of other components. The measured global power consumption difference is thus canceled, and the leakage is harder to be found. Therefore, localized EM analyses were proposed to measure local power consumption and locate the position with maximum leakage Sauvage et al. (2009). Marquardt showed that the most leakage-intensive location for localized EM analysis could be more efficiently found using CEMA Marquardt et al. (2011). Sauvage proposed an innovative method of measuring location-dependent leakage based on cross-correlation Sauvage et al. (2010). The implementation of this method does not require any knowledge of the target device. Heyszl et al. (2012) first generalized localized EM side-channel attacks against cryptography systems. A high spatial resolution probe was used to measure the EM emanation from a decapsulated FPGA implementing Elliptic Curve Scalar Multiplication (ECSM). A known scalar was input to ECSM, and the measured EM traces were grouped, aligned, and averaged based on the secret bit. The differences between two averaged traces at different locations indicated that leakage at different positions varied significantly, and the global leakage is much more minor. With the location with maximum leakage identified, a single trace measured is enough to recover a secret scalar input to ECSM.

Physical Unclonable Functions (PUFs) are circuit primitives used to derive secrets from the physical characteristics of circuits. Ring Oscillator PUFs proposed by Suh in Suh and Devadas (2007) were found to be vulnerable to global EM analysis, and countermeasures were proposed in Merli et al. (2011). In the protected RO PUFs, ROs are separated into several non-overlapped groups. ROs in each group are connected to a multiplexer and further connected to an asynchronous counter. The final outputs are obtained by comparing the counter's values, so the key to attacking an RO PUF is obtaining the frequency of each RO. In Merli et al. (2013) localized EM analysis was implemented to circumvent the mitigation against global EM attack on RO PUFs. Due to the limited spatial resolution, locations of ROs can not be easily distinguished, so the locations of logic and registers connected to ROs(e.g., multiplexers and counters) are examined. When different RO select sequences are applied, the asynchronous counters' frequency is the same as the corresponding selected ROs. Similar to Heyszl et al. (2012), the first step of this attack is locating the counters by looking for the locations causing maximum leakages(i.e., maximum deviations in the frequency spectra). With enough traces, the location of the path each counter is connected to was identified by clustering the locations obtained above. Then different RO select sequences are applied, and the frequency in each path was be measured, which correspond to the selected RO's sequence in that path. Repeat this step while changing select sequences, and all RO's frequency can be identified. Finally, matching the frequencies in every run with the output allowed the comparison function to be easily found, and the entire RO PUF's behavior became predictable.

Besides attacking simple circuits, localized EM attack was also successfully implemented on high-end devices. Longo investigated the leakage of SoC-based devices executing cryptographic algorithms Longo et al. (2015). This work characterized the side-channel leakage for software-based AES, Hardware AES, and NEON core. The leakage was quantified using the Test Vector Leakage Assessment (TVLA) methodology of GoodwillGilbert Goodwill et al. (2011). While attacking the software AES, the probe was put near the SoC. EM traces collected during AES implementation were filtered by removing traces that are seriously affected by noises. Sub-traces for each encryption operation were extracted and matched with associated ciphertexts. The results indicated that fewer than 100 traces were needed to identify the leakage. To attack hardware AES, the alignment of traces is realized through triggering the AES operations with interrupts. A process coresident on the target device is needed to saturate the DMA engine so that any memory management causes an interrupt. 500,000 traces were collected, and wavelet post-processing was performed to obtain exploitable traces with maximum SNR. A single-bit correlation-based attack was implemented to recover the key. Attacking hardware AES required much more effort, but it was still proven possible using approaches in Longo et al. (2015). The leakages on NEON, a general-purpose SIMD extension to Cortex A-series ARM cores, were also measured. Operation-dependent leakages and write-back data-dependent leakages were identified. Then the leakage from the hamming weight of arithmetic, logic, and comparison operations' results written back was found, and this leakage can be used to distinguish a single sub-word in the results. This hamming weight leakage was exploited to recover the key of NEON-based bit-sliced AES implementation using 5,000 traces collected during decryption.

In Batina et al. (2019) EM side-channel analysis was used to reverse engineering a neural network model on a 32-bit ARM microcontroller. This method was demonstrated on two widely used algorithms: multi-layer perceptron (MLP) and convolutional neural networks (CNN). The EM emanations are collected when known random values are input to a black-box neural network to perform this attack. The author successfully identified the activation functions being used, recovered the number of neurons and layers, and estimated the weights by analyzing the collected samples. Activation functions used in a neural network were identified based on the execution time. Weights of a trained neural network could be estimated using CPA/CEMA. Recovering the number of neurons and layers consists of two steps. First, SPA/SEMA was used to identify the boundaries of different neurons. Then the boundaries of different layers were found using CPA/CEMA. In this paper, the reverse-engineered MLP and CNN models have similar performance to the actual model. The only problem is that the evaluated models in this paper are executed in serial while most neural network models in real-world are executed in parallel. So the practicalness of this attack could be a problem.

Similar to our work in Chapter 5, Sehatbakhsh *et al.* have also exploited EM emanations involved with power management for attacks very recently Sehatbakhsh et al. (2020). While there are similarities, fundamental differences exist. *First*, our work leverages the effect of DVFS on GPU clocks, whereas their work uses the effect of demand-based switching on CPU voltage regulators. *Second*, although there is some overlap in terms of inferring keystroke timing, the work presented in Sehatbakhsh et al. (2020) mainly focuses on building covert channels for data exfiltration, while our work focuses more on stealing sensitive information. *Finally*, our work demonstrates that disabling DVFS as a countermeasure may not be fully effective.

2.4 Physical Covert Channel Attacks

The confinement problem was brought forth by Lampson in 1973 Lampson (1973), which made the first mention of possible data exfiltration via covert channels. Since then, extensive research has been conducted on this topic. Basically, a covert channel is an unintended communication channel that can be used to transfer information between a sender and a receiver. Depending on the construction, covert channels can be classified into logical and physical ones. Logical covert channels usually manipulate the microarchitectural states in a processor to encode and transfer information Szefer (2019), and the receiver normally runs on the same processor/platform/cloud as the sender Wang and Lee (2006); Ristenpart et al. (2009); Xu et al. (2011);

Maurice et al. (2015); Wu et al. (2012); Masti et al. (2015); Sullivan et al. (2018). On the other hand, physical covert channels are usually used to enable illegitimate communication between air-gapped computers and are constructed from certain physical side effects of computation. In this section, we will mainly focus on physical covert channels.

Successful implementation of physical covert channel is based on the assumption that a victim computer is infected by malware which behaves as an attacker and serves as a sender. The sender program inserted in the victim computer intentionally performs computations to manipulate the physical side effects in a controlled way such that the information to exfiltrate can be encoded into the physical side effects. These physical side effects can propagate over a certain distance in the air so that they can circumvent air-gapping(isolate a computer from unsecured networks), the strongest protection against data exfiltration. A receiver deployed nearby can measure these physical effects and recover the embedded information. Because an attacker has full control over the sender program, the covert channel can be established even with only a small amount of physical side-channel leakages available Guri et al. (2015b). Numerous research has been conducted to implement various physical covert channels exploiting all sorts of physical side-channel information, including acoustic, optical, thermal, power, electromagnetic emanations.

Many components in a running computer can produce sounds that can be received by any device equipped with a microphone(e.g., laptop, cell phone, smartwatches, etc.) to establish a covert channel. Carrara *et al.* first used speakers and microphones on computers to communicate through ultrasound Carrara and Adams (2014). Furthermore, Hanspach *et al.* leveraged ultrasound to establish a covert acoustical mesh network Hanspach and Goetz (2013). Because ultrasound frequencies are higher than the upper audible limit of human hearing, communication cannot be easily noticed. Later, Guri *et al.* designed speakerless acoustic covert channels, where cooling fans Guri et al. (2020), hard disk drives Guri et al. (2017a) and power supply unit Guri (2020) were used to generate acoustic emissions. However, these two speakerless covert channels transmit information using audible sound instead of ultrasound. Even though most people do not pay much attention to the components' noise, it is still possible for perceptive people to notice the abnormal noise, which makes these methods less stealthy. One advantage of the acoustic covert channel is that the transmission distance is long because sound waves have a long propagation distance in the air. Nevertheless, acoustic signals have high attenuation in most materials and can be easily mitigated by spatially isolating protected computers in a secured room.

Optical emissions can also be exploited to create covert channels. The exploitable optical emissions may be generated by light-emitting diode (LED) in components like keyboards Loughry and Umphress (2002), monitors Sepetnitsky et al. (2014), and even hard disk drives Guri et al. (2017b). Most LED-based optical covert channels use OOK modulation, and Zhou *et al.* showed that the efficiency could be improved by replacing OOK modulation with binary frequency-shift keying modulation (B-FSK) Zhou et al. (2017b). Another kind of optical covert channel manipulates the monitor screen Guri et al. (2016a, 2019). By modifying a small amount of content displayed on the screen, information may be transmitted without being noticed by humans. Theoretically, optical covert channels can reach a very high bandwidth with the help of optical instruments as long as the sender is in the sight of the attacker. However, exploiting optical emissions is harder than expected in practice because it is rare that a malicious camera can monitor a highly secured target machine. In addition, it is very difficult, if not impossible, to create optical covert channels when the target machine is enclosed in a room with non-transparent walls. Similar to acoustic covert channels, some optical emissions like abnormal blinking of LED can also raise the administrator's suspicion.

No successful side-channel analyses were performed exploiting thermal side-channel effects due to the short propagation distance and high delay effects. However, an attacker with full control of the sender in a covert channel can exploit the thermal leakage by accumulating the signal to improve the SNR. A thermal covert channel was constructed in Guri et al. (2015b) to transmit information between two physically adjacent but air-gapped computers. Because the heat source and thermal sensor are all components planted in computers, this is one of the few covert channels that can realize two-way communication without any external devices. However, the performance of this covert channel is extremely poor. The maximum bandwidth reported is 8 bits/hour, and the sender and the receiver must be very close to each other. A similar thermal covert channel can be implemented using a smartphone as a receiver to exfiltrate data from air-gapped computers Guri (2019). This covert channel also has the problems of low bandwidth and short transmission distance.

Power consumption is also exploitable for establishing covert channel Guri et al. (2018b). In this work, the CPU was manipulated to affect the power consumption of a computer to transmit information through power lines. The receiver can be mounted either on the in-home power lines directly attached to the electrical outlet or on the main electrical service panel. The bandwidth of this covert channel can reach 1,000 bps, but it requires the installation of malicious hardware devices on the power lines connected to victim machines, which makes this attack less practical.

Since many components (e.g., clocks and voltage regulators) in a computer have switching behavior and thus emit strong EM signals, several EM covert channels have been created. One way to exploit these EM emanations is measuring the magnetic field around the computer. It can affected by components like hard disk drives Matyunin et al. (2016) and CPUs Guri et al. (2018a,c). Measurement of the magnetic field can be performed using either specialized instruments like digital magnetometers or magnetic sensors equipped in hardware like cell phones. Magnetic covert channel is advantageous considering that the low-frequency magnetic emanations can travel through most obstacles including metal shields like Faraday Cages. However, this obstacle-penetrating ability doesn't mean the invalidation of spatial isolation mitigation. In fact, due to the short propagation distance and weak signal strength, magnetic covert channels are limited in both transmission speed and distance and spatial isolation can successfully mitigate magnetic covert channel in most cases. Compared to measuring magnetic field near a device, EM covert channels are more often implemented by receiving strong EM radiations over a longer distance that can pose more practical threat. For example, EM emanations from graphics card's clock were exploited to implement covert channel Davidov and Oldenburg (2020). Guri *et al.* implemented multiple EM covert channels by exploiting the EM emanations from either video display unit Guri et al. (2014), USB connectors Guri et al. (2016b), or DRAM bus clock Guri et al. (2015a). A recent study presented an EM covert channel Sehatbakhsh et al. (2020) exploiting EM emanations from power management unit capable of exfiltrating data at the bandwidth of 4,000 bps. This is the fastest EM covert channel prior to our work.

DRAM bus clock was shown to radiate strong EM emanations Zajic and Prvulovic (2014); Callan et al. (2015), which was exploited to establish the EM covert channel GSMEM Guri et al. (2015a). Similar to our *BitJabber* covert channel presented in Chapter 4, their *GSMem* covert channel described in Guri et al. (2015a) also relies on the EM signals related to the DRAM clock. They discovered that memory accesses could increase the strength of the EM signals in a wide frequency range around the DRAM clock frequency. Using the fact that EM emanations from DRAM bus clock are AM-modulated by memory activities, in *GSMEM* the bus clock is modulated through on-off keying (OOK), which is realized by controlling the presence/absence of intense memory access. The receiver collected the EM signals at certain frequency ranges and compute the amplitudes, which were used to decide the transmitted bits. What is special of GSMEM is that a DRAM bus clock's frequency is usually close to the GSM, UMTS and LTE bands, which make a modified cell phone be a possible receiver in the covert channel. In our work, *BitJabber* is implemented using a different carrier with a much higher SNR and new modulation techniques. *EMloRA* is a study parallel to our work Shen et al. (2021). Both *BitJabber* and *EMLoRa* are featured by explicitly addressing spread spectrum clocking to improved communication performance. However, *EMLoRa* focuses on ultra-long-distance data transmission with lower bandwidth while *BitJabber* aims at achieving extremely high-speed data exfiltration.

Comparisons of Existing Covert Channels

To highlight the advantages of *BitJabber* in Chapter 4, we compare the existing physical covert channels in Table 2.1. The comparisons are made in terms of their maximum achievable bandwidth and wall-penetrating ability. From the table we can see, before our work, the fastest physical covert channels was the one proposed in Guri et al. (2017b), which can achieve 4,000 bps. Compared to that covert channel, our *BitJabber* improves the performance by 75x.

Moreover, most of the existing physical covert channels have difficulties in penetrating physical obstacles like a wall. (We mark "maybe" on acoustic covert channels in terms of wall-penetrating ability, although we think it is very unlikely that they can actually penetrate a wall.) From the table, we can observe that the EM covert channels have considerable advantages over others in terms of penetrating walls. However, as illustrated in Chapter 4, when penetrating concrete walls, approaches like *GSMem* actually have a too large error rate (from 38% to 50%) to be actually used in reality, while our *BitJabber* has an error rate even less than 0.5%. Therefore, compared to other physical covert channels, it can be found that our *BitJabber* imposes more realistic security risks on air-gapped isolation protection.

Table 2.1: Comparison of existing physical covert channels.

Covert Channel	Type	Wall-Penetrating	Bandwidth
GPU Clock Davidov and Oldenburg (2020)	Electromagnetic	Yes	N/A
BitWhisper Guri et al. (2015b)	Thermal	No	0.002 bps
HOTSPOT Guri (2019)	Thermal	No	0.03 bps
Fansmitter Guri et al. (2020)	Acoustic	Maybe	0.25 bps
Matyunin Matyunin et al. (2016)	Magnetic	No	2 bps
DiskFiltration Guri et al. (2017a)	Acoustic	Maybe	3 bps
MAGNETO Guri et al. (2018a)	Magnetic	No	5 bps
Screen Brightness Guri et al. (2019)	Optical	No	10 bps
EMLoRA Shen et al. (2021)	Electromagnetic	Yes	14 bps
Monitor LED Sepetnitsky et al. (2014)	Optical	No	20 bps
ODINI Guri et al. (2018c)	Magnetic	No	40 bps
POWER-SUPPLaY Guri (2020)	Acoustic	Maybe	50 bps
UltraSonic Carrara and Adams (2014)	Acoustic	Maybe	230 bps
Keyboard LED Loughry and Umphress (2002)	Optical	No	450 bps
AirHopper Guri et al. (2014)	Electromagnetic	Yes	480 bps
USBee Guri et al. (2016b)	Electromagnetic	Yes	640 bps
GSMem Guri et al. (2015a)	Electromagnetic	Yes	1,000 bps
PowerHammer Guri et al. (2018b)	Power	N/A	1,000 bps
Hard Drive LED Guri et al. (2017b)	Optical	No	4,000 bps
PMU Sehatbakhsh et al. (2020)	Electromagnetic	Yes	4,000 bps
BitJabber Zhan et al. (2020)	Electromagnetic	Yes	300,000 bps

2.5 Security Problems on GPU

Given the rising popularity of modern GPUs, research on their security implications has begun drawing attentions in recent years. Several studies exploit possible residues in GPU memory that may not be properly cleared after use. Pietro *et al.* have shown that the lack of memory-zeroizing operations can be exploited to attack CUDA AES implementations Pietro et al. (2016), and the memory residual leakage vulnerabilities in virtualized and cloud computing environments have been investigated by Maurice *et al.* Maurice et al. (2014). Furthermore, Zhou *et al.* have studied how to extract raw images from GPU memory residues Zhou et al.

(2017a). In Lee et al. (2014), Lee *et al.* have successfully inferred the web browsing history by examining GPU memory dumps.

Other than relying on GPU memory residues, many attacks exploit possible logical or physical side-channel information of GPU to breach confidentiality. Jiang *et al.* have studied the timing differences due to contentions on shared GPU caches or memory banks and leveraged such timing side-channels to break AES implementations on GPU Jiang et al. (2016, 2019). Luo *et al.* have demonstrated that CUDA RSA implementations are also susceptible to timing side-channel analysis Luo et al. (2019). Aside from logical side-channel information, Luo *et al.* have also used GPU power consumption traces for cryptanalysis of CUDA AES Luo et al. (2015), while Gao *et al.* have used near-field localized EM emanations from GPU for the same purpose Gao et al. (2018). In addition to subverting GPU-accelerated cryptosystems, Naghibijouybari *et al.* have studied GPU side-channel attacks in a more general context, where they used performance counters or resource tracking APIs to measure shared GPU resource contentions and exploited such information to fingerprint websites, infer user activities, and reverse engineer neural networks Naghibijouybari et al. (2018).

GPUs can also be a new venue of which malware takes advantage to increase its stealthiness Vasiliadis et al. (2015). In Ladakis et al. (2013), Ladakis *et al.* have implemented a piece of GPU-based malware that directly monitors the keyboard buffer from GPU via DMA to log keystrokes, and in Zhu et al. (2017), Zhu *et al.* have conducted a more comprehensive study on such a topic. In Naghibijouybari et al. (2017), Naghibijouybari *et al.* have showed that malware may exploit contentions on shared GPU resources to construct covert channels for data exfiltration. In Davidov and Oldenburg (2020), Davidov and Oldenburg have also exploited GPU's EM emanations, but they only showed a malware-enabled covert channel on an AMD GPU.

CHAPTER 3

Rowhammer Attack Detection Via A Radio

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 Kim et al. (2014), 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 Kim et al. (2014); Lanteigne (2016). 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 Seaborn and Dullien (2015); Qiao and Seaborn (2016); Gruss et al. (2016a), privilege escalation Seaborn and Dullien (2015); Gruss et al. (2016a); Xiao et al. (2016); Bosman et al. (2016); van der Veen et al. (2016); Gruss et al. (2018), cryptography subversion Bhattacharya and Mukhopadhyay (2016); Razavi et al. (2016), denial-of-service Lipp et al. (2018a); Zeitouni et al. (2018); Jang et al. (2017), and even confidentiality breaching Kwong et al. (2020). Furthermore, rowhammer attacks have been effectively demonstrated in the presence of ECC mechanism Cojocar et al. (2019) as well as in the context of only sending network packets Tatar et al. (2018b); Lipp et al. (2018a).

In response, many defense techniques against rowhammer attacks have been proposed in recent years, including several detection-based approaches Aweke et al. (2016); Herath and Fogh (2015); Payer (2016); Irazoqui et al. (2018); Gruss et al. (2016b). Unfortunately, as more sophisticated rowhammer attacks are developed, the effectiveness of detection-based rowhammer defenses becomes questionable. As demonstrated in Gruss et al. (2018), 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 Irazoqui et al. (2018) or dynamic monitoring Aweke et al. (2016); Herath and Fogh (2015); Payer (2016); Gruss et al. (2016b) will become ineffective.

In this chapter, we introduce a new direction to addressing the problem of rowhammer attack detection[1]. 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 Gandolfi et al. (2001); Quisquater and Samyde (2001); Agrawal et al. (2002); Kuhn (2004); Enev et al. (2011);

[1]The work in this chapter has been previously published in Zhang et al. (2020b)

Heyszl et al. (2012); Genkin et al. (2015a, 2016a,b); Alam et al. (2018). However, it has been realized that, as a double-edged sword, such side-channel information can also be used to help build security defenses Han et al. (2017); Nazari et al. (2017). 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 chapter 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 chapter is organized as follows: Section 3.1 briefly sets the background; Section 3.2 formulates the threat model; Section 3.3 presents a new direction to rowhammer attack detection; Section 3.4 studies the correlation between EM side-channel information and rowhammer attacks; Section 3.5 proposes our RADAR system, which can achieve rowhammer attack detection in a non-intrusive manner; Section 3.6 evaluates the proposed RADAR system; Section 3.7 concludes this chapter.

3.1 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 chapter, namely the EM emanations.

3.1.1 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. 3.1, 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[2], 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 Gruss et al. (2018); Lipp et al. (2018a).

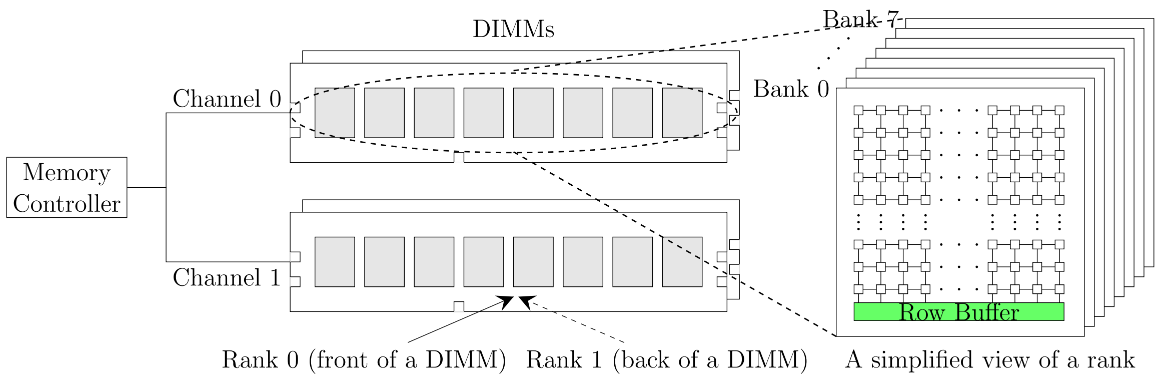


Figure 3.1: A representative DRAM architecture (two channels and four dual-rank DIMMs). A rank consists of all the chips on the same side (front or back) of a DIMM.

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

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

between 32ms to 64ms.

3.1.2 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 Mutlu (2017). First thoroughly studied in Kim et al. (2014), 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[3]. 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.

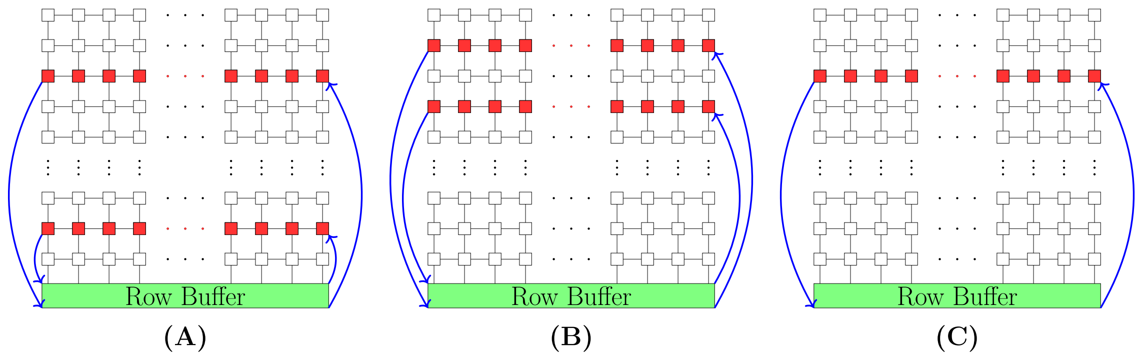


Figure 3.2: Three possible hammering techniques in the literature: (A) single-sided hammering Kim et al. (2014); (B) double-sided hammering Seaborn and Dullien (2015); and (C) one-location hammering Gruss et al. (2018).

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. 3.2 (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. 3.2 (B). As demonstrated in practice, double-sided hammering is much more effective and efficient than single-sided hammering Seaborn and Dullien (2015). 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 Gruss et al. (2018), as shown in Fig. 3.2 (C).

[3]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.

3.1.3 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 Seaborn and Dullien (2015), memory grooming van der Veen et al. (2016), memory waylaying Gruss et al. (2018), and memory ambush Cheng et al. (2018).

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 Seaborn and Dullien (2015); Qiao and Seaborn (2016). While memory deduplication can be exploited to ease this challenge Bosman et al. (2016), attackers have tried to recover such mapping information, especially by reverse engineering the physical-to-DRAM address mapping Pessl et al. (2016); Xiao et al. (2016); Tatar et al. (2018a), for more efficient and effective double-sided hammering.

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. 3.3 shows a typical computing platform, and each of the dashed lines in the figure represents a possible

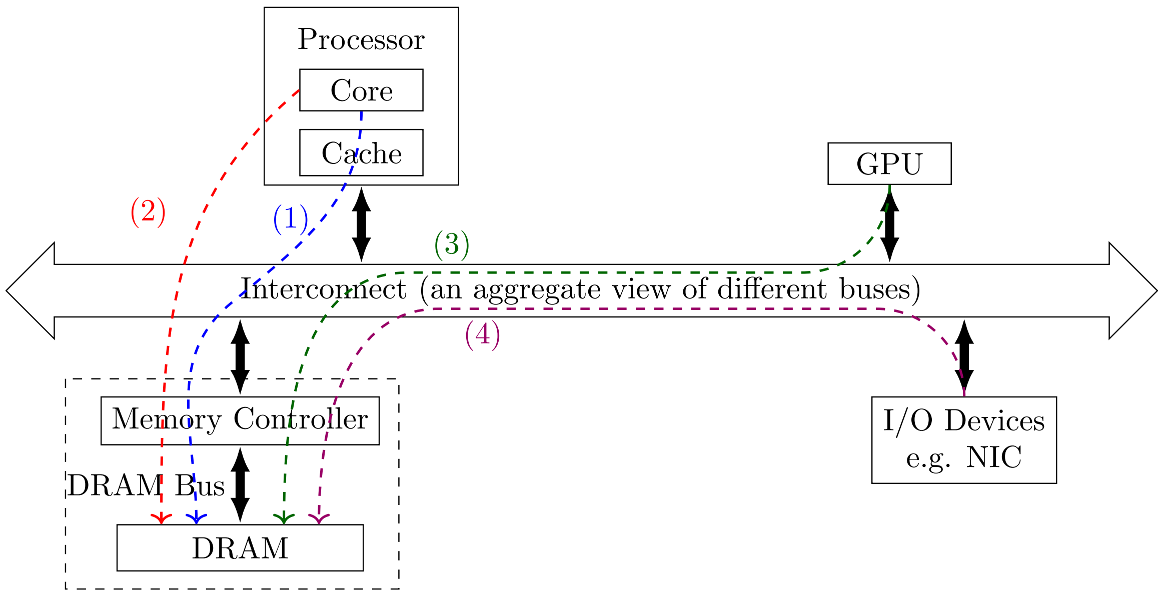


Figure 3.3: Different types of techniques for rapidly and repeatedly access the same locations in DRAM.

path taken to enable fast access to the same location in DRAM: (1) flushing or evicting CPU caches Kim et al. (2014); Aweke et al. (2016); Gruss et al. (2016a); Aga et al. (2017); (2) bypassing CPU caches Qiao and Seaborn (2016); van der Veen et al. (2016); (3) evicting GPU caches Frigo et al. (2018); and (4) maneuvering DMA buffers from I/O devices Tatar et al. (2018b).

3.1.4 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 Genkin et al. (2015a, 2016a,b); Alam et al. (2018), or inferring privacy Kuhn (2004); Enev et al. (2011). Yet, other than being exploited for side-channel attacks, EM emanations have also been used to track code execution for ensuring control-flow integrity Han et al. (2017); Nazari et al. (2017) or profiling Sehatbakhsh et al. (2016); Callan et al. (2016).

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) Callan et al. (2015); Prvulovic et al. (2017). 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 Callan et al. (2015). Therefore, these signals act as carrier signals that convey information about the modulating activities.

3.2 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 chapter, 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 chapter, 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 van der Veen et al. (2016); Frigo et al. (2018); van der Veen et al. (2018).

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.

3.3 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 Gruss et al. (2018); Cheng et al. (2018); Tatar et al. (2018a). 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. 3.3, 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 chapter, 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 3.1, EM emanations are inevitable physical side effects during any computation, issued both intentionally and unintentionally Agrawal et al. (2002); Zajic and Prvulovic (2014); Callan et al. (2015); Prvulovic et al. (2017).
- 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 Zajic and Prvulovic (2014); Callan et al. (2014, 2015); Prvulovic et al. (2017), 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.

3.4 Finding Hammering Information in EM Side-Channel Emanations

As mentioned in Section 3.1, 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.

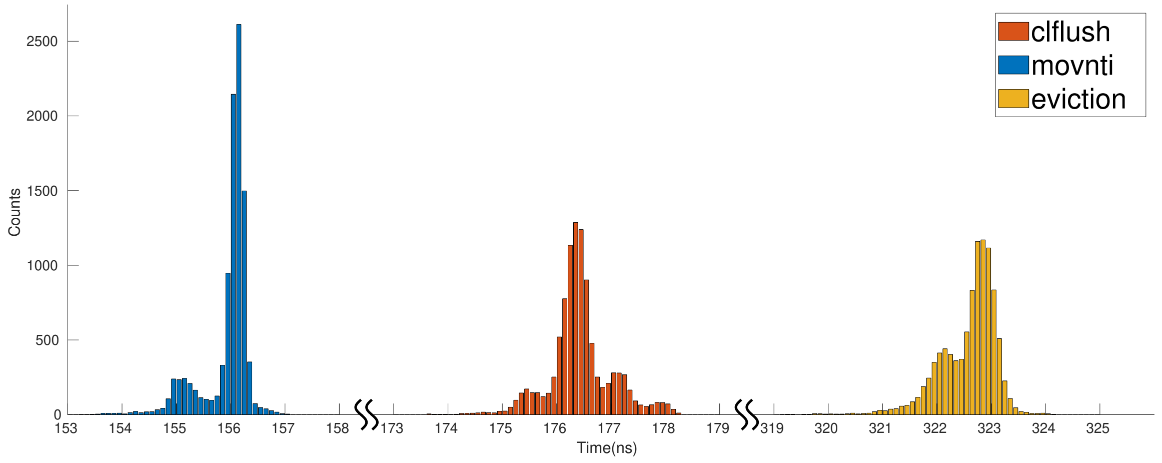


Figure 3.4: The timing distributions of 10,000 hammering iterations in terms of approaches using `clflush`, `movnti`, and eviction.

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. 3.4 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 chapter, 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.)

3.4.1 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.

3.4.2 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 Callan et al. (2015), 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 Callan et al. (2015), 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.

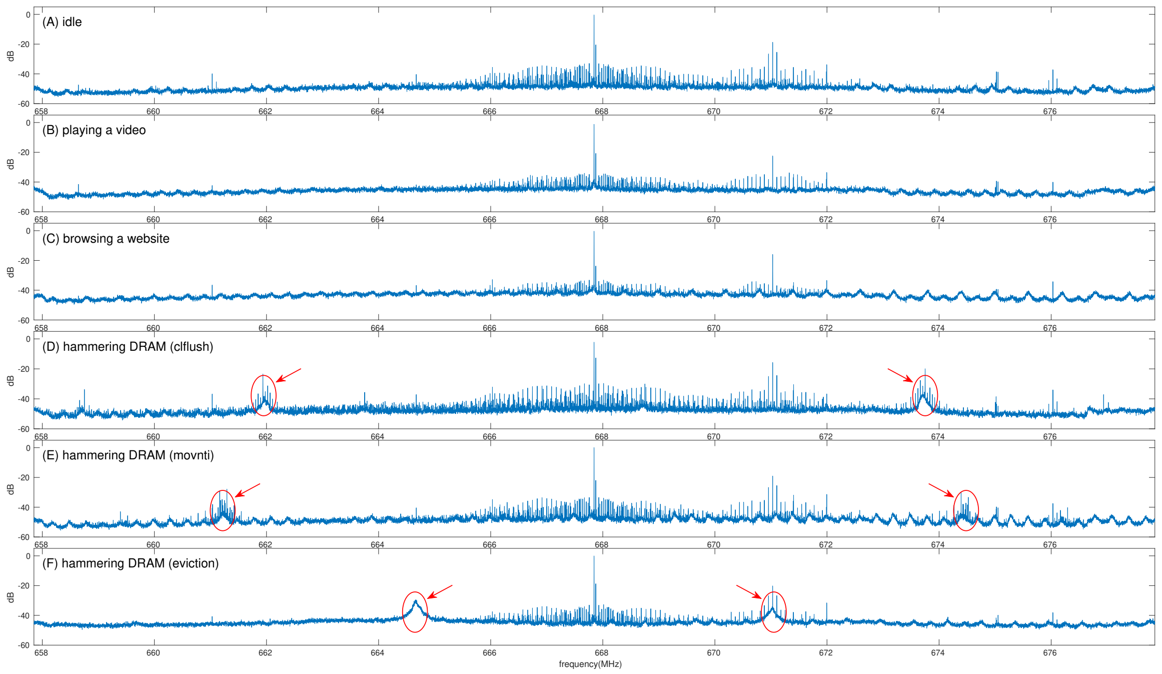


Figure 3.5: The power spectra under six scenarios. Note that the vertical axis is on a logarithmic dB scale. Each spectrum is derived by averaging 78 FFTs of 16,384 values with 50% overlap sampled in 25 MHz over 32 ms.

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. 3.5 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. 3.5. 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. 3.5, 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 and pointed to by arrows in Fig. 3.5 (D), (E), and (F). By referring to Fig. 3.4, 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. 3.4, we can see the dominant period of hammering iterations is around 156 ns. As shown in Fig. 3.5 (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. 3.5 (D) and (F) are slightly wider than that in Fig. 3.5 (E). This is because the timing variances when using `clflush` and eviction are larger than that when using `movnti`, as shown in Fig. 3.4.

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. 3.5 (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.

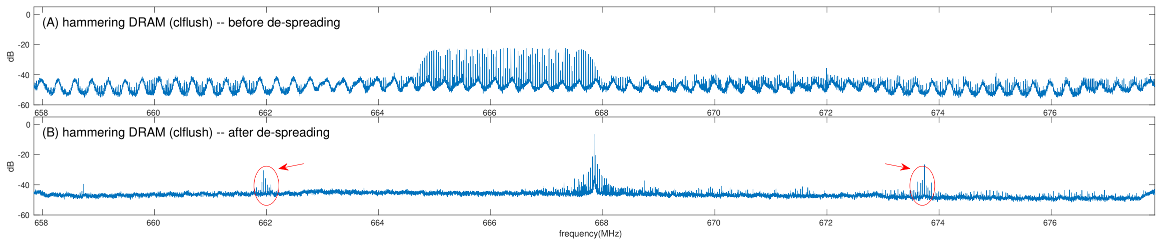


Figure 3.6: The power spectra under hammering by means of `clflush` before and after de-spreading. Each spectrum is derived by averaging 78 FFTs of 16,384 values with 50% overlap sampled in 25 MHz over 32 ms.

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 3.6.4, we will illustrate some of the experimental results related to this random delay addition.

3.4.3 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 Hardin et al. (1994).

Under the situation in which the underlying DRAM is being hammered, Fig. 3.6 (A) demonstrates the problem when SSC is 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. 3.6 (B) shows the power spectrum of the measured signal after de-spreading. Compared with Fig. 3.6 (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.

3.5 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. 3.7. In the following, we describe each component of our RADAR system.

3.5.1 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

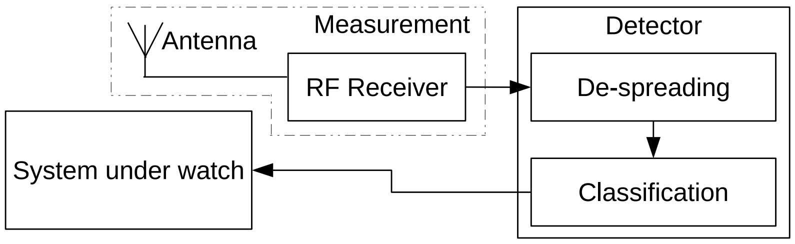


Figure 3.7: RADAR system illustration.

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.

3.5.2 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:

$$f_i(t) = f_c + Kf_m(t) \ , \tag{3.1}$$

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:

$$\theta(t) = e^{j2\pi \int_0^t Kf_m(t)dt} \ , \tag{3.2}$$

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 kth sample corresponding to the clock signal at a specific time τ, namely

$$v_k = |v_k| e^{j\phi_k} , \tag{3.3}$$

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

$$\frac{d\phi_k}{dt} = 2\pi K f_m(\tau) \tag{3.4}$$

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

$$\theta(\tau) = e^{j2\pi \int_0^\tau K f_m(t)dt} = e^{j(\phi_k + \Theta)} = e^{j\phi_k} e^{j\Theta} , \tag{3.5}$$

where Θ is a constant phase angle. Although we do not know the exact value of Θ, 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. 3.5 to have it rely on only the values acquired by quadrature sampling:

$$\theta(\tau) = \frac{v_k}{|v_k|} = e^{j\phi_k} \tag{3.6}$$

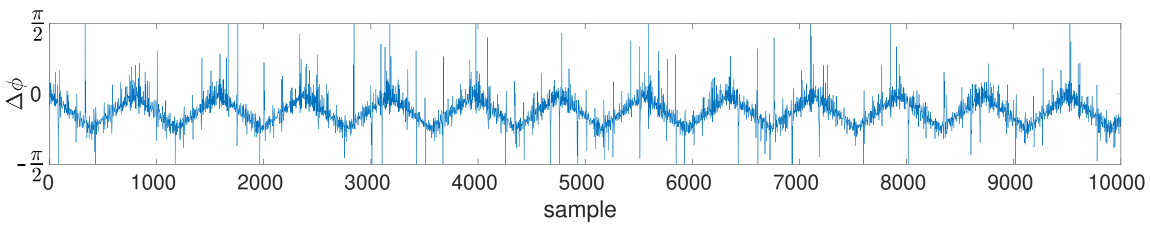


Figure 3.8: The phase difference $\phi_k - \phi_{k-1}$ between successive samples, where $1 \leq k \leq 10,000$. Many of the apparent spikes above 0 are actually caused by *phase wrapping*, i.e., any negative difference in the range of $-\pi > \Delta\phi \geq -2\pi$ will be converted to an angle in the 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 Hardin et al. (1994), 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[4], 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:

$$\phi_k - \phi_{k-1} \approx \phi_l - \phi_{l-1} \text{ , where } l = k + \lfloor T_m f_s \rfloor \tag{3.7}$$

For example, in terms of the platform used in Section 3.4, Fig. 3.8 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 Δ once for each hardware platform, as it is software-independent. When sampling the clock signal for Δ 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 Δ for de-spreading during detection, we first need to achieve Δ alignment, which is to find a point p in the stream $\mathscr{S}$ of the sampled values such that the phase of $\theta(t)$ varies by $\Delta[0]$ between $\mathscr{S}[p+1]$ and $\mathscr{S}[p]$, by $\Delta[1]$ between $\mathscr{S}[p+2]$ and $\mathscr{S}[p+1]$, and so forth. It is straightforward to see that Δ alignment is periodic, namely if $\mathscr{S}[p]$ aligns with Δ, $\mathscr{S}[p+kN]$ will also align with Δ. (Because it is very likely that $T_m f_s$ is not an integer, strictly speaking, Δ 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 Δ 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 Δ and the phase changes of the mth point relative to the first point. Therefore,

[4]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.

given a point q, we use the following equation to calculate the cross-correlation:

$$\begin{aligned}\rho(q) &= \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\frac{\mathscr{S}[q+m]}{|\mathscr{S}[q+m]|}}{\frac{\mathscr{S}[q]}{|\mathscr{S}[q]|}} e^{-j\sum\limits_{i=0}^{m-1}\Delta[i]} \right| \\ &= \left| \frac{1}{\frac{\mathscr{S}[q]}{|\mathscr{S}[q]|}} \right| \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\mathscr{S}[q+m]}{|\mathscr{S}[q+m]|} e^{-j\sum\limits_{i=0}^{m-1}\Delta[i]} \right| \\ &= \left| \frac{1}{N} \sum_{m=1}^{N} \frac{\mathscr{S}[q+m]}{|\mathscr{S}[q+m]|} e^{-j\sum\limits_{i=0}^{m-1}\Delta[i]} \right| \end{aligned} \tag{3.8}$$

Using Eq. 3.8, 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 Δ alignment in that round. (Again, if $T_m f_s$ is an integer, we will need at most N rounds to find Δ 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. 3.9 shows the cross-correlation results in the first 800 rounds with respect to the example given in Fig. 3.6, and we can clearly see that the initial Δ alignment is found in the 266th round.

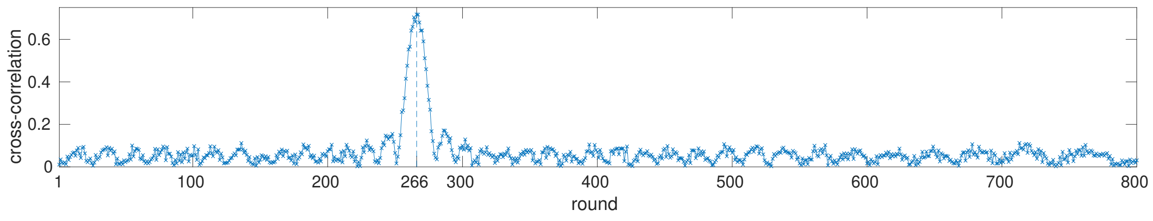


Figure 3.9: Using cross-correlation to achieve the initial Δ alignment.

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

$$\begin{aligned} \mathscr{D}[p+i] = \mathscr{S}[p+i]e^{-j\varphi_{p+i}}, \text{ where} \\ \varphi_{p+i} = \varphi_{p+i-1} + \Delta[(i-1) \bmod N], \text{ and } \varphi_p = 0 \end{aligned} \tag{3.9}$$

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 $\mathscr{S}[p+kN+1]$ aligns with Δ, instead of $\mathscr{S}[p+kN]$. 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 Δ if $\rho(p+kN+1)$ is larger, which means performing a right circular shift on Δ by one position, namely, we derive and use a new Δ as follows:

Interestingly, de-spreading will inadvertently help reduce background noise unrelated to the EM ema-

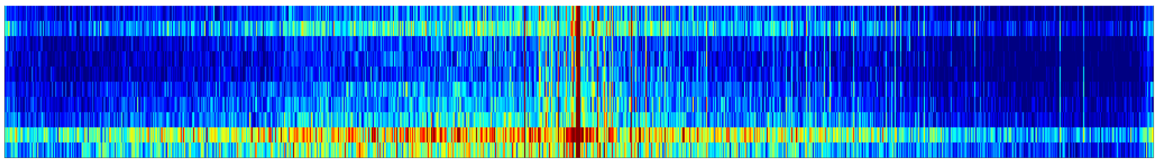
```
if $\rho(p+kN+1) > \rho(p+kN)$ then
    for $j = 0;\ j < N;\ j = j+1$ do
        $\Delta_{\text{new}}[(j+1) \bmod N] = \Delta[j]$;
    end
```

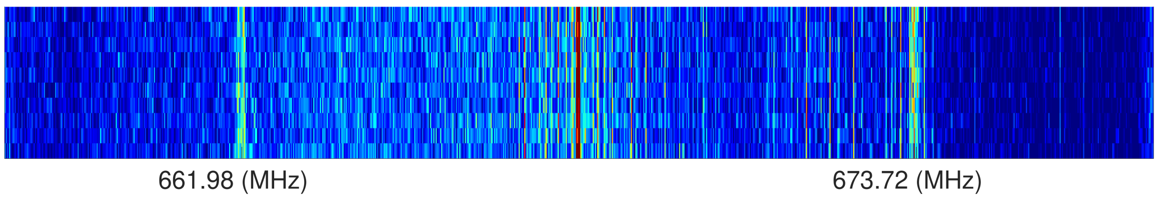
nations of interest. This effect is due to the fact that de-spreading will act like SSC on such noise, whose energy will be scattered over a range of frequencies. Because of this, the robustness of the proposed system is increased, as later shown in Section 3.6.3.

3.5.3 Classification Component

Having the stream of samples that are processed according to Eq. 3.9, 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.


(a) Playing a video



(b) Hammering the DRAM (`clflush`)

Figure 3.10: Spectrogram patterns of different activities.

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. 3.10 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.

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 3.6.1.

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 ± 0.05 MHz of the clock frequency to zero, namely, we zero out the central red stripes in Fig. 3.10. 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. 3.5), 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.

3.5.4 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.

3.6 Evaluation

We have implemented a RADAR prototype to demonstrate its practicality, and have evaluated it on four platforms that are summarized in Tab. 3.1. As stated in Section 3.2, 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 3.1: 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

3.6.1 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 Lime Microsystems (2021), 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. Fig. 3.11 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.

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-

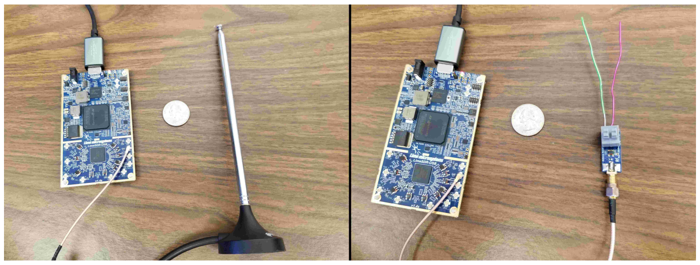


Figure 3.11: Two antennas that have been used in RADAR. In both figures, the used LimeSDR is also shown.

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[5].

We train a 3-layer CNN model using the positive and nega-tive examples collected from the four platforms listed in Tab. 3.1. 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 chapter, 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 Section 3.6.2, 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[6], we perform 8192-point FFTs that can provide about 3 KHz frequency resolution and spans only 327.68 μs. 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

[5]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.

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

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.

3.6.2 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)

Figure 3.12: Different hammering loop bodies.

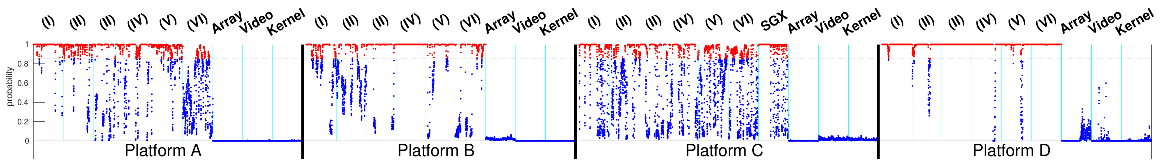


Figure 3.13: The detection results in the form of the probability of hammering.

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, 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. 3.12. 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* Gruss et al. (2018). 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 Tatar et al. (2018a); Pessl et al. (2016); Xiao et al. (2016), 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. 3.12, 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

Table 3.2: 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

a Linux kernel, which will generate a large amount of processor-memory-storage traffic. Fig. 3.13 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. 3.3. 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 Gruss et al. (2018), 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. 3.13. Thus, the proposed RADAR can effectively detect elusive rowhammer attacks.

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. 3.2. 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. 3.1). Likewise, both platforms D and F have DDR4-2133 modules, but their memory chips are also from different vendors.

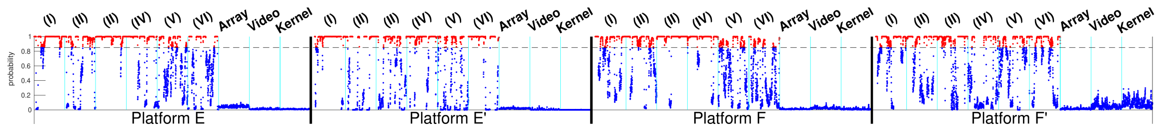


Figure 3.14: The detection results on additional platforms in the form of the probability of hammering. The CNN model for classification is the one trained in Section. 3.6 without any change.

Furthermore, we change the memory modules of E and F to form another two platforms E′ and F′. As listed in Tab. 3.2, 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. 3.12 on these four platforms.

The detection results are presented in Fig. 3.14. 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.

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*[7] Seaborn and Dullien (2015), 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*[8] Tatar et al. (2018a), which can achieve more effective double-sided hammering by considering the detailed information about end-to-end address translation; and, (3) Gruss's *flipfloyd*[9] Gruss et al. (2018), which has a tool for testing one-location hammering.

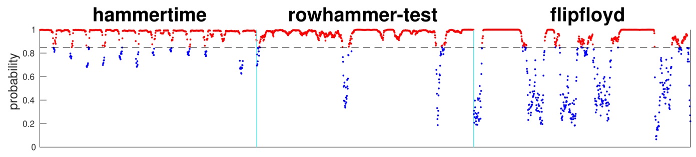


Figure 3.15: The detection results on platform A *w.r.t.* three well-known tools.

We run each tool as is, and Fig. 3.15 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**

[7]https://github.com/google/rowhammer-test
[8]https://github.com/vusec/hammertime
[9]https://github.com/IAIK/flipfloyd

only considering prevention of bit flips, the false negative 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 Aweke et al. (2016), `gcc` induces many false positives under ANVIL; yet, from Fig. 3.13, 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. 3.16, and the results for other SPEC benchmarks are similar to `bzip2` (integer) and `lbm` (floating-point). From Fig. 3.16, we can clearly see that **no false positives** arise. Note that no floating-point SPEC benchmarks are used in Aweke et al. (2016), but we argue that these benchmarks should be included for evaluation due to their pervasive manipulation on very large matrices.

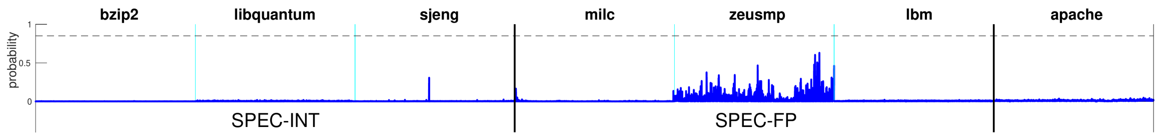


Figure 3.16: The detection results on platform `A` *w.r.t.* SPEC integer and floating-point benchmarks as well as Apache HTTP server.

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. 3.13, 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.

3.6.3 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. 3.3 for sideband patterns caused by `clflush`, `movnti`, and eviction. Note that Tab. 3.3 only shows the first one that is recognizable for each case.

Table 3.3: The relative power (measured in dB) of the hammering-correlated sideband patterns caused by `clflush`, `movnti`, and eviction respectively.

Scenario \ Platform	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. 3.3 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 3.4: The impacts of external noise on RADAR for platform A.

Scenario \ Distance	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. 3.4. This is because, as mentioned in Section 3.5.2, de-spreading will inadvertently help reduce such noise. Recall that, for de-spreading, the hardware-dependent Δ is aligned and used to modulate the measured signal, and if the measured signal has components unrelated to the used Δ, the energy of these components will be spread. Since the Δ of `A` is different from that of platform `B`, when using the Δ of `A` for de-spreading, the EM-emanated DRAM clock signals of `B` are unrelated to the used Δ, and their energy is scattered to become negligible noise.

A more interesting scenario arises from using identical motherboards, as they have the same Δ. 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 Δ 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 Δ 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 Δ 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 can generally manage to place the telescopic antenna inside the mini tower (or bigger) cases. By contrast, we use a very simple self-built antenna, which consists of two pieces of 7.5 cm metal wire. This antenna can be easily placed inside the computer case of **any size** (e.g., small form factor or server chassis).

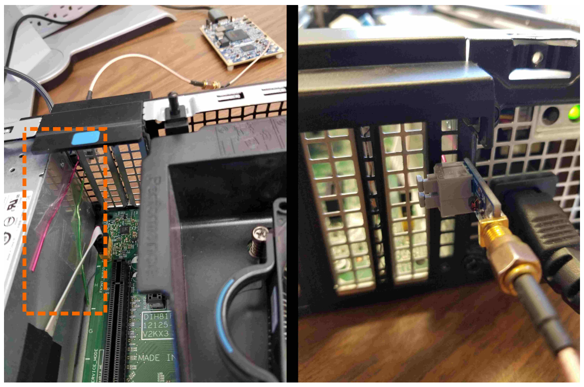


Figure 3.17: Placing the self-built antenna inside the metal case of a SFF computer.

For example, Fig. 3.17 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. 3.17 (denoted by the dashed line). We simply leave the antenna balun outside the case, as shown in the right part of Fig. 3.17. 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. 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. 3.4. 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.

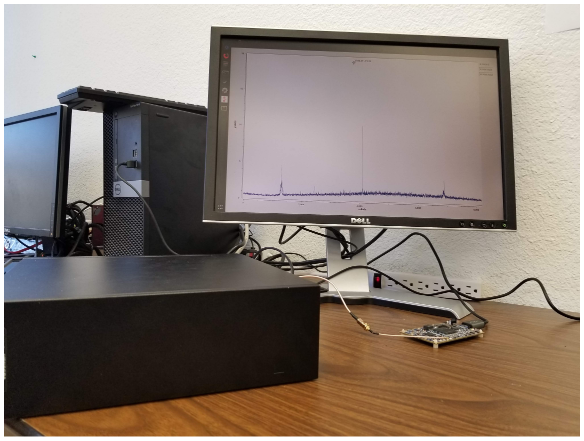


Figure 3.18: Apparent hammering-correlated sideband patterns.

Given the aforementioned antenna placement, we execute a program for hammering. As we can observe from Fig. 3.18, the hammering-correlated sideband patterns are extremely clear. Again, the SSC is always on and the spectrum is shown after de-spreading.

3.6.4 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. 3.19. The outer loop in Fig. 3.19 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.

Fig. 3.20 shows the DRAM clock spectra of platform `D` under different M values. (Note that the ex-

```
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
```

Figure 3.19: Add random delay to each iteration to disturb the hammering period.

periments 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.

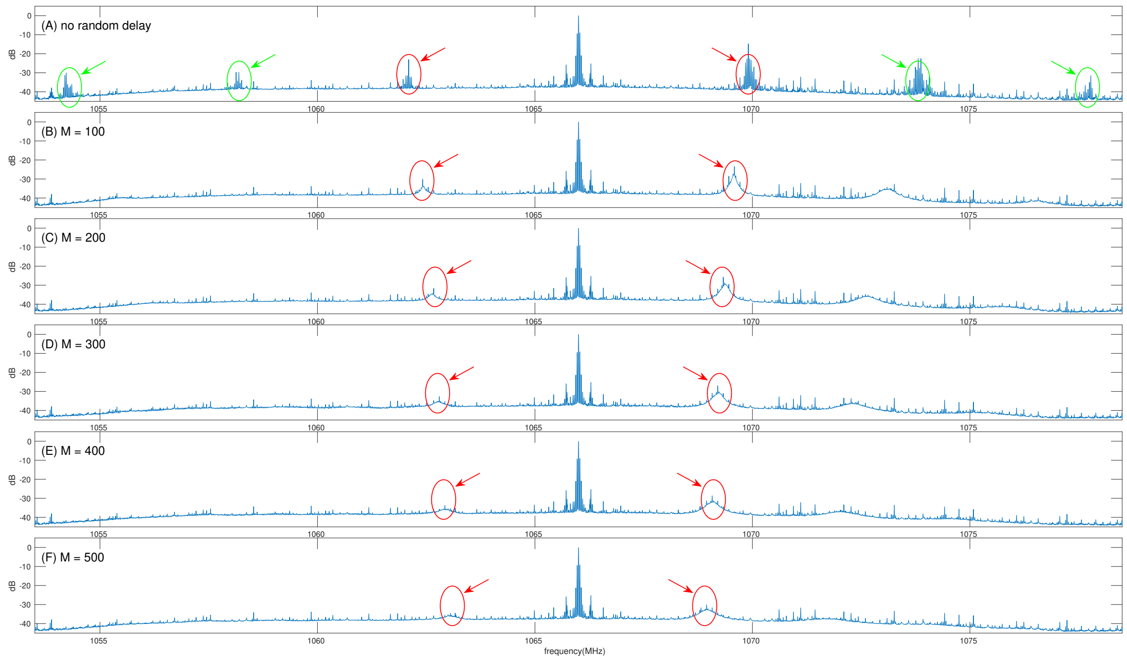


Figure 3.20: The power spectra under different M. In a hammering attempt, *each* hammering iteration will be delayed by a loop whose bound is randomly chosen in the range of 1 to M. The larger M is, the more disturbance is added into the hammering period.

From Fig. 3.20 (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. 3.21 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.)

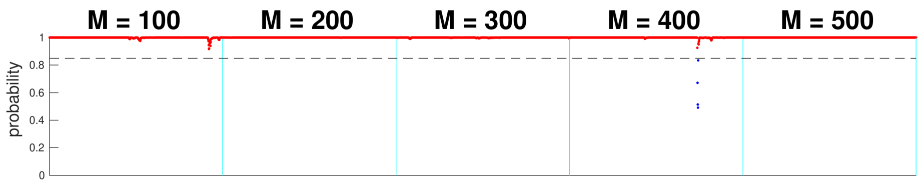


Figure 3.21: The detection results on platform `D` *w.r.t.* different M values.

Compared to the (I) results on platform `D` in Fig. 3.13, some of the results in Fig. 3.21 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.

3.7 Conclusion

In this chapter, 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[10].

[10]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 chapter) 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.

CHAPTER 4

The Fastest EM Covert Channel In The World

In organizations where information security and privacy are top priorities, physical isolation is often used to prevent data exfiltration. Air-gapping is considered as one of the strongest physical isolation method that has been widely used by, e.g., militaries and governments. An air-gapped computer has no connections with the outside unsecured networks, so that it is believed that protection against unauthorized data transfer can be effectively guaranteed.

However, recent research has discovered that many physical side effects of computation on air-gapped computers can be exploited to construct so-called physical covert channels to re-enable data exfiltration. The physical side effects that can be exploited are various, including thermal Guri et al. (2015b), optical Loughry and Umphress (2002); Sepetnitsky et al. (2014); Guri et al. (2017b); Lopes and Aranha (2017), magnetic Matyunin et al. (2016); Guri et al. (2018a,c), acoustic Hanspach and Goetz (2013); Carrara and Adams (2014); Guri et al. (2020, 2017a), or electromagnetic (EM) Guri et al. (2014); Zajic and Prvulovic (2014); Guri et al. (2016b, 2015a). The communication distance of such covert channels is usually very short, ranging from several centimeters to several meters, due to the high attenuation of the exploited physical effects in the distance. Information is encoded within the physical effects and transferred over the air gaps between a sender and a receiver. Normally, a sender is a piece of malware, like a Trojan horse, that has been stealthily inserted into a victim's computer, and a receiver is some device in the proximity of the sender that can capture the exploited physical effects.

Nevertheless, the security risks of such covert channels are often neglected, as they are considered hardly posing any real hazards for two reasons. First, the bandwidth of such physical covert channels is usually very low. For example, the transmission rate of the covert channel proposed in Guri et al. (2015b) is only 8 bits/hour (i.e., 0.002 bps). Even the fastest one reported in Guri et al. (2017b) can only reach 4,000 bps. Therefore, if a large amount of data needs to be exfiltrated, an attacker has to maintain the covertly communicating status for a long period of time. In a situation where the attacker can briefly have her foothold in the proximity to the targeted computer, any lingering action may cause suspicion. Second, most of these covert channels require no physical obstacles between the sender and receiver. Thus, an attacker may encounter great difficulties in managing the placement of the receiving device. In particular, locking an air-gapped computer in an enclosed room has been regarded as a sufficiently secure protection against such physical covert channels.

In this chapter, we demonstrate that there in effect exist powerful covert channels that are extremely fast and strong enough to penetrate even thick walls[1]. Specifically, we construct such a covert channel named *BitJabber* from the EM signals generated by the DRAM clock. As discovered in Callan et al. (2015), there are strong EM signals generated by different clocks in a computer that can propagate far, and these EM signals can be amplitude-modulated (AM) by activities driven by the corresponding clocks. Therefore, the EM signals generated by the DRAM clock can be AM-modulated by normal memory accesses to carry and transfer information over the air gaps between a pair of sender and receiver, namely forming an electromagnetic covert channel. Our experimental results show that the transmission rate of this new covert channel can reach 100,000 bps using binary frequency-shift keying modulation (B-FSK) with error rate around 0.05%, and 300,000 bps using multiple frequency-shift keying modulation (M-FSK) with error rate less than 0.1%. Moreover, this covert channel is resilient to a reasonable level of background noise and works well even in the presence of 15 cm thick walls between the sender and the receiver.

The main contributions of this chapter are three-fold:

- We present a new physical covert channel named *BitJabber* that can allow expedited data exfiltration between air-gapped sender and receiver.
- We verify that our *BitJabber* covert channel is much more resilient to background noise compared with the state-of-the-art ones.
- We demonstrate that this new covert channel can achieve reliable communication within a few meters, even under the scenario where the sender and the receiver are in separate rooms with walls in-between.

The rest of this chapter is organized as following: Section 4.1 states the threat model considered in this chapter. Section 4.2 presents our *BitJabber* covert channel in detail, including the techniques for modulation, demodulation and synchronization. Section 4.3 evaluates the performance of *BitJabber*. Section 4.4 lists some possible countermeasures against this new covert channel and Section 4.5 concludes this chapter.

4.1 Threat Model

Similar to the previous work Matyunin et al. (2016); Guri et al. (2018a,c,b, 2015b); Hanspach and Goetz (2013); Carrara and Adams (2014); Guri et al. (2016a, 2017a); Loughry and Umphress (2002); Sepetnitsky et al. (2014); Guri et al. (2017b); Lopes and Aranha (2017); Guri et al. (2014, 2016b), in this chapter, we explore how to construct a covert communication channel between a pair of air-gapped sender and receiver. We assume that the sender has been placed on the victim computer that stores or processes the secret data of interest, and the sender can acquire the secret through techniques like microarchitectural side-channels Ge et al. (2018). (How to place the sender there is out of scope, but, as presumed in the previous work, the

[1]The work in this chapter has been previously published in Zhan et al. (2020) and reported in Zhan et al. (2021a)

attacker is capable of achieving this by methods like social engineering, USB interface, or physical access.) Note that we do not assume the sender has any privilege higher than the regular user level.

We assume that the attacker can use a radio frequency (RF) receiver (like a cheap software-defined radio) to collect the EM signals emanated from the victim machine somewhere nearby. Note that we do not require the receiving device to share the same room with the sender or to be physically adjacent to the sender. The sender and the receiver may be in different rooms with concrete walls, and the straight-line distance between them can be one or two meters.

4.2 The Design of *BitJabber* Covert Channel

As mentioned above, our *BitJabber* is an EM-based covert channel. The carrier EM signal is generated by the DRAM clock, and memory accesses are used to modulate the carrier signal to encode information. When modulated carrier signal is captured, demodulation is used to decode information from that signal. The overview of our *BitJabber* covert channel is illustrated in Fig. 4.1. In the following, we will describe the main components and techniques used in *BitJabber*.

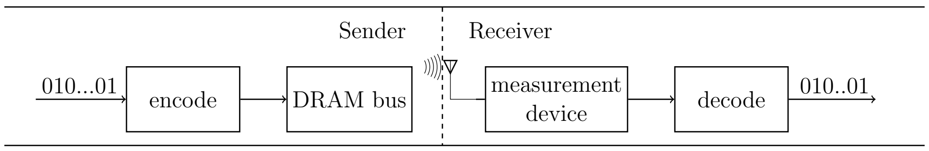


Figure 4.1: Overview of *BitJabber* cover channel.

4.2.1 Spread Spectrum Clocking

As aforementioned in 3, SSC has been widely used in electronic products like computers for meeting electromagnetic compatibility (EMC) regulations Departments and agencies of the Federal Government (2019). Due to SSC, the energy of the EM signals generated by the DRAM clock will be spread over a wide range of frequencies. Such an energy dispersion makes the exploitation of these EM signals much harder, because the power of the exploitable signals becomes weaker but the power of the background noise stays the same. As a result, the signal-to-noise ratio (SNR) is much decreased, and thus our covert channel capacity will be considerably affected. To increase the SNR, we need to use a de-spreading technique to gather the scattered signal energy back, which has been described in Section 3.5.2 in detail. De-spreading can significantly improve the capacity of our covert channel in several ways. First, de-spreading gathers the scattered energy of the exploitable EM signals (i.e., it helps strengthen the signal), while de-spreading also inadvertently acts like SSC on background noise (i.e., it helps weaken the noise). Thus, the SNR will be greatly increased. Second, the EM signals of interest will be located in a narrow frequency range after de-spreading, which allows us to use more advanced modulation techniques to utilize the spectra.

4.2.2 Modulation

To encode information into the EM signals generated by the DRAM clock, modulation is required to vary the EM wave with respect to the message contents. As it is known that the EM radiation of the DRAM clock is AM-modulated by memory accesses, the modulation for *BitJabber* covert channel is accomplished through manipulating the memory access behavior.

To understand how memory access behaviors affect the EM signals generated by the DRAM clock, we perform different memory activities on a computer equipped with DDR3-1600 memory modules (i.e., the DRAM clock frequency is 800MHz) and investigate the corresponding spectra, which are shown in Fig. 4.2. At first, no intense memory accesses are performed. As illustrated in Fig. 4.2, the EM radiation after de-spreading has most of its energy concentrated near the clock frequency (i.e., 800MHz). When memory accesses with execution time around 350ns are repeatedly performed, raised energy can be observed at certain frequencies in the lower and upper sidebands. The offsets of these lobes from 800MHz are multiples of the memory access frequencies (i.e., 2.86MHz), which indicates that the EM radiation is AM-modulated by a non-sinusoidal wave with the same frequency as the memory accesses. If some delay is added to make the memory accesses slower, the positions where the lobes locate indicate that the frequency of the modulating non-sinusoidal wave also decreases. (Note that we use non-temporal load/store instructions like `MOVNTI` to avoid memory accesses being served directly from the CPU caches.)

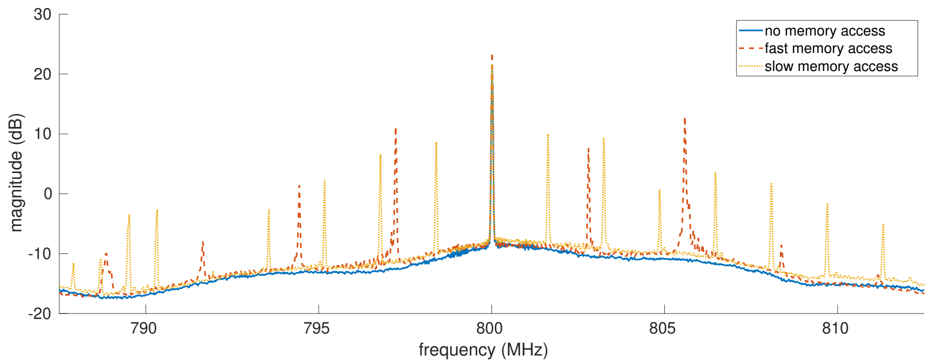
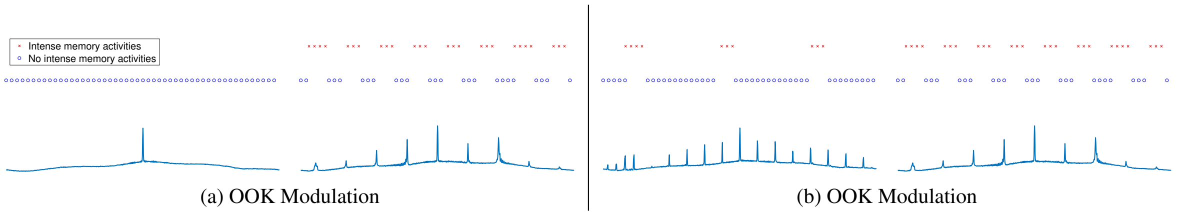


Figure 4.2: Spectra of different memory access behaviors

The above observation shows that not only do intense memory accesses introduce obvious lobes in the sidebands, but also the memory access frequency has influence on where these lobes locate. Accordingly, two modulation techniques can be applied to encode information into the EM signals generated by the DRAM clock:

- The first and also the simplest modulation method is OOK. As shown in Fig. 4.3 (a), OOK uses the



(a) OOK Modulation (b) OOK Modulation

Figure 4.3: Encoding of 0 and 1 using two modulation methods – OOK and FSK

presence and absence of repeated memory accesses to encode bit ‘1’ and bit ‘0’. Consequently, the AM-modulated EM signal will have side lobes in its spectrum only when ‘1’ is transmitted; otherwise, ‘0’ is sent.

- The other modulation method is FSK, indicated by Fig. 4.3 (b), where different symbols are represented by different memory access frequencies. For example, to send bit ‘1’, fast memory accesses are repeated, and to send bit ‘0’, slow accesses are repeatedly made. Thus, different distances between the side lobes and the clock frequency in the spectra can distinguish these two cases. To realize different memory access frequencies, we can use a normal memory access as the fast one and introduce some delay to derive the slow one.

Note that the above-mentioned FSK modulation is not limited to B-FSK, in which case either bit ‘0’ or ‘1’ is transmitted. Because any two different memory access frequencies can result in distinguishable side lobe positions in the spectra, M-FSK modulation is also achievable by adding distinct delays to a base memory access activity `BaseMemAcc` as depicted in Algorithm 1. (The details of the `BaseMemAcc` activity will be described later.)

```
Input: $T_i$ = delay time for transmitting symbol $i$
if Transmitting symbol $i$ then
    BaseMemAcc;
    DELAY($T_i$);
end
```

Algorithm 1: Memory activities for M-FSK modulation

4.2.3 Base Memory Access Design

We have observed that randomly accessing some memory addresses may not AM-modulate the EM signals generated by the DRAM clock well. Thus, we need to have a systematic way to construct a memory access activity such that the probability of AM-modulating the EM signals of interest well is very high. We term this systematically-constructed memory access activity as base memory access `BaseMemAcc`.

We need `BaseMemAcc` to have the following three properties:

1. It should have a very short execution time (e.g., a few hundreds of nanoseconds).

2. It should have a relatively stable execution time.

3. It should induce obvious change in the amplitude of the EM signals generated by the DRAM clock.

To design such a base memory access activity, we need to understand how memory accesses affect the DRAM clock. Although it has been investigated in some prior work Callan et al. (2015), factors that influence the AM-modulation effect were not fully identified.

To satisfy the first two properties, we decide to use non-temporal memory access instructions, such as `MOVNTI`, `MOVNTDQ` and `VMOVNTDQ`. Since they will bypass the CPU caches, we can use them to directly access the main memory in a rapid manner. Otherwise, `CLFLUSH` instruction needs to be used to flush the cache after each memory access, which brings in more overhead and execution time variation. These non-temporal memory access instructions can support operands of different sizes, e.g., either 32-bit or 64-bit operands can be used in `MOVNTI`. The operand size can affect the execution time slightly, but can result in observable differences in side lobe positions in the spectrum.

We notice that memory locations may have a significant influence on the AM-modulation. In order to find out how the AM-modulation effect is related to the memory access instructions and memory locations, we conduct experiments and empirically conclude the following:

1. When the same memory access instruction is used to access the same memory location, the AM-modulation effect (e.g., side lobe positions and their energy) is fixed.

2. When different types of non-temporal instructions are used to access the same memory location, the AM-modulation effect is slightly different.

3. When the same instruction is used to access different memory locations, the amount of amplitude change of the EM signals of interest may be significantly different. The relationship between accessed address and the amount of amplitude change is still not clear, but in our tested platforms we notice that accessing memory addresses in the same DRAM bank tends to change the amplitude similarly.

Therefore, the more memory locations are accessed, the higher the possibility that obvious amplitude change will arise is. Based on the above observations, to satisfy the third property, `BaseMemAcc` needs to access several fixed memory locations using the same non-temporal memory access instruction. Apparently, there is a trade-off, because the more memory locations are accessed, the slower `BaseMemAcc` will become. We empirically find that accessing 4 memory locations is sufficient to have obvious AM-modulation effect while keeping the execution time short.

Note that if these fixed memory locations are randomly selected, it may incur unpredictable variations in the execution time due to row buffer conflicts in the same DRAM banks Hassan et al. (2015); Pessl et al. (2016). Such variations may make the second required property of `BaseMemAcc` violated. Therefore, it is preferable to have these memory locations in different DRAM banks. Moreover, considering that in some platforms the amplitude change is bank-dependent, this memory location selection strategy can even help `BaseMemAcc` hold the third property. Thus, we design `BaseMemAcc` to be a memory access activity that uses a fixed non-temporal memory access instruction to access 4 fixed memory locations in different DRAM banks.

```
Input  : $AP$ = address pool
Output: $G$ = addresses mapped to the same bank
RefAddr $\leftarrow$ $AP$.DEQUEUE();
$G$.ENQUEUE(RefAddr);
$n$ $\leftarrow$SIZEOF($AP$);
for $i \leftarrow 1...n$ do
    RemAddr $\leftarrow$ $AP$.DEQUEUE();
    if LATENCY(RemAddr,RefAddr) then
        $G$.ENQUEUE(RemAddr)
    else
        $AP$.ENQUEUE(RemAddr)
    end
end
```

Algorithm 2: Grouping virtual addresses *w.r.t.* banks

However, finding memory locations belonging to different DRAM banks can be a problem, because the address mapping information is unavailable to unprivileged attackers. To obtain such memory locations, we use a method exploiting a timing side-channel introduced by the row buffer conflicts in the same DRAM banks Hassan et al. (2015); Pessl et al. (2016). Given two virtual addresses a_1, a_2, a function `LATENCY`(a_1, a_2) is used to check whether they are in the same bank. If they are in the same bank, accessing them consecutively is relatively slow due to the delay induced by the row buffer conflict, and `LATENCY`(a_1, a_2) returns `True`; otherwise, accessing them is faster and `LATENCY`(a_1, a_2) returns `False`. The memory location selection method is described in Algorithm 2. By repeating this method, we can derive several groups, in each of which the addresses are located in the same DRAM bank.

4.2.4 Communication Protocol and Demodulation

As indicated in Section 4.2.2, modulated signals with varied energy distribution on frequency domain are transmitted to send symbols of different values. On the receiver's side, after the EM signals are captured, demodulation is a necessary step to recover the encoded information from the modulated signals. In order to demodulate the received signals correctly, three problems need to be tackled:

1. How can we extract the features to distinguish different transmitted signals?
2. How is the receiver synchronized with the sender?
3. How can the receiver map the extracted features to correct symbol values?

In this section, we will describe the feature extraction method and communication protocol implemented to handle these problems.

4.2.4.1 Feature Extraction

For our *BitJabber* covert channel, the key problem of demodulation is to classify different symbol values according to the signal's energy distribution in the frequency domain. As shown in Fig. 4.2, when memory accesses are performed at a fixed frequency to transmit a symbol value corresponding to that frequency, side lobes appear at the first few harmonics of that frequency. Instinctively, features corresponding to these side lobes should be extracted.

To better describe the feature extraction process, we will use an example in which B-bit FSK modulation is employed. In this case, S possible symbol values may be sent, where $S = 2^B$. We assume the clock frequency is f_c, memory access frequencies f_0, f_1, ..., f_{S-1}, are used for encoding S different symbol values. M symbols are transmitted with a known symbol rate R_{symbol} and the EM signal of interest is sampled with a known sampling rate R_{sample}. The steps of feature extraction are as follows:

1. Find all the frequencies where side lobes locate in the spectrum of the captured EM signal (which is a sequence of samsample size ofpled values). In our example, for each symbol value $s \in \{0, ..., S-1\}$, let us assume there are $2K_s$ lobes at $f_c \pm k_s f_s$ where $1 \leq k_s \leq K_s$.
2. For each frequency where a side lobe locates, apply a bandpass filter on the original signals and extract the envelope of filtered signals to preserve only the energy of that frequency. For our example, we can obtain K filtered signals, where $K = \sum_{s=0}^{S-1} 2K_s$. Hence, the captured signals are converted to a sequence of K-dimension vectors $\mathbf{v}$.
3. Segment the vector series $\mathbf{v}$ using the boundary finding technique described in Section 4.2.4.3. The length of each segment L is:
$$L = \frac{R_{\text{sample}}}{R_{\text{symbol}}} \tag{4.1}$$
(Note that we choose R_{sample} divisible by R_{symbol} by design, so L is an integer.)
4. Average all the values within each segment. Assume the segment head found for symbol m is sample

n, the correspoinding feature vector $\mathbf{V}_m$ is computed using:

$$\mathbf{V}_m = \frac{1}{L}\sum_{l=0}^{L-1}\mathbf{v}_{n+l} \tag{4.2}$$

After this step, M K-dimension feature vectors are derived for all symbols.

4.2.4.2 Message Structure

To implement our *BitJabber*, we structure the message Q as shown in Fig. 4.4. It consists of a header $\{Q_0^H, ..., Q_{M_h-1}^H\}$ and its payload$\{Q_0^P, ..., Q_{M_p-1}^P\}$. The header is a pseudo random number sequence whose seed is shared by the sender and receiver, which is used for signal synchronization and deriving symbol mapping.

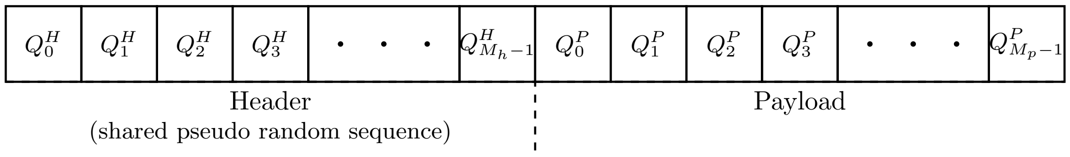


Figure 4.4: Message structure

4.2.4.3 Finding Segment Boundaries

Successful synchronization is the prerequisite of demodulation, which guarantees that the feature vectors in Eq. 4.2 is computed at the right position, i.e., the correct pair of (m,n) are found. Assume in a symbol sequence, we know segment for symbol m_0 starts at sample n_0, the segment for the next symbol m_0+1 will start at sample n_0+L. Because the sender and receiver are driven by different clocks, there exists inevitable clock drift δ. Although in reality δ is very small (e.g., around 0.0001%), the accumulated error can reach a level such that a compensation in the symbol length is needed. Therefore, the symbol m will actually start at sample n expressed as:

$$n = n_0 + (m-m_0)\times L + \lfloor (m-m_0)\times\delta\times L\rfloor \tag{4.3}$$

If we make $m_0 = 0$, symbol m starts at:

$$n = \lfloor m\times L\times(1+\delta)\rfloor + n_0 \tag{4.4}$$

Thus, finding segment boundaries means finding the values of δ and n_0, which can be accomplished through performing linear fit on some known pairs of (m,n). Such pairs can be found within the header with it being shared knowledge between the sender and receiver.

To find the correct segment head n_r for a symbol m_r in header, we use the following steps:

1. With a guessed segment head $n_r = n'$, we can obtain a feature vector $\mathbf{V}^H_{m_r}(n')$ using Eq. 4.2.

2. Find an integer Δm such that $2\Delta m + 1$ is a large enough sample size for psenterforming statistical analysis while satisfying $\Delta m \delta \ll 1$. In reality, Δm is usually a number ranging from 100 to 1000.

3. With a guessed segment head n' and a properly chosen Δm, segment heads for symbols $\{m_r - \Delta m, ..., m_r + \Delta m\}$ can be estimated as $\{n' - L\Delta m, ..., n' + L\Delta m\}$. Subsequently, a sequence of feature vectors $\{\mathbf{V}^H_{m_r - \Delta m}(n'), ..., \mathbf{V}^H_{m_r + \Delta m}(n')\}$ can be obtained.

4. For each possible symbol value s, these feature vectors can be split into two groups according to the value of Q^H_m, which gives us $\mathbf{V}^H_s(n') = \{\mathbf{V}^H_m(n') | Q^H_m = s\}$ and $\mathbf{V}^H_{\sim s}(n') = \{\mathbf{V}^H_m(n') | Q^H_m \neq s\}$.

5. With this splitting, we take advantage of the fact that *with the correct segmentation, feature values in each dimension of the feature vectors will have the minimum standard deviations within the same group and the maximum differences between different groups*. Intuitively, to measure the segmentation quality for each symbol value s w.r.t each feature dimension k, we can define a score T as follows:

$$T(n', s, k) = \ln\left(\frac{\sum_{m_i, m_j}\left(V^H_{s,k,m_i}(n') - V^H_{\sim s,k,m_j}(n')\right)^2}{|V^H_{s,k}(n')| \times |V^H_{\sim s,k}(n')| \times \sigma\left[V^H_{s,k}(n')\right] \times \sigma\left[V^H_{\sim s,k}(n')\right]}\right) \tag{4.5}$$

   Then the actual n_r can be found through searching for the maximum score.

$$n_r = \arg\max_{n'} \sum_{s,k} T(n', s, k) \tag{4.6}$$

After enough pairs of symbol and segment heads are identified in the header, we can fit Eq. 4.4 to obtain δ and n_0 for the header. With this knowledge, segment heads n of a symbol m in the payload can be computed using:

$$n = \lfloor (m + M_h) \times L \times (1 + \delta) \rfloor + n_0 \tag{4.7}$$

4.2.4.4 Payload Decoding

After successful synchronization, we can correctly compute the feature vectors for all symbol transmitted in the message. The last step of demodulation is mapping these feature vectors to the correct symbol values. As the header is a symbol sequence shared by the sender and receiver, the feature vectors obtained from the header can be used to train a simple classifier. This classifier is then used to translate the feature vectors in payload part to symbol values. Because the feature vectors to be recognized are simple, any classification technique can be used. In our case, we find the performance of SVM (support vector machine) to be satisfactory as demonstrated in Section 4.3.

4.3 Experimental Results

In this section, we will evaluate the performance of our *BitJabber* covert channel in terms of its bandwidth, error rate, and capability of wall-penetrating. In the evaluations, we also compare our *BitJabber* with the existing *GSMem* covert channel for the following two reasons:

1. The performance of covert channels depends on many factors like background noise and the physical architecture of the sender machine.
2. Both *BitJabber* and *GSMem* covert channels use the EM emanations generated from the DRAM clock.

4.3.1 Experimental Setup

The performance of *BitJabber* and *GSMem* covert channels are evaluated on three different platforms listed in Tab. 4.1. These platforms use different motherboards and DRAMs of multiple frequencies. On all platforms, two DIMMs are installed on two DRAM channels.

Table 4.1: Platforms on which our covert channel is evaluated.

Platform	Motherboard	Memory	Case Material
A	Dell Optiplex 990	$2 \times$ 4GB DDR3-1333	Metal
B	Dell Optiplex 3020	$2 \times$ 4GB DDR3-1600	Metal
C	Asus PRIME Z270-P	$2 \times$ 8GB DDR4-2400	Metal & Plastic

The receiver uses a log-periodic (LP) antenna, a telescope antenna and a software-defined radio (SDR) platform LimeSDR-USB development board to collect the EM signals around the DRAM clock frequency as shown in the left part of Fig. 4.5. The EM signals are preprocessed using the GNU Radio.

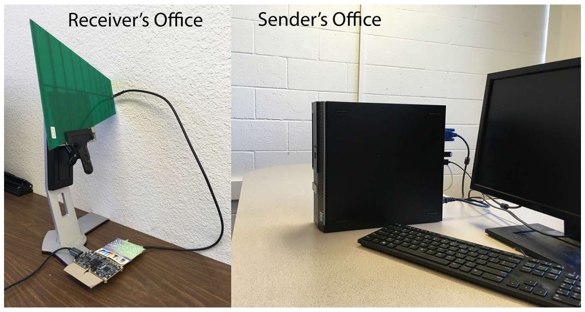


Figure 4.5: Experimental setup for wall-penetrating performance evaluation

The experiments are performed in a typical office environment. In such an environment, much background noise exists, including EM waves radiated from wireless communication systems (e.g., radio stations and cell towers), nearby electronic devices, and other components in the victim computers.

The experiments are performed in three different scenarios. First, the antenna is put close to the victim machine to receive the strongest EM emanations from the DRAM clock. This experiment will show the performance upper bound of different approaches. The second scenario is to experiment with a more practical setting, as shown in Fig. 4.5, where the sender and receiver are located in two different offices sharing a 15cm thick wall. This experiment compares the wall-penetrating data exfiltration capability of the covert channels. Additionally, we evaluate the performance of *BitJabber* with the antenna and victim machine separated by different distances, with no obstacles in between. This experiment evaluates *BitJabber*'s long-distance data exfiltration ability.

Our paper focuses on high-speed data exfiltration only, so the lowest evaluated symbol rate is 1,000 Bd. Nevertheless, in all evaluations, we use a 20 Bd symbol rate sequence to speed up the process of locating the beginning of signals (which is optional for the implementation). These 20 Bd patterns are very visually perceptible in all evaluations, i.e., all measurements in this section have zero error rates at 20 Bd symbol rate.

4.3.2 Symbol Distinguishability

For all covert channels exploiting physical side channel effects, the receiver measures certain physical changes introduced by senders and transforms the measurements into different symbols. A good covert channel should have good symbol distinguishabilities. In Fig. 4.6, we compare the symbol distinguishabilities of two covert channels *GSMem* and *BitJabber* using the B-FSK modulation.

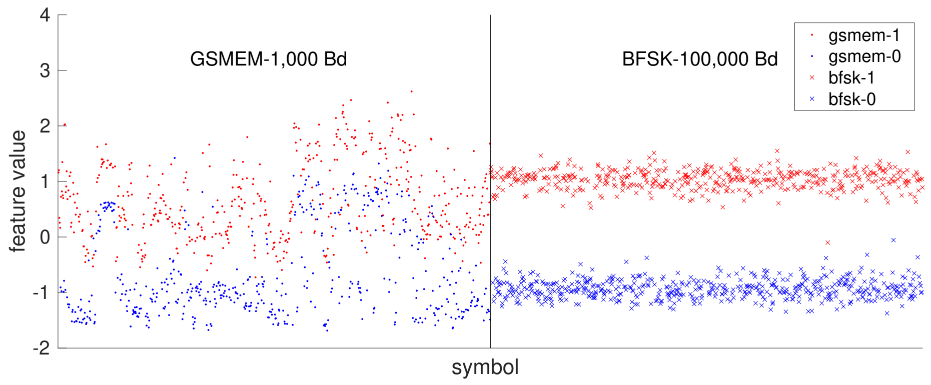
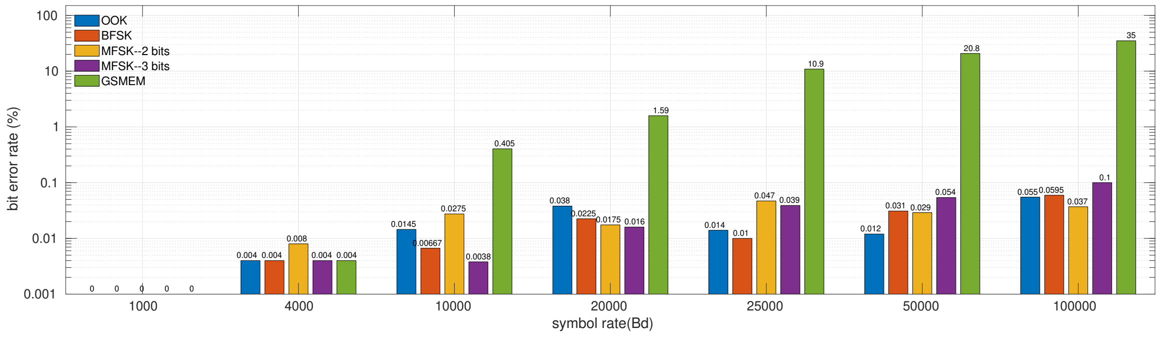


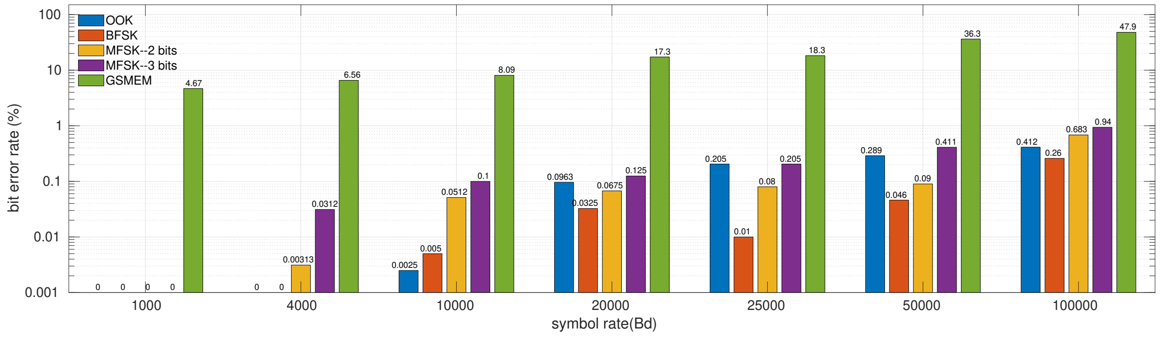
Figure 4.6: Symbol distinguishability of *GSMem* and *BitJabber* using the B-FSK modulation.

For transmitting binary symbols, we can use a single feature value to represent how likely a measurement is identified to a certain symbol (either '0' or '1'). In *GSMem*, only the magnitude of the EM signal is used for distinguishing symbols with binary values, and thus we can use this as the feature value. For *BitJabber* using B-FSK modulation, an SVM model is trained to distinguish the feature vectors, and thus we use the difference of two prediction scores as the feature value. The feature values of *GSMem* at 1,000Bd symbol rate and *BitJabber* using the B-FSK modulation at 100,000Bd symbol rate are illustrated in Fig. 4.6. Compared to *GSMem*, it is apparent that the measurements of *BitJabber* have much larger difference between different symbol values and smaller variances between same symbol values even if the symbol rate is 100 times higher. This comparison indicates that our *BitJabber* can greatly outperform the *GSMem*, which is demonstrated by the following experimental results.

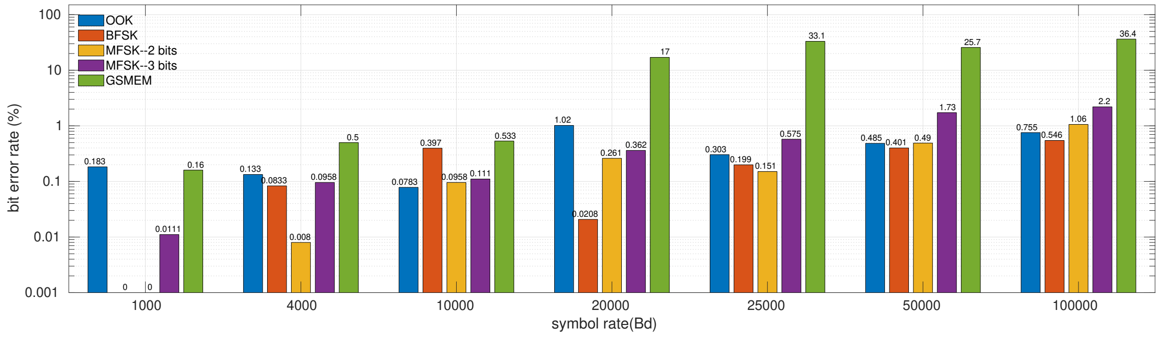
4.3.3 Bandwidth Evaluation



(a) Platform A



(b) Platform B



(c) Platform C

Figure 4.7: Bit error rate at different symbol rate for *GSMem* and *BitJabber* using different modulation methods.

The first group of experiments measure the maximum bandwidth of *GSMem* and our *BitJabber*. To measure the performance upper bound, all measurements are performed with the antenna set at a fixed position, at which the strongest EM emanations from the DRAM clock can be collected. The EM signals are modulated by the OOK, B-FSK, and M-FSK modulation methods. Examined symbol rates range from 1,000 Bd to 100,000 Bd and the evaluation results are shown in Fig. 4.7. Because of the huge performance difference between *GSMem* and our *BitJabber*, **logarithmic scale** is used in this plot. Note that the original *GSMem* uses the EM signals at only 800MHz. To make a fair comparison, here we evaluate *GSMem* using the EM signals at the frequencies of DRAM clocks, where memory behaviors cause the maximum amplitude changes.

In our evaluations, we limit the maximum symbol rate to 100,000 Bd and the maximum symbol length for M-FSK to 3 bits. Theoretically, larger symbol rate and symbol length can be used for this covert channel. Nevertheless, selection of these two parameters highly depends on the hardware device used to implement this covert channel. The `BaseMemAcc` used in victim computers typically takes several hundred nanoseconds to execute. If symbol rate higher than 100,000 Baud is used, the actual symbol duration tends to be more unstable, which will greatly increase the error rate. As for the symbol length, when 3 bits are represented by a single symbol, 8 different memory access frequencies are used and the resulting EM emanations almost affect the entire 25MHz frequency range. If more bits are transmitted, frequency ranges affected during transmission of different symbol values may overlap too much and variance between different symbol values' feature vectors tends to be smaller, which will also increase the error rate. With more advanced SDR devices used to exfiltrate data from more powerful computers, *BitJabber* may be implemented at larger symbol rates and symbol lengths.

The evaluations on different platforms have slight differences but in general for different covert channels and modulation methods we can summarize that:

- For all evaluated approaches, the error rates increase as symbol rates get higher.
- The error rate reported in at the symbol rate 1,000 Bd is 0.087%. Our evaluations indicate that the performance of *GSMem* is highly dependent on the platforms where it is implemented and the error rates range from 0 to 4.67% at 1,000 Bd symbol rate. Even though *GSMem* can be implemented with relatively low error rate in some platforms when the symbol rate is low, as the symbol rate increases, the error rates of *GSMem* implemented in all platforms become extremely high. Therefore, *GSMem* covert channel can not be used to exfiltrate data in high bandwidth.
- When the OOK modulation is used in *BitJabber*, it has a low error rate which is close to 0 at low bandwidth. On most platforms, the error rates are also low when the bandwidth is 100,000 bps. Using the same OOK modulation, *BitJabber* outperforms *GSMem*.

- Most of the time, *BitJabber* implementing B-FSK modulation exfiltrate data with the lowest error rate among all evaluated approaches.
- Using the M-FSK modulation, *BitJabber* can transmit multiple bits with each symbol effectively, the error rate is very low at a low symbol rate. Under the same conditions, 2-bit M-FSK always results in lower error rate than 3-bit M-FSK.
- Considering that 3-bit M-FSK modulation can transmit 3 bits with each symbol, the fastest transmission can reach **300,000 bps**. Compared to *GSMem* at its fastest transmission rate (i.e., 1000 bit/sec), **BitJabber increases the bandwidth by 300 times even with significantly lower error rate**.

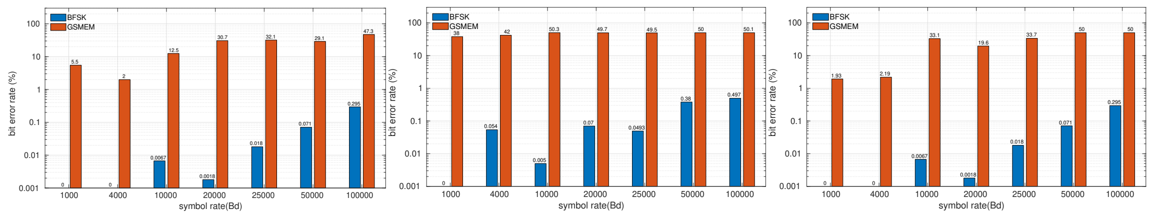
According to Fig. 4.7, evaluated covert channels' performances highly depend on the victim platforms. By comparing the evaluation results and features of EM signals, the intensities of emanated EM signals and background noise have significant impacts on evaluated covert channels, especially for *GSMem*. When the antenna is put close to the victim platforms, the received EM signals from computers all have high intensities, but background noise at different frequency ranges varies a lot. The strongest noise is observed around 800MHz when platform B is evaluated and it is not emanated from the victim computer. Therefore, we can observe that on platform B, *GSMem* has the worst performance. As mentioned before, the despread technique used for implementing *BitJabber* enhances EM signal emanated from computers and suppresses the other irrelevant signals so the performance of *BitJabber* on platform B is not seriously affected. More detailed analysis of the factors influencing error rates will be given in Section 4.3.6.

4.3.4 Through-Wall Evaluations

Compared to the other covert channels, one advantage of EM covert channels is that EM signals can travel through many non-metal obstacles with little energy loss. In this experiment, *GSMem* and *BitJabber* are evaluated in a more practical scenario. The sender machine is put in an isolated room with a 15cm thick wall. The distance between the sender and the wall is 50cm. The receiver is set in the next door sharing the same wall with the sender's room.

Similar to the previous evaluation, background noise exists in both rooms and there are even some wire cables with unknown layout in the wall. In this scenario, the received EM emanations generated by the DRAM clock is weaker and more noise is in the transmission process. Wall-penetrating performance of *GSMem* and our *BitJabber* using the B-FSK modulation are evaluated and the results are shown in Fig. 4.8. From the figures, we can conclude that:

- Compared to results in Fig.4.7, performances of both covert channels get worse to some extent.



(a) Platform A (b) Platform B (c) Platform C

Figure 4.8: Bit error rate of *GSMem* and *BitJabber* using the B-FSK modulation measured with a wall between the receiver and sender

- *GSMem*'s performance is seriously affected and the error rates exceed 25% with symbol rate of 25,000 Bd on all platforms.
- Performance of our *BitJabber* using the B-FSK modulation is only slightly affected compared to *GSMem*.

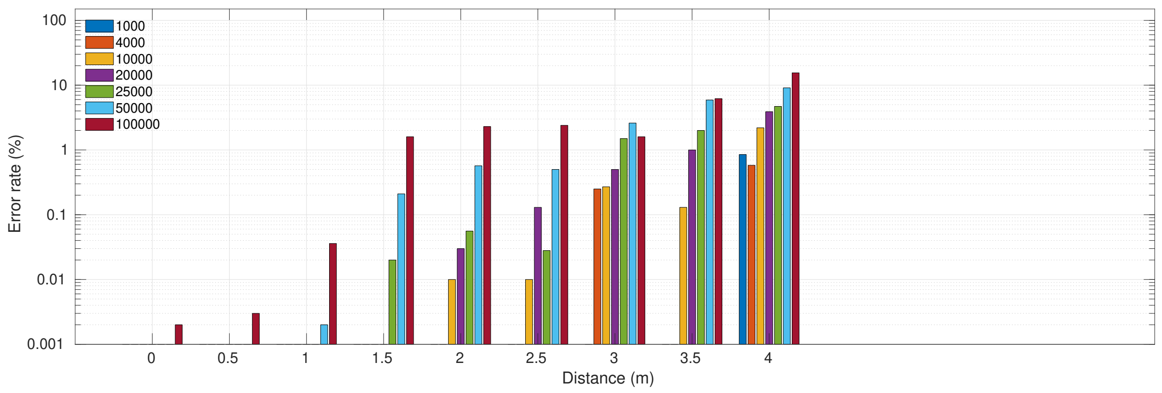
During the evaluations, we found that the office wall has little influence of EM signal intensities but the distance between senders and receivers matter. Similar results to Fig. 4.8 can be obtained when the receivers and senders are separated in same distances but without being wall-gapped.

4.3.5 Attacking Distance Evaluations

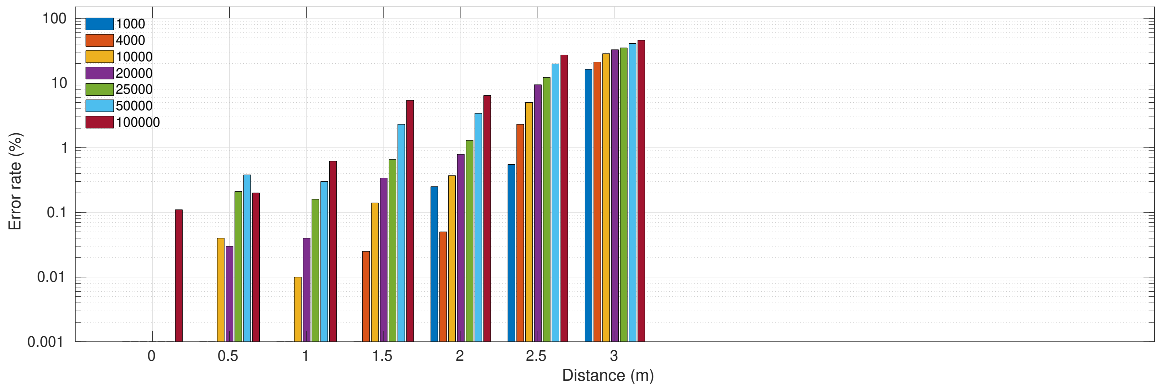
Benefit from the stronger carrier signal, *BitJabber* can be used to perform long-distance data exfiltration. In our experimental environment, implementing *GSMem* at distances longer than 1 meter is very hard because the SNR gets too low to be exploitable. Therefore, in this section, we only measure the performance of *BitJabber* when the receiver is located at different distances away from the sender. In the experiments, *BitJabber* implementing B-FSK are evaluated at bandwidth range between 1,000 bps and 100,000 bps. For platforms A and B, the measured distances range from 0 meters to the longest distance where the carrier signals are visible. For platform C, the longest measured distance is 6 meters due to the space limitations of our experiment environment.

The experimental results are shown in Fig. 4.9 From the figures, we can observe that:

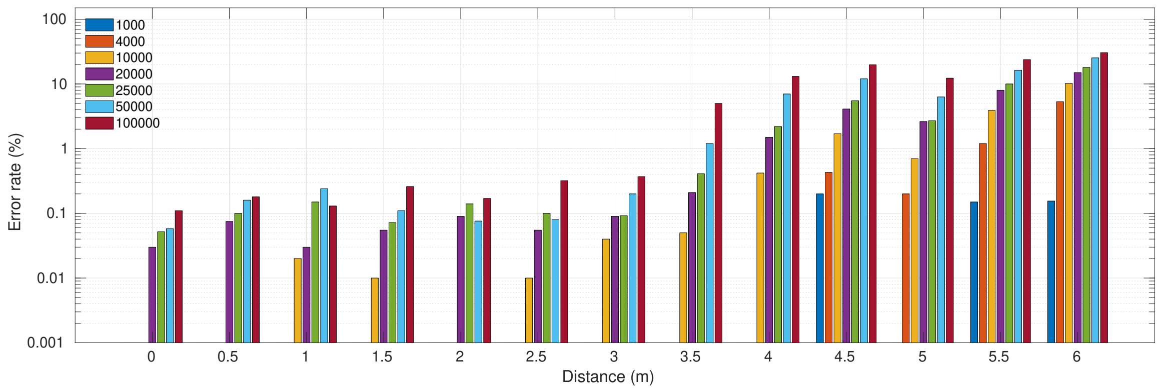
- The error rates are low for all methods with short attacking distances, and they increase as the attacking distances become longer.
- The correlation between error rates and distances gets strong when the error rate is high. While this correlation is weak when the error rate is low.



(a) Platform A



(b) Platform B



(c) Platform C

Figure 4.9: Error rate measured at varied distances for BitJabber implementing B-FSK at different bandwidths

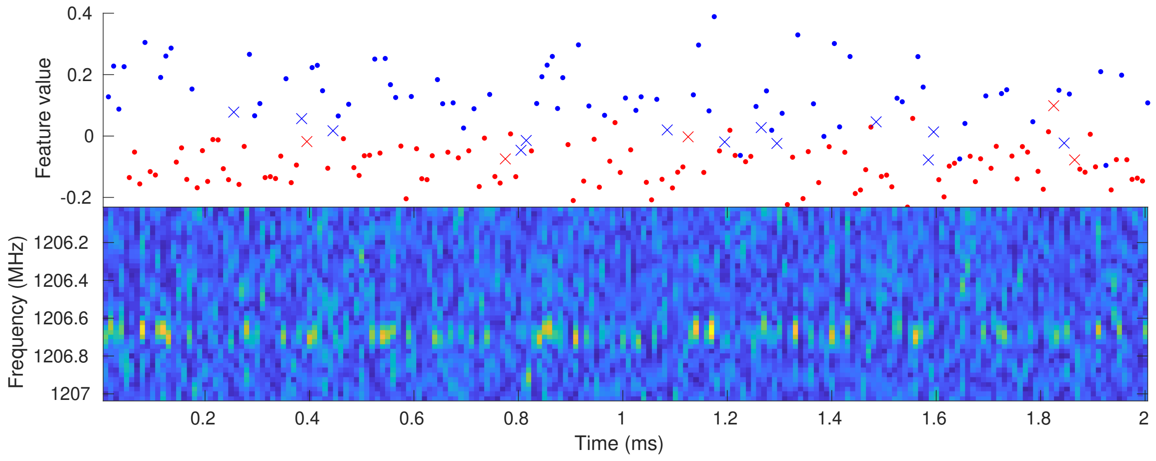
- Platform C is most vulnerable to *BitJabber* at long distances, and the error rate is around 0.1%, with the bandwidth being 1,000 bps at 6 meters away.
- *BitJabber* implemented on Platform C has the highest error rates when the attacking distance is short.

Besides the result presented in Fig. 4.9, one thing worth mentioning is that longer distance between senders and receivers not only reduces the EM signal intensities, but also increases the difficulty of setting antennas. The EM signals emanated from victim computers have different intensities in different directions so the location and orientation of antenna have large influence on the collected signals. When the receiver is moved away from the victim computers, the antenna locations receiving the strongest signals is harder to be determined, i.e. an attacker needs much more effort to receive the exfiltrated data. The difficulty of finding the best antenna locations is related to the computer case materials. When the receivers and senders are put more than 4 meters away, we can not find any antenna locations to collect exploitable EM signals from platform A and B, but the EM emanations from platform C can still be easily observed. If we replace a metal plate of platform A's case with tampered glass, we can observe strong EM emanations even if the receiver is put more than 6 meters away.

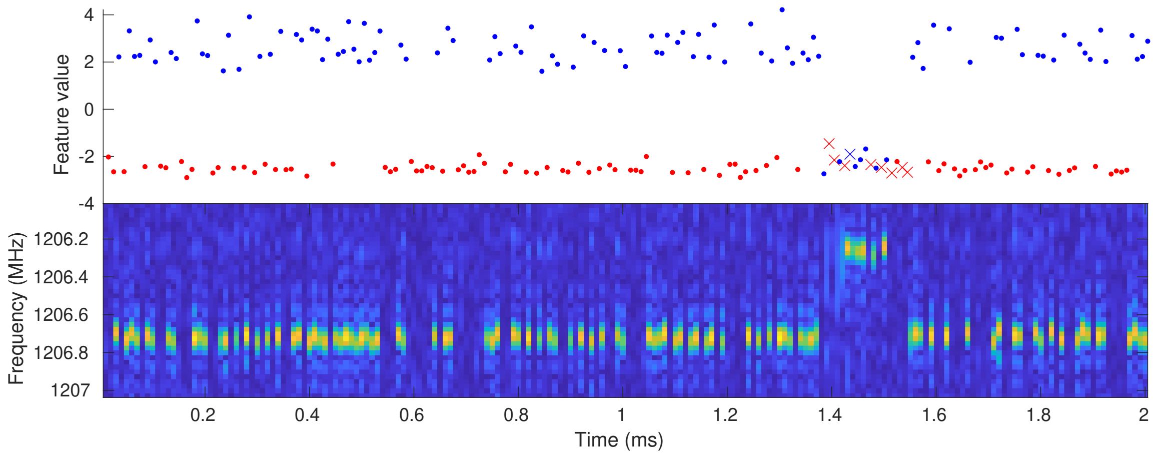
4.3.6 Error Analysis

We have observed that the error rate is influenced by factors including symbol rates and attacking distances from the above evaluations. Although we can conclude that the error rate generally increases with higher symbol rates and longer attacking distances, which also agrees with our intuition, we notice that some measurements do not strictly follow this relation. In order to better understand the threat posed by *BitJabber*, we try to find out all factors influencing the error rate using data collected at Section 4.3.5. Finally, we identify three types of errors. For each type of error, we select a symbol value sequence where that error occurs. We plot the symbol value sequences and spectrograms around the corresponding frequencies in Fig. 4.10

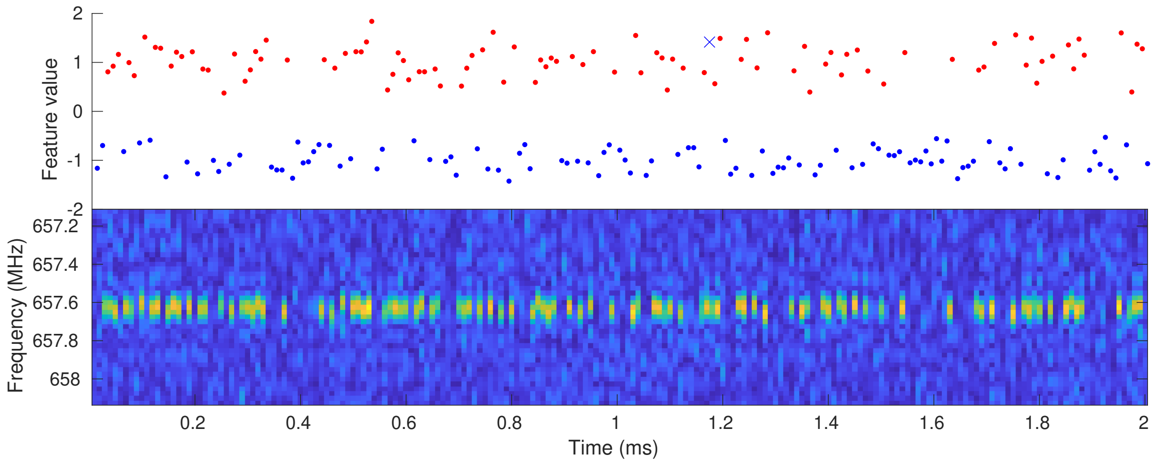
The first type of error is caused by low SNR, as shown in Fig. 4.10a. This sequence is collected when *BitJabber* implements 100,000 bps B-FSK modulation on platform C with the sender and receive located 3.5 meters away from each other. As we can see from the spectrogram, the signal collected at this distance is very noisy. Two symbol values can not be clearly distinguished by looking at the feature values due to the low SNR. This type of error can explain how the error rate is related to the symbol rate and attacking distance. Because the SNR decreases with increased symbol rate and attacking distance, we observe the inverse correlations between error rate and symbol rate and between error rate and attacking distance. Under the same conditions, signals emitted from platform C have the highest SNR, which makes it most vulnerable to long-range attack. However, when the SNR is high enough for distinguishing different symbols, higher



(a) error caused by low SNR



(b) error caused by shifted memory access frequency



(c) error caused by the sender

Figure 4.10: Three types of error. Feature value sequences and the spectrograms around the corresponding frequencies when three different types of error occur. Color blue and red represent transmitted symbols with values '0' and '1' respectively. Dots (●) and crosses (×) denote correctly and wrongly classified symbols respectively.

SNR does not help further lower the error rate, which is why the above correlations get weaker with low error rates. As evaluations in previous sections indicated, error rates do not always reach zero with very short distances between the sender and the receiver when the SNRs are high enough for distinguishing two symbol values. The other two types of errors dominate in these cases.

The second type of error is caused by shifted memory access frequency, as shown in Fig. 4.10b. This sequence is collected when *BitJabber* implements 100,000 bps B-FSK modulation on platform C with the sender and receive located very close to each other. As we can see from the spectrogram, the captured signal has a very high SNR. Nevertheless, we observe that the side lobe frequency is shifted by 0.5 MHz at 1.5 ms. Accordingly, the computed feature vectors are erroneous at this part and many errors occur. In our experiments, this kind of unexpected frequency shift is most often observed on platform C but rarely on platforms A and B. This type of error explains why *BitJabber* implemented on platform C fails to reach a very low error rate despite the highest SNR.

The third type of error is introduced when the wrong symbol is transmitted by the sender, as shown in Fig.4.10c. This sequence is collected when *BitJabber* implements 100,000 bps B-FSK modulation on platform A with the sender and receive located very close to each other. As we can see from the spectrogram, the received signal has a very high SNR with no unexpected frequency shift. The only misclassified symbol has a feature value entirely in accordance with the feature values of the other symbol value. When the sender transmits data at a very high speed, it may fail to update the data to transmit in time (e.g., due to task scheduling). Subsequently, the sender modulates the carrier based on the incorrect data and transmits the wrong information. All our evaluations are performed with sender running on relatively idle computers, in which case this type of error appears occasionally on platform A and B but rarely on platform C.

In conclusion, the first type of error is related to the signal quality received by the receiver. Both the second and third types of errors depend on the working states of senders on victim machines. These error analysis results offer us deeper insight into countermeasures of *BitJabber*.

4.4 Countermeasures

Based on the error analysis results in Section 4.3.6, we propose countermeasures against *BitJabber* in several different directions.

One direction of countermeasures aims at lowering the SNR of received signals to increase the first type of error. This can be achieved either by reducing the EM emanation intensity or increasing the background noise. EM shielding is a commonly used technique to reduce the EM emanation intensities. Since EM signals can travel through normal walls, metal shields like the Faraday cage are needed to block EM wave propagation. As reported in Zajic and Prvulovic (2014), EM emanations from metal-shielded computers are weakened. Our evaluations in Section. 4.3.5 also show that long-range data exfiltration from computers in metal cases using *BitJabber* is harder to implement. However, we need to keep in mind that metal-shielding does not completely eliminate this covert channel, as we've seen in our evaluations that data exfiltration is still possible from 4 meters away for platform A that has a metal case. SNR can also be lowered by increasing the noise level in the surrounding environment, which can be achieved using signal interference devices to jam the frequency range around the carrier signal. However, our approach will disperse the power of random noise after de-spreading the EM signal generated by the DRAM clock. As our evaluations indicate, the random noise irrelevant to victim computers only has a slight influence on *BitJabber*'s performance. However, the noise with SSC patterns can effectively increase the error rate. Therefore, to better mitigate this covert channel, the noise generator can produce noise with an SSC pattern to disturb the de-spreading process.

The other direction of countermeasures targets the sender running on the guarded computer. Because performing memory activities in stable frequencies is essential to implement *BitJabber* with a low error rate, we can execute some memory-intensive applications to disturb the sender's memory access behavior. In this way, the second type of error may be significantly increased. Besides, we can also increase the third type of error by introducing more computation load in the protected computers. However, all these methods require the computer to stay busy to some extent, which may hurt the computer's performance sometimes.

Furthermore, since *BitJabber*'s performance is highly dependent on the de-spreading of SSC signal, a good idea of mitigation is preventing the de-spreading process. In most modern computers, SSC is implemented by FM modulating the clock signal with a simple periodical signal. This de-spreading process can be easily reversed to recover the modulating signal. If we use a more complicated SSC technique (e.g., using a secret random number sequence to FM modulate the clock signal), the attacker can not restore the high-SNR carrier, and the implementation of *BitJabber* is much harder.

4.5 Conclusion

In this chapter, the EM radiation of the DRAM clock is exploited to implement a covert channel. We restore a high-SNR carrier by de-spreading the DRAM clock's EM emanations and applying multiple modulation techniques to exploit EM signals to efficiently exfiltrate data from air-gapped computers. The performance of our covert channel *BitJabber* is evaluated and compared with an existing covert channel *GSMem*, which exploited the same EM emanations from the DRAM clock. *BitJabber* can reach a bandwidth of 300,000 bps with an error rate under 1%. It can also perform wall-penetrating data exfiltration and long-range data exfiltration. According to Davidov and Oldenburg (2020), people used to consider the SSC technique a countermeasure for EM side-channel attacks, but our work shows that this countermeasure can be easily invalidated. Although only the DRAM clock's EM emanations are investigated in this work, we also notice that many other components in computers also generate strong EM emanations after de-spreading, which may also be exploitable for performing threatening covert channels. This covert channel greatly increases the maximum data exfiltration speed for air-gapped computers by exploiting EM side-channels, making people pay more attention to the protection against EM attacks.

CHAPTER 5

Screen Content Detection Using EM Emanations From GPU

Over the past few years, graphics processing units (GPUs) have become an integral part of modern computer systems, which are used not only for graphics rendering but also for intensive parallel computing. Given the fact that many tasks running on a GPU operate on sensitive information, concerns about the security of GPUs, especially potential information leakage, have been raised. Several recently developed attacks on GPU have justified such concerns Lee et al. (2014); Zhou et al. (2017a); Pietro et al. (2016); Luo et al. (2015); Jiang et al. (2016, 2019); Gao et al. (2018); Maurice et al. (2014); Naghibijouybari et al. (2018).

Although these existing GPU attacks focus on different application scenarios, they all require that GPUs be either logically shared with or physically accessible[1] to adversaries. If an attacker has no physical or logical access to the GPUs used by her targets, is it still possible that the attacker can steal sensitive information from such GPUs? In this chapter, we answer this question affirmatively by presenting a new physical side-channel vulnerability of modern GPUs as well as some examples of its exploitation[2].

Specifically, we have discovered certain electromagnetic (EM) emanations[3] from GPUs which are: ❶ exploitable – we find that these EM signals are computation-dependent and can reveal fine-grained information about the ongoing activity; and ❷ easy to measure – we find that such EM signals are strong and can propagate very far (e.g., more than 6 meters in many cases) as well as even penetrate thick walls. A further investigation reveals that the root cause of such exploitable far-field EM emanations is the dynamic voltage and frequency scaling (DVFS) feature of GPU, which has been playing an important role in either saving energy or improving GPU performance Nath and Tullsen (2015); Mei et al. (2013); Abe et al. (2014).

By exploiting the discovered EM side-channel information, for the first time, we demonstrate that it is not only possible but also practical to mount realistic eavesdropping attacks. Given a victim who uses a modern GPU without sharing, we show that an attacker can spy on the victim and identify the webpages visited by the victim (i.e., website fingerprinting attack) with a high accuracy. In addition, we show that the keystroke timings of the victim can be further inferred, which may be used to recover typed words or passphrases Song et al. (2001); Zhang and Wang (2009). As the attacker can be several meters away from the victim and the attacker can even hide in a separate cubicle or room, the presented attacks are in effect extremely stealthy.

To the best of our knowledge, our work serves as the *first* physical side-channel attack on non-shared

[1] In Gao et al. (2018), although contactless, the attack needs to remove GPU's heat sink and place a probe near GPU chip surface. In Luo et al. (2015), it is the power supply of the computer rather than the GPU that is instrumented by the attacker.

[2] The work in this chapter has been reported in Zhan et al. (2021b)

[3] In this chapter, we use EM emanations and EM signals interchangeably.

GPUs at a distance. Prior to our work, there are only a few studies on leveraging physical side effects of GPU computation (e.g., power Luo et al. (2015) or near-field EM Gao et al. (2018)) to breach confidentiality, but all of them require physical access to the GPUs. On the contrary, we discover and exploit a far-field EM side-channel vulnerability, which empowers more practical attacks without too unrealistic proximity requirements.

Even outside of the GPU security area, we find that most of the existing *long-range* EM-based attacks are just to build covert communication channels Guri et al. (2015a); Sehatbakhsh et al. (2020); Shen et al. (2021). Surprisingly (or not), there are only very few works showing that it is possible to mount EM side-channel attacks on modern computers to steal sensitive information from several meters away.[4] This is because EM emanations that are both far-field and exploitable for eavesdropping attacks appear to be hard to discover. Thus, our work also serves as a good demonstration of long-range EM side-channel attacks.

The main contributions of this chapter are as follows:

- We present a new EM side-channel vulnerability that we have discovered in modern GPUs and can be exploited to carry out attacks at a distance and/or through a wall. We identify the ubiquitously used DVFS as the root cause of this side-channel and find that such a vulnerability exists in many GPUs of both NVIDIA and AMD.

- We formulate a signal processing framework to address the challenges introduced by potential EM shielding and strong noise contamination. With the proposed techniques, we can exploit the EM emanations of interest even when they are greatly attenuated and/or overwhelmed by strong legitimate communication signals.

- We conduct two case studies on the exploitation of this newly found EM side-channel vulnerability. The first one is a website fingerprinting attack, and up to 93.2% accuracy can be achieved in a scenario where the attacker and victim are 6 meters apart. The second case study is a keystroke timing inference attack, where we show that keystroke events can be reliably detected to deduce inter-keystroke times.

- We show that even though disabling GPU DVFS can be an effective approach to mitigating the discovered EM side-channel vulnerability, it will unfortunately introduce another new one into many GPUs which can be exploited to mount comparable EM side-channel attacks. We also discuss some potential countermeasures.

As DVFS has been used or may appear in many other hardware components Miyoshi et al. (2002); Kim et al.

[4]So far, we have only found the attack range in Kuhn (2004) comparable to ours, but it is not clear if that exploited EM side-channel vulnerability still remains when contemporary cables, e.g., HDMI or DP, are used.

(2008); Mishra et al. (2009); Deng et al. (2011); Chen et al. (2013), our research also gives a pointer to what may need additional attention during certain security investigations.

Responsible Disclosure

We have reported our findings to both NVIDIA and AMD. NVIDIA replied to us that "NVIDIA is continually investigating ways to minimize board emissions in future product designs and will take these findings and recommendations under advisement". AMD informed us that their engineering team reviewed the issue and "no effective mitigations or fixes have been identified".

5.1 Background

In this section, we provide a brief overview of GPU architecture and GPU DVFS feature. Note that, in this chapter, we do not consider any integrated GPUs in CPU processors, namely the term GPU is used to indicate the discrete ones designed by NVIDIA or AMD only. Moreover, we briefly present the physical side effects exploited in this chapter, namely the EM emanations.

5.1.1 GPU Architecture

GPUs have evolved from hardwired graphics accelerators into highly parallel programmable computing devices. Usually, a modern GPU contains a number of single-instruction multiple-thread (SIMT) processors. Each SIMT processor has many simple GPU cores, and each core can perform scalar integer and floating-point arithmetic operations. Such SIMT processors are called *streaming multiprocessors* by NVIDIA and *compute units* by AMD. SIMT processors manage, schedule, and execute groups of parallel threads, which are named *warps* and *wavefronts* in the terminology of NVIDIA and AMD respectively.

To store large amounts of data, a modern GPU normally has several gigabytes of memory. Such large GPU memory is shared by all the SIMT processors on the GPU, and consists of multiple memory modules. These memory modules are of special DRAM type tailored for use in GPUs (e.g., GDDR5 and GDDR6). In general, a GPU needs high memory bandwidth to sustain its high computational throughput, and there are multiple memory controllers used to enable massively parallel access to the memory modules to reach the desired bandwidth.

In a GPU, the SIMT processors and the GPU memory are connected through an on-chip interconnect such as a crossbar. The GPU memory is independent of the main memory on the host side and also managed in its own manner. Data transfers between the main memory and GPU memory are via the PCIe bus.

5.1.2 Dynamic Voltage and Frequency Scaling

DVFS is a power management technique that has been widely used with respect to CPUs Weiser et al. (1994). It dynamically changes voltage and frequency to adjust performance for power savings.[5] Instead of always staying at the highest level, performance is actively regulated according to current workloads, which can make very efficient use of energy.

As GPUs continue to grow powerful, their increasing power consumption has become a radical problem. To address this problem, the DVFS technique has been applied to GPUs. In fact, almost every modern GPU provides hardware support for DVFS Nath and Tullsen (2015). For a GPU, DVFS governs the supply voltage and frequency of both its cores and memory.

Normally, there are multiple performance levels specified by GPU DVFS. (The performance levels are often called performance/power states, namely, P-states.) Each performance level defines a setting of voltage and frequency for the GPU cores and memory. (At a performance level, the frequency and/or voltage of the GPU cores may not be fixed but can vary within a specific range, while the frequency of the GPU memory usually does not change.) GPU DVFS dynamically switches performance levels to meet the current computation needs and minimize power draw, heat generation, and fan noise. In general, the approach to determining performance levels in official GPU drivers is proprietary and not well-documented. The default GPU DVFS approach is employed automatically, although an end user may choose to disable its functionality by manually setting fixed frequencies or to provide a customized approach Ma et al. (2012).

5.1.3 Electromagnetic Emanations

Given the fact that electric current in the circuitry of a device varies with time, EM emanations inevitably arise. The EM emanations generated by a computer system are distributed widely in the spectrum. As these EM signals generally carry information about the underlying electronic activities, which can be linked with certain high-level activities, some of them have been leveraged in the context of security for attacks Kuhn (2004); Genkin et al. (2015a, 2016a,b); Alam et al. (2018); Sehatbakhsh et al. (2020) as well as defenses Han et al. (2017); Nazari et al. (2017).

Although the sources of many of the EM emanations are unknown, a few of them are in effect easy to determine, e.g., emanations generated by some components whose activities are periodic, such as voltage regulators and DRAM clocks Callan et al. (2015). The EM signals created by these components having periodic switching behavior are also strong and may propagate to a distance of several meters. Interestingly, some of these signals may be unintentionally modulated by other activities in the form of amplitude-modulation

[5]DVFS principally reduces dynamic power consumption P_D, which is proportional to the voltage V quadratically and frequency f linearly, namely, $P_D \propto V^2 \times f$.

(AM) or frequency-modulation (FM) Callan et al. (2015); Prvulovic et al. (2017). For example, the EM signals created by voltage regulators may be AM-modulated by activities in the circuits they power. Therefore, these signals act as carrier signals that convey information about the modulating activities. Moreover, to measure these far-field EM signals, very simple equipment suffices. For instance, a whip antenna and a cheap software-defined radio (SDR) device are adequate Sehatbakhsh et al. (2020); Zhang et al. (2020b).

5.2 Threat Model

We assume that there is an attacker who intends to eavesdrop on a victim to extract some of his/her sensitive information, e.g., the webpages the victim is browsing. The attacker is in the proximity of the victim, but they may still be well spaced apart. For example, the attacker and the victim may be colleagues or neighbors. Furthermore, they may be physically isolated from each other. For instance, the victim may be in a separate cubicle, office, or apartment, to which the attacker has no access.

The victim uses a desktop computer system which is equipped with a discrete GPU. (We do not consider laptops or other mobile computing devices in this threat model.) The GPU may be a product of either NVIDIA or AMD. We assume that the victim uses the official driver and its default settings, which is the most prevalent case in reality. The attacker is assumed to be able to find and employ the same type of GPU as the one used by the victim for profiling. (The attacker may have known what GPU the victim is using, but if this a priori knowledge is not available, we will show that the attacker may still be able to deduce it.) Unlike the prior work on GPU side-channel or memory dump attacks Lee et al. (2014); Naghibijouybari et al. (2018); Maurice et al. (2014); Zhou et al. (2017a), we do not require that the use of the GPU or any other computational resources be shared between the victim and the attacker. We neither require the presence of any software vulnerabilities.

5.3 New Exploitable EM Emanations

Given the aforementioned threat model, an attacker may opt for EM side-channel attacks, because (1) they are passive and non-intrusive; and (2) they may be mounted at a distance and work through walls. Nevertheless, one challenging problem is to find certain exploitable EM signals which may hide in any place of the spectrum. In this section, we present our newly discovered, exploitable EM emanations that are generated by the memory clock in a modern GPU.

5.3.1 Experimental Setup

In this chapter, we carried out all the experiments using an SDR device, USRP B210, and an ultra-wideband directional antenna, RFSPACE UWB-3. As shown in Figure 5.1, the antenna is directly connected to the SDR

Table 5.1: List of GPUs investigated in this chapter

GPU Card (Vendor)	Architecture	Memory	WCK Frequencies	Release Data
AMD Radeon RX 580 (GIGABYTE)	GCN 4.0	8 GB GDDR5	600 MHz, 2000 MHz, 4000 MHz	Apr. 2017
AMD Radeon RX 5600 XT (MSI)	RDNA 1.0	6 GB GDDR6	400 MHz, 2000 MHz, 2500 MHz, 3500 MHz	Jan. 2020
AMD Radeon RX 5700 XT (XFX)	RDNA 1.0	8 GB GDDR6	400 MHz, 2000 MHz, 2500 MHz, 3500 MHz	Jul. 2019
NVIDIA Geforce GTX 1080 (PNY)	Pascal	8 GB GDDR5X	405 MHz, 810 MHz, 4513 MHz, 5005 MHz	May 2016
NVIDIA Geforce GTX 1650 (MSI)	Turing	4 GB GDDR5	405 MHz, 810 MHz, 4001 MHz	Apr. 2019
NVIDIA Geforce RTX 3060 OC (ASUS)	Ampere	12 GB GDDR6	405 MHz, 810 MHz, 5001 MHz, 7301 MHz, 7501 MHz	Jan. 2021

device via a coaxial cable without any other amplifier/filter front-end modules in-between. The bandwidth we need here is 25 MHz, and the USRP B210 can provide 56 MHz of instantaneous bandwidth in the frequency range of 70 MHz to 6 GHz, which is more than sufficient for our needs. The GNU Radio framework is used to capture and process the signal of interest.

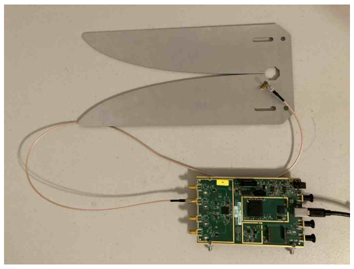


Figure 5.1: Signal measurement equipment: USRP B210 and RFSPACE UWB-3

The modern-looking computer case with a translucent side panel (which is acrylic) is AeroCool Cylon RGB Mid Tower. The monitor is HP VH240A. The motherboard installed in this case is ASUS PRIME Z270-P. The CPU used in this system is Intel i5-6500T. The power supply unit (PSU) installed in this case is Apevia ATX-JP1000 Jupiter 1000W.

The all-metal computer case is Thermaltake Versa H22 Mid Tower. The monitor is HP VH240A. The motherboard installed in the case is ASRock Z270 Killer SLI. The CPU used in this system is Intel i3-6100. The PSU installed in this case is Thermaltake Smart 700W.

When we performed the experiments reported in Section 5.3.4, we used AMD GPU driver for Linux (AMDGPU 20.20) to set the performance level to the second lowest one. However, NVIDIA GPU driver does not allow us to fix the GPU to a specific performance level except for the highest one. To ensure the appearance of the signals of interest, we played YouTube videos to make sure that the performance level bumps up to the second lowest one frequently.

5.3.2 EM Signal of GPU Memory Clock

First of all, we show the characteristics of the EM emanations of interest. As mentioned in Section 5.1.2, GPU DVFS defines multiple performance levels, and it switches between these levels in accordance with the current GPU workloads. In other words, the clock frequencies of the GPU cores and memory often change. Clocks usually create strong EM emanations, and hence when a performance level is on/off, we expect to observe the appearance/disappearance of clear EM signals at the corresponding clock frequencies in the spectrum. To verify this anticipated feature, we test several AMD and NVIDIA GPUs made by different vendors and equipped with different types of GDDR, which are listed in Table 5.1. These GPUs are very commonly used, and almost all the recent architectures of AMD and NVIDIA are covered by them, including the latest NVIDIA Ampere architecture.

We alter the performance level of each GPU, and examine the spectral behavior at the corresponding core and memory frequencies. The inspection results are as follows: (1) the EM emanations generated by the GPU core clock can be hardly found; but, (2) the EM signal of the GPU memory clock is very noticeable and its behavior matches our anticipation; and (3) interestingly, the EM signal consists of many frequency components that are over a wide range in the spectrum. For example, Figure 5.2[6] illustrates (2) and (3) when the first GPU in Table 5.1 (i.e., AMD Radeon RX 580) is used.

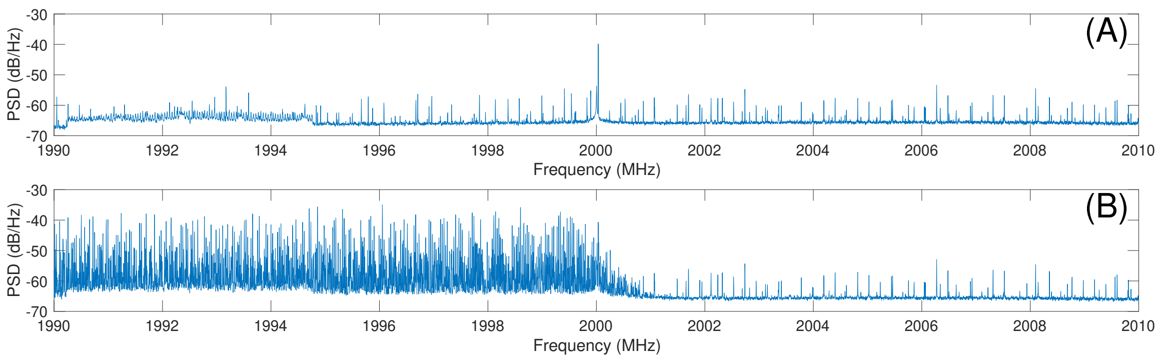


Figure 5.2: Spectra around 2000 MHz in the case of using AMD Radeon RX 580: (A) when GPU memory clock frequency is 600 or 4000 MHz, and (B) when frequency is 2000 MHz

Figure 5.2 (A) shows that we can barely see any signal energy around 2000 MHz if RX 580 memory clock is not set to the corresponding level. On the other hand, Figure 5.2 (B) shows that we can clearly observe the EM signal of RX 580 memory clock in the frequency-domain when the clock is set to 2000 MHz. Moreover, we can see that the signal has a large number of spectral components in the frequency range below 2000 MHz. The reason for such a phenomenon is due to the use of a hardware feature called spread spectrum clocking (SSC) for meeting electromagnetic compatibility (EMC) regulations. EMC standards impose allowable limits on EM signal energy at any frequency above 30 MHz, and many clock signals (e.g., the GPU memory 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

[6]The power spectral density (PSD) is computed using the Welch's method. The FFT size is 8192, and a Hamming window is used. Ten segments without overlap are averaged.

energy is spread over that range of frequencies Hardin et al. (1994).

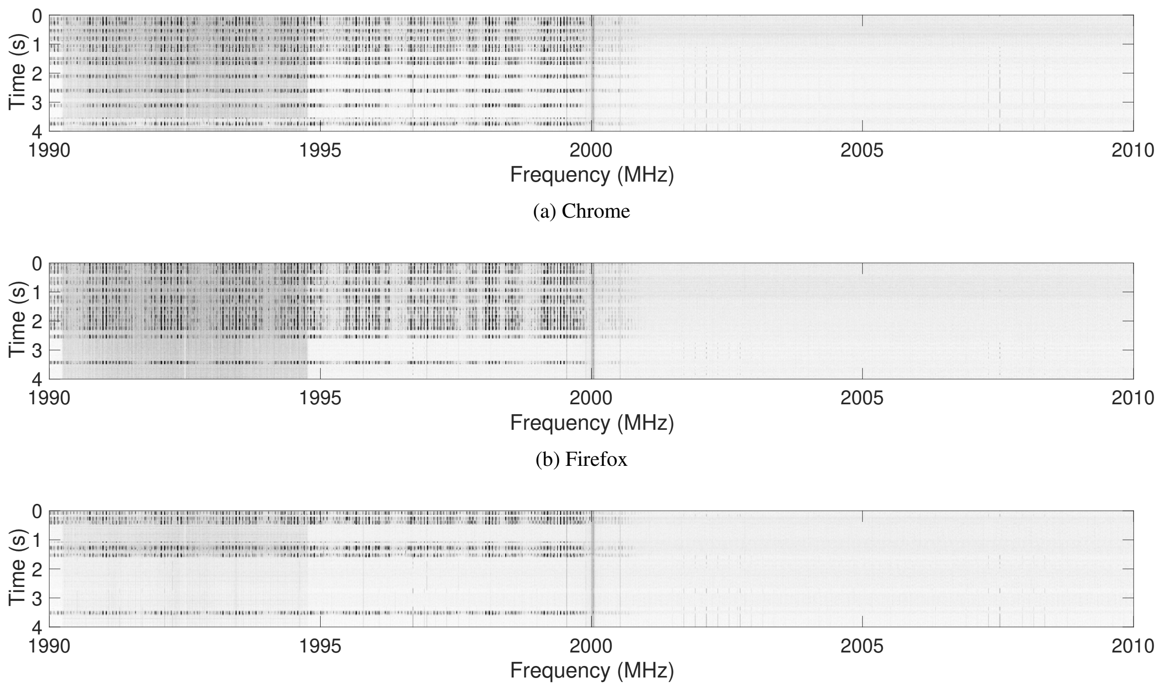
Note that, since GDDR5, GPU memory operates with two types of clocks. One type is referred to as command clock (CK) that is used for sending commands and addresses, and the other one is referred to as write clock (WCK) that is used for data reads and writes. *The EM emanations of interest are specifically generated by the WCK*. In the case of GDDR5, the frequency of WCK is half the data rate Micron Technology, Inc. (2014). In the cases of GDDR5X and GDDR6, the frequency of WCK is half the data rate if the operating mode is set as double data rate (DDR), or one fourth the data rate if the operating mode is set as quad data rate (QDR) Micron Technology, Inc. (2017).

Since many GPUs have different sets of WCK frequencies, an attacker may exploit this fact to find out what GPU is being used by a victim if this knowledge is unknown to the attacker beforehand. Essentially, the attacker monitors the spectrum at all of the possible WCK frequencies and uses the appearance of the EM signals similar to the one shown in Figure 5.2 (B) to determine which WCK frequencies the target GPU has. Such reconnaissance information can be used to pinpoint potential GPUs.

5.3.3 Activity Identification

To be exploitable, the EM emanations should be computation-dependent so that high-level activities can be inferred to reveal certain sensitive information. We notice that the performance level of a modern GPU is usually changed rapidly to seek a balance between performance and power consumption. Given a computational activity that creates some GPU workloads, there can be multiple GPU performance level switches during the activity. As different activities potentially impose different loads on GPU at different times, distinct performance level switching behaviors should be induced, which can thus serve as activity signatures. In the above discussion, we have learnt the correlation between the performance level switches and the appearance/disappearance of targeted EM signals, which implies that the EM emanations of interest can be leveraged to identify different activities.

To fully capture the behavior of performance level switching, we may try to monitor all of the frequency ranges where the GPU memory clock signals can arise in a synchronized manner. However, our experiments show that such a heavyweight approach to gaining a complete picture of switching behavior is not necessary, but a partial picture concentrating on when a specific performance level is switched on/off suffices to distinguish different activities. For instance, Figure 5.3 demonstrates the spectrograms when launching three different programs, that are Chrome, Firefox, and LibreOffice Writer, on a system equipped with an AMD Radeon RX 580 GPU. (The OS is Ubuntu 18.04 and the GPU driver is AMDGPU 20.20.) The frequency range on which we focus corresponds to the second lowest WCK level (c.f. Table 5.1 and Figure 5.2). As we can see from the figure, the patterns of stripe appearance on the spectrograms are distinguishable from



(a) Chrome

(b) Firefox

(c) LibreOffice Writer

Figure 5.3: Spectrograms having frequency range of 20 MHz centered at 2000 MHz when launching three applications on a system equipped with an AMD Radeon RX 580 GPU

each other and we have also verified that they are fully repeatable. Thus, such patterns can be treated as fingerprints to help infer which program is being launched.

Therefore, we need only to consider one specific frequency of WCK and keep track of when the corresponding EM signal energy appears and disappears in the focused frequency range to gain knowledge (e.g., in the form of spectrogram) that can be exploited for activity identification. Theoretically, we may choose any of the possible WCK frequencies, but empirically, we find that the second lowest one generally yields the optimal exploitation results. The reasons for the second lowest WCK frequency being a better choice than others include: (1) this frequency is reached much more often than the other higher ones, especially when an activity does not use the GPU intensively; and (2) we notice that it normally has a higher signal-to-noise ratio (SNR) than the lowest one. Thus, in the following, if not otherwise specified, we focus on the EM emanations generated by the WCK when switched to its second lowest level.

Note that, to further substantiate the tight correlation between the patterns of stripe appearance on the spectrograms in Figure 5.3 and the WCK frequency switching behavior, we also obtain the traces of GPU memory clock frequency changes via an interface exposed by the AMD GPU driver[7] and compare the traces with the spectrograms. The experiments verify that the patterns well match the frequency change traces, and

[7]The interface is `/sys/class/drm/card0/device/pp_dpm_mclk` that does not give WCK frequency directly. In terms of AMD Radeon RX 580, we need to double the reading to get the current WCK frequency.

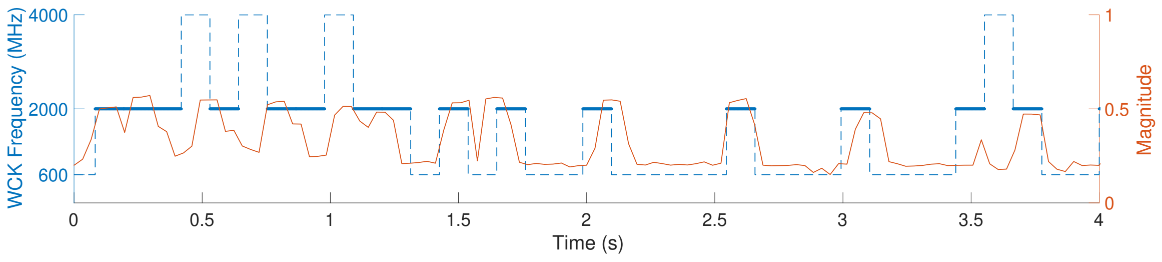


Figure 5.4: Traces of WCK frequency alterations and averaged magnitude with respect to Figure 5.3 (A)

Figure 5.4 shows such an example corresponding to Figure 5.3 (A), i.e., launching Chrome. In Figure 5.4, we can see that the WCK frequency is switched among its three possible levels (namely the blue dashed line), and we make the line segments solid and bold when the WCK frequency is switched to its second lowest one, namely, 2000 MHz. For each of the spectra constituting the spectrogram in Figure 5.3 (A), we plot the average of the magnitudes in the frequency range from 1990 MHz to 2000 MHz (namely the red solid line). As we can see from the figure, the local peaks of the averaged magnitude match closely with the line segments indicating that the WCK is switched to 2000 MHz.

5.3.4 Propagation Distance and Wall Penetration

To exploit a physical side-channel in an air-gapped setting, we also need to consider how far it can propagate and whether it can go through obstacles like a wall. To this end, we have performed several experiments on the GPUs listed in Table 5.1 and found that the EM emanations of interest have a desirable wall-penetrating property and can be measured by an attacker from a distance long enough to carry out attacks practically.

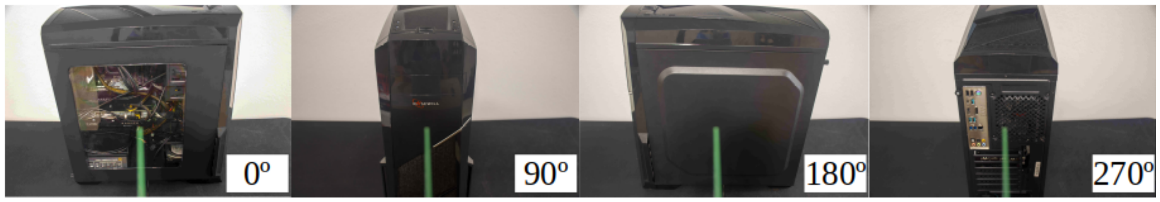
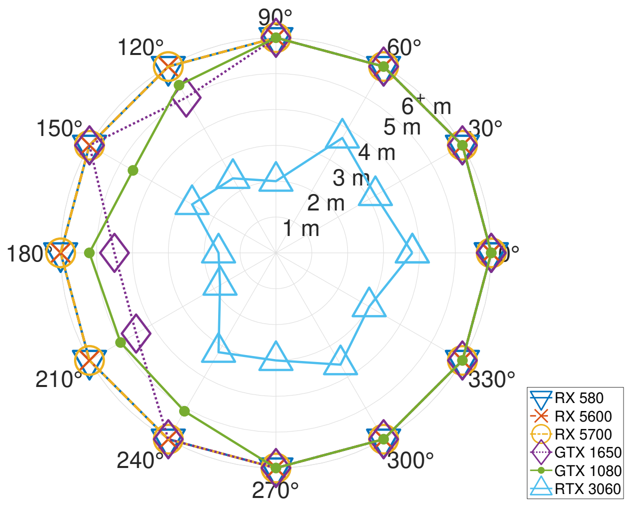
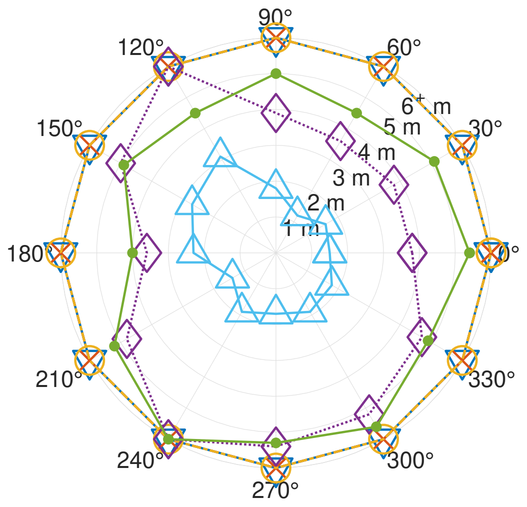


Figure 5.5: The definition of directions used in this chapter

Figure 5.6 shows the distances with respect to the directions from which the EM emanations of interest can still be “picked up” when there is no obstacle in between the GPU machine and the antenna. Here we define “pick up” as that the SNR after applying the technique described in Section 5.4 is at least 7 dB. In general, we can find that the longest measurement distance changes with the direction. Note that, due to our office space limitation, the maximum distance we can reach is 6 meters. The definition of directions is illustrated in Figure 5.5. We can see that 0° is defined as when the antenna is orthogonal to the case side from which the motherboard and GPU are installed, and 90° is defined as when the antenna is perpendicular to the front side of the case.



(a) Modern-looking case



(b) All-metal case

Figure 5.6: Propagation distance (no obstacles in between)

Figure 5.6 shows two scenarios in which two types of computer cases are used. In the first scenario, a modern-looking computer case with a translucent side panel is used (as shown in Figure 5.5), and in the second scenario, a computer case whose every side is made of metal is used. Comparing Figure 5.6 (A) and (B), we can observe that the translucent side panel in the first scenario benefits the propagation of the exploitable EM signals. As illustrated by Figure 5.6 (A), when a modern-looking case is used, we can capture the EM emanations of our interest from more than 3 meters away in almost every direction no matter which GPU is used. On the other hand, in the second scenario, the all-metal computer case can attenuate the strength of the EM signals in the directions from 270° to 90° counter-clockwise, but the signals of our interest can still be picked up at a distance of 3 meters or more in many other directions for each GPU. Notice that, nowadays, modern-looking computer cases with a translucent side panel dominate the market Newegg (2021) and are in effect extremely popular among users of mid-range to high-end GPUs. Therefore, in reality, it is very likely that an attacker can easily capture the EM emanations of interest at a very far distance. Even if a computer case with all metal sides is used, the exploitable EM signals can still be measured from several meters away.

Table 5.2: The signal strength reduction due to the walls

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
Plaster Wall	-1.06 dB	-1.03 dB	-1.18 dB	-1.32 dB	-0.21 dB	-0.51 dB
Concrete Wall	-4.41 dB	-4.28 dB	-6.92 dB	-8.64 dB	-5.24 dB	-4.50 dB

We also isolate the GPU machine having the modern-looking computer case in two rooms and test if the EM emanations of our interest can be captured from the outside of the rooms. The first one is an office room whose wall is as thick as 15 cm and made of plaster, and the second one is a lab room which has very thick concrete walls ($\sim$15 cm). We measure and compare the strengths of the exploitable EM signals when the antenna and the target machine are separated by 1 m with and without the walls in between. Table 5.2 shows the EM signal strength reduction due to the walls. From the results, we can observe that the plaster wall can only reduce the EM signal strength by at most 1.32 dB while the concrete wall can reduce the strength by up to 8.64 dB. These results indicate negligible effects of normal plaster walls and manageable effects of thick concrete walls on potential attacks exploiting this newly discovered EM side-channel vulnerability.

5.4 Signal Transformation and Enhancement

Although we may directly exploit the derived spectrograms like the ones shown in Figure 5.3 to identify activities, it can become very difficult to do so under circumstances where the SNR is too low to induce visible stripes on the spectrograms. To address this problem, we introduce two signal processing techniques that can preserve the target signal's appearance and disappearance patterns even when the SNR becomes very low.

5.4.1 Time Series Derivation

As mentioned in Section 5.3.2, SSC is used to spread the energy of a clock signal over a frequency range for meeting the EMC regulations. Given a clock whose frequency is f_c, SSC in effect scatters its energy to a series of N sub-clocks at $f_c - nf_m$, where $0 \leq n < N$ and f_m is the modulating frequency Shen et al. (2021). Typically, f_m is 30 to 33 kHz. Due to factors like path loss or better EM shielding, the power of such sub-clock signals may become too weak compared to the background noise. Inspired by the work in Shen et al. (2021), we leverage the folding technique to amplify the manifestation of the targeted memory clock signal being present.

Assume we perform an M-point discrete Fourier transform (DFT) to derive a spectrum X. Since the frequencies of the sub-clocks are separated by f_m, they should be separated by Δ DFT bins in X, where

$$\Delta = f_m \times \frac{M}{f_s} , \tag{5.1}$$

and f_s is the sampling rate. Let us define $S[i]$ as the sum of the magnitudes of the i^{th}, $(i-\Delta)^{\text{th}}$, $\cdots$, $[i-(N-1)\times\Delta]^{\text{th}}$ DFT bins of X, namely we have

$$S[i] = \sum_{j=0}^{N-1} X[i - j \times \Delta] . \tag{5.2}$$

If the clock frequency f_c is located in the k^{th} bin of X, $S[k]$ can be treated as the accumulated energy of all the sub-clocks, and it will reach a much higher value compared to any $S[i]$ where $i \neq k$, given the fact that sub-clocks coherently increase the power at the corresponding frequency locations.

Therefore, given a sequence of sampled values, it is divided into subsequences without overlap and each subsequence has L samples. (We should have $L \leq M$, and if $L < M$, it is expanded to M using zero-padding.) For the i^{th} subsequence, we calculate $S[k]_i$, and again the k^{th} bin in X contains the highest sub-clock's spectral content. The sequence $\{S[k]_0, S[k]_1, \cdots\}$ will be the one-dimensional *time series data* derived from the measured EM emanations. Here we use an example to demonstrate the effectiveness of using this technique to overcome the relatively low SNR issue. We use AMD RX 5700 and Linux OS in this example.

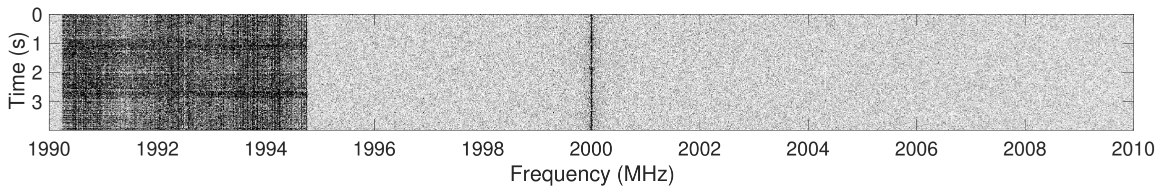


Figure 5.7: Spectrogram corresponding to a scenario where the periodic performance level change is invisible

We have RX 5700 in the all-metal computer case, and place it far away from the antenna (about 6 meters). We make a script which uses the AMD GPU driver to periodically switch the performance level between the lowest and the second lowest. Figure 5.7 shows the corresponding spectrogram. On the spectrogram, we cannot visibly find any patterns of stripe appearance. If we simply use such spectrograms for attacks like website fingerprinting, it is very unlikely that the attacks can be successfully mounted.

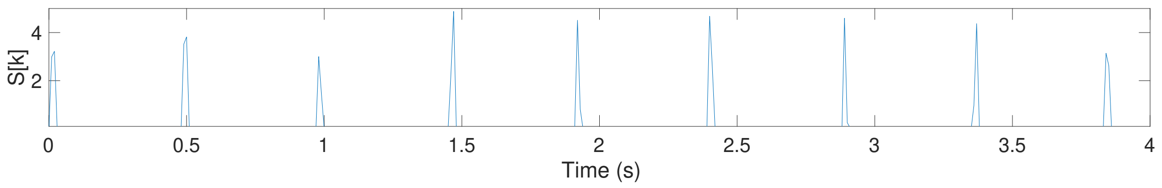


Figure 5.8: Derived time series in which peaks appear periodically and match the desired pattern

By contrast, Figure 5.8 shows the time series derived using the techniques described in Section 5.4. From the figure, we can clearly see the designed periodic peak appearance pattern.

Note that, although f_m is unknown to us, it can be exhaustively searched given the fact that its search space is small (30 to 33 kHz). An incorrect f_m will not produce noticeably high $S[k]$'s. Another unknown parameter is N, i.e., the number of sub-clocks. Nevertheless, we do not need to know the exact number. The GPU memory sub-clocks generated by SSC span at least 4 MHz, which means that there are at least 121 sub-clocks even when f_m is 33 kHz. Thus, the number 120, which is large enough to make $S[k]$ stand out, may be chosen as N if no other information is available.

Another issue is that although f_c is known and theoretically fixed, it may still vary in a small range due to clock skews; hence, if the DFT frequency resolution is fine-grained (e.g., in this chapter we use 100 Hz), the k^{th} bin computed directly from f_c may not be the one where the highest sub-clock truly locates. To address this problem, we compute multiple $S[i]$'s around k and update k to the one whose result is significantly larger than others.

5.4.2 Strong Noise Contamination Effect Reduction

The second lowest WCK frequency of all NVIDIA GPUs is 810 MHz, and their SSC sub-clocks are distributed in the 800 MHz – 810 MHz frequency band. In certain areas, this band may be too noisy for us. For instance, in North America, the Federal Communications Commission allocates the 614 MHz – 806 MHz frequency band for TV communication use. The contamination induced by such strong background noise makes it very difficult to find the correct sub-clock positions using the aforementioned technique. As an example, Figure 5.9 (A) shows a spectrum where communication signals exist, and we cannot rely on comparing different $S[i]$'s around the initial k to find where the highest sub-clock truly locates, because one significantly large noise peak can easily dwarf the sum of all the sub-clocks.

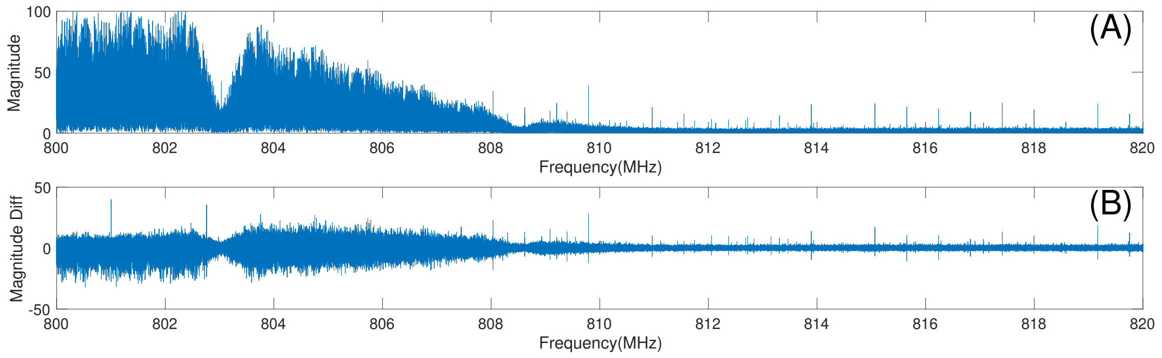


Figure 5.9: Spectrum comparison: (A) the original spectrum X, and (B) the derived spectrum X' after the proposed operation is performed

To address this issue, we propose to process the spectrum X using a convolution kernel $[-0.5,\ 1,\ -0.5]$. In other words, we derive X' from X as

$$X'[i] = -\frac{1}{2}X[i-1] + X[i] - \frac{1}{2}X[i+1]\,, \tag{5.3}$$

and replace X with X' in Equation 5.2. Note that, after this operation, the local peaks originally in X should have positive values in X'; otherwise, negative values. Thus, this operation will pinpoint all the local peaks which include (most of) the sub-clocks. Figure 5.9 (B) shows the spectrum after the operation is performed on the one in Figure 5.9 (A).

The reason why using X' can help reduce the negative effect of strong background noise on finding the correct sub-clock positions is that: (1) if the highest sub-clock is in the j^{th} bin, $S[j]$ will sum up the bins which are certainly dependent (as they correspond to sub-clocks), and thus it should be a positive value; but (2) if the highest sub-clock is not in the j^{th} bin, $S[j]$ will be the sum of bins which are independent, and given the fact that the kernel makes the expectation of randomly summing up X' bins be 0, $S[j]$ is very likely to be close to 0 in this case. Therefore, we can still find the correct sub-clock positions even in the presence of strong background noise.

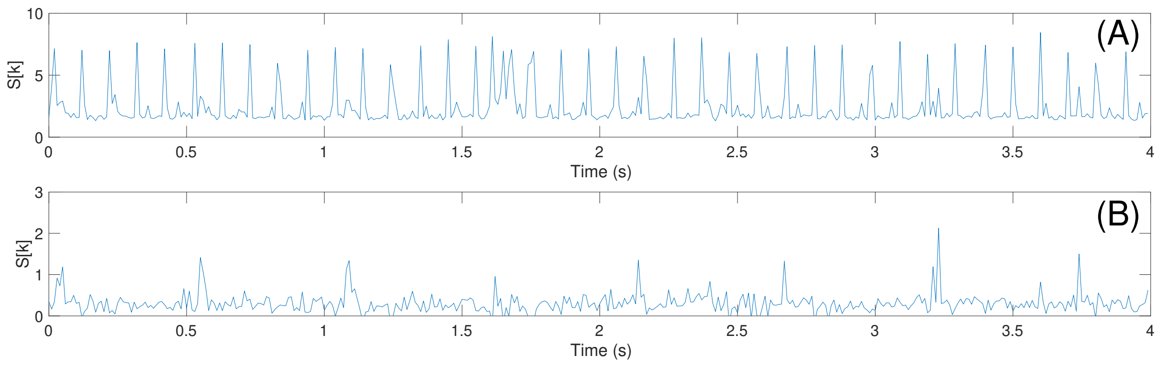


Figure 5.10: Time series comparison: (A) time series derived from X's, and (B) time series derived from X''s

Figure 5.10 demonstrates the effectiveness of our approach in the context of NVIDIA RTX 3060 being used with the existence of strong background noise as in Figure 5.9 (A). The EM signal of the second lowest WCK should appear around every 0.5 s, namely, there should be a peak in the derived time series about every 0.5 s. However, due to the strong noise contaminating the frequency band of our interest, the peak appearance pattern is completely incorrect in Figure 5.10 (A). After applying the proposed approach, we can observe the correct peak appearance pattern in Figure 5.10 (B).

5.5 Case Study 1: Website Fingerprinting

In Section 5.3, we have illustrated that GPU performance level switching patterns derived from the EM emanations of interest can be exploited to identify which application is being launched by a user. To further exemplify the exploitability of this DVFS-induced EM side-channel vulnerability, we show a case study in this section where an attacker can leverage this vulnerability to infer which webpages have been visited by a victim user, namely, to mount a website fingerprinting attack.

5.5.1 GPU-Accelerated Webpage Rendering

When browsing websites, the GPU is actually involved in a much more complicated fashion than simply showing pages on the screen. Modern web browsers such as Chrome and Firefox use GPU not only for displaying but also for helping webpage rendering.

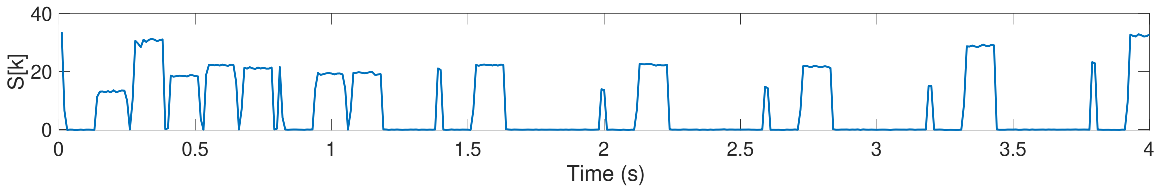
Webpage rendering is a procedure that translates an HTML file along with its associated cascading style sheets (CSS) and JavaScript code into a rasterized image. The whole procedure consists of multiple stages: it builds a document object model (DOM) tree, calculates the style for each DOM node, creates the layout of the page, separates the DOM-represented page into layers, rasterizes each layer, and combines the rasterized results into a final screen image Kosaka (2018). In such a complicated procedure, GPU is often leveraged to accelerate operations that involve large numbers of pixels. For example, a layer is normally divided into a grid of tiles, and these tiles need to be rasterized into bitmaps which are then uploaded to the GPU as textures. In the presence of GPU-accelerated rasterization, the GPU may be directly used to rasterize many tiles into textures, based on certain heuristics (e.g., if the tiles can be affected by animations or transition effects, it is better to employ the GPU). In addition, the GPU can be used to accelerate compositing textures into screen images.

5.5.2 EM-Based Website Fingerprinting

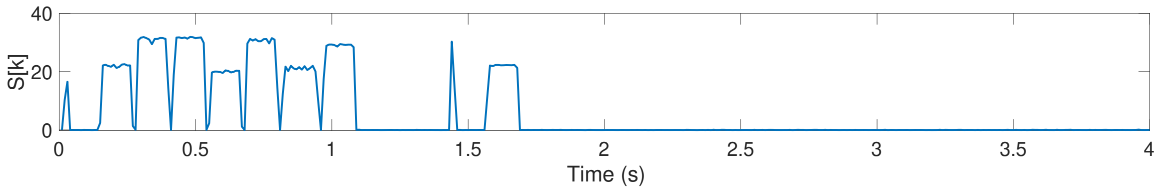
As stated above, GPU is extensively used during webpage rendering in a modern web browser. Since different webpages have different designs and contents, rendering them are likely to have different GPU workloads generated. In the light of the investigation presented in Section 5.3.3, such differences in workloads should be able to induce different patterns of GPU performance level switches, and these patterns can be captured approximately through monitoring the EM emanations of the GPU memory clock at a specific frequency. Exploiting such derived patterns, we should be able to distinguish the rendered webpages from each other (i.e., fingerprinting).

Specifically, we monitor the EM emanations generated by the WCK at its second lowest frequency and leverage the techniques described in Section 5.4 to derive time series to fingerprint webpage rendering activities. To illustrate this, we use three popular websites, Google, Amazon, and Youtube, as an example, and compare the time series derived from the signals captured when opening these three websites in Chrome on a system equipped with an AMD Radeon RX 580 GPU. Chrome uses GPU-accelerated webpage rendering by default. Accordingly, rendering the homepages of these three websites should create different GPU workloads, as their contents differ significantly (e.g., Google homepage is more concise, Amazon has more animations, and Youtube is populated with videos). Figure 5.11 shows the corresponding time series, and as expected, we can notice clear differences between them.

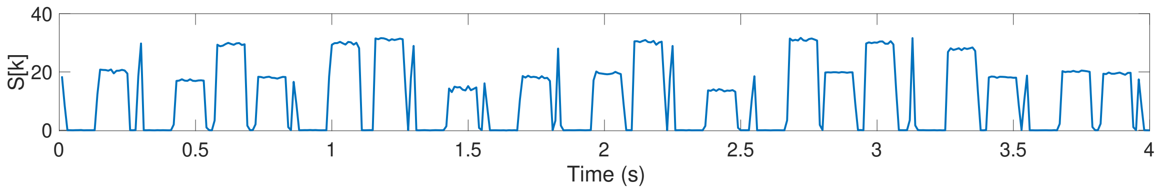
From Figure 5.11, we can observe that the peaks in the time series data appear very frequently corresponding to Youtube and Google. In terms of Youtube, when opened, it has some video being played, which will continuously employ the GPU for displaying (or even decoding); yet, we can still see the adjustment of



(a) Google



(b) Amazon



(c) Youtube

Figure 5.11: Time series derived from the EM emanations that are measured when opening three websites using Chrome on a system equipped with an AMD Radeon RX 580 GPU

the performance level due to GPU DVFS. In terms of Google, it has a blinking text cursor in its input box, whose blinking rate is about 1.67 Hz; and every time it blinks, the affected tiles need to be re-rasterized and screen image needs to be re-composited by the GPU. We can see that after the initial 1 s, the interval between each wide peak in Figure 5.11 (A) is about 600 ms, which matches the periodic blinking behavior of the cursor. Although not shown in Figure 5.11 (B), Amazon also induces periodic peak appearance in the time series data at about 0.2 Hz due to its animated advertisement pictures that are switched about every 5 s.

Notice that users normally tend to have multiple tabs opened in a browser. We find that using multiple tabs does **not** affect our website fingerprinting at all. The reason is that popular browsers like Chrome and Firefox only send the workloads of the currently focused tab to the GPU for optimizing resource utilization. To verify this, we use Chrome or Firefox to open a website in a tab while having several other tabs with YouTube playing videos, and we confirm that the EM signal pattern of interest is not disturbed by other unfocused tabs.

5.5.3 Evaluation

Because our main intention is to showcase the exploitation of the newly discovered EM side-channel, the evaluation is just performed in a closed-world scenario, where a victim is assumed to visit a list of popular websites and an attacker tries to pinpoint the webpages browsed by the victim from the set of possibilities. The open-world scenarios need novelty detection, which is left as our future work, where we may use some recently proposed techniques Ruff et al. (2018); Perera and Patel (2019).

We evaluate the EM-based website fingerprinting technique on all the GPUs listed in Table 5.1. The evaluation is focused on Chrome web browser, since it dominates the market share[8]. We try to use different operating systems, but we surprisingly find that all the AMD GPUs under Windows seldom change the performance level when opening a website. (This phenomenon should be caused by its driver, and we leave the further inspection to our future work.) Therefore, we evaluate the attack on AMD GPUs only under Linux, where the official driver AMDGPU 20.20 is used. In contrast, the attack can be mounted against all the NVIDIA GPUs under either Windows or Linux. We pair NVIDIA GTX 1080 with Linux where the official Linux driver 450.51.06 is installed, and pair NVIDIA GTX 1650 with Windows where the official Windows driver 461.40 is installed. For NVIDIA RTX 3060, we evaluate the attack under both Windows and Linux. Table 5.3 summarizes these circumstances.

Table 5.3: Feasibility of website fingerprinting attack under Windows and Linux

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
⊞	✗	✗	✗	✓	✓*	✓*
🐧	✓*	✓*	✓*	✓*	✓	✓*

The symbol ✓ indicates that website fingerprinting can be performed.
The symbol ✗ indicates that website fingerprinting cannot be performed.
The symbol * indicates that the evaluation is performed in this section.

We use an USRP B210 SDR and a RFSPACE UWB-3 antenna to capture the EM signals of interest, and we use the GNU Radio to manage the entire measurement process and process the captured raw data. The SDR is tuned to the second lowest WCK frequency of the corresponding GPU, and is set to use a 25 MHz sampling frequency[9].

Table 5.4: Different spots where EM signals are measured

Nearby Spots	Distance	Direction	Faraway Spots	Distance	Direction
N1	0.5 m	315°	F1	3 m	315°
N2	1 m	315°	F2	6 m	315°
N3	0.5 m	0°	F3	3 m	0°
N4	1 m	0°	F4	6 m	0°
N5	0.5 m	30°	F5	3 m	30°
N6	1 m	30°	F6	6 m	30°
N7	0.5 m	60°			
N8	1 m	60°			

We select 50 websites according to Alexa Top Sites, which are listed in Table 5.5.

[8]According to NetMarketShare, Chrome has around 70% browser market.
[9]Since quadrature sampling is used, it provides 25 MHz bandwidth.

Table 5.5: List of fingerprinted websites

9gag.com	abs-cbn.com	adobe.com	amazon.com	amazonaws.com	aol.com
apple.com	archive.org	ask.com	battle.net	bing.com	blogger.com
booking.com	businessinsider.com	cnn.com	deviantart.com	dictionary.com	discord.com
duckduckgo.com	ebay.com	espncricinfo.com	exoclick.com	facebook.com	feedly.com
foxnews.com	gamepedia.com	github.com	go.com	goodreads.com	google.com
imdb.com	linkedin.com	live.com	microsoft.com	msn.com	netflix.com
office.com	paypal.com	pinterest.com	reddit.com	roblox.com	stackoverflow.com
twitch.tv	twitter.com	whatsapp.com	wikipedia.org	yahoo.com	youtube.com
zillow.com	zoom.us				

For each website, we measure the EM signals from different directions (which are 315°, 0°, 30°, and 60°) at different distances (which are 0.5 m, 1 m, 3 m, and 6 m), as listed in Table 5.4. At each spot, we measure the EM emanations for 8 seconds when its webpage is opened, and we repeat this process for 50 times. For each measured signal, we use the techniques described in Section 5.4 to generate a time series. Given the 25 MHz sampling rate, we use $L = M = 250,000$, namely, each subsequence has 250,000 samples and a 250,000-point DFT is used, which means that the DFT bin resolution is 100 Hz and each derived point $S[k]$ represents 10 ms.

5.5.3.1 Nearby Scenario

We start with a nearby scenario, where only the EM signals measured at N1, N2, N3, and N4 are used to train a classification model, and the EM signals measured at N5, N6, N7, and N8 are used for testing. Given the fact that the EM signals have been transformed into time series, we adopt the ResNet model from Wang et al. (2017), whose architecture is duplicated in Figure 5.12. The details about the model can be found in Wang et al. (2017) and its code repository (https://github.com/cauchyturing/UCR_Time_Series_Classification_Deep_Learning_Baseline). (As for each website there are 50 EM signals measured at each spot, there are 200 time series for training and 200 time series for testing with respect to each website.)

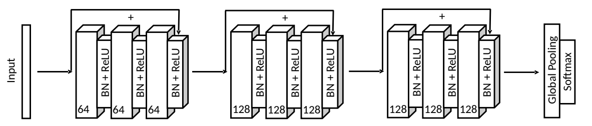


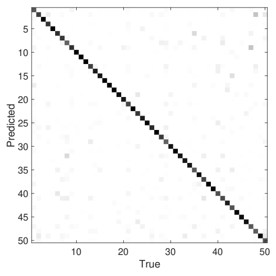
Figure 5.12: The neural network model used for our website fingerprinting attack (duplicated from Wang et al. (2017))

The evaluation results in terms of accuracy are shown in Table 5.6, and the confusion matrices are shown in Figure 5.13.

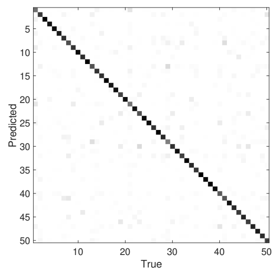
Note that, in reality, an attacker can profile the EM signals from any direction at any distance. Therefore, this scenario is *biased against* attackers, and the evaluation *underestimates* the achievable accuracy. Even so, from the results, we can see that the averaged classification accuracy is above 63% in all cases, and it

Table 5.6: Fingerprinting accuracy in the nearby scenario

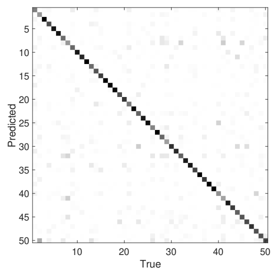
	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060_1	RTX 3060_2
N5	85.9%	83.3%	74.3%	81.9%	84.0%	80.7%	56.7%
N6	84.0%	86.0%	79.5%	88.2%	85.4%	61.3%	62.7%
N7	85.3%	82.6%	74.0%	73.3%	83.6%	72.4%	67.6%
N8	86.0%	83.2%	70.4%	72.4%	78.6%	70.0%	68.5%
Avg.	85.3%	83.8%	74.6%	79.0%	82.9%	71.1%	63.9%
Std.	0.8%	1.3%	3.2%	6.5%	2.6%	6.9%	4.7%



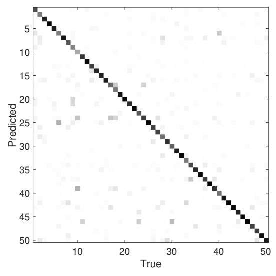
(a) AMD RX 580



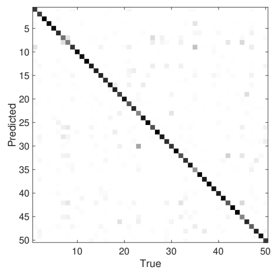
(b) AMD RX 5600



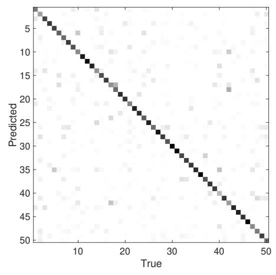
(c) AMD RX 5700



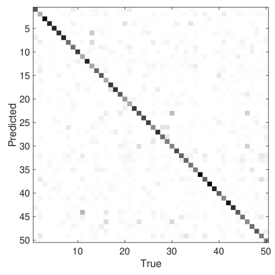
(d) NVIDIA GTX 1080



(e) NVIDIA GTX 1650



(f) NVIDIA RTX 3060^L



(g) NVIDIA RTX 3060^W

Figure 5.13: Confusion matrices corresponding to the evaluation reported in Table 5.6

reaches 85.3% in the case of RX 580. Given a test example, if we randomly guess which of the 50 websites it corresponds to, the accuracy will be only 2% (i.e., 1/50). Thus, the results signify that an abundant amount of information can be leaked through this new EM side channel.

An interesting case is to compare the results corresponding to using the same GPU but under different OSes. The last two columns of Table 5.6 show such a case, where NVIDIA RTX 3060 is evaluated under Windows (the last column) and Linux (the second last column). We can observe that, except for the anomalous spot N5, the performance for any other spot appears to be very similar, although it is slightly better under Linux. We also use the Linux-related model to classify the data captured under Windows and vice versa, but interestingly the accuracy is just slightly better than random guessing (4.3% and 5.6% respectively). This means that factors like drivers and Chrome engines for different OSes can strongly affect the GPU DVFS behavior when rendering webpages. Notice that, an attacker can simply perform profiling against both OSes and combine the training data, and thus we train a single model in such a manner to test the performance.

The resultant accuracy becomes 64.1%, which is very similar to the one in the last column (i.e., 63.9%). Therefore, this fingerprinting can work no matter which OS is used on the target.

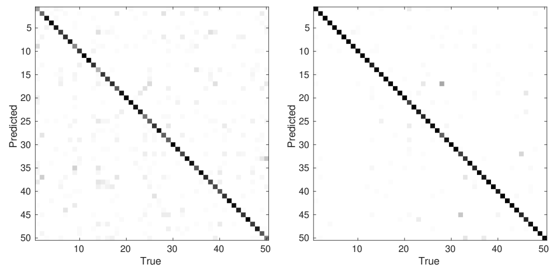
AMD RX 5600 and RX 5700 do differ but both of them are based on AMD RDNA 1.0 architecture. We attempt to use the models trained for them to cross fingerprint each other. The accuracy of using the model trained for RX 5600 to fingerprint the signals of RX 5700 is 63.3%, and the accuracy of using the model trained for RX 5700 to fingerprint the signals of RX 5600 is 50.6%. Even though they are made by different vendors and use different GPU chipsets, we can still obtain reasonable website fingerprinting results. This implies that an attacker can use the model trained with respect to his/her own GPU to fingerprint the signals of another similar GPU (of course, more accurately if two GPU chipsets also match).

5.5.3.2 Faraway Scenario

Next, we perform evaluations in a faraway scenario, where the EM signals measured at F1, F2, F3, and F4 are used to train a classification model, and the EM signals measured at F5 and F6 are used for testing. In addition, we test signals measured at N6 using this model. We evaluate two GPUs that are AMD RX 580 and NVIDIA GTX 1650. We place GTX 1650 in the modern-looking computer case, while we place RX 580 in the all-metal computer case. The evaluation results are shown under "Faraway Scenario 1" in Table 5.7, and the confusion matrices are shown in Figure 5.14.

Table 5.7: Fingerprinting accuracy in the faraway scenarios

	Faraway Scenario 1		Faraway Scenario 2	
	RX 580	GTX 1650	RX 580	GTX 1650
F5	80.1%	95.4%	83.6%	95.6%
F6	78.3%	93.2%	80.4%	94.1%
N6	70.9%	83.9%	87.0%	94.8%



(a) AMD RX 580 (b) NVIDIA GTX 1650

Figure 5.14: Confusion matrices corresponding to the evaluation reported in "Faraway Scenario 1" of Table 5.7

Similar to the previous nearby scenario, the evaluation also *underestimate* the achievable accuracy. Nevertheless, we can observe that the accuracy is very high, and in terms of NVIDIA GTX 1650, the resulting performance at far distances is even much better than that at near distances (e.g., it reaches 95.4% at 3 m and

93.2% at 6 m in comparison to 85.4% at 1 m). Moreover, in this scenario, training examples are neither from N6's direction nor around its distance, but the result *w.r.t.* N6 is comparable to that in Table 5.6 for GTX 1650 and fairly decent for RX 580. The results indicate that, as long as the EM signals of interest can be picked up, the differences in direction and distance are generally tolerable.

In the second faraway scenario, we include the EM signals measured at N2 and N4 for training as well, and still test the EM signals measured at F5, F6, and N6. The evaluation results are shown under "Faraway Scenario 2" in Table 5.7. We can observe that after expanding training examples with the ones measured at nearby spots N2 and N4, the accuracy *w.r.t.* faraway spots F5 or F6 does not change much, but it becomes much higher *w.r.t.* N6. The results indicate that, between the two profiling factors distance and direction, distance affects performance more. Thus, if resources are limited, an attacker should choose distance over direction during profiling.

5.5.3.3 Evaluations with Relaxed Limitations

We relax the limitations in the nearby scenario setting by allowing the training data set to include examples collected at all the other seven spots when testing each of the last four spots (i.e., spots N5, N6, N7, and N8). Table 5.8 shows the evaluation results. Compared with the results in Table 5.6, we can observe that the accuracy is increased by more than 10% in many cases (e.g., in terms of NVIDIA GTX 1080, the accuracy is increased by 11.1%, and in terms of NVIDIA RTX 3060, the accuracy is increased by 11.6% in the Linux case and 13.3% in the Windows case). Similarly, Table 5.9 shows the evaluation results when we relax the limitations in the first faraway scenario. Since an attacker can freely choose different spots for profiling at his/her own place, the accuracies reported here actually represent more pragmatic results.

Table 5.8: Fingerprinting accuracy in the relaxed nearby scenario (training examples are collected at all the other spots)

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060_1	RTX 3060_2
Avg.	93.0%	86.9%	80.6%	90.1%	89.9%	82.7%	77.2%
Std.	1.4%	0.7%	1.6%	3.1%	1.9%	3.9%	4.1%

Table 5.9: Fingerprinting accuracy in the relaxed faraway scenario (training examples are collected at all the other spots)

	RX 580	GTX 1650
Avg.	83.4%	95.7%
Std.	2.7%	1.1%

5.6 Case Study 2: Keystroke Timing Inference

In this section, we present the second case study we have conducted on the exploitation of the DVFS-induced EM side-channel vulnerability, which is to detect the keystroke events and learn the time between successive keystrokes, namely, a keystroke timing inference attack. Even though such an attack cannot directly recover the specific keys pressed by a user, it is still treated as a type of keylogging Monaco (2018), because the knowledge about the keystroke timing can be exploited to infer the typed passphrases or other words Song et al. (2001); Zhang and Wang (2009). Thus, this attack poses a greater hazard to security and privacy.

When combined with the website fingerprinting attack studied in the last section, it can even cause more serious violation of user privacy. For instance, when it is detected that a user has opened the login page of some website, the attacker can easily recognize the length of typed username and password through the number of identified keystrokes and further the attacker can infer the details of such items via the timing information using some well-studied statistical techniques Monaco (2018); Song et al. (2001); Zhang and Wang (2009).

5.6.1 Keystroke Detection

If we can detect the keystroke events and mark them precisely on the time axis, it will be a straightforward task to learn the time between successive keystrokes. Hence, we investigate if keystrokes are detectable from the EM emanations of the GPU memory clock, especially during the time when a user is typing on a webpage.

In essence, typing in a text box on a webpage makes the affected tiles of the corresponding layer re-rasterized and the final screen image re-composited. As previously mentioned, a browser often delegates the computation generated by these operations to GPU for acceleration. According to our earlier observations, the GPU performance level will be consequently changed by DVFS, and such level switches can be captured by monitoring the EM emanations of the GPU memory clock. Therefore, we expect that the keystroke events can be detected by exploiting the DVFS-induced EM side-channel vulnerability in modern GPUs.

To verify this hypothesis, we carry out several experiments. Firstly, we use keyboard activity generation tools to create a sequence of fake keystrokes regarding certain patterns and check if the appearance of the EM signals of the GPU memory clock match these patterns. Figure 5.15 shows an example of this experiment performed on NVIDIA GTX 1080, where we use a script to repeatedly generate a sequence of '*a*', '*b*', and '*c*' in the search box of Google. After each '*a*', there is a 200 ms pause; after each '*b*', there is a 350 ms pause; and, after each '*c*', there is a 500 ms pause. The time series in Figure 5.15 is derived using the techniques described in Section 5.4. From this figure, we can easily see that the appearance of peaks match the '*a*', '*b*', '*c*' keystroke timing pattern.

Next, we ask three people to quickly type "`username`" and "`password`" in the corresponding boxes on the Facebook login page. The keyboard is Dell L30U. Figure 5.16 illustrates the processed EM signals of interest in terms of NVIDIA RTX 3060 when the fastest typist among the participants is typing. From the results, we confirm that the keystroke events can be correctly detected and the inter-keystroke timing

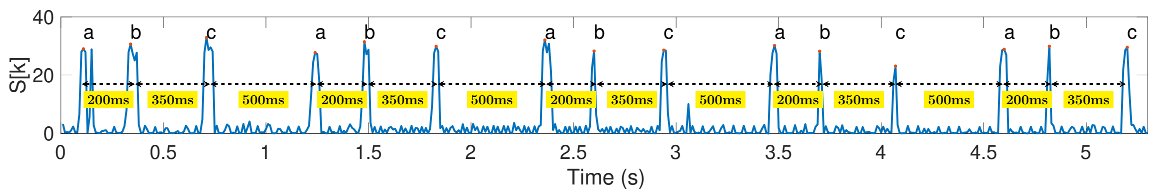


Figure 5.15: Keystrokes generated using xdotool with a predefined pattern on Google (performed *w.r.t.* NVIDIA GTX 1080)

information can be precisely derived. Notice that, even though we are able to determine the time between the keystrokes by exploiting the EM emanations of interest, we have not found any correlation between the signal and the typed characters.

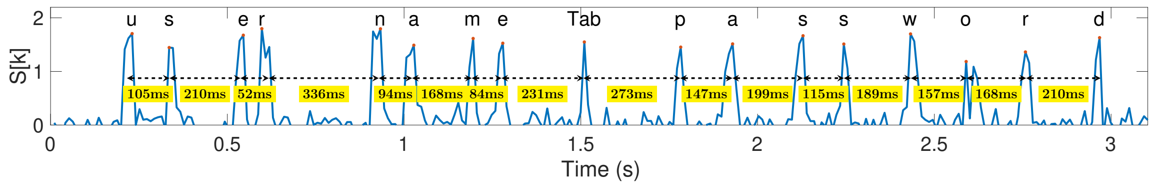


Figure 5.16: Keystrokes typed by a user on Facebook (performed *w.r.t.* NVIDIA RTX 3060)

5.6.2 Evaluation

In this evaluation, we mainly explore how close in time two successive keystrokes can be such that they are still detectable via exploiting the discovered DVFS-induced EM side-channel vulnerability. To facilitate the evaluation, we use the keyboard activity generation tools to create fake keystrokes, since they are certainly much more precise in time than manual inputs. As mentioned above, we focus on deriving the time between successive keystrokes when typing on webpages in a browser. Thus, the evaluation is primarily conducted in such a scenario. Like in Section 5.5, we exclusively use Chrome as the browser. We choose Google homepage and PayPal login page to be the representative venues for the evaluation. Currently, we do not use any automatic approach to identifying the keystroke events in the processed data but only perform a manual analysis.

Table 5.10: How close in time two keystrokes could be such that they are still distinguishable from each other

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
Interval	150 ms	N/A	N/A	50 ms	70 ms	30 ms

We create sequences of keystrokes with the inter-keystroke time interval being 10 ms, 20 ms, 30 ms, $\cdots$, respectively. We test each sequence on each GPU target machine listed in Table 5.1 to check if all the keystrokes in the sequence can be detected via the peak appearance and disappearance patterns in the derived time series. Table 5.10 shows the evaluation results.

From the results, we can see that NVIDIA RTX 3060 has the highest time resolution, where keystrokes at 30 ms intervals can be clearly recognized. (When it is lower than 30 ms, e.g., 20 ms, more than 90% keystrokes can also be recognized.) In terms of other NVIDIA GPUs, GTX 1080 and GTX 1650, a high resolution can also be achieved.[10]

Compared to NVIDIA GPUs, the timing resolution in terms of AMD RX 580 is much coarser, that is almost 150 ms. The reason for this discrepancy is that when RX 580 is at the second lowest performance level, it appears to stay there longer than those tested NVIDIA GPUs. Hence, if two or more keystrokes occur very closely in time, they can be treated as one keystroke event. (Nevertheless, a recent study on human typing behavior and performance has revealed that 250 ms inter-keystroke time interval is already very fast for many normal people Dhakal et al. (2018).) The interesting cases are AMD RX 5600 and RX 5700, on which we cannot mount the discussed attack. Even though their performance level changes correspondingly when a webpage is being rendered (as shown in Section 5.5), we find that this switching behavior seldom happens when typing on webpages. Therefore, we may not exploit the DVFS-induced EM side-channel vulnerability for this attack when AMD GPUs are used.

5.7 Ineffectiveness of Disabling GPU DVFS

To mitigate the aforementioned exploitation possibilities, a straightforward approach is to disable GPU DVFS by setting the GPU to run at a specific performance level. However, such a countermeasure has two major problems. The first one is that this countermeasure hurts either performance or energy efficiency. If a relatively low performance level is selected, it will contradict with the purpose of using the GPU for acceleration; yet, if a high performance level is selected, it will be highly energy-inefficient. The second and much severer problem is that, when NVIDIA GPUs are used, this countermeasure will unfortunately introduce another highly exploitable EM side-channel vulnerability.

5.7.1 AM-Modulated EM Emanations

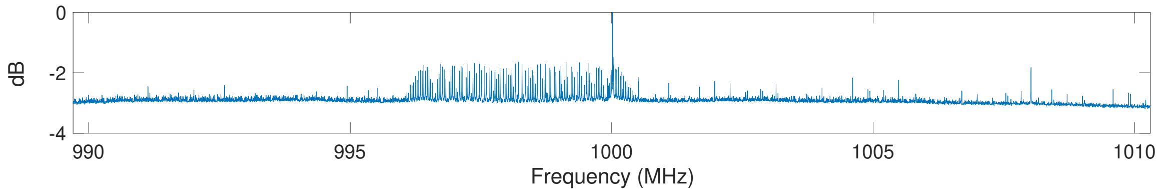
The reason for such ineffectiveness is that we have discovered a new type of exploitable EM emanations appearing when the performance level of an NVIDIA GPU is fixed. Given an NVIDIA GPU, a user may use tools included in the official driver to set its performance level to be maximum. (Unlike AMD GPUs whose driver allows us to fix the performance level at any defined one, the performance level of NVIDIA GPUs can only be fixed at the highest.) We find that when the performance level is fixed as such, there appears

[10]Additionally, we temporarily borrow two other NVIDIA GPUs, RTX 2060 Super and RTX 2080, from another group, and find that they behave the same as RTX 3060 and can reach 30 ms.

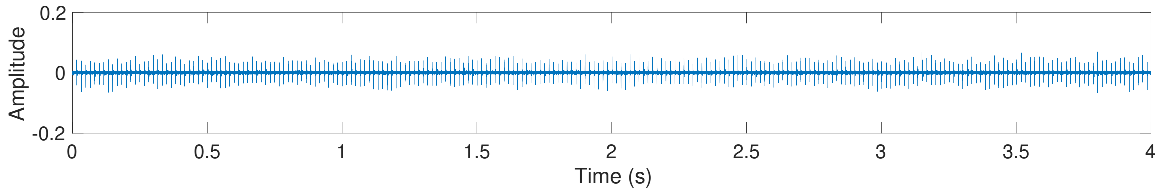
strong EM emanations that are inadvertently AM-modulated by the GPU memory accesses. In other words, the strength of these EM emanations varies when the amount of data reads and writes changes.

Interestingly, the EM emanations are around the frequency that is one eighth the data rate in the cases of all NVIDIA GPUs we have tested (i.e., GTX 1080, GTX 1650, and RTX 3060). Although we do not know the exact cause of such EM signals at the moment, an educated guess is that they are created by some clock driving certain components in the GPU memory system. We leave the search for this clock to our future work.

Since the emerging EM emanations will be AM-modulated by the GPU memory accesses, even though the DVFS-induced EM side-channel vulnerability were removed by using a fixed GPU memory clock frequency, information about the patterns of GPU memory traffic would be encoded into these new EM emanations, which can be exploited to effectively identify the high-level activities. Essentially, such EM emanations act as a modulated carrier signal that bears the modulating activity information and propagates to large distances.



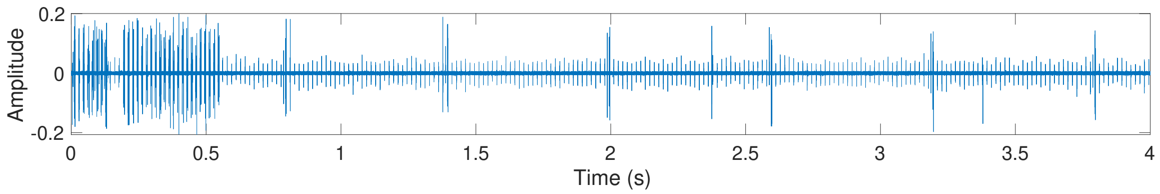
(a) Frequency-domain representation



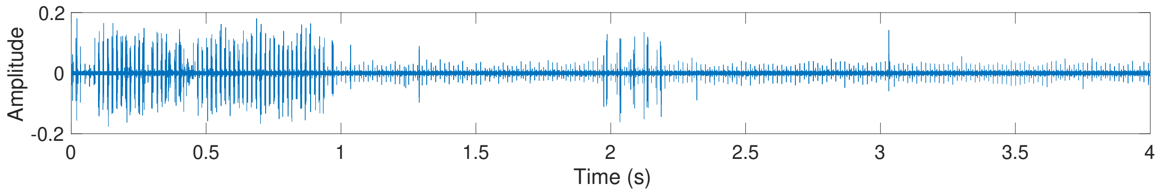
(b) Time-domain representation

Figure 5.17: Carrier signal emitted by NVIDIA GTX 1650 that can be AM-modulated by GPU memory accesses

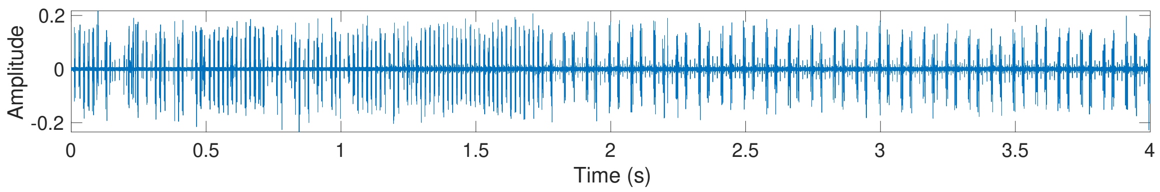
As an example, Figure 5.17 shows the above-mentioned carrier signal of interest that emerges when we set the performance level of NVIDIA GTX 1650 to its maximum. Given the 8 Gbps data rate of GTX 1650 at its maximum level, the EM carrier signal on which we focus will be at 1000 MHz. Yet, from Figure 5.17 (A), which illustrates the carrier signal of interest in the frequency-domain, we can observe that it has a number of spectral components in the frequency range from 996 MHz to 1000 MHz. Recall the discussion in Section 5.3.2 that SSC is used to vary the frequency of a clock over a range for meeting EMC regulations. These components spread over the 4 MHz range in Figure 5.17 (A) are caused by SSC, which indicates that the carrier signal is due to a clock. Figure 5.17 (B) shows the signal in the time-domain and its amplitude will change over time when AM-modulated.



(a) Google



(b) Amazon



(c) Youtube

Figure 5.18: The amplitude of carrier signal is modulated when opening three websites in Chrome

To exemplify that the EM signal shown in Figure 5.17 can be AM-modulated and exploitable, we examine the signal when browsing different websites using Chrome. Like the example demonstrated in Section 5.5.2, we use the three most popular websites, Google, Amazon, and Youtube. As the performance level is fixed this time, we cannot exploit its switching patterns to discern these websites. However, Figure 5.18 shows that the EM signal arising after the DVFS-induced EM side-channel vulnerability is mitigated can be exploited to infer the website identities, as its amplitude is modulated by the GPU memory accesses during webpage rendering, which should be different in general if different pages are being rendered. Comparing Figure 5.11 and Figure 5.18 with respect to the time axis, we can note resemblances. For instance, the 1.67 Hz blinking cursor behavior on Google search page is obvious in both cases. The AM-modulated signal of interest certainly carries information fine-grained enough for being exploited.

Notice that, when GPU DVFS is disabled, we have found this EM side-channel vulnerability only in NVIDIA GPUs but not in AMD GPUs. Nevertheless, it does not mean that disabling DVFS is a completely effective countermeasure in terms of AMD GPUs. There may be some other undiscovered EM vulnerabilities emerging when DVFS is disabled on AMD GPUs. We will discuss some possible countermeasures in the next section.

5.7.2 Evaluation and Discussion

To show the exploitability of such AM-modulated EM emanations, we evaluate the performance of website fingerprinting attack on NVIDIA RTX 3060 with Windows being used as the OS. The evaluation setting is exactly the same as the one used in Table 5.6, namely, the signals measured at N1, N2, N3, and N4 are used to train a classification model and the signals measured at N5, N6, N7, and N8 are used for testing. At each spot, it is still 50 signals being captured for each website.

The signal measurement setup is the same as that in Section 5.5.3 with one exception – we tune the SDR to center at 1875 MHz, as it is the frequency that is one eighth the corresponding data rate when we set the performance level of RTX 3060 to be the highest. We directly use the measured signal in the time-domain, as shown in Figure 5.18. Since it is also time series data, we still use the same ResNet classification model. The evaluation results are shown in Figure 5.19.

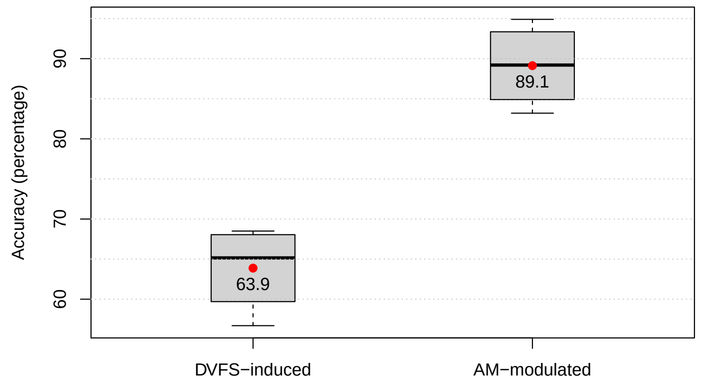


Figure 5.19: Website fingerprinting accuracy comparison of exploiting DVFS-induced and AM-modulated EM emanations from NVIDIA RTX 3060 with Windows being the OS

From the results, we can find that the accuracy is as high as 89.1%, which is much better than that exploiting the DVFS-induced EM emanations (i.e., 63.9%). Such good performance indicates that the AM-modulated EM emanations can be exploited to effectively mount website fingerprinting attacks. Even compared with the last column in Table 5.8 where data collected at the other seven spots is used for training, we can observe that the accuracy here is much higher. If considering only the fact that disabling GPU DVFS is actually used as a countermeasure, it would be unexpected to find that such a mitigation method helps rather than thwarts attacks.

We have also verified that we are able to perform keystroke timing inference by exploiting such AM-modulated EM signals. In this case, two keystrokes separated by 20 ms are still well distinguishable no matter which NVIDIA GPU is used.

Our conjecture about the considerably increased attack performance when leveraging the AM-modulated EM emanations is that the granularity of information carried in such EM signals is much finer than the

DVFS-induced ones. Although GPU DVFS is rapid enough to reveal changes in GPU workloads, it cannot enable us to pry more detailed activities inside each workload. In the specific case of website fingerprinting attack, if some finer-grained information can be acquired, it may help distinguish webpages that generate similar sequences of GPU workloads during their rendering. By contrast, the emanations whose strength is AM-modulated by GPU memory accesses inevitably contain information about the modulating memory activities of each workload. Thus, given the webpages which are often misclassified as each other when the DVFS-induced EM side-channel vulnerability is exploited, they become more discernible from each other when using the AM-modulated EM emanations.

Notice that, the EM side-channel vulnerability presented in this section and the DVFS-induced one cannot be exploited at the same time. We find that the AM-modulated vulnerability emerges only when the performance level is fixed. Therefore, although the AM-modulated EM side-channel vulnerability can be leveraged to achieve more effective attacks, it is the DVFS-induced one that will be exploited in many realistic situations, because, most of the time, a user will not change the settings on GPU DVFS which is active by default. However, as we have seen in this section, if the GPU DVFS is turned off, the other EM vulnerability becomes available for being exploited.

5.8 Countermeasures

As described in Section 5.7, the straightforward mitigation approach that disables GPU DVFS by fixing the performance level fails to effectively prevent information leakage from many GPUs. In this section, we propose several other countermeasures that we believe can potentially mitigate the EM side-channel vulnerabilities of GPU.

The first mitigation direction is to try to significantly lower the SNR of the exploitable EM emanations that an attacker can measure. To achieve this, we may have different options (e.g., using an RF device to generate some EM noise over the targeted frequency range), but a plausible and attainable method is to shield the computer for reducing the intensity of the emitted EM signals. For example, we may be tempted to tape some shielding Faraday fabric on the computer case sides. We have performed such experiments and found that, if we just naively rely on the patterns of stripe appearance on the spectrograms for exploitation, this can indeed make far-off attacks much harder or even impossible. *However, if we apply the techniques proposed in Section 5.4 first, we can still acquire the target signal's appearance/disappearance patterns even when the SNR becomes very low, which defeats the purpose of having EM shielding in such a simple way.* Therefore, more carefully engineered EM shielding computer cases are needed. Moreover, hardware manufacturers should devote more efforts to minimizing EM emissions in their future product designs, as replied by NVIDIA to us.

The second mitigation direction is to make the granularity of the EM side-channel information much coarser. For this purpose, we may try to reduce the sensitivity of GPU DVFS to workload changes. In such a case, the performance level may not be switched during activities like webpage rendering, and thus the leaked information will be coarse-grained so that very little sensitive information can be inferred. Although this method does not try to eliminate side-channel information, it can reduce the entropy of that side-channel and thus reduce the overall exploitability. Since this will hurt energy efficiency, a problem is how to find a good trade-off between security and power consumption.

In addition to reducing the SNR of the EM signals and increasing the granularity of the side-channel information, a third direction is through obfuscation. For instance, when a sensitive activity is using GPU, we may deliberately generate some random GPU workloads to disrupt its original GPU use pattern such that the predictability is reduced. With respect to the website fingerprinting attacks studied in this chapter, we may implement a thread in the browser to randomly use GPU for some dummy computation during webpage rendering via certain interfaces like WebGL. In terms of Chrome in particular, we may simply create an extension to send random requests to the GPU process which is the specific process in Chrome for managing interactions with GPU.

5.9 Conclusion

In this chapter, we have presented our newly discovered EM side-channel vulnerability that exists in many modern GPUs and conducted two case studies, website fingerprinting and keystroke timing attacks, to demonstrate that this new EM vulnerability is highly exploitable. Even though the root cause of this vulnerability is identified as the commonly used DVFS feature in GPU, we have shown that simply disabling DVFS by setting GPU to a specific performance level may not be an effective countermeasure since another AM-modulated EM vulnerability emerges. We have also discussed some potential mitigation approaches.

As research on information leakage vulnerabilities of GPU has just started lately, we believe that the currently disclosed ones and their exploitation represent only the tip of the iceberg, and many other exploitable ones lurk in the darkness. The study carried out in this chapter argues for more rigid evaluation on the security of GPU from different perspectives.

CHAPTER 6

Future Work

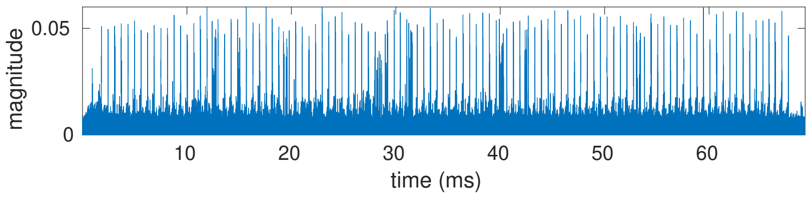
The growing research on deep learning nowadays prompts the use of optimal hardware for running neural network models. The GPU is found to perfectly fit the requirements of deep learning because of its high parallelism, so it is most frequently used for training and evaluating deep neural networks. Following the observation in Chapter 5 that the EM emanations from GPU are AM-modulated by ongoing computations, we find that this modulation effect will leak information about the running deep neural network model. In this section, we will present some preliminary experimental results to show how DNN running on GPU affects its EM emanations.

The EM traces are collected with a simple neural network model being evaluated on an NVIDIA Geforce RTX 2080 graphics card. The model consists of a normalization layer, a convolutional layer, and a fully connected layer, and random input values are used to repeat the evaluation 100 times. As mentioned above, DVFS is implemented in GPU to balance the power consumption and performance. In our experiments, we observe that the DVFS always shifts the performance level to the second-highest level whenever a neural network model is evaluated. Therefore, we fix the receiver's center frequency at the second-highest WCK frequency.

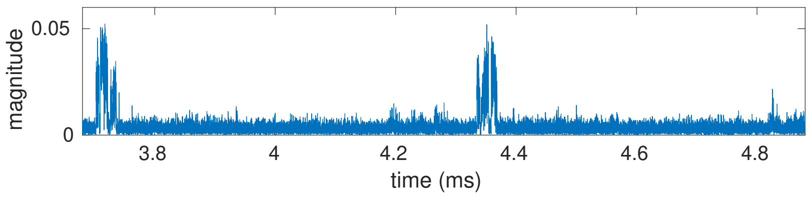
In Fig. 6.1 we plotted the magnitude of an EM trace. Then we zoom into different scales to examine the signals. From the figures we can find that:

- The periodical peaks repeating 100 times can be clearly identified in Fig. 6.1a. Using this pattern, the trace can be separated into 100 segments, each corresponding to one iteration of the DNN model evaluation.
- By examining the traces of two consecutive iterations in Fig. 6.1b, we can find that two segments are similar with respect to the peak width and duration. The magnitudes of peaks are similar but with a little difference.
- When we zoom in to examine only the peak part in Fig. 6.1c, the boundaries between three different layers are very obvious, based on which we can further separate each segment into three parts. Each part maps to a specific type of DNN layer.

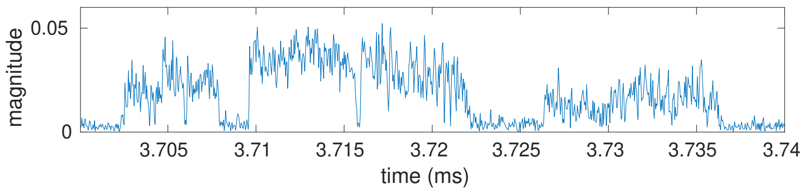
With the boundaries between iterations and layers correctly identified, we can select 100 traces mapped to it for each layer. In order to examine the relationship between the EM signals and the layer types, we



(a) Trace of 100 iterations



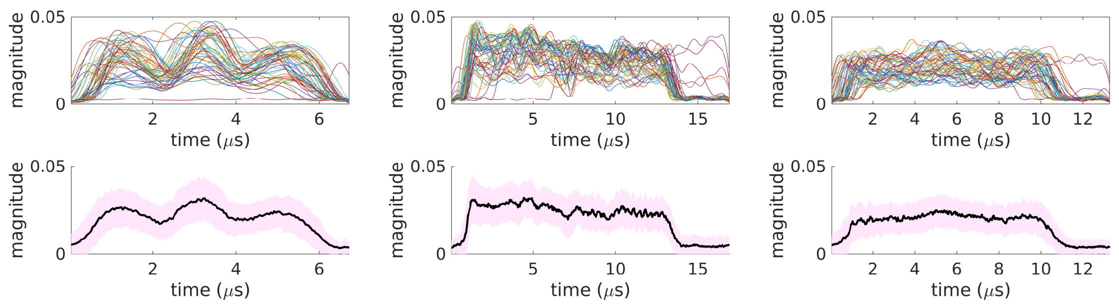
(b) Trace of 2 iterations



(c) A single peak in the trace

Figure 6.1: EM traces collected during neural network evaluation

perform some statistical analysis on these traces. The first step is aligning the traces, which is accomplished by looking for the position with maximum cross-correlation. Then we plot the aligned traces in Fig. 6.2, along with the averaged traces and the standard deviations below. The results showed in Fig. 6.1 indicate that signals mapped to the same type of layer share some features w.r.t both length and shape, while for different types of layers, the signals have some features that can be used to distinguish them from each other.



(a) Aligned traces of the normalization layer (b) Aligned traces of the convolutional layer (c) Aligned traces of the fully-connected layer

Figure 6.2: EM traces of different layers

According to the above preliminary experimental results, we have identified that EM emanations from GPU may leak information about the DNN model. Besides the results presented, we also observed that other parameters like convolution kernel size, node number, and layer connection could influence the EM emanations differently. In the future, we need to investigate deeper in order to fully understand the DNN model-related leakages from GPU's EM emanations.

CHAPTER 7

Conclusion

In this thesis, we have investigated the footprints left by rowhammer attacks in the EM emanations from DRAM clocks and provided a new direction for rowhammer attack detection. This novel rowhammer detection technique also provides new insight into the side-channel effects, showing that they can not only be exploited for attacks but can also help defend a system. We have studied the energy scattering effects caused by SSC and designed the first de-spreading technique to recover a strong EM carrier capable of carrying rich information of ongoing activities. The successful implementation of this de-spreading technique proves that some seemingly weak EM emanations from a computer system can actually be recovered to produce strong leakages with some proper technique. Furthermore, we have highlighted the significant improvement on SNR after applying the de-spreading technique by recovering a powerful EM covert channel carrier from the DRAM clock's EM emanations. The restored strong carrier can carry data using various modulation methods and construct a powerful covert channel capable of performing high-speed, through-wall, and long-range data exfiltration. This new EM covert channel increases the maximum physical covert channel bandwidth by 75$\times$ and breaks the illusion that physical covert channels can barely pose practical hazards. Henceforth, we believe that threats brought by physical covert channels are worth more attention. Besides side-channel-based defenses and physical covert channels, we have also implemented a long-range EM side-channel attack stealing information of content displayed on the screen. This is the first research exploiting far-field EM emanations from GPU to steal user privacy. Because this attack can be implemented from a certain distance away with obstacles (e.g., concrete walls) in between, very practical hazards can be posed by this leakage. The attacks presented in this work should urge manufactures to pay more attention to the EM leakages during the design and implementation of electronic components. All work presented in this thesis stressed the fact that many components in modern computers can generate strong EM emanations with abundant leakages. Even though these leakages were considered hard to exploit due to both the high complexity of systems and the massive amount of EM signals that interfere with each other, we have shown that sensitive information can be easily extracted with specially designed techniques. Therefore, both the industry and academia should pay more attention to these strong electromagnetic leakages in order to improve current hardware security.

References

Aarestad, J., Acharyya, D., Rad, R., and Plusquellic, J. (2010). Detecting trojans through leakage current analysis using multiple supply pad I_{ddq} s. *IEEE Transactions on information forensics and security*, 5(4):893–904.

Abdelrahman, Y., Khamis, M., Schneegass, S., and Alt, F. (2017). Stay cool! understanding thermal attacks on mobile-based user authentication. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3751–3763. ACM.

Abe, Y., Sasaki, H., Kato, S., Inoue, K., Edahiro, M., and Peres, M. (2014). Power and Performance Characterization and Modeling of GPU-Accelerated Systems. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 113–122. IEEE.

Aboulkassimi, D., Fournier, J., Freund, L., Robisson, B., and Tria, A. (2013). Ema as a physical method for extracting secret data from mobile phones. *International Journal of Computer Science and Application*, 2(1):16–25.

Aga, M. T., Aweke, Z. B., and Austin, T. (2017). 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, pages 8–13.

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

Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., and Sunar, B. (2007). Trojan detection using ic fingerprinting. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 296–310. IEEE.

Akkar, M.-L., Bevan, R., Dischamp, P., and Moyart, D. (2000). Power analysis, what is now possible... In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 489–502. Springer.

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

Andriotis, P., Tryfonas, T., Oikonomou, G., and Yildiz, C. (2013). A pilot study on the security of pattern screen-lock methods and soft side channel attacks. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 1–6. ACM.

Asonov, D. and Agrawal, R. (2004). Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11. IEEE.

Aweke, Z. B., Yitbarek, S. F., Qiao, R., Das, R., Hicks, M., Oren, Y., and Austin, T. (2016). 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, pages 743–755.

Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., and Sporleder, C. (2010). Acoustic Side-channel Attacks on Printers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security '10.

Batina, L., Bhasin, S., Jap, D., and Picek, S. (2019). CSINN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 515–532.

Becker, G. T., Kasper, M., Moradi, A., and Paar, C. (2010). Side-Channel Based Watermarks for Integrated Circuits. In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust*, HOST '10, pages 30–35.

Berger, Y., Wool, A., and Yeredor, A. (2006). Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 245–254. ACM.

Bertoni, G. M., Grassi, L., and Melzani, F. (2015). Simulations of optical emissions for attacking aes and masked aes. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 172–189. Springer.

Bhattacharya, S. and Mukhopadhyay, D. (2016). Curious case of rowhammer: flipping secret exponent bits using timing analysis. In *International Conference on Cryptographic Hardware and Embedded Systems*, CHES '16, pages 602–624.

Biham, E. and Shamir, A. (1999). Power analysis of the key scheduling of the aes candidates. In *Proceedings of the second AES Candidate Conference*, pages 115–121.

Bosman, E., Razavi, K., Bos, H., and Giuffrida, C. (2016). Dedup est machina: Memory deduplication as an advanced exploitation vector. In *2016 IEEE symposium on security and privacy*, S&P '16, pages 987–1004.

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

Brier, E., Clavier, C., and Olivier, F. (2004). Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer.

Brouchier, J., Dabbous, N., Kean, T., Marsh, C., and Naccache, D. (2009). Thermocommunication. *IACR Cryptology ePrint Archive*, 2009:2.

Callan, R., Behrang, F., Zajic, A., Prvulovic, M., and Orso, A. (2016). Zero-overhead Profiling via EM Emanations. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, pages 401–412.

Callan, R., Zajić, A., and Prvulovic, M. (2014). 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, pages 242–254.

Callan, R., Zajić, A., and Prvulovic, M. (2015). FASE: Finding Amplitude-modulated Side-channel Emanations. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 592–603.

Carlier, V., Chabanne, H., Dottax, E., and Pelletier, H. (2004). Electromagnetic side channels of an fpga implementation of aes. In *CRYPTOLOGY EPRINT ARCHIVE, REPORT 2004/145*. Citeseer.

Carmon, E., Seifert, J.-P., and Wool, A. (2016). Simple photonic emission attack with reduced data complexity. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 35–51. Springer.

Carmon, E., Seifert, J.-P., and Wool, A. (2017). Photonic side channel attacks against rsa. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 74–78. IEEE.

Carrara, B. and Adams, C. (2014). On acoustic covert channels between air-gapped systems. In *International Symposium on Foundations and Practice of Security*, pages 3–16. Springer.

Chari, S., Rao, J. R., and Rohatgi, P. (2002). Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer.

Chen, X., Xu, Z., Kim, H., Gratz, P., Hu, J., Kishinevsky, M., and Ogras, U. (2013). In-Network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches. *ACM Transactions on Design Automation of Electronic Systems*, 18(4).

Cheng, Y., Ji, X., Zhang, J., Xu, W., and Chen, Y.-C. (2019). Demicpu: Device fingerprinting with magnetic signals radiated by cpu. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1149–1170. ACM.

Cheng, Y., Zhang, Z., Nepal, S., and Wang, Z. (2018). Still Hammerable and Exploitable: on the Effectiveness of Software-only Physical Kernel Isolation. *CoRR*, abs/1802.07060.

Cho, K.-T. and Shin, K. G. (2017). Viden: Attacker Identification on In-Vehicle Networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1109–1123.

Clark, S. S., Ransford, B., Rahmati, A., Guineau, S., Sorber, J., Xu, W., and Fu, K. (2013). Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *Presented as part of the 2013 USENIX Workshop on Health Information Technologies*.

Clavier, C., Coron, J.-S., and Dabbous, N. (2000). Differential power analysis in the presence of hardware countermeasures. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 252–263. Springer.

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

Corbet, J. (2016). Defending against Rowhammer in the kernel.

Coron, J.-S., Kocher, P., and Naccache, D. (2000). Statistics and secret leakage. In *International Conference on Financial Cryptography*, pages 157–173. Springer.

Davidov, M. and Oldenburg, B. (2020). TEMPEST@Home - Finding Radio Frequency Side Channels. https://duo.com/labs/research/finding-radio-sidechannels#section9.

De Mulder, E., Buysschaert, P., Ors, S., Delmotte, P., Preneel, B., Vandenbosch, G., and Verbauwhede, I. (2005). Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem. In *EUROCON 2005-The International Conference on" Computer as a Tool"*, volume 2, pages 1879–1882. IEEE.

Deng, Q., Meisner, D., Ramos, L., Wenisch, T. F., and Bianchini, R. (2011). MemScale: Active Low-Power Modes for Main Memory. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 225–238.

Departments and agencies of the Federal Government (2019). Code of federal regulations.

Dhakal, V., Feit, A. M., Kristensson, P. O., and Oulasvirta, A. (2018). Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Di-Battista, J., Courrege, J.-C., Rouzeyre, B., Torres, L., and Perdu, P. (2010). When failure analysis meets side-channel attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 188–202. Springer.

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

Ferrigno, J. and Hlaváč, M. (2008). When aes blinks: introducing optical side channel. *IET Information Security*, 2(3):94–98.

Frigo, P., Giuffrida, C., Bos, H., and Razavi, K. (2018). Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In *2018 IEEE Symposium on Security and Privacy*, S&P '18, pages 357–372.

Gandolfi, K., Mourtel, C., and Olivier, F. (2001). Electromagnetic Analysis: Concrete Results. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, pages 251–261.

Gao, Y., Zhang, H., Cheng, W., Zhou, Y., and Cao, Y. (2018). Electro-Magnetic Analysis of GPU-Based AES Implementation. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*.

Ge, Q., Yarom, Y., Cock, D., and Heiser, G. (2018). A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27.

Genkin, D., Pachmanov, L., Pipman, I., and Tromer, E. (2015a). Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. In *International Conference on Cryptographic Hardware and Embedded Systems*, CHES 2015, pages 207–228.

Genkin, D., Pachmanov, L., Pipman, I., and Tromer, E. (2016a). Ecdh key-extraction via low-bandwidth electromagnetic attacks on pcs. In *Cryptographers' Track at the RSA Conference*, pages 219–235. Springer.

Genkin, D., Pachmanov, L., Pipman, I., Tromer, E., and Yarom, Y. (2016b). 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, pages 1626–1638.

Genkin, D., Pipman, I., and Tromer, E. (2015b). Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. *Journal of Cryptographic Engineering*, 5(2):95–112.

Genkin, D., Shamir, A., and Tromer, E. (2014). Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *Annual Cryptology Conference*, pages 444–461. Springer.

Genkin, D., Shamir, A., and Tromer, E. (2017). Acoustic Cryptanalysis. *Journal of Cryptology*, 30(2):392–443.

Gilbert Goodwill, B. J., Jaffe, J., Rohatgi, P., et al. (2011). A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136.

Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W., and Yarom, Y. (2018). Another Flip in the Wall of Rowhammer Defenses. In *2018 IEEE Symposium on Security and Privacy*, S&P '18, pages 489–505.

Gruss, D., Maurice, C., and Mangard, S. (2016a). Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '16, pages 300–321.

Gruss, D., Maurice, C., Wagner, K., and Mangard, S. (2016b). 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, pages 279–299.

Guri, M. (2019). Hotspot: Crossing the air-gap between isolated pcs and nearby smartphones using temperature. In *2019 European Intelligence and Security Informatics Conference (EISIC)*, pages 94–100. IEEE.

Guri, M. (2020). Power-supplay: Leaking data from air-gapped systems by turning the power-supplies into speakers. *arXiv preprint arXiv:2005.00395*.

Guri, M., Bykhovsky, D., and Elovici, Y. (2019). Brightness: Leaking sensitive data from air-gapped workstations via screen brightness. In *2019 12th CMI Conference on Cybersecurity and Privacy (CMI)*, pages 1–6. IEEE.

Guri, M., Daidakulov, A., and Elovici, Y. (2018a). Magneto: Covert channel between air-gapped systems and nearby smartphones via cpu-generated magnetic fields. *arXiv preprint arXiv:1802.02317*.

Guri, M., Hasson, O., Kedma, G., and Elovici, Y. (2016a). An optical covert-channel to leak data through an air-gap. In *2016 14th Annual Conference on Privacy, Security and Trust (PST '16)*, pages 642–649.

Guri, M., Kachlon, A., Hasson, O., Kedma, G., Mirsky, Y., and Elovici, Y. (2015a). GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 849–864.

Guri, M., Kedma, G., Kachlon, A., and Elovici, Y. (2014). Airhopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies. In *2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*, pages 58–67. IEEE.

Guri, M., Monitz, M., and Elovici, Y. (2016b). Usbee: air-gap covert-channel via electromagnetic emission from usb. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 264–268. IEEE.

Guri, M., Monitz, M., Mirski, Y., and Elovici, Y. (2015b). Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 276–289. IEEE.

Guri, M., Solewicz, Y., Daidakulov, A., and Elovici, Y. (2017a). Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise ('diskfiltration'). In *European Symposium on Research in Computer Security*, pages 98–115. Springer.

Guri, M., Solewicz, Y., and Elovici, Y. (2020). Fansmitter: Acoustic data exfiltration from air-gapped computers via fans noise. *Computers & Security*, 91:101721.

Guri, M., Zadov, B., Bykhovsky, D., and Elovici, Y. (2018b). Powerhammer: Exfiltrating data from air-gapped computers through power lines. *arXiv preprint arXiv:1804.04014*.

Guri, M., Zadov, B., Daidakulov, A., and Elovici, Y. (2018c). Odini: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields. *arXiv preprint arXiv:1802.02700*.

Guri, M., Zadov, B., and Elovici, Y. (2017b). Led-it-go: Leaking (a lot of) data from air-gapped computers via the (small) hard drive led. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 161–184. Springer.

Han, Y., Etigowni, S., Liu, H., Zonouz, S., and Petropulu, A. (2017). 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, pages 1095–1108.

Hanspach, M. and Goetz, M. (2013). On covert acoustical mesh networks in air. *Journal of Communications*, 8(11).

Hardin, K. B., Fessler, J. T., and Bush, D. R. (1994). Spread Spectrum Clock Generation for the Reduction of Radiated Emissions. In *Proceedings of IEEE Symposium on Electromagnetic Compatibility*, EMC '94, pages 227–231.

Hassan, M., Kaushik, A. M., and Patel, H. (2015). Reverse-engineering embedded memory controllers through latency-based analysis. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '15)*, pages 297–306.

He, J., Zhao, Y., Guo, X., and Jin, Y. (2017). Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10):2939–2948.

Herath, N. and Fogh, A. (2015). These are Not Your Grand Daddy's CPU Performance Counters - CPU Hardware Performance Counters for Security. In *Black Hat Briefings*.

Heyszl, J., Mangard, S., Heinz, B., Stumpf, F., and Sigl, G. (2012). Localized electromagnetic analysis of cryptographic implementations. In *Cryptographers' track at the RSA conference*, pages 231–244. Springer.

Hoque, T., Narasimhan, S., Wang, X., Mal-Sarkar, S., and Bhunia, S. (2017). Golden-free hardware trojan detection with high sensitivity under process noise. *Journal of Electronic Testing*, 33(1):107–124.

Hutter, M. and Schmidt, J.-M. (2013). The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 219–235. Springer.

Irazoqui, G., Eisenbarth, T., and Sunar, B. (2018). MASCAT: Preventing Microarchitectural Attacks Before Distribution. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, pages 377–388.

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

Jiang, Z. H., Fei, Y., and Kaeli, D. (2016). A Complete Key Recovery Timing Attack on a GPU. In *2016 IEEE International symposium on high performance computer architecture (HPCA)*, pages 394–405.

Jiang, Z. H., Fei, Y., and Kaeli, D. (2019). Exploiting Bank Conflict-Based Side-Channel Timing Leakage of GPUs. *ACM Transactions on Architecture and Code Optimization*, 16(4).

Kaczmarek, T., Ozturk, E., and Tsudik, G. (2018). Thermanator: Thermal residue-based post factum attacks on keyboard password entry. *arXiv preprint arXiv:1806.10189*.

Kim, W., Gupta, M. S., Wei, G.-Y., and Brooks, D. (2008). System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture (HPCA)*, pages 123–134.

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J. H., Lee, D., Wilkerson, C., Lai, K., and Mutlu, O. (2014). 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, pages 361–372.

Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., et al. (2019). Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19. IEEE.

Kocher, P. C., Jaffe, J., and Jun, B. (1999). Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397.

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

Kosaka, M. (2018). Inside Look at Modern Web Browser.

Krämer, J., Nedospasov, D., Schlösser, A., and Seifert, J.-P. (2013). Differential photonic emission analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 1–16. Springer.

Kuhn, M. G. (2004). Electromagnetic Eavesdropping Risks of Flat-Panel Displays. In *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*, PET '04, pages 88–107.

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

Kwong, A., Xu, W., and Fu, K. (2019). Hard drive of hearing: Disks that eavesdrop with a synthesized microphone. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 905–919.

Ladakis, E., Koromilas, L., Vasiliadis, G., Polychronakis, M., and Ioannidis, S. (2013). You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger. In *Proceedings of the 6th European Workshop on System Security (EuroSec)*.

Lampson, B. W. (1973). A note on the confinement problem. *Communications of the ACM*, 16(10):613–615.

Lanteigne, M. (2016). How Rowhammer Could Be Used to Exploit Weaknesses in Computer Hardware.

Le Masle, A. and Luk, W. (2012). Detecting power attacks on reconfigurable hardware. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pages 14–19. IEEE.

Lee, S., Kim, Y., Kim, J., and Kim, J. (2014). Stealing Webpages Rendered on Your Browser by Exploiting GPU Vulnerabilities. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy (S&P)*, pages 19–33.

Lenovo Inc. (2015). Row Hammer Privilege Escalation Lenovo Security Advisory (LEN-2015-009).

Lime Microsystems (2021). LimeSDR.

Lipp, M., Aga, M. T., Schwarz, M., Gruss, D., Maurice, C., Raab, L., and Lamster, L. (2018a). Nethammer: Inducing Rowhammer Faults through Network Requests. *CoRR*, abs/1805.04956.

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., et al. (2018b). Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 973–990.

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

Longo, J., De Mulder, E., Page, D., and Tunstall, M. (2015). Soc it to em: electromagnetic side-channel attacks on a complex system-on-chip. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 620–640. Springer.

Lopes, A. C. and Aranha, D. F. (2017). Platform-agnostic low-intrusion optical data exfiltration. In *ICISSP*, pages 474–480.

Loughry, J. and Umphress, D. A. (2002). Information leakage from optical emanations. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):262–289.

Luo, C., Fei, Y., and Kaeli, D. (2019). Side-Channel Timing Attack of RSA on a GPU. *ACM Transactions on Architecture and Code Optimization*, 16(3).

Luo, C., Fei, Y., Luo, P., Mukherjee, S., and Kaeli, D. (2015). Side-Channel Power Analysis of a GPU AES Implementation. In *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD)*, pages 281–288.

Ma, K., Li, X., Chen, W., Zhang, C., and Wang, X. (2012). GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures. In *Proceedings of the 2012 41st International Conference on Parallel Processing (ICPP)*, pages 48–57.

Marquardt, P., Verma, A., Carter, H., and Traynor, P. (2011). (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562. ACM.

Masti, R. J., Rai, D., Ranganathan, A., Müller, C., Thiele, L., and Capkun, S. (2015). Thermal covert channels on multi-core platforms. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 865–880.

Matyunin, N., Szefer, J., Biedermann, S., and Katzenbeisser, S. (2016). Covert channels using mobile device's magnetic field sensors. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 525–532. IEEE.

Maurice, C., Neumann, C., Heen, O., and Francillon, A. (2014). Confidentiality Issues on a GPU in a Virtualized Environment. In *International Conference on Financial Cryptography and Data Security (FC)*, pages 119–135.

Maurice, C., Neumann, C., Heen, O., and Francillon, A. (2015). C5: cross-cores cache covert channel. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '15)*, pages 46–64.

Mayer-Sommer, R. (2000). Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 78–92. Springer.

Mei, X., Yung, L. S., Zhao, K., and Chu, X. (2013). A Measurement Study of GPU DVFS on Energy Conservation. In *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower)*.

Merli, D., Heyszl, J., Heinz, B., Schuster, D., Stumpf, F., and Sigl, G. (2013). Localized electromagnetic analysis of ro pufs. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 19–24. IEEE.

Merli, D., Schuster, D., Stumpf, F., and Sigl, G. (2011). Semi-invasive em attack on fpga ro pufs and countermeasures. In *Proceedings of the Workshop on Embedded Systems Security*, page 2. ACM.

Messerges, T. S. (2000). Using second-order power analysis to attack dpa resistant software. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 238–251. Springer.

Messerges, T. S., Dabbish, E. A., and Sloan, R. H. (1999). Investigations of power analysis attacks on smartcards. *Smartcard*, 99:151–161.

Micron Technology, Inc. (2014). GDDR5 SGRAM Introduction.

Micron Technology, Inc. (2017). GDDR6: The Next-Generation Graphics DRAM.

Mishra, A. K., Das, R., Eachempati, S., Iyer, R., Vijaykrishnan, N., and Das, C. R. (2009). A Case for Dynamic Frequency Tuning in On-Chip Networks. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 292–303.

Miyoshi, A., Lefurgy, C., Van Hensbergen, E., Rajamony, R., and Rajkumar, R. (2002). Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling. In *Proceedings of the 16th International Conference on Supercomputing (ICS)*, pages 35–44.

Monaco, J. V. (2018). SoK: Keylogging Side Channels. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P)*, pages 211–228.

Mowery, K., Meiklejohn, S., and Savage, S. (2011). Heat of the moment: Characterizing the efficacy of thermal camera-based attacks. In *Proceedings of the 5th USENIX conference on Offensive technologies*, pages 6–6. USENIX Association.

Mutlu, O. (2017). 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, pages 1116–1121.

Naghibijouybari, H., Khasawneh, K. N., and Abu-Ghazaleh, N. (2017). Constructing and Characterizing Covert Channels on GPGPUs. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 354–366.

Naghibijouybari, H., Neupane, A., Qian, Z., and Abu-Ghazaleh, N. (2018). Rendered Insecure: GPU Side Channel Attacks Are Practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2139–2153.

Narasimhan, S., Wang, X., Du, D., Chakraborty, R. S., and Bhunia, S. (2011). Tesr: A robust temporal self-referencing approach for hardware trojan detection. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 71–74. IEEE.

Nath, R. and Tullsen, D. (2015). The CRISP Performance Model for Dynamic Voltage and Frequency Scaling in a GPGPU. In *Proceedings of the 48th International Symposium on Microarchitecture (MICRO)*, pages 281–293.

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

Newegg (2021). Best Selling Computer Cases. https://www.newegg.com/d/Best-Sellers/Computer-Cases/s/ID-7.

Nguyen, L. N., Yilmaz, B. B., Prvulovic, M., and Zajic, A. (2020). A novel golden-chip-free clustering technique using backscattering side channel for hardware trojan detection. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 1–12. IEEE.

Pant, S. (2008). *Design and Analysis of Power Distribution Networks in VLSI Circuits.* PhD thesis, University of Michigan.

Payer, M. (2016). HexPADS: A Platform to Detect "Stealth" Attacks. In *Proceedings of the 8th International Symposium on Engineering Secure Software and Systems*, ESSoS 2016, pages 138–154.

Perera, P. and Patel, V. M. (2019). Deep Transfer Learning for Multiple Class Novelty Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

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

Pietro, R. D., Lombardi, F., and Villani, A. (2016). CUDA Leaks: A Detailed Hack for CUDA and a (Partial) Fix. *ACM Transactions on Embedded Computing Systems*, 15(1).

Prvulovic, M., Zajić, A., Callan, R. L., and Wang, C. J. (2017). A Method for Finding Frequency-Modulated and Amplitude-Modulated Electromagnetic Emanations in Computer Systems. *IEEE Transactions on Electromagnetic Compatibility*, 59(1):34–42.

Qiao, R. and Seaborn, M. (2016). A new approach for rowhammer attacks. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust*, HOST '16, pages 161–166.

Quisquater, J.-J. and Samyde, D. (2001). 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, pages 200–210.

Rad, R., Plusquellic, J., and Tehranipoor, M. (2008). Sensitivity analysis to hardware trojans using power supply transient signals. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 3–7. IEEE.

Rao, J. R. and Rohatgi, P. (2001). Empowering side-channel attacks. *IACR Cryptology ePrint Archive*, 2001:37.

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

Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security (CCS '09)*, pages 199–212. ACM.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 4393–4402.

Sauvage, L., Guilley, S., Flament, F., Danger, J.-L., and Mathieu, Y. (2010). Cross-correlation cartography. In *2010 International Conference on Reconfigurable Computing and FPGAs*, pages 268–273. IEEE.

Sauvage, L., Guilley, S., and Mathieu, Y. (2009). Electromagnetic radiations of fpgas: High spatial resolution cartography and attack on a cryptographic module. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(1):4.

Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S., and Seifert, J.-P. (2012). 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, pages 41–57.

Schwarz, M., Weiser, S., Gruss, D., Maurice, C., and Mangard, S. (2017). Malware Guard Extension: Using SGX to Conceal Cache Attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '17, pages 3–24. Springer.

Seaborn, M. and Dullien, T. (2015). Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges. In *Black Hat Briefings*.

Sehatbakhsh, N., Nazari, A., Zajic, A., and Prvulovic, M. (2016). Spectral Profiling: Observer-effect-free Profiling by Monitoring EM Emanations. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49, pages 59:1–59:11.

Sehatbakhsh, N., Yilmaz, B. B., Zajic, A., and Prvulovic, M. (2020). A New Side-Channel Vulnerability on Modern Computers by Exploiting Electromagnetic Emanations from the Power Management Unit. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 123–138.

Sepetnitsky, V., Guri, M., and Elovici, Y. (2014). Exfiltration of information from air-gapped machines using monitor's led indicator. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 264–267. IEEE.

Shen, C., Liu, T., Huang, J., and Tan, R. (2021). When LoRa Meets EMR: Electromagnetic Covert Channels Can Be Super Resilient. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (S&P)*.

Skorobogatov, S. (2009). Using optical emission analysis for estimating contribution to power analysis. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 111–119. IEEE.

Song, D. X., Wagner, D., and Tian, X. (2001). Timing Analysis of Keystrokes and Timing Attacks on SSH. In *USENIX Security Symposium (USENIX Security 01)*.

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

Su, Y., Genkin, D., Ranasinghe, D., and Yarom, Y. (2017). USB snooping made easy: Crosstalk leakage attacks on USB hubs. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1145–1161.

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

Suh, G. E. and Devadas, S. (2007). Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14. IEEE.

Sullivan, D., Arias, O., Meade, T., and Jin, Y. (2018). Microarchitectural minefields: 4k-aliasing covert channel and multi-tenant detection in iaas clouds. In *NDSS '18*.

Szefer, J. (2019). Survey of microarchitectural side and covert channels, attacks, and defenses. *Journal of Hardware and Systems Security*, 3(3):219–234.

Tatar, A., Giuffrida, C., Bos, H., and Razavi, K. (2018a). Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer. In *Research in Attacks, Intrusions, and Defenses*, RAID '18, pages 47–66.

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

Tsang, J. C., Kash, J. A., and Vallett, D. P. (2000). Picosecond imaging circuit analysis. *IBM Journal of Research and Development*, 44(4):583–603.

van der Veen, V., Fratantonio, Y., Lindorfer, M., Gruss, D., Maurice, C., Vigna, G., Bos, H., Razavi, K., and Giuffrida, C. (2016). Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1675–1689.

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

Van Eck, W. (1985). Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269–286.

Vasiliadis, G., Polychronakis, M., and Ioannidis, S. (2015). GPU-Assisted Malware. *International Journal of Information Security*, 14(3):289–297.

Wang, Z. and Lee, R. B. (2006). Covert and side channels due to processor architecture. In *2006 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pages 473–482. IEEE.

Wang, Z., Yan, W., and Oates, T. (2017). Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585.

Wei, S., Aysu, A., Orshansky, M., Gerstlauer, A., and Tiwari, M. (2019). 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.

Weiser, M., Welch, B., Demers, A., and Shenker, S. (1994). Scheduling for Reduced CPU Energy. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

Wodo, W. and Hanzlik, L. (2016). Thermal imaging attacks on keypad security systems. In *SECRYPT*, pages 458–464.

Wu, Z., Xu, Z., and Wang, H. (2012). Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 159–173.

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

Xu, Y., Bailey, M., Jahanian, F., Joshi, K., Hiltunen, M., and Schlichting, R. (2011). An exploration of l2 cache covert channels in virtualized environments. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop (CCSW '11)*, pages 29–40.

Yao, F., Rakin, A. S., and Fan, D. (2020). Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1463–1480.

Zajic, A. and Prvulovic, M. (2014). Experimental Demonstration of Electromagnetic Information Leakage From Modern Processor-Memory Systems. *IEEE Transactions on Electromagnetic Compatibility*, 56(4):885–893.

Zalewski, M. (2005). Cracking safes with thermal imaging. *ser. http://lcamtuf. coredump. cx/tsafe*.

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

Zhan, Z., Zhang, Z., and Kousoukos, X. (2021a). A high-speed, long-distance and wall-penetrating covert channel based on em emanations from dram clock. Technical report.

©2020 IEEE. Reprinted, with permission, from Zhan, Z., Zhang, Z., and Koutsoukos, X. (2020). BitJabber: The World's Fastest Electromagnetic Covert Channel. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*.

Zhan, Z., Zhang, Z., Liang, S., Yao, F., and Kousoukos, X. (2021b). Graphics peeping unit: Exploiting em side-channel information of gpus to eavesdrop on your neighbors. Technical report.

Zhang, K. and Wang, X. (2009). Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User Systems. In *USENIX Security Symposium (USENIX Security 09)*, pages 17–32.

Zhang, Z., Cheng, Y., Liu, D., Nepal, S., Wang, Z., and Yarom, Y. (2020a). Pthammer: Cross-user-kernel-boundary rowhammer through implicit accesses. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 28–41. IEEE.

Zhang, Z., Zhan, Z., Balasubramanian, D., Koutsoukos, X., and Karsai, G. (2018). Triggering rowhammer hardware faults on arm: A revisit. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*, pages 24–33.

©2020 IEEE. Reprinted, with permission, from Zhang, Z., Zhan, Z., Balasubramanian, D., Li, B., Volgyesi, P., and Kousoukos, X. (2020b). Leveraging EM Side-Channel Information to Detect Rowhammer Attacks. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P)*.

Zhao, M. and Suh, G. E. (2018). Fpga-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244. IEEE.

Zhou, Z., Diao, W., Liu, X., Li, Z., Zhang, K., and Liu, R. (2017a). Vulnerable GPU Memory Management: Towards Recovering Raw Data from GPU. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2017(2):57–73.

Zhou, Z., Zhang, W., Yang, Z., and Yu, N. (2017b). Exfiltration of data from air-gapped networks via unmodulated led status indicators. *arXiv preprint arXiv:1711.03235*.

Zhu, T., Ma, Q., Zhang, S., and Liu, Y. (2014). Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 453–464. ACM.

Zhu, Z., Kim, S., Rozhanski, Y., Hu, Y., Witchel, E., and Silberstein, M. (2017). Understanding The Security of Discrete GPUs. In *Proceedings of the General Purpose GPUs (GPGPU)*, pages 1–11.

Zhuang, L., Zhou, F., and Tygar, J. D. (2009). Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):3.