Jonas Juffinger

Attacking and Securing Leaky Systems at the Hardware-Software Boundary

PhD Thesis

Assessors: Daniel Gruss, Onur Mutlu

July 2025

Institute of Information Security
Graz University of Technology

Abstract

The interactions at the boundary between hardware and software components in modern computers are often a source of leakage. Side-channel attacks leak private data such as passwords or web browsing behavior, e.g., via timing differences. Software-triggered hardware faults let attackers elevate their privileges and fully subvert systems. Continuously smaller and faster hardware increases the attack surface by undermining mitigation efforts and introducing new vulnerabilities that can go unnoticed for years.

In this thesis, we significantly advance the understanding of the attack surface at the hardware-software boundary and how defenses can constrain it. In the direction of fault attacks, we show that DRAM disturbance attacks, like Rowhammer, can exist for a long time before being fully understood by discovering a link between effects observed in recent work and our prior work from 2017. We also extend DRAM disturbance research to new hardware by investigating how SSDs can be utilized in Rowhammer attacks, finding lower bounds for SSD-based attacks. Both of these works motivate principled mitigations against DRAM disturbance attacks that are not tailored to specific attack patterns and targets. Hence, we present a novel principled mitigation against DRAM disturbance attacks based on a practical and efficient hardware-software co-design, exceeding the detection and correction capabilities of state-of-the-art solutions significantly. Furthermore, we show that with a secure hardware-software co-design against software-based fault attacks on the CPU, we can even reduce CPU power consumption while increasing performance.

In the direction of side-channel attacks, we perform the first side-channel analyses on modern commodity off-the-shelf SSDs. Even though SSDs are widely used, they have not yet been studied as a source of side channels. We close this research gap, by presenting two novel software-based timing side-channel attacks on SSDs that leak sensitive user information, achieving both, high temporal and spatial resolution.

This thesis consists of two parts. The first part contextualizes my contributions within the state of the art. The second part presents my unmodified[1] first-author publications. All of these papers were anonymously peer-reviewed and accepted at renowned international conferences.

[1] The content of the papers is unmodified from the camera-ready versions. The format of the included papers was modified to fit the layout of this thesis.

Acknowledgments

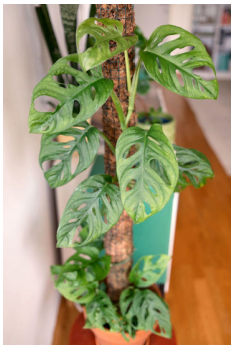
First and foremost, I want to thank my advisor, Daniel Gruss. You persuaded me to do a PhD, and I am so grateful you did. Thank you for all your support during my PhD, the interesting discussion, for giving me the freedom to experiment and pursue my own research, and for being generous with work from home, making my travels between Vienna and Graz bearable.

I also want to thank Onur Mutlu for taking the time and effort to assess my PhD thesis. Thank you also for giving me the chance to present my work to your group and the great discussions that followed.

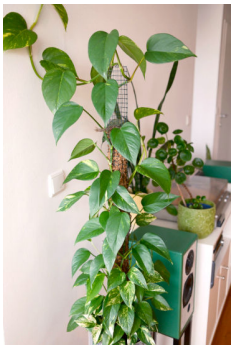
I want to thank Andreas Kogler. Your limitless perseverance keeps inspiring me, and I often think back to our travels and presentations; they were always fun and very well-prepared. I want to thank Stefan Gast. I still remember our first day at the institute and our many great discussions in the own exclusive office. You will continue to inspire me to question absolutely everything. I want to thank Lukas Giner; you taught me important life lessons and a lot about caches.

Especially, I want to thank my other colleagues, Sudheendra Raghav Neela, Fabian Rauscher, Martin Schwarzl, Claudio Canella, Moritz Lipp, Hannes Weissteiner, Carina Fiedler, Roland Czerny, Martin Heckel, Lukas Lamster, Lukas Maar, and Theresa Dachauer. I could not have dreamed of better people to work with. It was a pleasure discussing new, old and stupid ideas, complaining about reviewer B, and celebrating successes with you. All the best for finishing your PhD theses. I also want to thank all the people I met and had a great time with at conferences, Stepan Kalinin, Daniel Weber, Leon Trampert, Lukas Gerlach, Fabian Thomas, Marton Bognar, Michele Marazzi, and many more. Thanks to all the unnamed people and friends I met during this journey.

I want to thank my parents and family. Franziska and Christian, you are the best parents one could wish for. Thank you for your unmeasurable support, your drive to always strive for something bigger, and for giving me the freedom to pursue my dreams. Thank you, especially Wolfgang and all Höcks, Färbingers, as well as Rinderers, for being such great bonus families. Thank you, Thomas; I think we perfectly complemented each



(a) Monstera Adansonii's beautiful perforated leaves.



(b) Golden pothos in the morning sun.

Figure 1.: The fruits of procrastination. Golden pothos and Monsteras grow aerial roots to climb trees and absorb water. A pole filled with moss is a perfect medium for these plants' and author's support and hydration.[2]

other's hunger for success. Thank you, Julian, for igniting my interest in photography, a wonderful creative shared hobby not involving computers.

Thank you, Matthias, for all the years we lived together through school and university. I know that I can always count on you, be it for life, technical or running expertise. I also want to thank the friends I made in Graz and Vienna, Florian, Christoph, Mike, Dominik, and Silva.

A very special thanks go to Paul Smisek and Paul Schmidmayr. While you probably do not know each other, you set me up on a trajectory that led to this thesis by supplying me with my own computers at a very young age that allowed me to experiment with them freely, and introducing me to Linux and programming.

I also want to thank my plants, shown in Figure 1, that gave me great reasons to procrastinate, helping me to free my headspace and view problems from new angles.

Finally, I want to thank my wonderful partner, Sarah. You are my biggest love, inspiration, friend, and supporter. Thank you for enduring the many lonely days while I was in Graz and at conferences. Thank you for sharing the same slightly unhealthy work ethic that made it possible to work through many weekends when deadlines were approaching. Thank you for proofreading my paper drafts. Thank you for always cheering me up. I would not have been able to finish my PhD without you.

[2]The author added this figure while procrastinating.

Contents

Part I.

Attacking and Securing Leaky Systems at the Hardware-Software Boundary

leaky
adjective

Something that is leaky has a hole or crack in it that allows liquid or gas to get through [46].

1 Introduction

Users store plenty of secret data on their computers, from secret keys and passwords to private information like web browsing behavior. As users, we have to trust the programs handling this information. But for programs and the operating system to guarantee security, they themselves have to trust the hardware. Leaky hardware, however, can expose secrets to an attacker. This leakage can come in different forms, for instance, direct information leakage through side channels, as well as software-based fault attacks, like Rowhammer, introducing errors in the computer's electronics, *i.e.*, faults, because of "leaking" stray electrons.

Electronics require specific environments to run in. If requirements are violated, the electronic circuit can be disturbed, causing faults. These requirements can be direct, like the supply voltage and maximum clock frequencym or specifications on how the circuit must be operated, but also outside factors, like temperature or the maximum strength of radiation the circuit withstands. Faults in CPUs or microcontrollers can impact the software running on them. When timed correctly or if the software state is prepared properly, purposefully injected faults can predictably break the software, undermining the system's security [15, 25, 71, 229].

Until 2014, all fault attacks were performed with physical access to the target device [14, 15, 71, 236]. The discovery that they can be caused only by software has been more recent with Rowhammer [137] and overclocking or undervolting attacks [179, 208, 243].

Rowhammer is a software-based fault attack affecting the DRAM [137]. It was first identified as a potential security issue in 2014 [137], and only 9 months later, the first privilege escalation exploits were presented [229]. With the underlying effect being difficult to mitigate, seemingly every new proposed mitigation [10, 30, 41, 45, 83, 145, 168, 171, 257] was defeated by the new exploits and hammering techniques, published over the following years [39, 52, 53, 74, 105, 125, 144, 156, 169, 190, 215, 216, 244, 245,

256, 288]. Proposed and implemented mitigations were broken mainly for two reasons: a deficient implementation due to cost or optimistically misconfigured threshold values due to lack of insights into RowHammer vulnerability [39, 53, 74, 105, 144, 163, 188, 277]. For example, target row refresh (TRR) was implemented insufficiently and could, therefore, be tricked to protect the wrong rows [53, 105]. Additionally, the discovery of Half-Double Rowhammer [144] showed that TRR could be used as a confused deputy, breaking the mitigation in two different ways. Already existing memory protection mechanisms like error correcting codes (ECC) are also ineffective as they can only correct a single bit flip, which is insufficient for targeted Rowhammer attacks [39].

The other group of previously exploited software-based fault attacks includes overclocking and undervolting attacks [36, 130, 179, 208, 243]. These attacks are enabled by kernel-accessible hardware interfaces that allow software to control the CPU clock frequency and CPU supply voltage independently. These two values, clock frequency and supply voltage, are dependent on each other, and can cause faults in the CPU if misconfigured [82]. The faults manifest in a small number of CPU instructions rarely outputting wrong computation results, which can be used to corrupt cryptographic algorithms, but also program flow and array accesses. These can be used to attack trusted execution environments like ARM TrustZone [208, 243] or Intel SGX [36, 130, 179]. However, as numerous publications show, undervolting also has a great potential to save CPU power [11, 12, 68, 124, 146, 147, 172, 193, 194].

Side-channel leakage happens whenever a secret influences an unintended physical property that an attacker can measure. Safe crackers can feel the slightest manufacturing imperfections when turning locks [184], programs influence the power consumption of CPUs [31, 141, 157], or take a different amount of time based on the secret [19, 139]. Information can also leak through the state of a system, like the state of a CPU cache, influenced by the secret. The CPU cache can again be measured through a timing side channel to reconstruct the secret [191, 199]. Side-channel attacks can be divided into physical and software-based attacks. Physical side channels include electromagnetic emissions, power consumption, acoustic emissions, temperature, optical emissions, and vibrations, among others. An attacker typically needs physical access or close proximity to the device to measure the side channel signal [23, 80, 141]. Software-based side channels can be measured from software, and while the underlying reason can be manifold, they mainly cause variations in timing [19, 139, 191, 199].

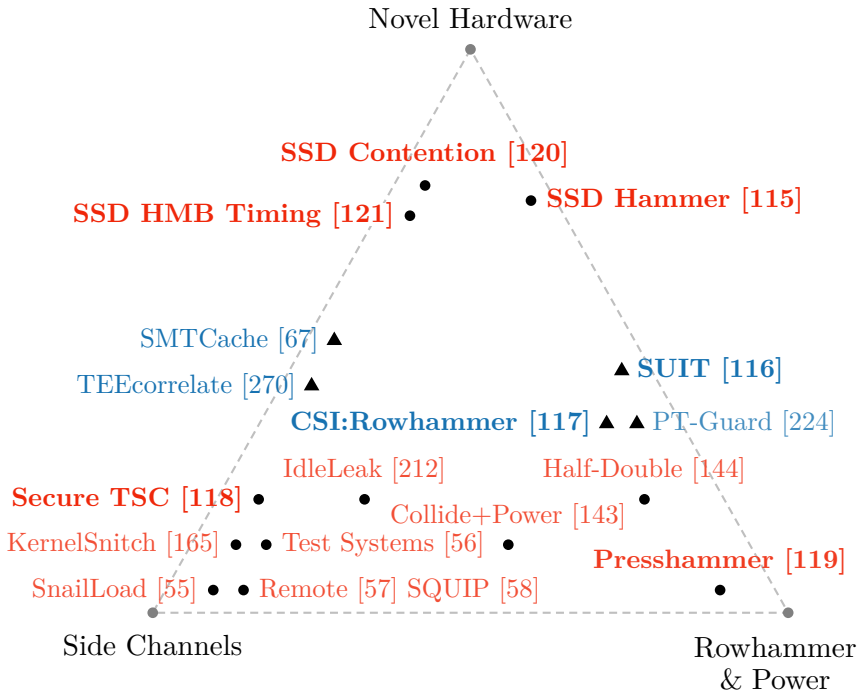


Figure 1.1.: Overview over all my contributions. My main contributions to this thesis are highlighted in bold. Circles with red text define attacks and triangles with blue text defenses or neutral work.

Due to their high spatial and temporal precision, CPU caches were the main software-based side-channel attack target for many years [76, 77, 89, 139, 176, 191, 202, 204, 211, 279, 290]. More recently, research has also looked into side channels in other CPU and computer components in general. These components include random-number generation logic [50], execution ports [4, 22], execution schedulers [57, 58], cache occupancy [233], the PCIe bus [241], idle states [212, 283], operating system data structures [111, 164, 165], device sensors [174, 186, 231, 242], software-based power measurements [143, 157, 186, 242], GPUs [3, 47, 66, 113, 183, 242, 262, 268], frequency scaling [143, 159, 263, 264], and hard-disk drives [23, 80, 128, 155]. These countless side channels were shown to leak not only cryptographic keys but also private user data like browsing behavior and user input, or secret kernel information like heap pointers and KASLR offsets.

However, one component used in almost every modern computer is absent from this list. Commercial off-the-shelf SSDs have seen only very little research on their side channel behavior. While storage systems were already identified as a means to construct covert channels in 1973 [150], only a small body of work showed potential side channel leakage in smart SSDs containing a programmable FPGA on the cloud [250, 251], and Liu et al. [161] analyzed the now-discontinued Intel Optane persistent storage.

1.1. Main Contributions

This section introduces the first-authored papers included in this PhD thesis. In total, I first-authored 7 papers, three of which were published at top-tier conferences. Figure 1.1 gives an overview of all my contributions. They range from a novel principled defense against Rowhammer [117]; a mitigation against privileged CPU undervolting attacks that does not compromise on potential energy savings [116]; a study of the recently discovered RowPress effect in light of our, in 2017 discovered, one-location Rowhammer, including the first RowPress exploit [119]; two papers on the security of the SSD host memory buffer feature, one analyzing its vulnerability to Rowhammer [115] and one showing that the HMB causes an exploitable timing side channel [121]; a comprehensive study on the vulnerability of SSDs to a contention side-channel attack [120]; and, finally, a virtual machine co-location detection method using the new secure TSC feature of AMD [118].

CSI:Rowhammer – Cryptographic Security and Integrity against Rowhammer [117]. With CSI:Rowhammer, we designed a principled defense against Rowhammer that can be implemented with minimal hardware changes and cannot be affected by an incomplete understanding of the Rowhammer problem that broke earlier mitigations. CSI:Rowhammer repurposed the additional memory used for ECCs to store a cryptographically secure message authentication code (MAC) computed over the data stored in the DRAM. This code guarantees data integrity on every read, completely independent of the cause of bit flips in the data. As MACs are, by design, not invertible, they cannot correct bit flips in the data. We show that bit flips can be corrected efficiently by search and that by including the operating system into the correction effort, more advanced corrections can be performed, e.g., by reloading corrupted data from the

disk. We evaluated the security guarantees and showed that on a system that is not under attack, silent data corruption happens on average once every 10^9 billion years. A very fortunate Rowhammer attacker has a chance of 9.75×10^{-5} % to produce a second pre-image when causing bit flips once every 128 ms for a year straight. We evaluated the performance overhead with gem5, showing that it is only 0.74 % on average. This work was published at the IEEE Symposium on Security & Privacy (S&P) in 2023 [117] in collaboration with Lukas Lamster, Andreas Kogler, Maria Eichlseder, Moritz Lipp, and Daniel Gruss.

SUIT: Secure Undervolting with Instruction Traps [116].
The voltage requirement of a CMOS circuit cannot be defined exactly. Many unpredictable environmental factors like process variation, temperature, aging, or input voltage fluctuations do have an impact on the propagation delay of a CMOS circuit at specific voltages [64, 148, 153]. If the propagation delay is too high, undetectable faults can happen inside a CPU [167, 192]. Prior work showed that these faults can be caused by privileged software on ARM and Intel CPUs and used to attack trusted execution environments [15, 36, 130, 179, 208, 243]. However, apart from these faults, undervolting a CPU can significantly reduce power consumption [11, 12, 68, 124, 146, 147, 172, 193, 194] and even increase performance due to thermal and power throttling of CPUs [70, 197, 263]. With SUIT, we show that CPU undervolting can be made secure and reliable by trapping or slightly modifying a small subset of "faulting" instructions. With SUIT's modifications, the CPU only runs in an undervolted state with these faulting instructions disabled. If the running program executes a disabled instruction, the CPU traps, and the operating system increases the voltage to guarantee successful execution. Handling the undervolting state in software allows the operating system to optimize the CPU to the currently running workload dynamically. With SUIT we can reduce the power consumption by up to 20 % with a performance increase of over 3 %. This work was published at the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) in 2024 [116] in collaboration with Stepan Kalinin, Daniel Gruss, and Frank Mueller.

Presshammer: Rowhammer and Rowpress without Physical Address Information [119]. In 2018, we discovered one-location Rowhammer, a new hammer method that only accesses a single row in DRAM [74].

We explained the unexpected finding of one-location Rowhammer flipping bits, with the memory controller employing a closed-row policy [74]. Five years later, in 2023, Luo et al. [163] discovered a different DRAM read-disturbance phenomenon, called Rowpress. Rowpress flips bits by keeping a DRAM row open for long periods, unlike Rowhammer, which opens and closes rows as quickly as possible [137, 163]. With single-row Rowpress being almost identical to one-location Rowhammer, we revisited the latter, analyzed it again, and compared it to Rowpress. In our work, we show that one-location Rowhammer causes bit flips not only due to the memory controller's closed-row policy but also the Rowpress effect, coining the term Presshammer. This finding shows that Rowhammer attacks can exist for a long time before they are fully understood; only Luo et al. [163] were able to actually realize the new underlying phenomenon. With our new understanding of Rowpress, we built the first privilege escalation exploit that uses timing side-channels to find the correct aggressor memory locations for Rowpress. On our system, we were able to exploit Rowpress in less than 10 minutes. This work was published at the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) in 2024 [119] in collaboration with Sudheendra Raghav Neela, Martin Heckel, Lukas Schwarz, Florian Adamsky, and Daniel Gruss.

An Analysis of HMB-based SSD Rowhammer [115]. Solid-state drives can use a part of the main memory, called the host memory buffer (HMB), to cache logical to physical storage address translations [185]. The HMB improves the SSD's performance, as DMA accesses to the main memory are faster than to the flash memory. In this work, we analyze whether these DMA accesses can pose a security risk by causing Rowhammer bit flips in the HMB. While we show that software-induced bit flips in the HMB can cause denial of service, data loss, and even break SSDs, the accesses from the SSD to the HMB are too infrequent to cause actual Rowhammer bit flips. This work was published at the International Conference on Applied Cryptography and Network Security (ACNS) in 2025 [115].

The HMB Timing Side Channel: Exploiting the SSD's Host Memory Buffer [121]. In our third work on SSDs, we again analyze the host memory buffer, this time as a potential source of timing side-channel leakage. We analyze four SSDs that use the HMB feature and reverse engineer how they use the HMB and the cache replacement policies. We

show that the HMB induces timing differences in SSD accesses, which are clearly measurable from user space. We exploit this timing side channel in four case studies: a covert channel between processes; with up to 8.3 kbit/s; a UI redress attack that detects when the `pkexec` binary is executed; a covert channel between virtual machines; and finally, a remote covert channel exploiting a web server as a confused deputy to transmit data over the network. This work was published at the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) in 2025 [121] in collaboration with Hannes Weissteiner, Thomas Steinbauer, and Daniel Gruss.

Secret Spilling Drive: Leaking User Behavior through SSD Contention [120]. Hard disk drives (HDDs) have long been known to be a source of side-channel leakage, including acoustic emanation [80], electromagnetic radiation [23], and timing variation [34, 155]. Chen et al. [34] suggest that SSDs would mitigate certain HDD timing covert channels because transfer speeds and response times of modern SSDs are magnitudes faster than HDD's. In this work, we show that user space applications can, nevertheless, mount contention side-channel attacks. We analyzed 12 different SSD models from 7 vendors, ranging from entry-level PCIe 3.0 SSDs without DRAM to the latest PCIe 4.0 SSDs with DRAM caches. First, we showed that, although different SSDs exhibit very different contention behavior, a covert channel between virtual machines can transmit data with up to 1 503 bit/s. Second, we invade user privacy by leaking websites currently visited by the victim. We exploit the fact that web browsers cache assets of websites, like images and code, and retrieve them from the disk on subsequent visits [177]. Since every website has a different number of assets and loads them at different times, they create an exact fingerprint. Using a machine learning model, we could fingerprint 100 websites in an open-world scenario with an F_1 score of up to 97.0 %. Both the covert channel and website fingerprinting are surprisingly noise resilient. This work was published at the Network and Distributed System Security (NDSS) Symposium in 2025 [120] in collaboration with Fabian Rauscher, Giuseppe La Manna, and Daniel Gruss.

Not So Secure TSC [118]. In cloud computing, programs of many different tenants are running on the same machine, each in its own virtual machine or container. This sharing of physical resources leads to countless

different side-channel attacks [20, 101, 176, 240, 274, 285, 286, 290]. However, the attacker must be co-located on the same machine to exploit these side channels [94, 289]. Distrusting the cloud provider or legal restrictions can prevent tenants from moving to the cloud [201, 239]. AMD was the first to introduce a confidential virtual machine CPU extension, SEV [127], that isolates the virtual machine from the host. Recently, AMD added the SecureTSC feature to SEV, which provides a fully trusted TSC timing source for guests [5, 42]. Until then, the host could modify every timing source inside virtual machines. In our work, we show that SecureTSC can be used for reliable and fast co-location detection. It works because every virtual machine can compute the exact uptime of the CPU it is running on. Under the assumption that it is very unlikely for two CPUs to share the exact same uptime, it can be used as a unique identifier. Using one coordination server that collects and compares all uptimes, we can detect co-location in a noisy environment in 0.13 seconds. Making use of the birthday problem, an attacker only needs 480 virtual machines to co-located at least two of them with a 90 % probability in a data center with 50 000 machines. This work was published at the International Conference on Applied Cryptography and Network Security (ACNS) in 2025 [118] in collaboration with Sudheendra Raghav Neela, and Daniel Gruss.

1.2. Other Contributions

This section briefly introduces the peer-reviewed papers I co-authored during my PhD studies. I co-authored 10 papers, 6 of which were published at top-tier conferences. These papers range from microarchitectural side-channel attacks and defenses, to web- and kernel side-channel attacks, a Rowhammer defense, and a software-based power-analysis attack.

Superscalar processors need a way to schedule ready instructions to the many execution units efficiently. AMD uses a split scheduler design with different schedulers for different execution units. In SQUIP [58], we show that attackers can intentionally fill these schedulers to specific levels. If a scheduler is full, this can cause a pipeline stall, which is measurable by the attacker. Using this timing side channel, for example, the exact executions of integer multiplications on the sibling SMT thread can be detected. We exploit this side channel to leak an RSA private key and build a covert channel. This work was published at the IEEE Symposium on Security & Privacy (S&P) in 2023 [58] in collaboration with Stefan Gast, Martin

Schwarzl, Gururaj Saileshwar, Andreas Kogler, Simone Franza, Markus Köstl, and Daniel Gruss.

Virtual memory page tables are an often-exploited target of Rowhammer attacks, leading to privilege escalation [39, 53, 105, 125, 144, 229, 256, 288]. PT-Guard [224] is a defense that utilizes unused bits of page-table entries to store a cryptographic MAC for integrity protection. This work only requires hardware changes and no changes to the operating system software. The performance overhead is only 0.2 % We performed a study of page-table entries on real systems regarding their value continuity within page tables and showed that most entries can easily be corrected. This work was published at the IEEE/IFIP International Conference on Dependable Systems and Networks in 2023 [224] in collaboration with Anish Saxena, Gururaj Saileshwar, Andreas Kogler, Daniel Gruss, and Moinuddin Qureshi.

Software-based power side-channel attacks exploit secret-dependent power consumption of CPUs only from software [157, 263, 264]. This is possible because older CPUs provide direct energy measurements to software [157] and because energy consumption influences the CPU clock frequency, which is measurable from user space [263]. However, all previous works only attacked specific targets on the system, mainly cryptographic instructions or code [157, 263, 264]. With Collide+Power [143], we are the first ones to show that software-based power side-channel attacks can be used to leak arbitrary data. We do this by "colliding" the secret data with known data known to the attacker in the memory hierarchy to force Hamming weight leakage. This work was published at the USENIX Security Symposium in 2023 [143] in collaboration with Andreas Kogler, Lukas Giner, Lukas Gerlach, Martin Schwarzl, Michael Schwarz, Daniel Gruss, and Stefan Mangard.

Minimizing the energy consumption of CPUs has become a very high priority in the current age of mobile computing. Intel introduced the `tpause` instruction that allows unprivileged software to enter new shallow idle states. In IdleLeak [212], we show that this instruction can be exploited to build a covert channel, an inter-keystroke timing, as well as a website and video fingerprinting attack. IdleLeak is based on the fact that the paused SMT threat is woken up by interrupts to the sibling thread, which makes it possible to get the exact timings of interrupts of the other thread. This work was published at the Network and Distributed System Security (NDSS) Symposium in 2024 [212] in collaboration with Fabian Rauscher, Andreas Kogler, and Daniel Gruss

Our work Remote Scheduler Contention Attacks [57] shows that the AMD split scheduler design can be exploited not only with handcrafted assembly but also from JavaScript. This greatly increases the attack surface as, for example, malicious advertisements could distribute the attack all over the World Wide Web. We analyzed all remaining schedulers, including FPUs absent from the original SQUIP paper [58]. Due to the lack of high-precision timers, we used a microarchitectural race condition to measure scheduler occupancy. With it, we built a high-precision inter-keystroke timing attack and a covert channel between browser windows. This work was published at the International Conference on Financial Cryptography and Data Security in 2024 [57] in collaboration with Stefan Gast, Lukas Maar, Christoph Royer, Andreas Kogler, and Daniel Gruss.

SnailLoad [55] is a novel timing side channel exploiting buffer occupancy between endpoints on the Internet. The victim connects and downloads something from the attacker's server, for example, an image in an advertisement. The attacker's server sends the image very slowly, 400 B/s. The round-trip time between each sent TCP packet and the returning acknowledgment packet gives the attacker a very detailed picture of current network utilization at the victim's side. With SnailLoad, an attacker is able to fingerprint the first 90 seconds of 10 videos with up to 98 % accuracy and fingerprint 100 websites in an open-world scenario with up to 63 % accuracy. This work was published at the USENIX Security Symposium in 2024 [55] in collaboration with Stefan Gast, Roland Czerny, Fabian Rauscher, Simone Franza, and Daniel Gruss.

KernelSnitch [165] is a novel generic timing side-channel attack exploiting various buffers in the Linux kernel. By measuring hash collisions in specific hash table buckets using timing differences of syscalls, addresses of kernel objects can be leaked because they are used as a part of the hash function input. Leaking kernel object pointers can make kernel exploitation more stable. We also showed a covert channel and a website fingerprinting attack. This work was published at the Network and Distributed System Security (NDSS) Symposium in 2025 [165] in collaboration with Lukas Maar, Thomas Steinbauer, Daniel Gruss, and Stefan Mangard.

In our real-world study of the security of educational test systems [56], we evaluate the security of test systems from computer science university classes and identify various security issues. In three case studies, we show that security bypasses are possible. Finally, we perform a user study, asking educators responsible for the test systems about the impact of potential breaches. The breaches could compromise sensitive student

records, confidential research data, and in some cases even embargoed vulnerabilities. This work was published at the Workshop on Operating Systems and Virtualization Security in 2025 [56] in collaboration with Stefan Gast, Sebastian Daniel Felix, Alexander Steinmaurer, and Daniel Gruss.

SMTCache [67] is a new design for secure isolated L1 caches. SMTCache has multiple L1 caches per core and uses only one L1 cache per security domain. Therefore, different security domains cannot interfere with each other. Because only one L1 cache is used at a time, this design does not increase latency and increases power consumption only slightly. This work was published at the International Conference on Availability, Reliability and Security in 2025 [67] in collaboration with Lukas Giner, Roland Czerny, Simon Lammer, Aaron Giner, Paul Gollob, and Daniel Gruss.

With TEEcorrelate [270], we show how performance counter values of a confidential virtual machine guest can be made leakage-free, while still preserving relevant information to the host. TEEcorrelate decorrelates performance counter values using two components, aggregation of performance counter values within configurable length windows, and deferred and speculative performance counter increases within a configurable deviation range. We propose a window length of a few milliseconds and a range of 1024 deviation. These values are enough to mitigate performance counter attacks while the host can still perform load-balancing, accounting, and detection of unusual or malicious activity. This work was published at the USENIX Security Symposium in 2025 [270] in collaboration with Hannes Weissteiner, Fabian Rauscher, Robin Leander Schröder, Stefan Gast, Jan Wichelmann, Thomas Eisenbarth, and Daniel Gruss.

1.3. Outline

Chapter 2 provides background on digital circuits, CPUs, memory, solid state disks, and confidential computing. Chapter 3 gives an overview of the state-of-the-art Rowhammer attacks and mitigations, dynamic voltage- and frequency-scaling studies and attacks, side-channel attacks, and cloud co-location detection. Chapter 4 concludes Part I. Part II provides a complete list of the first- and co-authored papers and the camera-ready versions of the main contributions of this thesis in Chapters 5 to 11.

2

Background

In this chapter, we provide background on relevant knowledge for this thesis. First, we explain how digital circuits and their main building block, MOSFET transistors, work and how supply voltage influences circuit timing. Then, we provide background on the memory subsystem of a computer, detailing virtual memory and DRAM. Finally, we give a short background on solid state drives (SSDs).

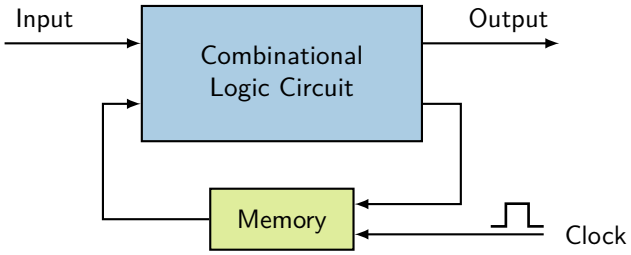


Figure 2.1.: A synchronous sequential digital circuit. The combinational circuit transforms the input and data stored in the memory into output and data to store for the next clock cycle.

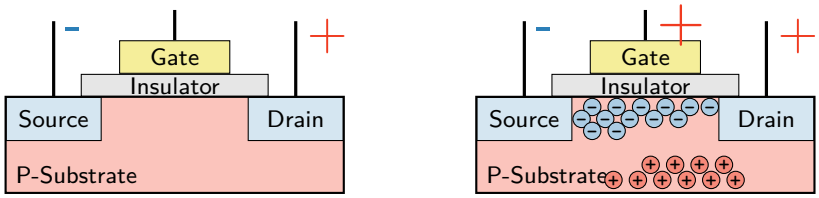
2.1. Digital Circuits

Strongly simplified, digital circuits are built using a large number of connected transistors (electronic switches) to transform input to output signals and store data [82, 148]. The transistors are connected to build different basic building blocks [82, 148]. One large set of building blocks is logical gates that perform a logical operation on their inputs and output the results; another set is flip-flops, used as a memory to store a binary value between clock cycles.

These logical gates are then connected to form combinational logical circuits to perform, for example, mathematical operations like addition or multiplication. The flip-flops are combined to build SRAM memory, storing multiple bits, like registers or caches [82, 148]. Connecting a combinational logical circuit with memory creates a sequential digital circuit, as shown in Figure 2.1. The combinational circuit transforms an input and a previous state into an output and values to store in the memory. On each clock cycle, the memory stores the values at its inputs and outputs them until the next clock cycle [82, 148].

2.1.1. Metal-Oxide-Semiconductor Field-Effect Transistor

Metal-Oxide-Semiconductor Field-Effect Transistors, short MOSFETs, are the dominantly used type of transistors for digital circuits [82, 255]. Their advantages are small static power consumption and size. Figure 2.2 shows a section view of a planar n-channel MOSFET. The substrate is slightly p-doped silicon, the drain and source contacts are n-doped. An insulator



(a) Open MOSFET, no current is flowing between source and drain.

(b) Closed MOSFET, current is flowing from the source to the drain.

Figure 2.2.: The structure of a planar N-channel MOSFET in silicon. The gate is used to "open" and "close" the transistor. To "close" the transistor, *i.e.*, make it conductive between the source and the drain, a positive charge is supplied to the gate. This attracts negative charges that conduct current.

between the substrate and the gate prevents charges from flowing between the two [82]. The substrate is not conductive.

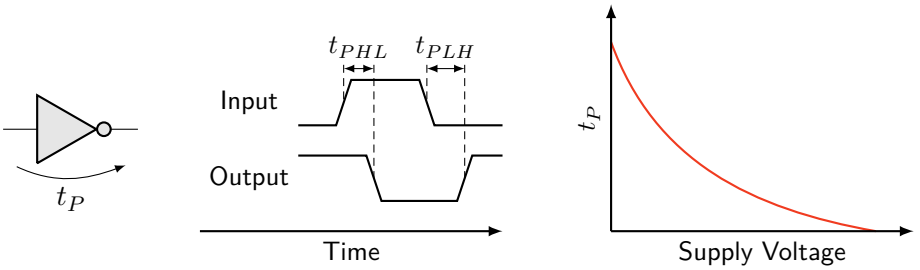
If a positive charge is supplied to the gate, it attracts electrons from the substrate, repulsing positive charges, *i.e.*, holes. Because of the insulator between the substrate and the gate, the electrons gather between the drain and source as shown in Figure 2.2b. If enough electrons gather, the MOSFET starts conducting between the drain and source[1]. When the positive charge is removed from the gate, the electrons recombine with the holes, making the substrate non-conductive again [82, 255].

P-channel MOSFETs work the same as n-channel MOSFETs, with silicon dopings and charges reversed. N- and p-channel MOSFETs are used together in pairs to build Complementary Metal-Oxide-Semiconductor (CMOS) logic used in every modern digital circuit [82, 255].

2.1.2. Circuit Timing and Voltage Requirement

For a digital circuit to work correctly, it must adhere to circuit specific timing and supply voltage limits. If these limits are not met, the circuit can experience faults [68, 82]. For example, in the case of CPUs, the integer multiplication unit can output wrong results, which can be used to attack trusted execution environments [179], see Section 3.2.

[1] Drain and source are electrically identical. Per definition, the source is the connector with a higher potential than the drain. The source is "emitting" positive charges.



(a) t_P of an inverter. (b) Impact of the t_Ps on the signal through an inverter. (c) Relationship between supply voltage and t_P.

Figure 2.3.: The impact of the propagation delay t_P on the traveling through an inverter and the relationship between supply voltage and t_P.

As discussed in the previous section, MOSFETs are switched on and off by attracting charge carriers to the gate, a process that takes time. The gate, insulator, and substrate form a capacitor that must be charged[2] to attract enough charge carriers to create a conductive channel in the substrate [68, 187, 255]. The time it takes to charge a capacitor is a function of its capacity and charging voltage. The capacity is defined by the dimensions of the transistor [235], smaller transistors switch more quickly. This is also the reason for smaller node size CPUs reaching higher frequencies with smaller supply voltages.

These switching delays of transistors in a logical gate cause a so-called propagation delay t_P as shown for an inverter[3] in Figure 2.3. The propagation delay of a circuit is defined by the addition of the propagation delays of all individual logic gates on the longest path, also called the critical path [187]. For a synchronous sequential digital circuit as shown in Figure 2.1, the clock frequency must not be higher than the inverse of the propagation delay of the critical path in the combinational logic circuit. Otherwise, the output or the values stored in the memory may not be fully computed and wrong [82].

In a real environment, the voltage frequency dependency of CMOS circuits is influenced by many internal and external factors [92, 187]. Process

[2]There are many additional parasitic capacitors in a MOSFET transistor, all influencing switching time [255].

[3]Because of the slightly different physical properties of p-channel and n-channel MOSFETs, there is a difference in the t_P from high to low t_{PHL} and low to high t_{PLH}.

variation makes every transistor slightly different. Temperature greatly influences the propagation delay of a circuit. Aging effects like hot-carrier injection or bias temperature instability cause the propagation delay to increase over time [227]. Additionally, sudden increases in current can cause voltage droops. To counter all these effects, digital circuits are supplied with a higher voltage than the absolute minimum required. This additional voltage is called the guardband [92].

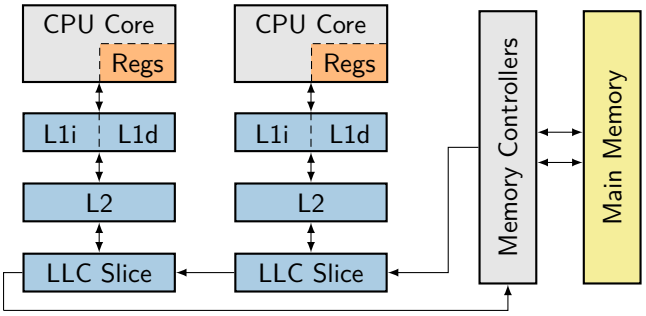


Figure 2.4.: Memory organization in a modern computer. The further away a memory is from the CPU core, the larger and slower it gets.

2.2. Computer Memory

A computer's memory stores all the data the CPU needs during execution. This includes the executed instruction stream as well as the data it processes [259]. In all modern computers, DRAM is used as the main memory because of its high density, low price, and low power consumption. However, the DRAM is too slow to feed enough instructions and data directly to the CPU, which is many magnitudes faster. Therefore, CPUs contain multiple levels of caches that make recently executed instructions and accessed data available with lower latency [237].

A logical layer, virtual memory, partitions the memory between all processes and enables access rights and execution permission management. It is configured with page tables that define to which physical address a virtual address of a specific process is mapped to. The page tables are set up by the operating system and used by the CPU to translate each memory access [98].

2.2.1. Organization

Figure 2.4 shows memory organization from the registers, the memory closest to the CPU, to the main memory, furthest away.

Registers are the memory closest to the CPU's execution units, accessible in a single clock cycle by CPU instructions. In the x86 architecture, instructions can also directly address memory [97]. This is not possible on ARM [9] or RISC-V [267], which have load and store instructions to

access the main memory. There are typically 16 to 32 general-purpose registers. Microarchitecturally, register names are not mapped to fixed locations but are dynamically allocated in the register file, which is usually a few hundred registers large. Registers use static random access memory (SRAM).

Between the CPU cores and the main memory are multiple SRAM cache levels to make data accessible with lower latency. Caches store data that has a high chance of being accessed in the near future. They work based on the principle of locality, which states that recently accessed data or data close by has a higher chance of being accessed again soon. There are multiple cache levels with increasing size and latency. Modern Intel and AMD x86 CPUs typically use three cache levels [6, 97]. The last-level cache is shared between all cores but partitioned into slices. Each core has one slice, and the slices are connected with a ring bus or a mesh [178]. Apart from a handful of exceptions, caches use SRAM [82, 232].

Apart from caches for instructions and data, CPUs also contain a cache for virtual-to physical address mappings, called the translation lookaside buffer, short TLB. It is also split into multiple levels and caches for translations of different page sizes [96].

The main memory is the largest and slowest memory of a computer that is directly accessible by CPU instructions. It uses DRAM technology for its high density and low cost. The main memory is described in more detail in Section 2.3.

2.2.2. Virtual Memory

Modern computers execute many processes simultaneously besides the operating system. For stability and security reasons, these processes must not access the memory of other processes running. All modern CPUs implement a way to isolate processes called paging. On a system with paging, a process never accesses the physical memory directly. Instead, it works in its own private virtual memory space that is translated to physical memory by the CPU.

Paging fragments the physical memory into pages, typically 4 kB large. A set of page tables, unique per process, maps the process's virtual memory pages to physical memory pages. Page tables are managed in a tree structure. The highest-level page table is unique per virtual memory space.

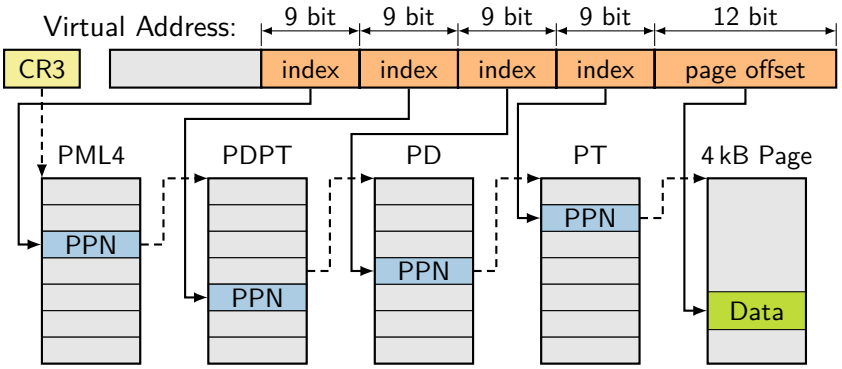
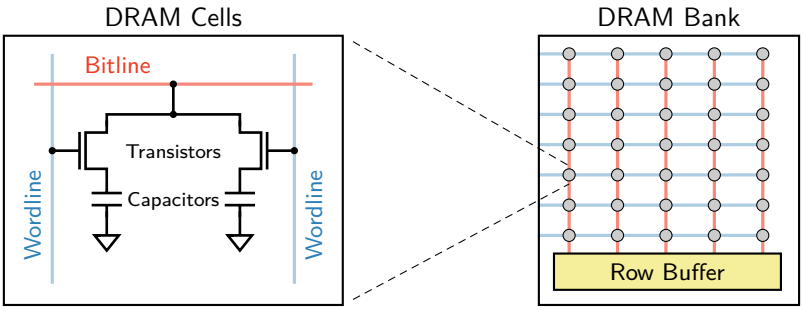


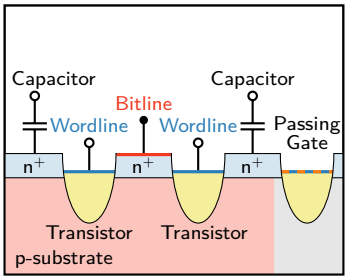
Figure 2.5.: On systems with memory paging, the virtual address is a set of indices into a tree of page tables. The lowest-level page table contains the page frame number or address of the physical memory page.

On x86, the physical address of the highest-level page table is stored in the `CR3` register. Every page table has 512 entries pointing to a lower-level page table. The lowest-level page table points to the mapped page in physical memory. As shown in Figure 2.5, the virtual addresses the process uses are actually only a set of indices into all levels of page tables and an offset within the page.

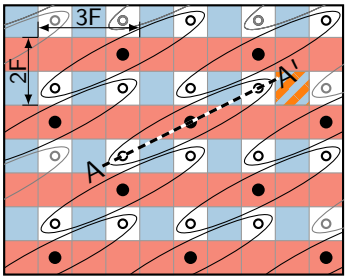
The operating system is responsible for managing the page tables for each process and itself. A process can only access the physical pages that are mapped in its own page-table structure. Apart from page-frame numbers pointing to the next lower-level page table, they also contain various bits that control, for example, access rights, *i.e.*, whether the program is allowed to read, write, and execute memory pages, or the cacheability of the mapped page. Therefore, it is crucial that only the operating system has access to page-table pages and not the user space process itself. Some Rowhammer exploits gain access to their own page tables to escalate privileges [37, 39, 52, 53, 75, 119, 125, 144, 256, 275, 288], see Section 3.1.3.



(a) The structure of DRAM. A cell consists of a transistor and a capacitor. In a DRAM bank, cells are aligned in a grid, connected by the word- and bitlines. The bitlines are connected to the row buffer.



(b1) The cross section of two cells as marked by A---A' in Figure 2.6b2.



(b2) Top view of the cell layout. A single cell is $2F \cdot 3F = 6F^2$ large. The row buffer is not shown.

(b) The physical cell layout of modern $6F^2$ DRAM [88, 170]. The cells are arranged slightly rotated to the word- and bitline grid for maximum density. Passing gates exist because of the grid, but do not connect a bitline to a capacitor.

Figure 2.6.: The schematic and physical layout of DRAM cells and banks.

2.3. DRAM

Dynamic random access memory (DRAM) is a type of memory that is characterized by its low price, low power consumption, and high density compared to SRAM. These properties make it the practically exclusive choice for the main memory of all computers, smartphones, and servers. This makes the DRAM main memory the largest memory of a computer directly accessible by CPU instructions.

2.3.1. DRAM Cell

Its high density and low power consumption in comparison to static random access memory (SRAM) is possible because every bit is stored with only two components, one transistor and one capacitor, also called a storage node. Because these capacitors slowly discharge, they require periodic refreshes not to lose any data, hence, the name *dynamic* RAM [107].

Figure 2.6 shows the schematic and the physical layout of DRAM cells. A cell is connected to two signal lines, as shown in Figure 2.6a. The wordline controls the gate of the transistor, connecting the storage capacitor to the bitline. The charge in the capacitor is read and written through the bitline. Cells are connected through the word- and bitlines in a grid. The ends of the bitlines are connected to the row buffer. By activating a wordline, all capacitors in its row are sensed by the row buffer. DRAM operation is described in more detail in Section 2.3.3.

Figure 2.6b shows the physical cell layout of modern $6F^2$ DRAM [88]. Figure 2.6b1 shows how two capacitors (storage nodes) are connected to a bitline through a MOSFET transistor each. As shown in Figure 2.6b1, the active region (p-substrate) is shared by multiple transistors and their capacitors. This shared active region, which allows electrons to travel between storage nodes, is the reason for the Rowhammer disturbance effect [137] described in detail in Section 3.1.1.

Figure 2.6b2 shows the $6F^2$ layout of the storage node grid in silicon [88]. It is optimized for maximum density by arranging the transistors and capacitors interleaved. The cross section shown in Figure 2.6b1 is marked in Figure 2.6b2 with the dotted line. In the $6F^2$ layout, not all parts of a wordline sit in an active region connecting a capacitor to a bitline. One example of this is marked by the orange-striped field in Figure 2.6b1, also marked in Figure 2.6b2. These regions are no real transistor gates and are, therefore, called passing gates. They are the reason for the passing gate disturbance effect exploited by the Rowpress attack [163], described in more detail in Section 3.1.2.

2.3.2. DRAM Structure and Addressing Functions

The individual DRAM cells are segmented into banks that work independently and allow for parallelized accesses from the CPU. The memory

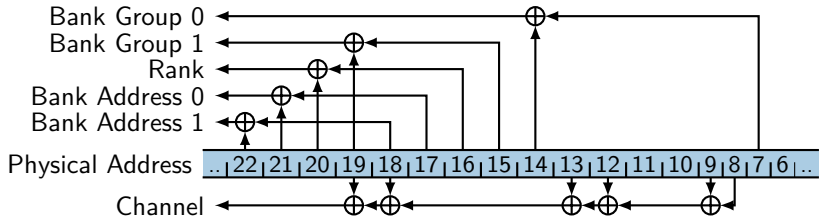


Figure 2.7.: The DRAM addressing functions of an Intel Skylake CPU with two DIMMs, one for each channel and two ranks per DIMM [200].

controller of the CPU applies functions, called the DRAM addressing functions, on the physical address to select a specific bank for each access.

The grids of transistors and capacitors storing the individual bits are split up into banks. Rows of this grid are typically 8 kB wide. There is one row buffer per bank, connected to the rows through the bitlines. On some DRAM modules, the row indexes, *i.e.*, wordlines, are scrambled. This is relevant for Rowhammer attacks, the reason for it, however, is probably due to electrical engineering [244]. A DRAM DIMM contains multiple banks, 16 on DDR4 and 32 on DDR5 per rank. Four banks each are combined into a bank group [107, 108].

A DDR4 DRAM DIMM is connected to the memory controller over a channel. Most consumer CPUs contain two memory controllers and have, therefore, two independent DRAM channels. DDR5 DRAM DIMMs have two independent channels, doubling the number of channels in most CPUs for more parallelism [108]. As the number of physical connections between CPU and DRAM are still approximately the same for DDR4 and DDR5, DDR5 has a doubled burst length to be able to transmit one cache line (64 B) in a single burst over the 32 bit wide channel [108].

Commands to the different independent banks can be interleaved, reducing the overall time the memory controller has to wait for finished commands. To make the best use of this bank-level parallelism, the memory controller interleaves the banks [96] by applying DRAM addressing functions on the physical address to compute the channel, rank, bank group, and bank select bits. DRAM addressing functions were first reverse-engineered by Pessl et al. [200]. The DRAM addressing functions of an Intel Skylake CPU are shown in Figure 2.7. New research followed up with newer tools, getting the functions for newer CPU generations [16, 53, 60, 85, 86, 106, 261]. Jung et al. [122] further reverse-engineered the physical DRAM

mapping by using faults induced by targeted heating to get the on-chip location of DRAM cells.

2.3.3. DRAM Operation

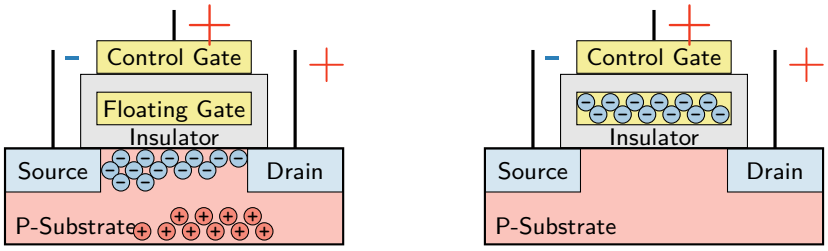
The DRAM protocol is asynchronous. This means that the CPU adheres to specific timings when sending commands, and the DRAM guarantees that every command is finished within that time.

Data Access. To access data on the DRAM, the CPU selects a rank, bank, and row and opens the row to move the data of the row into the row buffer. To do this, the bitlines of the bank must first be precharged (`PRE`). The wordline is activated, connecting the capacitors of a row to the precharged bitlines, slightly changing the voltage. The sense amplifiers detect the voltage change and store the result in the row buffer. Physically, the row buffer and sense amplifiers are one component [170]. This process is destructive, and it removes all charge from the capacitors in the row. Therefore, if there is already data in the row buffer, it must first be written into the previously opened row.

The memory controller can then access the data in the row buffer. The row buffer acts like an 8 kB large cache, decreasing the latency of accesses to the currently opened row. This creates a timing side channel, which was first reported by Pessl et al. [200].

Refresh. The memory controller is also responsible for the refreshes of all cells in the DRAM. It does so by periodically sending refresh commands to the DRAM. The DRAM itself keeps track of which rows were least recently refreshed and can refresh multiple rows within one refresh command window.

The asynchronous nature of DRAM became a problem for on-DRAM Rowhammer mitigations like TRR (see Section 3.1.4), as they only have limited time to perform mitigating refreshes additional to the normal refreshes within a refresh command window. Therefore, the DDR5 standard implements an “Alert Back-Off” (ABO) back channel from the DRAM to the CPU that allows the DRAM to request additional refreshes from the CPU [108].



(a) An erased flash memory cell, storing a logical 1. No electrons are trapped in the floating gate. The voltage at the control gate makes the transistor conductive between the source and the drain.

(b) A programmed flash memory cell. Electrons are trapped in the floating gate, raising the threshold voltage. The same voltage at the control gate does not make the transistor conductive between the source and the drain.

Figure 2.8.: A floating gate MOSFET. To program the cell, electrons are tunneled into the isolated floating gate where they can stay for many years.

2.4. Solid-State Drives (SSDs)

Solid-state drives (SSDs) are storage devices that persistently store data on integrated circuits, typically NAND flash memory. Because they do not use any moving parts, they are faster than hard disk drives (HDDs). Especially, the number of random input-output operations per second (IOPS) is magnitudes higher on an SSD than on an HDD because of the HDD's required disk arm movements for random accesses.

2.4.1. Flash Memory

NAND flash memory is an integrated circuit solid-state memory [8, 280]. The storage of individual bits happens in floating-gate MOSFETs, as shown in Figure 2.8. With a high voltage at the control gate, electrons can be trapped in the floating gate (programming). These electrons increase the threshold voltage of the transistor. The floating gate is fully isolated, and the electrons stay there for many years. To erase a flash memory cell, a voltage in the opposite direction is applied to the control gate to push the electrons back out of the floating gate.

Due to the different high voltage required, NAND flash memory cannot be written randomly, such as DRAM [8, 280]. However, data can be randomly

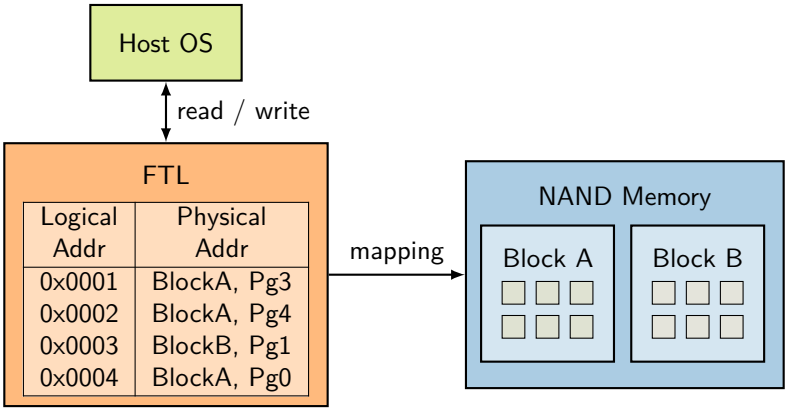


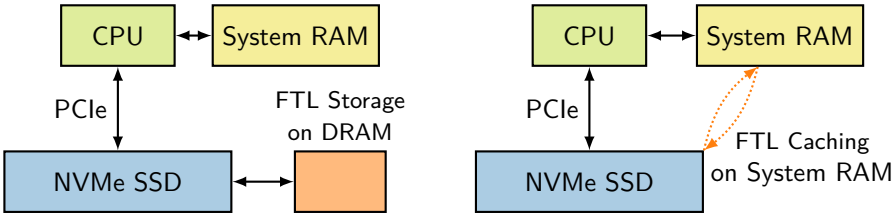
Figure 2.9.: The flash translation layer (FTL) translates the logical address used by the host operating system to the actual physical location of the page in the NAND memory on each access.

read from every location. Erased NAND flash memory stores all `1`'s, to write data, specific bits are then set (programmed) to `0`. Reprogramming is possible as long as the new data is a subset with only `1` to `0` transitions. Erasing NAND flash memory takes significantly longer than reading and programming it. Additionally, the number of bits that must be read, programmed or erased at once differs. Typically, the page size used for reads and programming is 512 B, 2 048 B, or 4 096 B. Erasures happen in blocks of 32, 64, or 128 pages.

The number of these program-erase cycles of flash memory is limited because the insulation between the substrate and floating gate is slightly damaged each time [8, 280]. To extend the lifetime of an SSD, the controller counts the number of P/E cycles each flash memory block and dynamically remaps data to wear out all blocks equally fast. This process is called wear-leveling and is automatically performed by the SSD controller.

2.4.2. Flash Translation Layer

Wear-leveling, garbage collection, and hiding the erase-before-write limitation of flash memory for performance causes the data on an SSD to be scattered [8, 13, 99, 126, 129]. This makes a persistent translation layer necessary that translates the ordered logical page addresses, accessed by the operating system, to the actual physical locations of the data on the



(a) SSD with DRAM. (b) SSD with HMB.

Figure 2.10.: NVMe SSDs with the host memory buffer (HMB) feature can use a part of the system RAM to cache FTL entries.

NAND flash chips. This flash translation layer (FTL) is also stored in the flash memory itself, as it must be persistent.

On every read and write to the SSD, the memory controller must access the FTL to either get the current translation or update it. Therefore, accesses to the FTL must be fast. If the SSD controller were to access the FTL in the flash memory for every read, each read would incur an additional read, basically halving the IOPS. Therefore, SSD controllers use caches for their FTL. Budget SSDs use a small on-chip cache in the SSD controller that caches a small subset of the FTL. High-end "pro"-level SSDs use a separate DRAM chip next to the memory controller that is large enough to hold the entire FTL while the SSD is turned on. When starting, the SSD copies the whole FTL into the DRAM and only reads it from there, greatly increasing random IOPS. Most mid-range NVMe SSDs can use a feature called Host Memory Buffer as a trade-off between cost and performance.

2.4.3. Host Memory Buffer (HMB)

With the HMB feature, NVMe SSDs can request main memory from the operating system that they then use to cache parts of the FTL [185], as shown in Figure 2.10. The operating system can then reserve memory for exclusive access from the SSD. The SSD accesses the HMB through direct memory accesses (DMA) over PCIe. Kim et al. [133] were the first to analyze the HMB usage of SSDs and found that the HMB memory is mainly used to cache parts of the FTL. We found the same to be true [115, 121], see Chapter 8 and 9. Caching parts of the FTL has a performance advantage over accessing the flash memory for each FTL entry, but is

slower than accessing an integrated DRAM. As the HMB is typically not large enough to store the whole FTL, only parts are stored, and different eviction policies and prefetching define which parts [121].

2.4.4. HMB Prevalence

In our HMB side channel paper [121], we used TechPowerUp's SSD database [54] containing 869 SSDs from 109 manufacturers to understand the prevalence of the HMB feature in SSDs. Of these 869 SSDs, 694 use the PCIe interface. PCIe is a requirement of the HMB as SATA does not support DMA accesses. 37 % or 255 of all PCIe SSDs do not have DRAM. Of these DRAM-less PCIe SSDs, 97.6 % or 249 support the HMB feature. This shows that the feature is very prevalent in DRAM-less SSDs, as it enables a performance gain at almost no cost.

3

State of the Art

This chapter discusses state-of-the-art DRAM disturbance attacks and defenses, CPU undervolting, and side-channel attacks. We detail state-of-the-art DRAM-disturbance exploit techniques and mitigations in Section 3.1. Section 3.2, shows how CPU undervolting was used to attack trusted-execution environments and its potentials. Finally, we provide an overview of side-channel attacks in Section 3.3, ranging from contention to storage side-channel attacks and cloud co-location detection.

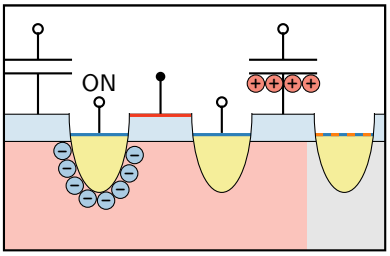
3.1. DRAM Disturbance Attacks

In this section, we discuss state-of-the-art software-based DRAM disturbance attacks and defenses. Software-based DRAM disturbance attacks allow an attacker to flip bits in the DRAM by only accessing it from software. There are two DRAM disturbance phenomena: Rowhammer [137] and Rowpress [163]. Rowhammer was identified as a potential security issue by Kim et al. [137] in 2014. Since then, countless different Rowhammer methods [53, 74, 105, 144, 206, 229], see Section 3.1.1, exploit techniques [21, 26, 37, 39, 52, 53, 75, 103, 119, 125, 144, 149, 213, 215, 249, 256, 275, 288], see Section 3.1.3, and also possible defenses [10, 18, 27, 30, 32, 33, 38, 41, 45, 51, 63, 72, 76, 83, 87, 88, 100, 102, 109, 114, 117, 123, 131, 137, 145, 151, 171, 189, 196, 198, 210, 221, 223, 224, 225, 230, 238, 253, 257, 258, 276, 281, 284], see Section 3.1.4, were published and implemented. Rowpress is another software-based DRAM disturbance attack that was only discovered recently by Luo et al. [163] in 2023, see Section 3.1.2.

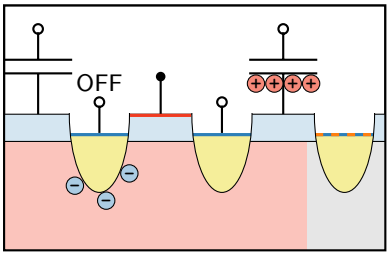
In this section, we show that although more than ten years have passed since the publication of Rowhammer, the problem still exists to this day. A countless number of defenses were repeatedly broken by novel Rowhammer patterns and exploit techniques [39, 53, 74, 105, 144]. With this thesis, we concur with others that a full mitigation of Rowhammer is only possible with full integrity protection of all data in the DRAM [51, 220]. We argue that the underlying problem is that academics, as well as DRAM manufacturers, try to mitigate Rowhammer without it being fully understood yet [74, 144, 163, 188, 277]. Our work, CSI:Rowhammer [117] (Chapter 5) shows that the performance overhead of adding integrity protection to all data in the DRAM is negligible. Integrity protection, in addition to mitigations targeting Rowhammer more directly, could solve the problem of DRAM disturbance attacks once and for all. In our works Presshammer [119] (Chapter 7) and HMB Rowhammer [115] (Chapter 8) we further the understanding of DRAM disturbance attacks.

3.1.1. Rowhammer

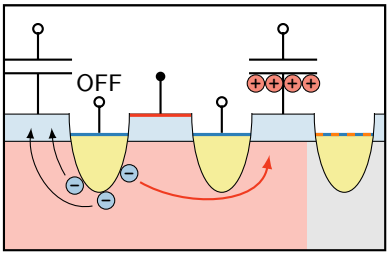
Rowhammer allows an attacker to flip bits in DRAM by frequently accessing (hammering) one or multiple rows adjacent to the victim row [137].



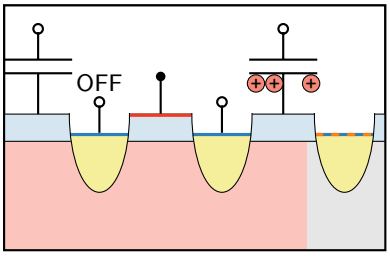
(a) Channel Inversion: When the wordline is activated, electrons collect at the channel to connect the capacitor with the bitline.



(b) Electron Spreading: After the wordline is deactivated, the electrons are free to move around.



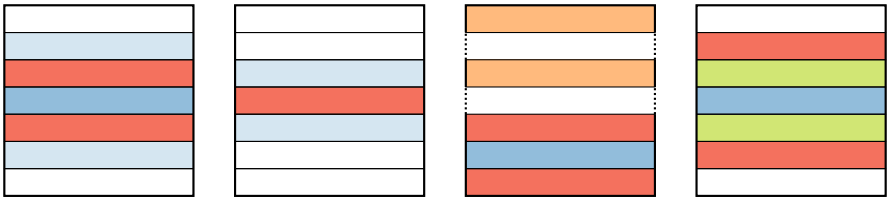
(c) Electron Injection: Most electrons move back into the closest capacitor, but some "leak" and are injected into a neighboring one.



(d) Capacitor Discharge: Every injected electron slightly discharges the neighboring capacitor until the stored bit flips.

Figure 3.1.: How the Rowhammer effect recombines charges in the victim capacitor, flipping the stored bit [88, 137].

Root Cause. The underlying cause of Rowhammer is the shared substrate in which multiple cells are embedded [88, 195, 219, 260, 278], as we explained in Section 2.3.1. Figure 3.1 illustrates the physical cause of the Rowhammer effect. When the wordline is on, the storage capacitor is connected to the bitline, and electrons from the capacitor flow along the transistor's channel. After the wordline is turned off, the electrons are no longer held to the transistor's channel and spread out. Most are injected back into the storage capacitor where they came from, as it is the closest, but some electrons "leak" and are injected into other neighboring storage capacitors. With each leaking electron, the charge in this storage capacitor is depleted slightly. If the wordline is toggled often enough, the charge depletes to a point below the threshold where the bit flips.



(a) Double-Sided (b) One-Location (c) Multi-Sided (d) Half-Double

Figure 3.2.: Four hammer patterns. Double-sided Rowhammer [229] sandwiches the victim (blue) between two attacker rows (red). One-location Rowhammer [74] accesses only a single row in a bank. Multi-sided Rowhammer patterns [53, 105] access additional dummy rows (orange) to evade TRR mitigations. Half-double Rowhammer [144] provokes mitigative TRR accesses (green) around the victim row to flip bits.

Hammer Patterns. Figure 3.2 shows four different hammer patterns. The hammering patterns describe which rows in the DRAM an attacker accesses. Initially, Kim et al. [137] only hammered by frequently accessing multiple single rows in a DRAM bank, *i.e.*, single-sided Rowhammer. In 2015, Seaborn et al. [229] showed that Rowhammer can induce more bit flips if not any two rows are hammered but two rows sandwiching the victim rows, *i.e.*, double-sided Rowhammer.

One-location Rowhammer only accesses a single row of the DRAM and still causes bit flips [74]. This is unexpected because the memory controller usually keeps the row open in the row buffer, serving the data from there. This behavior of a memory controller, keeping the row open as long as possible, is called the open-row policy. However, memory controllers can also work with a closed-row policy where they close a row after each access [74]. This can have performance benefits on highly multi-threaded systems, where subsequent accesses are unlikely to go to the same DRAM row. In Presshammer [119] (Chapter 7), we show that one-location Rowhammer also causes bit flips due to the Rowpress effect that was not yet known when one-location Rowhammer was found.

DRAM implementing the target row refresh (TRR) mitigation, see Section 3.1.4, is not vulnerable to single- and double-sided Rowhammer [53, 83]. To evade TRR, Frigo et al. [53] included decoy accesses to their multi-sided Rowhammer pattern. These decoy accesses are synchronized to refresh commands and can confuse certain TRR implementations to protect the wrong victim rows. Additionally, they can also cause too many

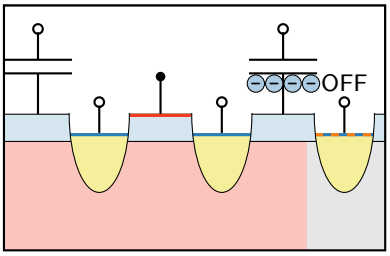
potential victim rows, so that the DRAM is overwhelmed and cannot refresh all of them in the time it has available during the periodic refreshes. Their fuzzer was able to induce bit flips in 13 of 42 tested DDR4 modules. With Blacksmith, Jattke et al. [105] describe the different decoy accesses with a frequency, phase and amplitude and their fuzzer varies them to find long and complex hammer patterns. Jattke et al. [105] were able to flip bits in all of their 40 tested DDR4 modules. Gerlach et al. [61] reproduced bit flips in 8 of 10 tested DDR4 modules using the Blacksmith fuzzer.

TRR can also be exploited as a confused deputy to aid Rowhammer attacks. With Half-Double Rowhammer [144], we present a new physical property of Rowhammer. When hammering one row further away from the victim row (far aggressor), very infrequent accesses to the row neighboring the victim row (near aggressors) are enough to "transport" the leaked stray electrons to the victim row capacitors. This hammer pattern is shown in Figure 3.2d. On DRAM that is protected with TRR, the accesses to the near aggressor are caused by TRR, which tries to protect these rows from the far aggressor. However, due to the half-double effect, these TRR accesses cause the bit flips in the actual victim row. We were able to flip bits on 5 out of 7 mobile devices with LPDDR4x DRAM [144].

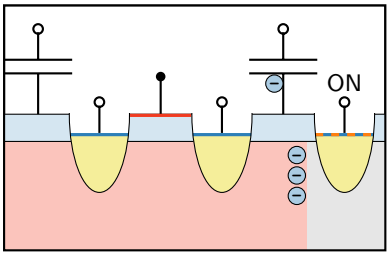
3.1.2. Rowpress

Rowpress also allows an attacker to flip bits in DRAM. Instead of opening and closing a row as quickly as possible, Rowpress keeps (presses) one or multiple rows adjacent to the victim row open for as long as possible [163]. Rowpress is caused by the passing-gate effect and affects different cells than Rowhammer [163]. Figure 3.3 shows how the passing-gate effect causes bits to flip in the DRAM. Due to the physical layout of a modern $6F^2$ DRAM cell, not all gates on a wordline connect a storage node to a bitline. This is shown in Figure 2.6b in Section 2.3. When a wordline is activated, these so-called passing gates on this wordline attract electrons from the storage node. The longer the wordline is active, the more electrons are attracted. When the wordline is turned off, most electrons return to the storage node they came from, but some "leak" further out into the substrate to another capacitor. This reduces the charge in this neighboring capacitor, flipping bits [88, 93].

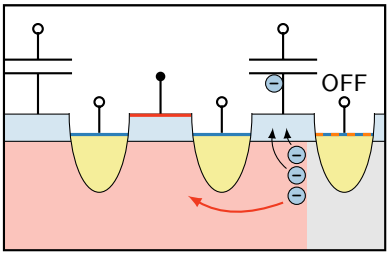
Luo et al. [163] analyzed 164 DRAM chips using an FPGA platform and showed that it is a common DRAM vulnerability across all three major



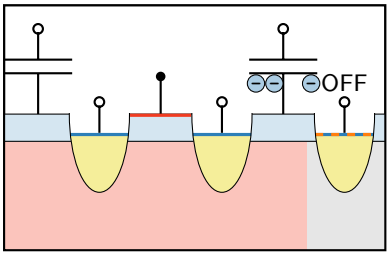
(a) Passing Gate Channel Inversion: When a wordline is activated, electrons from storage nodes are attracted to passing gates.



(b) Electron Spreading: When the wordline is deactivated again, the electrons are free to move around.



(c) Electron Injection: Most electrons move back into the closest capacitor, but some "leak" and are injected into the substrate.



(d) The electrons injected into the substrate are now missing in the capacitor, flipping the bit.

Figure 3.3.: How the Rowpress effect removes charges from the victim capacitor, flipping the stored bit [88, 163].

DRAM manufacturers. Similar to Rowhammer, Rowpress is also more effective if the victim row is sandwiched between two aggressor rows. To cause Rowpress bit flips from software, Luo et al. [163] keep a row open as long as possible by accessing up to all 128 cache lines within one row. They were able to cause Rowpress bit flips in one Samsung DRAM DIMM from 2018. The DIMM we used in the PressHammer paper [119] (Chapter 7) is also from 2017. This suggests that modern Rowhammer mitigations are well equipped to also prevent Rowpress bit flips.

Recent work by Jattke et al. [104] used a DRAM interposer and a oscilloscope to analyze Rowpress on real systems in more detail. They found that the pattern accessing all 128 cache lines within one row keeps the row open for at most 292.95 ns, much less than allowed by the standard.

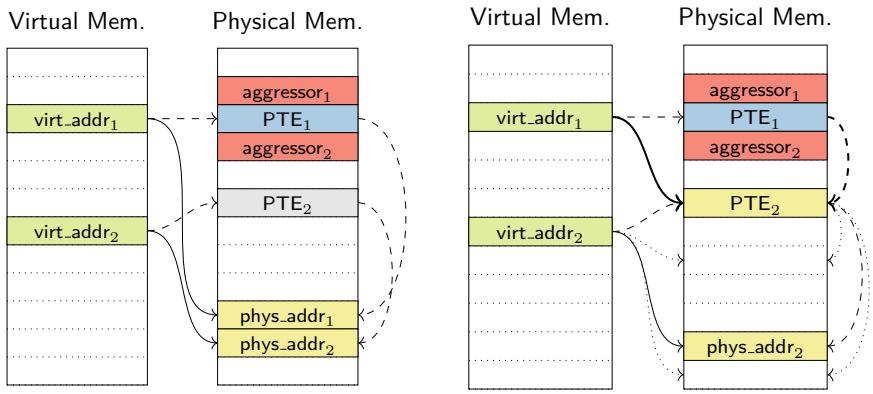
3.1.3. DRAM Disturbance Exploitation

From now on, we will use Rowhammer as a synonym for Rowhammer and Rowpress, as the following concepts apply similarly to both DRAM disturbance attacks. Rowhammer flips bits at seemingly random locations in the DRAM. Apart from the different hammering patterns to get bit flips, actually causing Rowhammer bit flips from unprivileged programs and exploiting these bit flips is well-researched [21, 26, 35, 37, 39, 52, 53, 75, 103, 119, 125, 144, 149, 213, 215, 249, 256, 275, 288].

DRAM Addressing. Programs running on a CPU cannot directly address specific rows in the DRAM. Two layers of indirection, *i.e.*, mappings are between the address a program accesses and the row and bank addressed in the DRAM [200]. The first indirection is virtual memory, as described in Section 2.2.2, that maps the seemingly contiguous and private virtual memory space to physical memory [98]. The second indirection is the DRAM addressing functions, which map the physical addresses to ranks, bank groups, banks, and rows, described in Section 2.3.2.

The virtual-to-physical address mapping is not readable by unprivileged processes anymore after it was disabled to prevent the first Rowhammer exploits [138]. Therefore, many different methods to get physically contiguous memory were proposed [75, 144, 256]. Gruss et al. [75] were the first to use 2 MB transparent huge pages. Web browsers automatically allocate them for large arrays [75], and they can also be requested from Linux using `madvise` on most systems [154, 247]. However, transparent huge pages are not available on all systems, e.g., Android, and can easily be turned off on systems where they are enabled by default [247]. Alternatively, Van der Veen et al. [256] massaged Linux's buddy allocator [69] to get predictable physical page placement. Memory allocator massaging was also used by multiple exploits afterwards [52, 149, 215]. Other exploits [52, 144] used the fact that when allocating enough memory, chunks will be contiguous, and use the bank conflict side channel to detect this contiguity.

Contiguous memory gives an attacker knowledge about additional physical address bits above the page offset. These bits can then be used with the DRAM addressing functions, which we explained in Section 2.3.2, of the attacked CPU to address rows in specific banks [52, 144].



(a) Correct Mapping (b) Successful Exploit

Figure 3.4.: The Rowhammer page table exploit. By hammering PTE_1 using $aggr_1$ and $aggr_2$ the PFN is changed to point to the physical address of PTE_2. Now the attacker has write access to PTE_2 through its virtual mapping $vaddr_1$ and can access the whole physical memory.

Exploit Targets. Rowhammer can be used to attack a large variety of systems. These attacks can be classified by their goal, e.g., privilege escalation in the operating system or the browser, stealing of cryptographic keys, or denial of service, and the target, e.g., Intel or ARM CPUs.

The first two Rowhammer exploits were presented by Seaborn et al. [229]. They flipped bits in machine code executed in the Chrome NaCl sandbox to disable the jump target sanitization, so that they could jump to unaligned targets, e.g., a `syscall` instruction hidden in a `movabs` instruction. The second exploit they developed is the most used Rowhammer privilege escalation exploit to this day [37, 39, 52, 53, 75, 119, 125, 144, 256, 288]. It uses Rowhammer to flip bits in the page-frame-number in a page-table entry to gain write access to a page table, as shown in Figure 3.4. This exploit can also be executed by hammering through implicit page-table accesses [287, 288]. A similar technique was also demonstrated against hypervisors [35, 275].

Rowhammer is also possible from JavaScript [26, 52, 75, 215]. A similar approach to the page table exploit was used in browser sandbox escapes [26, 52, 215]. For this exploit, a fake array object is created, and a pointer is hammered to point to this object. This fake array object can then be modified to point everywhere in the browser's memory. Tatar et al. [245] and Lipp et al. [156] demonstrated Rowhammer attacks over the network.

Gruss et al. [74] were the first to flip bits in a binary to elevate privileges. They identified 29 different offsets in the `sudo` binary that can be flipped to break the password verification logic. They combined this technique with one-location Rowhammer from within an SGX enclave for a stealth attack.

Kwong et al. [149] shows that Rowhammer can also be used to directly leak data, exploiting the data dependency of Rowhammer [137]. Tobah et al. [249] used Rowhammer on kernel code to reenable Spectre attacks. Rowhammer can also be used to steal cryptographic keys [21, 213], also in post-quantum schemes [7, 180]. Jang et al. [103] and Gruss et al. [74] exploit Rowhammer for denial of service attacks by halting SGX enclaves. Recently, Jattke et al. [106] demonstrated that Rowhammer is also possible from AMD CPUs, and Marazzi et al. [169] demonstrated Rowhammer on RISC-V. The first exploit on ARM was already presented in 2016 [256] using uncachable memory, followed by Rowhammer from the integrated ARM GPU [52]. Weissmann et al. [269] performed a Rowhammer attack from an FPGA through DMA. Qiao et al. [206] demonstrated Rowhammer using non-temporal instructions and Heckel et al. [84] increased the hammer effectiveness by up to 830 with multithreaded Rowhammer using the `cmpsb` and `repe` instructions.

In our work "An Analysis of HMB-based SSD Rowhammer" [115] (Chapter 8), we analyzed whether SSDs using a host memory buffer, see Section 2.4.3, can be used as a confused deputy to hammer the HMB in the main memory. However, our results show that the HMB itself is integrity protected, and bit flips are detected. Upon detection, all of our tested SSDs freeze, leading to a denial of service or a data loss in the worst case. Additionally, the SSD's accesses to the HMB are not frequent enough to induce bit flips even on highly susceptible DRAM.

3.1.4. Software-Only Defenses

Since the first publication about Rowhammer, attackers and defenders have been constantly working to outsmart each other [74, 181]. Additionally, the ever-shrinking node size exacerbates the problem with every new DRAM generation [132]. While attackers are mostly academic researchers, Rowhammer defenses were proposed by academia but also developed and implemented by DRAM and CPU manufacturers [109]. Defenses can be categorized into software-only defenses, which are easy to implement and

deploy, and defenses (partly) implemented in hardware, which are more difficult to deploy but potentially more effective.

Especially in the first years of Rowhammer research, many software-only defenses were proposed [10, 30, 41, 100, 145, 257]. As Rowhammer attacks cause suspicious CPU cache usage, similar to CPU cache side-channel attacks, cache side-channel detection methods were also proposed to detect ongoing Rowhammer attacks [38, 76, 87, 100, 198, 284].

ANVIL by Aweke et al. [10] detects ongoing Rowhammer attacks using cache hardware performance counters similarly to cache side-channel detection works [38, 76, 87, 198, 284]. ANVIL reduces false positives by only intervening if at least two rows on the same bank are hammered. Machek [41] proposed counting the number of cache misses in the kernel and halting the CPU until the next refresh if it exceeds a preconfigured threshold. Irazoqui et al. [100] use static code analysis to detect instructions often used in cache and Rowhammer attacks like `clflush`, non-temporal moves, `rdtsc`, or fences. Brasser et al. [30] add guard rows around kernel memory to increase the distance between potential aggressor and victim rows. GuardION by Van der Veen et al. [257] is a mitigation against their attack on Android ARM devices [256]. They put guard rows around DMA memory requested using Android's ION driver. ZebRAM [145] also works by adding guard rows, however, between every row. It then uses all these guard rows as an integrity-protected swap memory to not halve the memory capacity.

Di et al. [45] present "Copy-on-Flip", a defense that can be purely implemented in software if the system uses ECC DRAM. This makes it a software-only defense relying on a readily available hardware feature. ECC DRAM is not enough to prevent a Rowhammer attack, as shown by Cojocar et al. [39]. However, the exploit of Cojocar et al. [39] requires a templating phase where it collects single-bit flips that are corrected by ECC to combine these bit flips in the second step. Copy on flip [45] prevents this by remapping pages whenever correctable bit flips happen.

3.1.5. Hardware-Based Defenses

Over the years, improving attacks have shown that software-only defenses are insufficient to solve Rowhammer [74]. Therefore, many mitigations that require hardware changes have been proposed and also implemented [18, 27, 32, 33, 51, 63, 72, 83, 88, 102, 109, 114, 117, 123, 131, 137, 151, 171,

189, 196, 210, 221, 223, 224, 225, 230, 238, 253, 258, 276, 281]. These hardware-based defenses can be classified into defenses that prevent bit flips and defenses that detect bit flips to correct them. Both methods have disadvantages: While preventive defenses can become obsolete with new attack methods, correcting detected bit flips becomes inefficient quickly if bit flips are too frequent. We argue that combining both methods can most efficiently prevent current and future Rowhammer attacks.

Defenses Preventing Bit Flips

This class of defenses tries to prevent bit flips by protecting potential victim rows before their cells are discharged below the threshold [18, 27, 32, 88, 102, 110, 114, 134, 136, 137, 151, 168, 171, 189, 196, 209, 210, 221, 225, 230, 238, 253, 258, 271, 272, 276, 281]. We further classify these defenses into three different methods of identifying potential victim rows and by their actions after a potential victim row is identified.

Probabilistic Victim Row Identification. Already with the first paper on Rowhammer, Kim et al. [137] proposed a number of potential mitigations, the most prominent being probabilistic adjacent row activation (PARA), now called probabilistic target row refresh (pTRR) [123]. Intel has implemented pTRR in some of its CPUs since 2014 [123]. With pTRR, whenever a row is accessed, its neighboring rows are refreshed with a low probability. If the probability is chosen well, the performance overhead is negligible, while Rowhammer bit flips become very unlikely. Similar probabilistic mitigations were proposed by Son et al. [238], You et al. [281], and Kim et al. [136].

Qureshi et al. [210] use only a single counter to count row activations and probabilistically select which row is counted at each refresh. Jaleel et al. [102] use a small number of counters per bank and do not use heuristics to select which row is counted intentionally. Because of that, they claim to be immune to specifically crafted access patterns that manage to break policy-driven defenses [53, 105].

Counter-Based Victim Row Identification. DRAM manufacturers and JEDEC acted rather quickly and added target row refresh (TRR) to many later DDR4 DIMMs and as a required feature to the LPDDR4 standard [109]. TRR does not probabilistically refresh neighboring rows

but actually counts the accesses to rows and refreshes the neighbors if a certain DIMM-specific threshold is reached [72, 83, 109, 112, 131].

However, to save die area and power consumption, these first TRR implementations only counted a limited number of rows [53, 105]. With complex access patterns, the sampler can be tricked to count the accesses to decoy rows and not the actual aggressors [53, 105], as explained in Section 3.1.1. Another issue is that the DRAM has only limited time to perform preventive row refreshes. If more victim rows are hammered than the DRAM can refresh, bit flips still happen, even if the DRAM detected the victim rows in time [53, 168, 171, 253]. A third issue with TRR, half-double Rowhammer, was presented by Kogler et al. [144] where TRR actually aids the attacker to hammer from a greater distance, as explained in Section 3.1.1. Half-double Rowhammer can be mitigated by increasing the range of protected rows around a detected aggressor row [33, 83, 171, 189, 210, 223, 276].

A large body of works continuously improves the area-, performance- and power-overhead of counter-based mitigations [18, 27, 88, 102, 134, 168, 171, 189, 253]. These works aim to make the mitigation cheaper and at the same time more difficult to evade.

Hong et al. [88] propose a stochastic and approximate counting (DSAC) algorithm that filters out the accesses to decoy rows that evaded prior TRR implementations. However, DSAC was later broken by Qazi et al. [205] due to the internal state of the used LFSR random number generator being too small. Bostanci et al. [27] use a data structure called count-min sketch to estimate the activation count of rows, requiring significantly less area than having a counter per row. This design may overestimate but never underestimates the activation count. Olgun et al. [189] make the key observation that workloads usually access the same row ID in multiple banks due to the DRAM addressing functions aiming for bank-level parallelism, see Section 2.3.2. Therefore, they use the Misra-Gries algorithm to track aggressor rows, not per bank but shared across all banks.

Per Row Activation Counting (PRAC). Jedec recently added per-row activation counting (PRAC) to the DDR5 standard [108, 222]. A DRAM with PRAC contains a counter for every single row in each bank. Bennett et al. [18] showed that this approach is viable by using a novel

DRAM mat design that does not add any performance overhead. Bennett et al. [18] also use the already existing ALERT_n signal to pause the memory controller, called ALERT-Back-Off (ABO). This is required to perform additional preventive refreshes that did not fit into the refresh commands.

Canpolat et al. [33] are the first to perform a comprehensive study of the PRAC feature as defined in the DDR5 standard [108]. They show that due to the ABO signal, a crafted adversarial access pattern can hog up to 94 % of DRAM throughput and degrade system throughput by up to 95 %. Woo et al. [271] and Qureshi et al. [209] both show that the initial design proposed by Bennett et al. [18] is insecure, additionally to the potentially high overhead found by Canpolat et al. [33]. Both works propose secure designs with less performance overhead.

Other Victim Row Identification Methods. Apart from the designs that track potential victim rows in the DRAM, some designs use other methods inside the memory controller. Vig et al. [258] use a sliding window mechanism to detect currently attacked victim rows and add them to an integrity tree to protect the data in them. Seyedzadeh et al. [230] use adaptive trees of counters, Lee et al. [151] time window counters, Park et al. [196] content addressable memory [110], Yaglikci et al. [276] bloom-filters and Joardar et al. [114] machine learning.

Preventive Refresh. Most proposed defenses refresh victim rows after they exceed the activation threshold. To have enough time to perform all required refreshes, the DDR5 standard adds two mechanisms [108]. The first one is Refresh Management (RFM). The memory controller keeps track of the number of activations sent to the DRAM banks and gives a bank additional time for refreshes by sending RFM commands if the activations exceed a threshold. The second mechanism is the ALERT-Back-Off (ABO) signal that the DRAM can send to the memory controller to force it to send RFM commands. Jattke et al. [104] recently found that neither Intel nor AMD CPUs send RFM commands even if the DRAM requires them to mitigate Rowhammer effectively.

Recent works also looked at ways to make preventive refreshes more efficient. Rega [171] proposes a DRAM design with an additional set of buffering sense amplifiers that are only used for data transfers. These free up the other sense amplifiers to perform refreshes in parallel. Tugrul et al.

[253] study real DRAM chips and find that the refresh latency t_{RAS} can be decreased by 64 % while requiring only 0.54 % additional refreshes [253].

Row Remapping. Some defenses do not refresh potential victim rows but also remap them to another location in the DRAM bank to shield them from future attacks. Saileshwar et al. [221] detect aggressor rows similar to Park et al. [196], however, they do not refresh the victims but swap the aggressor rows with other randomly selected rows. This is to prevent attacks like Half-Double [144]. Saxena et al. [225] propose a very similar mitigation, remapping the aggressor rows to a quarantine area. Woo et al. [272] find an attack pattern that breaks the defense by Saileshwar et al. [221] in under a day and propose a more secure design themselves.

Data Integrity-Based Defenses

These defenses aim to detect data integrity violations and then correct the bit flips whenever possible. Multiple mitigations have been proposed using message authentication codes or hashes to ensure the integrity of data in the DRAM [51, 91, 220]. However, all of them have shortcomings regarding the threat model of a Rowhammer attack. Ivec [91] and Synergy [220] protect against local attackers with capabilities exceeding those of a Rowhammer attacker. Their integrity trees reduce the performance significantly. Safeguard [51] can only correct a single bit flip.

In our work PT-Guard [224], we propose a mitigation to protect page tables from bit flips because they are an easy attack target for privilege escalation [35, 39, 52, 53, 75, 119, 125, 144, 256, 275, 288]. PT-Guard [224] combines unused bits in groups of 8 page-table entries to store a 96-bit MAC for data integrity protection.

In CSI:Rowhammer [117] (Chapter 5), we protect all data against integrity violations using a MAC, albeit with a small memory overhead similar to ECC DRAM. This approach guarantees that no bit flips, whatever their reason is, go undetected and can be exploited. The only remaining possible attack on a system with CSI:Rowhammer would be a denial of service. We show that the performance overhead is small, and correction of multiple bit flips is realistic as we perform the correction in the kernel. This correction of multiple bit flips in the kernel enables techniques like reloading corrupted cached data from disk. However, CSI:Rowhammer does not have Chipkill-like correction capabilities [43] for all data, similar

to current SECDED ECC. We argue, that the possibility to correct bit flips spread over multiple DRAM chips is more important if Rowhammer is a threat. Future work could focus on further improving the trade-off between performance and Chipkill-like correction capabilities of data integrity-based defenses.

Currently, no data integrity-based defense has the correction capabilities to efficiently correct the high number of bit flips in modern high-density DRAM if they were otherwise unprotected. Data integrity-based defenses must be paired with defenses preventing bit flips. On the other hand, defenses preventing bit flips should also be paired with data integrity-based defenses to protect against novel attack methods [144], incomplete defense implementations [104], or other causes of bit flips [43].

3.2. CPU Undervolting

As described in Section 2.1.2, digital circuits are typically supplied with a slightly higher voltage than the minimum required. This so-called voltage guardband guarantees the circuit's correct functionality even when changing die temperature, aging, or supply voltage droops [70, 197]. Reducing this voltage guardband saves energy and can even increase performance on modern CPUs [70, 116]. This is due to the fact that the CPU clock rate is typically limited by the thermal design power (TDP). The TDP defines the maximum power the CPU is allowed to draw for a prolonged time, or the maximum CPU die temperature directly. With a reduced power consumption due to a lower supply voltage, the CPU can clock higher before reaching the TDP. Not surprisingly, given these benefits, the undervolting potential of CPUs and resulting power savings and performance increases were thoroughly researched [68, 70, 124, 146, 193, 194].

However, CPU undervolting can also be used to attack trusted-execution environments [15, 130, 142, 179, 208, 243]. On many CPUs, the multiplication and AES circuits are among the circuits with the highest voltage requirement. If this requirement is not met, the calculations can produce faulty results. This can be exploited by undervolting CPUs to precisely the voltage level where the CPU continues running normally, but these instructions fail [130, 179, 208, 243]. Tang et al. [243] attack ARM TrustZone by overclocking the CPU, which has the same effect as undervolting. Qiu et al. [208] manipulate the voltage to attack ARM TrustZone. Murdock et al.

[179] and Kenjar et al. [130] exploit Intel SGX enclaves by undervolting to steal cryptographic keys or induce memory safety into bug-free enclaves.

Barenghi et al. [17] analyze different countermeasures to protect cryptographic algorithms against fault attacks. Their framework can automatically add instruction duplication, instruction triplication, and parity checking for stored values to programs. Kogler et al. [142] presented a targeted countermeasure against CPU undervolting attacks against trusted execution environments. They analyzed a range of CPUs and confirmed that integer multiplication is the first faulting instruction on most CPUs at different frequencies. An enclave can therefore use multiplications where the correct results are known as trap instructions to detect whether the CPU is undervolted.

A number of works also try to enable stable and secure undervolting [11, 12, 172], including our work SUIT [116] (Chapter 6). Bacha et al. [11, 12] utilize on-chip ECC of Intel Itanium CPUs to guide their undervolting. On Intel Itanium CPUs, the cache and register file are the first components that become erroneous when undervolting. Intel Itanium CPUs can also report corrected ECC errors in the cache and register file to the operating system. This allows Bacha et al. [11, 12] to keep the voltage at exactly a level where no error occurs. Unfortunately, errors in multiplication circuits cannot easily be detected. Koutsovasilis et al. [147] undervolt the CPU depending on the running workload based on the performance monitor counter changes it causes. Therefore, Maroudas et al. [172] additionally also differentiate between kernel and user space, based on their observation that user space code can be undervolted more than kernel code. They do, however, require voltage changes on every kernel entry and exit. Ernst et al. [49] proposed a hardware solution to enable secure undervolting. Their design Razor, uses slightly skewed clock edges and shadow circuitry to detect when critical paths are close to violating their timing constraints. Razor would add considerable complexity to modern chip designs and is not used in practice. In our work SUIT [116] (Chapter 6), we trap potentially faulting instructions when undervolting to only execute them while the CPU is supplied with a high enough voltage. The performance overhead of our design is smaller than the potential performance gain from undervolting, enabling a higher performance at a lower CPU power consumption.

3.3. Side-Channel Attacks

In this thesis, we will only focus on software-based side channels. Software-based side channels do not require physical access to the device that is attacked. The first software-based side-channel attack by Kocher [139] leaked keys from Diffie-Hellman, RSA, and DSS via timing. Percival [199] was the first to perform a Prime+Probe attack on CPU caches, attacking an RSA algorithm. Osvik et al. [191] performed the first Prime+Probe attack on AES T-tables and coined the term Prime+Probe. Bernstein [19] timed the execution of AES T-tables over the network with many different messages to leak the key. Since then, the very high spatial and temporal resolution of CPU cache side channels has been exploited for a variety of attacks that became more generic compared to attacks on specific cryptographic implementations [76, 77, 89, 176, 202, 204, 211, 279, 290].

Recently, with CPU caches being very well researched, more research has focused on other parts of a computer system like, execution units and schedulers [1, 4, 22, 57, 58, 218, 265], the memory bus [90, 273, 274], operating system data structures [111, 164, 165], DRAM row conflicts [200, 228], the page cache [73], software-based power and frequency scaling [143, 157, 159, 207, 263, 264], device sensors [174, 186, 231], CPU fans [79], performance counters [59, 78], the PCIe bus [65, 234, 241, 248], in-memory computing architectures [48, 266], FGPAs [65, 203, 250, 251], interrupt detection [40, 212, 283], GPUs [3, 47, 66, 113, 183, 242, 262, 268], and random number generation logic [50]. However, while storage has been identified as a potential source for side- and covert channels long before the first cache side channels [90, 128, 150, 162, 226, 252], the research on commercial off-the-shelf SSDs is sparse.

Covert Channels. Side channels typically have a specific target, e.g., cryptographic keys, with an attack on this target evaluated in the work. However, comparing their capabilities became a non-trivial task given the wide range of side channels. One way to compare side channels is by their channel capacity, the rate at which information can be transmitted over the channel. To measure the channel capacity, a covert channel is well suited as the sender and receiver are cooperating. Most side-channel papers implement and benchmark a covert channel [4, 22, 50, 57, 58, 65, 73, 76, 81, 90, 111, 165, 175, 176, 200, 204, 211, 212, 228, 234, 250, 251, 265, 273, 274, 283]. Covert channels are also used in transient-execution attacks to transmit the transiently leaked data to the outside world [140, 158].

Covert channels were also already researched long before the first software-based side-channel attacks. Lampson [150] was the first to recognize the difficulty of confining a program on a system with a shared operating system and shared hardware. This was followed by further research on covert channels and their mitigation [90, 128, 162, 226, 252]. However, these works focused on special "secure systems" as, for example, defined by the Trusted Computer System Evaluation Criteria [44] and not commercial off-the-shelf systems, which are the target of most recent side-channel research.

Contention Side Channels. The first covert channels by Lampson and others [90, 128, 150, 162, 226, 252] exploited contention on a shared resource, e.g., the hard drive. Apart from these secure systems [44], contention was mainly a concern for the system's performance. With the emergence of multiprocessor systems, contention on the shared memory or last-level cache became a concern [173, 214, 254]. Contention was also researched in databases [2, 24].

If contention can be focused on a small subset of a system, e.g., execution ports or schedulers, powerful attacks are possible from the leaking of encryption keys [1, 4, 22, 58] and remote attacks from JavaScript [57, 218]. Contention on the network has also been shown to leak private user information [55, 62, 246]. Memory bus contention [273, 274], as well as HDD contention [155], has been used for covert communication. The additional contention caused by RowHammer mitigation-induced memory latency differences has been used for covert communication and website fingerprinting [29]. PCIe contention has been shown to enable a variety of attacks [65, 234, 241, 248] if PCIe switches or PCIe platform controller hubs are used to share a link, which is not generally the case for SSDs. As a part of this thesis [120] (Chapter 10), we show that contention on SSDs can be used for covert communication and allows to fingerprint websites visited by the victim with very high accuracy.

Storage Side Channels. Shared storage has been identified very early as a potential source for side channels [150]. Karger et al. [128] exploit the optimizations of hard disks' arm movements to build a covert channel. Hard disks were again attacked more recently, first, by Lipinski et al. [155], who presented a contention-based covert channel with up to 0.1 bit/s. Biedermann et al. [23] measured the electromagnetic emissions of a hard

drive using a smartphone to perform operating system and application fingerprinting. Finally, Guri et al. [80] measured the acoustic emissions of a hard drive to build a covert channel. Guri et al. [80] highlight that SSDs mitigate this covert channel as they do not emit noises. Chen et al. [34] also suggest that SSDs would mitigate certain timing side channels that are present in HDDs.

Trochatos et al. [250, 251] focused on smart SSDs that include a user-programmable FPGA to enable near-storage compute [152]. They use them to build a covert channel between users using the smart SSD consecutively and between those using it simultaneously. Lui et al. [161] analyze the (now discontinued) Intel Optane persistent memory for side channels and reverse engineer the internal cache to develop four new attacks: a covert channel, a keystroke-timing attack, a fully remote covert channel, and a note board attack. Gruss et al. [73] exploit the operating system's page cache that caches recently accessed storage pages to build a 7 kB/s to 273 kB/s covert channel, an ASLR break on Windows 10, a UI redressing attack, and a keystroke-timing attack. Jiang et al. [111] exploit the `fsync` syscall on the file system that writes dirty data back to storage to build a 20 kbit/s covert channel.

However, even though the storage subsystem has long been known to be a source of side-channel leakage and research exists on HDDs [23, 80, 128, 155], smart SSDs [250, 251], and Intel Optane persistent memory [161], this thesis is the first to analyze the side-channel leakage of regular off-the-shelf SSDs. In our work "Secret Spilling Drive" [120] (Chapter 10), we show that contention on SSDs can be used for covert communication and allows to fingerprint websites with very high accuracy.

Cloud Co-Location. In cloud environments, multiple tenants are running on the same shared hardware. Many of them store confidential data of themselves or their customers. Therefore, cloud environments have long been a target of side-channel research. A large body of works studied attacks [20, 176, 240, 274, 285, 286] but also their mitigation [135, 160, 284]. For most of these attacks to work, the attacker must be co-located with the victim on the same physical hardware. This motivated a specialized branch of research on cloud co-location [94, 95, 166, 217, 289], with cloud providers constantly eliminating known channels for co-location detection [95]. Ristenpart et al. [217] were the first to show that targeted co-location is feasible in the AWS cloud. Inci et al. [95] use the last-level

cache as a covert channel to detect co-location and mount an attack on RSA. Makrani et al. [166] massage the resource provisioning systems with their attack, called Cloak & Co-locate, to get targeted co-location. Zhao et al. [289] achieve co-location in Google Cloud's Function-as-a-Service environment and steal secret ECDSA nonce bits using a cache side channel [290].

In our work "Not So Secure TSC" [118] (Chapter 11), we show that the SecureTSC feature of AMD's confidential computing platform SEV SNP enables very fast and low-overhead co-location detection.

4

Conclusion

In this thesis, we presented novel Rowhammer and side-channel attacks. Motivated by these attacks, we explored how hardware-software co-designs can be effective and efficient mitigations against software-based fault attacks. We conclude this thesis with three key insights:

DRAM disturbance attacks like Rowhammer are still not fully understood, motivating the need for principled mitigations. After more than 10 years, Rowhammer is still not perfectly mitigated because of an incomplete understanding of DRAM disturbance effects [74, 144] and insufficient mitigations [105, 106, 125]. We demonstrated that phenomena like one-location Rowhammer can exist for years before being fully understood [119] (Chapter 7). We also extended Rowhammer research to new hardware and showed that SSDs using the main memory can also hammer it [115] (Chapter 8). *It is likely that future research* will uncover further previously unknown links between existing DRAM disturbance effects. Future research must also continuously assess the Rowhammer risk in new threat models that include peripheral hardware similar to SSDs. Every PCIe revision doubles the throughput, doubling the potential hammer rate of DMA accesses to the main memory. The same danger comes from novel hardware accelerators like neural processing units that access the DRAM directly. When compute-in-memory architectures become widespread, their susceptibility to Rowhammer and also their potential to be used in an attack will be a relevant research question [28, 182, 282]. With a continuously increasing understanding of DRAM disturbance attacks and novel threat models, mitigations focusing on the currently existing attacks are unlikely to guarantee sustained security. Only principled mitigations that can guarantee data integrity regardless of the specific attack and simultaneously provide strong correction capabilities can sustainedly solve the problem [117] (Chapter 5).

Principled mitigations can be inexpensive and even increase efficiency. To prevent hardware faults under all circumstances, manufacturers add guardbands to properties like timings and supply voltages. By employing principled mitigations against hardware faults, the guardbands can be reduced as they are not longer solely responsible for preventing faults. The guarantee to detect all data corruption in the DRAM of CSI:Rowhammer [117] (Chapter 5) could also be used to optimize the refresh rate of DRAM dynamically. In the future, the memory controller could reduce the refresh rate to find a sweet spot between the energy "wasted" on bit flip correction and the energy savings from the reduced refresh rate. As the reduced refresh rate also increases the performance of the DRAM, this could lead to a faster and more energy-efficient system. With increasing hardware-software system complexity, *it is likely that future research* will increase the role of software in the handling and correction of bit flips, for example, by performing bit flip correction in user space or by taking knowledge about the DRAM's structure and properties into account to reduce the search space significantly. In our work SUIT [116] (Chapter 6), we showed that the CPU voltage can be significantly decreased if the CPU guarantees that potentially faulting instructions are not executed with this lower voltage. Overall, SUIT can decrease the power consumption of the CPU while even increasing the system's performance. To improve CPU efficiency further, with more fine-grained voltage control based on more factors than just specific instructions, *we need further research* to understand the CPU's susceptibility to faults and crashes in low-voltage scenarios. Additionally, the operating system could also interact with the CPU to inform the CPU about required voltage levels in the near future. This could significantly reduce the overhead from CPU voltage change delays, similar to cache prefetchers.

For new hardware, known classes of problems, like side channels, are often not considered in the design phase, leaving the new hardware vulnerable. Even though SSDs are widely used in almost all computers, they have not yet been studied as a source of side channels. We close this research gap and perform the first side-channel analyses on modern commodity off-the-shelf SSDs, presenting two novel software-based timing side-channel attacks on SSDs. The first attack is a cache timing side channel that provides leakage with high spatial resolution in combination with a software interface [121] (Chapter 9). The second side channel leaks SSD contention with high temporal resolution [120] (Chapter 10). We performed a large study of SSDs' susceptibility to this contention side channel and showed that all tested SSDs were vulnerable. These works show that single components

like SSDs can be the source of multiple side channels, and *it is likely that future work* will uncover many more, given their complexity and wide use. Similar to potential DRAM disturbance attacks, this also extends to other hardware components where research is sparse, either because they were newly introduced or received little attention from the academic community until now. While storage contention has already been identified as a source for side channels for a very long time [150], we are the first that showed that it also impacts high performance NVMe SSDs. The high performance actually enables more fine-grained attacks, such as fingerprinting the subtle signals from websites being opened on the machine with very high accuracy. As the performance is continuously increasing, it is likely that these attacks will get worse in the future. In our work on AMD's Secure TSC [118] (Chapter 11), we showed another example of a new feature that was introduced without taking side channels into account. The Secure TSC timer allows an attacker to detect the co-location of virtual machine guests. Without a principled approach to consider known classes of problems, in particular, side-channel leakage will continue to be introduced in new hardware components and features.

References

[1] Onur Aciicmez and Jean-Pierre Seifert. Cheap Hardware Parallelism Implies Cheap Security. In: FDTC. 2007 (pp. 47, 48).

[2] Rakesh Agrawal, Michael J Carey, and Miron Livny. Concurrency Control Performance Modeling: Alternatives and Implications. In: ACM Transactions on Database Systems (TODS) (1987) (p. 48).

[3] Jaeguk Ahn, Cheolgyu Jin, Jiho Kim, Minsoo Rhu, Yunsi Fei, David Kaeli, and John Kim. Trident: A Hybrid Correlation-Collision GPU Cache Timing Attack for AES Key Recovery. In: HPCA. 2021 (pp. 5, 47).

[4] Alejandro Cabrera Aldaya, Billy Bob Brumley, Sohaib ul Hassan, Cesar Pereida García, and Nicola Tuveri. Port Contention for Fun and Profit. In: S&P. 2019 (pp. 5, 47, 48).

[5] AMD. AMD64 Architecture Programmer's Manual. 2023 (p. 10).

[6] AMD. AMD64 Architecture Programmer's Manual. 2024 (p. 21).

[7] Samy Amer, Yingchen Wang, Hunter Kippen, Thinh Dang, Daniel Genkin, Andrew Kwong, Alexander Nelson, and Arkady Yerukhimovich. PQ-Hammer: End-to-end Key Recovery Attacks on Post-Quantum Cryptography Using Rowhammer. In: S&P. 2025 (p. 39).

[8] Seiichi Aritome. NAND Flash Memory Technologies. John Wiley & Sons, 2015 (pp. 27, 28).

[9] ARM. ARM A64 Instruction Set Architecture. Sept. 2018 (p. 20).

[10] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd Austin. ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks. In: ASPLOS (2016) (pp. 3, 32, 40).

[11] Anys Bacha and Radu Teodorescu. Dynamic Reduction of Voltage Margins by Leveraging On-chip ECC in Itanium II Processors. In: ISCA. 2013 (pp. 4, 7, 46).

[12] Anys Bacha and Radu Teodorescu. Using ECC Feedback to Guide Voltage Speculation in Low-Voltage Processors. In: MICRO. 2014 (pp. 4, 7, 46).

[13] Amir Ban. Flash file system. US Patent 5,404,485. 1995 (p. 28).

[14] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. In: Proceedings of the IEEE (2006) (p. 3).

[15] Alessandro Barenghi, Guido Bertoni, Emanuele Parrinello, and Gerardo Pelosi. Low Voltage Fault Attacks on the RSA Cryptosystem. In: Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). 2009 (pp. 3, 7, 45).

[16] Alessandro Barenghi, Luca Breveglieri, Niccolò Izzo, and Gerardo Pelosi. Software-only Reverse Engineering of Physical DRAM Mappings for Rowhammer Attacks. In: International Verification and Security Workshop (IVSW). 2018 (p. 25).

[17] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni. Countermeasures Against Fault Attacks on Software Implemented AES: Effectiveness and Cost. In: Workshop on Embedded Systems Security. 2010 (p. 46).

[18] Tanj Bennett, Stefan Saroiu, Alec Wolman, and Lucian Cojocar. Panopticon: A Complete In-DRAM Rowhammer Mitigation. DRAMSec. 2021 (pp. 32, 40–43).

[19] Daniel J. Bernstein. Cache-Timing Attacks on AES. Tech. rep. 2005. URL: `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf` (pp. 4, 47).

[20] Johann Betz, Dirk Westhoff, and Günter Müller. Survey on covert channels in virtual machines and cloud computing. In: Transactions on Emerging Telecommunications Technologies (2016) (pp. 10, 49).

[21] Sarani Bhattacharya and Debdeep Mukhopadhyay. Curious Case of Rowhammer: Flipping Secret Exponent Bits Using Timing Analysis. In: CHES. 2016 (pp. 32, 37, 39).

[22] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. SMoTherSpectre: Exploiting Speculative Execution through Port Contention. In: CCS. 2019 (pp. 5, 47, 48).

[23] Sebastian Biedermann, Stefan Katzenbeisser, and Jakub Szefer. Hard Drive Side-Channel Attacks using Smartphone Magnetic Field Sensors. In: FC. 2015 (pp. 4, 5, 9, 48, 49).

[24] Mike Blasgen, Jim Gray, Mike Mitoma, and Tom Price. The Convoy Phenomenon. In: ACM SIGOPS Operating Systems Review (1979) (p. 48).

[25] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In: EUROCRYPT. 1997 (p. 3).

[26] Erik Bosman, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector. In: S&P. 2016 (pp. 32, 37, 38).

[27] F Nisa Bostanci, ISmail Emir Yüksel, Ataberk Olgun, Konstantinos Kanellopoulos, Yahya Can Tuğrul, A Giray Yağliçi, Mohammad Sadrosadati, and Onur Mutlu. CoMeT: Count-Min-Sketch-Based Row Tracking to Mitigate RowHammer at Low Cost. In: HPCA. 2024 (pp. 32, 40–42).

[28] F. Nisa Bostanci, Konstantinos Kanellopoulos, Ataberk Olgun, A. Giray Yaglikci, Ismail Emir Yuksel, Nika Mansouri Ghiasi, Zulal Bingol, Mohammad Sadrosadati, and Onur Mutlu. Revisiting Main Memory-Based Covert and Side Channel Attacks in the Context of Processing-in-Memory. In: arXiv:2404.11284 (2025) (p. 51).

[29] F Bostancı, Oğuzhan Canpolat, Ataberk Olgun, İsmail Emir Yüksel, Konstantinos Kanellopoulos, Mohammad Sadrosadati, A Giray Yağlıkçı, and Onur Mutlu. Understanding and Mitigating Side and Covert Channel Vulnerabilities Introduced by RowHammer Defenses. In: arXiv:2503.17891 (2025) (p. 48).

[30] Ferdinand Brasser, Lucas Davi, David Gens, Christopher Liebchen, and Ahmad-Reza Sadeghi. CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory. In: USENIX Security. 2017 (pp. 3, 32, 40).

[31] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In: CHES. 2004 (p. 4).

[32] Oğuzhan Canpolat, A Giray Yağlıkçı, Ataberk Olgun, Ismail Emir Yuksel, Yahya Can Tuğrul, Konstantinos Kanellopoulos, Oğuz Ergin, and Onur Mutlu. BreakHammer: Enhancing RowHammer Mitigations by Carefully Throttling Suspect Threads. In: MICRO. 2024 (pp. 32, 40, 41).

[33] Oğuzhan Canpolat, A Giray Yağlıkçı, Geraldo F Oliveira, Ataberk Olgun, Oğuz Ergin, and Onur Mutlu. Understanding the Security Benefits and Overheads of Emerging Industry Solutions to DRAM Read Disturbance. In: DRAMSec (2024) (pp. 32, 40, 42, 43).

[34] Ang Chen, W Brad Moore, Hanjun Xiao, Andreas Haeberlen, Linh Thi Xuan Phan, Micah Sherr, and Wenchao Zhou. Detecting Covert Timing Channels with Time-Deterministic Replay. In: OSDI. 2014 (pp. 9, 49).

[35] Wei Chen, Zhi Zhang, Xin Zhang, Qingni Shen, Yuval Yarom, Daniel Genkin, Chen Yan, and Zhe Wang. HyperHammer: Breaking Free from KVM-Enforced Isolation. In: ASPLOS. 2025 (pp. 37, 38, 44).

[36] Zitai Chen, Georgios Vasilakis, Kit Murdock, Edward Dean, David Oswald, and Flavio D Garcia. VoltPillager: Hardware-based fault injection attacks against Intel SGX Enclaves using the SVID voltage scaling interface. In: USENIX Security. 2020 (pp. 4, 7).

[37] Yueqiang Cheng, Zhi Zhang, and Surya Nepal. Still Hammerable and Exploitable: on the Effectiveness of Software-only Physical Kernel Isolation. In: arXiv:1802.07060 (2018) (pp. 22, 32, 37, 38).

[38] Marco Chiappetta, Erkay Savas, and Cemal Yilmaz. Real time detection of cache-based side-channel attacks using Hardware Performance Counters. In: Cryptology ePrint Archive, Report 2015/1034 (2015) (pp. 32, 40).

[39] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In: S&P. 2019 (pp. 3, 4, 11, 22, 32, 37, 38, 40, 44).

[40] Jack Cook, Jules Drean, Jonathan Behrens, and Mengjia Yan. There's Always a Bigger Fish: A Clarifying Analysis of a Machine-Learning-Assisted Side-Channel Attack. In: ISCA. 2022 (p. 47).

[41] Jonathan Corbet. Defending against Rowhammer in the kernel. Oct. 2016. URL: https://lwn.net/Articles/704920/ (pp. 3, 32, 40).

[42] Nikunj A Dadhania. [PATCH v7 00/16] Add Secure TSC support for SNP guests. 2023. URL: https://lore.kernel.org/all/20231220151358.2147066-1-nikunj@amd.com/ (p. 10).

[43] Timothy J. Dell. A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory. Tech. rep. IBM Microelectronics, 1997 (pp. 44, 45).

[44] Department of Defense. Trusted Computer System Evaluation Criteria (TCSEC). 1983 (p. 48).

[45] Andrea Di Dio, Koen Koning, Herbert Bos, and Cristiano Giuffrida. Copy-on-Flip: Hardening ECC Memory Against Rowhammer Attacks. In: NDSS. 2023 (pp. 3, 32, 40).

[46] Cambridge Dictionary, ed. “leaky”. Cambridge University Press, 2024 (p. 3).

[47] Sankha Baran Dutta, Hoda Naghibijouybari, Nael Abu-Ghazaleh, Andres Marquez, and Kevin Barker. Leaky Buddies: Cross-Component Covert Channels on Integrated CPU-GPU Systems. In: ISCA. 2021 (pp. 5, 47).

[48] Sina Sayyah Ensan, Karthikeyan Nagarajan, Mohammad Nasim Imtiaz Khan, and Swaroop Ghosh. SCARE: Side Channel Attack on In-Memory Computing for Reverse Engineering. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2021) (p. 47).

[49] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, Krisztian Flautner, et al. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In: MICRO. 2003 (p. 46).

[50] Dmitry Evtyushkin and Dmitry Ponomarev. Covert Channels Through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations. In: CCS. 2016 (pp. 5, 47).

[51] Ali Fakhrzadehgan, Yale N Patt, Prashant J Nair, and Moinuddin K Qureshi. SafeGuard: Reducing the Security Risk from Row-Hammer via Low-Cost Integrity Protection. In: HPCA. 2022 (pp. 32, 40, 44).

[52] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In: S&P. 2018 (pp. 3, 22, 32, 37–39, 44).

[53] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. TRRespass: Exploiting the Many Sides of Target Row Refresh. In: S&P. 2020 (pp. 3, 4, 11, 22, 25, 32, 34, 37, 38, 41, 42, 44).

[54] Ferraz Gabriel. SSD Database. `https://www.techpowerup.com/ssd-specs/`. 2025 (p. 30).

[55] Stefan Gast, Roland Czerny, Jonas Juffinger, Fabian Rauscher, Simone Franza, and Daniel Gruss. SnailLoad: Exploiting Remote Network Latency Measurements without JavaScript. In: USENIX Security. 2024 (pp. 5, 12, 48).

[56] Stefan Gast, Sebastian Daniel Felix, Alexander Steinmaurer, **Jonas Juffinger**, and Daniel Gruss. Real-World Study of the Security of Educational Test Systems. In: Workshop on Operating Systems and Virtualization Security. 2025 (pp. 5, 12, 13).

[57] Stefan Gast, Jonas Juffinger, Lukas Maar, Christoph Royer, Andreas Kogler, and Daniel Gruss. Remote Scheduler Contention Attacks. In: FC. 2024 (pp. 5, 12, 47, 48).

[58] Stefan Gast, Jonas Juffinger, Martin Schwarzl, Gururaj Saileshwar, Andreas Kogler, Simone Franza, Markus Köstl, and Daniel Gruss. SQUIP: Exploiting the Scheduler Queue Contention Side Channel. In: S&P. 2023 (pp. 5, 10, 12, 47, 48).

[59] Stefan Gast, Hannes Weissteiner, Robin Leander Schröder, and Daniel Gruss. CounterSEVeillance: Performance-Counter Attacks on AMD SEV-SNP. In: NDSS. 2025 (p. 47).

[60] Lukas Gerlach, Simon Schwarz, Nicolas Faroß, and Michael Schwarz. Efficient and generic microarchitectural hash-function recovery. In: S&P. 2024 (p. 25).

[61] Lukas Gerlach, Fabian Thomas, Robert Pietsch, and Michael Schwarz. A Rowhammer Reproduction Study Using the Blacksmith Fuzzer. In: ESORICS. 2023 (p. 35).

[62] Jim Gettys. Bufferbloat: Dark buffers in the internet. In: IEEE Internet Computing 15.3 (2011), pp. 96–96 (p. 48).

[63] Mohsen Ghasempour, Mikel Lujan, and Jim Garside. ARMOR: A Run-time Memory Hot-Row Detector. 2015. URL: `http://apt.cs.manchester.ac.uk/projects/ARMOR/RowHammer` (pp. 32, 40).

[64] Swaroop Ghosh and Kaushik Roy. Parameter Variation Tolerance and Error Resiliency: New Design Paradigm for the Nanoscale Era. In: Proceedings of the IEEE (2010) (p. 7).

[65] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. Cross-VM Covert- and Side-Channel Attacks in Cloud FPGAs. In: ACM Transactions on Reconfigurable Technology and Systems (2022) (pp. 47, 48).

[66] Lukas Giner, Roland Czerny, Christoph Gruber, Fabian Rauscher, Andreas Kogler, Daniel De Almeida Braga, and Daniel Gruss. Generic and Automated Drive-by GPU Cache Attacks from the Browser. In: AsiaCCS. 2024 (pp. 5, 47).

[67] Lukas Giner, Roland Czerny, Simon Lammer, Aaron Giner, Paul Gollob, Jonas Juffiger, and Daniel Gruss. Fast and Efficient Secure L1 Caches for SMT. In: ARES. 2025 (pp. 5, 13).

[68] Dimitris Gizopoulos, George Papadimitriou, Athanasios Chatzidimitriou, Vijay Janapa Reddi, Behzad Salami, Osman S Unsal, Adrian Cristal Kestelman, and Jingwen Leng. Modern hardware margins: CPUs, GPUs, FPGAs recent system-level studies. In: International Symposium on On-Line Testing and Robust System Design (IOLTS). 2019 (pp. 4, 7, 17, 18, 45).

[69] Mel Gorman. Understanding the Linux Virtual Memory Manager. Prentice Hall Upper Saddle River, 2004 (p. 37).

[70] Corey Gough, Ian Steiner, Winston Saunders, Corey Gough, Ian Steiner, and Winston Saunders. CPU Power Management. In: Energy Efficient Servers: Blueprints for Data Center Optimization (2015) (pp. 7, 45).

[71] Sudhakar Govindavajhala and Andrew W Appel. Using Memory Errors to Attack a Virtual Machine. In: S&P. 2003 (p. 3).

[72] Zvika Greenfield and Tomer Levy. Throttling support for row-hammer counters. US Patent 9251885. 2014 (pp. 32, 40, 42).

[73] Daniel Gruss, Erik Kraft, Trishita Tiwari, Michael Schwarz, Ari Trachtenberg, Jason Hennessey, Alex Ionescu, and Anders Fogh. Page Cache Attacks. In: CCS. 2019 (pp. 47, 49).

[74] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. Another Flip in the Wall of Rowhammer Defenses. In: S&P. 2018 (pp. 3, 4, 7, 8, 32, 34, 39, 40, 51).

[75] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In: DIMVA. 2016 (pp. 22, 32, 37, 38, 44).

[76] Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+Flush: A Fast and Stealthy Cache Attack. In: DIMVA. 2016 (pp. 5, 32, 40, 47).

[77] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In: USENIX Security. 2015 (pp. 5, 47).

[78] Berk Gulmezoglu, Andreas Zankl, Thomas Eisenbarth, and Berk Sunar. PerfWeb: How to violate web privacy with hardware performance events. In: ESORICS. 2017 (p. 47).

[79] Mordechai Guri. Exfiltrating data from air-gapped computers via ViBrAtIoNs. In: Future Generation Computer Systems (2021) (p. 47).

[80] Mordechai Guri, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici. Acoustic Data Exfiltration from Speakerless Air-Gapped Computers via Covert Hard-Drive Noise ('DiskFiltration'). In: ESORICS. 2017 (pp. 4, 5, 9, 49).

[81] Jawad Haj-Yahya, Lois Orosa, Jeremie S Kim, Juan Gómez Luna, A Giray Yağlıkçı, Mohammed Alser, Ivan Puddu, and Onur Mutlu. IChannels: Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors. In: ISCA. 2021 (p. 47).

[82] Sarah Harris and David Harris. Digital Design and Computer Architecture. Morgan Kaufmann, 2015 (pp. 4, 16–18, 21).

[83] Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu. Uncovering In-DRAM RowHammer Protection Mechanisms:A New Methodology, Custom RowHammer Patterns, and Implications. In: MICRO. 2021 (pp. 3, 32, 34, 40, 42).

[84] Martin Heckel and Florian Adamsky. Flipper: Rowhammer on Steroids. In: uASC. 2025 (p. 39).

[85] Martin Heckel and Florian Adamsky. Reverse-Engineering Bank Addressing Functions on AMD CPUs. In: DRAMSec Workshop. 2023 (p. 25).

[86] Christian Helm, Soramichi Akiyama, and Kenjiro Taura. Reliable Reverse Engineering of Intel DRAM Addressing Using Performance Counters. In: Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE. 2020 (p. 25).

[87] Nishad Herath and Anders Fogh. These are Not Your Grand Daddys CPU Performance Counters – CPU Hardware Performance Counters for Security. In: Black Hat USA. 2015 (pp. 32, 40).

[88] Seungki Hong, Dongha Kim, Jaehyung Lee, Reum Oh, Changsik Yoo, Sangjoon Hwang, and Jooyoung Lee. DSAC: Low-Cost Rowhammer Mitigation Using In-DRAM Stochastic and Approximate Counting Algorithm. In: arXiv:2302.03591 (2023) (pp. 23, 24, 32, 33, 35, 36, 40–42).

[89] Gal Horowitz, Eyal Ronen, and Yuval Yarom. Spec-o-Scope: Cache Probing at Cache Speed. In: CCS. 2024 (pp. 5, 47).

[90] Wei-Ming Hu. Reducing Timing Channels with Fuzzy Time. In: Journal of Computer Security (1992) (pp. 47, 48).

[91] Ruirui Huang and G. Edward Suh. IVEC: Off-Chip Memory Integrity Protection for Both Security and Reliability. In: ISCA. 2010 (p. 44).

[92] Vincent Huard, Souhir Mhira, A Barclais, X Lecocq, F Raugi, M Cantournet, and Alain Bravaix. Managing electrical reliability in consumer systems for improved energy efficiency. In: IEEE International Reliability Physics Symposium (IRPS). 2018 (pp. 18, 19).

[93] Jisung Im, Hansol Kim, Jinsu Kim, Seungmin Woo, Taeseong Kwon, Young Jun Yoon, Jong-Ho Bae, and Sung Yun Woo. Analysis of Row Hammer and Passing Gate Effect in DRAM Cells by BCAT Structural Design. In: IEEE Silicon Nanoelectronics Workshop (SNW). 2024 (p. 35).

[94] Mehmet Sinan Inci, Berk Gulmezoglu, Thomas Eisenbarth, and Berk Sunar. Co-location detection on the cloud. In: COSADE. 2016 (pp. 10, 49).

[95] Mehmet Sinan Inci, Berk Gulmezoglu, Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud. In: Cryptology ePrint Archive, Report 2015/898 (2015) (p. 49).

[96] Intel. Intel 64 and IA-32 Architectures Optimization Reference Manual. 2023 (pp. 21, 25).

[97] Intel. Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture. 2016 (pp. 20, 21).

[98] Intel. Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3 (3A, 3B & 3C): System Programming Guide. 2024 (pp. 20, 37).

[99] Intel. Understanding the Flash Translation Layer (FTL) Specification. 1998 (p. 28).

[100] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. MASCAT: Stopping Microarchitectural Attacks Before Execution. In: Cryptology ePrint Archive, Report 2016/1196 (2017) (pp. 32, 40).

[101] Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Cross-VM Side Channels and Their Use to Extract Private Keys. In: Big Data and Cloud Computing. 2014 (p. 10).

[102] Aamer Jaleel, Gururaj Saileshwar, Stephen W. Keckler, and Moinuddin Qureshi. PrIDE: Achieving Secure Rowhammer Mitigation with Low-Cost In-DRAM Trackers. In: ISCA. 2024 (pp. 32, 40–42).

[103] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In: SysTEX. 2017 (pp. 32, 37, 39).

[104] Patrick Jattke, Michele Marazzi, Flavien Solt, Max Wipfli, and Stefan Gloor Kaveh Razavi. MCSEE: Evaluating Advanced Rowhammer Attacks and Defenses via Automated DRAM Traffic Analysis. In: Usenix Security. 2025 (pp. 36, 43, 45).

[105] Patrick Jattke, Victor van der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. BLACKSMITH: Rowhammering in the Frequency Domain. In: S&P. 2021 (pp. 3, 4, 11, 32, 34, 35, 41, 42, 51).

[106] Patrick Jattke, Max Wipfli, Flavien Solt, Michele Marazzi, Matej Bölcskei, and Kaveh Razavi. ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms. In: USENIX Security. 2024 (pp. 25, 39, 51).

[107] JEDEC Solid State Technology Association. DDR4 SDRAM Standard. 2021. URL: https://www.jedec.org/standards-documents/docs/jesd79-4a (pp. 24, 25).

[108] JEDEC Solid State Technology Association. DDR5 SDRAM Standard. 2024. URL: https://www.jedec.org/standards-documents/docs/jesd79-5c01 (pp. 25, 26, 42, 43).

[109] JEDEC Solid State Technology Association. Low Power Double Data Rate 4. 2017. URL: http://www.jedec.org/standards-documents/docs/jesd209-4b (pp. 32, 39–42).

[110] Supreet Jeloka, Naveen Bharathwaj Akesh, Dennis Sylvester, and David Blaauw. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory. In: IEEE Journal of Solid-State Circuits (2016) (pp. 41, 43).

[111] Qisheng Jiang and Chundong Wang. Sync+Sync: A Covert Channel Built on fsync with Storage. In: USENIX Security. 2024 (pp. 5, 47, 49).

[112] Yichen Jiang, Huifeng Zhu, Haoqi Shan, Xiaolong Guo, Xuan Zhang, and Yier Jin. TRRScope: Understanding Target Row Refresh Mechanism for Modern DDR Protection. In: HOST. 2021 (p. 42).

[113] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. A complete key recovery timing attack on a GPU. In: HPCA. 2016 (pp. 5, 47).

[114] Biresh Kumar Joardar, Tyler K Bletsch, and Krishnendu Chakrabarty. Machine Learning-Based Rowhammer Mitigation. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2022) (pp. 32, 40, 41, 43).

[115] Jonas Juffinger. An Analysis of HMB-based SSD Rowhammer. In: uASC. 2025 (pp. 5, 6, 8, 29, 32, 39, 51).

[116] Jonas Juffinger, Stepan Kalinin, Daniel Gruss, and Frank Mueller. SUIT: Secure Undervolting with Instruction Traps. In: ASPLOS. 2024 (pp. 5–7, 45, 46, 52).

[117] Jonas Juffinger, Lukas Lamster, Andreas Kogler, Maria Eichlseder, Moritz Lipp, and Daniel Gruss. CSI: Rowhammer - Cryptographic Security and Integrity against Rowhammer. In: S&P. 2023 (pp. 5–7, 32, 40, 44, 51, 52).

[118] Jonas Juffinger, Sudheendra Raghav Neela, and Daniel Gruss. Not So Secure TSC. In: ACNS. 2025 (pp. 5, 6, 9, 10, 50, 53).

[119] Jonas Juffinger, Sudheendra Raghav Neela, Martin Heckel, Lukas Schwarz, Florian Adamsky, and Daniel Gruss. Presshammer: Rowhammer and Rowpress without Physical Address Information. In: DIMVA. 2024 (pp. 5–8, 22, 32, 34, 36–38, 44, 51).

[120] Jonas Juffinger, Fabian Rauscher, Giuseppe La Manna, and Daniel Gruss. Secret Spilling Drive: Leaking User Behavior through SSD Contention. In: NDSS. 2025 (pp. 5, 6, 9, 48, 49, 52).

[121] Jonas Juffinger, Hannes Weissteiner, Thomas Steinbauer, and Daniel Gruss. The HMB Timing Side Channel: Exploiting the SSD's Host Memory Buffer. In: DIMVA. 2025 (pp. 5, 6, 8, 9, 29, 30, 52).

[122] Matthias Jung, Carl C Rheinländer, Christian Weis, and Norbert Wehn. Reverse engineering of DRAMs: Row hammer with crosshair. In: International Symposium on Memory Systems. 2016 (p. 25).

[123] Marcin Kaczmarski. Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – component selection guidelines. Infobazy 2014. Aug. 2014. URL: https://web.archive.org/web/20240418012923/http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf (pp. 32, 40, 41).

[124] Manolis Kaliorakis, Athanasios Chatzidimitriou, George Papadimitriou, and Dimitris Gizopoulos. Statistical analysis of multicore CPUs operation in scaled voltage conditions. In: IEEE Computer Architecture Letters (2018) (pp. 4, 7, 45).

[125] Ingab Kang, Walter Wang, Jason Kim, Stephan van Schaik, Youssef Tobah, Daniel Genkin, Andrew Kwong, and Yuval Yarom. SledgeHammer: Amplifying Rowhammer via Bank-level Parallelism. In: USENIX Security. 2024 (pp. 3, 11, 22, 32, 37, 38, 44, 51).

[126] Jeong-Uk Kang, Heeseung Jo, Jin-Soo Kim, and Joonwon Lee. A Superblock-based Flash Translation Layer for NAND Flash Memory. In: International Conference on Embedded Software. 2006 (p. 28).

[127] David Kaplan, Jeremy Powell, and Tom Woller. AMD Memory Encryption. 2016 (p. 10).

[128] Paul A Karger and John C Wray. Storage Channels in Disk Arm Optimization. In: S&P. 1991 (pp. 5, 47–49).

[129] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda. A Flash-Memory Based File System. In: USENIX. 1995 (p. 28).

[130] Zijo Kenjar, Tommaso Frassetto, David Gens, Michael Franz, and Ahmad-Reza Sadeghi. V0LTpwn: Attacking x86 Processor Integrity from Software. In: USENIX Security. 2020 (pp. 4, 7, 45, 46).

[131] Dae-Hyun Kim, Prashant J Nair, and Moinuddin K Qureshi. Architectural support for mitigating row hammering in DRAM memories. In: IEEE Computer Architecture Letters 14 (2015) (pp. 32, 40, 42).

[132] Jeremie S. Kim, Minesh Patel, A. Giray Yağlıkçı, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu. Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques. In: ISCA. 2020 (p. 39).

[133] Kyusik Kim and Taeseok Kim. HMB in DRAM-less NVMe SSDs: Their usage and effects on performance. In: PloS one (2020) (p. 29).

[134] Michael Jaemin Kim, Jaehyun Park, Yeonhong Park, Wanju Doh, Namhoon Kim, Tae Jun Ham, Jae W Lee, and Jung Ho Ahn. Mithril: Cooperative Row Hammer Protection on Commodity DRAM Leveraging Managed Refresh. In: HPCA. 2022 (pp. 41, 42).

[135] Taesoo Kim, Marcus Peinado, and Gloria Mainar-Ruiz. StealthMem: system-level protection against cache-based side channel attacks in the cloud. In: USENIX Security. 2012 (p. 49).

[136] Woongrae Kim, Chulmoon Jung, Seongnyuh Yoo, Duckhwa Hong, Jeongjin Hwang, Jungmin Yoon, Ohyong Jung, Joonwoo Choi, Sanga Hyun, Mankeun Kang, et al. A 1.1V 16Gb DDR5 DRAM with Probabilistic-Aggressor Tracking, Refresh-Management Functionality, Per-Row Hammer Tracking, a Multi-Step Precharge, and Core-Bias Modulation for Security and Reliability Enhancement. In: International Solid-State Circuits Conference (ISSCC). IEEE. 2023 (p. 41).

[137] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In: ISCA. 2014 (pp. 3, 8, 24, 32–34, 39–41).

[138] Kirill A. Shutemov. Pagemap: Do Not Leak Physical Addresses to Non-Privileged Userspace. 2015. URL: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=ab676b7d6fbf4b294bf198fb27ade5b0e865c7ce (p. 37).

[139] Paul Kocher. Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems. In: CRYPTO. 1996 (pp. 4, 5, 47).

[140] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. In: S&P. 2019 (p. 47).

[141] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In: CRYPTO. 1999 (p. 4).

[142] Andreas Kogler, Daniel Gruss, and Michael Schwarz. Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks. In: USENIX Security. 2022 (pp. 45, 46).

[143] Andreas Kogler, Jonas Juffinger, Lukas Giner, Lukas Gerlach, Martin Schwarzl, Michael Schwarz, Daniel Gruss, and Stefan Mangard. Collide+Power: Leaking Inaccessible Data with Software-based Power Side Channels. In: USENIX Security. 2023 (pp. 5, 11, 47).

[144] Andreas Kogler, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. Half-Double: Hammering From the Next Row Over. In: USENIX Security. 2022 (pp. 3–5, 11, 22, 32, 34, 35, 37, 38, 42, 44, 45, 51).

[145] Radhesh Krishnan Konoth, Marco Oliverio, Andrei Tatar, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks. In: USENIX OSDI. 2018 (pp. 3, 32, 40).

[146] Panos Koutsovasilis, Christos D Antonopoulos, Nikolaos Bellas, Spyros Lalis, George Papadimitriou, Athanasios Chatzidimitriou, and Dimitris Gizopoulos. The Impact of CPU Voltage Margins on Power-Constrained Execution. In: IEEE Transactions on Sustainable Computing (2020) (pp. 4, 7, 45).

[147] Panos Koutsovasilis, Konstantinos Parasyris, Christos D Antonopoulos, Nikolaos Bellas, and Spyros Lalis. Dynamic undervolting to improve energy efficiency on multicore x86 cpus. In: IEEE Transactions on Parallel and Distributed Systems (2020) (pp. 4, 7, 46).

[148] A Anand Kumar. Fundamentals of Digital Circuits. 2016 (pp. 7, 16).

[149] Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom. RAMBleed: Reading Bits in Memory Without Accessing Them. In: S&P. 2020 (pp. 32, 37, 39).

[150] Butler W Lampson. A note on the confinement problem. In: Communications of the ACM (1973) (pp. 6, 47, 48, 53).

[151] Eojin Lee, Ingab Kang, Sukhan Lee, G Edward Suh, and Jung Ho Ahn. TWiCe: preventing row-hammering by exploiting time window counters. In: ISCA. 2019 (pp. 32, 40, 41, 43).

[152] Joo Hwan Lee, Hui Zhang, Veronica Lagrange, Praveen Krishnamoorthy, Xiaodong Zhao, and Yang Seok Ki. SmartSSD: FPGA Accelerated Near-Storage Data Analytics on SSD. In: IEEE Computer Architecture Letters (2020) (p. 49).

[153] Lanny L Lewyn, Trond Ytterdal, Carsten Wulff, and Kenneth Martin. Analog Circuit Design in Nanoscale CMOS Technologies. In: Proceedings of the IEEE (2009) (p. 7).

[154] Linux. madvise(2) — Linux manual page. https://man7.org/linux/man-pages/man2/madvise.2.html. 2025 (p. 37).

[155] Bartosz Lipinski, Wojciech Mazurczyk, and Krzysztof Szczypiorski. Improving Hard Disk Contention-based Covert Channel in Cloud Computing Environment. In: S&P Workshops. 2014 (pp. 5, 9, 48, 49).

[156] Moritz Lipp, Misiker Tadesse Aga, Michael Schwarz, Daniel Gruss, Clémentine Maurice, Lukas Raab, and Lukas Lamster. Nethammer: Inducing Rowhammer Faults through Network Requests. In: SILM Workshop. 2020 (pp. 3, 38).

[157] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In: S&P. 2021 (pp. 4, 5, 11, 47).

[158] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In: USENIX Security. 2018 (p. 47).

[159] Chen Liu, Abhishek Chakraborty, Nikhil Chawla, and Neer Roggel. Frequency Throttling Side-Channel Attack. In: CCS. 2022 (pp. 5, 47).

[160] Fangfei Liu, Qian Ge, Yuval Yarom, Frank Mckeen, Carlos Rozas, Gernot Heiser, and Ruby B Lee. CATalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing. In: HPCA. 2016 (p. 49).

[161] Sihang Liu, Suraaj Kanniwadi, Martin Schwarzl, Andreas Kogler, Daniel Gruss, and Samira Khan. Side-Channel Attacks on Optane Persistent Memory. In: USENIX Security. 2023 (pp. 6, 49).

[162] Keith Loepere. Resolving covert channels within a B2 class secure system. In: ACM SIGOPS Operating Systems Review (1985) (pp. 47, 48).

[163] Haocong Luo, Ataberk Olgun, Abdullah Giray Yağlıkçı, Yahya Can Tuğrul, Steve Rhyner, Meryem Banu Cavlak, Joël Lindegger, Mohammad Sadrosadati, and Onur Mutlu. RowPress: Amplifying Read Disturbance in Modern DRAM Chips. In: ISCA. 2023 (pp. 4, 8, 24, 32, 35, 36).

[164] Lukas Maar, Stefan Gast, Martin Unterguggenberger, Mathias Oberhuber, and Stefan Mangard. SLUBStick: Arbitrary Memory Writes through Practical Software Cross-Cache Attacks within the Linux Kernel. In: USENIX Security. 2024 (pp. 5, 47).

[165] Lukas Maar, Jonas Juffinger, Thomas Steinbauer, Daniel Gruss, and Stefan Mangard. KernelSnitch: Side-Channel Attacks on Kernel Data Structures. In: NDSS. 2025 (pp. 5, 12, 47).

[166] Hosein Mohammadi Makrani, Hossein Sayadi, Najmeh Nazari, Khaled N Khasawneh, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. Cloak & Co-locate: Adversarial Railroading of Resource Sharing-based Attacks on the Cloud. In: Secure and Private Execution Environment Design (SEED). 2021 (pp. 49, 50).

[167] Aniruddha Marathe, Yijia Zhang, Grayson Blanks, Nirmal Kumbhare, Ghaleb Abdulla, and Barry Rountree. An empirical survey of performance and energy efficiency variation on Intel processors. In: International Workshop on Energy Efficient Supercomputing. 2017 (p. 7).

[168] Michele Marazzi, Patrick Jattke, Flavien Solt, and Kaveh Razavi. PROTRR: Principled yet Optimal In-DRAM Target Row Refresh. In: S&P. 2022 (pp. 3, 41, 42).

[169] Michele Marazzi and Kaveh Razavi. RISC-H: Rowhammer Attacks on RISC-V. In: DRAMSec Workshop. 2024 (pp. 3, 39).

[170] Michele Marazzi, Tristan Sachsenweger, Flavien Solt, Peng Zeng, Kubo Takashi, Maksym Yarema, and Kaveh Razavi. HiFi-DRAM: Enabling High-fidelity DRAM Research by Uncovering Sense Amplifiers with IC Imaging. In: ISCA. 2024 (pp. 23, 26).

[171] Michele Marazzi, Flavien Solt, Patrick Jattke, Kubo Takashi, and Kaveh Razavi. REGA: Scalable Rowhammer Mitigation with Refresh-Generating Activations. In: S&P. 2023 (pp. 3, 32, 40–43).

[172] Emmanouil Maroudas, Spyros Lalis, Nikolaos Bellas, and Christos D Antonopoulos. Exploring the Potential of Context-Aware Dynamic CPU Undervolting. In: ACM International Conference on Computing Frontiers. 2021 (pp. 4, 7, 46).

[173] Marsan, Balbo, Conte, and Gregoretti. Modeling Bus Contention and Memory Interference in a Multiprocessor System. In: IEEE Transactions on Computers (1983) (p. 48).

[174] Nikolay Matyunin, Yujue Wang, Tolga Arul, Kristian Kullmann, Jakub Szefer, and Stefan Katzenbeisser. MagneticSpy: Exploiting Magnetometer in Mobile Devices for Website and Application Fingerprinting. In: ACM Workshop on Privacy in the Electronic Society. 2019 (pp. 5, 47).

[175] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. C5: Cross-Cores Cache Covert Channel. In: DIMVA. 2015 (p. 47).

[176] Clémentine Maurice, Manuel Weber, Michael Schwarz, Lukas Giner, Daniel Gruss, Carlo Alberto Boano, Stefan Mangard, and Kay Römer. Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. In: NDSS. 2017 (pp. 5, 10, 47, 49).

[177] Mozilla. HTTP caching — MDN. Dec. 2024. URL: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching` (p. 9).

[178] David Mulnix. Intel Xeon Processor Scalable Family Technical Overview. `https://www.intel.com/content/www/us/en/developer/articles/technical/xeon-processor-scalable-family-technical-overview.html`. 2022 (p. 21).

[179] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In: S&P. 2020 (pp. 3, 4, 7, 17, 45, 46).

[180] Koksal Mus, Saad Islam, and Berk Sunar. QuantumHammer: A Practical Hybrid Attack on the LUOV Signature Scheme. In: CCS. 2020 (p. 39).

[181] Onur Mutlu, Ataberk Olgun, and A Giray Yağlıkcı. Fundamentally Understanding and Solving RowHammer. In: Asia and South Pacific Design Automation Conference. 2023 (p. 39).

[182] Onur Mutlu, Ataberk Olgun, and İsmail Emir Yüksel. Memory-Centric Computing: Solving Computing's Memory Problem. In: International Memory Workshop (IMW). 2025 (p. 51).

[183] Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu-Ghazaleh. Rendered Insecure: GPU Side Channel Attacks are Practical. In: CCS. 2018 (pp. 5, 47).

[184] Northland Locksmith. Safe lock manipulation. `https://web.archive.org/web/20161222161226/https://northlandlocksmith.com/2016/12/09/safe-lock-manipulation/`. 2016 (p. 4).

[185] NVM Express, Inc. NVM Express, rev 1.2.1. 2016 (pp. 8, 29).

[186] Mathias Oberhuber, Martin Unterguggenberger, Lukas Maar, Andreas Kogler, and Stefan Mangard. Power-Related Side-Channel Attacks using the Android Sensor Framework. In: NDSS. 2025 (pp. 5, 47).

[187] Vojin G Oklobdzija, Vladimir M Stojanovic, Dejan M Markovic, and Nikola M Nedovic. Digital System Clocking: High-Performance and Low-Power Aspects. Wiley-IEEE Press, 2005. ISBN: 9780471274476 (p. 18).

[188] Ataberk Olgun, F. Nisa Bostanci, Ismail Emir Yuksel, Oguzhan Canpolat, Haocong Luo, Geraldo F. Oliveira, A. Giray Yaglikci, Minesh Patel, and Onur Mutlu. Variable Read Disturbance: An Experimental Analysis of Temporal Variation in DRAM Read Disturbance. In: HPCA. 2025 (pp. 4, 32).

[189] Ataberk Olgun, Yahya Can Tugrul, Nisa Bostanci, Ismail Emir Yuksel, Haocong Luo, Steve Rhyner, Abdullah Giray Yaglikci, Geraldo F Oliveira, and Onur Mutlu. ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation. In: USENIX Security. 2024 (pp. 32, 41, 42).

[190] Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu. A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses. In: MICRO. 2021 (p. 3).

[191] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: the Case of AES. In: CT-RSA. 2006 (pp. 4, 5, 47).

[192] George Papadimitriou, Athanasios Chatzidimitriou, and Dimitris Gizopoulos. Adaptive Voltage/Frequency Scaling and Core Allocation for Balanced Energy and Performance on Multicore CPUs. In: HPCA. 2019 (p. 7).

[193] George Papadimitriou, Manolis Kaliorakis, Athanasios Chatzidimitriou, Dimitris Gizopoulos, Peter Lawthers, and Shidhartha Das. Harnessing Voltage Margins for Energy Efficiency in Multicore CPUs. In: MICRO. 2017 (pp. 4, 7, 45).

[194] George Papadimitriou, Manolis Kaliorakis, Athanasios Chatzidimitriou, Charalampos Magdalinos, and Dimitris Gizopoulos. Voltage margins identification on commercial x86-64 multicore microprocessors. In: IEEE Symposium on On-Line Testing (IOLTS). 2017 (pp. 4, 7, 45).

[195] Kyungbae Park, Chulseung Lim, Donghyuk Yun, and Sanghyeon Baeg. Experiments and root cause analysis for active-precharge hammering fault in DDR3 SDRAM under 3× nm technology. In: Microelectronics Reliability (2016) (p. 33).

[196] Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, and Jae W Lee. Graphene: Strong yet Lightweight Row Hammer Protection. In: MICRO. 2020 (pp. 32, 41, 43, 44).

[197] Michael K Patterson. The Effect of Data Center Temperature on Energy Efficiency. In: Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems. 2008 (pp. 7, 45).

[198] Matthias Payer. HexPADS: a platform to detect "stealth" attacks. In: ESSoS. 2016 (pp. 32, 40).

[199] Colin Percival. Cache Missing for Fun and Profit. In: BSDCan. 2005 (pp. 4, 47).

[200] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. In: USENIX Security. 2016 (pp. 25, 26, 37, 47).

[201] Florian Pfarr, Thomas Buckel, and Axel Winkelmann. Cloud Computing Data Protection – A Literature Review and Analysis. In: HICSS. 2014 (p. 10).

[202] Antoon Purnal, Marton Bognar, Frank Piessens, and Ingrid Verbauwhede. ShowTime: Amplifying Arbitrary CPU Timing Side Channels. In: AsiaCCS. 2023 (pp. 5, 47).

[203] Antoon Purnal, Furkan Turan, and Ingrid Verbauwhede. Double Trouble: Combined Heterogeneous Attacks on Non-Inclusive Cache Hierarchies. In: USENIX Security. 2022 (p. 47).

[204] Antoon Purnal, Furkan Turan, and Ingrid Verbauwhede. Prime+Scope: Overcoming the Observer Effect for High-Precision Cache Contention Attacks. In: CCS. 2021 (pp. 5, 47).

[205] Salman Qazi and Daniel Moghimi. SoothSayer: Bypassing DSAC Mitigation by Predicting Counter Replacement. In: DRAMSec. 2024 (p. 42).

[206] Rui Qiao and Mark Seaborn. A New Approach for Rowhammer Attacks. In: HOST. 2016 (pp. 32, 39).

[207] Yi Qin and Chuan Yue. Website Fingerprinting by Power Estimation Based Side-Channel Attacks on Android 7. In: TrustCom/BigDataSE. 2018 (p. 47).

[208] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-core Frequencies. In: CCS. 2019 (pp. 3, 4, 7, 45).

[209] Moinuddin Qureshi and Salman Qazi. MOAT: Securely Mitigating Rowhammer with Per-Row Activation Counters. In: ASPLOS. 2025 (pp. 41, 43).

[210] Moinuddin Qureshi, Salman Qazi, and Aamer Jaleel. MINT: Securely Mitigating Rowhammer with a Minimalist In-DRAM Tracker. In: MICRO. 2024 (pp. 32, 41, 42).

[211] Fabian Rauscher, Carina Fiedler, Andreas Kogler, and Daniel Gruss. A Systematic Evaluation of Novel and Existing Cache Side Channels. In: NDSS. 2025 (pp. 5, 47).

[212] Fabian Rauscher, Andreas Kogler, Jonas Juffinger, and Daniel Gruss. IdleLeak: Exploiting Idle State Side Effects for Information Leakage. In: NDSS. 2024 (pp. 5, 11, 47).

[213] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip Feng Shui: Hammering a Needle in the Software Stack. In: USENIX Security. 2016 (pp. 32, 37, 39).

[214] Randall Rettberg and Robert Thomas. Contention is no obstacle to shared-memory multiprocessing. In: Communications of the ACM (1986) (p. 48).

[215] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. SMASH: Synchronized Many-sided Rowhammer Attacks From JavaScript. In: USENIX Security. 2021 (pp. 3, 32, 37, 38).

[216] Finn de Ridder, Patrick Jattke, and Kaveh Razavi. Posthammer: Pervasive Browser-based Rowhammer Attacks with Postponed Refresh Commands. In: USENIX Security. 2025 (p. 3).

[217] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In: CCS. 2009 (p. 49).

[218] Thomas Rokicki, Clémentine Maurice, Marina Botvinnik, and Yossi Oren. Port Contention Goes Portable: Port Contention Side Channels in Web Browsers. In: AsiaCCS. 2022 (pp. 47, 48).

[219] Seong-Wan Ryu, Kyungkyu Min, Jungho Shin, Heimi Kwon, Donghoon Nam, Taekyung Oh, Tae-Su Jang, Minsoo Yoo, Yongtaik Kim, and Sungjoo Hong. Overcoming the Reliability Limitation in the Ultimately Scaled DRAM Using Silicon Migration Technique by Hydrogen Annealing. In: International Electron Devices Meeting (IEDM). 2017 (p. 33).

[220] Gururaj Saileshwar, Prashant J Nair, Prakash Ramrakhyani, Wendy Elsasser, and Moinuddin K Qureshi. SYNERGY: Rethinking Secure-Memory Design for Error-Correcting Memories. In: HPCA. 2018 (pp. 32, 44).

[221] Gururaj Saileshwar, Bolin Wang, Moinuddin Qureshi, and Prashant J Nair. Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation between Aggressor and Victim Rows. In: ASPLOS. 2022 (pp. 32, 41, 44).

[222] Stefan Saroiu. DDR5 Spec Update Has All It Needs to End Rowhammer: Will It? Apr. 2024. URL: https://web.archive.org/web/20240628014749/https://stefan.t8k2.com/rh/PRAC/index.html (p. 42).

[223] Anish Saxena, Aamer Jaleel, and Moinuddin Qureshi. ImPress: Securing DRAM Against Data-Disturbance Errors via Implicit Row-Press Mitigation. In: MICRO. 2024 (pp. 32, 41, 42).

[224] Anish Saxena, Gururaj Saileshwar, Jonas Juffinger, Andreas Kogler, Daniel Gruss, and Moinuddin Qureshi. PT-Guard: Integrity-Protected Page Tables to Defend Against Breakthrough Rowhammer Attacks. In: DSN. 2023 (pp. 5, 11, 32, 41, 44).

[225] Anish Saxena, Gururaj Saileshwar, Prashant J Nair, and Moinuddin Qureshi. AQUA: Scalable Rowhammer Mitigation by Quarantining Aggressor Rows at Runtime. In: MICRO. 2022 (pp. 32, 41, 44).

[226] Marvin Schaefer, Barry Gold, Richard Linde, and John Scheid. Program Confinement in KVM/370. In: ACM National Conference. 1977 (pp. 47, 48).

[227] Christian Schlünder, Stefano Aresu, Georg Georgakos, Werner Kanert, Hans Reisinger, Karl Hofmann, and Wolfgang Gustin. HCI vs. BTI?-Neither one's out. In: IEEE International Reliability Physics Symposium (IRPS). 2012 (p. 19).

[228] Michael Schwarz, Clémentine Maurice, Daniel Gruss, and Stefan Mangard. Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript. In: FC. 2017 (p. 47).

[229] Mark Seaborn and Thomas Dullien. Exploiting the DRAM Rowhammer bug to gain kernel privileges. In: Black Hat USA. 2015 (pp. 3, 11, 32, 34, 38).

[230] Seyed Mohammad Seyedzadeh, Alex K Jones, and Rami Melhem. Mitigating Wordline Crosstalk using Adaptive Trees of Counters. In: ISCA. 2018 (pp. 32, 41, 43).

[231] Carlton Shepherd, Jan Kalbantner, Benjamin Semal, and Konstantinos Markantonakis. A Side-channel Analysis of Sensor Multiplexing for Covert Channels and Application Fingerprinting on Mobile Devices. In: arXiv:2110.06363 (2021) (pp. 5, 47).

[232] Anand Shimpi. Intel Iris Pro 5200 Graphics Review: Core i7-4950HQ Tested. June 2013. URL: https://www.anandtech.com/show/6993/intel-iris-pro-5200-graphics-review-core-i74950hq-tested/3 (p. 21).

[233] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. Robust Website Fingerprinting Through The Cache Occupancy Channel. In: USENIX Security. 2019 (p. 5).

[234] Mert Side, Fan Yao, and Zhenkai Zhang. LockedDown: Exploiting Contention on Host-GPU PCIe Bus for Fun and Profit. In: Euro S&P. 2022 (pp. 47, 48).

[235] SN Singh. Basic Electrical Engineering. PHI Learning Pvt. Ltd., 2010 (p. 18).

[236] Sergei P Skorobogatov and Ross J Anderson. Optical Fault Induction Attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. 2002 (p. 3).

[237] Alan Jay Smith. Design of CPU Cache Memories. Computer Science Division, University of California, 1987 (p. 20).

[238] Mungyu Son, Hyunsun Park, Junwhan Ahn, and Sungjoo Yoo. Making DRAM Stronger Against Row Hammering. In: Design Automation Conference (DAC). 2017 (pp. 32, 41).

[239] Mary-Jane Sule, Maozhen Li, and Gareth Taylor. Trust Modeling in Cloud Computing. In: Symposium on Service-Oriented System Engineering (SOSE). 2016 (p. 10).

[240] Dean Sullivan, Orlando Arias, Travis Meade, and Yier Jin. Microarchitectural Minefields: 4K-aliasing Covert Channel and Multitenant Detection in IaaS Clouds. In: NDSS. 2018 (pp. 10, 49).

[241] Mingtian Tan, Junpeng Wan, Zhe Zhou, and Zhou Li. Invisible Probe: Timing Attacks with PCIe Congestion Side-Channel. In: S&P. 2021 (pp. 5, 47, 48).

[242] Hritvik Taneja, Jason Kim, Jie Jeff Xu, Stephan van Schaik, Daniel Genkin, and Yuval Yarom. Hot Pixels: Frequency, Power, and Temperature Attacks on GPUs and ARM SoCs. In: USENIX Security. 2023 (pp. 5, 47).

[243] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management. In: USENIX Security. 2017 (pp. 3, 4, 7, 45).

[244] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer. In: RAID. 2018 (pp. 3, 25).

[245] Andrei Tatar, Radhesh Krishnan, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Throwhammer: Rowhammer Attacks over the Network and Defenses. In: USENIX ATC. 2018 (pp. 3, 38).

[246] Bjørn Ivar Teigen, Kai Olav Ellefsen, Tor Skeie, and Jim Torresen. Known Performance Issues Are Prevalent in Consumer WiFi Routers. In: International Conference on Network and Service Management (CNSM). 2021 (p. 48).

[247] The Linux Kernel. Transparent Hugepage Support. 2025. URL: https://docs.kernel.org/admin-guide/mm/transhuge.html (p. 37).

[248] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. Cloud FPGA Cartography using PCIe Contention. In: International Symposium on Field-Programmable Custom Computing Machines (FCCM). 2021 (pp. 47, 48).

[249] Youssef Tobah, Andrew Kwong, Ingab Kang, Daniel Genkin, and Kang G Shin. SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks. In: S&P. 2022 (pp. 32, 37, 39).

[250] Theodoros Trochatos, Anthony Etim, and Jakub Szefer. Covert-channels in FPGA-enabled SmartSSDs. In: ACM Transactions on Reconfigurable Technology and Systems (2023) (pp. 6, 47, 49).

[251] Theodoros Trochatos, Anthony Etim, and Jakub Szefer. Security Evaluation of Thermal Covert-channels on SmartSSDs. In: arXiv:2305.09115 (2023) (pp. 6, 47, 49).

[252] C-R Tsai, Virgil D. Gligor, and C. Sekar Chandersekaran. On the Identification of Covert Storage Channels in Secure Systems. In: IEEE Transactions on Software Engineering (1990) (pp. 47, 48).

[253] Yahya Can Tuğrul, A. Giray Yağlıkçı, İsmail Emir Yüksel, Ataberk Olgun, Oğuzhan Canpolat, Nisa Bostancı, Mohammad Sadrosadati, Oğuz Ergin, and Onur Mutlu. Understanding RowHammer Under Reduced Refresh Latency: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions. In: 2025 (pp. 32, 41, 42, 44).

[254] Dean M Tullsen, Susan J Eggers, and Henry M Levy. Simultaneous Multithreading: Maximizing On-Chip Parallelism. In: ISCA. 1995 (p. 48).

[255] John P Uyemura. CMOS Logic Circuit Design. Springer Science & Business Media, 1999 (pp. 16–18).

[256] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In: CCS. 2016 (pp. 4, 11, 22, 32, 37–40, 44).

[257] Victor van der Veen, Martina Lindorfer, Yanick Fratantonio, Harikrishnan Padmanabha Pillai, Giovanni Vigna, Christopher Kruegel, Herbert Bos, and Kaveh Razavi. GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM. In: DIMVA. 2018 (pp. 3, 32, 40).

[258] Saru Vig, Siew-Kei Lam, Sarani Bhattacharya, and Debdeep Mukhopadhyay. Rapid Detection of RowHammer Attacks using Dynamic Skewed Hash Tree. In: Workshop on Hardware and Architectural Support for Security and Privacy (HASP). 2018 (pp. 32, 41, 43).

[259] John Von Neumann. First Draft of a Report on the EDVAC. In: IEEE Annals of the History of Computing (1993) (p. 20).

[260] Andrew J Walker, Sungkwon Lee, and Dafna Beery. On DRAM Rowhammer and the Physics of Insecurity. In: IEEE Transactions on Electron Devices (2021) (p. 33).

[261] Minghua Wang, Zhi Zhang, Yueqiang Cheng, and Surya Nepal. DRAMDig: A Knowledge-assisted Tool to Uncover DRAM Address Mapping. In: Design Automation Conference (DAC). 2020 (p. 25).

[262] Yingchen Wang, Riccardo Paccagnella, Zhao Gang, Willy R Vasquez, David Kohlbrenner, Hovav Shacham, and Christopher W Fletcher. GPU. zip: On the Side-Channel Implications of Hardware-Based Graphical Data Compression. In: S&P (2024) (pp. 5, 47).

[263] Yingchen Wang, Riccardo Paccagnella, Elizabeth He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86. In: USENIX Security. 2022 (pp. 5, 7, 11, 47).

[264] Yingchen Wang, Riccardo Paccagnella, Alan Wandke, Zhao Gang, Grant Garrett-Grossman, Christopher W Fletcher, David Kohlbrenner, and Hovav Shacham. DVFS Frequently Leaks Secrets: Hertzbleed Attacks Beyond SIKE, Cryptography, and CPU-Only Data. In: S&P. 2023 (pp. 5, 11, 47).

[265] Zhenghong Wang and Ruby B Lee. Covert and Side Channels due to Processor Architecture. In: ACSAC. 2006 (p. 47).

[266] Ziyu Wang, Fan-hsuan Meng, Yongmo Park, Jason K Eshraghian, and Wei D Lu. Side-Channel Attack Analysis on In-Memory Computing Architectures. In: IEEE Transactions on Emerging Topics in Computing (2023) (p. 47).

[267] Andrew Waterman and Krste Asanović. The RISC-V Instruction Set Manual, Vol. I: Unprivileged ISA, Version 20191213. 2019 (p. 20).

[268] Junyi Wei, Yicheng Zhang, Zhe Zhou, Zhou Li, and Mohammad Abdullah Al Faruque. Leaky DNN: Stealing Deep-Learning Model Secret with GPU Context-Switching Side-Channel. In: DSN. 2020 (pp. 5, 47).

[269] Zane Weissman, Thore Tiemann, Daniel Moghimi, Evan Custodio, Thomas Eisenbarth, and Berk Sunar. JackHammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms. In: arXiv:1912.11523 (2019) (p. 39).

[270] Hannes Weissteiner, Fabian Rauscher, Robin Leander Schröder, Jonas Juffinger, Stefan Gast, Jan Wichelmann, Thomas Eisenbarth, and Daniel Gruss. TEEcorrelate: An Information-Preserving Defense against Performance Counter Attacks on TEEs. In: USENIX Security. 2025 (pp. 5, 13).

[271] Jeonghyun Woo, Shaopeng Chris Lin, Prashant J Nair, Aamer Jaleel, and Gururaj Saileshwar. QPRAC: Towards Secure and Practical PRAC-based Rowhammer Mitigation using Priority Queues. In: HPCA. IEEE. 2025 (pp. 41, 43).

[272] Jeonghyun Woo, Gururaj Saileshwar, and Prashant J Nair. Scalable and Secure Row-Swap: Efficient and Safe Row Hammer Mitigation in Memory Systems. In: HPCA. 2023 (pp. 41, 44).

[273] Zhenyu Wu, Zhang Xu, and Haining Wang. Whispers in the Hyper-space: High-bandwidth and Reliable Covert Channel Attacks inside the Cloud. In: ACM Transactions on Networking (2014) (pp. 47, 48).

[274] Zhenyu Wu, Zhang Xu, and Haining Wang. Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud. In: USENIX Security. 2012 (pp. 10, 47–49).

[275] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation. In: USENIX Security. 2016 (pp. 22, 32, 37, 38, 44).

[276] A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu. BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows. In: HPCA. 2021 (pp. 32, 41–43).

[277] A. Giray Yaglikci, Yahya Can Tuğrul, Geraldo F De Oliviera, Ismail Emir Yüksel, Ataberk Olgun, Haocong Luo, and Onur Mutlu. Spatial Variation-Aware Read Disturbance Defenses: Experimental Analysis of Real DRAM Chips and Implications on Future Solutions. In: HPCA. 2024 (pp. 4, 32).

[278] Thomas Yang and Xi-Wei Lin. Trap-Assisted DRAM Row Hammer Effect. In: IEEE Electron Device Letters (2019) (p. 33).

[279] Yuval Yarom and Katrina Falkner. Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack. In: USENIX Security. 2014 (pp. 5, 47).

[280] Chi-Weon Yoon. The Fundamentals of NAND Flash Memory: Technology for tomorrow's fourth industrial revolution. In: IEEE Solid-State Circuits Magazine (2022) (pp. 27, 28).

[281] Jung Min You and Joon-Sung Yang. MRLoc: Mitigating Rowhammering based on memory Locality. In: Design Automation Conference (DAC). 2019 (pp. 32, 41).

[282] Ismail Emir Yuksel, Akash Sood, Ataberk Olgun, Oguzhan Canpolat, Haocong Luo, Nisa Bostanci, Mohammad Sadrosadati, A. Giray Yaglikci, and Onur Mutlu. PuDHammer: Experimental Analysis of Read Disturbance Effects of Processing-using-DRAM in Real DRAM Chips. In: ISCA. 2025 (p. 51).

[283] Ruiyi Zhang, Taehyun Kim, Daniel Weber, and Michael Schwarz. (M)WAIT for It: Bridging the Gap between Microarchitectural and Architectural Side Channels. In: USENIX Security. 2023 (pp. 5, 47).

[284] Tianwei Zhang, Yinqian Zhang, and Ruby B. Lee. CloudRadar: A Real-Time Side-Channel Attack Detection System in Clouds. In: RAID. 2016 (pp. 32, 40, 49).

[285] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter. HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis. In: S&P. 2011 (pp. 10, 49).

[286] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-Tenant Side-Channel Attacks in PaaS Clouds. In: CCS. 2014 (pp. 10, 49).

[287] Zhi Zhang, Yueqiang Cheng, Dongxi Liu, Surya Nepal, and Zhi Wang. TeleHammer: Cross-Privilege-Boundary Rowhammer through Implicit Accesses. In: arXiv:1912.03076 (2019) (p. 38).

[288] Zhi Zhang, Yueqiang Cheng, Dongxi Liu, Surya Nepal, Zhi Wang, and Yuval Yarom. PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses. In: MICRO. 2020 (pp. 4, 11, 22, 32, 37, 38, 44).

[289] Zirui Neil Zhao, Adam Morrison, Christopher W Fletcher, and Josep Torrellas. Everywhere All at Once: Co-Location Attacks on Public Cloud FaaS. In: ASPLOS. 2024 (pp. 10, 49, 50).

[290] Zirui Neil Zhao, Adam Morrison, Christopher W Fletcher, and Josep Torrellas. Last-Level Cache Side-Channel Attacks Are Feasible in the Modern Public Cloud. In: ASPLOS. 2024 (pp. 5, 10, 47, 50).

Part II.

Publications

Note that this is only the first part of the thesis. For Part II, please download the full thesis.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used anything other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.