Microarchitectural Attacks in Heterogeneous Systems: A Survey

HODA NAGHIBIJOUYBARI, Binghamton University, USA
ESMAEIL MOHAMMADIAN KORUYEH, University of California Riverside, USA
NAEL ABU-GHAZALEH, University of California Riverside, USA

With the increasing proliferation of hardware accelerators and the predicted continued increase in the heterogeneity of future computing systems, it is necessary to understand the security properties of such systems. In this survey article, we consider the security of heterogeneous systems against microarchitectural attacks, with a focus on covert- and side-channel attacks, as well as fault injection attacks. We review works that have explored the vulnerability of the individual accelerators (such as Graphical Processing Units, GPUs and Field Programmable Gate Arrays, FPGAs) against these attacks, as well as efforts to mitigate them. We also consider the vulnerability of other components within a heterogeneous system such as the interconnect and memory component. We believe that this survey is especially timely, as new accelerators and heterogeneous systems are being designed such that these designs understand the security threats and develop systems that are not only performant but also secure.

CCS Concepts: • **Computer systems organization → Heterogeneous (hybrid) systems**; • **Security and privacy → Hardware attacks and countermeasures**.

Additional Key Words and Phrases: heterogeneous systems, hardware security, microarchitectural attacks

1 INTRODUCTION

Modern Computing systems are increasingly heterogeneous–this is true at every scale, from high-end servers and high-performance computing clusters all the way down to low-power end-user devices including mobile phones and tablets. Typically heterogeneous systems combine one or more traditional processors (CPUs) to run the operating system and traditional workloads along with general purpose or specialized accelerators to perform application-specific computations – these specialized accelerators can achieve higher performance and significant power advantages. These systems may also include complex memory hierarchies, some of which may use new technologies or incorporate some processing in memory. Interconnecting these systems are a variety of interconnect and memory architectures and provide both performance and energy efficiency for a wide range of computational tasks. While these systems offer superior performance and power, they also potentially expose new vulnerabilities and security problems due to their complex architectures.

On more traditional systems, microarchitectural attacks are proving to be an increasingly serious threat. For example, covert and side channel attacks threaten system-provided isolation when trusted software is co-located with untrusted or compromised software on the same hardware resources. Such attacks exploit shared microarchitectural resources to exfiltrate sensitive information. Until recently, most of the covert and side channel attacks have been demonstrated on CPUs, exploiting different optimization structures such as

Authors' addresses: Hoda Naghibijouybari, Binghamton University, Binghamton, NY, USA, hnaghibi@binghamton.edu; Esmaeil Mohammadian Koruyeh, University of California Riverside, Riverside, USA, emoha004@ucr.edu; Nael Abu-Ghazaleh, University of California Riverside, Riverside, USA, naelag@ucr.edu.

0360-0300/2022/6-ART $15.00
https://doi.org/10.1145/3544102

caches [80, 90, 100, 123, 155, 212, 226, 230], and branch predictors [59, 60]. Similarly, fault injection attacks, where an attacker is able to cause an exploitable fault in a victim program have also become more common in recent years. For example, many variants of practical Rowhammer attacks have been demonstrated on different generations of DRAM chips, where an unprivileged software can cause faults in system memory owned by a victim process [63, 78, 79, 108, 203].

In this survey, we review and classify research exploring microarchitectural attacks in the context of heterogeneous systems. Both the architecture and the accessibility of the components (such as GPUs and other accelerators) within these systems differ significantly from traditional CPUs which have been the focus of most microarchitectural attacks. As a result, both attacks and potential defenses can differ significantly from their counterparts on CPUs. Moreover, beyond attacks within a single component, some attacks can involve multiple components, for example, an attack from the CPU to an accelerator or vice versa. These types of attacks are new to these environments. Both types of attacks (within a single component or across components) are dangerous because they bypass mitigations that focus primarily on the CPU.

We believe this survey provides important insights into how these threat models manifest in such systems. They extend our understanding of the threat to guide further research into defenses to secure emerging systems. In future heterogeneous systems, secure hardware and software design requires protection to extend from a single component (mainly CPU) to the entire heterogeneous system. We review the general architecture of heterogeneous systems in Section 2.

In this survey, we focus on software accessible attacks that target the underlying microarchitecture in the context of emerging heterogeneous systems. Specifically, we consider side channel and covert channel attacks, as well as fault injection attacks (primarily Rowhammer and software-based fault injection attacks). Section 3 introduces these attacks. These attacks are dangerous because they do not require physical access to the victim device. We consider the different modes where the malicious software can co-locate on a victim machine: (1) As a co-located malicious application (e.g., a downloaded app with normal user permissions); (2) Through co-location on the cloud: cloud providers gain efficiency by sharing resources among different users; and (3) Through web interfaces: increasingly, web standards are evolving to enable web applications to use available accelerators. We also consider different threat models including attacks from a co-located spy within the same computational component, e.g., GPU or FPGA, as well as cross-component attacks.

Section 4 discusses the opportunities and challenges that heterogeneous systems introduce in developing microarchitectural attacks and also presents an attack classification and the overview of this survey. In Sections 5, 6, and 7, we review the existing attacks and some defenses that arose in heterogeneous systems in three main categories: (1) microarchitectural attacks on accelerators, and extended trusted execution to accelerators, (2) microarchitectural attacks on interconnects, and (3) memory attacks in heterogeneous systems. Section 8 overviews the attacks that originate from the web in heterogeneous systems. Finally, Section 9 presents concluding remarks and outlines potential future research directions.

2 WHAT ARE HETEROGENEOUS SYSTEMS?

CPU architecture has been evolving over the years to take advantage of the increase of transistor count as the feature sizes of transistors have continued to shrink. This has led first to improvements in processor designs with increasing cache sizes and more aggressive out of order execution and eventually to increases in core count with the introduction of chip multi-processors.

With the end of Dennard's scaling [54], processors are increasingly limited by the power wall [84] leading to the Dark Silicon era [57] where it is not possible to continue to power all the transistors on a chip at the same time. These trends increasingly limit the advantages in performance that are possible from traditional architecture optimizations as well as the use of multi-cores since it is not possible to power these additional resources.

One of the promising approaches to continue to gain performance in this environment is hardware acceleration/specialization: by creating dedicated accelerators for important applications we can bypass the performance and energy overheads of general purpose processing, improving both performance and energy efficiency [83]. For example, modern mobile System-on-Chips (SoCs) include multiple accelerators for common application classes including GPUs, Neural Processing Units (NPUs), video and audio accelerators as well as others. Moreover, due to overheads in moving increasingly large data around, processing support is also being investigated in memory and storage, making additional processing available within the system [33, 140].

As a result of these trends, computing systems are already heterogeneous and are forecast to continue to be increasingly so [45]. Primarily, these heterogeneous elements are integrated into the system as co-processors tasked by the CPU, but it is likely that more flexible models may evolve [180]. The heterogeneous CPU/accelerator computing approach improves resource utilization, while CPU offloads compute-intensive tasks to accelerators, it remains idle to perform latency-critical applications, at the same time. The federation of latency oriented CPUs and throughput oriented accelerators provides both performance and energy efficiency for a wide variety of computational tasks.

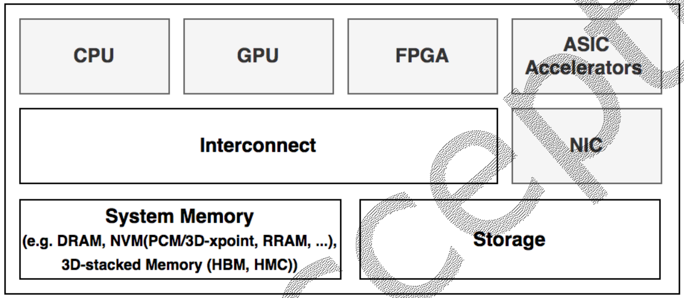

CPU
GPU
FPGA
ASIC Accelerators
Interconnect
NIC
System Memory
(e.g. DRAM, NVM(PCM/3D-xpoint, RRAM, ...), 3D-stacked Memory (HBM, HMC))
Storage

Fig. 1. Overview of common heterogeneous system components

The most common accelerators in current widely-used heterogeneous systems include:

- **Graphics Processing Unit (GPU):** GPUs are integral components to most modern computing devices including embedded systems, mobile phones, personal computers, and workstations. They were primarily used to optimize the performance of graphics and multi-media heavy workloads. Their highly parallel structure makes them more efficient than CPUs to process large blocks of data in parallel. Therefore, General Purpose GPUs (GPGPUs) are also increasingly integrated on computing servers, computational clouds, and clusters to accelerate a broad range of applications from domains including security, computer vision, computational finance, bio-informatics, and many others [134]. GPUs can be either as a stand-alone device (discrete GPUs), which is connected with the CPU using a PCIe bus, or integrated on the same die as the main CPU (integrated GPUs).
- **Field-Programmable Gate Array (FPGA):** FPGAs implement re-configurable hardware to speed up application-specific tasks, particularly for cloud computing applications. FPGAs have been increasingly used in computational clouds to accelerate highly sensitive applications including cryptography, financial modeling, and genomic sequencing. Recently, FPGAs have been also tightly integrated on the same chip as the CPUs to offer cache-coherent memory systems for even better performance [88].
- **Domain-Specific Accelerators:** New domain-specific accelerators (or ASIC accelerators) are being designed and released each year to accelerate a variety of applications, mostly in the domain of machine learning and artificial intelligence such as speech recognition and video object detection. These specialized accelerators optimize memory use and the use of lower precision arithmetic to accelerate calculation and

increase the throughput of computation, as a result, provide major improvements in energy and performance. They include Tensor Processing Units (TPU) [74] deployed in data centers and clouds, and Neural Processing Units (NPU) [61] tightly coupled with the main processors. Specialized hardware accelerators have been recently designed to exploit data specialization and parallelism for many other application domains as well, including cryptography[144, 221] and audio/video/image processing[159, 172].

These accelerators as co-processors are connected to the main processor (CPU) and also memory subsystems through high bandwidth interconnect to build a heterogeneous system. In addition to provide heterogeneity in processors, future systems are likely to be deeply heterogeneous in memory subsystem. New memory technologies are proliferating, including Non-Volatile Memories (e.g. Phase Change Memory (PCM) [218], 3D-xpoint [2], and Resistive RAM (ReRAM) [217]) and 3D-stacked Memories with different architectures (High-Bandwidth Memory (HBM) [107] and Hybrid Memory Cube (HMC) [154])) that enable a new design paradigm as Processing-in-Memory (PIM) [33].

These new memory technologies/architectures are different in latency, bandwidth, power consumption, cost, and other new design parameters such as persistence. To build large, fast, and reliable memory systems at low power, future systems will combine different memory technologies [125]. Figure 1 demonstrates the overview of main components in heterogeneous system architecture.

3 ATTACK TYPES

Our focus in this survey is on microarchitectural attacks, primarily covert and side channel attacks, as well as fault injection attacks. Microarchitectural covert and side channel attacks exploit unintended leakage that occurs when different applications compete for shared microarchitecture structures such as caches, ports, or interconnection buses. Over the last several years, covert and side channel attacks have been shown to pose a substantial threat to modern computing systems. In addition, they have been leveraged as a critical part of transient execution attacks such as Spectre [110] and Meltdown [119] and their many variants [42, 112, 131, 173, 202, 204, 214] The other major attack vector we consider is fault injection attacks: an attacker causes an error (typically a bit flip) in the victim program state typically by driving the system outside its physical operating envelope. Originally, such attacks were conducted physically by heating a device or bombarding it with radiation. However, recently, software controlled attacks have been discovered that can cause exploitable faults. Perhaps the most widely studied is the Rowhammer attack [63, 78, 79, 108, 203], where repeatedly accessing DRAM memory rows induces faults in neighboring rows. More recently, software-based voltage control has also been exploited to induce faults [102, 139, 160, 187], indicating that fault injection is a more general threat model.

We focus on microarchitectural attacks that are exploitable by software since these can be exploited by a remote attacker (provided they are able to run on a victim machine) without physical access. Further, we do not consider supply chain attacks such as hardware trojans [190] as part of our threat model.

In the remainder of this section, we overview microarchitectural attacks in more detail.

3.1 Microarchitectural covert and side channels

When a microarchitectural structure is shared between two processes, they observe unintentional side effects due to contention on this resource. In the context of traditional systems, attacks have been developed on a variety of microarchitectural structures, including different levels of cache [80, 90, 100, 123, 155, 212, 226, 230], branch predictors [59, 60], random number generators [58, 163], and others [47, 105].

Covert- and Side-channel attacks: In a typical covert channel scenario, two malicious processes (Spy and Trojan) run concurrently on the system and co-operate in building an unintended communication channel to transfer secret data. In a side channel scenario, there is no co-operation. One malicious application (Spy) gets access to the system and extracts secret or sensitive information from a Victim application that is running on the same system. In both channels, there is no direct channel between two applications based on the security policy of the system.

In the context of microarchitectural channels, both covert channel and side channel can be built by creating/monitoring contention on shared hardware resources. For example, in a covert channel, the Trojan (sender) creates contention on a shared resource (e.g. cache) to send "1" and does nothing for a while to send a "0", and the Spy (receiver) measures the access time to the shared resource to decode the transferred bit. In a side channel, a Spy application measures the access time to a shared resource to observe contention from Victim application's activities, which are potentially correlated to its secret data due to some data dependent software or hardware implementation details. By analyzing the collected data, the Spy can extract the Victim's secret data. Alternatively, a Spy application can get access to the processor and measure the execution time of the Victim process to build a timing side channel. Figure 2 shows an example of a timing side channel to extract the secret encryption key through the execution time, while the CPU is encrypting the data using Victim's secret key [100, 123].

Although timing is a commonly used source of leakage, covert and side channels can be developed using other leakage vectors as well, including power, electromagnetic, acoustic, and thermal emanation [64, 65, 129]. Typically, these attacks require physical access to the system for the measurement, although software accessible measurements of power are offered by some systems [118].

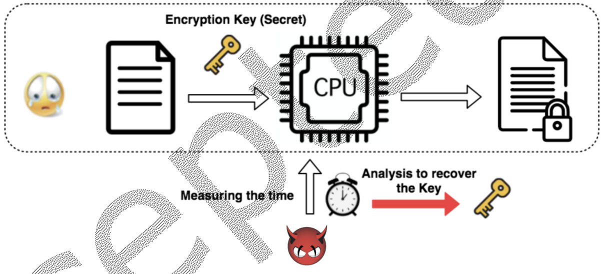


Fig. 2. Timing Side Channel on Encryption Application

In traditional microarchitectural channels, a Spy process can remotely get co-located with the Victim/Trojan by launching and executing its process on the same processor, simultaneously with the Victim/Trojan, which is a common threat model in virtualized systems and multi-tenant clouds. We do not consider physical covert and side channel attacks to be part of our threat model because they require physical measurements and because do not target the direct behavior of microarchitectural structures.

Transient Execution attacks: Recently, a novel class of microarchitectural attacks have emerged on CPUs, called *transient-execution attacks* [41, 222] such as well-known Meltdown [119, 202, 214], Spectre [110, 112, 131], and microarchitectural data sampling (MDS) attacks [42, 163, 173, 204] that rely on side channel attacks. In these attacks, Out-of-Order and transient (speculative) execution in modern CPUs are exploited to execute instructions that are then reverted as if they were never executed. However, these transient instructions and their effects can be observed on the microarchitectural level and can be leaked by side channel attacks. These attacks are out of the scope of this survey since we review the attacks that arise beyond the CPUs and current accelerators do not support (basically do not require, due to high parallelism) out-of-order and speculative execution that are the main enablers of transient-execution attacks. However, it is possible that speculation may be used as part of future architectures, in which case this threat model may become relevant again.

3.2 Fault Injection Attacks

In fault injection attacks, the attacker alters the correct functionality of a system by injecting malicious faults. Typically, the attacker applies physical stress (such as clock glitches, supply voltage glitches, electromagnetic (EM) pulses, and laser shots) on the microprocessor to induce hardware faults [233]. However, recent works exploited software interfaces to inject fault leveraging dynamic voltage and frequency scaling (DVFS), or through access behavior utilizing Rowhammer on DRAM.

DVFS in modern processors is an energy management technique that saves energy by regulating the frequency and voltage of the processor cores according to runtime computing demands. In a typical DVFS scheme, kernel level drivers control the frequency and voltage of a processor through on-chip regulators. An adversary can exploit the interface between the software drivers and hardware regulators to induce faults in a multi-core processor. Recent works exploited this software-based voltage control to inject faults into a trusted execution environment such as Intel SGX and ARM Trustzone [102, 139, 160, 187].

Rowhammer based attacks are the most well-known fault injection attacks in modern computing systems [63, 78, 79, 108, 203]. As it is demonstrated in Figure 3, Rowhammer exploits a vulnerability in system DRAM, in which repeatedly accessing a row of memory (hammering) can cause bit flips in adjacent rows, leading to extremely dangerous exploitations like privilege escalation of a normal user to a system administrator.

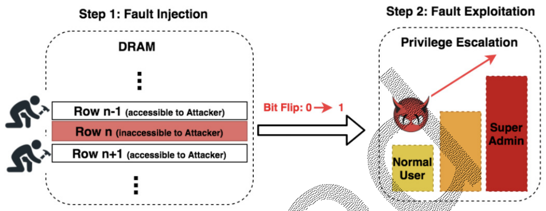

Step 1: Fault Injection
DRAM
Row n-1 (accessible to Attacker)
Row n (inaccessible to Attacker)
Row n+1 (accessible to Attacker)
Bit Flip: 0 → 1
Step 2: Fault Exploitation
Privilege Escalation
Normal User
Super Admin

Fig. 3. Fault Injection Attack: Rowhammer

Most microarchitectural covert and side channel attacks and Rowhammer attacks (and other software-based fault injection attacks) have focused and developed on the CPU and its structures. However, modern computing systems are increasingly made up of a federation of the CPU with other specialized accelerators and processors to gain performance, and it is essential for hardware security researchers to understand how microarchitectural attacks manifest within such complex environments and how to secure the systems against potential attacks.

4 ATTACKS IN HETEROGENEOUS SYSTEMS AND TAXONOMY

The push for improving the performance and efficiency of processors and memory subsystems and the ongoing transition from homogeneous to heterogeneous systems inevitably creates new threat models and security vulnerabilities. These systems are complex federations of different heterogeneous elements and therefore expose new interfaces and operating points from which malicious software can operate. Moreover, different components offer different computational models that can potentially impact attack opportunities. It is important to understand these new attack opportunities to inform how systems should be designed to mitigate them. In this section, we summarize the opportunities offered by heterogeneous system architectures for the attackers, and the main challenges to develop microarchitectural attacks on such systems.

In comparison with homogeneous multi-core CPUs, heterogeneous systems provide the attackers with unique opportunities from different aspects, including:

(1) New accelerators with new computational modes (GPU, FPGA, NPU, and other accelerators) lead to new attacks; Accelerators in heterogeneous systems can serve as the target or origin of attacks, expanding the attack surface and providing more co-location opportunities to build microarchitectural attacks. Massive parallelism in accelerators facilitates development of high bandwidth and high quality attacks. Moreover, new programming models or APIs offered by accelerators (e.g. GPU-accelerated Web APIs) can expose additional attack opportunities. Of course, these attacks also introduce new challenges in terms of how to reverse engineer these new components and how to control scheduling and co-location to create attack opportunities.

(2) New attacks across components (accelerator to CPU, or to other components); This new class of microarchitectural attacks relieves the attacker from co-location even on the same processor and can bypass existing defense mechanisms that mainly focus on one component of the system. However, synchronization across these components with frequency disparity and completely different architecture and computational models will be a critical challenge in developing a robust and error-free attack. In addition, to develop microarchitectural attacks

across the components (e.g. CPU and accelerator on the shared resource) with asymmetric view on the shared resource, an attacker requires to reverse engineer access hierarchy from both asymmetric sides, making it more challenging.

(3) Complex interconnect and memory can create new contention and co-allocation opportunities; The access from multiple components (with substantially different memory request pattern and intensity) to the bandwidth and throughput limited interconnects can lead to interference and facilitate contention-based and cross-component microarchitectural attacks. Efficient data sharing and transferring across the components (CPUs and accelerators) to support heterogeneous computing also necessitates optimizations in the protocol layer of interconnects (e.g. support of hardware coherency), that lead to new attacks. In addition, emerging memory technologies and architectures offer new sharing opportunities (e.g. logic units in PIM) for attackers.

(4) New defense mechanisms are needed; Existing defense mechanisms are primarily designed to isolate the contexts/processes that are running within a processor. To protect against accelerator-based attacks and cross-component attacks, microarchitectural defense mechanisms need to be extended to isolate processes that run within different processors (CPUs or accelerators) or span several components.

While heterogeneous systems are being widely deployed in computing platform at all scales, it is critical to expand our understanding of the threat model within these systems that can lead to new defenses to mitigate such attacks.

4.1 Overview of the survey

Previous works surveyed both microarchitectural attacks and defenses with the main focus on CPUs [41, 66, 185, 222]. In this survey, we review microarchitectural attacks (specifically covert and side channels) and fault injection attacks (primarily Rowhammer and software-based fault injection attacks) that arise beyond CPU in heterogeneous systems. As shown in Figure 4, we categorize these attacks into three main classes based on the component they are originated from or are developed on: (1) accelerators (Section 5): these are attacks that target accelerators and their internal structures; (2) interconnects (Section 6): these include attacks on the interconnect and the communication protocols among components in heterogeneous system; and (3) memory subsystem (Section 7): this section reviews attacks on the memory subsystem in heterogeneous systems. Although we categorize and discuss the attacks based on the component and the attack types, in all of these three main categories, we overview the existing attacks and discuss possible future direction in different threat models, including attacks from a co-located attacker in multi-tenant clouds or user devices, as well as cross-component attacks that span several components in heterogeneous systems. Threat models are summarized in tables in Sections 5, 6, and 7. We also consider three different modes on how the malicious software may have access to the victim's machine to launch the attack; (1) As a co-located malicious application (e.g. a downloaded app); (2) Through co-location on the cloud; and (3) Through web interfaces. We discuss the first two access modes in Sections 5, 6, and 7 and discuss the new attacks that originate from the web in heterogeneous systems in Section 8.

5 SECURITY OF ACCELERATORS

Accelerators in heterogeneous systems could be either a target of an attack themselves or alternatively serve as a vector for launching or amplifying attacks targeting the CPU or other components.

An important pre-requisite to all microarchitectural attacks is *colocation*: how can the attacker position itself (i.e., its running code) such that it is in a position to interact with the victim, for example, to cause a fault, or to cause contention and measure side-channel leakage. While this is a problem also in CPUs and on the cloud [168] the problem is different in heterogeneous systems due to the heterogeneity of available resources and the different placement algorithms. For accelerators, the presence of parallelism further complicates the problem:

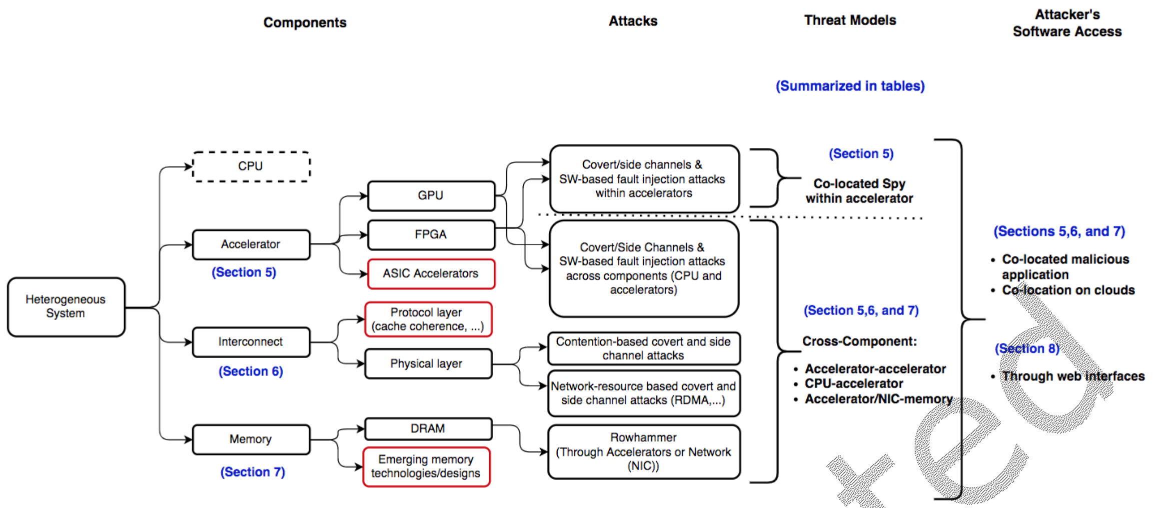

Components
Attacks
Threat Models
Attacker's Software Access
(Summarized in tables)
Heterogeneous System
CPU
Accelerator
(Section 5)
GPU
FPGA
ASIC Accelerators
Interconnect
(Section 6)
Protocol layer (cache coherence, ...)
Physical layer
Memory
(Section 7)
DRAM
Emerging memory technologies/designs
Covert/side channels & SW-based fault injection attacks within accelerators
Covert/Side Channels & SW-based fault injection attacks across components (CPU and accelerators)
Contention-based covert and side channel attacks
Network-resource based covert and side channel attacks (RDMA,...)
Rowhammer (Through Accelerators or Network (NIC))
(Section 5)
Co-located Spy within accelerator
(Section 5,6, and 7)
Cross-Component:
• Accelerator-accelerator
• CPU-accelerator
• Accelerator/NIC-memory
(Sections 5,6, and 7)
• Co-located malicious application
• Co-location on clouds
(Section 8)
• Through web interfaces

Fig. 4. Heterogeneous systems components and attack classification. (Red boxes are some open research directions)

for example, on a GPU, some attacks may require placement on a specific core (or SM in Nvidia terminology) to cause contention on the resources of that core. This may require the attacker to reverse engineer the placement schedulers to control the placement of applications on the processor/accelerator (e.g., spy and trojan in covert channel, or spy and victim in side channel) to establish co-location. Then, based on their co-location options, covert and side channel can be constructed on the available shared microarchitectural resources. The attacker may also exploit these schedulers to try to amplify their attack, for example by blocking other applications to reduce the noise.

With the proliferation of accelerators in multi-tenant clouds and HPC clusters [36, 75, 136, 177], multiprogramming support is also being increasingly improved in accelerators [3, 4, 25], making resource sharing and as a result microarchitectural attacks possible. Furthermore, end-user devices that integrate accelerators (e.g. GPUs, FPGAs, AI accelerators) tightly on the same chip with CPU, provide more co-location opportunities in the system. In such systems, some microarchitectural resources (e.g. interconnects, last level cache, and memory subsystem) are shared across processors. Thus, even if two applications are not able to co-locate on the same processor, they can still construct cross-component microarchitectural covert and side channel attacks.

In this section, we survey the security of widely-used accelerators in modern heterogeneous systems. We first review the existing microarchitectural attacks within GPUs and FPGAs. We also review attacks that span different components in integrated CPU-GPU and CPU-FPGA systems. We also discuss the likely future directions on the security of domain-specific accelerators including emerging ML accelerators. Finally, we survey some of the important works in extending the trusted execution to these accelerators in heterogeneous systems.

5.1 Security of GPUs

GPUs, as the most common accelerators are important components of heterogeneous systems due to their superior performance and power properties for both general-purpose computational and multimedia workloads. GPUs are part of computing platforms at all scales: at the mobile and embedded system scale, GPUs are often used because of their high performance on multi-media and machine learning workloads. In laptops and personal computers, GPUs provide support for graphics and multimedia workloads. On server-class machines, GPUs are used to support data-intensive workloads that fit their programming models.

GPUs come in two forms (1) discrete GPUs: a separate device, which is connected with the rest of the systems typically using a PCIe bus (or NVLink in new generations of Nvidia GPUs [145]), which addresses a separate physical memory; (2) Integrated GPUs (iGPUs): which are built on the same die as the main CPU and physically share the same system memory. Figure 5 demonstrates how discrete and integrated GPUs are connected to the CPU.

GPUs were originally designed to accelerate graphics and multimedia workloads. They are usually programmed using application programming interfaces such as OpenGL for 2D/3D graphics [29], or WebGL [31] and WebGPU [7] which is usable within browsers. On embedded systems like smartphones and tablet computers, a modified version of OpenGL is available called OpenGL-ES [28]. In the past few years, GPU manufacturers have also enabled general purpose programmability for GPUs, allowing them to be used to accelerate data-intensive applications using programming interfaces such as CUDA [24], OpenCL [27], and Vulkan [30].

Despite their improving performance and increasing range of applications, the security vulnerabilities of GPUs and other accelerators have not been studied until recent years. Olson et al. [148] developed a taxonomy of vulnerabilities and security threats for accelerators based on threat types into attacks that affect the security triad of Confidentiality, Integrity, or Availability. They also classify the risk categories in terms of what part of the accelerator attacks affect. Although the paper offers no concrete attacks, it highlights that security of accelerators warrants significant attention.

Perhaps because they are the most widely deployed accelerators, their security has been studied extensively both from an offensive as well as defensive perspective, in recent years. In this section, we review and classify this research. In addition to its importance in understanding GPU security, we believe that some of these vulnerabilities and defense principles will also apply to other accelerators and inform how to design them to be secure.

In recent years, researchers explored the vulnerability of GPUs at both hardware/architecture and software levels. They exploited these vulnerabilities to demonstrate practical attacks and also proposed mitigation against some of them. Mittal et al. [183] present a survey of techniques for analyzing and improving GPU security. They classify GPU security vulnerabilities and also discuss potential countermeasures. However, this classification does not cover significant new work that has appeared since 2018. Moreover, the focus of the survey primarily was on software attacks such as leftover state due to lack of data erasure after context switching, buffer overflows, denial of service, malware, and the few covert and side channels that were published at the time. Since then, many new microarchitectural attacks have appeared on GPUs.

In this subsection, we review existing attacks and defenses on GPUs in terms of microarchitectural covert and side channel attacks, software-based fault injection attacks, and also some related attacks on GPU software stack. We also discuss the defense mechanisms against microarchitectural attacks and survey the most important works on extending the trusted execution environment to GPUs in Section 5.4. Table 1 summarizes the attack, threat model, leakage vectors, microarchitectural structure, and the attacker's goal on existing GPU microarchitectural attacks.

5.1.1 *Covert and Side Channel Attacks on Discrete GPUs.*

In recent years, several works developed microarchitectural covert and side channel attacks on GPUs. These covert and side channel attacks can be categorized in several ways, based on: (1) the threat model; whether the spy launches the GPU kernel and measures the leakage from the CPU (host) side, or the spy is co-located on the GPU with the victim process (either concurrently or with fine-grained context switching) and measures the contention on the shared resources through different leakage vectors; (2) leakage vector (e.g. timing, performance counters, EM, and power traces); (3) microarchitectural structure that causes the leakage; and (4) the attacker's goal (e.g. recovering encryption keys, DNN model extraction, workload detection).

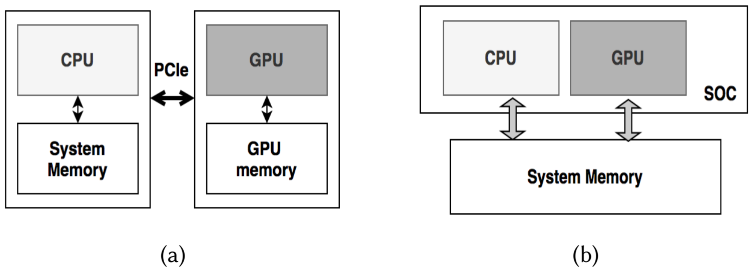


Fig. 5. GPU in system (a) Discrete GPU; (b) Integrated GPU

In this section, we review the existing works in two categories based on their threat model: First, we study the attacks in which leakage is measured from the CPU side and then attacks that are developed by a co-located spy on the GPU.

Attacks from the CPU: In these attacks, the attacker typically launches a GPU workload that holds some secrets and attempts to determine from its data-dependent behavior (typically measured through timing the GPU workload) information regarding the secret. We overview these attacks next.

Jiang et al. [94] conduct the first timing attack at the architecture level on GPUs. The attack exploits the difference in timing between addresses generated by different threads as they access memory: if the addresses are *coalesced* such that they refer to the same memory block, they are much faster than uncoalesced accesses which require several expensive memory operations. Execution time relies on the number of unique memory requests (unique cache lines) after coalescing. Thus, in a cryptographic application, the key affects the address pattern accessed by the threads, and therefore the observed run time of the encryption algorithm, opening the door for a timing attack where the encryption time is used to infer the likely key. Ahn et al. [34] demonstrate that [94] is not applicable to modern GPUs with sectored-cache microarchitecture in which cache lines are being accessed at the "sector" (i.e., 32B) granularity and brought into the L1 cache. When data was accessed in cache line (128B) granularity and combined with a relatively small T-table size in AES, all of the table entries can be easily loaded into the L1 cache. However, that is no longer the case with a 32B access granularity. As a result, memory accesses will hit in either the L1 or L2 cache and the positive correlation between the number of unique cache lines and overall execution time no longer exists. Ahn et al. [34] propose Trident, a hybrid GPU timing channel to recover all AES keys. In their proposed attack, they leverage negative timing correlation (as data that are brought into the L1 cache help to reduce the execution time for latter key bytes) to recover earlier key bytes of AES, while exploiting cache-collision attacks for the latter AES key bytes.

Jiang et al. [95] present another timing attack on table-based AES encryption on GPUs. They find a correlation between the execution time of one table lookup of a warp and a number of shared memory bank conflicts generated by threads within the warp. They use these key-dependent differences in timing to correlate measured execution time to the key at the last round of the AES encryption as it executes on the GPU. Luo et al. [128] present a timing side channel attack on GPU accelerated RSA encryption. They use two existing optimizations: Sliding Window Exponentiation and Montgomery Multiplication and exploit the correlation between total execution time of a message decryption and the number of reductions in each window of decryption to extract the RSA private key.

Wang and Zhang [210] propose a profiling-based side-channel attack to fully recover the AES encryption secret key. Rather than execution time, they profile two performance matrices, the number of the memory load and memory store requests. Based on these, the number of unique memory load requests in the last round encryption for each byte except the first byte then can be determined. They recover 16-bytes AES key byte by byte. They

Table 1. Summary of microarchitectural attacks on GPUs (dGPU: discrete GPU, iGPU: integrated GPU)

	Attack	**Threat model**	**Leakage**	**Microarch structure**	**Attacker's goal (for side channel & fault injection)**
Jiang et al. [94]	side channel	from CPU to dGPU	timing	caches & memory coalescing unit	AES key recovery
Ahn et al. [34]	side channel	from CPU to dGPU	timing	caches & memory coalescing unit	AES key recovery
Jiang et al. [95]	side channel	from CPU to dGPU	timing	shared memory	AES key recovery
Luo et al. [128]	side channel	from CPU to dGPU	timing	GPU's parallel execution & blocking (e.g. warp divergence)	RSA key recovery
Wang and Zhang [210]	side channel	from CPU to dGPU	perf counters	memory (loads & stores)	AES key recovery
Hu et al. [87]	side channel	from CPU & co-located spy within dGPU	EM & bus snooping (memory reads /writes)	GPU's memory bus & PCIe	DNN model extraction
Naghibijouybari et al. [141]	covert channel	co-located spy	timing	caches, computation units & memory	-
Nayak et al. [143]	covert channel	co-located spy within dGPU	timing	TLB	-
Ahn et al. [35]	covert channel	co-located spy within dGPU	timing	GPU's internal NoC	-
Naghibijouybari et al. [142]	side channel	co-located spy within dGPU	perf counters, timing, and memory utilization	memory, caches & instruction issue	Website fingerprinting, keystroke timig and DNN model extraction
Wei et al. [96]	side channel	co-located spy within dGPU	perf counters	caches & memory	DNN model extraction
Liu et al. [124]	side channel	co-located spy within dGPU	timing	memory & computation units	workload detection
Dutta et al. [56]	covert channel	CPU-iGPU	timing	LLC & ring bus	-
Frigo et al. [62]	Rowhammer	from iGPU	-	DRAM	scape Firefox sandbox, break ASLR and remote code execution
Sabbagh et al. [169]	SW-based fault injection	from CPU to dGPU	VFS software interface (using GPU driver)	-	Recover AES keys

calculate the number of unique memory load requests with 256 possibilities of a single byte of the AES key and compare it to the profiled number. By repeating the same procedure with different input data, they can successfully extract the exact byte key.

Hu et a. [87] propose DeepSniffer, a learning-based DNN model extraction framework. They obtain the complete model architecture information while running on GPU without any prior knowledge of the victim model. Through electromagnetic (EM) side channel and bus snooping attacks on discrete GPU's memory bus and also Peripheral Component Interconnect express (PCIe) bus that connects discrete GPU to the CPU, they collect architectural hints (e.g., volumes of memory reads/writes and memory access traces) from the CPU side or a co-located spy on GPU. They use the correlation between extracted architectural events and model internal architectures to recover the DNN models.

Attacks within the GPU: The attacks discussed so far are measured from the CPU side, requiring the spy to be able to launch (or anticipate the launch) of a GPU kernel to measure the execution time, performance counters, or other leakage vectors.

With the widespread deployment of GPUs in end-user devices, major cloud platforms [36, 75, 136], HPC clusters, and all computing domains including security sensitive ones, and at the same time increasing multiprogramming support in modern GPUs [3, 4, 25] to improve resource utilization, general covert and side channel attacks between co-located applications become substantial threat models. As an example, in a multi-tenant GPU-based cloud, a malicious VM can now spy on other applications that share a GPU (a side channel attack) or collude with another to covertly communicate sensitive information to bypass information isolation boundaries (a covert channel attack).

This is a new threat model and offers substantially higher access for an attacker to colocate and spy on victims. Because the attacker and victim are separate programs, it is also possible for covert channel attacks to be conducted where a spy and a trojan are attempting to covertly communicate through microarchitectural leakage. This threat model does not apply in the CPU to GPU scenario since both the CPU and the GPU code are part of the same process. We review the most relevant efforts in this space next.

The first work in this area was by Naghibijouybari et al. [141, 142] who investigate general covert and side channels between two concurrent applications on the GPU. They first reverse engineer the co-location of concurrent applications on General Purpose GPUs (GPGPU), characterize the contention on different shared resources, and construct covert channels on a variety of resources on GPGPUs [141], including L1 and L2 caches, different types of functional units, and memory. They demonstrate that GPU hardware schedulers can provide unique opportunities to force exclusive co-location of spy and trojan to prevent interference of other workloads, achieving noise-free communication. Their experiments show that very high quality and bandwidth covert channels (over 4Mbps error-free bandwidth in Nvidia Kepler GPUs) are feasible on GPGPUs. Nayak et al. [143] develop a similar microarchitectural covert channel on another resource, GPU's shared last level translation lookaside buffer(TLB). Ahn et al. [35] observe that GPU's on-chip network (NoC) connects Streaming Multiprocessors (SMs) at one level and also clusters of SMs at another level. They exploit the contention on these shared on-chip interconnects to build microacrhitectural covert channels within discrete GPUs.

Naghibijouybari et al. [142] also demonstrate side channel attacks that can be launched within both the computational and graphics software stacks, as well as across them. They implement a family of end-to-end attacks where the spy can interleave execution with the victim to extract side channel information. The first attack implements website fingerprinting through GPU memory utilization API or GPU performance counters. The spy probes the memory API to obtain a trace of the memory allocation operations carried out by the victim as it renders different objects on a webpage visited by the user. The second attack tracks user activities as they interact with a website or type characters on a keyboard. They accurately track re-rendering events on GPU and measure the timing of keystrokes as they type characters in a textbox (e.g., a password box on Facebook website), making it possible to carry out keystroke timing analysis to infer the characters being typed by the user. A third attack uses a CUDA spy on the computational stack of GPU to infer the internal structure of a neural network application, which is often a trade secret (a model extraction attack). When a victim is running concurrently on the GPU and utilizing the shared resources, depending on number of input layer size or other parameters of neural network model, the intensity and pattern of contention on the cache, memory and functional units is different over time, which they demonstrated creates measurable leakage in the spy enabling the model to be extracted.

Wei et al. [96] conduct a similar attack to infer the hyper-parameters of a Deep Neural Network model. Rather than the concurrent running of applications on GPU (as in [142]), they consider fine-grained time-sliced sharing of applications and exploit context-switching penalties on performance counters as leakage vector. Although in direct response to side channel attacks presented in [142], Nvidia released a patch [6] in their new GPU drivers to disabling the normal users to access to the performance counter, Wei et al. bypass this patch by downgrading the GPU driver on the spy VM which is invisible to the victim. Zou et al. [238] also utilize performance counters on GPUs as features fed to machine learning based classification models to classify running workloads on GPU accelerated HPC systems. This work is not a covert or side channel and does not include any threat model (performance counters are measured by the same application for workload characterization).

Liu et al. [124] investigate a side-channel attack in virtualized GPU environments. Specifically, the attack originates from one virtual machine to another where both share the same physical GPU, and each virtual machine has its own GPU system stack. In such settings, GPU hardware performance counters are (typically) not available to VMs. They conduct a side channel on Intel integrated GPUs, launch an OpenCL-based probing application that does some read-compute-write pattern of operations to create contention on different resources on GPU, and measure the execution time as coarse-grained information leakage. Then they utilize machine learning approaches to identify the victim's GPU workload among a small set of several entertainment and deep learning workloads.

5.1.2 Side Channel Defenses on Discrete GPUs. Some recent works proposed mitigations to secure GPU-based systems against existing covert and side channel attacks. We review the existing works in two categories: (1) Defenses against the attacks from the CPU to GPU, and (2) Defenses within the GPU against the attacks from a co-located spy.

Defenses against the attacks from the CPU: To prevent side-channels such as those used in the attack presented in [94], Kadam et al. [97] propose Rcoal, a redesign of GPUs to eliminate predictable memory coalescing behavior. They propose the memory coalescing randomization techniques in several ways including the number of subwarps, the threads assigned to each subwarp, or a combination of both. These randomization techniques generate additional accesses and alleviate such correlation-based timing attacks by making the relationship between execution time and coalesced memory accesses less predictable.

To bypass Rcoal mitigation, Wang and Zhang [209] developed profiling based side channel attacks against RCoal focusing on the configurations with high variance in the number of coalesced accesses. Kadam et al. [98] propose BCoal, a mechanism to further reduces the variance making it a much stronger defense and to efficiently address the limitations of RCoal in terms of performance degradation. BCoal is a bucketing-based coalescing technique that generates the number of coalesced accesses equal to one of the pre-determined values (known as buckets), irrespective of program secrets. As the number of accesses is always equal to the pre-determined values, the variance in the number of accesses drops, reduces the correlation in timing attack. To reduce the performance overhead of additional accesses, they select optimal bucket features by analyzing the application level coalescing profile, such that overall fewer additional accesses are generated. Rcoal and BCoal are generic hardware based coalescing mechanisms applicable to all security-sensitive GPGPU applications that are vulnerable to coalescing-based correlation timing attacks.

To protect the GPU accelerated AES encryption against both timing and cache based side channels, Lin et al. [117] proposed a new software-based mechanism. They rely on Scatter and Gather approach which slices and re-organizes the AES pre-computation tables such that key-dependent table lookups will not leak any timing or address pattern information. This software based approach is specific to AES encryption.

Defenses within the GPU: To mitigate the attacks from a spy co-located on the GPU, Xu et al. [223] propose GPUGuard, a dynamic decision tree based detection and a hierarchical defense framework that can reliably close contention based covert and side channels on GPUs. Once contention on different resources is detected across the concurrent applications, they use GPU-specific characteristics to isolate contending applications into security domains at different hierarchy levels to maximize sharing (and performance) when it is safe, but to close contention based channels when there is a possibility for the existence of attack.

Nvidia designed Multi-Instance GPU (MIG) Technology [146] in their new generations of discrete GPUs (Ampere). In this design, GPU can be securely partitioned into separate GPU instances for multiple users with the isolated paths through the entire memory system; the on-chip crossbar ports, L2 cache banks, memory controllers, and DRAM address busses are all assigned uniquely to an individual instance. Although this feature is designed for performance and optimal GPU utilization, it can potentially mitigate covert and side channel attacks by isolating concurrent applications on discrete GPUs.

GPU-accelerated computing is being increasingly deployed in multi-tenant clouds and virtualized environments for a broad range of applications including DNN and sensitive workloads. In addition, clouds, data centers, and HPC clusters are equipped with multi-GPU systems [18] to offer high performance computing for large-scale AI/DNN workloads. All of these emerging computing platforms provide novel opportunities for attackers to co-locate and as a result share hardware resources with the running applications and extract sensitive data through microatchitectural attacks. To secure these emerging environments, we need to extend the secure design beyond the CPUs to include the specialized and general purpose accelerators and also mitigate cross-processor (e.g. CPU-GPU, Multi-GPU) attack scenarios in heterogeneous computing systems. Achieving this vision will require investigating all threat surfaces in different platforms (with different architectures) and also developing defenses that mitigate such attacks.

5.1.3 Attacks on Integrated CPU-GPU Systems. In integrated heterogeneous systems, accelerators (such as GPUs and FPGAs) are tightly integrated on the same die as CPUs (as shown in Figure 5) and share some hardware resources such as last level cache and memory subsystem. This integration creates the potential of new attacks that exploit common resources to create interference between these components, leading to cross-component microarchitectural attacks.

Dutta et al. [56] develop covert channels (secret communication channels that exploit contention) on integrated CPU-GPU systems in which two malicious applications, located on two different components (CPU and iGPU) transfer secret information via shared hardware resources. These cross-component attacks require solving new problems due to the asymmetric nature of the two communication ends (one on the CPU, and the other on the GPU). These problems include the different views of the memory hierarchy, reconciling different computational models and memory hierarchies, the need for synchronization across heterogeneous components with frequency disparity, and creating reliable fine-grained timing mechanisms. By overcoming these challenges, they constructed two covert channels in cross-component systems. The first covert channel uses the shared LLC cache between CPU and GPU in Intel's integrated GPU architectures and implements a PRIME+PROBE style attack. The second attack is a contention based channel targeting the ring bus connecting the CPU and GPU to the LLC.

Olson et al. [147] propose sandboxing accelerators when the CPU and the accelerators share the same address space (e.g., under a Heterogeneous System Architecture configuration). This type of defense does not protect against side and covert channel vulnerabilities.

5.1.4 Fault Injection Attacks on GPUs. With the advent of software-based fault injection attacks on CPU-based systems in recent years (voltage scaling-based fault injection [139, 187] and Rowhammer [63, 108]), a few recent works investigated and developed such attacks in GPU systems.

Sabbagh et al. [169] present the first non-invasive fault based attack on GPUs called *overdrive attack*. They use software interfaces to the GPU driver to customize the GPU hardware settings, including voltage and frequency scaling (VFS). Leveraging this feature, they inject random faults during GPU kernel execution that can cause silent data corruption (SDC). They exploited SDC-based injected faults into an advanced encryption standard (AES) running kernel to recover all the keys.

Frigo et al. [62] develop web-based GPU accelerated Rowhammer attack on memory in integrated CPU-GPU systems in mobile SOCs. In such integrated systems, the memory subsystem is shared across CPU and GPU, and they demonstrate that GPU can be utilized to amplify the Rowhammer attack. They exploit bit flips from

Rowhammer to escape the Firefox sandbox, break ASLR and obtain arbitrary read and write primitives for remote code execution.

Perhaps not surprisingly, GPUs are also vulnerable to fault injection attacks. In fact, their large degree of parallelism and high memory bandwidth may make them more vulnerable to attacks such as Rowhammer because they are able to generate accesses faster than CPUs. We believe that the threat these attacks pose especially when they can potentially be controlled by software without requiring physical access should be considered as part of securing GPUs and other accelerators. It is possible that general solutions to Rowhammer such as Para [108] and Target Row Refresh (TRR) [116, 135] could also protect GPU memories, but other fault injection vectors such as voltage control should also be considered.

5.2 Security of FPGAs

FPGA-accelerated computations are increasingly being used in low-cost embedded devices to high-performance multi-tenant clouds [10, 15, 177], as they can be reconfigured for the needs of different users at different times. Like other accelerators, FPGAs also provide high performance processing with energy efficiency. Similar to GPUs (Figure 5), FPGAs can be connected to the system using PCIe bus as a discrete or standalone accelerator, or connected tightly on the same die with CPU in an integrated form. In this section, we review some of the existing attacks on FPGAs in heterogeneous systems and discuss possible future microarchitectural attacks on multi-tenant FPGAs in the clouds.

5.2.1 Covert and Side Channel Attacks: FPGAs are being increasingly integrated into major cloud platforms [10, 15, 177]. To date, both Microsoft and Amazon clouds allow only one user access to an FPGA resource at a time. Recent works have shown several microarchitectural attacks in single-tenant FPGA-accelerated cloud environments across components.

Giechaskiel et al. [68] demonstrate that while the FPGAs themselves are not shared among different users, other parts of the data center infrastructure are, including Power Supply Unit (PSU). As a result, this shared PSU between FPGAs, GPUs, and CPUs can be exploited to build remote FPGA-to-FPGA, CPU-to-FPGA, and GPU-to-FPGA covert channels between independent boards, without the need for physical access.

Tian et al [192] observe that memory accesses between the host computer and an FPGA board become a bottleneck when two or more FPGAs from the same Non Uniform Memory Access (NUMA) node within a server are accessing memory simultaneously. Through observing PCIe contention, they can find which FPGAs are co-located in the same NUMA node within a server. Giechaskiel et al.[70] exploit the interference and bus contention on shared PCIe bus to conduct covert channel and side channel within a NUMA node to leak information across separate virtual machines running in an FPGA cloud.

Additionally, integrated CPU-FPGA platforms connect the FPGA tightly into the CPU bus interconnect giving the FPGA direct access into CPU last level cache and memory [88]. This tight integration provides an adversary with a co-location opportunity on the system across the components. Weissman et al. [215] develop shared last level cache based covert channel attacks between CPU and integrated FPGA.

In a different threat model, Ye et al. [231] study cross-component attacks in CPU-FPGA embedded systems using malware and hardware trojan: 1) CPU to FPGA attacks, accessing the secret data in shared Block RAM (BRAM) by a malicious application on CPU side through DMA or monitoring the data transfer on the bus line, 2) FPGA to CPU attacks, embedding a hardware trojan into FPGA to leak or modify the secret data on CPU side. To mitigate these attacks, they also extend the CPU-based hardware isolation primitive to the heterogeneous FPGA components (we will discuss it in Section 5.4).

While allowing multi-tenant FPGAs can improve the resource utilization on the cloud, it is still under investigation and not currently deployed in cloud computing architectures. However, researchers in academia and industry have been investigating the security of spatial sharing of FPGAs where the FPGA resources are spatially shared among multiple users (i.e., it holds multiple designs at the same time) [201]. Sadeghi et al. [235] study the security of multi-tenant FPGAs in clouds against different classes of physical attacks, including remotely-exploitable physical attacks.

Sharing an FPGA fabric between multiple users (IP cores) in multi-tenant FPGAs leads to some attacks through the shared physical resources on the chip. Gnad et al. [72] propose a voltage-based covert channel in multi-tenant FPGAs. They exploit voltage fluctuations in the Power Distribution Network (PDN) (shared for the full FPGA chip) to build a covert channel between two users that reside in two different logically isolated FPGA areas. Glamocanin et al.[71] also exploit the shared PDN between multiple FPGA tenants in a real cloud system to develop a remote power-analysis attack and recover AES key.

Given the distributed interconnect in FPGAs, even if the logic elements for different sub-circuits are isolated, their routing resources in channels may be in close proximity. Giechaskiel et al. [67] observe that a long routing wire carrying a logical value impacts the delay of other adjacent long wires in the FPGA interconnect, thereby leaking information about its state. The paper constructs a covert channel through this interference. Ramesh et al. [166] exploit the same leakage to build a side channel to extract AES encryption keys by adding a snooping (receiver) circuit. Giechaskiel and Szefer[69] identify that medium wires are also affected by this crosstalk. They demonstrate that medium wires of victim designs can leak information to attacker designs routed on adjacent wires. They also present a new source of information leakage through the internal structure of Configurable Logic Blocks (CLBs). They observe that the delays of elements using multiplexer (MUX) outputs are sensitive to the values of nearby Lookup Tables (LUTs). As a result, an adversary can exploit this leakage to detect changes in the inputs and outputs of victim logic. These attacks exploit physical effects in multi-tenant FPGAs.

Many physical covert and side channel (e.g. thermal or power-based) have been studied on FPGAs. Many of these physical attacks require physical access to the FPGAs for the leakage measurement. Some works propose physical side channel attacks to extract FPGA's bitstream encryption key through power [137] or thermal laser stimulation [127]. Tian et al. [193] observe that heat generated by one user on cloud FPGAs can be observed by another user who later uses the same FPGA. They exploited this leakage to construct covert channel.

Trimberger et al. [195] review several categories of attacks on FPGA: cloning/overbuilding, reverse engineering to steal the design, tampering, spoofing, and denial of service. To study the secure use of FPGAs in the cloud, Turan et al. [199] survey existing researches that target secure remote FPGA configuration, the protection of intellectual property, and a discussion on secure shared use of FPGAs. All of these attacks and many similar physical attacks are out of the scope of this survey. Our focus is on microarchitectural attacks, so we do not discuss these types of attacks in detail in this article.

However, some of these physical attacks can be conducted remotely, for example through access to integrated voltage or power sensors, or implemented using a co-located malicious design. Remote power based attacks assume that the adversary has access to some of the lookup tables (LUTs) in the remote FPGA shared with the victim and can implement a power monitor on the FPGA fabric. Gravellier et al. [76] use FPGA voltage sensors to implement a remote power based side channel attack running on FPGA to spy on CPU computation, specifically to retrieve the secret key of the AES crypto-algorithm. Zhao and Suh [236] demonstrate the same attack on RSA encryption, from FPGA to CPU in an SoC.

In general, FPGA security against microarchitectural attacks and potential secure design principles remain in the early stages. As sharing models for FPGAs continue to mature, and they continue to be integrated within heterogeneous systems, we believe that these threats will grow. Anticipating potential attacks and developing techniques to enable sharing without vulnerabilities is important at this stage of FPGA development.

5.2.2 Fault Injection Attacks: Fault injection attacks have also been studied on FPGAs. Many of these physical attacks require physical access for stimulation to inject the faults. Canivet et al. [43] presents voltage glitch and laser fault injection attacks against a secured AES architecture implemented on FPGA. Rakin et al. [165] propose

Table 2. Summary of microarchitectural attacks on FPGA-based systems (including remote side channels and SW-based fault injection attacks)

	Attack	**Threat model**	**Leakage**	**Structure**
Giechaskiel et al. [68]	covert channel (remote)	FPGA-FPGA FPGA-CPU FPGA-GPU	power (voltage fluctuations)	power supply unit
Tian et al. [192]	Reverse-engineer co-location of FPGAs in cloud	cross-FPGA (cross-VM)	contention	PCIe
Giechaskiel et al.[70]	covert channel & side channel	cross-FPGA (cross-VM)	contention	PCIe
Gnad et al. [72]	covert channel (remote)	multi-tenant FPGA (cloud or SoC)	power (voltage fluctuations)	power distribution network
Glamocanin et al.[71]	side channel (remote)	multi-tenant FPGA (cloud)	power (voltage fluctuations)	power distribution network
Gravellier et al. [76]	side channel (remote)	FPGA-CPU (SoC)	power	time-digital converter (TDC)
Zhao and Suh [236]	side channel (remote)	FPGA-FPGA FPGA-CPU (SoC)	power (voltage fluctuations)	power distribution network
Weissman et al. [215]	covert channel, side channel & Rowhammer	CPU-FPGA	timing	LLC, DRAM
Krautter et al. [113]	SW-based fault injection attack	multi-tenant FPGA (cloud or SoC)	power (voltage fluctuations)	power distribution network

a hardware based fault injection attack on a multi-tenant FPGA. They exploit and overload the shared power distribution system (PDS) to cause voltage drooping. By timing this drooping carefully, they are able to cause duplication of selected DNN model weight parameters of the victim tenant during data transmission between off-chip memory and on-chip buffer. Mahmoud et al.[130] show that an adversary can use the FPGA fabric to create a significant supply voltage drop, and as a result, inject faults in the software computation performed by the CPU in an integrated CPU-FPGA SoC.

Recently, some software-based fault injection attacks have been developed on FPGAs. Krautter et al. [113] demonstrate that a spatially and logically separated attacker in one region of the FPGA fabric can inject faults to a victim in another region. They exploited supply voltage drops generated by means of malicious switching activity to develop software-initiated fault attacks in multi-tenant FPGAs. Weissman et al. [215] develop Rowhammer attacks from the FPGA to the host's main memory in integrated CPU-FPGA systems. Table 2 summarizes the microarchitectural attacks on FPGA systems that were discussed in this subsection.

5.3 Security of Domain-Specific Hardware Accelerators

Recently, domain-specific accelerators have emerged to continue to improve performance and efficiency of computing systems. Many Application-Specific Integrated Circuit (ASIC) accelerators specialized for a particular domain of applications are being proposed; for example, these include accelerators for ML/AI/Deep learning [61, 74], audio/video/image processing[159, 172], cryptography[144, 221], and many other tasks[51, 82, 164, 198]. ASIC accelerators exploit specialized operations on domain-specific data types, parallelism, as well as local and optimized memory to offer orders of magnitude improvements in performance compared to general-purpose processors such as CPUs, or even GPGPUs. In this subsection, we overview the security of this emerging class of hardware accelerators and possible future directions.

5.3.1 ML/AI Accelerators: The importance of machine learning workloads has recently played a critical role in the design of computing hardware and software systems targeted towards making them more efficient for both training and inference and at a variety of scales. For example, there are numerous proposals for accelerating the performance of inference and training, leading to products such as Google's Tensor Processing Units (TPUs) [74], and Tensor Cores integrated with Graphical Processing units in new generations of GPUs [26], as well as a class of accelerators being termed Neural Processing Units (NPUs) [61] already being integrated with high-end Mobile SoCs and phones [11]. Moreover, a large number of systems and accelerators dedicated to accelerating machine learning training and inference [38, 39, 48, 86, 179, 205] continue to emerge.

These emerging accelerators with unique architectures and optimization infrastructures introduce potential new microarchitectural attacks. In addition, existing hardware and microarchitectural attacks can be generalized to these new accelerators. For example, present works on microarchitectural side channel attacks on GPU-accelerated ML and DNN model extraction [87, 96, 142] can be implemented on specialized ML accelerators such as TPUs and NPUs. In addition, some efforts have already demonstrated that fault injection attacks can be used to compromise ML classifiers [126], and can be generalized to the systems with specialized ML accelerators.

Another class of critical attacks on ML is adversarial attacks [73] that manipulate the inputs of ML forces the ML model to misclassify and cause incorrect results that can be beneficial to the attacker. To efficiently detect adversarial examples and defend against (or increase the robustness of ML models against) adversarial attacks, recent works rely on hardware acceleration (e.g. [171, 208], or hardware support through approximate computing [81]. This class of attacks and defenses is out of the scope of this survey.

5.3.2 Other ASIC accelerators: Although most of ASIC accelerator products and proposals have been focusing on ML/AI application domain, due to proliferation of Ml-based application in every domain of computations at all scale, other domains of applications can significantly benefit from specialized accelerators. This includes audio, video, and image processing[159, 172], as well as cryptography[144, 221] and many other workloads. All of these applications work with sensitive data and/or algorithms; any security vulnerabilities in these domain-specific accelerators will lead to information leakage or integrity violation of systems. However, to the best of our knowledge, security of these ASIC accelerators have not been studied so far. We believe it is the time to investigate the threat models and design secure architectures and integration of these accelerators with the rest of the system, before these specialized accelerators are widely deployed for critical applications and in critical environments.

5.4 Defense Mechanisms for Accelerators

In this section, we study the defense mechanisms for accelerators in two important classes. We first review the existing defense mechanisms against microarchitectural attacks and emphasize the necessity of secure designs at a critical time when accelerators are being widely deployed in multi-tenant clouds and virtualized environments. Then, we survey the existing works that extend the trusted execution to the accelerators, which is a timely and critical research area in the security of accelerators in heterogeneous systems.

Defense mechanisms against microarchitectural attacks. To defend against microarchitectural attacks, mitigations can be designed at different levels of software and hardware. There are many defense proposals to close micro architectural covert and side channel attacks on the CPUs which mostly focus on caches and memory controllers. These proposals include (1) Static or dynamic partitioning of resources like L1 cache or LLC [55, 92, 121, 152, 162]. (2) Randomizing memory-to-cache mapping, including randomization in the replacement of the cache lines in the entire cache and in the cache fill strategy [122, 158, 161, 170, 213, 216]. (3) Adding noise to timing by manipulating the time measurement structure of processor [132]. (4) Traffic control in memory controllers [178, 211]. (5) Online detection of contention-based covert communication [47, 53, 101, 150, 224].

A few defense mechanisms have been also proposed to mitigate covert and side channels on GPUs [97, 98, 146, 223] that are discussed in detail in Section 5.1.

With increasing deployment of accelerators including, FPGAs, AI accelerators, and others, in virtualized environments and multi-tenant clouds, co-location and as a result sharing microarchitectural resources will offer unique opportunities for the attacker to build novel microarchitectural attacks in the future. This necessitates careful study and development of systematic detection tools and defense mechanisms to reduce the threat posed by these attacks and secure the accelerators and the entire heterogeneous systems.

In addition to design defense mechanisms to minimize the interference of concurrent processes within a single component (either CPU or accelerator) and secure each component in isolation, the defense designs in heterogeneous systems require to consider the system-wide security, functionality, performance, and power goals.

Heterogeneous system architectures are employed in different computing contexts: end-user devices and computational clouds, as well as high performance computing (HPC) clusters. Each of these computing environments has a unique architecture, connecting different components based on the performance and efficiency of the system, leading to special security concerns and challenges to be addressed.

Trusted Execution Environments on Accelerators. To protect sensitive code and data, hardware manufacturers have started integrating trusted hardware in CPUs in the form of trusted execution environments(TEE), such as Intel SGX [133] and ARM Trust-zone [22]. TEE hardware protects the sensitive code and data from administrators and from attackers who control privileged software, including the operating system and the hypervisor. Currently, offloading massive computation to the accelerators (especially on the clouds) requires enormous trust in providers and administrators. However, these TEEs are limited to CPU and can not protect the offloaded sensitive data and code to the accelerators. In recent years, several works extended the TEE support to accelerators such as GPUs and FPGAs.

Volos et al. [206] propose Graviton, an architecture framework to support TEEs on GPUs. In Graviton, sensitive kernel and data that are offloaded to the GPU, are cryptographically bound to a public/private key pair (the key is hosted in a CPU TEE). Graviton executes kernels in isolation from other contexts running on the GPU and all untrusted software on the host, including the device driver, the operating system, and the hypervisor. In Graviton, the GPU hardware is modified to prevent the device driver from directly accessing critical GPU interfaces, such as communication channels and page table entry. Jang et al. [93] propose HIX (Heterogeneous Isolated eXecution), a hardware and software framework to isolate the I/O interconnect and GPU driver from the control of the OS, without requiring any change to the hardware GPU architecture. HIX uses hardware extensions of the I/O components and SGX supports in the CPU side to protect the discrete GPU platform connected with PCIe buses against attacks from compromised privileged software.

Hunt et al. [89] demonstrate that GPU TEEs can not protect against timing side channels from the CPU-side that exploits secret-dependent communications between CPU and GPU. They propose Telekine, a secure GPU computing in the clouds that makes communication with the GPU TEE data oblivious, that is, completely independent of secrets contained in the input data. With a novel GPU stream abstraction, Telekine protects the application and GPU runtime by moving it from the cloud to the client and decouples the user library from low-level GPU control, and forwarding these calls to the server (a technique known as API remoting). As a result, it ensures secret-dependent behaviors occur only on trusted components, and execution and interaction through untrusted components are independent of any secret data.

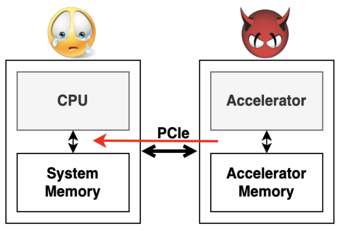
As GPUs are used to process and render the display output in computing systems, a trusted display service is required to maintain the confidentiality and authenticity of content output by a security-sensitive application. A trusted display prevents a compromised operating system or application from reading or modifying the displayed output. Yu et al. [232] propose a trusted display with minimal trusted code base for GPU-based platforms without the need to any modification of OS/Apps code and GPU hardware or reduce their functionality. Their proposed design relies on a GPU separation kernel that (1) defines different types of GPU objects, (2) mediates access to security-sensitive objects, and (3) emulates objects whenever required by commodity-platform compatibility.

To support the hardware trusted execution in CPU-FPGA heterogeneous systems, Ye et al. [231] propose HISA, a hardware isolation-based secure architecture. HISA adopts a CPU-based hardware isolation mechanism to physically separate the system components, including the processes and data in the CPU domain and the third-party IP cores in the FPGA domain, into a secure world and a normal world. This hardware isolation at the physical bus level of the system ensures that the normal world cannot directly access the secure world, so prevents the attack flows from the normal world. HISA also deploys a secure agent in the secure world, which enforces a set of security policies to block the illegal access from the attack flow (i.e., access control policies) and prevent sensitive information from being leaked (i.e., output verification policies). The security policies ensure that the secure world resources and services are accessible only by legitimate processes or IP cores, while the attack flows in both directions between CPU and FPGA are blocked.

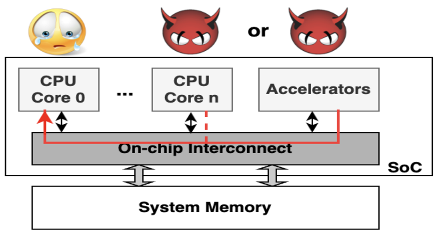
6 ATTACKS ON INTERCONNECT

In heterogeneous systems, interconnect is a component that enables the communication between the different units (e.g. CPU cores, accelerators, last level cache, and system agent). We classify attacks on the interconnect into two different classes: physical layer and protocol layer attacks. We elaborate on each below.

- Physical layer attacks: Interconnect links (either on-chip or off-chip) are shared across processing units to enable data transfer and offer significant benefits to efficiency and cost. The access from multiple components to the bandwidth and throughput limited interconnects can facilitate contention-based and cross-component microarchitectural attacks. This threat model has been recently started to be explored in heterogeneous systems. Additionally, shared memory controllers and routing logic across CPU and accelerators can also be targets of this threat model in integrated heterogeneous systems, that require further study in the future.
  In another scenario, new technologies (such as RDMA [99]) have emerged in clouds and clusters for remote direct access to the cache or memory of a machine over the network. As a result, a new class of microarchitectural attacks has appeared that exploit this remote access to launch cache-based side channel attacks between two client machines.
- Protocol layer attacks: Some specific protocols are designed and implemented for consistent data sharing among different components that are physically connected to each other and also to the system memory by interconnects. The most well-known protocol in multi-more processors is the cache coherence protocol that preserves data coherence in private caches.
  Several works exploited cache coherency protocols in multi-core CPUs to build microarachitectural covert and side channel attacks. Yao et al. [226] demonstrates timing covert channels exploiting shared cache coherency protocol states. Yan et al. [225] exploited directory-based cache coherency protocol in modern processors and use directory to develop timing side channel attacks in multi-core CPUs with non-inclusive cache hierarchy. Trippel et al. [196] leverage the cache line invalidation mechanism in modern cache coherence protocols and demonstrate that by exploiting invalidation messages that are sent to sharer cores on a write request, it is possible to develop Meltdown/Spectre attack.
  All of these proposed attacks are developed within a multi-core CPU, and so far, this threat model has not been investigated in the context of heterogeneous systems. Today, with the increasing interest in developing heterogeneous cache coherent SoCs and fully coherent accelerators in both academia and industry [13, 23, 49, 114, 153], this class of attacks (protocol-based) become more dangerous and relevant in



(a)



(b)

Fig. 6. Thread model of contention-based attacks on (a) PCIe bus; (b) On-Chip Interconnect

integrated heterogeneous systems across the components (e.g. between CPUs and accelerators) and require comprehensive investigation.

In this section, we review the existing attacks that are developed through the physical layer of interconnect in heterogeneous systems in two main categories: (1) Contention based covert and side channel attacks on on-chip and off-chip interconnects between multiple processing units, and (2) Covert and side channel attacks over the network through remote direct access to the cache or memory.

6.1 Contention based attacks on Interconnects:

Recent works developed contention based side channel attacks on on-chip or off-chip interconnects in multi-core CPUs or heterogeneous systems with accelerators. In such scenarios, contention arises when the two components share a bandwidth or capacity limited microarchitectural structure such as buses or ports. When the two processes running on the two components access the same structure concurrently, a measurable contention can be achieved (observing slowdowns). These threat models are demonstrated in Figure 6 in discrete accelerator-CPU systems through off-chip PCIe bus and also in integrated CPU-accelerator systems through on-chip interconnect (either between CPU cores, or CPU and accelerator). Table 3 summarizes contention-based attacks on interconnects in heterogeneous systems.

Tan et al. [186] developed timing attacks through congestion in Peripheral Component Interconnect express (PCIe) bus that connects CPU and peripheral devices like GPU, Network Interface Card (NIC), and SSD drive. Congestion happens when the collective PCIe traffic from the devices overwhelm the PCIe link capacity, and as a result, transmission delay is introduced. By exploiting this congestion, they develop two attacks: using Remote DMA NIC to attack GPU and using Non-Volatile Memory Express (NVMe) SSD to attack Ethernet NIC, and spy on keystroke typing, webpage browsing, and training machine-learning model.

We already showed that a number of accelerator (or cross-component) attacks also use signals from the interconnection buses such as PCIe bus or accelerators' internal buses [33, 70, 87, 192]. Similarly, Zhu et al [237] developed a model extraction attack on a DNN running on discrete GPU, through monitoring the traffic over PCIe bus that connects CPU and GPU.

Paccagnella et al. [151] leverage the contention on the ring interconnect connecting multiple cores in CPUs to extract key bits from vulnerable EdDSA and RSA implementations, as well as infer the precise timing of keystrokes typed by a victim user. In a similar work, Wan et al. [207] exploit the contention on mesh interconnect in recent Intel server CPUs to partially extract the RSA private key.

In integrated heterogeneous systems, accelerators are tightly integrated on the same chip as CPUs and also system memory through the on-chip interconnect. Dutta et al. [56] developed covert channels across different processors on an SoC (CPU and integrated GPU) through contention on the ring bus connecting them.

Contention based channels on interconnects require further investigation on modern interconnect architectures and technologies on different computing platforms. For example, recent AMD Accelerated Processing Units (APUs) use Infinity Fabric/Architecture [37] to interconnect CPU, GPU, and other accelerators. In ARM SoCs, big.LITTLE Cortex CPUs are connected with Mali GPUs in a single chip using CoreLink Cache Coherent Interconnect [40]. This interconnect provides full coherency for CPU processor clusters and also high performance I/O coherency for accelerators (GPU) and other interfaces.

Additionally, contention based attacks can be generalized to many other shared resources on heterogeneous systems, including shared cache ports, memory controller, and routing logics.

Table 3. Summary of contention-based attacks on interconnects

	Attack	**Threat model**	**Interconnect**
Tan et al. [186]	side channel (keystroke, website fingerprinting & ML model inference)	NIC-GPU & NVMe SSD-NIC	PCIe
Tian et al [192]	Reverse engineering co-located FPGAs in clouds	cross-vm (cross-FPGA)	PCIe
Giechaskiel et al.[70]	covert and side channel	cross-vm (cross-FPGA)	PCIe
Zhu et al [237]	side channel (DNN model extraction)	CPU-discrete GPU	PCIe
Hu et al.[87]	side channel (DNN model extraction)	from CPU & co-located spy on GPU	GPU's memory bus and PCIe
Ahn et al. [35]	covert channel	co-located spy on GPU	GPU's internal NoC
Paccagnella et al. [151]	side channel (EdDSA & RSA key extraction)	cross-core in CPU	ring interconnect
Wan et al. [207]	side channel (RSA key extraction)	cross-core in server CPU	mesh interconnect
Dutta et al. [56]	covert channel	integrated CPU-GPU	ring interconnect

6.2 Attacks through remote direct access to cache or memory:

As many AI and HPC applications work on datasets that continue to increase in size, slow I/O and communication become a bottleneck for computation resources. To address this issue many cloud vendors use newer technologies like Non-Volatile Memory Express (NVMe) for local storage, NVMe over fabric [20] for remote storage, Remote Direct Memory Access (RDMA) Network Interface Controllers (NICs) [9, 99], GPUDirect [19], GPUDirect RDMA [17] and more recently GPUDirect Storage [191] which enables a direct data path between two local or remote GPUs or between a GPU and storage. The aforementioned hardware devices that enable one-sided communication need to bypass the host's CPU and software stack to achieve high performance.

Several works exploited the direct access to memory or last level cache (bypassing the CPU) to develop remote side channel attacks (summarized in Table 4). The threat model overview is demonstrated in Figure 7.

Tsai et al. [197] conduct a timing channel on RDMA NICs that allows an attacker on one client machine to learn how victims on other client machines access data a server exports. Since, the RDMA bypasses the CPU and software stack in the destination nodes, the RDMA NICs (RNIC) cache different types of metadata like page table entries in its own SRAM to be able to perform remote accesses to the memory. When the SRAM is full, RNICs swap metadata to main memory across the PCIe bus. This attack tries to exploit the timing difference when these metadata are cached and establish a side channel to leak the access pattern of the victim to the attacker.

Kurth et al. [115], propose another side channel over the network. Intel's Data-Direct I/O (DDIO) [16] enabled processor performs I/O directly on the last level cache (LLC). Instead of having Direct Memory Access, DDIO uses Direct Cache Access (DCA) to improve the performance. So, the LLC is shared between the CPU and all peripheral devices, including the network card. The proposed attack exploits the timing difference between a network packet that is served from the remote processor's cache versus a packet served from memory and enables the attacker to perform the PRIME+PROBE attack on some part of the LLC and leak sensitive information over the network.

In a concurrent work, Taram et al. [188], showed another network-based attack which can locate the network buffer in the cache as well as the packet size and sequence. Using this information, an attacker would be able to create a covert and side channel to leak sensitive information like a trace of victim web access activity. The proposed attack exploits the same underlying vulnerability that has been exploited in [115]. However, [115] only detects the arrival time of the packets but this attack detects both arrival time and packet size which is less noisy. Also, [188] showed that unlike the proposed attack in [115] which needs both DDIO and RDMA to be enabled, this attack is still possible in absence of those technologies, by exploiting the latency between I/O writes and driver reads.

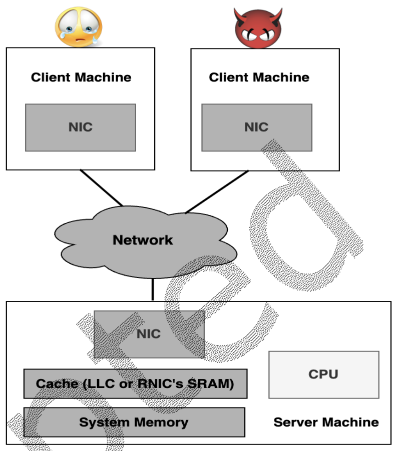


Fig. 7. Threat model of attacks over network through direct access to cache or memory.

Ustiugov et al. [200] introduce Bankrupt, a covert channel between the spy and the receiver running on different nodes in an RDMA network. In Bankrupt, the spy communicates with the receiver by issuing RDMA network packets to a private memory region allocated to it on a different machine (an intermediary). The receiver similarly allocates a separate memory region on the same intermediary, also accessed via RDMA. By steering RDMA packets to a specific set of remote memory addresses, the spy causes deep queuing at one memory bank, which is the finest addressable internal unit of main memory. This exposes a timing channel that the receiver can listen to by issuing probe packets to addresses mapped to the same bank but in its own private memory region.

Although it is not well investigated yet, we believe this threat model can be generalized to other components of a heterogeneous system, for example on GPUs using GPUDirect [19], or GPUDirect RDMA [17].

Table 4. Summary of RDMA-based attacks in heterogeneous systems

	Attack	**Threat model and attacker's goal**
Tsai et al. [197]	side channel	steal access patterns of victims on other machines through RDMA NIC
Kurth et al. [115]	side channel	keystroke timing attack on SSH connection of victim client through DCA from NIC on LLC
Taram et al. [188]	covert and side channel	steal the web page access patterns of a victim on the network from NIC on LLC
Ustiugov et al. [200]	covert channel	communication of processes on different nodes in RDMA network

7 ATTACKS ON MEMORY

Memory is a key component of all modern computing systems. Conventional memory technology (DRAM) has been facing many challenges in terms of reliability, energy, and performance. From the security and reliability aspects, aggressive DRAM technology scaling has lead to new security vulnerabilities, RowHammer attacks [108] in most modern DRAM chips.

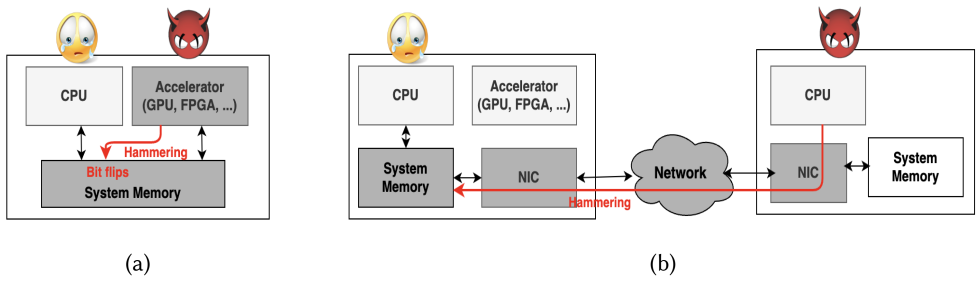


Fig. 8. Threat model of Rowhammer attacks in heterogeneous systems through (a) Accelerators; (b) Network (and NIC)

Furthermore, memory systems have been evolving in different ways to overcome memory bottlenecks and offer large, reliable, and energy-efficient memory systems. The emergence of new memory technologies and architectures such as Non-Volatile Memories (NVMs) and 3D-stacked memory, as well as trends to move processing in memory (PIM), make memory and storage other points for malicious attacks to start.

In this section, we first review the existing Rowhammer attacks on DRAM that are exploited through different components (rather than CPUs) in the context of heterogeneous systems. Next, we investigate the future directions on the security of emerging memory technologies/architectures.

7.1 Rowhammer Attacks in Heterogeneous Systems

The most prominent fault injection attack in computing systems is the exploitation of Rowhammer bug in DRAM, initially introduced in [108]. Different implementations and exploitation techniques have been developed to exploit this bug in native environments [63, 78, 203] to gain kernel privileges by triggering bit flips on page table entries, in web browsers to escape the JavaScript sandbox [79], and also from the virtual machines in the clouds [167, 220]. Rowhammer vulnerability has been also exploited to induce bit flips in model weights to compromise DNN inference [227]. All of these Rowhammer attacks have been studied on CPU-based systems with different architectures (e.g. ARM, Intel x86). In this section, we review Rowhammer attacks that have been developed through other components of heterogeneous systems (rather than CPUs) on the system memory. In recent years, Rowhammer has been exploited through accelerators and also network cards over fast networks (summarized in Table 5). Figure 8 demonstrates these two categories of threat models.

Table 5. Summary of Rowhammer attacks in heterogeneous systems

	Threat model	**Exploitation**
Frigo et al. [62]	From GPU in integrated CPU-GPU systems through JS	escape the Firefox JS sandbox
Weissman et al. [215]	From FPGA in integrated CPU-FPGA systems	WolfSSL RSA Fault Injection
Tatar et al. [189]	from RDMA NIC	gain code execution on a remote key-value server application
Lipp et al. [120]	from RDMA NIC	kernel image corruption & bit flips in user-space executable

In integrated heterogeneous systems, the memory subsystem is shared across CPU and accelerators. As a result, accelerators have direct access to the system memory and can be utilized to launch and accelerate Rowhammer attacks. In the last few years, several works demonstrated Rowhammer attacks in such systems, from accelerators (GPU, or FPGA) to the host's main memory. Frigo et al. [62] implement GPU accelerated Rowhammer attack on memory in integrated CPU-GPU systems in mobile SOCs. They develop the attack in the web browser through WebGL (GPU-accelerated web interface) and exploit bit flips from Rowhammer to escape the Firefox sandbox on android platforms. Specifically, using bit flips in memory, they break ASLR and obtain arbitrary read and write primitives for remote code execution. Weissman et al. [215] study Rowhammer attack from the FPGA to the CPU main memory in integrated CPU-FPGA systems. They develop a practical fault injection attack from FPGA to RSA implementation running on its host CPU. This threat model can be generalized to other accelerators in integrated systems (e.g. AI accelerators) with direct access to system memory.

Besides accelerators, several works develop remote Rowhammer attacks over the network and NIC. Tatar et al. [189] propose Throwhammer that exploits Rowhammer directly from a remote machine by only transferring packets over RDMA-enabled networks and as a result, accessing remote DMA buffers very quickly. RDMA enabled network cards are deployed in data centers and the cloud to improve the performance of their clusters. Throwhammer uses these Rowhammer bit flips induced by network traffic to gain code execution on a remote key-value server application. In a concurrent work, Lipp et al. [120] propose Nethammer, a remote Rowhammer by repeatedly sending packets over the fast network connection between the attacker and victim. As frequently-used addresses are served from the cache for performance, the cache must be bypassed such that the access goes directly into the DRAM to cause the row conflicts required for hammering. In Nethammer, if the network driver or other parts of the network stack use uncached memory or flush instructions for interaction with the network device, an attacker can induce bit flips. They also demonstrate that even without uncached memory and flush instruction, attacks on cloud systems can still be practical if Intel CAT (Cache Allocation Technology (CAT) to address quality of service in multi-core server platforms) is activated and as a result, memory accesses lead to fast cache eviction and thus frequent DRAM accesses. Lipp et al. [120] demonstrated that bit flips induced by Nethammer can lead to kernel crashes, or crash the running processes and services.

7.2 Security of Emerging Memory Technologies/Designs

The advancement in data-intensive and high performance computing, e.g. Large scale machine learning and large-scale graph analytics workloads, has increased the demands for more efficient and scalable memory systems besides specialized accelerators. As a result, the cost of memory (DRAM) is becoming an important concern in data centers and other high performance computing facilities dealing with large scale data analysis. Besides the cost of the memory, keeping the leakage power in tolerable limits and bridging the bandwidth gap between processor and memory are two major challenges that need to be addressed in new memory designs.

Non-Volatile Memory: Several promising Non-Volatile Memories (NVMs) such as, Spin-Transfer Torque RAM (STTRAM) [109], Phase Change Memory (PCM) [218], Resistive RAM (ReRAM) [217], and Ferroelectric RAM (FeRAM) [176] are being studied to address the mentioned challenges and offer high density and zero leakage power.

Emerging NVMs memory can be integrated in different levels of memory hierarchy from caches to main memory. Due to promising aspects, emerging NVMs are already being commercialized by industries e.g., Everspin (MRAM) [12], Adesto (ReRAM) [8], Intel/Micron (PCM) [2], and Cypress (FeRAM) [14]. Intel's 3D Xpoint memory [2] is a recent example of NVM's adoption as a cache for Solid State Drives.

The unique characteristics of these emerging memories may lead to security and privacy issues that need to be investigated. Most of these new emerging memories have high write current that can be potentially exploited to launch fault injection [104] attack, Denial-of-Service (DoS) attack, information leakage attack, and Rowhammer attack (e.g. Rowhammer attack on STTRAM [103]). The other common vulnerability among these types of memory is asymmetric read/write current which could potentially lead to side channel attack and more specifically power analysis side channel attack [46, 91].

The other potential vulnerabilities in these types of memory could be the timing side channel attacks[156]. In order to improve the efficiency of reads and writes in NVM memories, many architecture-level performance optimizations have been studied which can lead to timing side channel attack. For example, [85] shows that

accessing different regions of Multi-Level Cell (MLC) PMC have different latency which may lead to a timing side channel attack. Also, when NVM memory is placed in memory hierarchy and interacting with other parts of the memory system, it can potentially create a time side channel attack. For example, when Intel Optane persistent memory [21] operates as main memory in the system and DRAM is considered as the last level cache the timing difference between accessing the DRAM and Non-Volatile RAM (NVRAM) can create a side channel.
Processing-in-Memory: The use of byte-addressable memories and modern 3D-stacked memories [154] enabled a new design paradigm called processing in memory (PIM) [33]. The key idea is to place computation mechanisms in or near where the data is stored (e.g. inside the memory chip or in the logic layer of 3D-stacked memory), as a result, the data movement (which has long latency and energy cost) is reduced or eliminated compared to traditional processor-centric systems.

By eliminating data movement, this new processing paradigm provides opportunities to increase the security of systems in some aspects, such as eliminating the exposure of data and computations to many attacks (e.g. bus snooping attacks). However, it introduces new security concerns that need to be addressed as they are integrated in real world computing systems. For example, placing PIM computation units in/near memory and naively providing shared access to concurrent processes (either launched on a single processor or across different processors/accelerators) may lead to information leakage through covert or side channel attacks. Also, frequently activating and deactivating memory rows to enable PIM computations (especially when computations are done inside the memory chips) can increase the danger of Rowhammer attacks in such memory systems.

8 WEB-ORIGINATING ATTACKS

Thusfar, we reviewed microarchitectural attacks within accelerators and across components in heterogeneous systems when the malicious software is co-located in the victim user's device or co-located on a cloud platform. In this section, we study a new threat model targeting heterogeneous systems through web interfaces.

In recent years, microarchitectural attacks have been implemented in modern web browsers through JavaScript. Although, JavaScript runs in a sandbox, several works demonstrated that it enables a remote attacker to exploit properties inherent to the design of the microarchitecture, such as timing differences in memory accesses to develop microarchitectural side channel attacks[149, 174], as well as Rowhammer attacks [52, 79]. To mitigate these timing attacks in JavaScript, all major browsers limited the resolution of the timer [1, 50, 234], or disabled the JavaScript interfaces that enabled an attacker to build a custom timer (e.g. counting through a shared memory area SharedArrayBuffers [5]). Some works have shown that coarse-grained side channels are still feasible through memory page deduplication timing attack[77] or cache occupancy channels[181, 182]. All of these attacks have been studied in traditional CPU-based systems.

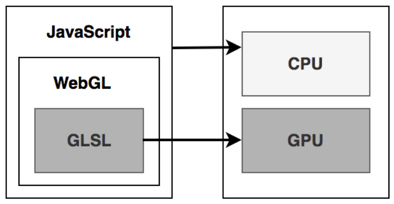


Fig. 9. Structure of a WebGL application.

There is an increasing interest in allowing web-based code to benefit from local accelerators on devices to improve user experience. Web-based APIs are available for accelerators (currently only on GPUs) in heterogeneous systems, providing an additional operating point for attackers. As an example, WebGL is a JavaScript API to accelerate rendering 3D and 2D graphics within web browsers [31] using GPUs. This API enables the attacker to launch GPU-based attacks in JavaScript, leading to dangerous remote attacks that bypass existing JavaScript mitigations. A WebGL application code is a combination of JavaScript and OpenGL Shader Language (GLSL). JavaScript is required to communicate with the CPU and OpenGL Shader Language is required to communicate with and execute on the GPU. Figure 9 demonstrates the structure of a WebGL application.

Yao et al. [228] summarize all WebGL vulnerability disclosures and find that most are related to reading GPU memory (either uninitialized or other process's memory). Based on this analysis and the WebGL security analysis by the Khronos group [32], no GPU side channels have been reported in the context of WebGL, although it is a potential attack vector to launch GPU-based side channels in JavaScript.

Yao et al. [228] leverage modern GPU virtualization to secure GPU acceleration in the web browser by separating WebGL computations into separate virtual GPUs. Yao et al. [229] also propose Milkomeda, a solution to automate and improve the existing WebGL security checks to further increase the security of the mobile graphics interface.

The WebGL API has been utilized to implement browser fingerprinting [44, 184, 194]. Browser fingerprinting refers to the process of collecting information through a web browser to build a fingerprint of a device [157]. Wu et al. [219] propose a software solution called UniGL to protect against WebGL based browser fingerprinting. UniGL redefines floating operations explicitly written in GLSL programs or implicitly invoked by WebGL, such that every device running UniGL will have the exact same WebGL fingerprint for a specific rendering task. Although these defenses are able to contain software vulnerabilities within the WebGL stack, they cannot close side channels or micro-architectural attacks.

Frigo et al. [62] use WebGL timing APIs [106] to implement GPU accelerated Rowhammer attack on memory in integrated CPU-GPU systems. They use WebGL timer as the major clock and implement a variant of clock-edging [111, 175] that executes a (padding) count down over an empty loop before checking for the new timestamp value. In response to this attack, both Chrome and Firefox disabled the WebGL timer [138].

We believe that this attack vector is feasible, although it has not been investigated well. It is critical to understand this threat and how to build these APIs securely at a time when they are being developed [7, 31]. It is likely that additional accelerators such as NPUs or FPGAs may also be made available to web APIs in the future and these principles may also be useful to protect them.

9 CONCLUSION AND FUTURE DIRECTIONS

Heterogeneous system architecture is being evolved in different aspects to provide high performance with energy efficiency. However, the security of these modern systems has received little attention so far. This survey reviews recently introduced attacks that arose beyond the CPU or across several components in heterogeneous systems and also some well-known researches that offer defense mechanisms to secure these systems. We reviewed existing attacks that are developed on widely used accelerators such as GPUs and FPGAs, across components through interconnects, and also on the memory subsystem. As none of these attack categories has been comprehensively investigated so far, we also highlighted potential directions on the security of modern heterogeneous systems in each category of accelerator, interconnect, and memory.

In the future, given the growing number and range of domain-specific accelerators (such as AI accelerators, video, and audio accelerators, and many others) being used in a variety of computing environments with unique system architecture including computational clouds, large scale data-centers, end-user systems, and IoT devices and also for safety-critical applications such as self-driving cars, many new microarchitectural threat models will appear. In addition, secure execution, secure shared use of accelerators in multi-tenant environments (e.g. accelerated cloud platforms), and secure integration of these accelerators to the rest of the system merit further investigation.

In addition to the security investigation of individual components in heterogeneous systems, interactions between these components through emerging high bandwidth and fully coherent buses and interconnects require thorough and systematic explorations. Furthermore, with the introduction of modern memory technologies such as three-dimensional memories, nonvolatile memories, and processing-in-memory designs, novel memory vulnerabilities and attacks arise in the context of heterogeneous systems.

With all of these system architecture innovations, attack surfaces will expand, and as a result, the trust models and security defense tools/mechanisms need to be evolved while these systems are being widely deployed in every computing domain.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. The work is supported by the National Science Foundation under CNS-2053383, CNS-2130978, and CNS-1955650 grants.

REFERENCES

[1] 2015. Chromium: window.performance.now does not support sub-millisecond precision on Windows. https://bugs.chromium.org/p/chromium/issues/detail?id=158234#c110.

[2] 2015. Intel and Micron Produce Breakthrough Memory Technology. https://newsroom.intel.com/news-releases/intel-and-micron-produce-breakthrough-memory-technology/#gs.5irpfz.

[3] 2016. *GRID VIRTUAL GPU.* Technical Report. Nvidia. https://docs.nvidia.com/grid/latest/grid-vgpu-user-guide/index.html.

[4] 2016. *Whitepaper: AMD multiuser GPU: hardware-enabled GPU virtualization for a true workstation experience.* Technical Report. AMD.

[5] 2018. Mitigations landing for new class of timing attack. https://blog.mozilla.org/security/2018/01/03/mitigations-landing-new-class-timing-attack/.

[6] 2019. Nvidia Security Notice. https://nvidia.custhelp.com/app/answers/detail/a_id/4738.

[7] 2020. *WebGPU.* Retrieved 2020 from https://github.com/gpuweb/gpuweb/wiki/Implementation-Status

[8] 2021. Adesto Touts ReRAM for Automotive. https://www.eetimes.com/adesto-touts-reram-for-automotive/.

[9] 2021. Alibaba Cloud. Super Computing Cluster. https://www.alibabacloud.com/product/scc.

[10] 2021. Alibaba Cloud.FPGA-based compute-optimized instance families. https://www.alibabacloud.com/help/doc-detail/108504.html.

[11] 2021. Arm Ethos-N series processors. https://developer.arm.com/ip-products/processors/machine-learning/arm-ethos-n.

[12] 2021. Automotive Temperature Range MRAM. https://www.everspin.com/file/882/download..

[13] 2021. Developing Heterogeneous Cache Coherent SoCs – and More! https://www.computer.org/publications/tech-news/heterogeneous-system-architecture/developing-heterogeneous-cache-coherent-socs-and-more.

[14] 2021. Ferroelectric RAM (FeRAM) – Instant non-volatile memory. https://www.cypress.com/products/f-ram-nonvolatile-ferroelectric-ram.

[15] 2021. Inside the Microsoft FPGA-based configurable cloud. https://azure.microsoft.com/en-us/resources/videos/build-2017-inside-the-microsoft-fpga-based-configurable-cloud/.

[16] 2021. Intel Data Direct I/O Technology. https://www.intel.com/content/www/us/en/io/data-direct-i-o-technology.html.

[17] 2021. Nvidia. Developing a Linux Kernel Module using GPUDirect RDMA. https://docs.nvidia.com/cuda/gpudirect-rdma/index.html.

[18] 2021. NVIDIA DGX-2. https://www.nvidia.com/en-us/data-center/dgx-2/.

[19] 2021. Nvidia GPUDirect. https://developer.nvidia.com/gpudirect.

[20] 2021. NVM Express Moves Into The Future. https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf.

[21] 2021. Optane™ PMem. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html.

[22] 2021. Security on ARM Trustzone. https://www.arm.com/products/security-on-arm/trustzone.

[23] 2021. The Arm CoreLink CCI-550 Cache Coherent Interconnect. https://www.arm.com/products/silicon-ip-system/corelink-interconnect/cci-550.

[24] 2022. CUDA, Nvidia. https://developer.nvidia.com/cuda-zone/.

[25] 2022. MPS. https://docs.nvidia.com/deploy/mps/index.html.

[26] 2022. *NVIDIA V100 TENSOR CORE GPU.* Technical Report. NVIDIA. https://www.nvidia.com/en-us/data-center/v100/.

[27] 2022. OpenCL Overview, Khronos Group. https://www.khronos.org/opencl/.

[28] 2022. OpenGL ES, Khronos Group. https://www.khronos.org/opengles/.

[29] 2022. OpenGL Overview, Khronos Group. https://www.khronos.org/opengl/.

[30] 2022. Vulkan Overview, Khronos Group. https://www.khronos.org/vulkan/.

[31] 2022. WebGL Overview, Khronos Group. https://www.khronos.org/webgl/.

[32] 2022. WebGL Security, Khronos Group. https://www.khronos.org/webgl/security/ https://www.khronos.org/webgl/security/.

[33] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi. 2015. A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. *SIGARCH Comput. Archit. News* 43, 3S (June 2015), 105–117. https://doi.org/10.1145/2872887.2750386

[34] Jaeguk Ahn, Cheolgyu Jin, Jiho Kim, Minsoo Rhu, Yunsi Fei, David Kaeli, and John Kim. 2021. Trident: A Hybrid Correlation-Collision GPU Cache Timing Attack for AES Key Recovery. In *2021 IEEE International Symposium on High-Performance Computer Architecture*

(HPCA). 332–344. https://doi.org/10.1109/HPCA51647.2021.00036

[35] Jaeguk Ahn, Jiho Kim, Hans Kasan, Leila Delshadtehrani, Wonjun Song, Ajay Joshi, and John Kim. 2021. Network-on-Chip Microarchitecture-Based Covert Channel in GPUs *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 565–577. https://doi.org/10.1145/3466752.3480093

[36] Amazon AWS. 2019. Amazon Elastic Graphics. https://aws.amazon.com/ec2/Elastic-GPUs/.

[37] AMD. 2021. CoreLink Cache Coherent Interconnect Family. https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-cache-coherent-interconnect-family.

[38] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, et al. 2019. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 715–731.

[39] Aayush Ankit, Abhronil Sengupta, Priyadarshini Panda, and Kaushik Roy. 2017. Resparc: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks. In *Proceedings of the 54th Annual Design Automation Conference 2017*. 1–6.

[40] ARM. 2021. CoreLink Cache Coherent Interconnect Family. https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-cache-coherent-interconnect-family.

[41] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. 2019. A Systematic Evaluation of Transient Execution Attacks and Defenses. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 249–266. https://www.usenix.org/conference/usenixsecurity19/presentation/canella

[42] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, and Yuval Yarom. 2019. Fallout: Leaking Data on Meltdown-Resistant CPUs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 769–784. https://doi.org/10.1145/3319535.3363219

[43] Gaetan Canivet, Paolo Maistri, Regis Leveugle, Jessy Clédière, Florent Valette, and Marc Renaudin. 2010. Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA. *Journal of Cryptology* 24 (2010), 247–268.

[44] Yinzhi Cao, Song Li, and Erik Wijmans. 2017. (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In *NDSS*.

[45] Luis Ceze, Mark Hill, and Thomas Wenisch. 2016. Arch2030: A Vision of Computer Architecture Research over the Next 15 Years. *ArXiv* abs/1612.03182 (2016).

[46] Abhishek Chakraborty, Ankit Mondal, and Ankur Srivastava. 2017. Correlation power analysis attack against STT-MRAM based cyptosystems. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 171–171. https://doi.org/10.1109/HST.2017.7951835

[47] Jie Chen and Guru Venkataramani. 2014. CC-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware. In *47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'14)*. IEEE, Cambridge UK, 216–228. https://doi.org/10.1109/MICRO.2014.42

[48] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News* 42, 1 (2014), 269–284.

[49] Byn Choi, Rakesh Komuravelli, Hyojin Sung, Robert Smolinski, Nima Honarmand, Sarita V. Adve, Vikram S. Adve, Nicholas P. Carter, and Ching-Tsun Chou. 2011. DeNovo: Rethinking the Memory Hierarchy for Disciplined Parallelism. In *2011 International Conference on Parallel Architectures and Compilation Techniques*. 155–166. https://doi.org/10.1109/PACT.2011.21

[50] Alex Christensen. 2015. Reduce resolution of performance.now. https://bugs.webkit.org/show_bug.cgi?id=146531.

[51] William J. Dally, Yatish Turakhia, and Song Han. 2020. Domain-Specific Hardware Accelerators. *Commun. ACM* 63, 7 (jun 2020), 48–57. https://doi.org/10.1145/3361682

[52] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. 2021. SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 1001–1018. https://www.usenix.org/conference/usenixsecurity21/presentation/ridder

[53] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, and Salvatore Stolfo. 2013. On the Feasibility of Online Malware Detection with Performance Counters. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*.

[54] Robert H. Dennard, Feritz H. Gaensslen, Hwa-Nien Yu, V.Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974), 256–268. https://doi.org/10.1109/JSSC.1974.1050511

[55] Leonid Domnitser, Aamer Jaleel, Jason Loew, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2012. Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks. *ACM Transactions on Architecture and Code Optimization* 8, 4 (2012). https://doi.org/10.1145/2086696.2086714

[56] Sankha B. Dutta, Hoda Naghibijouybari, Nael Abu-Ghazaleh, Andres Marquez, and Kevin Barker. 2021. Leaky Buddies: Cross-Component Covert Channels on Integrated CPU-GPU Systems. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*.

[57] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. 365–376.

[58] Dmitry Evtyushkin and Dmitry Ponomarev. 2016. Covert Channels through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations. In *CCS*.

[59] Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. 2016. Understanding and mitigating covert channels through branch predictors. *ACM Transactions on Architecture and Code Optimization* 13, 1 (2016), 10.

[60] Dmitry Evtyushkin, Ryan Riley, Nael CSE Abu-Ghazaleh, ECE, and Dmitry Ponomarev. 2018. BranchScope: A New Side-Channel Attack on Directional Branch Predictor. 53, 2 (mar 2018), 693–707. https://doi.org/10.1145/3296957.3173204

[61] Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Logan Adams, Mahdi Ghandi, Stephen Heil, Prerak Patel, Adam Sapek, Gabriel Weisz, Lisa Woods, Sitaram Lanka, Stephen K. Reinhardt, Adrian M. Caulfield, Eric S. Chung, and Doug Burger. 2018. A Configurable Cloud-Scale DNN Processor for Real-Time AI. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 1–14.

[62] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In *Proceedings of IEEE Symposium on Security and Privacy*. 357–372. https://doi.org/10.1109/SP.2018.00022

[63] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. TRRespass: Exploiting the Many Sides of Target Row Refresh. In *S&P*. Paper=https://download.vusec.net/papers/trrespass_sp20.pdfSlides=https://download.vusec.net/slides/trrespass_sp20.pdfWeb=https://www.vusec.net/projects/trrespassCode=https://github.com/vusec/trrespassPress=https://bit.ly/2UXWKJ4 Best Paper Award, Pwnie Award for Most Innovative Research, IEEE Micro Top Picks Honorable Mention.

[64] Yiwen Gao, Hailong Zhang, Wei Cheng, Yongbin Zhou, and Yuchen Cao. 2018. Electro-Magnetic Analysis of GPU-Based AES Implementation. In *Proceedings of the 55th Annual Design Automation Conference* (San Francisco, California) *(DAC '18)*. Association for Computing Machinery, New York, NY, USA, Article 121, 6 pages. https://doi.org/10.1145/3195970.3196042

[65] Yiwen Gao, Yongbin Zhou, and Wei Cheng. 2018. How Does Strict Parallelism Affect Security? A Case Study on the Side-Channel Attacks against GPU-based Bitsliced AES Implementation. *IACR Cryptol. ePrint Arch.* 2018 (2018), 1080.

[66] Qian Ge, Y. Yarom, David Cock, and G. Heiser. 2016. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering* 8 (2016), 1–27.

[67] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. 2018. Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea) *(ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 15–27. https://doi.org/10.1145/3196494.3196518

[68] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. 2020. C<sup>3</sup>APSULe: Cross-FPGA Covert-Channel Attacks through Power Supply Unit Leakage. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1728–1741. https://doi.org/10.1109/SP40000.2020.00070

[69] Ilias Giechaskiel and Jakub Szefer. 2020. Information Leakage from FPGA Routing and Logic Elements. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9.

[70] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. 2021. Cross-VM Information Leaks in FPGA-Accelerated Cloud Environments. In *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 91–101. https://doi.org/10.1109/HOST49136.2021.9702277

[71] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. 2020. Are Cloud FPGAs Really Vulnerable to Power Analysis Attacks?. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1007–1010. https://doi.org/10.23919/DATE48585.2020.9116481

[72] Dennis R. E. Gnad, Cong Dang Khoa Nguyen, Syed Hashim Gillani, and Mehdi Tahoori. 2019. Voltage-based Covert Channels in Multi-Tenant FPGAs. *IACR Cryptol. ePrint Arch.* 2019 (2019), 1394.

[73] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2015).

[74] Google Cloud. 2020. Cloud Tensor Processing Units (TPUs). https://cloud.google.com/tpu/docs/tpus.

[75] Google Cloud Platform. 2019. Cloud GPUs. https://cloud.google.com/gpu/.

[76] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, Philippe Loubet Moundi, and Francis Olivier. 2020. Remote Side-Channel Attacks on Heterogeneous SoC. In *Smart Card Research and Advanced Applications*, Sonia Belaïd and Tim Güneysu (Eds.). Springer International Publishing, Cham, 109–125.

[77] Daniel Gruss, David Bidner, and Stefan Mangard. 2015. Practical Memory Deduplication Attacks in Sandboxed Javascript. In *ESORICS*.

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

[79] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer.js: A remote software-induced fault attack in javascript. In *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 300–321.

[80] Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. 2016. Flush+Flush: A Fast and Stealthy Cache Attack. In *DIMVA*.

[81] Amira Guesmi, Ihsen Alouani, Khaled N. Khasawneh, Mouna Baklouti, Tarek Frikha, Mohamed Abid, and Nael Abu-Ghazaleh. 2021. Defensive Approximation: Securing CNNs Using Approximate Computing. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS 2021)*. Association for Computing Machinery, New York, NY, USA, 990–1003. https://doi.org/10.1145/3445814.3446747

[82] Tae Jun Ham, Lisa Wu, Narayanan Sundaram, Nadathur Satish, and Margaret Martonosi. 2016. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–13.

[83] Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. 2010. Understanding Sources of Inefficiency in General-Purpose Chips. In *Proceedings of the 37th Annual International Symposium on Computer Architecture* (Saint-Malo, France) *(ISCA '10)*. Association for Computing Machinery, New York, NY, USA, 37–47. https://doi.org/10.1145/1815961.1815968

[84] John L Hennessy and David A Patterson. 2011. *Computer architecture: a quantitative approach*. Elsevier.

[85] Morteza Hoseinzadeh, Mohammad Arjomand, and Hamid Sarbazi-Azad. 2014. Reducing access latency of MLC PCMs through line striping. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 277–288. https://doi.org/10.1109/ISCA.2014.6853228

[86] Miao Hu, Hai Li, Qing Wu, and Garrett S Rose. 2012. Hardware realization of BSB recall function using memristor crossbar arrays. In *DAC Design Automation Conference 2012*. IEEE, 498–503.

[87] Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, Yu Ji, Xinfeng Xie, Yufei Ding, Chang Liu, Timothy Sherwood, and Yuan Xie. 2020. DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 385–399. https://doi.org/10.1145/3373376.3378460

[88] Jennifer Huffstetler. 2018. Intel processors and FPGAs – better together. https://itpeernetwork.intel.com/intel-processors-fpga-better-together/.

[89] Tyler Hunt, Zhipeng Jia, Vance Miller, Ariel Szekely, Yige Hu, Christopher J. Rossbach, and Emmett Witchel. 2020. Telekine: Secure Computing with Cloud GPUs. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 817–833. https://www.usenix.org/conference/nsdi20/presentation/hunt

[90] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. 2015. S$A: A Shared Cache Attack That Works across Cores and Defies VM Sandboxing – and Its Application to AES. In *2015 IEEE Symposium on Security and Privacy*. 591–604. https://doi.org/10.1109/SP.2015.42

[91] Anirudh Iyengar, Swaroop Ghosh, Nitin Rathi, and Helia Naeimi. 2016. Side channel attacks on STTRAM and low-overhead countermeasures. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 141–146. https://doi.org/10.1109/DFT.2016.7684086

[92] Aamer Jaleel, Eric Borch, Malini Bhandaru, Simon C. Steely Jr., and Joel Emer. 2010. Achieving Non-Inclusive Cache Performance with Inclusive Caches - Temporal Locality Aware (TLA) Cache Management Policies. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[93] Insu Jang, Adrian Tang, Taehoon Kim, Simha Sethumadhavan, and Jaehyuk Huh. 2019. Heterogeneous Isolated Execution for Commodity GPUs. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 455–468. https://doi.org/10.1145/3297858.3304021

[94] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. 2016. A complete key recovery timing attack on a GPU. In *IEEE International Symposium on High Performance Computer Architecture (HPCA'16)*. IEEE, Barcelona Spain, 394–405. https://doi.org/10.1109/HPCA.2016.7446081

[95] Zhen Hang Jiang, Yunsi Fei, and David Kaeli. 2017. A Novel Side-Channel Timing Attack on GPUs. In *Proceedings of the on Great Lakes Symposium on VLSI (VLSI'17)*. 167–172. https://doi.org/10.1145/3060403.3060462

[96] Zhe Zhou Zhou Liy Junyi Wei, Yicheng Zhangy and Mohammad Abdullah Al Faruque. 2020. Leaky DNN: Stealing Deep-learning Model Secret with GPU Context-switching Side-channel. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (Valencia, Spain).

[97] Gurunath Kadam, Danfeng Zhang, and Adwait Jog. 2018. RCoal: Mitigating GPU Timing Attack via Subwarp-based Randomized Coalescing Techniques. In *Proc. International Symposium on High Performance Computer Architecture (HPCA)*. Accessed online at http://adwaitjog.github.io/docs/pdf/rcoal-hpca18.pdf.

[98] Gurunath Kadam, Danfeng Zhang, and Adwait Jog. 2020. BCoal: Bucketing-Based Memory Coalescing for Efficient and Secure GPUs. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 570–581.

[99] Tejas Karmarkar. 2021. Availability of Linux RDMA on Microsoft Azure. https://azure.microsoft.com/en-us/blog/azure-linux-rdma-hpc-available/.

[100] Mehmet Kayaalp, Dmitry Ponomarev, Nael Abu-Ghazaleh, and Aamer Jaleel. 2016. A high-resolution side-channel attack on last-level cache. In *Proceedings of the 53th Annual Design Automation Conference*.

[101] Mikhail Kazdagli, Vijay Janapa Reddi, and Mohit Tiwari. 2016. Quantifying and improving the efficiency of hardware-based mobile malware detectors. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[102] Zijo Kenjar, Tommaso Frassetto, David Gens, Michael Franz, and Ahmad-Reza Sadeghi. 2020. V0LTpwn: Attacking x86 Processor Integrity from Software. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1445–1461. https://www.usenix.org/conference/usenixsecurity20/presentation/kenjar

[103] Mohammad Nasim Imtiaz Khan and Swaroop Ghosh. 2018. Analysis of Row Hammer Attack on STTRAM. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*. 75–82. https://doi.org/10.1109/ICCD.2018.00021

[104] Mohammad Nasim Imtiaz Khan and Swaroop Ghosh. 2018. Fault Injection Attacks on Emerging Non-Volatile Memory and Countermeasures. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy* (Los Angeles, California) *(HASP '18)*. Association for Computing Machinery, New York, NY, USA, Article 10, 8 pages. https://doi.org/10.1145/3214292.3214302

[105] S. Karen Khatamifard, Longfei Wang, Selcuk Köse, and Ulya R. Karpuzcu. 2018. A New Class of Covert Channels Exploiting Power Management Vulnerabilities. *IEEE Computer Architecture Letters* 17, 2 (2018), 201–204.

[106] Khronos. 2016. WebGL timer. https://www.khronos.org/registry/webgl/extensions/EXT_disjoint_timer_query/.

[107] Joonyoung Kim and Younsu Kim. 2014. HBM: Memory solution for bandwidth-hungry processors. In *2014 IEEE Hot Chips 26 Symposium (HCS)*. IEEE, 1–24.

[108] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 361–372.

[109] Emre Kültürsay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 256–267. https://doi.org/10.1109/ISPASS.2013.6557176

[110] 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. 2019. Spectre Attacks: Exploiting Speculative Execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*.

[111] David Kohlbrenner and Hovav Shacham. 2016. Trusted Browsers for Uncertain Times. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 463–480. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kohlbrenner

[112] Esmaeil Mohammadian Koruyeh, Khaled N. Khasawneh, Chengyu Song, and Nael Abu-Ghazaleh. 2018. Spectre Returns! Speculation Attacks using the Return Stack Buffer. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*.

[113] Jonas Krautter, Dennis R. E. Gnad, and M. Tahoori. 2018. FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018 (2018), 44–68.

[114] Snehasish Kumar, Arrvindh Shriraman, and Naveen Vedula. 2015. Fusion: Design tradeoffs in coherent cache hierarchies for accelerators. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. 733–745. https://doi.org/10.1145/2749469.2750421

[115] Michael Kurth, Ben Gras, Dennis Andriesse, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. NetCAT: Practical Cache Attacks from the Network. In *2020 IEEE Symposium on Security and Privacy (SP)*. 20–38. https://doi.org/10.1109/SP40000.2020.00082

[116] J.-B. Lee. 2014. Green Memory Solution. in Samsung Electronics, Investor's Forum.

[117] Zhen Lin, Utkarsh Mathur, and Huiyang Zhou. 2019. Scatter-and-Gather Revisited: High-Performance Side-Channel-Resistant AES on GPUs. In *Proceedings of the 12th Workshop on General Purpose Processing Using GPUs* (Providence, RI, USA) *(GPGPU '19)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/3300053.3319415

[118] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-Based Power Side-Channel Attacks on x86. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 355–371. https://doi.org/10.1109/SP40001.2021.00063

[119] 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. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*.

[120] Moritz Lipp, Michael Schwarz, Lukas Raab, Lukas Lamster, Misiker Tadesse Aga, Clémentine Maurice, and Daniel Gruss. 2020. Nethammer: Inducing Rowhammer Faults through Network Requests. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 710–719. https://doi.org/10.1109/EuroSPW51379.2020.00102

[121] Fangfei Liu, Qian Ge, Yuval Yarom, Frank Mckeen, Carlos Rozas, Gernot Heiser, and Ruby B. Lee. 2016. Catalyst: Defeating last-level cache side channel attacks in cloud computing. In *IEEE International Symposium on High Performance Computer Architecture (HPCA'16)*. Barcelona, Spain, 406–418. https://doi.org/10.1109/HPCA.2016.7446082

[122] Fangfei Liu and Ruby B. Lee. 2014. Random Fill Cache Architecture. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[123] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. 2015. Last-level cache side-channel attacks are practical. In *IEEE Symposium on Security and Privacy (SP'15)*. IEEE, San Jose, CA, USA. https://doi.org/10.1109/SP.2015.43

[124] Sihang Liu, Yizhou Wei, Jianfeng Chi, Faysal H. Shezan, and Yuan Tian. 2019. Side Channel Attacks in Computation Offloading Systems with GPU Virtualization. In *2019 IEEE Security and Privacy Workshops (SPW)*. 156–161.

[125] Xiao Liu, David Roberts, Rachata Ausavarungnirun, Onur Mutlu, and Jishen Zhao. 2019. Binary Star: Coordinated Reliability in Heterogeneous Memory Systems for High Performance and Scalability. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 807–820. https://doi.org/10.1145/3352460.3358262

[126] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 131–138.

[127] Heiko Lohrke, Shahin Tajik, Thilo Krachenfels, Christian Boit, and Jean-Pierre Seifert. 2018. Key Extraction Using Thermal Laser Stimulation: A Case Study on Xilinx Ultrascale FPGAs. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (08 2018), 573–595. https://doi.org/10.46586/tches.v2018.i3.573-595

[128] Chao Luo, Yunsi Fei, and David Kaeli. 2019. Side-Channel Timing Attack of RSA on a GPU. *ACM Transactions on Architecture and Code Optimization* 16, 3, Article 32 (Aug. 2019), 18 pages. https://doi.org/10.1145/3341729

[129] Chao Luo, Yunsi Fei, Pei Luo, Saoni Mukherjee, and David Kaeli. 2015. Side-Channel Power Analysis of a GPU AES Implementation. In *33rd IEEE International Conference on Computer Design (ICCD'15)*. https://doi.org/10.1109/ICCD.2015.7357115

[130] Dina G. Mahmoud, Samah Hussein, Vincent Lenders, and Mirjana Stojilović. 2022. FPGA-to-CPU Undervolting Attacks. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 999–1004. https://doi.org/10.23919/DATE54114.2022.9774663

[131] Giorgi Maisuradze and Christian Rossow. 2018. Ret2spec: Speculative Execution Using Return Stack Buffers *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 2109–2122. https://doi.org/10.1145/3243734.3243761

[132] Robert Martin, John Demme, and Simha Sethumadhavan. 2012. TimeWarp: rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks. In *39th Annual International Symposium on Computer Architecture (ISCA'12)*. Portland, OR, USA, 118–129. https://doi.org/10.1109/ISCA.2012.6237011

[133] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. 2013. Innovative Instructions and Software Model for Isolated Execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy* (Tel-Aviv, Israel) *(HASP '13)*. Association for Computing Machinery, New York, NY, USA, Article 10, 1 pages. https://doi.org/10.1145/2487726.2488368

[134] Wen mei Hwu. 2011. *GPU Computing Gems* (1st. ed.). Elsevier.

[135] Micron. 2016. DDR4 SDRAM Datasheet. p.380.

[136] Microsoft Azure. 2019. GPU-Accelerated Microsoft Azure. http://www.nvidia.com/object/gpu-accelerated-microsoft-azure.html.

[137] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. 2011. On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II FPGAs. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '11)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/2046707.2046722

[138] Mozilla. 2020. WebGL timer extension. https://developer.mozilla.org/en-US/docs/Web/API/EXT_disjoint_timer_query.

[139] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P'20)*.

[140] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. 2020. A Modern Primer on Processing in Memory. arXiv:2012.03112 [cs.AR]

[141] Hoda Naghibijouybari, Khaled Khasawneh, and Nael Abu-Ghazaleh. 2017. Constructing and Characterizing Covert Channels on GPGPUs. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[142] Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael Abu-Ghazaleh. 2018. Rendered Insecure: GPU Side Channel Attacks are Practical. In *Conference on Computer and Communications Security (CCS)*. 2139–2153.

[143] Ajay Nayak, Pratheek B., Vinod Ganapathy, and Arkaprava Basu. 2021. (Mis)Managed: A Novel TLB-Based Covert Channel on GPUs. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (Virtual Event, Hong Kong) *(ASIA CCS '21)*. Association for Computing Machinery, New York, NY, USA, 872–885. https://doi.org/10.1145/3433210.3453077

[144] Hamid Nejatollahi, Nikil D. Dutt, Indranil Banerjee, and Rosario Cammarota. 2018. Domain-specific Accelerators for Ideal Lattice-based Public Key Protocols. *IACR Cryptol. ePrint Arch.* 2018 (2018), 608.

[145] NVIDIA. 2020. *NVLink and NVSwitch*. Retrieved June 5, 2020 from https://www.nvidia.com/en-us/data-center/nvlink/

[146] Nvidia. 2022. Nvidia Multi-Instance GPU. https://www.nvidia.com/en-us/technologies/multi-instance-gpu/.

[147] Lena E. Olson, Jason Power, Mark D. Hill, and David A. Wood. 2015. Border Control: Sandboxing Accelerators. In *48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'15).* Waikiki HI USA, 470–481. https://doi.org/10.1145/2830772.2830819

[148] Lena E. Olson, Simha Sethumadhavan, and Mark D. Hill. 2015. Security Implication of Third-Party Accelerator. *IEEE Computer Architecture Letters* 15, 1 (2015), 50–53. https://doi.org/10.1109/LCA.2015.2445337

[149] Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. 2015. The Spy in the Sandbox: Practical Cache Attacks in JavaScript and Their Implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15).* Association for Computing Machinery, New York, NY, USA, 1406–1418. https://doi.org/10.1145/2810103.2813708

[150] Meltem Ozsoy, Caleb Donovick, Iakov Gorelik, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2015. Malware-aware processors: A framework for efficient online malware detection. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA).*

[151] Riccardo Paccagnella, Licheng Luo, and Christopher W. Fletcher. 2021. Lord of the Ring(s): Side Channel Attacks on the CPU On-Chip Ring Interconnect Are Practical. In *30th USENIX Security Symposium (USENIX Security 21).* USENIX Association. https://www.usenix.org/conference/usenixsecurity21/presentation/paccagnella

[152] Dan Page. 2005. Partitioned Cache Architecture as a Side-Channel Defense Mechanism. In *Crypt. ePrint Arch.*

[153] Neil Parris. 2021. Exploring how Cache Coherency Accelerates Heterogeneous Compute. https://www.computer.org/publications/tech-news/heterogeneous-system-architecture/exploring-how-cache-coherency-accelerates-heterogeneous-compute.

[154] J Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *2011 IEEE Hot Chips 23 Symposium (HCS).* IEEE, 1–24.

[155] Colin Percival. 2005. Cache missing for fun and profit. In *BSDCan.*

[156] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. 2016. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. In *25th USENIX Security Symposium (USENIX Security 16).* USENIX Association, Austin, TX, 565–581. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/pessl

[157] Benoit Baudry Gildas Avoine Pierre Laperdrix, Nataliia Bielova. 2019. Browser Fingerprinting: A survey. In *arXiv.* arXiv:1905.01051

[158] Antoon Purnal, Lukas Giner, Daniel Gruss, and Ingrid Verbauwhede. 2021. Systematic Analysis of Randomization-based Protected Cache Architectures. In *2021 IEEE Symposium on Security and Privacy (SP).* 987–1002. https://doi.org/10.1109/SP40001.2021.00011

[159] Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark Horowitz. 2015. Convolution Engine: Balancing Efficiency and Flexibility in Specialized Computing. *Commun. ACM* 58, 4 (mar 2015), 85–93. https://doi.org/10.1145/2735841

[160] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2019. VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST).* 1–6. https://doi.org/10.1109/AsianHOST47458.2019.9006701

[161] Moinuddin K. Qureshi. 2018. CEASER: Mitigating Conflict-Based Cache Attacks via Encrypted-Address and Remapping. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* 775–787. https://doi.org/10.1109/MICRO.2018.00068

[162] Moinuddin K. Qureshi and Yale N. Patt. 2006. Utility-Based Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches. In *Proceedings of the International Symposium on Microarchitecture (MICRO).*

[163] Hany Ragab, Alyssa Milburn, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2021. CrossTalk: Speculative Data Leaks Across Cores Are Real. In *2021 IEEE Symposium on Security and Privacy (SP).* 1852–1867. https://doi.org/10.1109/SP40001.2021.00020

[164] Shafiur Rahman, Nael Abu-Ghazaleh, and Rajiv Gupta. 2020. Graphpulse: An event-driven hardware accelerator for asynchronous graph processing. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE, 908–921.

[165] Adnan Siraj Rakin, Yukui Luo, Xiaolin Xu, and Deliang Fan. 2021. Deep-Dup: An Adversarial Weight Duplication Attack Framework to Crush Deep Neural Network in Multi-Tenant FPGA. In *30th USENIX Security Symposium (USENIX Security 21).* USENIX Association, 1919–1936. https://www.usenix.org/conference/usenixsecurity21/presentation/rakin

[166] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA Side Channel Attacks without Physical Access. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).* 45–52. https://doi.org/10.1109/FCCM.2018.00016

[167] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. 2016. Flip feng shui: Hammering a needle in the software stack. In *25th {USENIX} Security Symposium ({USENIX} Security 16).* 1–18.

[168] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (Chicago, Illinois, USA) *(CCS '09).* Association for Computing Machinery, New York, NY, USA, 199–212. https://doi.org/10.1145/1653662.1653687

[169] Majid Sabbagh, Yunsi Fei, and David Kaeli. 2020. A Novel GPU Overdrive Fault Attack. In *2020 57th ACM/IEEE Design Automation Conference (DAC).* 1–6. https://doi.org/10.1109/DAC18072.2020.9218690

[170] Gururaj Saileshwar and Moinuddin Qureshi. 2021. MIRAGE: Mitigating Conflict-Based Cache Attacks with a Practical Fully-Associative Design. In *30th USENIX Security Symposium (USENIX Security 21).* USENIX Association, 1379–1396. https://www.usenix.org/conference/usenixsecurity21/presentation/saileshwar

[171] Mohammad Hossein Samavatian, Saikat Majumdar, Kristin Barber, and R. Teodorescu. 2021. HASI: Hardware-Accelerated Stochastic Inference, A Defense Against Adversarial Machine Learning Attacks. *ArXiv* abs/2106.05825 (2021).

[172] Ingo Schmädecke and Holger Blume. 2013. Hardware-accelerator design for energy-efficient acoustic feature extraction. In *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*. 135–139. https://doi.org/10.1109/GCCE.2013.6664775

[173] Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, and Daniel Gruss. 2019. ZombieLoad: Cross-Privilege-Boundary Data Sampling. In *CCS*.

[174] Michael Schwarz, Clémentine Maurice, Daniel Gruss, and Stefan Mangard. 2017. Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript. In *Financial Cryptography*.

[175] Michael Schwarz, Clémentine Maurice, Daniel Gruss, and Stefan Mangard. 2017. Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript. In *Financial Cryptography and Data Security*, Aggelos Kiayias (Ed.). Springer International Publishing, Cham, 247–267.

[176] James F Scott and Carlos A Paz De Araujo. 1989. Ferroelectric memories. *Science* 246, 4936 (1989), 1400–1405.

[177] Amazon Web Services. 2021. Amazon EC2 F1 instances. https://aws.amazon.com/ec2/instance-types/f1/.

[178] Ali Shafiee, Akhila Gundu, Manjunath Shevgoor, Rajeev Balasubramonian, and Mohit Tiwari. 2015. Avoiding Information Leakage in the Memory Controller with Fixed Service Policies. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[179] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 14–26.

[180] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. 2018. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 69–87. https://www.usenix.org/conference/osdi18/presentation/shan

[181] Anatoly Shusterman, Ayush Agarwal, Sioli O'Connell, Daniel Genkin, Yossi Oren, and Yuval Yarom. 2021. Prime+Probe 1, JavaScript 0: Overcoming Browser-based Side-Channel Defenses. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2863–2880. https://www.usenix.org/conference/usenixsecurity21/presentation/shusterman

[182] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. 2019. Robust Website Fingerprinting Through the Cache Occupancy Channel. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 639–656. https://www.usenix.org/conference/usenixsecurity19/presentation/shusterman

[183] Manish Reddy Sparsh Mittal, S. B. Abhinaya and Irfan Ali. 2018. A Survey of Techniques for Improving Security of GPUs. *Journal of Hardware and Systems Security* 2 (2018), 266–285.

[184] Paul Stone. 2013. Pixel Perfect Timing Attacks with HTML5. https://www.contextis.com/media/downloads/Pixel_Perfect_Timing_Attacks_with_HTML5_Whitepaper.pdf https://www.contextis.com/media/downloads/Pixel_Perfect_\Timing_Attacks_with_HTML5_Whitepaper.pdf.

[185] Jakub Szefer. 2016. Survey of Microarchitectural Side and Covert Channels, Attacks, and Defenses. *Journal of Hardware and Systems Security* (2016), 1–16.

[186] Mingtian Tan, Junpeng Wan, Zhe Zhou, and Zhou Li. 2021. Invisible Probe: Timing Attacks with PCIe Congestion Side-channel. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1016–1032. https://doi.org/10.1109/SP40001.2021.00059

[187] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1057–1074. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/tang

[188] Mohammadkazem Taram, Ashish Venkat, and Dean Tullsen. 2020. Packet Chasing: Spying on Network Packets over a Cache Side-Channel. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 721–734. https://doi.org/10.1109/ISCA45697.2020.00065

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

[190] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design Test of Computers* 27, 1 (2010), 10–25. https://doi.org/10.1109/MDT.2010.7

[191] Adam Thompson and CJ Newburn. 2021. GPUDirect Storage: A Direct Path Between Storage and GPU Memory. https://developer.nvidia.com/blog/gpudirect-storage/.

[192] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. 2021. Cloud FPGA Cartography using PCIe Contention. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 224–232. https://doi.org/10.1109/FCCM51124.2021.00035

[193] Shanquan Tian and Jakub Szefer. 2019. Temporal Thermal Covert Channels in Cloud FPGAs. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Seaside, CA, USA) *(FPGA '19)*. Association for Computing Machinery,

New York, NY, USA, 298–303. https://doi.org/10.1145/3289602.3293920

[194] Antonin Durey Vitaly Dyadyuk Pierre Laperdrix Clémentine Maurice Yossi Oren Romain Rouvoy Walter Rudametkin Tomer Laor, Naif Mehanna and Yuval Yarom. 2022. DRAWNAPART: A Device Identification Technique based on Remote GPU Fingerprinting. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium, pages = , numpages = , location = San Diego, CA, USA*. https://doi.org/10.14722/ndss.2022.24093

[195] Stephen M Trimberger and Jason J Moore. 2014. FPGA security: Motivations, features, and applications. *Proc. IEEE* 102, 8 (2014), 1248–1265.

[196] Caroline Trippel, Daniel Lustig, and Margaret Martonosi. 2018. MeltdownPrime and SpectrePrime: Automatically-Synthesized Attacks Exploiting Invalidation-Based Coherence Protocols. *CoRR* abs/1802.03802 (2018). arXiv:1802.03802 http://arxiv.org/abs/1802.03802

[197] Shin-Yeh Tsai, Mathias Payer, and Yiying Zhang. 2019. Pythia: Remote Oracles for the Masses. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 693–710. https://www.usenix.org/conference/usenixsecurity19/presentation/tsai

[198] Yatish Turakhia, Gill Bejerano, and William J. Dally. 2018. *Darwin: A Genomics Co-Processor Provides up to 15,000X Acceleration on Long Read Assembly*. Association for Computing Machinery, New York, NY, USA, 199–213. https://doi.org/10.1145/3173162.3173193

[199] Furkan Turan and Ingrid Verbauwhede. 2020. Trust in FPGA-Accelerated Cloud Computing. *ACM Comput. Surv.* 53, 6, Article 128 (Dec. 2020), 28 pages. https://doi.org/10.1145/3419100

[200] Dmitrii Ustiugov, Plamen Petrov, M. R. Siavash Katebzadeh, and Boris Grot. 2020. Bankrupt Covert Channel: Turning Network Predictability into Vulnerability. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/ustiugov

[201] Anuj Vaishnav, Khoa Dang Pham, and Dirk Koch. 2018. A Survey on FPGA Virtualization. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. 131–1317. https://doi.org/10.1109/FPL.2018.00031

[202] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *Proceedings of the 27th USENIX Security Symposium*. USENIX Association. See also technical report Foreshadow-NG [**?**].

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

[204] Stephan van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. 2019. RIDL: Rogue In-flight Data Load. In *S&P*.

[205] Shreyas Kolala Venkataramanaiah, Yufei Ma, Shihui Yin, Eriko Nurvitadhi, Aravind Dasu, Yu Cao, and Jae-sun Seo. 2019. Automatic compiler based FPGA accelerator for CNN training. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 166–172.

[206] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. 2018. Graviton: Trusted Execution Environments on GPUs. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 681–696. https://www.usenix.org/conference/osdi18/presentation/volos

[207] Junpeng Wan, Yanxiang Bi, Zhe Zhou, and Zhou Li. 2021. Volcano: Stateless Cache Side-channel Attack by Exploiting Mesh Interconnect. *ArXiv* abs/2103.04533 (2021).

[208] Xingbin Wang, Rui Hou, Boyan Zhao, Fengkai Yuan, Jun Zhang, Dan Meng, and Xuehai Qian. 2020. DNNGuard: An Elastic Heterogeneous DNN Accelerator Architecture against Adversarial Attacks. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 19–34. https://doi.org/10.1145/3373376.3378532

[209] Xin Wang and Wei Zhang. 2019. Cracking Randomized Coalescing Techniques with An Efficient Profiling-Based Side-Channel Attack to GPU. In *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy* (Phoenix, AZ, USA) *(HASP '19)*. Association for Computing Machinery, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/3337167.3337169

[210] Xin Wang and Wei Zhang. 2020. An Efficient Profiling-Based Side-Channel Attack on Graphics Processing Units. In *National Cyber Summit (NCS) Research Track*, Kim-Kwang Raymond Choo, Thomas H. Morris, and Gilbert L. Peterson (Eds.). Springer International Publishing, Cham, 126–139.

[211] Yao Wang and G. Edward Suh. 2014. Timing channel protection for a shared memory controller. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*.

[212] Zhenghong Wang and Ruby B Lee. 2006. Covert and Side Channels Due to Processor Architecture.. In *Computer Security Applications Conference (ACSAC)*.

[213] Zhenghong Wang and Ruby B. Lee. 2008. A Novel Cache Architecture with Enhanced Performance and Security. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[214] Ofir Weisse, Jo Van Bulck, Marina Minkin, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Raoul Strackx, Thomas F. Wenisch, and Yuval Yarom. 2018. Foreshadow-NG: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution.

Technical report (2018). See also USENIX Security paper Foreshadow [**?**].

[215] Zane Weissman, Thore Tiemann, D. Moghimi, Evan Custodio, T. Eisenbarth, and B. Sunar. 2020. JackHammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020 (2020), 169–195.

[216] Mario Werner, Thomas Unterluggauer, Lukas Giner, Michael Schwarz, Daniel Gruss, and Stefan Mangard. 2019. ScatterCache: Thwarting Cache Attacks via Cache Set Randomization. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 675–692. https://www.usenix.org/conference/usenixsecurity19/presentation/werner

[217] H.-S. Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T. Chen, and Ming-Jinn Tsai. 2012. Metal–Oxide RRAM. *Proc. IEEE* 100, 6 (2012), 1951–1970. https://doi.org/10.1109/JPROC.2012.2190369

[218] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. 2010. Phase change memory. *Proc. IEEE* 98, 12 (2010), 2201–2227.

[219] Shujiang Wu, Song Li, Yinzhi Cao, and Ningfei Wang. 2019. Rendered Private: Making GLSL Execution Uniform to Prevent WebGL-based Browser Fingerprinting. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1645–1660. https://www.usenix.org/conference/usenixsecurity19/presentation/wu

[220] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. 2016. One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation. In *25th {USENIX} security symposium ({USENIX} security 16)*. 19–35.

[221] Guozhu Xin, Jun Han, Tianyu Yin, Yuchao Zhou, Jianwei Yang, Xu Cheng, and Xiaoyang Zeng. 2020. VPQC: A Domain-Specific Vector Processor for Post-Quantum Cryptography Based on RISC-V Architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 8 (2020), 2672–2684. https://doi.org/10.1109/TCSI.2020.2983185

[222] Wenjie Xiong and Jakub Szefer. 2021. Survey of Transient Execution Attacks and Their Mitigations. *ACM Comput. Surv.* 54, 3, Article 54 (may 2021), 36 pages. https://doi.org/10.1145/3442479

[223] Qiumin Xu, Hoda Naghibijouybari, Shibo Wang, Nael Abu-Ghazaleh, and Murali Annavaram. 2019. GPUGuard: Mitigating Contention Based Side and Covert Channel Attacks on GPUs. In *Proceedings of the ACM International Conference on Supercomputing* (Phoenix, Arizona) *(ICS '19)*. ACM, New York, NY, USA, 497–509. https://doi.org/10.1145/3330345.3330389

[224] Mengjia Yan, Yasser Shalabi, and Josep Torrellas. 2016. ReplayConfusion: Detecting Cache-based Covert Channel Attacks Using Record and Replay. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

[225] Mengjia Yan, Read Sprabery, Bhargava Gopireddy, Christopher W. Fletcher, R. Campbell, and J. Torrellas. 2019. Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World. *2019 IEEE Symposium on Security and Privacy (SP)* (2019), 888–904.

[226] Fan Yao, Milos Doroslovacki, and Guru Venkataramani. 2018. Are Coherence Protocol States Vulnerable to Information Leakage?. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*.

[227] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. 2020. DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 1463–1480. https://www.usenix.org/conference/usenixsecurity20/presentation/yao

[228] Zhihao Yao, Zongheng Ma, Ardalan Sani, and Aparna Chandramowlishwaran. 2018. Sugar: Secure GPU Acceleration in Web Browsers. In *Proc. International Conference on Architecture Support for Operating Systems and Programming Languages (ASPLOS)*.

[229] Zhihao Yao, Saeed Mirzamohammadi, Ardalan Amiri Sani, and Mathias Payer. 2018. Milkomeda: Safeguarding the Mobile GPU Interface Using WebGL Security Checks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1455–1469. https://doi.org/10.1145/3243734.3243772

[230] Yuval Yarom and Katrina Falkner. 2014. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 719–732. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom

[231] Mengmei Ye, Xianglong Feng, and Sheng Wei. 2018. HISA: Hardware Isolation-based Secure Architecture for CPU-FPGA Embedded Systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8. https://doi.org/10.1145/3240765.3240814

[232] Miao Yu, Virgil D. Gligor, and Zongwei Zhou. 2015. Trusted Display on Untrusted Commodity Platforms. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 989–1003. https://doi.org/10.1145/2810103.2813719

[233] Bilgiday Yuce, Patrick Schaumont, and Marc Witteman. 2018. Fault attacks on secure embedded software: Threats, design, and evaluation. *Journal of Hardware and Systems Security* 2, 2 (2018), 111–130.

[234] Boris Zbarsky. 2015. Reduce resolution of performance.now. https://hg.mozilla.org/integration/mozilla-inbound/rev/48ae8b5e62ab.

[235] Shaza Zeitouni, Ghada Dessouky, and Ahmad-Reza Sadeghi. 2020. SoK: On the Security Challenges and Risks of Multi-Tenant FPGAs in the Cloud. *CoRR* abs/2009.13914 (2020). arXiv:2009.13914 https://arxiv.org/abs/2009.13914

[236] Mark Zhao and G. Edward Suh. 2018. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. 229–244.

[237] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. 2021. Hermes Attack: Steal DNN Models with Lossless Inference Accuracy. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 1973–1988. https://www.usenix.org/conference/usenixsecurity21/presentation/zhu

[238] Pengfei Zou, Ang Li, Kevin Barker, and Rong Ge. 2019. Fingerprinting Anomalous Computation with RNN for GPU-accelerated HPC Machines. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*. 253–256.