Secure Machine Learning Hardware: Challenges and Progress

Kyungmi Lee, *Member, IEEE*, Maitreyi Ashok, *Student Member, IEEE*, Saurav Maji, *Member, IEEE*, Rashmi Agrawal, *Member, IEEE*, Ajay Joshi, *Senior Member, IEEE*, Mengjia Yan, *Member, IEEE*, Joel S. Emer, *Fellow, IEEE*, and Anantha P. Chandrakasan, *Fellow, IEEE*

Abstract

With the rising adoption of deep neural networks (DNNs) for commercial and high-stakes applications that process sensitive user data and make critical decisions, security concerns are paramount. An adversary can undermine the confidentiality of user input or a DNN model, mislead a DNN to make wrong predictions, or even render a machine learning application unavailable to valid requests. While security vulnerabilities that enable such exploits can exist across multiple levels of the technology stack that supports machine learning applications, the hardware-level vulnerabilities can be particularly problematic. In this article, we provide a comprehensive review of the hardware-level vulnerabilities affecting domain-specific DNN inference accelerators and recent progress in secure hardware design to address these. As domain-specific DNN accelerators have a number of differences compared to general-purpose processors and cryptographic accelerators where the hardware-level vulnerabilities have been thoroughly investigated, there are unique challenges and opportunities for secure machine learning hardware. We first categorize the hardware-level vulnerabilities into three scenarios based on an adversary's capability: 1) an adversary can only attack the off-chip components, such as the off-chip DRAM and the data bus; 2) an adversary can directly

Digital Object Identifier 10.1109/MCAS.2024.3509376
Date of current version: 7 February 2025

attack the on-chip structures in a DNN accelerator; and 3) an adversary can insert hardware trojans during the manufacturing and design process. For each category, we survey recent studies on attacks that pose practical security challenges to DNN accelerators. Then, we present recent advances in the defense solutions for DNN accelerators, addressing those security challenges with circuit-, architecture-, and algorithm-level techniques.

Index Terms—Hardware security, DNN accelerators, side-channel attacks, fault injection attacks, memory security, hardware trojan.

I. Introduction

The past decade has witnessed the remarkable success of deep neural networks (DNNs) in a wide range of applications and benchmarks. Beyond the success in academia and research, DNNs have been adopted for numerous commercial and real-world applications, such as chatbots [1], medical imaging [2], drug discovery [3], autonomous driving [4], and circuit design [5]. However, with this wide usage, there is a growing concern over the *security* of DNNs, especially when they are deployed for high-stakes applications.

Security has three key aspects—confidentiality, integrity, and availability—each of which has direct connections to machine learning applications (Fig. 1). First, *confidentiality* of user-provided input data is crucial to ensure data privacy. For example, in biomedical applications [2], user input data can contain private health information that should not be accessed by unauthorized users and is protected under government regulations [6]. Also, confidentiality is important for the intellectual property of proprietary DNNs. Developing modern DNNs can require significant resources, such as a proprietary data set, high-performance computing systems, and custom training algorithms [7]. Thus, the model architectures and parameters of DNNs can be important assets for the developers, and they can be the target of an adversary who attempts to create a functional copy of the DNN.

Second, *integrity* of both user input data and DNN model parameters is essential for trustworthy AI applications. In order for a user to trust the prediction made by a DNN, there has to be a guarantee that both user-provided data and DNN model parameters are authentic. Recent work showed that the prediction of DNNs can be dramatically changed by the addition of a small amount of noise in either input data [8], [9] or model parameters [10], [11], [12], [13], illustrating the importance of integrity for DNNs. Furthermore, errors that are introduced during the computation of a DNN, either random or adversarial, can also undermine the trustworthiness [14], [15], [16], [17].

Finally, *availability* of machine learning applications can be compromised when an adversary wages denial-of-service attacks. By preventing the accelerator from performing necessary operations, the entire system can be slowed down or completely halted [18], [19]. For

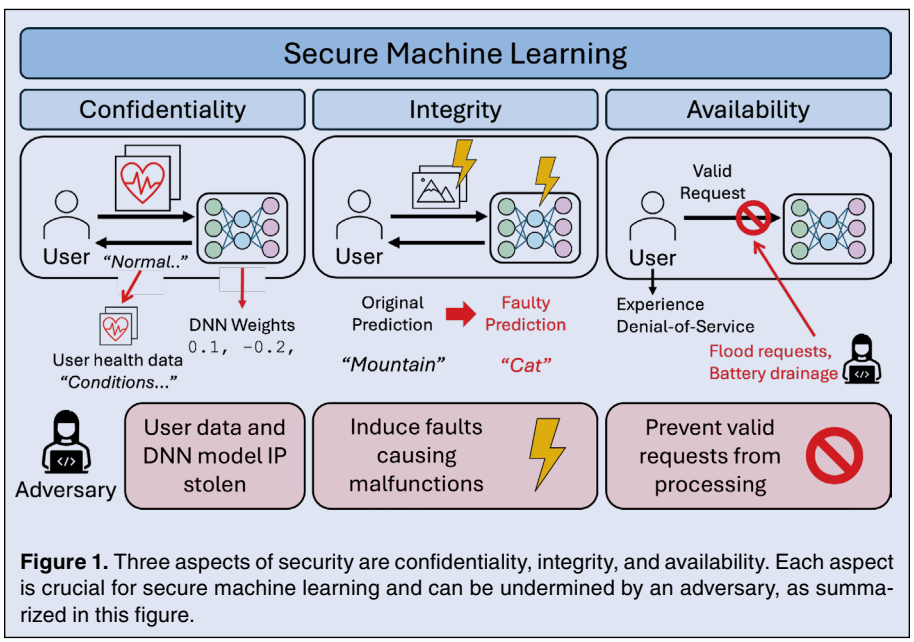


Figure 1. Three aspects of security are confidentiality, integrity, and availability. Each aspect is crucial for secure machine learning and can be undermined by an adversary, as summarized in this figure.

Kyungmi Lee, Maitreyi Ashok, Mengjia Yan, and Anantha P. Chandrakasan are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: kyungmi@mit.edu; maitreyi@mit.edu; mengjiay@mit.edu; anantha@mit.edu).
Saurav Maji was with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with Intel Labs, Hillsboro, OR 97124 USA (e-mail: majisaurav1@gmail.com).
Rashmi Agrawal is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA, and also with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rashmi23@mit.edu).
Ajay Joshi is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA (e-mail: joshi@bu.edu).
Joel S. Emer is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA, and also with Nvidia Corporation, Westford, MA 01886 USA (e-mail: jsemer@mit.edu).

latency-critical applications, such as autonomous driving, this can result in real harm from not being able to make decisions on-demand.

These security concerns pose novel challenges for domain-specific accelerators targeting machine learning applications. Crucially, *hardware-level vulnerabilities* can be exploited to undermine the security required for machine learning applications. Hardware-level vulnerabilities often arise from inherent or intended characteristics of hardware designs, such as data-dependent power consumption [20] or electromagnetic radiation [21], interference [22], [23] and remanence [24], [25] properties of memory cells, and resource sharing across multiple tenants and processes [26], [27], [28]. As they are inherent to hardware designs, not architecture- or implementation-level flaws, these vulnerabilities cannot be simply removed. Furthermore, adversaries can embed hardware trojans in the manufacturing and design process of semiconductor chips [29], [30], [31], [32], [33], [34], [35], [36], [37], creating backdoor channels and causing malicious behavior without the knowledge of the original designer. These vulnerabilities motivate the need for *secure* DNN accelerators equipped with the appropriate defense solutions.

As such, there is a growing interest in securing DNN accelerators from hardware-level vulnerabilities. This article aims to provide a comprehensive overview of recent progress in secure machine learning hardware, specifically focusing on domain-specific accelerators targeting DNN inference. These accelerators service inference requests on user inputs with trained and fixed DNN model parameters, and all three aspects of security are pertinent to this scenario. In this article, we first categorize the source of hardware-level vulnerabilities in DNN accelerators and analyze the unique characteristics of DNN accelerators that affect hardware security (Section II). Then, we survey recent works exploiting hardware-level vulnerabilities to undermine the security of DNN accelerators (Section III). Next, we discuss recent progress in secure DNN accelerator designs, covering diverse circuit-, architecture-, and algorithm-level techniques proposed for the security (Section IV).

II. Understanding the Sources of Vulnerabilities

Hardware systems face various security vulnerabilities. The characteristics of underlying circuits, such as currents fluctuating depending on the data being processed [20], [21] and properties of memory bit cells [23], [38], can leak information or allow manipulation by an adversary. Often, an adversary can insert hardware trojans. These hardware-level security vulnerabilities have been extensively studied for conventional general-purpose microprocessors or cryptographic accelerators.

Emerging domain-specific DNN accelerators [39], [40], [41], [42], [43], [44] share many of the conventional hardware-level vulnerabilities. However, it is important to recognize the characteristics of DNN accelerators that distinguish the attacks targeting them from conventional hardware attacks. DNN accelerators employ high levels of parallelism for the arithmetic operations and typically have regular and predetermined data orchestration across their memory hierarchies [40], [45], [46], [47]. In this section, we first describe the sources of hardware-level vulnerabilities for DNN accelerators from the perspective of conventional hardware security (Section II-A). Next, we discuss the implications of these unique characteristics of DNN accelerators on the hardware-level vulnerabilities and attacks (Section II-B).

While this paper primarily focuses on hardware-level vulnerabilities affecting DNN accelerators, it is crucial to acknowledge vulnerabilities originating from the host processor when DNN accelerators are used as co-processors. A host processor with compromised system software, such as hypervisors or operating systems, can allow an adversary to gain root privileges [48], [49], [50], [51]. Similarly, hardware attacks targeting the host processor can enable privilege escalation [23], [27], [28], [52]. Since the host processor off-loads DNN workloads to accelerators, an adversary with root privileges can potentially access data originating from the host, without further sophisticated attacks targeting the accelerators we describe in this section. Thus, the security of the host processor and the potential for an adversary to gain root access represent another layer of vulnerabilities affecting the overall system. Although our discussion focuses on hardware-level vulnerabilities specific to DNN accelerators, readers are directed to [23], [27], [28], [48], [51], [53], [54], and [55] for a comprehensive overview of host processor security issues and software-level vulnerabilities.

A. Conventional Taxonomy for Hardware-Level Vulnerabilities

We can categorize the hardware-level vulnerabilities according to the scope of what can be trusted in hardware systems (Fig. 2). In the first scenario, an adversary cannot directly observe or manipulate a victim DNN accelerator, thus the on-chip structures in a victim DNN accelerator can be trusted to be secure. However, the off-chip components, such as the main memory and peripherals that are connected to a victim DNN accelerator, can still be vulnerable to an adversary (Section II-A1). Alternatively, there can be a scenario where the on-chip structures are exposed to an adversary, and an

adversary can obtain direct information from a victim DNN accelerator (Section II-A2).

The above two scenarios assumed that a DNN accelerator is manufactured faithfully according to the original design. This assumption fails when an adversary can insert hardware trojans to the design, such that the behavior of a victim DNN accelerator cannot be trusted (Section II-A3).

In the rest of this section, we provide an overview of vulnerabilities for each scenario. These vulnerabilities have been conventionally investigated for general-purpose CPUs/GPUs and cryptographic accelerators. As DNN accelerators share similar security challenges, where appropriate, we will also describe related work on CPUs, GPUs, and cryptographic accelerators that we believe will be applicable to DNN accelerators.

1) Vulnerabilities of the Off-Chip Components: Since DNNs have a large memory footprint for their model parameters and intermediate tensors, an off-chip memory element such as DRAM acts as the main memory for a DNN accelerator by holding the data that cannot fit on the limited on-chip memory [40], [46], [47]. However, this off-chip memory element and a bus connecting it to the DNN accelerator can be targets of an adversary [24], [25], [38]. Several works [25], [38] showed that an adversary can read out data stored in a DRAM by executing an abnormal booting sequence or by physically removing a DRAM and connecting it to an adversary-controlled machine. These attacks are often referred to as "cold-boot" attacks, and they exploit the characteristic of a DRAM that data remains for a certain period of time even after the power is disconnected. These attacks undermine the confidentiality of data stored in off-chip memory.

Off-chip memory is also vulnerable to fault injection attacks that undermine integrity [11], [12], [22], [23]. An adversary with physical access to an off-chip memory can use electromagnetic pulses [56] and lasers [57] to induce bit flips in data stored in a DRAM. Furthermore, bit flips can be remotely induced by an adversary who exploits interference between bit cells in a DRAM, which causes one bit cell's activity to affect its neighboring bit cells [22], [23]. For example, Rowhammer [22] is an attack that induces predictable bit flips in a DRAM by repeatedly accessing the neighboring rows of the victim bit cells, without physical access to a DRAM.

As an off-chip memory is on a separate die and sometimes a separate package from the DNN accelerator, a bus connecting those two can be susceptible to both confidentiality and integrity breaches. An adversary who can probe this bus can monitor and modify the data traffic and the metadata such as addresses and request types (e.g., read or write) for each transaction [58], [59], [60]. Therefore, even when an adversary lacks the capability to directly attack a victim DNN accelerator, the off-chip components can be vulnerable to attacks undermining confidentiality and integrity.

We briefly note that the emerging trend of 2.5D/3D integration for off-chip memory elements can reduce some of the vulnerabilities associated with traditional off-chip DRAMs [61], [62], [63]. 3D integration allows stacking memory elements and processors vertically within the same package. From the security perspective, memory elements residing in the same package reduce the attack surface as bus snooping and tampering attacks are not applicable. As such, some threat models for GPUs that use HBMs assume that the on-package HBMs can be considered trusted, similar to the on-chip structures [61], [64], unlike conventional assumptions on off-chip memory elements.

2) Vulnerabilities of the On-Chip Structures: Data-dependent characteristics of a victim DNN accelerator, such as the power consumption [20], timing information [65], and electromagnetic radiation [21] of an accelerator can act as *side-channels* that leak information. An adversary exploits the fact that these characteristics often have a strong correlation with data and operations of the accelerator. For example, the power consumption of an electronic circuit depends on the activity factor and thus the data being processed, and an adversary who can observe the power consumption of an accelerator can reverse-engineer the data [20]. Similarly, timing information can be exploited if instructions have conditional branches that require different number of cycles to complete an operation (e.g., multiplications, nonlinear functions) depending on the

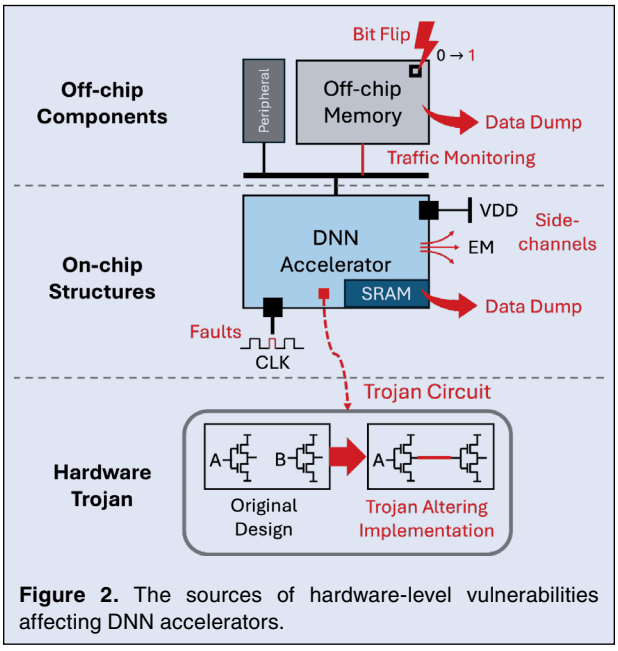


Figure 2. The sources of hardware-level vulnerabilities affecting DNN accelerators.

data [66], [67], [68]. Spatially distributed currents can also induce electromagnetic radiation that can be measured non-invasively [69].

An adversary can learn about confidential data, such as user inputs [70], [71] and model parameters [72], [73], [74], [75], using these side-channel leakages. When fine-grained side-channel leakages, such as those at the single instruction level, are available, an adversary can obtain powerful insights into exact data values being processed. However, such fine-grained information can be challenging to be obtained in some scenarios. For example, many computations are performed in parallel across several arithmetic units in GPUs (also similarly in DNN accelerators), and often an adversary can only observe aggregate features, such as memory allocation sizes and hardware counter values [76], [77]. Still, these coarse-grained leakages can pose significant security risks as we elaborate in Section III-B1 of this article.

Moreover, an adversary can manipulate the power or clock supply to the device to induce faulty operations in a DNN accelerator [15], [16], [17]. Clock glitches can induce skipping of instructions in microprocessors [78] or timing violations in combinatorial logic [15]. Glitches in the power supply can also cause timing violations [16], [17] as the timing requirement of a circuit depends on the supply voltage [16]. High-precision electromagnetic pulses and lasers can cause bit flips in the internal state of an accelerator [56], [79]. As faults induced by these physical manipulation methods alter the result of operations, they can undermine the integrity of DNN accelerators.

Fully invasive attacks can also affect the privacy and integrity of these accelerators. Attacks such as voltage microprobing [80] can allow attackers to find the values stored in specific memory locations or communicated in buses. Furthermore, IC delayering and imaging [81] can give full knowledge of the integrated circuit layout, which has successfully been used to recover the entire design.

Often, these attacks on a DNN accelerator require an adversary to have physical access to an accelerator to measure the side-channel leakages or interfere with the hardware. However, recent work demonstrated these attacks can be waged remotely by leveraging a software interface developed for power monitoring [82], microarchitectural side-channels available from resource sharing [83], or a monitoring circuit that is co-located with a victim DNN accelerator when multiple users share a remote FPGA [16], [17], [84], [85], [86], [87]. Thus, the security of the on-chip structures is relevant for both edge and cloud environments.

Finally, on-chip memory elements, such as SRAMs that act as buffers and scratchpad memory in DNN accelerators, can be also subjected to data readout attacks similar to the off-chip memory. Recent work from [88] showed that an adversary can exploit the power domain separation in a processor (i.e., SRAMs have a different supply voltage from other on-chip components for energy-efficiency) to induce data retention of SRAMs similar to the cold-boot attacks against the off-chip DRAM. Also, another work [89] demonstrated that a secret stored in the on-chip SRAM can be imprinted to the bit cells, causing it to be leaked. While these exploits are only demonstrated for general-purpose microprocessors, the same methodology can be potentially applied to on-chip SRAMs of DNN accelerators.

3) Hardware Trojans: Finally, an adversary can insert hardware trojans, which are circuits that cause malicious effects on the operation of a DNN accelerator without knowledge of the original designer [29], [30], [31]. These hardware trojans can operate as backdoor channels to leak information [90], [91], induce bit flips and rewire logic gates to generate faulty outputs [30], [37], and drain the battery of a victim accelerator to make it unavailable to users [92]. Thus, hardware trojans are powerful attacks that can broadly undermine the confidentiality, integrity, and availability of a DNN accelerator.

An adversary can insert hardware trojans at various stages throughout the design and fabrication process of a DNN accelerator. As the semiconductor supply chain gets more complex, with third-party hardware IP blocks and fabrication processes outsourced to foundries, the risk of hardware trojans is also increasing [93], [94]. While normal functional testing after the fabrication of a semiconductor chip exists, a carefully designed hardware trojan can evade such tests by only rarely triggering its malicious behavior [29], [32], [34].

B. Unique Characteristics of DNN Accelerators

While the conventional taxonomy for hardware security provides an excellent primer to understanding various hardware-level vulnerabilities affecting a DNN accelerator, it is important to recognize the unique characteristics of DNN accelerators and their ramifications on the vulnerabilities. These unique characteristics often make exploiting hardware-level vulnerabilities more challenging to an adversary targeting DNN accelerators.

First, DNN accelerators generally use predetermined and structured control across their memory hierarchy and processing elements, without using conditional branches in their instructions [39], [40], [47]. For example, the timing variation observed for floating-point arithmetic operations and ReLU activation functions (i.e., negative values are clipped to zero) in some microcontrollers [68] is not present in DNN accelerators as

they can be implemented with a dedicated circuit that completes in a fixed number of cycles. As a result, timing side-channel leakages are limited to coarse-grained patterns, such as the total cycles required to process a single layer in a DNN workload [95], and obtaining fine-grained timing information for each arithmetic operation or data access is challenging. Thus, an adversary has a much more limited opportunity for exploiting timing side channels in DNN accelerators.

Second, DNN accelerators have many processing elements that perform arithmetic operations running in parallel. An adversary observes the mixture of power consumption or electromagnetic radiation patterns from multiple processing elements that compute with different input and weight pairs. This large amount of parallelism can be an obstacle for an adversary trying to exploit side-channel leakages, especially when an adversary does not know the exact architecture and scheduling of operations in the accelerator [73], [74].

Third, as DNNs have a large memory footprint, the amount of secrets that an adversary has to recover from DNN accelerators is much larger than the 128 -bit or 256 -bit secret keys in cryptographic accelerators [20], [96]. Combined with the two above-mentioned characteristics that limit exploitable side-channel leakages in DNN accelerators, it is practically challenging to fully recover the fine-grained information of a victim DNN accelerator. Instead, many attacks on DNN accelerators target high-level features such as the model architecture of a victim DNN instead of every single weight parameter [95], [97], [87], [98], and the class information of user input instead of the exact input features [99]. These high-level features still present significant security challenges. DNN model architectures are often core intellectual property for developers, while input class information can be sensitive, particularly in biomedical applications where a DNN classification might involve detecting health conditions or diseases, making the class information itself personal health information that has to be protected.

Fourth, the robustness of DNNs to small random perturbations can increase the difficulty of fault injection attacks. In cryptographic accelerators, even a single bit flip in a secret key has an avalanche effect across all operations using the perturbed key [100], [101]. Often, a bit flip in the most significant bit (MSB) of the exponent field in a floating-point number can also have detrimental impacts on DNNs [12]. However, in general, the performance of a DNN is not significantly affected by small random noise added to its model parameters or input data [8], [10], [102], which is a characteristic often leveraged to design efficient hardware accelerators using approximate [103], [104] or analog computing [105], [106], [107], [108]. Therefore, an adversary must algorithmically identify the worst-case faults for its victim DNN, as random faults can be less effective [10], [13].

Finally, an adversary's prior knowledge about a victim DNN has to be carefully considered. Cryptographic accelerators usually implement a standardized cipher (i.e., AES [109]) whose algorithm is publicly known. Thus, the adversary can compute the expected results given known inputs to perform statistical side-channel analysis [20]. However, DNNs have a wide variety of model architectures [110], [111], [112], [113], numerical precision [114], [115], [116], [117], [118], and sparsity patterns [119], [120], [121], [122], and there is no single "standard" DNN algorithm. Thus, an adversary who has no knowledge about a victim DNN (i.e., "blackbox" attack) has to identify this high-level information before attempting to recover the model parameters [70], [71], [73]. Also, some fault-injection attacks require full knowledge of a victim DNN, including its model parameters (i.e., "whitebox" attack) [10], [13], [123]. Therefore, an adversary's prior knowledge of a victim DNN becomes a key component of a threat model.

III. Recent Advances in Attacks Targeting DNN Accelerators

In this section, we provide a comprehensive survey of recent advances in attack methodologies exploiting hardware-level vulnerabilities of DNN accelerators. For each category of vulnerabilities introduced in Section II-A, we outline how attacks can undermine confidentiality, integrity, and availability aspects of DNN applications. These advances demonstrate that hardware-level vulnerabilities can be a significant threat to DNN accelerators.

A. Attacks on the Off-Chip Memory and Bus

1) Confidentiality Attacks: Cold-boot attacks [124], [125] are a well-known methodology for extracting the data stored in an off-chip memory element. When a DNN accelerator uses an off-chip memory as its main memory, this attack methodology can be applied to steal a victim DNN's model architecture and parameters. A recent study [25] performed cold-boot attacks on a commercially available edge DNN accelerator. The attack targets the RAM on the host processor using low temperatures (utilizing the fact that data remains for a longer period of time when the temperature is low) and the abnormal booting sequence of the host processor. In this attack demonstration, key prior knowledge required for an adversary is the storage format of a victim DNN's model architecture and parameters, which can be publicly available if a DNN accelerator uses an open-source protocol to interface with the host processor [126]. Then,

an adversary can leverage this knowledge to determine bits corresponding to the model architecture and the parameters, and correct bit errors to reconstruct a functional DNN.

Furthermore, the metadata of off-chip memory traffic, such as the requested addresses, types (i.e., read or write), and timestamps, can act as a side-channel that an adversary can use to reverse engineer a victim DNN's model architecture [58], [59], [60]. Recall that DNN accelerators use structured and predetermined control for accessing their off-chip memories. While this characteristic can reduce fine-grained timing side-channel leakages (Section II-B), the off-chip memory access pattern still leaks relevant coarse-grained information about a victim DNN. For example, one layer's output tensor is used as the next layer's input tensor in many DNNs, and identifying the read-after-write dependencies across the off-chip memory access patterns can reveal the intermediate tensor size and the timing "boundary" of when one layer starts and ends [58]. Combined with another property of a DNN that the model parameters are usually read-only during the inference, [58], [60] showed that reverse engineering of a DNN's model architecture is feasible with only the memory traffic patterns.

2) Integrity Attacks: Fault Injections: DNNs are generally robust to small amounts of random noise, with some exceptions on the exponent field of a floating-point number [12]. Thus, fault injection attacks have to be more precise in terms of which bits to target (Section II-B). Recent studies proposed approaches to identify the most harmful bit flips in the model parameters of a DNN [10], [13], [123] and showed that only a handful of bit flips is sufficient to completely degrade the performance of a victim DNN. Reference [10] used the gradient information of each weight parameter to determine the most important bits in a linearly quantized DNN. Reference [10] chose the bits with a large magnitude of gradients and showed that only 10-20 bit flips are required to degrade the performance of a victim DNN to the random guess level. Reference [13] leveraged the characteristics of sparse matrices and the compressed storage formats to attack pruned DNNs with fine-grained sparsity in their weights. In particular, [13] exploited that the compressed storage formats for sparse matrices separately store the nonzero values and their positions in the original matrices, and that a bit flip in the positions leads to rewiring of the connections while not changing the nonzero value directly (Fig. 3). Reference [13] also showed that flipping only a small percentage of the total memory footprint, less than 0.00005%, of the compressed model parameters is sufficient to cause significant performance degradation as illustrated in Fig. 3(b). Note that these algorithms do require an adversary to have full knowledge of a victim DNN, such as the model architecture, parameters, and storage format.

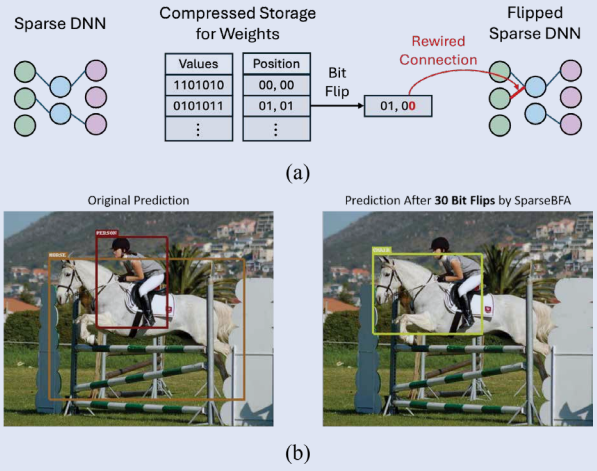


Figure 3. (a) In [13], the position of nonzero values in a sparse matrix is subjected to bit flip attacks, resulting in rewiring of connections in a victim DNN. (b) The original object detection model correctly predicts that there are a human and a horse in an input sample. However, after flipping 30 bits using [13], the model wrongly predicts that there is a chair in this example. This image is reproduced from [13] (© IEEE 2022).

Using these algorithms to identify a few worst-case bit flip targets, an adversary can use fault injection methodologies against the off-chip main memory of a DNN accelerator. Recent works [11], [12] showed that Rowhammer can be practically utilized to induce bit flips in the target bit cells in a DRAM storing the model parameters of a victim DNN. For example, in [11], flipping ~20 bits took ≤ 100 seconds, demonstrating that Rowhammer can be a practical threat for the off-chip DRAM. Finally, the faults in an off-chip memory element can be used to induce targeted misclassification only for certain inputs, making the attack more stealthy [123].

These integrity attacks are particularly challenging for inference accelerators. Inference accelerators typically do not update the parameters of DNNs they are servicing, and corrupted bits in the parameters can impact all subsequent inference requests if there is no mechanism to detect these attacks. In contrast, DNN *training* accelerators that update the parameters can often recover from corrupted bits, and the impact of integrity attacks can be less damaging [127].

B. Attacks on the On-Chip Structures of a DNN Accelerator

1) Confidentiality Attacks: Side-channel Attacks (SCAs): As we discussed in Section II-B, timing side-channels are limited for DNN accelerators that do not have data-dependent

branch instructions. As such, many works on side-channel attacks (SCAs) on DNN accelerators focused on other types of physical side-channel leakages, such as power consumption and electromagnetic radiation from a victim DNN accelerator. After an adversary collects those traces, an adversary uses statistics or pattern recognition algorithms to reverse-engineer secrets from those traces (Fig. 4). For example, a Simple Power Analysis (SPA) passively monitors the magnitude and peaks in the traces to infer high-level secrets like the DNN model architecture [87], [95], [97], [98], [128], [129] [Fig. 4(a)]. Another type of statistical analysis is a Correlation Power Analysis (CPA) [130], [131] [Fig. 4(b)]. For a CPA attack, an adversary algorithmically computes the power model of a victim circuit for several candidate values for the secret. Then, an adversary correlates the actual side-channel traces with an algorithmic model using statistical analysis and chooses the candidate value that has the highest correlation. Compared to a SPA attack, a CPA attack generally requires more information for an adversary, such as the inputs given to a victim circuit to generate the traces.

Recently, some SCAs leveraged powerful pattern recognition capabilities of DNNs to strengthen their attacks [132], [133], [134], [135] [Fig. 4(c)]. For example, [132] used deep learning to predict the key of AES cryptographic implementations with a much smaller attack time compared to traditional attack methods like CPA attacks. Furthermore, this was applicable across multiple devices, where the power SCA variation between devices is much higher than that between different key values on the same device. Similarly, [133] used neural networks to predict the data being converted in an analog to digital converter (ADC) based on power SCA data. Both multilayer perceptrons and convolutional neural networks (CNNs) were highly successful at learning the relationship between time-series spikes in power consumption from capacitors charging and discharging to the output digital bits, with over 99% accuracy for all 12 bits of an unprotected commercial ADC. In [133], CNNs could even reverse-engineer some information from an ADC equipped with protection mechanisms.

These DNN-methods use a profiled form of attack [136], where training data is required with the SCA traces corresponding to a particular known computation rather than just a simplified power model as in conventional SPA and CPA attacks. On the other hand, much less data is required during the actual attack [137], which can even be done in real-time without having active control over an input to repeat the operation enough times for successful statistical analysis as in CPA attacks [130], [131]. This tradeoff is valuable for an attacker, who has the ability to buy a device that is nominally identical to the target through regular commercial channels, and train a model that can apply with high accuracy even with process variations.

While physical SCAs have been well studied in cryptographic circuits [138], [139], [140], [141], their presence in DNN accelerators [142], [143], [144] has only been considered in the last decade. In this subsection, we introduce recent advances in physical SCAs against DNN accelerators, which aim to reverse-engineer user input data, DNN model parameters, and architectures.

Recovering User Input Data Recent studies [70], [71] demonstrated the use of physical SCAs to recover user input data for simple image processing workloads such as MNIST [145]. Such attacks can be particularly relevant in machine learning systems where an on-chip sensor and ADC generates the inputs for the accelerator, rather than being transmitted from off-chip. They leverage the first layer of a DNN that directly operates on user input data and note any significant changes of the power consumption during the processing of this first layer. These changes correlate to large differences between nearby pixels in a user input, which reveals the silhouette of the image. Note that an adversary requires the knowledge of a victim DNN's model architecture

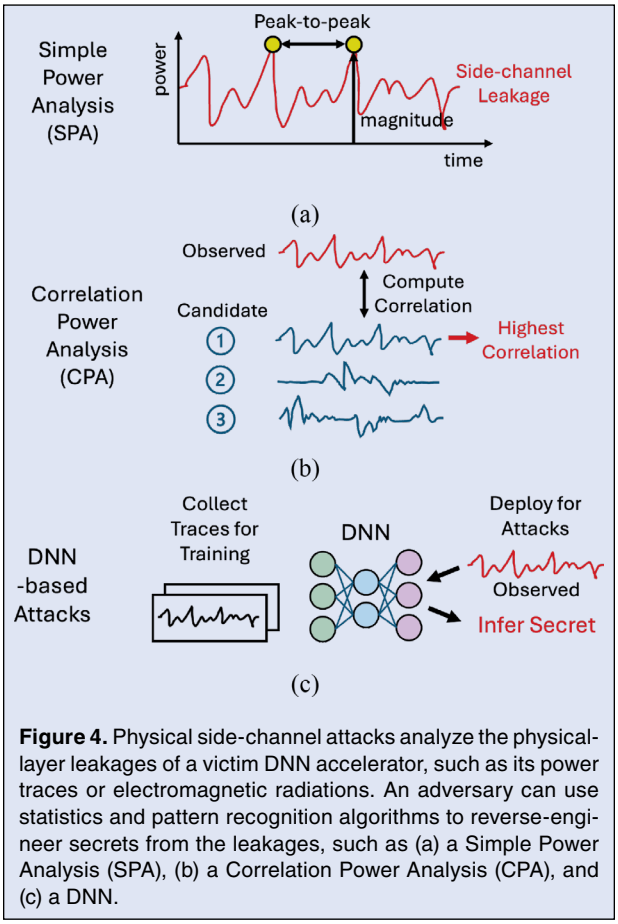


Figure 4. Physical side-channel attacks analyze the physical-layer leakages of a victim DNN accelerator, such as its power traces or electromagnetic radiations. An adversary can use statistics and pattern recognition algorithms to reverse-engineer secrets from the leakages, such as (a) a Simple Power Analysis (SPA), (b) a Correlation Power Analysis (CPA), and (c) a DNN.

to determine when an accelerator processes the first layer, but not the model parameters. This attack is currently limited to relatively simple user input data, such as MNIST or medical images that have a simple image on top of a plain background [70]. However, the success rate of this attack has been shown to be higher if an adversary can actively collect power traces from a DNN accelerator with known input data to build a template, which is then used to reconstruct secret user input data [70]. Furthermore, similar attacks can be applied to other layers of a DNN. While the intermediate activations might not directly leak the user input, they can be leveraged to infer the DNN model parameters.

Recovering Model Parameters The next type of attack attempts to recover the DNN model parameters [72], [73], [74], [75]. The threat model for these attacks assume that a victim DNN's model architecture is known to an adversary, and an adversary can wage CPA attacks with the known input data to a victim DNN accelerator.

In [73], a proof-of-concept CPA attack on a DNN accelerator using a spatial architecture using an array of processing elements [44] was demonstrated. This work computed the correlation after observing a chain of sequential computations in one processing element within the array, which improves the attack accuracy. However, this attack was demonstrated on a relatively simple accelerator architecture compared to commercial accelerators. Perhaps most importantly, [74] is the first to consider realistic hardware architectures, by attempting to perform SCA on a commercially available edge DNN accelerator, whose accelerator architecture is unknown. While this attack is not entirely successful even on very basic neural network structures, it reveals a real threat that could be further exploited with more complex analysis methods. In summary, many of these demonstrated attacks have difficulties in recovering inputs and model parameters with high accuracy without very limiting assumptions.

Recovering Model Architecture Instead, many recent works focused solely on recovering the DNN model architecture [87], [95], [97], [98]. Unlike CPA attacks described above, these attacks utilize SPA attacks that only require passive SCA measurements without any control over inputs and weights, and can even be performed remotely [87], [98] or on larger scale commercial accelerators [95], [98].

For example, [97] used electromagnetic radiation measurements to recover the number of layers and the number of parameters in each layer in a victim DNN. More recent works [87], [95], [98] trained machine learning models that infer a victim DNN's model architecture, including the number of layers and tensor shapes in each layer, from the side-channel measurements. While these attacks do not have 100% accuracy, they can be combined with prior knowledge of typical architecture hyperparameters (e.g., for image processing applications, convolutional neural networks with 2-dimensional kernels are typically used) and some fine tuning [87], [97] to achieve similar accuracy to the original network. Finally, we note that these attacks on recovering the model architecture can be combined with the above-mentioned attacks on recovering the model parameters to reconstruct the entire DNN model without any prior knowledge [146].

In-memory computing (IMC) So far, we discussed physical SCAs affecting DNN accelerators with conventional architectures, which have computing processing elements that interact with each other and a separate central bank of memory. However, DNN accelerators using the emerging in-memory computing (IMC) technique [147], [148], also referred to as compute in memory (CIM) have been proposed. These have unique sources of SCA vulnerabilities [149], [150], [151], as discussed in this section.

First, IMC accelerators depend on intrinsic analog properties of SRAM or non-volatile memory (NVM) bit cells to perform multiply-and-accumulate operations. This is in contrast to near memory compute accelerators, where the memory is placed close to the compute through methods such as 3D stacking [152]. While the computation itself is in the analog domain, IMC accelerators can also have side-channel leakages from the additional required peripheral analog-to-digital converter (ADC) components [133].

Second, unlike logic gates, the memory bit cells in IMC accelerators have a regular structure, which makes the electromagnetic side-channel leakages easier to obtain [153]. Also, this regular structure can easily reveal which regions of the memory are operating for a specific DNN layer, and prior work showed that the model architecture can be reverse-engineered using laser imaging of an IMC accelerator [154], [155].

Finally, the physical and circuit properties specific to the memory cells can be exploited to recover the data values. For example, the difference in currents when reading and writing 0 s and 1 s to the memory bit cells can be significantly large for some NVM types [156], [157], [158], allowing an adversary to easily distinguish the data values. Another example is memristor devices, where leakage currents of cells depend on the nearby stored data values [159].

2) Confidentiality Attacks: Invasive Readout: Aside from observing side-channel leakages, invasive attacks that directly tamper with the packaged devices can cause harm to data confidentiality. Voltage micro-probing can be used to directly tap memory cells or buses on the top metal layers of the IC after decapsulating the

circuit [80], [160]. This is particularly harmful for non-volatile memory based accelerators which will hold the model parameters even after powering off for the packaging removal process [161]. Furthermore, the entire design of an accelerator can be reconstructed by an attacker to either gain knowledge of the architecture or to produce counterfeit products. This is done through successive delayering and imaging of the IC, after which image processing tools reconstruct the netlist from GDS information [81], [162], [163], [164].

3) Integrity Attacks: Fault Injections: Fault injection attacks on DNN accelerators can cause malfunctioning of a victim DNN [15], [16], [17]. For example, [15] used clock glitches to induce timing violation in the logic of a DNN accelerator implemented with an FPGA. Even without prior knowledge of a victim DNN model, they showed that a small amount of glitches (e.g., affecting 0.1% of clock cycles) can significantly degrade the accuracy. Other works [16], [17] further demonstrated the possibility of remote power glitch attacks for a cloud FPGA that hosts multiple users. When an adversary can co-locate the malicious circuit with a victim DNN accelerator, remote power glitches can induce either random faults or duplication faults, both resulting in erroneous computation results [16], [17]. Other modalities such as laser fault injection can also be used for targeted attacks on portions of the circuitry [165].

Another potential attack model leverages physical fault injection attacks to reverse-engineer DNN model parameters [166]. When an adversary has prior knowledge of a victim DNN's model architecture and can observe the computation results of an accelerator, [168] showed that an adversary can compare the computation result using the original input and the fault injected data to obtain model parameters. Therefore, while integrity is the primary concern for fault injection attacks, confidentiality can also be undermined.

C. Inserting Hardware Trojans

Hardware trojans are circuits added to cause malicious effects on the operation without the knowledge of the original designer and can be inserted in DNN accelerators to target various operations (Fig. 5). They generally contain a trigger, which waits for some rare event to start the harmful operation while escaping detection during normal functional testing. In addition, the payload can have a variety of harmful effects like leaking information, draining the battery, or rewiring logic gates. As opposed to neural trojans that are implemented at a software level on the trained model, here we survey trojans specific to the hardware implementation.

1) *Harmful Payloads Affecting Confidentiality, Integrity, and Availability:* Many attacks [29], [30], [31] implement targeted changes to the weight parameter value at the circuit level similar to the bit flip attacks from RowHammer [23]. Others have similar effects to changing weights but instead target the computation logic circuitry or the input/output of the block [32], [33], [34], [35]. All of these can severely affect the integrity of the accelerator, especially if the model is known and can be used to create strong adversarial attacks. Other targets of the trojan that can cause integrity or availability issues include intentional glitches in the clock [36] to cause timing violations or reconfiguring the inter-processing element connections of the network-on-chip [37] to change the network structure.

Furthermore, trojan payloads can be targeted to alternate architectures such as IMC, where the analog properties can be utilized to cause failures. For example, internal voltages can be modified to break the sense amplifier margin, or transient bounces on the ground or power rail can cause read and write failures [167], [168]. In other cases, the analog IMC structure can be exploited to affect confidentiality, by directly relating the power consumption of large blocks such as the ADC to secret weight parameter information [169]. In addition, these payloads can target peripheral portions of the accelerator such as the memory controller, where the layer output activations being written back to off-chip DRAM can be modified [170].

2) Triggers: For all these payloads, there needs to be a method to determine when the trojan should start its operation. While some of the trojans are always on [31] and [33], this is rare since the RTL IP and post-fabrication functionality tests are likely to see the effect of any such attack. These specific works get past this issue by having a limited payload that only targets certain input images. Otherwise, many works focus on a trigger from specific

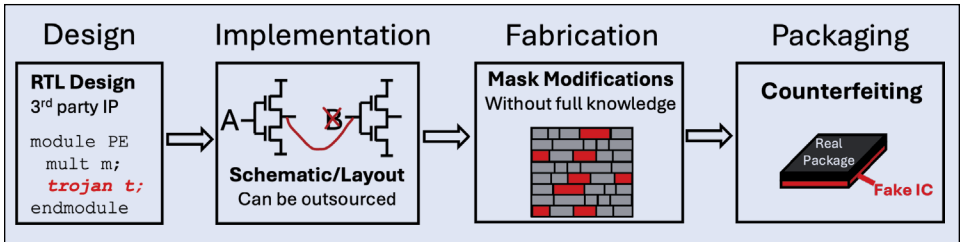


Figure 5. Steps of DNN accelerator production where hardware trojan insertion can occur.

neural network inputs, since we assume the attacker has some control over this. Thus, they utilize a specific rare data value or input image that is unlikely to occur during standard testing and can be accurately detected but still does not seem obviously malicious [29], [30], [36], [37], [167], [170]. This undetectability during testing can be strengthened by relying on a sequence of such rare inputs [32]. At the same time, other works generate a trojan trigger based on accelerator control signals. For example, while typical accelerators will cycle between different addresses and have a standardized memory access pattern based on the chosen dataflow, some triggers can depend on the same address being accessed many times [34], [168]. While digital accelerators would need to detect this using dedicated circuitry in the timing control logic, analog IMC adds further motivation for such attacks. Since RRAM resistances drift with many reads, the effect of repeated accesses can be directly read out as a change in delay or voltage of the readout [168].

IV. Recent Advances in Designing Secure DNN Accelerators

In this section, we present recent progress in defending DNN accelerators from various hardware-level exploits discussed in Section III. We present protections that provide security for the off-chip memory (Section IV-A), counter physical side-channel attacks (Section IV-B) and detect fault injections (Section IV-C) on the on-chip structures, and detect hardware trojans (Section IV-E). In addition, we present defense solutions for emerging in-memory computing accelerators (Section IV-D), which have their unique challenges compared to conventional accelerator architectures. Finally, we present a brief description of Fully Homomorphic Encryption (FHE) technology (Section IV-F), which ensures the privacy of both model parameters and input data by performing computations on encrypted data, without having to trust any hardware components.

A. Providing Security for the Off-Chip Memory

1) Authenticated Encryption for the Off-Chip Memory: One approach to defending an off-chip memory element from data readout attacks (e.g., cold-boot attack [124], [125] discussed in Section III-A1) and fault injection attacks (Section III-A2) is to *encrypt and authenticate* all data stored in an off-chip memory element using cryptographic primitives (Fig. 6). This approach is adopted for memory security in a trusted execution environment (TEE) for general-purpose CPUs [55], [171], [172], [173], [174], [175], [176], [177] and GPUs [61], which provides a secure "enclave" in the memory. Encrypting the off-chip data using strong cryptographic primitives, such as AES [109], prevents an adversary from recovering the actual plaintext data even if data readout attacks successfully extract the data. Furthermore, authenticating all data transfers with cryptographic hashes enables integrity verification to detect any bit flips in the off-chip data, since the computed hashes for perturbed data will not match the original values [171]. Adding the "timestamp" information (also known as "counters"), such as the number of updates performed on a certain address, when generating cryptographic hashes also prevents an adversary from 'replaying' the old data and hashes [171]. Note that the security of this authenticated encryption approach depends on the confidentiality of a secret key used for cryptographic operations, which can be stored on a special register on-chip.

Recent studies investigated adapting TEEs for DNN accelerators [178], [179], [180], [181], [182]. The structured and predetermined data access patterns of DNN accelerators are leveraged to reduce the overhead of

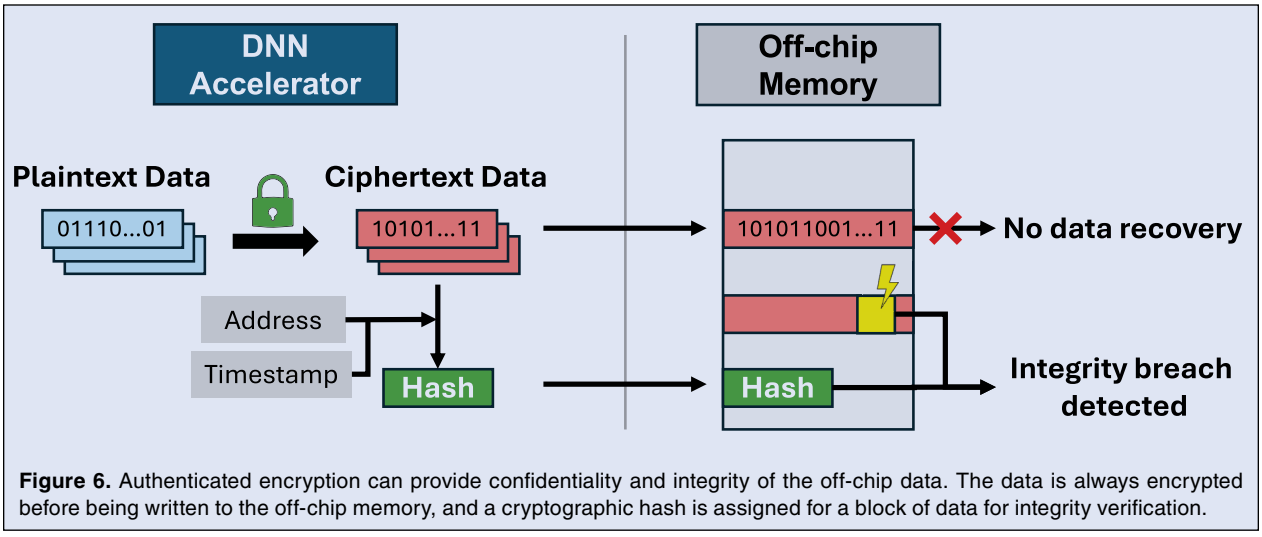


Figure 6. Authenticated encryption can provide confidentiality and integrity of the off-chip data. The data is always encrypted before being written to the off-chip memory, and a cryptographic hash is assigned for a block of data for integrity verification.

supporting a TEE in these works [178], [180]. A key observation was that managing the timestamps, which used to be a major source of performance overhead in CPUs, can be simplified with this structured data access pattern. As a result, the major overhead of this approach reduces to cryptographic encryption and authentication operations required for all off-chip data. In addition, recent work from [182] proposed a software-defined approach to tailor a TEE for diverse DNN accelerator deployment conditions.

For authentication, a cryptographic hash is computed for a block of data and all data in this authentication block is needed for integrity verification. This can lead to additional off-chip data traffic when we only need a portion of data in one authentication block. In DNN accelerators, this property introduces a challenge when authentication blocks misalign with 'tiles' of data tensors, where a tile is a basic granularity of data movement in DNN accelerators chosen to optimize the data reuse and performance. All data in authentication blocks need to be fetched although they do not belong to a tile a DNN accelerator requested, causing performance degradation due to the additional off -chip data traffic and cryptographic operations.

Reference [181] pointed out that cross-layer dependency in a DNN (i.e., one layer's output is the next layer's input) complicates the optimal authentication block assignment that minimizes the additional off-chip traffic. Conventionally, tile assignment is typically done independently for each layer. Consequently, the same tensor data can have different tile assignments when it is used as an output tensor of one layer [e.g., 1×3 tiles in Fig. 7(a)] and an input tensor of the next layer [e.g., 2×2 tiles in Fig. 7(a)]. Suppose we use each tile in an output tensor as an authentication block when we compute the first layer. Then, when the accelerator requests one tile in an input tensor to be fetched for processing the next layer, two output tensor tiles need to be fetched due to authentication, incurring a large amount of redundant reads [Fig. 7(b)].

To overcome this challenge, [181] proposed an algorithm that determines the optimal authentication block and tile assignment for DNN accelerators. In order to identify the optimal authentication block assignment, [181] applied an exhaustive search and analytically computed the amount of the additional off-chip traffic for each configuration. Reference [181] showed that even for the same DNN accelerator architecture, the performance can be improved by up to 33% with the optimal assignment. In summary, several works [178], [179], [180], [181] proposed techniques to reduce the overhead of encrypting and authenticating all off-chip data, which provides a strong hardware-level security guarantee.

Comparison with GPU TEEs TEEs for GPUs and DNN accelerators share some key similarities in their security requirements [61], [64]. Both must establish secure communication channels with their host processors, even when the host processor's system software can be compromised. Also, they need to protect against bus snooping and tampering when they are connected to the host processor using PCIe.

There can be differences regarding the assumptions of off-chip memory security. GPUs often integrate off-chip memory elements within the same package using vertical integration, and some work on TEEs for GPUs consider this on-package memory to be trusted [61]. However, this assumption cannot be shared among all GPUs, when many consumer-grade GPUs opt for conventional DRAMs on separate packages. As such, other work on GPU TEEs [183] considered hardware-level off-chip memory protection, similar to CPUs and DNN accelerators. Although DNN accelerators can benefit from similar vertical integration in the future, especially for high-performance datacenters [44], DNN accelerators also choose to use separate off-package DRAMs and

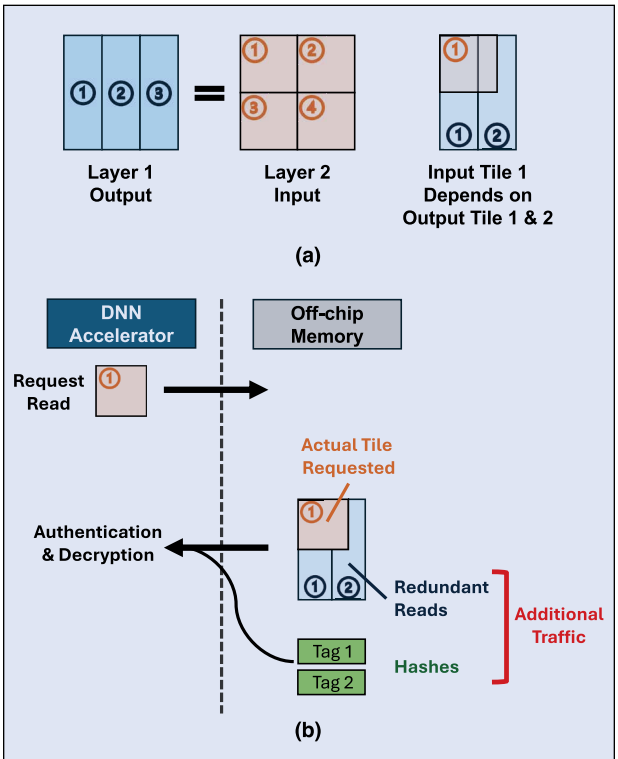


Figure 7. Cross-layer dependency in a DNN complicates the authentication block assignment. (a) An intermediate activation tensor can have different tiling strategies when it is generated as an output tensor of one layer versus when it is used as an input tensor of the next layer. (b) Redundant reads are incurred when each output tile is assigned as an authentication block. Combined with hashes that correspond to authentication blocks, there can be significant additional traffic due to cross-layer dependency.

low-power DRAMs, which require TEEs to provide off-chip memory security.

Finally, DNN accelerators are still evolving, with recent active research into multi-tenancy and virtualization [182], [184], and TEEs for DNN accelerators may need to address new challenges for memory management in the future. For example, TEEs for GPUs require complex virtual-physical address translation to ensure memory allocated for a secure process is isolated [61]. While current DNN accelerators use simple memory control based on regular patterns, future accelerator designs might have to adopt more complex memory management under TEEs.

2) Alternative Approaches to Memory Encryption: Instead of encrypting and authenticating all off-chip data, alternative approaches have been proposed to selectively encrypt a few important weights [185], [186], [187]. While the rest of the model parameters and intermediate tensors that are not encrypted are vulnerable to an adversary, a few encrypted values prevent an adversary from reconstructing a functional DNN with comparable performance to the original victim DNN.

Another approach is to shuffle elements in weight tensors of a DNN and use shuffling information as a secret to provide the confidentiality of DNN model parameters [188], [189], [190]. To reduce the overhead of decrypting the shuffled data, shuffling can be performed coarsely by swapping rows and columns in a matrix [189], [190] or selectively applied to important layers [188]. These approaches benefit applications requiring the confidentiality of DNN model parameters with low-cost protection, although it does not provide the confidentiality of user input data or protection against fault injection attacks.

Finally, [191] proposed a low overhead authentication technique using algorithmic/parity checksums [192], [193], [194]. Compared to cryptographic hashes, algorithmic/parity checksums can have low computational and storage overhead. For example, [192] set the sum of weight parameters in a row as a checksum, without using more sophisticated cryptographic operations. Reference [193] incorporated parity-based checksums where the least significant bit of each model parameter acts as a parity bit. This approach eliminates the storage overhead for hashes, while its impact on the performance of a DNN is negligible. Similarly, [194] used a 2-bit algorithmic checksum for each group of model parameters. Conventionally, algorithmic and parity checksums [192], [193], [195] can be susceptible to an adversary who can flip multiple bits or flip the checksum bits. Reference [191] addressed this susceptibility by encrypting the model parameters along with a parity-based checksum [193], reducing the probability of an adversary successfully counter-feiting checksums. Note that the overhead of these checksums also depends on the precision used for DNN inference. For example, the overhead of a single- or 2-bit checksum can potentially become more expensive for lower precision weight parameters and inputs.

3) Hiding Memory Traffic Patterns: Recall that an adversary can reverse-engineer a victim DNN's model architecture, such as the number of layers and tensor shapes in each layer, by observing the metadata like addresses and request types of the off-chip traffic [58], [59], [60] (Section III-A1). Since the metadata is still visible to an adversary even when all off-chip data is encrypted, different protection mechanisms are needed for this attack. In general-purpose processors, Oblivious RAM [195], [196] was proposed to hide the data access pattern of a program. While this technique provides a theoretical guarantee that the resulting data access pattern is independent of the true pattern, it incurs a large overhead in the off-chip access latency and bandwidth requirement [195], [196].

In DNN accelerators, recent works [189], [197], [198] instead proposed more lightweight defenses that aim to obfuscate the read-after-write dependency of the intermediate tensors and the total number of data accesses that reveal the tensor sizes, instead of completely anonymizing the data access pattern as in Oblivious RAM. For example, [197] shuffles all data accesses around layer boundaries (i.e., when a DNN accelerator completes processing one layer and moves on to the next layer) so that an adversary cannot observe the actual start and end of each layer. Reference [189] breaks a layer in a DNN into multiple smaller sub-layers such that an adversary is misled to observe a wrong number of layers in a DNN. Reference [189] also proposes a mechanism to reduce or increase the total number of accesses, by either partially caching the intermediate tensors (i.e., the cached data does not have to be written and read back from the off-chip memory) or issuing dummy requests.

Another approach is entirely algorithm dependent, where an obfuscated DNN that preserves the functionality of the original DNN but with a different model architecture is designed [199]. Reference [199] used a careful search algorithm to ensure that the obfuscated DNN architecture has a small hardware performance overhead (e.g., the total number of computations). Overall, these works [189], [197], [198], [199] showed that an adversary is misled to the wrong DNN model architecture and ends up with a reconstructed DNN with worse performance than the original victim DNN.

4) Limitations: There are many potential research topics related to the memory security of DNN accelerators. First, the performance of a DNN accelerator using memory encryption and authentication (Section IV-A1) can be throttled by cryptographic operations required for all

data traffic. While [178], [179], [180], [181] proposed several techniques to minimize the overhead and optimize the trade-off considering cryptographic operations, scaling this approach to future memory technologies offering a substantially higher bandwidth can be challenging. For example, having multiple cryptographic accelerators running in parallel to match a high off-chip memory bandwidth can prevent the performance slowdown at a significant cost for area and energy. As DNNs can be memory-intensive and memory technologies are rapidly improving, scaling the defense solution is an important challenge.

Second, many approaches with non-cryptographic defenses often lack the theoretical guarantee of security. Although they have a low hardware overhead for defenses, such a benefit should be understood in the context of a trade-off with the security level. While the empirical results can provide a practical demonstration of security [192], [193], [194], [200], [201], [202], a well-principled analysis on those approaches will be useful for future research.

B. Physical-Layer Side-Channel Attack (SCA) Protections

As physical SCAs broadly affect the confidentiality of DNN accelerators (Section III-B1), there is a growing interest in defending DNN accelerators from physical SCAs. The core idea for the defense is to decorrelate the circuit currents from the data so that side-channel leakages (e.g., power consumption, electromagnetic radiation) do not contain any information about the data being processed. There are two proposed approaches to achieve this decorrelation, masking [75], [203], [204], [205], [206], [207] and shuffling [207], [208].

1) Masking: Masking splits up data into multiple shares, each of which is independently unrelated to the original value [209], [210], [211]. Here we denote the original value as $\bar{x}$, and n independent shares as x^1, x^2, …, x^n. In order to recover the original value, all n shares are needed, and an adversary who only knows $\leq n-1$ shares cannot recover the true value. Computing on these shares is performed in such a way that every output value generated during the computation is independent of at least one input share, so the power consumption for each function is unrelated to the original data $\bar{x}$.

There are two sharing techniques investigated in recent works. First, arithmetic masking [212] uses modular summation to split up each data value, where K is a modulus:

$$x^1 + x^2 + \ldots + x^n \bmod K \equiv \bar{x} \tag{1}$$

Note that simple linear arithmetic operations like additions can be performed over each share separately without having to combine them in intermediate steps.

Alternatively, Boolean masking [209], [210], [213] uses bitwise exclusive OR (XOR) operations:

$$x^1 \oplus x^2 \ldots \oplus x^n = \bar{x} \tag{2}$$

Unlike arithmetic sharing, Boolean masking is more geared towards operations requiring bit-wise manipulations, such as comparing sign bits. However, Boolean masking can result in more complicated implementations for arithmetic operations [206], [207], since each bit-level logic gate has to be modified.

Recent work adopted Boolean sharing or chose to use a combination of techniques for different functions (e.g., arithmetic masking for multiply-and-accumulate operations but Boolean masking for sign bit comparison) [203], [205]. While the latter can be more optimized for each function, it requires secure conversion between the two types of sharing and incurs additional cost [214], [215].

Recently, the first SCA defense for a binary neural network accelerator was proposed, which does not require multiplications and only needs to support additions [75]. They used arithmetic masking to split input tensors into two independent shares, which are passed to two adder trees separately to compute the results for each share. However, recall that arithmetic sharing involves modular operations (Eq. (1)), while additions required for DNN computations are standard arithmetic. This difference results in the leakage of the sign bit in [75]'s approach, and [75] augmented arithmetic sharing with a circuit-level technique that equalizes the power consumption regardless of the data for sign bit computations. Furthermore, the ReLU activation function that clips negative values to zero requires comparing the sign bits of data, which can be better supported using Boolean masking. Thus, [75] alternates between arithmetic masking for additions and Boolean masking for the ReLU function, with an additional overhead for secure conversion.

More recent work [203], [204], [205] tried to push the limitations of [75]. References [203] and [204] modified a DNN algorithm to use modular arithmetic for computations such that arithmetic sharing can be directly applied without the leakage issue, but they require careful selection of the modulus K and significant algorithm modifications. Reference [205] further extended the arithmetic masking solutions to multi-bit fixed point precision, thus improving the practicality of SCA defenses.

Other recent studies instead adopted Boolean masking for all computations, including multiply-and-accumulate operations, in DNN accelerators [206], [207], [216]. However, as we discussed earlier, Boolean masking requires all arithmetic units to be modified due to

bit-level sharing and incurs a large performance, energy, and area overhead. For example, Boolean masking increases the energy consumption of a DNN accelerator by 37.9 x when Threshold Implementation (TI), a type of Boolean masking that split data into three independent shares to guarantee a high-level of security [210], is used for the multiply-and-accumulate circuitry [207] (Fig. 8). So far, these works target a binarized neural network that does not require multipliers [206] or limit the model parameters of a DNN to be powers-of-2 (e.g., 2^0, 2^1, …) [207]. In particular, the power-of-2 approach of [207] reduces a multiply-and-accumulate operation to a shift and XOR operation, which is linear at the bit-level and easy to implement with Boolean masking. Furthermore, limiting the model parameters to powers-of-2 has negligible impact on the classification accuracy of DNN models for applications such as ECG arrhythmia detection or epileptic seizure recognition.

Nevertheless, implementing a multi-bit accumulator with TI introduces several challenges. The TI requirements stipulate that the Boolean shares of data should be uniformly distributed and mutually independent of each other [217], [218], [219]. However, the intermediate computation results of the shift and accumulate operations can violate this. Reference [207] carefully designed its accumulator circuit to not combine shares that are not jointly uniform and to add fresh randomness when combining dependent shares is unavoidable. Random bits are supplied using a lightweight cipher that is also implemented with the TI methodology. Overall, these techniques reduced the overhead of Boolean masking to only 64% more area and a 5.5x increase in the energy consumption for a DNN accelerator supporting multi-bit precision in [207].

2) Shuffling: Another approach to decorrelate side-channel measurements from the actual data is to *shuffle* the computations temporally and spatially [208]. A random sequence generator determines the temporal order of the multiplications in a DNN accelerator, and an adversary who does not know this random sequence cannot recover the true order of the data. Also, the spatial allocation of multiplications to each processing element in a DNN accelerator is randomly determined, such that an adversary utilizing high-resolution electromagnetic side-channel observations can be deterred as well. In addition to this randomization, [208] introduces a compensation mechanism to achieve overall constant power consumption, using a regression model that estimates the dynamic and static power of the processing elements. Similarly, [207] also takes advantage of temporal shuffling for input security when activations are processed by a DSP unit before MAC computations.

3) Limitations: While this set of works shows a promising start to securing DNN accelerators from physical SCA, there remains significant necessary progress. Many of the accelerator defenses incur significant overheads, at around 2 x or higher area and power for iso-latency [203], [206], [208]. The masking and shuffling schemes also depend on a constant supply of random bits, necessitating an on-chip pseudo-random number generator. While works like [205] and [207] start to address the reduction of this requirement through careful design and reuse of randomness, further exploration is necessary into the security trade-offs.

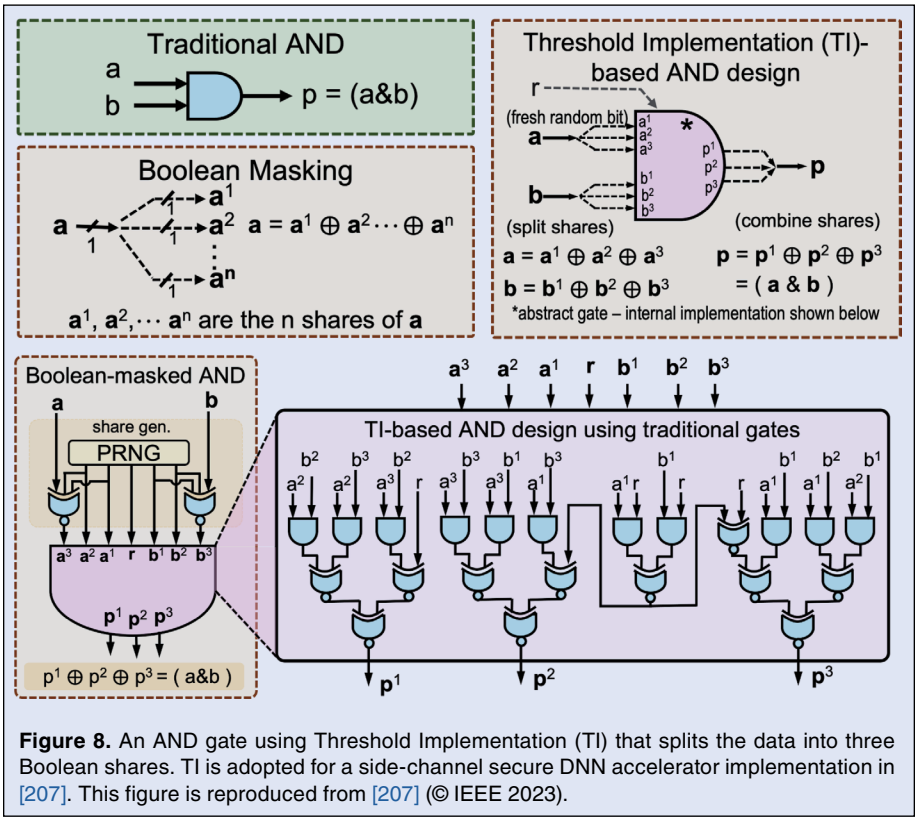


Figure 8. An AND gate using Threshold Implementation (TI) that splits the data into three Boolean shares. TI is adopted for a side-channel secure DNN accelerator implementation in [207]. This figure is reproduced from [207] (© IEEE 2023).

Furthermore, as with the side-channel attack research, the prior work mostly focuses on smaller architectures and models, particularly those implemented on an FPGA. We can not expect the overheads and optimizations to directly scale to more practical and larger designs with floating point precision. Particularly, some requirements such as constant conversion between the arithmetic and Boolean masking domains may become prohibitively expensive with many DNN layers of larger sizes.

C. On-Chip Fault Injection Detection

When an adversary attacks the power and clock supply to an accelerator chip, all modules in this chip sharing the same power and clock supply will be affected. Similarly, electromagnetic pulses with low spatial resolution will affect many modules in a chip. Using this property, [191] used an on-chip cryptographic engine performing encryptions and decryptions as a sensor to detect clock glitches and electromagnetic faults. They observed that cryptographic ciphers are highly sensitive to faults [220], and any glitch that affects a DNN accelerator will most likely also affect an on-chip cryptographic engine.

A lightweight Craft cipher [221] was used to sequentially perform encryption and decryption over an internal state, such that the internal state looped back to the original value after a certain number of encryption-decryption cycles (Fig. 9). If the internal state does not match the original value, it is likely to indicate an anomaly due to faults and processing can be halted. Reference [191] demonstrated that their detection method only incurs an area overhead of 5.9% and successfully identifies all 3×10^6 clock glitches and 0.1×10^6 electromagnetic glitches during testing.

1) Limitations: While [191] proposed an effective detection method for physical fault injection attacks that affect multiple modules in an accelerator, it cannot detect faults with higher spatial resolution that precisely target a victim component without affecting a detection circuit. Different detection methods have been studied in the context of cryptographic accelerators [222], [223], [224], [225], which can be further explored for DNN accelerators. In addition to detecting faults, correcting errors due to fault injection attacks can be necessary for applications requiring real-time processing without interruptions.

D. Defense Solutions for In-Memory Compute (IMC)

1) Side-Channel Defenses for IMC: IMC accelerators often have thousands or millions of parallel multiply-and-accumulate operations, and the overhead of securing these systems can be significant. Unfortunately, there has not been as much research in securing IMC accelerators from physical SCAs, and these are mostly limited to qualitative descriptions of potential defenses when novel attacks were proposed [150], [156]. For example, [150] suggests using dummy memristors and data to hide the DNN model architecture during operation at the expense of significant area, power, and latency.

Reference [226] demonstrated the first thorough defense for digital IMC accelerators using Boolean sharing. Recall that Boolean sharing can be efficiently implemented for operations that are bit-wise linear, such as XOR and XNOR. Reference [226] leveraged linear XNOR gates for multiplication instead of the conventional AND gates to reduce the overhead of Boolean sharing. Additionally, for the adder logic, [226] observed that full adders have better uniformity of outputs compared to half adders. Since uniformity is a key requirement of Boolean sharing and fresh randomness is required if it is not satisfied, utilizing full adders instead of half adders can improve efficiency and security. To this end, [226] used carrysave addition in the adder trees and bit-serial accumulators to eliminate half adder logic.

2) Data Remanence in IMC: IMC accelerators have a unique problem with data readout attacks when they use NVM cells that retain the data even after the system

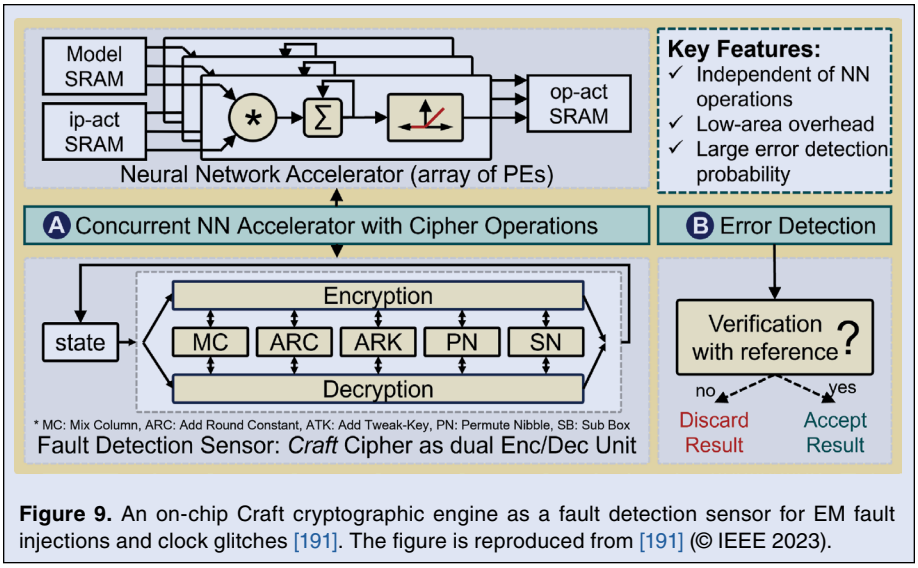


Figure 9. An on-chip Craft cryptographic engine as a fault detection sensor for EM fault injections and clock glitches [191]. The figure is reproduced from [191] (© IEEE 2023).

has been powered off. An adversary can simply read all the data stored in NVM cells with cold-boot attacks [123], [124]. Furthermore, even for volatile memories, IMC accelerators store the model parameters for long periods of time in regular structures on-chip, and they are susceptible to voltage microprobing attacks [81], [227], [228].

What further complicates the defense solutions for IMC accelerators is that compute and memory are no longer separable. Thus, while the data has to be stored in encrypted ciphertexts for memory security, we cannot simply decrypt the data before the computation (e.g., when the data is fetched from the off-chip memory) as in conventional DNN accelerators. Recent works proposed several approaches to address challenges with memory encryption in IMC accelerators.

The first approach is to integrate decryption with the computation with a simple encryption scheme [200], [201], [202], [229], [230]. For example, [229], [230] XOR-ed the model parameters with secrets before storing them in the bit cells of an IMC accelerator. Then, the inverse of those secrets is applied to the input tensors before the computation, such that the secrets will be canceled out in the computation outputs. Thus, these works can perform decryption on the fly without having to explicitly do this operation on the model parameters. However, the major source of overhead in this scheme is generating those secrets that have the same size as the model parameters of the entire DNN, which was achieved by using a stream cipher in [230]. Instead of generating highly-secure secrets that are only used once as in [230], other works often used less secure but more efficient schemes to generate secrets, such as using the intrinsic startup value of bit cells as secrets [200] and flipping entire rows and columns of the memory instead of XOR-ing the data with secrets [201], [202].

The second approach is to leverage arithmetic sharing such that IMC accelerators store and compute only one share of the model parameters [231]. The host secure processor operates on the other share of the model parameters and combines the computation results of the IMC accelerator to obtain the final outputs [231]. However, this approach incurs the overhead of twice the compute, as the computations have to be performed on two arithmetic shares.

Third, similarly to its use in physical SCA defenses (Section IV-B2), shuffling can also be applied to IMC accelerators for security [161], [232], [233]. The model parameters can be mapped to IMC bit cells in a permuted manner, and the computation can correct the permutation at the final step.

Lastly, some works [232], [234] leveraged the intrinsic physical characteristics of the analog components in IMC accelerators, such as the circuit offsets [232] and retention limitations [234], to provide the security. For example, [232] proposed fine-tuning a per-chip model that reflects the unavoidable analog-to-digital converter offset, such that the model parameters are specific to each chip and an adversary cannot achieve the same performance in another chip using the extracted model parameters.

3) Limitations: The defense solutions for IMC accelerators are limited except for very recent work on a digital IMC accelerator [226], and analysis of how such protections translate to more traditional analog IMC accelerators without affecting compute SNR and performance remains an open question.

E. Hardware Trojan Detection

There have been many methods proposed for hardware trojan detection. At a high level, these utilize methods such as functional or side-channel testing to detect the existence of the trojan, or modify the design to make trojan insertion difficult through methods such as redundancy and operation obfuscation.

Functional testing is likely to find trojans that have a high probability of being triggered with a payload that has an obvious impact on the chip's operation [235], [236], [237]. However, more complex trigger designs that aim for a low chance of accidental triggering will also make it unlikely for it to be detected with this form of testing.

On the other hand, side-channel testing uses unintentional effects of the addition of trigger and payload circuitry on the power consumption and currents to detect the presence of this additional circuitry [238], [239], [240], [241], [242]. However, since it is quite difficult to have an accurate power model for complex CMOS circuits, many such methods depend on a golden circuit that is guaranteed to be trojan-free to compare against. Prior work has shown the possibility of combining high spatial resolution and wide field of view novel sensing methods with unsupervised deep learning for unbiased and golden circuit-free trojan detection at high sensitivity [243].

Many of the accelerator trojan triggers and payloads have quite small footprints while also being rare enough to escape detection via traditional functional verification. However, some work has considered how to find such trojans in the context of machine learning accelerators [155], [244]. For example, some trojan triggers specifically wait for data around the three-sigma limit of the input distribution, which is learned based on the validation dataset provided to the hardware user for functional verification [35], [245]. One low overhead but not fool-proof method to avoid this changes the

The CKKS scheme is currently the most popular FHE scheme for DNNs. While the CKKS scheme can support DNNs, it does require an expensive operation known as bootstrapping to be performed frequently.

distribution of the inputs in the validation set from that of the expected data [245].

In addition, redundancy can be used for trojan detection by having an additional processing element that is nominally identical to each of the regular elements at different times [246]. Thus, any deviation between two outputs indicates some malicious change. However, similarly to ECC [247], [248], the amount of redundancy creates an obvious tradeoff between the significant overheads and the number of errors it can detect or correct.

Similarly, obfuscation can be used to shuffle images at the pixel and block levels and to apply random mapping between pixel values to hide the input image data [249]. Thus, triggers that depend on specific patterns of data in a certain order or location are difficult to target by an attacker, but this does not apply to triggers that just detect a certain value.

1) Limitations: Most importantly, many of these security or detection mechanisms are incumbent on a specific type of trigger or payload being present, which cannot be known a priori. A general technique that can apply to many varieties of trojans is necessary. One solution that has been explored is a verifiable ASIC, which provides a computationally secure way to detect any trojan that causes a change in the outputs [250]. By providing an interactive proof specialized for matrix multiplication [251] and non-linear activations, there is a negligible chance that a trojan-inserted chip can give the correct response to all the verifier's queries [250]. Some optimizations can be added to exploit neural network-specific parameters such as high sparsity and reuse of existing convolutional operations for the proof [250], but this does still add overheads to the original chip design. Furthermore, we need a detection scheme that is also verifiable against attacks with effects other than modifying the final outputs. To this end, a hybrid solution of functional and side-channel testing, which has minimal dependence on on-chip detection circuitry seems most appropriate.

F. Fully Homomorphic Encryption for Privacy-Preserving Computing

Fully Homomorphic Encryption (FHE) has emerged as a mainstream technology in privacy-preserving computing over the past decade. FHE enables cloud operators to perform computations on *encrypted* data without ever requiring to decrypt it. The result of such computation is also in an encrypted form. This encrypted result can only be decrypted by the data owner who holds the private/secret key. An illustrative use case of how a data owner can outsource computation on private data to an untrusted third-party cloud platform is shown in Figure 10.

Since the data remains encrypted throughout the computation process, physical SCAs (Section III-B1) on a processor or data readout attacks on an accelerator memory (Section III-A1) fail to disclose meaningful information about the plaintext data or intermediate computations. To guarantee the confidentiality of data, FHE does not require any trusted hardware, such as a trusted execution environment (Section IV-A1) either.

However, despite its robust data privacy guarantees, FHE is not widely adopted. This is primarily due to its inherent slowness when performing computations on encrypted data, which stems from its compute and memory-intensive nature [252], [253], [254]. This performance issue persists across different FHE schemes, including BGV [255], BFV [256], CKKS [257], and TFHE [258]. All of these schemes, by mathematical construction, support operations on either integers, real numbers, or bits, which enables a wide variety of real-world applications.

The CKKS scheme is currently the most popular FHE scheme for DNNs. While the CKKS scheme can support DNNs, it does require an expensive operation known as bootstrapping to be performed frequently. The noise added from the FHE scheme grows with each homomorphic encryption being performed. Once it grows beyond a certain level, it is impossible to continue computing any further with correctness. This bootstrapping operation de-noises the ciphertext and depending upon the scheme parameters, can take several seconds [259] to several minutes [254].

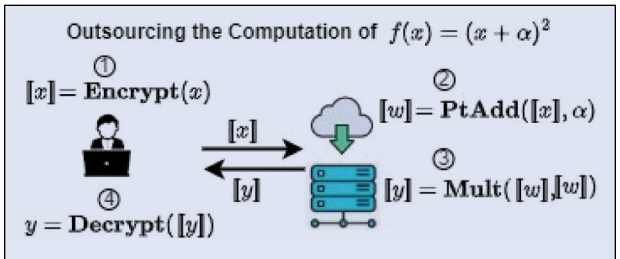


Figure 10. Third-party cloud platform with outsourced FHE-based computing. The figure is reproduced from [252].

There have been many recent algorithmic [260], [261], [262], [263], software [259], [264], [265], and hardware [266], [267], [268], [269], [270], [272] optimizations to the CKKS primitive and bootstrapping operations to help reduce the bootstrapping execution time. However, bootstrapping remains a performance bottleneck as it consumes up to 95% of the total application execution time [259], [271] when performing deep neural network training and inference. This underscores the critical need for accelerating bootstrapping to improve the performance of deep neural networks while performing training and inference using encrypted data.

There have been initiatives to accelerate CKKS on both FPGA and ASIC platforms. HEAX [272] emerged as an early FPGA-based accelerator for FHE which accelerated only CKKS encrypted multiplication, while all other operations were delegated to a host processor. One of the more recent FPGA implementations, FAB [266], implemented packed CKKS bootstrapping for the first time on an FPGA with support for practical parameter sets. The FAB design balances the compute and memory requirements of the encrypted deep neural networks, while being cognizant of the compute and on-chip memory constraints of the underlying FPGA. Even though FAB outperforms all prior CPU/GPU implementations by 9.5× to 456×, the performance was still limited by the bootstrapping implementation, which could not be parallelized on multiple FPGAs.

The first ASIC design was proposed by Samardzic et al. [267], and further optimized in their subsequent work, CraterLake [269]. In parallel, Kim et al. proposed three other ASIC designs namely BTS [268], ARK [270], and SHARP [271]. These designs incorporate various optimizations and have demonstrated remarkable performance in bootstrapping. These ASIC solutions are promising as they outperform CPU/GPU/FPGA implementations and narrow the gap between the performance of computing on plaintext vs. ciphertext when evaluating encrypted deep neural networks.

However, to achieve this performance improvement, they make use of a large number of custom modular multipliers, large register files, and large on-chip memory. To enable such architectures, these ASIC implementations must use expensive advanced technology nodes like 7 nm or 12 nm, making these solutions extremely expensive.

1) Limitations: All the aforementioned research works contribute to ongoing efforts aimed at improving the practicality and scalability of FHE schemes, particularly in the context of deep neural networks. Further FHE algorithmic improvements will play a significant role in this direction, demonstrating promising progress towards making secure computations on encrypted data more efficient and viable for real-world applications.

V. Future Work and Conclusion

Looking forward, the security of DNN accelerators will continue to be an important challenge. Also, as DNNs are rapidly evolving, new challenges and vulnerabilities can arise.

One particular topic for future research can be on the impact of large language models (LLMs) on hardware security. On one hand, LLMs can be exploited by adversaries to automate attacks, such as generating hardware trojans when provided with the source RTL code [273]. However, LLMs can also offer potential benefits to hardware designers. For example, they can be used to automatically identify vulnerabilities in hardware designs [274], [275].

Rapidly evolving DNN workloads that have increasingly large model sizes, such as LLMs, also present a dual challenge for DNN accelerators. DNN accelerators should not only be secured from hardware-level vulnerabilities, but should also provide efficient support for increasingly compute- and memory-intensive workloads. As such, the scalability of current defense solutions can become a critical challenge, as they have to protect more complex models without significant sacrifice in performance.

In summary, this article provide a comprehensive survey into recent advances in hardware security of DNN accelerators. Compared to general-purpose microprocessors or cryptographic accelerators for which hardware security has been thoroughly investigated, DNN accelerators have similarities (Section II-A) and unique differences (Section II-B) in their hardware-level vulnerabilities. We provide a comprehensive review of recent advances in attacking DNN accelerators exploiting those vulnerabilities (Section III) and defense solutions for DNN accelerators (Section IV). In addition to recent progress, there are many potential future directions spanning across circuits, architecture, and algorithms for secure machine learning hardware that satisfies confidentiality, integrity, and availability.

Acknowledgment

This work was supported in part by the MIT-IBM Watson AI Lab, in part by Samsung Electronics, in part by the Red Hat Collaboratory, in part by the Korea Foundation for Advanced Studies, in part by the National Science Foundation Graduate Research Fellowships Program under Grant 1745302, and in part by the MathWorks Engineering Fellowship. Kyungmi Lee and Maitreyi Ashok contributed equally to this work.

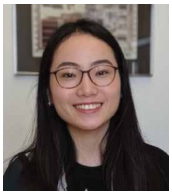
Kyungmi Lee (Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2018, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2020 and 2024, respectively. She is currently a Post-Doctoral Associate at MIT, supervised by Prof. Anantha P. Chandrakasan. Her research focuses on hardware security of deep neural network accelerators and energy-efficient hardware and software design for machine learning applications. She received the MIT Jacobs Presidential Fellowship in 2018, the Siebel Scholars in 2020, and the Doctoral Fellowship from Korea Foundation for Advanced Studies from 2018 to 2023. She received the Bob Owens Best Student Paper Award at the IEEE International Workshop on Signal Processing Systems in 2021.

Maitreyi Ashok (Student Member, IEEE) received the B.S. degree in electrical engineering from the California Institute of Technology (Caltech), Pasadena, CA, USA, in 2019, and the S.M. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2021. She is currently pursuing the Ph.D. degree at MIT, supervised by Prof. Anantha P. Chandrakasan. Her research focuses on hardware security in digital and mixed-signal integrated circuits against physical attacks including passive side-channels and hardware trojan insertion. She has previously interned at Intel Labs, Hillsboro, OR, USA, in 2021, and Microsoft, Redmond, WA, USA, in 2018 and 2019. She received the National Science Foundation Graduate Research Fellowship in 2019, the Analog Devices Fellowship in 2019, and the MathWorks Engineering Fellowship in 2021.

Saurav Maji (Member, IEEE) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology (IIT) Kharagpur, India, in 2017, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 2019 and 2023, respectively. Since July 2023, he has been affiliated as a Research Scientist with the Circuits Research Laboratory, Intel Labs, Intel, Hillsboro, OR, USA. His research interests include energy-efficient security solutions for the IoT applications, side-channel security, and fault analysis for machine learning and artificial intelligence applications, and circuits for biomedical applications. He was a recipient of the President of India Gold Medal from IIT Kharagpur in 2017, the Analog Devices Fellowship from MIT in 2017, and the MIT EECS MathWorks Fellowship from MIT in 2022. He received the Second Place Winner Award in the Design Contest at the 2020 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED). Additionally, he was a recipient of the Student Research Preview (SRP) Poster Award at the IEEE International Solid-State Circuits Conference (ISSCC) 2023.

Rashmi Agrawal (Member, IEEE) received the Ph.D. degree in computer engineering from Boston University, where her research focused on designing hardware accelerators for post-quantum cryptography and fully homomorphic encryption. She is currently a Research Scientist with the Massachusetts Institute of Technology (MIT) and Boston University. She is also a Co-Founder of CipherSonic Labs, a venture dedicated to advancing data security and privacy solutions. Previously, She was a Research Scientist at Intel Labs' Security and Privacy Research Group. This work aimed to enable postquantum secure, practical, and cost-effective private data analytics in cloud environments. She was honored with the EECS Rising Star Award in 2023 for her Ph.D. research. In addition, she has received the Best Paper Awards at ICCD 2020 and GLSVLSI 2020. During her Ph.D., she gained industry experience at Xilinx (AMD) and Analog Devices.

Ajay Joshi (Senior Member, IEEE) received the Ph.D. degree from the Georgia Institute of Technology. He was a Post-Doctoral Researcher at the Massachusetts Institute of Technology (MIT). He has worked as a Visiting Researcher at Google and as an Architect at Lightmatter Inc. He is currently a Professor with the ECE Department, Boston University. He recently co-founded CipherSonic Labs Inc., which is focused on commercializing FHE-based computing. His research interests include computer architecture and digital VLSI, with a focus on security and privacy, machine learning, and photonic computing. He received the NSF CAREER Award in 2012, the Boston University ECE Department's Award for Excellence in Teaching in 2014, the Best Paper Awards at ASIACCS 2018 and HOST 2023, and the Google Faculty Research Awards in 2018 and 2019. He also serves as an Associate Editor for IEEE Transactions on VLSI Systems.



Mengjia Yan (Member, IEEE) received the Ph.D. degree from the University of Illinois at Urbana–Champaign (UIUC). She is currently an Associate Professor with the Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology. Her research interests include computer architecture and hardware security, with a focus on side channel attacks and defenses. She received the IEEE/TCCA Young Computer Architect Award in 2024, the NSF CAREER Award in 2021, and the Intel Rising Star Faculty Award in 2022. Her work has been recognized as an ACM Research Highlight, multiple MICRO TopPicks in computer architecture, and the MICRO Best Paper Award.

Joel S. Emer (Fellow, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1974 and 1975, respectively, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 1979. He was with Intel, where he was an Intel Fellow and the Director of Microarchitecture Research. At Intel, he led the VSSAD Group, which he had previously been a member of Compaq and Digital Equipment Corporation. He is currently a Senior Distinguished Research Scientist with Nvidia Architecture Research Group, Westford, MA, USA, where he is responsible for exploration of future architectures and modeling and analysis methodologies. He is also a Professor of practice at the Massachusetts Institute of Technology, Cambridge, MA, USA, where he teaches computer architecture and supervises graduate students. He has held various research and advanced development positions investigating processor microarchitecture and developing performance modeling and evaluation techniques. He has made architectural contributions to a number of VAX, Alpha, and X86 processors and is recognized as one of the developers of the widely employed quantitative approach to processor performance evaluation. He has been recognized for his contributions in the advancement of simultaneous multithreading technology, processor reliability analysis, cache organization, and spatial architectures for deep learning. He is a fellow of the Association for Computing Machinery (ACM). He was a recipient of numerous public recognitions. He received the Eckert-Mauchly Award in 2009 and the B. Ramakrisna Rau Award in 2023 for lifetime contributions in computer architecture, the Purdue University Outstanding Electrical and Computer Engineer Alumni Award in 2010, and the University of Illinois Electrical and Computer Engineering Distinguished Alumni Award in 2011. His 1996 article on simultaneous multithreading received the ACM/SIGARCH-IEEECS/TCCA: Most Influential Paper Award in 2011. He was named to the ISCA and Micro Halls of Fame in 2005 and 2015, respectively. He has had six articles selected for the IEEE Micro Top Picks in computer architecture, in 2003, 2004, 2007, 2013, 2015, and 2016. He was the Program Chair of ISCA in 2000 and Micro in 2017.

Anantha P. Chandrakasan (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, in 1989, 1990, and 1994, respectively. Since September 1994, he has been with the Massachusetts Institute of Technology, Cambridge, where he is currently the Vannevar Bush Professor of electrical engineering and computer science. He was the Director of Microsystems Technology Laboratories, MIT, from 2006 to 2011. From July 2011 to June 2017, he was the Head of the Department of Electrical Engineering and Computer Science, MIT. Since July 2017, he has been the Dean of the School of Engineering, MIT, and since January 2024, he has also been the MIT Chief Innovation and Strategy Officer. He is a co-author of *Low Power Digital CMOS Design* (Kluwer Academic Publishers, 1995), *Digital Integrated Circuits* (Pearson Prentice-Hall, 2003, 2nd edition), and *Subthreshold Design for Ultra-Low Power Systems* (Springer, 2006). His research interests include ultra-low-power circuit and system design, energy harvesting, energy efficient RF circuits, and hardware security. He was elected as fellow of the Association for Computing Machinery (ACM) in 2020. He was a co-recipient of several awards including the 2007 ISSCC Beatrice Winner Award for Editorial Excellence and the ISSCC Jack Kilby Award for Outstanding Student Paper in 2007, 2008, and 2009. He received the 2009 Semiconductor Industry Association (SIA) University Researcher Award and the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits, an Honorary Doctorate from KU Leuven in 2016, the UC Berkeley EE Distinguished Alumni Award in 2017, and the 2019 IEEE Solid-State Circuits Society Distinguished Service Award. In 2015, he was elected to the National Academy of Engineering and elected to the American Academy of Arts and Sciences in 2019. He was a recipient of the 2022 IEEE Mildred Dresselhaus Medal. He has served in various roles for the IEEE ISSCC including Program Chair, the Signal Processing Sub-committee Chair, and theTechnology Directions Sub-Committee Chair. He served as the Conference Chair of ISSCC from 2010 to 2018. He served as the Senior Technical Advisor to the Conference from ISSCC 2019 to ISSCC 2023.

References

[1] *GPT-4 Technical Report*, OpenAI, San Francisco, CA, USA, 2024.

[2] F. Isensee et al., "NnU-Net: A self-configuring method for deep learning-based biomedical image segmentation," *Nature Methods*, vol. 18, no. 2, pp. 203–211, Feb. 2021.

[3] J. Jumper et al., "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, Aug. 2021, doi: 10.1038/s41586-021-03819-2.

[4] S. Grigorescu et al., "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020.

[5] A. Mirhoseini et al., "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[6] *The Health Insurance Portability and Accountability Act (HIPAA)*, U.S. Dept. Labor, Employee Benefits Security Administration, Washington, DC, USA, 2004.

[7] E. Strickland. (Apr. 2024). *15 Graphs That Explain the State of AI in 2024: The AI Index Tracks the Generative AI Boom, Model Costs, and Responsible AI Use*. IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/ai-index-2024

[8] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[9] A. Madry et al., "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[10] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1211–1220.

[11] F. Yao, A. S. Rakin, and D. Fan, "DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *Proc. USENIX Secur. Symp.*, 2020, pp. 1463–1480. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/yao

[12] S. Hong et al., "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *Proc. USENIX Secur. Symp.*, 2019, pp. 497–514.

[13] K. Lee and A. P. Chandrakasan, "SparseBFA: Attacking sparse deep neural networks with the worst-case bit flips on coordinates," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 4208–4212.

[14] G. Li et al., "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.,* Nov. 2017, pp. 1–12, doi: 10.1145/3126908.3126964.

[15] W. Liu et al., "Imperceptible misclassification attack on deep learning accelerator by glitch injection," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[16] Y. Luo et al., "DeepStrike: Remotely-guided fault injection attacks on DNN accelerator in cloud-FPGA," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 295–300.

[17] A. S. Rakin et al., "Deep-dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant FPGA," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1919–1936.

[18] M. Sinha et al., "Securing an accelerator-rich system from flooding-based denial-of-service attacks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 855–869, Jan. 2021.

[19] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of DoS attacks in NoC based SoCs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1160–1165.

[20] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 19th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, 1999, pp. 388–397.

[21] D. Agrawal et al., "The EM side—Channel(s)," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds., Berlin, Germany: Springer, 2003, pp. 29–45.

[22] Y. Kim et al., "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Minneapolis, MN, USA, Jun. 2014, pp. 361–372.

[23] O. Mutlu and J. S. Kim, "RowHammer: A retrospective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 8, pp. 1555–1571, Aug. 2020, doi: 10.1109/TCAD.2019.2915318.

[24] L. Szekeres et al., "SoK: Eternal war in memory," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2013, pp. 48–62.

[25] Y.-S. Won et al., "DeepFreeze: Cold boot attacks and high fidelity model recovery on commercial EdgeML device," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9.

[26] F. Liu et al., "Last-level cache side-channel attacks are practical," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 605–622.

[27] M. Lipp et al., "Meltdown: Reading kernel memory from user space," in *Proc. 27th USENIX Secur. Symp. (USENIX Secur.)*, 2018, pp. 973–990. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[28] P. Kocher et al., "Spectre attacks: Exploiting speculative execution," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1–19.

[29] J. Clements and Y. Lao, "Hardware trojan design on neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[30] V. Venceslai et al., "NeuroAttack: Undermining spiking neural networks security through externally triggered bit-flips," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[31] A. Warnecke et al., "Evil from within: Machine learning backdoors through hardware trojans," 2023, *arXiv:2304.08411*.

[32] Z. Liu et al., "Sequence triggered hardware trojan in neural network accelerator," in *Proc. IEEE 38th VLSI Test Symp. (VTS)*, Apr. 2020, pp. 1–6.

[33] H. Xu et al., "Exploiting logic locking for a neural trojan attack on machine learning accelerators," in *Proc. Great Lakes Symp. VLSI*, Jun. 2023, pp. 351–356, doi: 10.1145/3583781.3590242.

[34] S.-H. Huang, W.-C. Cheng, and J.-F. Li, "Hardware trojans of Computing-In-memories: Issues and methods," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2023, pp. 1–6.

[35] T. A. Odetola et al., "FeSHI: Feature map-based stealthy hardware intrinsic attack," *IEEE Access*, vol. 9, pp. 115370–115387, 2021.

[36] R. Mukherjee and R. S. Chakraborty, "Novel hardware trojan attack on activation parameters of FPGA-based DNN accelerators," *IEEE Embedded Syst. Lett.*, vol. 14, no. 3, pp. 131–134, Sep. 2022.

[37] C. Yang et al., "Hardware trojan attacks on the reconfigurable interconnections of convolutional neural networks accelerators," in *Proc. IEEE 15th Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Nov. 2020, pp. 1–3.

[38] J. A. Halderman et al., "Lest we remember: Cold boot attacks on encryption keys," in *Proc. 17th USENIX Secur. Symp. (USENIX Secur.)*, Jul. 2008, pp. 1–16. [Online]. Available: https://www.usenix.org/conference/17th-usenix-security-symposium/lest-we-remember-cold-boot-attacks-encryption-keys

[39] Y.-H. Chen et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[40] V. Sze et al., "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[41] T. Chen et al., "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. Int. Conf. Architect. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA, Mar. 2014, pp. 269–284, doi: 10.1145/2541940.2541967.

[42] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 243–254, doi: 10.1145/3007787.3001163.

[43] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[44] N. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architect.*, Jun. 2017, pp. 1–12, doi: 10.1145/3079856.3080246.

[45] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, 2016, pp. 367–379.

[46] M. Pellauer et al., "Buffets: An efficient and composable storage idiom for explicit decoupled data orchestration," in *Proc. Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, 2019, pp. 137–151, doi: 10.1145/3297858.3304025.

[47] A. Parasha et al., "Timeloop: A systematic approach to DNN accelerator evaluation," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2019, pp. 304–315.

[48] D. Perez-Botero, J. Szefer, and R. B. Lee, "Characterizing hypervisor vulnerabilities in cloud computing servers," in *Proc. Int. Workshop Secur. cloud Comput.,* May 2013, pp. 3–10, doi: 10.1145/2484402.2484406.

[49] P. Colp et al., "Breaking up is hard to do: Security and functionality in a commodity hypervisor," in *Proc. 33rd ACM Symp. Operating Syst. Princ.,* Oct. 2011, pp. 189–202, doi: 10.1145/2043556.2043575.

[50] J. Szefer et al., "Eliminating the hypervisor attack surface for a more secure cloud," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 401–412, doi: 10.1145/2046707.2046754.

[51] H. Chen et al., "Linux kernel vulnerabilities: State-of-the-art defenses and open problems," in *Proc. 2nd Asia–Pacific Workshop Syst.,* Jul. 2011, pp. 1–5, doi: 10.1145/2103799.2103805.
[52] Y. Xiao et al., "One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*, Aug. 2016, pp. 19–35. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/xiao
[53] M. D. Hill et al., "On the spectre and meltdown processor security vulnerabilities," *IEEE Micro*, vol. 39, no. 2, pp. 9–19, Mar. 2019.
[54] Y. Yarom and K. Falkner, "FLUSH+RELOAD: A high resolution, low noise, l3 cache side-channel attack," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, Aug. 2014, pp. 719–732. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14technical-sessions/presentation/yarom
[55] V. Costan and S. Devadas "Intel SGX explained," *Cryptol. ePrint Arch.,* 2016. [Online]. Available: https://eprint.iacr.org/2016/086
[56] Q. Liu, L. Guo, and H. Tang, "Fault model analysis of DRAM under electromagnetic fault injection attack," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.
[57] B. Colombier et al., "Laser-induced single-bit faults in flash memory: Instructions corruption on a 32-bit microcontroller," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2019, pp. 1–10.
[58] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6, doi: 10.1109/DAC.2018.8465773.
[59] X. Hu et al., "DeepSniffer: A DNN model extraction framework based on learning architectural hints," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 385–399, doi: 10.1145/3373376.3378460.
[60] D. Yang, P. J. Nair, and M. Lis, "HuffDuff: Stealing pruned DNNs from sparse accelerators," in *Proc. 28th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, vol. 2, Jan. 2023, pp. 385–399, doi: 10.1145/3575693.3575738.
[61] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on GPUs," in *Proc. 13th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Oct. 2018, pp. 681–696. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/volos
[62] H. Jun et al., "HBM (high bandwidth memory) DRAM technology and architecture," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2017, pp. 1–4.
[63] *High Bandwidth Memory (HBM) Dram*, document JESD235D, JEDEC, 2021.
[64] I. Jang et al., "Heterogeneous isolated execution for commodity GPUs," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.,* Apr. 2019, pp. 455–468, doi: 10.1145/3297858.3304021.
[65] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Comput. Netw.*, vol. 48, no. 5, pp. 701–716, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128605000125
[66] C. Gongye, Y. Fei, and T. Wahl, "Reverse-engineering deep neural networks using floating-point timing side-channels," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[67] V. Duddu et al., "Stealing neural networks via timing side channels," 2018, *arXiv:1812.11720*.
[68] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—Attacks and defenses," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12079–12092, Aug. 2021.
[69] L. Batina et al., "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *Proc. USENIX Secur. Symp.*, 2019, pp. 515–532. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/batina
[70] L. Wei et al., "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, Dec. 2018, pp. 393–406, doi: /3274694.3274696.
[71] S. Moini et al., "Remote power side-channel attacks on BNN accelerators in FPGAs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1639–1644.
[72] K. Yoshida et al., "Model-extraction attack against FPGA-DNN accelerator utilizing correlation electromagnetic analysis," in *Proc. IEEE 27th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2019, pp. 318–318.
[73] K. Yoshida et al., "Model reverse-engineering attack against systolic-array-based DNN accelerator using correlation power analysis," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E104.A, no. 1, pp. 152–161, 2021.
[74] Y.-S. Won et al., "WaC: First results on practical side-channel attacks on commercial machine learning accelerator," in *Proc. 5th Workshop Attacks Solutions Hardw. Secur.,* Nov. 2021, pp. 111–114, doi: 10.1145/3474376.3487284.
[75] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The first hardware inference engine aiming power side-channel protection," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Dec. 2020, pp. 197–208, doi: 10.1109/HOST45689.2020.9300276.
[76] H. Naghibijouybari et al., "Rendered insecure: GPU side channel attacks are practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 2139–2153, doi: 10.1145/3243734.3243831.
[77] E. Karimi, Z. H. Jiang, Y. Fei, and D. Kaeli, "A timing side-channel attack on a mobile GPU," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 67–74.
[78] W. Liu, "Imperceptible misclassification attack on deep learning accelerator by glitch injection," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[79] X. Hou et al., "Physical security of deep learning on edge devices: Comprehensive evaluation of fault injection attack vectors," *Microelectron. Rel.*, vol. 120, May 2021, Art. no. 114116. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271421000822
[80] H. Wang et al., "Probing attacks on integrated circuits: Challenges and research opportunities," *IEEE Des. Test*, vol. 34, no. 5, pp. 63–71, Oct. 2017.
[81] M. T. Rahman et al., "Physical inspection & attacks: New frontier in hardware security," in *Proc. IEEE 3rd Int. Verification Secur. Workshop (IVSW)*, Jul. 2018, pp. 93–102.
[82] X. Zhang, A. A. Ding, and Y. Fei, "Deep-learning model extraction through software-based power side-channel," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Oct. 2023, pp. 1–9.
[83] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *Proc. 29th USENIX Secur. Symp. (USENIX Secur.)*, Aug. 2020, pp. 2003–2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/yan
[84] F. Schellenberg et al., "An inside job: Remote power analysis attacks on FPGAs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 1111–1116.
[85] S. Tian et al., "A practical remote power attack on machine learning accelerators in cloud FPGAs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.
[86] S. Moini et al., "Power side-channel attacks on BNN accelerators in remote FPGAs," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 357–370, Jun. 2021.
[87] Y. Zhang et al., "Stealing neural network structure through remote FPGA side-channel analysis," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4377–4388, 2021.
[88] J. Mahmod and M. Hicks, "SRAM has no chill: Exploiting power domain separation to steal on-chip secrets," in *Proc. 27th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.,* Feb. 2022, pp. 1043–1055, doi: 10.1145/3503222.3507710.
[89] J. Mahmod and M. Hicks, "UnTrustZone: Systematic accelerated aging to expose on-chip secrets," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2024, pp. 4107–4124, doi: 10.1109/sp54263.2024.00069.
[90] L. Lin et al., "Trojan side-channels: Lightweight hardware trojans through side-channel engineering," in *Proc. 11th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 5747, Lausanne, Switzerland. Springer, 2009, pp. 382–395. [Online]. Available: https://www.iacr.org/archive/
[91] A. De et al., "HarTBleed: Using hardware trojans for data leakage exploits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 4, pp. 968–979, Apr. 2020.
[92] G. T. Becker et al., "Implementing hardware trojans: Experiences from a hardware trojan challenge," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 301–304.
[93] D. Knichel, T. Moos, and A. Moradi, "The risk of outsourcing: Hidden SCA trojans in third-party IP-cores threaten cryptographic ICs," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2020, pp. 1–6.
[94] P. Gaikwad et al., "Hardware IP assurance against trojan attacks with machine learning and post-processing," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 19, no. 3, pp. 1–23, Jun. 2023, doi: 10.1145/3592795.

[95] N. Gupta, A. Jati, and A. Chattopadhyay, "AI attacks AI: Recovering neural network architecture from NVDLA using AI-assisted side channel attack," Cryptol. ePrint Arch., Tech. Rep. 2023/368, 2023. [Online]. Available: https://eprint.iacr.org/2023/368
[96] J. A. Halderman et al., "Lest we remember: Cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, pp. 91–98, May 2009, doi: 10.1145/1506409.1506429.
[97] H. Yu et al., "DeepEM: Deep neural networks model recovery through EM side-channel information leakage," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Dec. 2020, pp. 209–218.
[98] X. Yan et al., "MERCURY: An automated remote side-channel attack to Nvidia deep learning accelerator," in *Proc. Int. Conf. Field Program. Technol. (ICFPT)*, Dec. 2023, pp. 188–197.
[99] S. Maji, K. Lee, and A. P. Chandrakasan, "SparseLeakyNets: Classification prediction attack over sparsity-aware embedded neural networks using timing side-channel information," *IEEE Comput. Archit. Lett.*, vol. 23, no. 1, pp. 133–136, Jan. 2024.
[100] H. Bar-El et al., "The Sorcerer's apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, no. 2, pp. 370–382, Feb. 2006.
[101] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology—EUROCRYpt '97*. Berlin, Germany: Springer, 1997, pp. 37–51.
[102] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–16.
[103] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2012, pp. 356–367.
[104] Q. Zhang et al., "ApproxANN: An approximate computing framework for artificial neural network," in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, 2015, pp. 701–706.
[105] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing," *Proc. IEEE*, vol. 107, no. 1, pp. 108–122, Jan. 2019.
[106] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 14–26, doi: 10.1109/ISCA.2016.12.
[107] M. Onen et al., "Nanosecond protonic programmable resistors for analog deep learning," *Science*, vol. 377, no. 6605, pp. 539–543, Jul. 2022.
[108] S. Ambrogio et al., "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, pp. 60–67, Jun. 2018.
[109] M. Dworkin et al., *Advanced Encryption Standard (AES)*. Gaithersburg, MD, USA: National Institute of Standards and Technology, Nov. 2001.
[110] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–11. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[111] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[112] M. Sandler et al., "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[113] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[114] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[115] M. Rastegari et al., "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Switzerland: Springer, 2016, pp. 525–542.
[116] M. Courbariaux et al., "Binarized neural networks," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 1–10. [Online]. Available: https://api.semanticscholar.org/CorpusID:14796162
[117] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Sep. 2018, pp. 2704–2713.
[118] C. Zhu et al., "Trained ternary quantization," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–7. [Online]. Available: https://openreview.net/forum?id=S1_pAu9xl
[119] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, Dec. 2015, pp. 1135–1143.
[120] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–18, Feb. 2017, doi: 10.1145/3005348.
[121] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5068–5076.
[122] P. Molchanov et al., "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11256–11264.
[123] A. S. Rakin et al., "T-BFA: Targeted bit-flip adversarial weight attack," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7928–78939, Sep. 2021.
[124] M. Gruhn and T. Müller, "On the practicability of cold boot attacks," in *Proc. Int. Conf. Availab., Rel. Secur.*, Regensburg, Germany, 2013, pp. 390–397.
[125] J. A. Halderman et al., "Lest we remember: Cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, pp. 91–98, May 2009, doi: 10.1145/1506409.1506429.
[126] (2024). *Intel Openvino Toolkit*. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html
[127] Y. He et al., "Understanding and mitigating hardware failures in deep learning training systems," in *Proc. 50th Annu. Int. Symp. Comput. Archit.,* Jun. 2023, pp. 1–16, doi: 10.1145/3579371.3589105.
[128] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO' 99*, M. Wiener, Ed., Berlin, Germany: Springer, 1999, pp. 388–397.
[129] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smartcard security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.
[130] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, 2004, pp. 16–29.
[131] D. Hwang et al., "AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis SideChannel attacks," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 781–792, Mar. 2006.
[132] D. Das et al., "X-DeepSCA: Cross-device deep learning side channel attack," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.
[133] T. Jeong, A. P. Chandrakasan, and H.-S. Lee, "S2ADC: A 12-bit, 1.25-MS/s secure SAR ADC with power side-channel attack resistance," *IEEE J. Solid-State Circuits*, vol. 56, no. 3, pp. 844–854, Mar. 2021.
[134] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: A survey," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 135–162, Jun. 2020.
[135] Z. Martinasek, J. Hajny, and L. Malina, "Optimization of power analysis using neural network," in *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds., Cham, Switzerland: Springer, 2014, pp. 94–107.
[136] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, B. S. Kaliski, Ç. K. Koç, and C. Paar, Eds., Berlin, Germany: Springer, 2003, pp. 13–28.
[137] L. Weissbart, S. Picek, and L. Batina, "One trace is all it takes: Machine learning-based side-channel attack on EdDSA," in *Security, Privacy, and Applied Cryptography Engineering*, S. Bhasin, A. Mendelson, and M. Nandi, Eds., Cham, Switzerland: Springer, 2019, pp. 86–105.
[138] S. B. Ors et al., "Power-analysis attack on an ASIC AES implementation," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, Apr. 2004, pp. 546–552.
[139] M. Alioto et al., "Leakage power analysis attacks: A novel class of attacks to nanometer cryptographic circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 2, pp. 355–367, Feb. 2010.
[140] Y.-I. Hayashi et al., "Analysis of electromagnetic information leakage from cryptographic devices with different physical structures," *IEEE Trans. Electromagn. Compat.*, vol. 55, no. 3, pp. 571–580, Jun. 2013.

[141] A. Ghosh et al., "An EM/power SCA-resilient AES-256 with synthesizable signature attenuation using digital-friendly current source and RO-bleed-based integrated local feedback and global switched-mode control," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 499–501.
[142] M. M. Real and R. Salvador, "Physical side-channel attacks on embedded neural networks: A survey," *Appl. Sci.*, vol. 11, no. 15, p. 6790, Jul. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/15/6790
[143] H. Chabanne et al., "Side channel attacks for architecture extraction of neural networks," *CAAI Trans. Intell. Technol.*, vol. 6, no. 1, pp. 3–16, Mar. 2021, doi: 10.1049/cit2.12026.
[144] V. Meyers, D. Gnad, and M. Tahoori, "Active and passive physical attacks on neural network accelerators," *IEEE Des. Test*, vol. 40, no. 5, pp. 70–85, Mar. 2023.
[145] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Oct. 2012.
[146] Y. Gao et al., "NNLeak: An AI-oriented DNN model extraction attack through multi-stage side channel analysis," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Dec. 2023, pp. 1–6.
[147] M. Kang et al., "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 8326–8330.
[148] N. Verma et al., "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Mag.*, vol. 11, no. 3, pp. 43–55, Aug. 2019.
[149] Z. Wang et al., "PowerGAN: A machine learning approach for power side-channel attack on compute-in-memory accelerators," *Adv. Intell. Syst.*, vol. 5, no. 12, Dec. 2023, Art. no. 2300313, doi: 10.1002/aisy.202300313.
[150] Z. Wang et al., "Side-channel attack analysis on in-memory computing architectures," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 1, pp. 109–121, Jan. 2024.
[151] F. Staudigl, F. Merchant, and R. Leupers, "A survey of neuromorphic computing-in-memory: Architectures, simulators, and security," *IEEE Des. Test*, vol. 39, no. 2, pp. 90–99, Apr. 2022.
[152] G. Singh et al., "A review of near-memory computing architectures: Opportunities and challenges," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czech Republic, 2018, pp. 608–617.
[153] M. T. Arafin and Z. Lu, "Security challenges of processing-in-memory systems," in *Proc. Great Lakes Symp. (VLSI)*, Sep. 2020, pp. 229–234, doi: 10.1145/3386263.3411365.
[154] J. Sepulveda, C. Reinbrecht, and J.-P. Diguet, "Security aspects of neuromorphic MPSoCs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–6.
[155] D. Sanlyde et al., "On a new way to read data from memory," in *Proc. 1st Int. IEEE Secur. Storage Workshop*, Dec. 2002, pp. 65–69.
[156] M. N. I. Khan and S. Ghosh, "Comprehensive study of security and privacy of emerging non-volatile memories," *J. Low Power Electron. Appl.*, vol. 11, no. 4, p. 36, Sep. 2021. [Online]. Available: https://www.mdpi.com/2079-9268/11/4/36
[157] A. Chakraborty, A. Mondal, and A. Srivastava, "Correlation power analysis attack against STT-MRAM based cyptosystems," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, 2017, pp. 171–171, doi: 10.1109/HST.2017.7951835.
[158] M. N. I. Khan et al., "Side-channel attack on STTRAM based cache for cryptographic application," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 33–40.
[159] V. R. Kommareddy et al., "Are crossbar memories secure? New security vulnerabilities in crossbar memories," *IEEE Comput. Archit. Lett.*, vol. 18, no. 2, pp. 174–177, Jul. 2019.
[160] S. Skorobogatov, *Physical Attacks and Tamper Resistance*. New York, NY, USA: Springer, 2012, pp. 143–173, doi: 10.1007/978-1-4419-8080-9_7.
[161] Y. Wang, S. Jin, and T. Li, "A low cost weight obfuscation scheme for security enhancement of ReRAM based neural network accelerators," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 499–504.
[162] M. Holler et al., "High-resolution non-destructive three-dimensional imaging of integrated circuits," *Nature*, vol. 543, no. 7645, pp. 402–406, Mar. 2017, doi: 10.1038/nature21698.
[163] Y.-Y. Tee et al., "Patch-based adversarial training for error-aware circuit annotation of delayered IC images," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 9, pp. 3694–3698, Apr. 2023.
[164] D. Cheng et al., "Hybrid K means clustering and support vector machine method for via and metal line detections in delayered IC images," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 12, pp. 1849–1853, Apr. 2018.
[165] J. Breier et al., "Practical fault attack on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 2204–2206, doi: 10.1145/3243734.3278519.
[166] J. Breier et al., "SNIFF: Reverse engineering of neural networks with fault attacks," *IEEE Trans. Rel.*, vol. 71, no. 4, pp. 1527–1539, Dec. 2022.
[167] M. N. I. Khan, K. Nagarajan, and S. Ghosh, "Hardware trojans in emerging non-volatile memories," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 396–401.
[168] K. Nagarajan, M. N. I. Khan, and S. Ghosh, "ENTT: A family of emerging NVM-based trojan triggers," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2019, pp. 51–60.
[169] L. Wu et al., "Hardware trojans in eNVM neuromorphic devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Apr. 2023, pp. 1–6.
[170] Y. Zhao et al., "Memory trojan attack on neural network accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1415–1420.
[171] B. Gassend, "Caches and hash trees for efficient memory integrity verification," in *Proc. 9th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2003, pp. 295–306.
[172] C. Yan et al., "Improving cost, performance, and security of memory encryption and authentication," in *Proc. 33rd Int. Symp. Comput. Archit.*, 2006, pp. 179–190.
[173] B. Rogers et al., "Using address independent seed encryption and Bonsai Merkle trees to make secure processors OS- and performance-friendly," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2007, pp. 183–196.
[174] M. Taassori, A. Shafiee, and R. Balasubramonian, "VAULT: Reducing paging overheads in SGX with efficient integrity verification structures," in *Proc. 23rd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2018, pp. 665–678, doi: 10.1145/3173162.3177155.
[175] G. Saileshwar et al., "Morphable counters: Enabling compact integrity trees for low-overhead secure memories," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2018, pp. 416–427.
[176] D. Lee et al., "Keystone: An open framework for architecting trusted execution environments," in *Proc. 15th Eur. Conf. Comput. Syst.*, Apr. 2020, pp. 1–16, doi: 10.1145/3342195.3387532.
[177] T. Alves and D. Felton, "Trustzone: Integrated hardware and software security," in *Proc. IECON 41st Annu. Conf. IEEE Ind. Electron. Soc.*, Jan. 2004, pp. 2589–2595.
[178] W. Hua et al., "MGX: Near-zero overhead memory protection for data-intensive accelerators," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, Jun. 2022, pp. 726–741, doi: 10.1145/3470496.3527418.
[179] W. Hua et al., "GuardNN: Secure accelerator architecture for privacy-preserving deep learning," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 349–354, doi: 10.1145/3489517.3530439.
[180] S. Lee et al., "TNPU: Supporting trusted execution with tree-less integrity protection for neural processing unit," in *Proc. IEEE Int. Symp. High-Performance Comput. Archit. (HPCA)*, Apr. 2022, pp. 229–243.
[181] K. Lee et al., "SecureLoop: Design space exploration of secure DNN accelerators," in *Proc. 56th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Nov. 2023, pp. 194–208, doi: 10.1145/3613424.3614273.
[182] S. Banerjee et al., "Triton: Software-defined threat model for secure multi-tenant ML inference accelerators," in *Proc. 12th Int. Workshop Hardw. Architectural Support Secur. Privacy*, Oct. 2023, pp. 19–28, doi: 10.1145/3623652.3623672.
[183] S. Na et al., "Common counters: Compressed encryption counters for secure GPU memory," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 1–13.
[184] S. Kim et al., "MoCA: memory-centric, adaptive execution for multi-tenant deep neural networks," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2023, pp. 828–841.
[185] Q. Liu, W. Wen, and Y. Wang, "Concurrent weight encoding-based detection for bit-flip attack on neural network accelerators," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–8, doi: 10.1145/3400302.3415726.
[186] Y. Cai et al., "Enabling secure in-memory neural network computing by sparse fast gradient encryption," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Westminster, CO, USA, Nov. 2019, pp. 1–8.

[187] P. Zuo et al., "SEALing neural network models in encrypted deep learning accelerators," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 1255–1260.
[188] N. Lin et al., "Chaotic weights: A novel approach to protect intellectual property of deep neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 7, pp. 1327–1339, Jul. 2021.
[189] Y. Che and R. Wang, "DNNCloak: Secure DNN models against memory side-channel based reverse engineering attacks," in *Proc. IEEE 40th Int. Conf. Comput. Design (ICCD)*, Oct. 2022, pp. 89–96.
[190] B. F. Goldstein et al., "Preventing DNN model IP theft via hardware obfuscation," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 267–277, Jun. 2021.
[191] S. Maji et al., "An energy-efficient neural network accelerator with improved protections against fault-attacks," in *Proc. IEEE 49th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2023, pp. 233–236.
[192] E. Ozen and A. Orailoglu, "Low-cost error detection in deep neural network accelerators with linear algorithmic checksums," *J. Electron. Test.*, vol. 36, no. 6, pp. 703–718, Dec. 2020.
[193] S. Burel, A. Evans, and L. Anghel, "Zero-overhead protection for CNN weights," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2021, pp. 1–6.
[194] J. Li et al., "RADAR: Run-time adversarial weight attack detection and accuracy recovery," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 790–795.
[195] E. Stefanov et al., "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 299–310, doi: 10.1145/2508859.2516660.
[196] M. Maas et al., "PHANTOM: Practical oblivious computation in a secure processor," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 311–324, doi: 10.1145/2508859.2516692.
[197] Y. Liu, D. Dachman-Soled, and A. Srivastava, "Mitigating reverse engineering attacks on deep neural networks," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 657–662.
[198] X. Wang et al., "NPUFort: A secure architecture of DNN accelerator against model inversion attack," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*, Apr. 2019, pp. 190–196, doi: 10.1145/3310273.3323070.
[199] T. Zhou, S. Ren, and X. Xu, "ObfuNAS: A neural architecture search-based DNN obfuscation approach," in *Proc. 41st IEEE/ACM Int. Conf. Computer-Aided Design*, Oct. 2022, pp. 1–9, doi: 10.1145/3508352.3549429.
[200] W. Li et al., "P 3 M: A PIM-based neural network model protection scheme for deep learning accelerator," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, Jan. 2019, pp. 633–638, doi: 10.1145/3287624.3287695.
[201] A. Malhotra, C. Wang, and S. K. Gupta, "BNN-flip: Enhancing the fault tolerance and security of compute-in-memory enabled binary neural network accelerators," in *Proc. 29th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2024, pp. 146–152, doi: 10.1109/aspdac58780.2024.10473947.
[202] M. Zou et al., "Enhancing security of memristor computing system through secure weight mapping," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2022, pp. 182–187.
[203] A. Dubey et al., "ModuloNET: Neural networks meet modular arithmetic for efficient hardware masking," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 506–556, Nov. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9306
[204] A. Dubey and A. Aysu, "A full-stack approach for side-channel secure ML hardware," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2023, pp. 186–195.
[205] M. Brosch et al., "A masked hardware accelerator for feed-forward neural networks with fixed-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 2, pp. 231–244, Feb. 2024.
[206] A. Dubey, R. Cammarota, and A. Aysu, "BoMaNet: Boolean masking of an entire neural network," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9, doi: 10.1145/3400302.3415649.
[207] S. Maji et al., "A threshold implementation-based neural network accelerator with power and electromagnetic side-channel countermeasures," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 141–154, Jan. 2023.
[208] Q. Fang et al., "Voltage scaling-agnostic counteraction of side-channel neural net reverse engineering via machine learning compensation and multi-level shuffling," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2023, pp. 1–2.
[209] O. Reparaz et al., "Consolidating masking schemes," in *Advances in Cryptology—CRYPTO 2015*. Berlin, Germany: Springer-Verlag, 2022, pp. 764–783, doi: 10.1007/978-3-662-47989-6_37.
[210] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *Proc. ICICS*, vol. 4307. Cham, Switzerland: Springer, 2006, pp. 529–545.
[211] S. Chari et al., "Towards sound approaches to counteract power-analysis attacks," in *Proc. 19th Annu. Int. Cryptol. Conf.*, vol. 1666, Santa Barbara, CA, USA. Berlin, Germany: Springer, 1999, pp. 398–412.
[212] T. S. Messerges, "Securing the AES finalists against power analysis attacks," in *Proc. Int. Workshop Fast Softw. Encryption*, 2001, pp. 150–164.
[213] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," in *Proc. ACM Workshop Theory Implement. Secur.*, Oct. 2016, p. 3, doi: 10.1145/2996366.2996426.
[214] J.-S. Coron, J. Großschädl, and P. K. Vadnala, "Secure conversion between Boolean and arithmetic masking of any order," in *Proc. CHES*. Cham, Switzerland: Springer, 2014, pp. 188–205. [Online]. Available: https://www.iacr.org/archive/ches2014/87310157/87310157.pdf
[215] J.-S. Coron and L. Goubin, "On Boolean and arithmetic masking against differential power analysis," in *Proc. Int. Workshop Cryptogr. Hardw. Embedded Syst.*, 2000, pp. 231–237.
[216] A. Dubey et al., "Guarding machine learning hardware against physical side-channel attacks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 3, pp. 1–31, Apr. 2022, doi: 10.1145/3465377.
[217] T. Beyne and B. Bilgin, "Uniform first-order threshold implementations," in *Selected Areas in Cryptography—SAC 2016*, R. Avanzi and H. Heys Eds., Cham, Switzerland: Springer, 2017, pp. 79–98.
[218] B. Bilgin et al., "Higher-order threshold implementations," in *Advances in Cryptology—ASIACRYPT 2014*. Berlin, Germany: Springer, 2014, pp. 326–343.
[219] B. Bilgin, J. Daemen, V. Nikov, S. Nikova, V. Rijmen, and G. Van Assche, "Efficient and first-order DPA resistant implementations of Keccak," in *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds. Cham, Switzerland: Springer, 2014, pp. 187–199.
[220] A. Baksi et al., "A survey on fault attacks on symmetric key cryptosystems," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–34, Apr. 2023.
[221] C. Beierle et al., "CRAFT: Lightweight tweakable block cipher with efficient protection against DFA attacks," *IACR Trans. Symmetric Cryptol.*, vol. 2019, no. 1, pp. 5–45, 2019. [Online]. Available: https://tosc.iacr.org/index.php/ToSC/article/view/7396
[222] K. Matsuda et al., "A 286F2/cell distributed bulk-current sensor and secure flush code eraser against laser fault injection attack," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 352–354.
[223] R. Kumar et al., "15.5 a 100Gbps fault-injection attack resistant AES-256 engine with 99.1-to-99.99% error coverage in Intel 4 CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2023, pp. 1–3.
[224] S. Song et al., "An FLL-based clock glitch detector for security circuits in a 5nm FINFET process," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Jun. 2022, pp. 146–147.
[225] S. Mathew et al., "A 6.5GHz 54 mW 64-bit parity-checking adder for 65nm fault-tolerant microprocessor execution cores," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2007, pp. 46–47.
[226] M. Ashok et al., "A secure digital in-memory compute (IMC) macro with protections for side-channel and bus probing attacks," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2024, pp. 1–2.
[227] S. Skorobogatov, "How microprobing can attack encrypted memory," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2017, pp. 244–251.
[228] C. Boit, C. Helfmeier, and U. Kerst, "Security risks posed by modern IC debug and diagnosis tools," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr.*, Aug. 2013, pp. 3–11.
[229] W. Li, "Secure-RRAM: A 40nm 16kb compute-in-memory macro with reconfigurability, sparsity control, and embedded security," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2021, pp. 1–2.
[230] S. Huang et al., "Secure XOR-CIM engine: Compute-in-memory SRAM architecture with embedded XOR encryption," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 12, pp. 2027–2039, Dec. 2021.
[231] W. Xiong et al., "SecNDP: Secure near-data processing with untrusted memory," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Apr. 2022, pp. 244–258.

[232] S. Huang et al., "New security challenges on machine learning inference engine: Chip cloning and model reverse engineering," 2020, *arXiv:2003.09739*.
[233] M. Zou et al., "Security enhancement for RRAM computing system through obfuscating crossbar row connections," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2020, pp. 466–471.
[234] C. Yang et al., "Thwarting replication attack against memristor-based neuromorphic computing system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2192–2205, Oct. 2020.
[235] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Des. Validation Test Workshop*, Nov. 2009, pp. 166–171.
[236] R. S. Chakraborty et al., "MERO: A statistical approach for hardware Trojan detection," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Cham, Switzerland: Springer, 2009, pp. 396–410.
[237] K. Xiao et al., "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 1, pp. 1–23, 2016.
[238] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *Proc. IEEE Int. Symp. Electromagn. Compat. (EMC)*, Aug. 2015, pp. 246–251.
[239] S. Narasimhan et al., "Hardware trojan detection by multiple-parameter side-channel analysis," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2183–2195, Nov. 2013.
[240] S. Bhunia et al., "Hardware Trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014.
[241] Y. Qin and T. Xia, "Sensitivity analysis of ring oscillator based hardware trojan detection," in *Proc. IEEE 17th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2017, pp. 1979–1983.
[242] N. B. Gunti and K. Lingasubramanian, "Efficient static power based side channel analysis for hardware trojan detection using controllable sleep transistors," in *Proc. SoutheastCon*, Apr. 2015, pp. 1–6.
[243] M. Ashok et al., "Hardware trojan detection using unsupervised deep learning on quantum diamond microscope magnetic field images," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 18, no. 4, pp. 1–25, Oct. 2022, doi: 10.1145/3531010.
[244] K. I. Gubbi et al., "Securing AI hardware: Challenges in detecting and mitigating hardware trojans in ML accelerators," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2023, pp. 821–825.
[245] T. A. Odetola, H. R. Mohammed, and S. R. Hasan, "A stealthy hardware trojan exploiting the architectural vulnerability of deep learning architectures: Input interception attack (IIA)," 2019, *arXiv:1911.00783*.
[246] P. Sun, B. Halak, and T. Kazmierski, "Towards hardware trojan resilient design of convolutional neural networks," in *Proc. IEEE 35th Int. Syst. Chip Conf. (SOCC)*, Sep. 2022, pp. 1–6.
[247] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[248] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 124–134, Mar. 1984.
[249] H. Yu, P. Sun, B. Halak, K. Shanthakumar, and T. Kazmierski, "Tamper resistant design of convolutional neural network hardware accelerator," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Dec. 2023, pp. 1–5.
[250] M. I. M. Collantes, Z. Ghodsi, and S. Garg, "SafeTPU: A verifiably secure hardware accelerator for deep neural networks," in *Proc. IEEE 38th VLSI Test Symp. (VTS)*, Apr. 2020, pp. 1–6.
[251] J. Thaler, "Time-optimal interactive proofs for circuit evaluation," in *Proc. Annu. Cryptol. Conf. (CRYPTO)*. Berlin, Germany: Springer, 2013, pp. 71–89.
[252] L. de Castro et al., "Does fully homomorphic encryption need compute acceleration?" 2021, *arXiv:2112.06396*.
[253] F. Turan, S. S. Roy, and I. Verbauwhede, "HEAWS: An accelerator for homomorphic encryption on the Amazon AWS FPGA," *IEEE Trans. Comput.*, vol. 69, no. 8, pp. 1185–1196, Aug. 2020.
[254] S. Halevi and V. Shoup, "Algorithms in HElib," in *Advances in Cryptology—CRYPTO 2014*. Berlin, Germany: Springer, 2014, pp. 554–571.
[255] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, Jan. 2012, pp. 309–325.
[256] S. Halevi, Y. Polyakov, and V. Shoup, "An improved RNS variant of the BFV homomorphic encryption scheme," in *Proc. Cryptographers Track RSA Conf.* Cham, Switzerland: Springer, 2019, pp. 83–105.
[257] J. H. Cheon et al., "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Seoul, South Korea: Seoul National Univ., 2017, pp. 409–437.
[258] I. Chillotti et al., "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, Jan. 2020.
[259] W. Jung et al., "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 114–148, Aug. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9062
[260] H. Chen, I. Chillotti, and J. Kilian, "Improved bootstrapping for approximate homomorphic encryption," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 11477. Cham, Switzerland: Springer, 2019, pp. 34–54.
[261] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Topics in Cryptology CT-RSA*. Cham, Switzerland: Springer, 2020, pp. 364–390.
[262] R. Agrawal et al., "MAD: Memory-aware design techniques for accelerating fully homomorphic encryption," in *Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Toronto, ON, Canada, Oct. 2023, pp. 685–697.
[263] R. Agrawal and A. Joshi, *On Architecting Fully Homomorphic Encryption-Based Computing Systems*. Cham, Switzerland: Springer, 2023.
[264] J.-P. Bossuat et al., "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Advances in Cryptology—EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert, Eds., Cham, Switzerland: Springer, 2021, pp. 587–617.
[265] K. Shivdikar et al., "GME: GPU-based microarchitectural extensions to accelerate homomorphic encryption," in *Proc. 56th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2023, pp. 670–684.
[266] R. Agrawal et al., "FAB: An FPGA-based accelerator for bootstrappable fully homomorphic encryption," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2023, pp. 882–895.
[267] N. Samardzic et al., "F1: A fast and programmable accelerator for fully homomorphic encryption," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchit.*, Jul. 2021, pp. 238–252, doi: 10.1145/3466752.3480070.
[268] S. Kim et al., "BTS: An accelerator for bootstrappable fully homomorphic encryption," 2021, *arXiv:2112.15479*.
[269] N. Samardzic et al., "CraterLake: A hardware accelerator for efficient unbounded computation on encrypted data," in *Proc. 49th Annu. Int. Symp. Comput. Archit.*, New York, NY, USA, Jun. 2022, pp. 173–187.
[270] J. Kim et al., "ARK: Fully homomorphic encryption accelerator with runtime data generation and inter-operation key reuse," in *Proc. IEEE/ACM Int. Symp. Microarchitect.*, Oct. 2022, pp. 1237–1254.
[271] J. Kim et al., "SHARP: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, Jun. 2023, pp. 1–15.
[272] M. S. Riazi et al., "HEAX: An architecture for computing on encrypted data," in *Proc. 25th Int. Conf. Architect. Support Program. Lang. Operat. Syst.*, 2020, pp. 1295–1309.
[273] G. Kokolakis, A. Moschos, and A. D. Keromytis, "Harnessing the power of general-purpose LLMs in hardware trojan design," in *Proc. nternational Conf. Appl. Cryptogr. Netw. Secur.*, Abu Dhabi, United Arab Emirates. Berlin, Germany: Springer, Mar. 2024, pp. 176–194, doi: 10.1007/978-3-031-61486-6_11.
[274] B. Ahmad et al., "On hardware security bug code fixes by prompting large language models," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 4043–4057, 2024.
[275] D. Saha et al., "Empowering hardware security with LLM: The development of a vulnerable hardware database," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2024, pp. 233–243, doi: 10.1109/HOST55342.2024.10545393.